APPENDIX B. SOURCE CODE FOR DESKTOPL2W

Note: The programming environment is Visual Basic Version 6.0.
GlobalVariables Module

Option Explicit
Option Base 0
'******************************************************************************
' * The GlobalVariables module simply dimensions variables that are used through- out the project. *
' * Created February 4, 1999 *
'******************************************************************************

'Define a special variable type for spatial data layers and their properties
Type typLayer
    Name As String  'The name of the layer as defined by the shape filename
    SymbolColor As MapObjects2.ColorConstants  'Color to be used for all the shapes
    FillStyle As MapObjects2.FillStyleConstants  'Allows fill styles for polygons
    OutlineColor As MapObjects2.ColorConstants  'Color to be used for outline of shapes
    Renderer As Boolean  'True if layer is to be classified
    PointStyle As String  'Type of symbol to be applied for point themes
End Type

'Define a special variable type to hold all necessary information from the grid pattern
Type typAttributeHolder
    develop As Long  '1 means developable, and 0 means undevelopable
    CenterX As Double  'Centroid X coordinate
    CenterY As Double  'Centroid Y coordinate
    oldForest As Double  'Current land use that is forest
    oldHerbAg As Double  'Current land use that is herbaceous/ag
    oldWater As Double  'Current land use that is open water
    oldDisturb As Double  'Current land use that is disturbed
    oldMixed As Double  'Current land use that is mixed -- Edge Forest/HerbAg
    LandSeg As Integer  'Land segment to which the grid square belongs
    LogPVHec As Double  'Log of partial land value per hectare
    BlParcVal As Double  'Baseline parcel(s) value
End Type

'Define a special variable type to hold data about each kind of development tract
'for the economics model
Type typEconTractDataHolder
    Low As Variant  'Holds values for low density tracts
    MidConv As Variant  'Holds values for mid density standard tracts
    MidClus As Variant  'Holds values for mid density cluster tracts
    High As Variant  'Holds values for high density tracts
    Comm As Variant  'Holds values for commercial tracts
    Agg As Variant  'Holds values for all types of tracts aggregated
End Type

'Sets the maximum number of layers that can appear on the map
Global Const glbMaxNumLayers As Integer = 20

'Sets the radius (half the length or width) of input development squares in meters
Public glbSquareSize As Integer

'Sets the min and max x and y coordinates for the input grid pattern
Public glbGridMaxX As Double, glbGridMinX As Double
Public glbGridMaxY As Double, glbGridMinY As Double

'Used to define spatial data layers
Public glbLayers(glbMaxNumLayers) As typLayer
Public glbLayersNewData(glbMaxNumLayers) As typLayer

'Stores the number of layers on the map
Public glbLayersCount As Integer
Public glbLayersCountNewData As Integer

'Collection of polygons to hold user-clicked development squares
Public colStoredPolys As New Collection
Public colDevelSquaresL As New Collection
Public colDevelSquaresMC As New Collection
Public colDevelSquaresMCL As New Collection
Public colDevelSquaresH As New Collection
Public colDevelSquaresC As New Collection

'This array is used to check whether a user has already added a particular
data set -- does not allow the user to add it again.
'Used in frmMoreData where the user adds extra data
Public glbAddedPreviously(50) As Boolean

'Define the names of the development square shapefiles
'Low Density Residential
Public glbDevLayerFilenameL As String
'Mid Density Conventional Residential
Public glbDevLayerFilenameMC As String
'Mid Density Cluster Residential
Public glbDevLayerFilenameMCL As String
'High Density Residential
Public glbDevLayerFilenameH As String
'Commercial
Public glbDevLayerFilenameC As String
'Combined
Public glbCombinedDevLayerFilename As String

'The following variables are for storing land use information for the _
land segments or subwatersheds

'Determine the number of land segments
Global Const numLandSegs As Integer = 10 'This value must be a constant for it to work in _
dimensioning arrays. It is equal to the Subwatersheds.records.count

'Find out the original composition of land uses for each land segment, i.e., before
'the user adds any developments
Public glbTotOldForest(numLandSegs) As Double
Public glbTotOldHerbAg(numLandSegs) As Double
Public glbTotOldWater(numLandSegs) As Double
Public glbTotOldDisturb(numLandSegs) As Double
Public glbTotOldMixed(numLandSegs) As Double
Public glbTotAllLandUses(numLandSegs) As Double

Public glbPctOldForest(numLandSegs) As Single
Public glbPctOldHerbAg(numLandSegs) As Single
Public glbPctOldWater(numLandSegs) As Single
Public glbPctOldDisturb(numLandSegs) As Single
Public glbPctOldMixed(numLandSegs) As Single
Public glbPctAllLandUses(numLandSegs) As Single

'the composition of land uses for each land segment after the user has
'added developments
Public glbTotNewForest(numLandSegs) As Double
Public glbTotNewHerbAg(numLandSegs) As Double
Public glbTotNewWater(numLandSegs) As Double
Public glbTotNewDisturb(numLandSegs) As Double
Public glbTotNewMixed(numLandSegs) As Double
Public glbTotAllLandUses(numLandSegs) As Double
Public glbPctNewForest(numLandSegs) As Single
Public glbPctNewHerbAg(numLandSegs) As Single
Public glbPctNewWater(numLandSegs) As Single
Public glbPctNewDisturb(numLandSegs) As Single
Public glbPctNewMixed(numLandSegs) As Single

'The following variables are for storing the amount of each type of land use a particular development type has. For example, low density developments have '10% disturbed area or 9000 square meters out of a 300x300 meter square
'These are for low density res development squares
Public glbForestTractL As Double
Public glbHerbAgTractL As Double
Public glbDisturbTractL As Double
Public glbMixedTractL As Double

'These are for mid density conv res development squares
Public glbForestTractMC As Double
Public glbHerbAgTractMC As Double
Public glbDisturbTractMC As Double
Public glbMixedTractMC As Double

'These are for mid density cluster res development squares
Public glbForestTractMCL As Double
Public glbHerbAgTractMCL As Double
Public glbDisturbTractMCL As Double
Public glbMixedTractMCL As Double

'These are for high density res development squares
Public glbForestTractH As Double
Public glbHerbAgTractH As Double
Public glbDisturbTractH As Double
Public glbMixedTractH As Double

'These are for commercial development squares
Public glbForestTractC As Double
Public glbHerbAgTractC As Double
Public glbDisturbTractC As Double
Public glbMixedTractC As Double

'The following variables count the number of tracts, parcels, and people in a scenario
Public glbTotNumberOfTracts As Integer
Public glbTotNumberOfParcels As Double
Public glbTotNumberOfPeople As Double
Public glbTotNumberOfParcelsL As Double
Public glbTotNumberOfPeopleL As Double
Public glbTotNumberOfParcelsMC As Double
Public glbTotNumberOfPeopleMC As Double
Public glbTotNumberOfParcelsMCL As Double
Public glbTotNumberOfPeopleMCL As Double
Public glbTotNumberOfParcelsH As Double
Public glbTotNumberOfPeopleH As Double
Public glbTotNumberOfParcelsC As Double
Public glbTotNumberOfPeopleC As Double

---

**BasicOperations Module**

Option Explicit
Option Base 0
' ******************************************************
' * The BasicOperations module contains a collection of subroutines/functions *
' Link Legend control to the Map control
Public Sub LinkLegend()
    frmMain.legMapDisplay.setMapSource frmMain.mapDisplay
    frmMain.legMapDisplay.LoadLegend True
    frmMain.legMapDisplay.ShowAllLegend
    frmMain.legMapDisplay.Active(0) = True
End Sub

' Exit the program
Public Sub QuitProgram()
    ' Delete Development Square Files that have been created
    Dim dataconnector As New MapObjects2.DataConnection
    dataconnector.Database = App.Path & "\BC_Data"

    ' Note -- in order to delete a shapefile, you have to FIND it first!
    dataconnector.FindGeoDataset (glbDevLayerFilenameL)
    dataconnector.DeleteGeoDataset (glbDevLayerFilenameL)
    dataconnector.FindGeoDataset (glbDevLayerFilenameMC)
    dataconnector.DeleteGeoDataset (glbDevLayerFilenameMC)
    dataconnector.FindGeoDataset (glbDevLayerFilenameMCL)
    dataconnector.DeleteGeoDataset (glbDevLayerFilenameMCL)
    dataconnector.FindGeoDataset (glbDevLayerFilenameH)
    dataconnector.DeleteGeoDataset (glbDevLayerFilenameH)
    dataconnector.FindGeoDataset (glbDevLayerFilenameC)
    dataconnector.DeleteGeoDataset (glbDevLayerFilenameC)
    dataconnector.Disconnect
End Sub

' Set the mouse pointer and do the action if it does not require a mouse operation
' on the map
Public Sub MapToolbarTask(ButtonKey As String)
    Select Case ButtonKey
        Case "ZoomIn"
            frmMain.mapDisplay.MousePointer = moZoomIn
        Case "ZoomOut"
            frmMain.mapDisplay.MousePointer = moZoomOut
        Case "Pan"
            frmMain.mapDisplay.MousePointer = moPan
        Case "ZoomFullExtent"
            frmMain.mapDisplay.MousePointer = moDefault
            frmMain.mapDisplay.Extent = frmMain.mapDisplay.FullExtent
        Case Else
            frmMain.mapDisplay.MousePointer = moDefault
    End Select

    ' Print button was pushed
    If ButtonKey = "Print" Then
        frmMain.mapDisplay.PrintMap "Back Creek", "", True
        MsgBox "Your map is being printed on your Windows Default Printer."
    End If
End Sub

' AddData button was pushed
If ButtonKey = "AddData" Then frmMoreData.Show

'Clear-All-Developments button was pushed
If ButtonKey = "ClearAll" Then
    Call ClearAllDevelopments 'In the AddDevelopmentsOperations Module
End If

'The Quit button was pushed
If ButtonKey = "Quit" Then
    Call QuitProgram 'In the BasicOperations Module
End If
End Sub

'Zooms in on the map when the user operates the zoom in tool in the map display
Public Sub ZoomIn()
    Dim curRectangle As MapObjects2.Rectangle
    Set curRectangle = frmMain.mapDisplay.TrackRectangle
    Set frmMain.mapDisplay.Extent = curRectangle
End Sub

'Zooms out on the map by a factor of 2 when the user operates the zoom out tool
'Can only occur when not zoomed to full extent
Public Sub ZoomOut()
    Dim curRectangle As MapObjects2.Rectangle
    Dim Loc As New MapObjects2.Point
    Dim X As Double, Y As Double
    Set Loc = frmMain.mapDisplay.ToMapPoint(X, Y)
    'Calculate the full width and height. Adding and substracting
    'the full values from Loc has the effect of zooming out by a factor of 2.
    Dim MapWidth As Double, MapHeight As Double
    Set curRectangle = frmMain.mapDisplay.Extent
    MapWidth = frmMain.mapDisplay.Extent.Width
    MapHeight = frmMain.mapDisplay.Extent.Height
    curRectangle.Right = Loc.X + MapWidth
    curRectangle.Left = Loc.X - MapWidth
    curRectangle.Top = Loc.Y + MapHeight
    curRectangle.Bottom = Loc.Y - MapHeight
    Set frmMain.mapDisplay.Extent = curRectangle
End Sub

'Pans the map when the user operates the pan tool
'Can only occur when not zoomed to full extent
Public Sub Pan()
    frmMain.mapDisplay.Pan
End Sub

' Automatically loads selected pre-prepared spatial data layers into
' map control (frmMain.mapDisplay) from a data directory
Public Sub AutoLoadDataLayer()
    Dim i As Integer
    i = 0

    '**********************************************************************************
    'Define spatial data layer names and other properties and put them in an array
    '**********************************************************************************

    'Note that the following layers will be placed on the map in order --
    'The first layer listed is at the bottom of the map and the last is on top
    'The same goes for the legend
'Define properties for grid pattern
glbLayers(i).Name = "grid_pattern"
glbLayers(i).Renderer = True
i = i + 1

'Define properties for subwatersheds/land segments
glbLayers(i).Name = "subwatersheds"
glbLayers(i).OutlineColor = moRed
glbLayers(i).FillStyle = moTransparentFill
i = i + 1

'Define properties for streams
glbLayers(i).Name = "streams"
glbLayers(i).SymbolColor = moBlue
i = i + 1

glbLayersCount = i

'**********************************************************************************
'Draw the layers defined above on the map by adding layers to the
'MapObjects Layers collection
'**********************************************************************************

Dim dataConnect As New MapObjects2.DataConnection
Dim geoSet As MapObjects2.GeoDataset
Dim newAutoLayer As New MapObjects2.MapLayer
Dim j As Integer
Dim k As Integer
Dim MapRenderer As New MapObjects2.ValueMapRenderer
dataConnect.Database = App.Path & "\BC_Data" 'Tells where to look for data
If dataConnect.Connect = True Then 'If the data connection path is ok
    For j = 0 To glbLayersCount - 1 'Loop through all auto layers
        Set geoSet = dataConnect.FindGeoDataset(glbLayers(j).Name) 'Find shapefile as GeoDataset in DataConnection
        newAutoLayer.GeoDataset = geoSet 'Set GeoDataset property of new MapLayer
        newAutoLayer.Name = glbLayers(j).Name 'Set Name property of new MapLayer
        If newAutoLayer.Symbol.SymbolType = moFillSymbol Then 'Fill style only applies to polygons
            newAutoLayer.Symbol.Style = glbLayers(j).FillStyle 'Set fill of new MapLayer
        End If
        If newAutoLayer.Symbol.SymbolType <> moLineSymbol Then 'Outline color does not apply to lines
        End If
        If glbLayers(j).Renderer = True Then 'If the layer is the grid pattern
            Set newAutoLayer.Renderer = MapRenderer
            With newAutoLayer.Renderer
                .ValueCount = 2
                .Field = "Develop"
                .Value(0) = "No Develop"
                .Symbol(0).OutlineColor = moLimeGreen
                .Symbol(0).Style = moLightGrayFill
                .Symbol(0).Color = moCyan
                .Value(1) = "Yes Develop"
                .Symbol(1).OutlineColor = moLimeGreen
                .Symbol(1).Style = moTransparentFill
            End With
End If
frmMain.mapDisplay.Layers.Add newAutoLayer 'Add MapLayer to Layers collection
Set newAutoLayer = Nothing 'Reset newAutoLayer to add the next file
Next j
Else 'geoSet Is Nothing
MsgBox "Error connecting to data", vbCritical
Exit Sub
End If

'Zoom the Map Display to the full extent
frmMain.mapDisplay.Extent = frmMain.mapDisplay.FullExtent
dataConnect.Disconnect

End Sub

AddDevelopmentsOperations Module

Option Explicit
Option Base 0
' ******************************************************************************
' * The AddDevelopments Operations module contains a collection of subroutines *
' * that allow the user to enter various development types in the watershed    *
' * Created February 7, 1999                                                *
' ******************************************************************************

'This subroutine adds user-placed, low-density development squares to a database
'It makes sure that the user doesn't try to add the same square more than once.
The input UserPt is the point that the user clicked on the map

Public Sub AddDevelopmentSquareL(UserPtL As MapObjects2.Point)

'Points of a square, which is the shape of development blocks
'point1 - point4 are vertices of development square
Dim point1L As New MapObjects2.Point, point2L As New MapObjects2.Point
Dim point3L As New MapObjects2.Point, point4L As New MapObjects2.Point
Dim colPointsL As New MapObjects2.Points
Dim polyL As New MapObjects2.Polygon

'These variables (along with colStoredPolys) are designed
to make sure the user doesn't keep clicking the same square
Dim StorePolyL As New MapObjects2.Polygon
Dim CountDuplicateL As Integer
CountDuplicateL = 0

'Get 4 points of a square based on userpt and add the points to a collection
'upper left (northwest) vertex
point1L.X = UserPtL.X - glbSquareSize
point1L.Y = UserPtL.Y + glbSquareSize
colPointsL.Add point1L

'upper right (northeast) vertex
point2L.X = UserPtL.X + glbSquareSize
point2L.Y = UserPtL.Y + glbSquareSize
colPointsL.Add point2L

'lower right (southeast) vertex
point3L.X = UserPtL.X + glbSquareSize
point3L.Y = UserPtL.Y - glbSquareSize

98
colPointsL.Add point3L

' lower left (southwest) vertex
point4L.X = UserPtL.X - glbSquareSize
point4L.Y = UserPtL.Y - glbSquareSize
colPointsL.Add point4L

' Create a polygon (square) based on the 4 derived points
Set polyL = CreateObject("MapObjects2.Polygon")
Set StorePolyL = CreateObject("MapObjects2.Polygon")
polyL.Parts.Add colPointsL

' Prevent the user from making duplicate squares
If colStoredPolys.Count <> 0 Then 'If the user has already made some squares
    For Each StorePolyL In colStoredPolys
        'If the centroid of current clicked square matches any stored squares
        'then increment a duplicate counter
    Next StorePolyL
    If CountDuplicateL = 0 Then
        colDevelSquaresL.Add polyL 'add the square to the collection of input
development squares
        frmMain.mapDisplay.TrackingLayer.Refresh True 'This sends the program to the
        subroutine _
        mapDisplay_afterTrackingLayerDraw to draw the clicked square on the map.
    End If
Else 'The user has not yet clicked any squares on the map
    colDevelSquaresL.Add polyL 'add the square to the collection of input
development squares
    frmMain.mapDisplay.TrackingLayer.Refresh True 'This sends the program to the
    subroutine _
    mapDisplay_afterTrackingLayerDraw to draw the clicked square on the map.
End If

' Add all clicked squares to the collection of stored polygons
StorePolyL.Parts.Add colPointsL
colStoredPolys.Add polyL

End Sub

' This subroutine adds user-placed, mid-density-conv development squares to a database
' It makes sure that the user doesn't try to add the same square more than once.
' The input UserPt is the point that the user clicked on the map
Public Sub AddDevelopmentSquareMC(UserPtMC As MapObjects2.Point)

' Points of a square, which is the shape of development blocks
' point1 - point4 are vertices of development square
Dim point1MC As New MapObjects2.Point, point2MC As New MapObjects2.Point
Dim point3MC As New MapObjects2.Point, point4MC As New MapObjects2.Point
Dim colPointsMC As New MapObjects2.Points
Dim polyMC As New MapObjects2.Polygon

' These variables (along with colStoredPolys) are designed
'to make sure the user doesn't keep clicking the same square
Dim StorePolyMC As New MapObjects2.Polygon
Dim CountDuplicateMC As Integer
CountDuplicateMC = 0

'Get 4 points of a square based on userpt and add the points to a collection
'upper left (northwest) vertex
point1MC.X = UserPtmc.X - glbSquareSize
point1MC.Y = UserPtmc.Y + glbSquareSize
colPointsMC.Add point1MC

'upper right (northeast) vertex
point2MC.X = UserPtmc.X + glbSquareSize
point2MC.Y = UserPtmc.Y + glbSquareSize
colPointsMC.Add point2MC

'lower right (southeast) vertex
point3MC.X = UserPtmc.X + glbSquareSize
point3MC.Y = UserPtmc.Y - glbSquareSize
colPointsMC.Add point3MC

'lower left (southwest) vertex
point4MC.X = UserPtmc.X - glbSquareSize
point4MC.Y = UserPtmc.Y - glbSquareSize
colPointsMC.Add point4MC

' Create a polygon (square) based on the 4 derived points
Set polyMC = CreateObject("MapObjects2.Polygon")
Set StorePolyMC = CreateObject("MapObjects2.Polygon")
polyMC.Parts.Add colPointsMC

'Prevent the user from making duplicate squares
If colStoredPolys.Count <> 0 Then 'If the user has already made some squares
  For Each StorePolyMC In colStoredPolys
    'If the centroid of current clicked square matches any stored squares
    'then increment a duplicate counter
    If polyMC.Centroid.X = StorePolyMC.Centroid.X And polyMC.Centroid.Y = StorePolyMC.Centroid.Y Then CountDuplicateMC = CountDuplicateMC + 1
  Next StorePolyMC
  If CountDuplicateMC = 0 Then
    colDevelSquaresMC.Add polyMC 'add the square to the collection of input
development squares
    frmMain.mapDisplay.TrackingLayer.Refresh True 'This sends the program to the
  End If
Else 'The user has not yet clicked any squares on the map
  colDevelSquaresMC.Add polyMC 'add the square to the collection of input
development squares
  frmMain.mapDisplay.TrackingLayer.Refresh True 'This sends the program to the
End If

'Add all clicked squares to the collection of stored polygons
StorePolyMC.Parts.Add colPointsMC
colStoredPolys.Add polyMC

End Sub

'This subroutine adds user-placed, mid-density-cluster development squares to a database
'It makes sure that the user doesn't try to add the same square more than once. 'The input UserPt is the point that the user clicked on the map

Public Sub AddDevelopmentSquareMCL(UserPtMCL As MapObjects2.Point)

    'Points of a square, which is the shape of development blocks
    'point1 - point4 are vertices of development square
    Dim point1MCL As New MapObjects2.Point, point2MCL As New MapObjects2.Point
    Dim point3MCL As New MapObjects2.Point, point4MCL As New MapObjects2.Point
    Dim colPointsMCL As New MapObjects2.Points
    Dim polyMCL As New MapObjects2.Polygon

    'These variables (along with colStoredPolys) are designed
    'to make sure the user doesn't keep clicking the same square
    Dim StorePolyMCL As New MapObjects2.Polygon
    Dim CountDuplicateMCL As Integer
    CountDuplicateMCL = 0

    'Get 4 points of a square based on userpt and add the points to a collection
    'upper left (northwest) vertex
    point1MCL.X = UserPtMCL.X - glbSquareSize
    point1MCL.Y = UserPtMCL.Y + glbSquareSize
    colPointsMCL.Add point1MCL

    'upper right (northeast) vertex
    point2MCL.X = UserPtMCL.X + glbSquareSize
    point2MCL.Y = UserPtMCL.Y + glbSquareSize
    colPointsMCL.Add point2MCL

    'lower right (southeast) vertex
    point3MCL.X = UserPtMCL.X + glbSquareSize
    point3MCL.Y = UserPtMCL.Y - glbSquareSize
    colPointsMCL.Add point3MCL

    'lower left (southwest) vertex
    point4MCL.X = UserPtMCL.X - glbSquareSize
    point4MCL.Y = UserPtMCL.Y - glbSquareSize
    colPointsMCL.Add point4MCL

    ' Create a polygon (square) based on the 4 derived points
    Set polyMCL = CreateObject("MapObjects2.Polygon")
    Set StorePolyMCL = CreateObject("MapObjects2.Polygon")
    polyMCL.Parts.Add colPointsMCL

    'Prevent the user from making duplicate squares
    If colStoredPolys.Count <> 0 Then 'If the user has already made some squares
        For Each StorePolyMCL In colStoredPolys
            'If the centroid of current clicked square matches any stored squares
            'then increment a duplicate counter
            If polyMCL.Centroid.X = StorePolyMCL.Centroid.X And polyMCL.Centroid.Y = StorePolyMCL.Centroid.Y Then CountDuplicateMCL = CountDuplicateMCL + 1
        Next StorePolyMCL
    End If

    If CountDuplicateMCL = 0 Then
        colDevelSquaresMCL.Add polyMCL 'add the square to the collection of input development squares
        frmMain.mapDisplay.TrackingLayer.Refresh True 'This sends the program to the subroutine
        _
        mapDisplay_afterTrackingLayerDraw to draw the clicked square on the map.
    End If

End Sub
Else 'The user has not yet clicked any squares on the map
    colDevelSquaresMCL.Add polyMCL 'add the square to the collection of input
development squares
    frmMain.mapDisplay.TrackingLayer.Refresh True 'This sends the program to the
    subroutine mapDisplay_afterTrackingLayerDraw to draw the clicked square on the map.
End If

'Add all clicked squares to the collection of stored polygons
StorePolyMCL.Parts.Add colPointsMCL
colStoredPolys.Add polyMCL

End Sub

'This subroutine adds user-placed, high-density development squares to a database
'It makes sure that the user doesn't try to add the same square more than once.
'The input UserPt is the point that the user clicked on the map
Public Sub AddDevelopmentSquareH(UserPtH As MapObjects2.Point)

    'Points of a square, which is the shape of development blocks
    'point1 - point4 are vertices of development square
    Dim point1H As New MapObjects2.Point, point2H As New MapObjects2.Point
    Dim point3H As New MapObjects2.Point, point4H As New MapObjects2.Point
    Dim colPointsH As New MapObjects2.Points
    Dim polyH As New MapObjects2.Polygon

    'These variables (along with colStoredPolys) are designed
    'to make sure the user doesn't keep clicking the same square
    Dim StorePolyH As New MapObjects2.Polygon
    Dim CountDuplicateH As Integer
    CountDuplicateH = 0

    'Get 4 points of a square based on userpt and add the points to a collection
    'upper left (northwest) vertex
    point1H.X = UserPtH.X - glbSquareSize
    point1H.Y = UserPtH.Y + glbSquareSize
    colPointsH.Add point1H

    'upper right (northeast) vertex
    point2H.X = UserPtH.X + glbSquareSize
    point2H.Y = UserPtH.Y + glbSquareSize
    colPointsH.Add point2H

    'lower right (southeast) vertex
    point3H.X = UserPtH.X + glbSquareSize
    point3H.Y = UserPtH.Y - glbSquareSize
    colPointsH.Add point3H

    'lower left (southwest) vertex
    point4H.X = UserPtH.X - glbSquareSize
    point4H.Y = UserPtH.Y - glbSquareSize
    colPointsH.Add point4H

    'Create a polygon (square) based on the 4 derived points
    Set polyH = CreateObject("MapObjects2.Polygon")
    Set StorePolyH = CreateObject("MapObjects2.Polygon")
    polyH.Parts.Add colPointsH

    'Prevent the user from making duplicate squares
    If colStoredPolys.Count <> 0 Then 'If the user has already made some squares
        For Each StorePolyH In colStoredPolys
            'If the centroid of current clicked square matches any stored squares
            If StorePolyH.Centroid.X = StorePolyH.Centroid.Y =
            End If
        Next StorePolyH
    End If
End Sub
'then increment a duplicate counter
Next StorePolyH

'If there were no duplicates then accept the current square and draw it
If CountDuplicateH = 0 Then
  colDevelSquaresH.Add polyH 'add the square to the collection of input development squares
  frmMain.mapDisplay.TrackingLayer.Refresh True 'This sends the program to the subroutine _
  mapDisplay_afterTrackingLayerDraw to draw the clicked square on the map.
End If

Else 'The user has not yet clicked any squares on the map
  colDevelSquaresH.Add polyH 'add the square to the collection of input development squares
  frmMain.mapDisplay.TrackingLayer.Refresh True 'This sends the program to the subroutine _
  mapDisplay_afterTrackingLayerDraw to draw the clicked square on the map.
End If

'Add all clicked squares to the collection of stored polygons
StorePolyH.Parts.Add colPointsH
colStoredPolys.Add polyH

End Sub

'This subroutine adds user-placed, commercial development squares to a database
'It makes sure that the user doesn't try to add the same square more than once.
The input UserPt is the point that the user clicked on the map
Public Sub AddDevelopmentSquareC(UserPtC As MapObjects2.Point)

'Points of a square, which is the shape of development blocks
'point1 - point4 are vertices of development square
Dim point1C As New MapObjects2.Point, point2C As New MapObjects2.Point
Dim point3C As New MapObjects2.Point, point4C As New MapObjects2.Point
Dim colPointsC As New MapObjects2.Points
Dim polyC As New MapObjects2.Polygon

'These variables (along with colStoredPolys) are designed
to make sure the user doesn't keep clicking the same square
Dim StorePolyC As New MapObjects2.Polygon
Dim CountDuplicateC As Integer
CountDuplicateC = 0

'Get 4 points of a square based on userPt and add the points to a collection
'upper left (northwest) vertex
point1C.X = UserPtC.X - glbSquareSize
point1C.Y = UserPtC.Y + glbSquareSize
colPointsC.Add point1C

'upper right (northeast) vertex
point2C.X = UserPtC.X + glbSquareSize
point2C.Y = UserPtC.Y + glbSquareSize
colPointsC.Add point2C

'lower right (southeast) vertex
point3C.X = UserPtC.X + glbSquareSize
point3C.Y = UserPtC.Y - glbSquareSize
colPointsC.Add point3C
'lower left (southwest) vertex
point4C.X = UserPtC.X - glbSquareSize
point4C.Y = UserPtC.Y - glbSquareSize
colPointsC.Add point4C

' Create a polygon (square) based on the 4 derived points
Set polyC = CreateObject("MapObjects2.Polygon")
Set StorePolyC = CreateObject("MapObjects2.Polygon")
polyC.Parts.Add colPointsC

'Prevent the user from making duplicate squares
If colStoredPolys.Count <> 0 Then 'If the user has already made some squares
    For Each StorePolyC In colStoredPolys
        'If the centroid of current clicked square matches any stored squares
        'then increment a duplicate counter
        If polyC.Centroid.X = StorePolyC.Centroid.X And polyC.Centroid.Y = StorePolyC.Centroid.Y Then CountDuplicateC = CountDuplicateC + 1
    Next StorePolyC

    'If there were no duplicates then accept the current square and draw it
    If CountDuplicateC = 0 Then
        colDevelSquaresC.Add polyC 'add the square to the collection of input
development squares
        frmMain.mapDisplay.TrackingLayer.Refresh True 'This sends the program to the
        subroutine _
        mapDisplay_afterTrackingLayerDraw to draw the clicked square on the map.
        End If
    Else 'The user has not yet clicked any squares on the map
        colDevelSquaresC.Add polyC 'add the square to the collection of input
development squares
        frmMain.mapDisplay.TrackingLayer.Refresh True 'This sends the program to the
        subroutine _
        mapDisplay_afterTrackingLayerDraw to draw the clicked square on the map.
        End If

    'Add all clicked squares to the collection of stored polygons
    StorePolyC.Parts.Add colPointsC
    colStoredPolys.Add polyC

End Sub

'This subroutine saves user-clicked squares as a shapefile complete with attribute
data
Public Sub SaveSquaresAsDatabase()
    'Establish a connection to the folder that holds shapefiles
    Dim dc As New MapObjects2.DataConnection
    dc.Database = App.Path & "\BC_Data"
    If (Not dc.Connect) Then
        MsgBox "Could not connect to database!"
        Exit Sub 'bad dataConnection
    End If

    'Set the names of the shapefiles and set up data fields
    'Low Density Residential
glbDevLayerFilenameL = "aaLowPolys"
    'Mid Density Conventional Residential
glbDevLayerFilenameMC = "aaMidConvPolys"
    'Mid Density Cluster Residential
glbDevLayerFilenameMCL = "aaMidClusterPolys"
'High Density Residential
glbDevLayerFilenameH = "aaHighPolys"
'Commercial
  glbDevLayerFilenameC = "aaCommercialPolys"

Dim tDesc As New TableDesc

'Define additional fields
'First set the number of fields (besides the shape field)
tDesc.FieldCount = 22

'Set the field names
tDesc.FieldName(0) = "Name"
tDesc.FieldName(1) = "Area"
tDesc.FieldName(2) = "SewDist"
tDesc.FieldName(3) = "Develop"
tDesc.FieldName(4) = "DevType"
tDesc.FieldName(5) = "oldForest"
tDesc.FieldName(6) = "oldHerbAg"
tDesc.FieldName(7) = "oldWater"
tDesc.FieldName(8) = "oldDisturb"
tDesc.FieldName(9) = "oldMixed"
tDesc.FieldName(10) = "newForest"
tDesc.FieldName(11) = "newHerbAg"
tDesc.FieldName(12) = "newWater"
tDesc.FieldName(13) = "newDisturb"
tDesc.FieldName(14) = "newMixed"
tDesc.FieldName(15) = "LandSeg"
tDesc.FieldName(16) = "LogPVHec"
tDesc.FieldName(17) = "BlParcVal"
tDesc.FieldName(18) = "WatDist"
tDesc.FieldName(19) = "SuperGrp"
tDesc.FieldName(20) = "CenterX"
tDesc.FieldName(21) = "CenterY"

'Set the type of field
tDesc.FieldType(0) = moString
tDesc.FieldType(1) = moDouble
tDesc.FieldType(2) = moDouble
tDesc.FieldType(3) = moDouble
tDesc.FieldType(4) = moString
tDesc.FieldType(5) = moDouble
tDesc.FieldType(6) = moDouble
tDesc.FieldType(7) = moDouble
tDesc.FieldType(8) = moDouble
tDesc.FieldType(9) = moDouble
tDesc.FieldType(10) = moDouble
tDesc.FieldType(11) = moDouble
tDesc.FieldType(12) = moDouble
tDesc.FieldType(13) = moDouble
tDesc.FieldType(14) = moDouble
tDesc.FieldType(15) = moLong
tDesc.FieldType(16) = moDouble
tDesc.FieldType(17) = moDouble
tDesc.FieldType(18) = moDouble
tDesc.FieldType(19) = moLong
tDesc.FieldType(20) = moDouble
tDesc.FieldType(21) = moDouble

'Set the length of the string fields
tDesc.FieldLength(0) = 20
tDesc.FieldLength(4) = 10
'Set the number of digits used in the numeric fields
	tDesc.FieldPrecision(1) = 15
	tDesc.FieldPrecision(2) = 15
	tDesc.FieldPrecision(3) = 1
	tDesc.FieldPrecision(4) = 10
	tDesc.FieldPrecision(5) = 10
	tDesc.FieldPrecision(6) = 10
	tDesc.FieldPrecision(7) = 10
	tDesc.FieldPrecision(8) = 10
	tDesc.FieldPrecision(9) = 10
	tDesc.FieldPrecision(10) = 10
	tDesc.FieldPrecision(11) = 10
	tDesc.FieldPrecision(12) = 10
	tDesc.FieldPrecision(13) = 10
	tDesc.FieldPrecision(14) = 10
	tDesc.FieldPrecision(15) = 2
	tDesc.FieldPrecision(16) = 10
	tDesc.FieldPrecision(17) = 10
	tDesc.FieldPrecision(18) = 15
	tDesc.FieldPrecision(19) = 5
	tDesc.FieldPrecision(20) = 13
	tDesc.FieldPrecision(21) = 13

'Set the number of digits to the right of the decimal point
	tDesc.FieldScale(1) = 3
	tDesc.FieldScale(2) = 3
	tDesc.FieldScale(3) = 0
	tDesc.FieldScale(5) = 3
	tDesc.FieldScale(6) = 3
	tDesc.FieldScale(7) = 3
	tDesc.FieldScale(8) = 3
	tDesc.FieldScale(9) = 3
	tDesc.FieldScale(10) = 3
	tDesc.FieldScale(11) = 3
	tDesc.FieldScale(12) = 3
	tDesc.FieldScale(13) = 3
	tDesc.FieldScale(14) = 3
	tDesc.FieldScale(15) = 0
	tDesc.FieldScale(16) = 4
	tDesc.FieldScale(17) = 0
	tDesc.FieldScale(18) = 3
	tDesc.FieldScale(19) = 0
	tDesc.FieldScale(20) = 6
	tDesc.FieldScale(21) = 6

'This defines the geodataset, layer, and records for the grid pattern.
'The newly created layer of development squares needs to extract data
'from the grid pattern.
	Dim gridGd As MapObjects2.GeoDataset
	Dim gridLyr As New MapObjects2.MapLayer
	Dim gridRecs As New MapObjects2.Recordset
	Set gridLyr.GeoDataset = dc.FindGeoDataset("grid_pattern")
	Set gridRecs = gridLyr.Records

Dim Counter As Long
Dim GridAttribute As typAttributeHolder
Dim finished1 As Boolean
Dim i As Integer

'Set the values of land use amounts for each development type
'Percentages based on Schuler (1998) and Arendt (1994)
glbForestTractL = ((glbSquareSize * 2) ^ 2) * 0.05
glbHerbAgTractL = ((glbSquareSize * 2) ^ 2) * 0.15
glbDisturbTractL = ((glbSquareSize * 2) ^ 2) * 0.1
glbMixedTractL = ((glbSquareSize * 2) ^ 2) * 0.7
'These are for mid density conv res development squares
'Percentages based on Schuler (1998) and Arendt (1994)

glbForestTractMC = ((glbSquareSize * 2) ^ 2) * 0
glbHerbAgTractMC = ((glbSquareSize * 2) ^ 2) * 0.5
glbDisturbTractMC = ((glbSquareSize * 2) ^ 2) * 0.27
glbMixedTractMC = ((glbSquareSize * 2) ^ 2) * 0.23
'These are for mid density cluster res development squares
'Percentages based on Schuler (1998) and Arendt (1994)

glbForestTractH = ((glbSquareSize * 2) ^ 2) * 0.05
glbHerbAgTractH = ((glbSquareSize * 2) ^ 2) * 0.6
glbDisturbTractH = ((glbSquareSize * 2) ^ 2) * 0.35
glbMixedTractH = ((glbSquareSize * 2) ^ 2) * 0
'These are for high density res development squares
'Percentages from GIS calculations on University City commercial area

Dim gdL As MapObjects2.GeoDataset
Set gdL = dc.AddGeoDataset(glbDevLayerFilenameL, moPolygon, tDesc)
If gdL Is Nothing Then Exit Sub 'invalid file
Dim lyrL As New MapObjects2.MapLayer
Set lyrL.GeoDataset = gdL
If colDevelSquaresL.Count <> 0 Then
    For Counter = 1 To colDevelSquaresL.Count
        finished1 = False
        With lyrL.Records
            .AddNew 'Adds a new record to the development squares database
            'The following block of code (the do loop and the .fields().value assignments)
            'is used to grab attribute data from the grid pattern
d gridRecs.MoveFirst 'Go to the first record in the grid pattern database
        Do While Not gridRecs.EOF And finished1 = False 'Loop thru the records
            'If the centroid of a square in the grid pattern matches the centroid of
            'the development square, then store the data from the grid square
            If Fix(colDevelSquaresL(Counter).Centroid.X) = Fix(gridRecs.Fields("centerx").Value) _
                And Fix(colDevelSquaresL(Counter).Centroid.Y) = Fix(gridRecs.Fields("centery").Value) Then
                GridAttribute.develop = gridRecs.Fields("devel").Value
                GridAttribute.CenterX = gridRecs.Fields("CenterX").Value
                GridAttribute.CenterY = gridRecs.Fields("CenterY").Value
                GridAttribute.oldForest = gridRecs.Fields("oldForest").Value
                GridAttribute.oldHerbAg = gridRecs.Fields("oldHerbAg").Value
                GridAttribute.oldWater = gridRecs.Fields("oldWater").Value
                GridAttribute.oldDisturb = gridRecs.Fields("oldDisturb").Value
                GridAttribute.oldMixed = gridRecs.Fields("oldMixed").Value
            End If
        Loop
    Next Counter
End If

107
GridAttribute.LandSeg = gridRecs.Fields("landseg").Value
GridAttribute.LogPVHec = gridRecs.Fields("LogPVHec").Value

finished1 = True
End If
gridRecs.MoveNext
Loop

' Set the values of the development square’s fields
.Fields("Shape").Value = colDevelSquaresL(Counter)
.Fields("Name").Value = "Low Density " & Str(Counter)
.Fields("Area").Value = colDevelSquaresL(Counter).Area
.Fields("develop").Value = GridAttribute.develop
.Fields("CenterX").Value = GridAttribute.CenterX
.Fields("CenterY").Value = GridAttribute.CenterY
.Fields("oldForest").Value = GridAttribute.oldForest
.Fields("oldHerbAg").Value = GridAttribute.oldHerbAg
.Fields("oldWater").Value = GridAttribute.oldWater
.Fields("oldDisturb").Value = GridAttribute.oldDisturb
.Fields("oldMixed").Value = GridAttribute.oldMixed
.Fields("LandSeg").Value = GridAttribute.LandSeg
.Fields("DevType").Value = "LDR"
.Fields("LogPVHec").Value = GridAttribute.LogPVHec
.Fields("newForest").Value = glbForestTractL
.Fields("newHerbAg").Value = glbHerbAgTractL
.Fields("newWater").Value = 0 'If developed, assumed to have no water
.Fields("newDisturb").Value = glbDisturbTractL
.Fields("newMixed").Value = glbMixedTractL
.Update
End With

Next Counter
lyrL.Records.StopEditing
End If

' This block adds fields to the database of mid density conv res development squares
' and populates those fields
' This sets up the geodataset and layer to save mid density conventional residential polygons
Dim gdMC As MapObjects2.GeoDataset
Set gdMC = dc.AddGeoDataset(glbDevLayerFilenameMC, moPolygon, tDesc)
If gdMC Is Nothing Then Exit Sub 'invalid file
Dim lyrMC As New MapObjects2.MapLayer
Set lyrMC.GeoDataset = gdMC

If colDevelSquaresMC.Count <> 0 Then
For Counter = 1 To colDevelSquaresMC.Count
    finished1 = False
    With lyrMC.Records
        .AddNew 'Adds a new record to the development squares database

' The following block of code (the do loop and the .fields().value assignments)
' is used to grab attribute data from the grid pattern
gridRecs.MoveFirst 'Go to the first record in the grid pattern database
Do While Not gridRecs.EOF And finished1 = False 'Loop thru the records
    'If the centroid of a square in the grid pattern matches the centroid of
    'the development square, then store the data from the grid square
    If Fix(colDevelSquaresMC(Counter).Centroid.X) = Fix(gridRecs.Fields("centerx").Value) _
And Fix(colDevelSquaresMC(Counter).Centroid.Y) = Fix(gridRecs.Fields("centery").Value) Then
  GridAttribute.develop = gridRecs.Fields("devel").Value
  GridAttribute.CenterX = gridRecs.Fields("CenterX").Value
  GridAttribute.CenterY = gridRecs.Fields("CenterY").Value
  GridAttribute.oldForest = gridRecs.Fields("oldForest").Value
  GridAttribute.oldHerbAg = gridRecs.Fields("oldHerbAg").Value
  GridAttribute.oldWater = gridRecs.Fields("oldWater").Value
  GridAttribute.oldDisturb = gridRecs.Fields("oldDisturb").Value
  GridAttribute.oldMixed = gridRecs.Fields("oldMixed").Value
  GridAttribute.LandSeg = gridRecs.Fields("landseg").Value
  GridAttribute.LogPVHec = gridRecs.Fields("LogPVHec").Value
  GridAttribute.BLParcVal = gridRecs.Fields("BLParcVal").Value
  finished1 = True
End If
gridRecs.MoveNext
Loop

' Set the values of the development square's fields
.Fields("Shape").Value = colDevelSquaresMC(Counter)
.Fields("Name").Value = "Mid Standard " & Str(Counter)
.Fields("Area").Value = colDevelSquaresMC(Counter).Area
.Fields("develop").Value = GridAttribute.develop
.Fields("CenterX").Value = GridAttribute.CenterX
.Fields("CenterY").Value = GridAttribute.CenterY
.Fields("oldForest").Value = GridAttribute.oldForest
.Fields("oldHerbAg").Value = GridAttribute.oldHerbAg
.Fields("oldWater").Value = GridAttribute.oldWater
.Fields("oldDisturb").Value = GridAttribute.oldDisturb
.Fields("oldMixed").Value = GridAttribute.oldMixed
.Fields("LandSeg").Value = GridAttribute.LandSeg
.Fields("DevType").Value = "MDRConv"
.Fields("LogPVHec").Value = GridAttribute.LogPVHec
.Fields("BLParcVal").Value = GridAttribute.BLParcVal
.Fields("newForest").Value = glbForestTractMC
.Fields("newHerbAg").Value = glbHerbAgTractMC
.Fields("newWater").Value = 0 'If developed, assumed to have no water
.Fields("newDisturb").Value = glbDisturbTractMC
.Fields("newMixed").Value = glbMixedTractMC
.Update
End With

Next Counter
lyrMC.Records.StopEditing
End If

'This block adds fields to the database of mid density cluster res development squares and populates those fields
Dim gdMCL As MapObjects2.GeoDataset
Set gdMCL = dc.AddGeoDataset(glbDevLayerFilenameMCL, moPolygon, tDesc)
If gdMCL Is Nothing Then Exit Sub 'invalid file
Dim lyrMCL As New MapObjects2.MapLayer
Set lyrMCL.GeoDataset = gdMCL
If colDevelSquaresMCL.Count <> 0 Then
  For Counter = 1 To colDevelSquaresMCL.Count
    finished1 = False
    With lyrMCL.Records
      .AddNew 'Adds a new record to the development squares database
        'The following block of code (the do loop and the .fields().value assignments)
'is used to grab attribute data from the grid pattern
gridRecs.MoveNext 'Go to the first record in the grid pattern database
Do While Not gridRecs.EOF And finished1 = False 'Loop thru the records
  If the centroid of a square in the grid pattern matches the centroid of
  the development square, then store the data from the grid square
  Fix(gridRecs.Fields("centerx").Value) =
  Fix(colDevelSquaresMCL(Counter).Centroid.X) =
  And Fix(colDevelSquaresMCL(Counter).Centroid.Y) =
  Fix(gridRecs.Fields("centery").Value) Then
    GridAttribute.develop = gridRecs.Fields("devel").Value
    GridAttribute.CenterX = gridRecs.Fields("centerX").Value
    GridAttribute.CenterY = gridRecs.Fields("centerY").Value
    GridAttribute.oldForest = gridRecs.Fields("oldForest").Value
    GridAttribute.oldHerbAg = gridRecs.Fields("oldHerbAg").Value
    GridAttribute.oldWater = gridRecs.Fields("oldWater").Value
    GridAttribute.oldDisturb = gridRecs.Fields("oldDisturb").Value
    GridAttribute.oldMixed = gridRecs.Fields("oldMixed").Value
    GridAttribute.LandSeg = gridRecs.Fields("landseg").Value
    GridAttribute.LogPVHec = gridRecs.Fields("LogPVHec").Value
    finished1 = True
  End If
  gridRecs.MoveNext
Loop

'Set the values of the development square's fields
.Fields("Shape").Value = colDevelSquaresMCL(Counter)
.Fields("Name").Value = "Mid Cluster " & Str(Counter)
.Fields("Area").Value = colDevelSquaresMCL(Counter).Area
.Fields("develop").Value = GridAttribute.develop
.Fields("CenterX").Value = GridAttribute.CenterX
.Fields("CenterY").Value = GridAttribute.CenterY
.Fields("oldForest").Value = GridAttribute.oldForest
.Fields("oldHerbAg").Value = GridAttribute.oldHerbAg
.Fields("oldWater").Value = GridAttribute.oldWater
.Fields("oldDisturb").Value = GridAttribute.oldDisturb
.Fields("oldMixed").Value = GridAttribute.oldMixed
.Fields("LandSeg").Value = GridAttribute.LandSeg
.Fields("DevType").Value = "MDRClus"
.Fields("LogPVHec").Value = GridAttribute.LogPVHec
.Fields("newForest").Value = glbForestTractMCL
.Fields("newHerbAg").Value = glbHerbAgTractMCL
.Fields("newWater").Value = 0 'If developed, assumed to have no water
.Fields("newDisturb").Value = glbDisturbTractMCL
.Fields("newMixed").Value = glbMixedTractMCL
.Update
End With

Next Counter
lyrMCL.Records.StopEditing
End If

'This block adds fields to the database of high density res development squares and
populates those fields
Dim gdH As MapObjects2.GeoDataset
Set gdH = dc.AddGeoDataset(glbDevLayerFilenameH, moPolygon, tDesc)
If gdH Is Nothing Then Exit Sub 'invalid file
Dim lyrH As New MapObjects2.MapLayer
Set lyrH.GeoDataset = gdH

If colDevelSquaresH.Count <> 0 Then
  For Counter = 1 To colDevelSquaresH.Count
finished1 = False

With lyrH.Records
  .AddNew 'Adds a new record to the development squares database

  'The following block of code (the do loop and the .fields().value assignments)
  'is used to grab attribute data from the grid pattern
  gridRecs.MoveNext 'Go to the first record in the grid pattern database
  Do While Not gridRecs.EOF And finished1 = False 'Loop thru the records
    'If the centroid of a square in the grid pattern matches the centroid of
    'the development square, then store the data from the grid square
      GridAttribute.develop = gridRecs.Fields("devel") .Value
      GridAttribute.CenterX = gridRecs.Fields("CenterX") .Value
      GridAttribute.CenterY = gridRecs.Fields("CenterY") .Value
      GridAttribute.oldForest = gridRecs.Fields("oldForest") .Value
      GridAttribute.oldHerbAg = gridRecs.Fields("oldHerbAg") .Value
      GridAttribute.oldWater = gridRecs.Fields("oldWater") .Value
      GridAttribute.oldDisturb = gridRecs.Fields("oldDisturb") .Value
      GridAttribute.oldMixed = gridRecs.Fields("oldMixed") .Value
      GridAttribute.LandSeg = gridRecs.Fields("landseg") .Value
      GridAttribute.LogPVHec = gridRecs.Fields("LogPVHec") .Value
      finished1 = True
    End If
  gridRecs.MoveNext
  Loop

  'Set the values of the development square’s fields
  .Fields("Shape") .Value = colDevelSquaresH(Counter)
  .Fields("Name") .Value = "High Density " & Str(Counter)
  .Fields("Area") .Value = colDevelSquaresH(Counter) .Area
  .Fields("develop") .Value = GridAttribute.develop
  .Fields("CenterX") .Value = GridAttribute.CenterX
  .Fields("CenterY") .Value = GridAttribute.CenterY
  .Fields("oldForest") .Value = GridAttribute.oldForest
  .Fields("oldHerbAg") .Value = GridAttribute.oldHerbAg
  .Fields("oldWater") .Value = GridAttribute.oldWater
  .Fields("oldDisturb") .Value = GridAttribute.oldDisturb
  .Fields("oldMixed") .Value = GridAttribute.oldMixed
  .Fields("LandSeg") .Value = GridAttribute.LandSeg
  .Fields("DevType") .Value = "HDR"
  .Fields("LogPVHec") .Value = GridAttribute.LogPVHec
  .Fields("newForest") .Value = glbForestTractH
  .Fields("newHerbAg") .Value = glbHerbAgTractH
  .Fields("newWater") .Value = 0 'If developed, assumed to have no water
  .Fields("newDisturb") .Value = glbDisturbTractH
  .Fields("newMixed") .Value = glbMixedTractH
  Update
End With

Next Counter
lyrH.Records.StopEditing
End If

' This block adds fields to the database of commercial development squares and
populates those fields
Dim gdC As MapObjects2.GeoDataset
Set gdC = dc.AddGeoDataset(glbDevLayerFilenameC, moPolygon, tDesc)
If gdC Is Nothing Then Exit Sub 'invalid file
Dim lyrC As New MapObjects2.MapLayer
Set lyrC.GeoDataset = gdC
If colDevelSquaresC.Count <> 0 Then
For Counter = 1 To colDevelSquaresC.Count
    finished1 = False
    With lyrC.Records
        .AddNew 'Adds a new record to the development squares database
        'The following block of code (the do loop and the .fields().value assignments)
        'is used to grab attribute data from the grid pattern
        gridRecs.MoveFirst 'Go to the first record in the grid pattern database
        Do While Not gridRecs.EOF And finished1 = False 'Loop thru the records
            'If the centroid of a square in the grid pattern matches the centroid of
            'the development square, then store the data from the grid square
            If Fix(colDevelSquaresC(Counter).Centroid.X) = Fix(gridRecs.Fields("centerx").Value)  
                And Fix(colDevelSquaresC(Counter).Centroid.Y) = Fix(gridRecs.Fields("centery").Value) Then
                GridAttribute.develop = gridRecs.Fields("devel").Value
                GridAttribute.CenterX = gridRecs.Fields("CenterX").Value
                GridAttribute.oldForest = gridRecs.Fields("oldForest").Value
                GridAttribute.oldHerbAg = gridRecs.Fields("oldHerbAg").Value
                GridAttribute.oldWater = gridRecs.Fields("oldWater").Value
                GridAttribute.oldDisturb = gridRecs.Fields("oldDisturb").Value
                GridAttribute.oldMixed = gridRecs.Fields("oldMixed").Value
                GridAttribute.LandSeg = gridRecs.Fields("landseg").Value
                GridAttribute.LogPVHec = gridRecs.Fields("LogPVHec").Value
                finished1 = True
            End If
            gridRecs.MoveNext
    Loop
    'Set the values of the development square's fields
    .Fields("Shape").Value = colDevelSquaresC(Counter)
    .Fields("Name").Value = "Commercial " & Str(Counter)
    .Fields("Area").Value = colDevelSquaresC(Counter).Area
    .Fields("develop").Value = GridAttribute.develop
    .Fields("CenterX").Value = GridAttribute.CenterX
    .Fields("CenterY").Value = GridAttribute.CenterY
    .Fields("oldForest").Value = GridAttribute.oldForest
    .Fields("oldHerbAg").Value = GridAttribute.oldHerbAg
    .Fields("oldWater").Value = GridAttribute.oldWater
    .Fields("oldDisturb").Value = GridAttribute.oldDisturb
    .Fields("oldMixed").Value = GridAttribute.oldMixed
    .Fields("LandSeg").Value = GridAttribute.LandSeg
    .Fields("DevType").Value = "COM"
    .Fields("LogPVHec").Value = GridAttribute.LogPVHec
    .Fields("newForest").Value = glbForestTractC
    .Fields("newHerbAg").Value = glbHerbAgTractC
    .Fields("newWater").Value = 0 'If developed, assumed to have no water
    .Fields("newDisturb").Value = glbDisturbTractC
    .Fields("newMixed").Value = glbMixedTractC
    .Update
    End With
Next Counter
lyrC.Records.StopEditing
End If

'This block takes the individual development square shapefiles (up to 5 types) and combines them into a single shapefile

'Set up the shapefile that combines all of the individual shapefiles
Dim GDCombined As New MapObjects2.GeoDataset
glbCombinedDevLayerFilename = "aaCombined"
Set GDCombined = dc.AddGeoDataset(glbCombinedDevLayerFilename, moPolygon, tDesc)
Dim LyrCombined As New MapObjects2.MapLayer
Set LyrCombined.GeoDataset = GDCombined

'Define the recordsets for each individual shapefile
Dim LowRecSet As MapObjects2.Recordset
Set LowRecSet = lyrL.Records
Dim MidConvRecSet As MapObjects2.Recordset
Set MidConvRecSet = lyrMC.Records
Dim MidClusRecSet As MapObjects2.Recordset
Set MidClusRecSet = lyrMCL.Records
Dim HighRecSet As MapObjects2.Recordset
Set HighRecSet = lyrH.Records
Dim ComRecSet As MapObjects2.Recordset
Set ComRecSet = lyrC.Records

'Add records from the low density layer
LowRecSet.MoveFirst
Do While Not LowRecSet.EOF
  With LyrCombined.Records
    .AddNew
    .Fields("Shape").Value = LowRecSet.Fields("Shape").Value
    .Fields("Name").Value = LowRecSet.Fields("Name").Value
    .Fields("Area").Value = LowRecSet.Fields("Area").Value
    .Fields("develop").Value = LowRecSet.Fields("develop").Value
    .Fields("CenterX").Value = LowRecSet.Fields("CenterX").Value
    .Fields("CenterY").Value = LowRecSet.Fields("CenterY").Value
    .Fields("oldForest").Value = LowRecSet.Fields("oldForest").Value
    .Fields("oldHerbAg").Value = LowRecSet.Fields("oldHerbAg").Value
    .Fields("oldWater").Value = LowRecSet.Fields("oldWater").Value
    .Fields("oldDisturb").Value = LowRecSet.Fields("oldDisturb").Value
    .Fields("oldMixed").Value = LowRecSet.Fields("oldMixed").Value
    .Fields("LandSeg").Value = LowRecSet.Fields("LandSeg").Value
    .Fields("DevType").Value = LowRecSet.Fields("DevType").Value
    .Fields("LogPVHec").Value = LowRecSet.Fields("LogPVHec").Value
    .Fields("SewDist").Value = LowRecSet.Fields("SewDist").Value
    .Fields("newForest").Value = LowRecSet.Fields("newForest").Value
    .Fields("newHerbAg").Value = LowRecSet.Fields("newHerbAg").Value
    .Fields("newWater").Value = LowRecSet.Fields("newWater").Value
    .Fields("newDisturb").Value = LowRecSet.Fields("newDisturb").Value
    .Fields("newMixed").Value = LowRecSet.Fields("newMixed").Value
    .Update
  End With
  LowRecSet.MoveNext
Loop

'Add records from the mid density conventional layer
MidConvRecSet.MoveFirst
Do While Not MidConvRecSet.EOF
  With LyrCombined.Records
    .AddNew
    .Fields("Shape").Value = MidConvRecSet.Fields("Shape").Value
  End With
  MidConvRecSet.MoveNext
Loop
Fields("Name").Value = MidConvRecSet.Fields("Name").Value
.Fields("Area").Value = MidConvRecSet.Fields("Area").Value
.Fields("develop").Value = MidConvRecSet.Fields("develop").Value
.Fields("CenterX").Value = MidConvRecSet.Fields("CenterX").Value
.Fields("CenterY").Value = MidConvRecSet.Fields("CenterY").Value
.Fields("oldForest").Value = MidConvRecSet.Fields("oldForest").Value
.Fields("oldHerbAg").Value = MidConvRecSet.Fields("oldHerbAg").Value
.Fields("oldWater").Value = MidConvRecSet.Fields("oldWater").Value
.Fields("oldDisturb").Value = MidConvRecSet.Fields("oldDisturb").Value
.Fields("oldMixed").Value = MidConvRecSet.Fields("oldMixed").Value
.Fields("LandSeg").Value = MidConvRecSet.Fields("LandSeg").Value
.Fields("DevType").Value = MidConvRecSet.Fields("DevType").Value
.Fields("LogPVHec").Value = MidConvRecSet.Fields("LogPVHec").Value
.Fields("BlParcVal").Value = MidConvRecSet.Fields("BlParcVal").Value
.Fields("SewDist").Value = MidConvRecSet.Fields("SewDist").Value
.Fields("newForest").Value = MidConvRecSet.Fields("newForest").Value
.Fields("newHerbAg").Value = MidConvRecSet.Fields("newHerbAg").Value
.Fields("newWater").Value = MidConvRecSet.Fields("newWater").Value
.Fields("newDisturb").Value = MidConvRecSet.Fields("newDisturb").Value
.Fields("newMixed").Value = MidConvRecSet.Fields("newMixed").Value
.Update

End With
MidConvRecSet.MoveNext
Loop

'Add records from the mid density cluster layer
MidClusRecSet.MoveFirst
Do While Not MidClusRecSet.EOF
With LyrCombined.Records
    .AddNew
    .Fields("Shape").Value = MidClusRecSet.Fields("Shape").Value
    .Fields("Name").Value = MidClusRecSet.Fields("Name").Value
    .Fields("Area").Value = MidClusRecSet.Fields("Area").Value
    .Fields("develop").Value = MidClusRecSet.Fields("develop").Value
    .Fields("CenterX").Value = MidClusRecSet.Fields("CenterX").Value
    .Fields("CenterY").Value = MidClusRecSet.Fields("CenterY").Value
    .Fields("oldForest").Value = MidClusRecSet.Fields("oldForest").Value
    .Fields("oldHerbAg").Value = MidClusRecSet.Fields("oldHerbAg").Value
    .Fields("oldWater").Value = MidClusRecSet.Fields("oldWater").Value
    .Fields("oldDisturb").Value = MidClusRecSet.Fields("oldDisturb").Value
    .Fields("oldMixed").Value = MidClusRecSet.Fields("oldMixed").Value
    .Fields("LandSeg").Value = MidClusRecSet.Fields("LandSeg").Value
    .Fields("DevType").Value = MidClusRecSet.Fields("DevType").Value
    .Fields("LogPVHec").Value = MidClusRecSet.Fields("LogPVHec").Value
    .Fields("SewDist").Value = MidClusRecSet.Fields("SewDist").Value
    .Fields("newForest").Value = MidClusRecSet.Fields("newForest").Value
    .Fields("newHerbAg").Value = MidClusRecSet.Fields("newHerbAg").Value
    .Fields("newWater").Value = MidClusRecSet.Fields("newWater").Value
    .Fields("newDisturb").Value = MidClusRecSet.Fields("newDisturb").Value
    .Fields("newMixed").Value = MidClusRecSet.Fields("newMixed").Value
    .Update
End With
midClusRecSet.MoveNext
Loop

'Add records from the high density layer
HighRecSet.MoveFirst
Do While Not HighRecSet.EOF
With LyrCombined.Records
    .AddNew
    .Fields("Shape").Value = HighRecSet.Fields("Shape").Value
    .Fields("Name").Value = HighRecSet.Fields("Name").Value

.Fields("Area").Value = HighRecSet.Fields("Area").Value
.Fields("develop").Value = HighRecSet.Fields("develop").Value
.Fields("CenterX").Value = HighRecSet.Fields("CenterX").Value
.Fields("CenterY").Value = HighRecSet.Fields("CenterY").Value
.Fields("oldForest").Value = HighRecSet.Fields("oldForest").Value
.Fields("oldHerbAg").Value = HighRecSet.Fields("oldHerbAg").Value
.Fields("oldWater").Value = HighRecSet.Fields("oldWater").Value
.Fields("oldDisturb").Value = HighRecSet.Fields("oldDisturb").Value
.Fields("oldMixed").Value = HighRecSet.Fields("oldMixed").Value
.Fields("LandSeg").Value = HighRecSet.Fields("LandSeg").Value
.Fields("DevType").Value = HighRecSet.Fields("DevType").Value
.Fields("LogPVHec").Value = HighRecSet.Fields("LogPVHec").Value
.Fields("BiParcVal").Value = HighRecSet.Fields("BiParcVal").Value
.Fields("SewDist").Value = HighRecSet.Fields("SewDist").Value
.Fields("newForest").Value = HighRecSet.Fields("newForest").Value
.Fields("newHerbAg").Value = HighRecSet.Fields("newHerbAg").Value
.Fields("newWater").Value = HighRecSet.Fields("newWater").Value
.Fields("newDisturb").Value = HighRecSet.Fields("newDisturb").Value
.Fields("newMixed").Value = HighRecSet.Fields("newMixed").Value
.Update
End With
HighRecSet.MoveNext
Loop

'Add records from the commercial layer
ComRecSet.MoveFirst
Do While Not ComRecSet.EOF
With LyrCombined.Records
  .AddNew
  .Fields("Shape").Value = ComRecSet.Fields("Shape").Value
  .Fields("Name").Value = ComRecSet.Fields("Name").Value
  .Fields("Area").Value = ComRecSet.Fields("Area").Value
  .Fields("develop").Value = ComRecSet.Fields("develop").Value
  .Fields("CenterX").Value = ComRecSet.Fields("CenterX").Value
  .Fields("CenterY").Value = ComRecSet.Fields("CenterY").Value
  .Fields("oldForest").Value = ComRecSet.Fields("oldForest").Value
  .Fields("oldHerbAg").Value = ComRecSet.Fields("oldHerbAg").Value
  .Fields("oldWater").Value = ComRecSet.Fields("oldWater").Value
  .Fields("oldDisturb").Value = ComRecSet.Fields("oldDisturb").Value
  .Fields("oldMixed").Value = ComRecSet.Fields("oldMixed").Value
  .Fields("LandSeg").Value = ComRecSet.Fields("LandSeg").Value
  .Fields("DevType").Value = ComRecSet.Fields("DevType").Value
  .Fields("LogPVHec").Value = ComRecSet.Fields("LogPVHec").Value
  .Fields("BiParcVal").Value = ComRecSet.Fields("BiParcVal").Value
  .Fields("SewDist").Value = ComRecSet.Fields("SewDist").Value
  .Fields("newForest").Value = ComRecSet.Fields("newForest").Value
  .Fields("newHerbAg").Value = ComRecSet.Fields("newHerbAg").Value
  .Fields("newWater").Value = ComRecSet.Fields("newWater").Value
  .Fields("newDisturb").Value = ComRecSet.Fields("newDisturb").Value
  .Fields("newMixed").Value = ComRecSet.Fields("newMixed").Value
  .Update
End With
ComRecSet.MoveNext
Loop
LyrCombined.Records.StopEditing
dc.Disconnect
End Sub

'This function takes the x coordinate from a user-clicked point and
'makes it snap to the grid pattern. The idea is to compare the user-
'clicked point's x coordinate to the set of possible x coordinates from
'the grid. The program will then reassign the user-clicked x coordinate
'to the grid value
Public Function SnapXToGrid(ClickedX As Double) As Double

    Dim ComparisonX As Double
    Dim AbsDiff As Double, MinAbsDiff As Double

    MinAbsDiff = 999999999999#
    For ComparisonX = glbGridMinX To glbGridMaxX Step (glbSquareSize * 2) 'The loop
        AbsDiff = Abs(ComparisonX - ClickedX) 'Calculate the absolute value of the
        If AbsDiff < MinAbsDiff Then 'Keep track of the minimum absolute difference
            MinAbsDiff = AbsDiff
            SnapXToGrid = ComparisonX 'Keep the X coordinate with the minimum abs
        End If
    Next ComparisonX
End Function

'This function takes the y coordinate from a user-clicked point and
'makes it snap to the grid pattern. This is exactly like SnapXtoGrid
Public Function SnapYToGrid(ClickedY As Double) As Double

    Dim ComparisonY As Double
    Dim AbsDiff As Double, MinAbsDiff As Double

    MinAbsDiff = 999999999999#
    For ComparisonY = glbGridMinY To glbGridMaxY Step (glbSquareSize * 2)
        AbsDiff = Abs(ComparisonY - ClickedY)
        If AbsDiff < MinAbsDiff Then
            MinAbsDiff = AbsDiff
            SnapYToGrid = ComparisonY
        End If
    Next ComparisonY
End Function

'This function checks to see if a user-clicked square is inside the grid that is
'provided for them. If it is, then this function returns a value of 1. If it
'is not then this function returns a value of 0.
Public Function SeeThatSquareIsInGrid(UserPt As MapObjects2.Point) As Integer

    Dim dataConn As New MapObjects2.DataConnection
    dataConn.Database = App.Path & "\BC_Data"
    If (Not dataConn.Connect) Then
        MsgBox "Could not connect to database!"
        Exit Function 'bad dataConnection
    End If
    SeeThatSquareIsInGrid = 0

    SeeThatSquareIsInGrid = 0
    'This defines the geodataset, layer, and records for the grid pattern.
    'The newly created layer of development squares needs to draw out data
    'from the grid pattern.
    Dim gridGeoSet As MapObjects2.GeoDataset
    Dim gridLayer As New MapObjects2.MapLayer
    Dim gridRecords As New MapObjects2.Recordset
    Set gridLayer.GeoDataset = dataConn.FindGeoDataset("grid_pattern")
    Set gridRecords = gridLayer.Records

    Dim finished2 As Boolean
    finished2 = False

gridRecords.MoveNext 'Go to the first record in the grid pattern database
Do While Not gridRecords.EOF And finished2 = False 'Loop thru the records
' If the centroid of a square in the grid pattern matches the centroid of
'the development square, then keep it. Otherwise, throw it out
If Fix(UserPt.X) = Fix(gridRecords.Fields("centerx").Value) _
And Fix(UserPt.Y) = Fix(gridRecords.Fields("centery").Value) Then
  SeeThatSquareIsInGrid = 1
  finished2 = True
End If
gridRecords.MoveNext
Loop

dataConn.Disconnect

End Function

'This subroutine allows the user to remove a development square from the map
'by clicking on the square when the proper tool is selected
Public Sub RemoveSingleSquare(RemovePt As MapObjects2.Point)
    Dim RemoveClickX As Double, RemoveClickY As Double
    Dim DevSquareIterator As Integer
    RemoveClickX = RemovePt.X
    RemoveClickY = RemovePt.Y
    RemovePt.X = SnapXToGrid(RemoveClickX) 'In the AddDevelopmentsOperations module --
    changes x coordinate to fit the grid
    RemovePt.Y = SnapYToGrid(RemoveClickY) 'In the AddDevelopmentsOperations module --
    changes y coordinate to fit the grid
    'Get rid of a low density square
    DevSquareIterator = 1 'Start with the first development square
    'As long as the development square's position in the collection is less than
    'or equal to the total count of development squares, check to see if the user
    'clicked it. If so, then remove it from the collection
    Do While DevSquareIterator <= colDevelSquaresL.Count
      'Check to see if the clicked point is a development square
      If Fix(colDevelSquaresL(DevSquareIterator).Centroid.X) = Fix(RemovePt.X) And _
         Fix(colDevelSquaresL(DevSquareIterator).Centroid.Y) = Fix(RemovePt.Y) Then
        colDevelSquaresL.Remove (DevSquareIterator) 'if it is, then remove it
      End If
      DevSquareIterator = DevSquareIterator + 1 'move on to the next devSquare in the
      collection
    Loop
    'Also need to remove the development square from the collection of stored squares
    'so that a user can put a new development square on a vacated spot
    DevSquareIterator = 1
    Do While DevSquareIterator <= colStoredPolys.Count
      'Check to see if the clicked point is in the collection of stored development
      squares
      If Fix(colStoredPolys(DevSquareIterator).Centroid.X) = Fix(RemovePt.X) And _
         Fix(colStoredPolys(DevSquareIterator).Centroid.Y) = Fix(RemovePt.Y) Then
        colStoredPolys.Remove (DevSquareIterator) 'if it is, then remove it
      End If
      DevSquareIterator = DevSquareIterator + 1 'move on to the next devSquare in the
      collection
    Loop
'Get rid of a mid density conventional square
DevSquareIterator = 1 'Start with the first development square
'As long as the development square's position in the collection is less than
'or equal to the total count of development squares, check to see if the user
'clicked it. If so, then remove it from the collection
Do While DevSquareIterator <= colDevelSquaresMC.Count
  'Check to see if the clicked point is a development square
  If Fix(colDevelSquaresMC(DevSquareIterator).Centroid.X) = Fix(RemovePt.X) And _
    Fix(colDevelSquaresMC(DevSquareIterator).Centroid.Y) = Fix(RemovePt.Y) Then
    colDevelSquaresMC.Remove (DevSquareIterator) 'if it is, then remove it
  End If
  DevSquareIterator = DevSquareIterator + 1 'move on to the next devSquare in the
  Loop

'Also need to remove the development square from the collection of stored squares
'so that a user can put a new development square on a vacated spot
DevSquareIterator = 1
Do While DevSquareIterator <= colStoredPolys.Count
  'Check to see if the clicked point is in the collection of stored development
  squares
  If Fix(colStoredPolys(DevSquareIterator).Centroid.X) = Fix(RemovePt.X) And _
    Fix(colStoredPolys(DevSquareIterator).Centroid.Y) = Fix(RemovePt.Y) Then
    colStoredPolys.Remove (DevSquareIterator) 'if it is, then remove it
  End If
  DevSquareIterator = DevSquareIterator + 1 'move on to the next devSquare in the
  Loop

'Get rid of a mid density cluster square
DevSquareIterator = 1 'Start with the first development square
'As long as the development square's position in the collection is less than
'or equal to the total count of development squares, check to see if the user
'clicked it. If so, then remove it from the collection
Do While DevSquareIterator <= colDevelSquaresMCL.Count
  'Check to see if the clicked point is a development square
  If Fix(colDevelSquaresMCL(DevSquareIterator).Centroid.X) = Fix(RemovePt.X) And _
    Fix(colDevelSquaresMCL(DevSquareIterator).Centroid.Y) = Fix(RemovePt.Y) Then
    colDevelSquaresMCL.Remove (DevSquareIterator) 'if it is, then remove it
  End If
  DevSquareIterator = DevSquareIterator + 1 'move on to the next devSquare in the
  Loop

'Also need to remove the development square from the collection of stored squares
'so that a user can put a new development square on a vacated spot
DevSquareIterator = 1
Do While DevSquareIterator <= colStoredPolys.Count
  'Check to see if the clicked point is in the collection of stored development
  squares
  If Fix(colStoredPolys(DevSquareIterator).Centroid.X) = Fix(RemovePt.X) And _
    Fix(colStoredPolys(DevSquareIterator).Centroid.Y) = Fix(RemovePt.Y) Then
    colStoredPolys.Remove (DevSquareIterator) 'if it is, then remove it
End If  
    DevSquareIterator = DevSquareIterator + 1 'move on to the next devSquare in the collection
Loop

'Get rid of a high density square  
DevSquareIterator = 1 'Start with the first development square  
'As long as the development square's position in the collection is less than  
'or equal to the total count of development squares, check to see if the user  
'clicked it. If so, then remove it from the collection  
Do While DevSquareIterator <= colDevelSquaresH.Count  
    'Check to see if the clicked point is a development square  
    If Fix(colDevelSquaresH(DevSquareIterator).Centroid.X) = Fix(RemovePt.X) And  
        Fix(colDevelSquaresH(DevSquareIterator).Centroid.Y) = Fix(RemovePt.Y) Then  
        colDevelSquaresH.Remove (DevSquareIterator) 'if it is, then remove it  
    End If  
    DevSquareIterator = DevSquareIterator + 1 'move on to the next devSquare in the collection
Loop

'Also need to remove the development square from the collection of stored squares  
'so that a user can put a new development square on a vacated spot  
DevSquareIterator = 1  
Do While DevSquareIterator <= colStoredPolys.Count  
    'Check to see if the clicked point is in the collection of stored development squares  
    If Fix(colStoredPolys(DevSquareIterator).Centroid.X) = Fix(RemovePt.X) And  
        Fix(colStoredPolys(DevSquareIterator).Centroid.Y) = Fix(RemovePt.Y) Then  
        colStoredPolys.Remove (DevSquareIterator) 'if it is, then remove it  
    End If  
    DevSquareIterator = DevSquareIterator + 1 'move on to the next devSquare in the collection
Loop

'Get rid of a commercial square  
DevSquareIterator = 1 'Start with the first development square  
'As long as the development square's position in the collection is less than  
'or equal to the total count of development squares, check to see if the user  
'clicked it. If so, then remove it from the collection  
Do While DevSquareIterator <= colDevelSquaresC.Count  
    'Check to see if the clicked point is a development square  
    If Fix(colDevelSquaresC(DevSquareIterator).Centroid.X) = Fix(RemovePt.X) And  
        Fix(colDevelSquaresC(DevSquareIterator).Centroid.Y) = Fix(RemovePt.Y) Then  
        colDevelSquaresC.Remove (DevSquareIterator) 'if it is, then remove it  
    End If  
    DevSquareIterator = DevSquareIterator + 1 'move on to the next devSquare in the collection
Loop

'Also need to remove the development square from the collection of stored squares  
'so that a user can put a new development square on a vacated spot  
DevSquareIterator = 1  
Do While DevSquareIterator <= colStoredPolys.Count
'Check to see if the clicked point is in the collection of stored development squares
If Fix(colStoredPolys(DevSquareIterator).Centroid.X) = Fix(RemovePt.X) And _
Fix(colStoredPolys(DevSquareIterator).Centroid.Y) = Fix(RemovePt.Y) Then _
colStoredPolys.Remove (DevSquareIterator) 'if it is, then remove it
End If
DevSquareIterator = DevSquareIterator + 1 'move on to the next devSquare in the collection
Loop
End Sub

'This subroutine clears all developments off the map by removing squares from the 'collections of development squares
Public Sub ClearAllDevelopments()

'Get rid of low density squares
Do While colDevelSquaresL.Count <> 0
   colDevelSquaresL.Remove (1)
Loop

Do While colDevelSquaresMC.Count <> 0
   colDevelSquaresMC.Remove (1)
Loop

'Get rid of mid density cluster squares
Do While colDevelSquaresMCL.Count <> 0
   colDevelSquaresMCL.Remove (1)
Loop

'Get rid of high density squares
Do While colDevelSquaresH.Count <> 0
   colDevelSquaresH.Remove (1)
Loop

'Get rid of commercial squares
Do While colDevelSquaresC.Count <> 0
   colDevelSquaresC.Remove (1)
Loop

frmMain.mapDisplay.TrackingLayer.Refresh True 'This sends the program to the subroutine that draws the map

'Clear the collection of stored development squares so that the user can put 'development squares down in spots that have been cleared
Do While colStoredPolys.Count <> 0
   colStoredPolys.Remove (1)
Loop

'Reenable the map display and input button (if they have been disabled), so the 'user can add new developments
frmMain.mapDisplay.Enabled = True
frmMain.cmdInputComplete.Enabled = True

'Clean up any report forms that are open
If frmHydroReport.Visible Then Unload frmHydroReport
If frmEconExcelView.Visible Then Unload frmEconExcelView
If FrmEconInput.Visible Then Unload FrmEconInput
If frmFishReport.Visible Then Unload frmFishReport
If frmHelp.Visible Then Unload frmHelp
'Hide the status message if it is visible
frmMain.fraStatusMessage.Visible = False

End Sub

'This subroutine looks for "super groups" of development squares, so they
'can be considered as one development. A single square is part of a
'super group if it shares a point with any other square.
Public Sub FindSuperGroupsOfSquares()

'Set up data connection
Dim dcSquares As New MapObjects2.DataConnection
dcSquares.Database = App.Path & \"BC_Data\"
If (Not dcSquares.Connect) Then
    MsgBox "Could not connect to database!"
    Exit Sub 'bad dataConnection
End If

'Set up connection to development squares
Dim gdSquares As New MapObjects2.GeoDataset
Dim lyrSquares As New MapObjects2.MapLayer
Dim rsSquares As New MapObjects2.Recordset
Set lyrSquares.GeoDataset = dcSquares.FindGeoDataset(glbCombinedDevLayerFilename)
Set rsSquares = lyrSquares.Records

'The field SuperGrp will be used to store whether a square is part of
'a super group -- 0 means it is not, and a number above 0 means it is.
'Squares in a particular super group have the same number in the SuperGrp field
Dim BaseX As Double, BaseY As Double, CompareX As Double, CompareY As Double
Dim CompareGrpNum As Integer
Dim BaseGrpNum As Integer
Dim i As Integer, j As Integer, k As Integer, recCount As Integer
Dim SuperGrpNum As Integer, SuperGrpNumHolder As Integer
Dim FoundAdjacent As Boolean
Dim OutRecCount As Integer, InRecCount As Integer
Dim NumBaseAdjacent As Integer, NumExtraAdjacent As Integer
Dim Xdiff As Double, Ydiff As Double

'Initialize all squares as not being part of a super group.
' Edit values in the database
' Update method causes a jump back to the first record in the recordset
' Use a loop to move back down again to the appropriate record
rsSquares.MoveFirst
recCount = 0
Do While Not rsSquares.EOF
    recCount = recCount + 1
    With lyrSquares.Records
        .MoveFirst
        For j = 2 To recCount
            .MoveNext
            Next j
        .Edit
        .Fields("SuperGrp").Value = 0
        .Update
    End With
    rsSquares.MoveNext
Loop

'Now look for squares that are adjacent to others
SuperGrpNumHolder = 1
'Make a loop that will go through each square. Each of the squares of this outer loop are considered to be the "base" square.
For OutRecCount = 1 To rsSquares.Count
    FoundAdjacent = False
    rsSquares.MoveFirst
    'Step down in the recordset to the base square in the loop
    For k = 2 To OutRecCount
        rsSquares.MoveNext
    Next k
    'Get the base x and y coordinates
    BaseX = rsSquares.Fields("CenterX").Value
    BaseY = rsSquares.Fields("CenterY").Value
    'Get the base supergroup number
    BaseGrpNum = rsSquares.Fields("SuperGrp").Value

    'Make a loop to run through each of the squares. The squares in this loop are the comparison squares which get compared to the base for adjacency
    rsSquares.MoveFirst
    InRecCount = 0
    'Run loop thru each development square until an adjacent one is found
    Do While Not rsSquares.EOF And FoundAdjacent = False
        'Track which record the loop is on
        InRecCount = InRecCount + 1
        'Get the comparison x and y coordinates
        CompareX = rsSquares.Fields("CenterX").Value
        CompareY = rsSquares.Fields("CenterY").Value
        'Get the comparison square's supergroup number
        CompareGrpNum = rsSquares.Fields("SuperGrp").Value

        'Calculate the difference between coordinates of the base square and the comparison square
        Xdiff = BaseX - CompareX
        Ydiff = BaseY - CompareY

        'Send the difference in coordinate values to functions that test for adjacency. If any of the functions show adjacency then adjust the supergroup number for the base and comparison squares.
        If NorthAdjacent(Xdiff, Ydiff) = True Or _
            SouthAdjacent(Xdiff, Ydiff) = True Or _
            EastAdjacent(Xdiff, Ydiff) = True Or _
            WestAdjacent(Xdiff, Ydiff) = True Or _
            NEAdjacent(Xdiff, Ydiff) = True Or _
            NWAdjacent(Xdiff, Ydiff) = True Or _
            SEAdjacent(Xdiff, Ydiff) = True Or _
            SWAdjacent(Xdiff, Ydiff) = True Then
            SuperGrpNum = CompareGrpNum
            Else: SuperGrpNum = SuperGrpNumHolder
                SuperGrpNumHolder = SuperGrpNumHolder + 1
        End If
        'If the groupnumber on either square has already been changed from 0
        'then set the supergrp number to the number of the comparison square.
        'Otherwise, set the supergrp number equal to the holder value.
        If CompareGrpNum > 0 Or BaseGrpNum > 0 Then
            SuperGrpNum = CompareGrpNum
        End If
        With lyrSquares.Records
            .MoveFirst
            For j = 2 To OutRecCount
                .MoveNext
            End For
        End With
    Loop
    'Edit the base square in the database
With lyrSquares.Records
    .MoveFirst
    For j = 2 To OutRecCount
        .MoveNext
    End For
End With
End Loop

Next j
  .Edit
  .Fields("SuperGrp").Value = SuperGrpNum
  .Update

  'Edit the comparison square in the database
  .MoveFirst
  For j = 2 To InRecCount
    .MoveNext
    Next j
  .Edit
  .Fields("SuperGrp").Value = SuperGrpNum
  .Update

  FoundAdjacent = True
End With
End If

rsSquares.MoveNext
Loop
Next OutRecCount

'Because the SuperGroupHolder value may have been incremented too
'many times depending on how the user laid down squares on the map, the
'supergroup values in the database need to be readjusted.

'Run an outer loop that will comb through all the squares to make sure the group
'numbering correction hits every tract
Dim catchallsquares As Integer
For catchallsquares = 1 To rsSquares.Count
  'Run an outer loop for the base square
  For OutRecCount = 1 To rsSquares.Count
    rsSquares.MoveFirst
    For k = 2 To OutRecCount
      rsSquares.MoveNext
    Next k
    'Get the x and y coordinates
    BaseX = rsSquares.Fields("CenterX").Value
    BaseY = rsSquares.Fields("CenterY").Value

    'Get the supergrp number for the base square
    BaseGrpNum = rsSquares.Fields("SuperGrp").Value

    'Run an inner loop for the comparison square
    rsSquares.MoveFirst
    InRecCount = 0
    Do While Not rsSquares.EOF
      InRecCount = InRecCount + 1
      'Get the x and y coordinates
      CompareX = rsSquares.Fields("CenterX").Value
      CompareY = rsSquares.Fields("CenterY").Value

      'Calculate the difference between base and comparison coordinates
      Xdiff = BaseX - CompareX
      Ydiff = BaseY - CompareY

      'Get the supergrp number for the comparison square
      CompareGrpNum = rsSquares.Fields("SuperGrp").Value

      'Use functions to check for adjacency
If NorthAdjacent(Xdiff, Ydiff) = True Or _
    SouthAdjacent(Xdiff, Ydiff) = True Or _
    EastAdjacent(Xdiff, Ydiff) = True Or _
    WestAdjacent(Xdiff, Ydiff) = True Or _
    NEAdjacent(Xdiff, Ydiff) = True Or _
    NWAdjacent(Xdiff, Ydiff) = True Or _
    SEAdjacent(Xdiff, Ydiff) = True Or _
    SWAdjacent(Xdiff, Ydiff) = True Then

    'if squares are adjacent and the comparison supergrp number is higher than
    'a non-zero base group number then set the supergrp number = to the base
    'supergrp number
    If CompareGrpNum > BaseGrpNum And BaseGrpNum > 0 Then
        SuperGrpNum = BaseGrpNum
        With lyrSquares.Records
            .MoveFirst
            For j = 2 To InRecCount
                .MoveNext
                Next j
            .Edit
            .Fields("SuperGrp").Value = SuperGrpNum
            .Update
        End With
    End If

    'if squares are adjacent and the base supergrp number is higher than
    'a non-zero comparison group number then set the supergrp number = to the
    'comparison supergrp number
    If BaseGrpNum > CompareGrpNum And CompareGrpNum > 0 Then
        SuperGrpNum = CompareGrpNum
        With lyrSquares.Records
            .MoveFirst
            For j = 2 To OutRecCount
                .MoveNext
                Next j
            .Edit
            .Fields("SuperGrp").Value = SuperGrpNum
            .Update
        End With
    End If

End If
rsSquares.MoveNext
Loop
Next OutRecCount
Next catchallsquares
End Sub

' This function tells whether a development square in a particular direction
' is adjacent to the square of interest. Xdiff and Ydiff are the differences in the
' two squares' centroid coordinates
Public Function EastAdjacent(Xdiff, Ydiff As Double) As Boolean
    If Xdiff = 2 * glbSquareSize And Ydiff = 0 Then
        EastAdjacent = True
    Else
        EastAdjacent = False
    End If
End Function

' This function tells whether a development square in a particular direction
' is adjacent to the square of interest. Xdiff and Ydiff are the differences in the
' two squares' centroid coordinates
Public Function WestAdjacent(Xdiff, Ydiff As Double) As Boolean
    If Xdiff = -(2 * glbSquareSize) And Ydiff = 0 Then
        WestAdjacent = True
    Else
        WestAdjacent = False
    End If
End Function

This function tells whether a development square in a particular direction
is adjacent to the square of interest. Xdiff and Ydiff are the differences in the
'two squares' centroid coordinates

Public Function NorthAdjacent(Xdiff, Ydiff As Double) As Boolean
    If Xdiff = 0 And Ydiff = 2 * glbSquareSize Then
        NorthAdjacent = True
    Else
        NorthAdjacent = False
    End If
End Function

This function tells whether a development square in a particular direction
is adjacent to the square of interest. Xdiff and Ydiff are the differences in the
'two squares' centroid coordinates

Public Function SouthAdjacent(Xdiff, Ydiff As Double) As Boolean
    If Xdiff = 0 And Ydiff = -(2 * glbSquareSize) Then
        SouthAdjacent = True
    Else
        SouthAdjacent = False
    End If
End Function

This function tells whether a development square in a particular direction
is adjacent to the square of interest. Xdiff and Ydiff are the differences in the
'two squares' centroid coordinates

Public Function NEAdjacent(Xdiff, Ydiff As Double) As Boolean
    If Xdiff = 2 * glbSquareSize And Ydiff = 2 * glbSquareSize Then
        NEAdjacent = True
    Else
        NEAdjacent = False
    End If
End Function

This function tells whether a development square in a particular direction
is adjacent to the square of interest. Xdiff and Ydiff are the differences in the
'two squares' centroid coordinates

Public Function NWAdjacent(Xdiff, Ydiff As Double) As Boolean
    If Xdiff = -(2 * glbSquareSize) And Ydiff = 2 * glbSquareSize Then
        NWAdjacent = True
    Else
        NWAdjacent = False
    End If
End Function

This function tells whether a development square in a particular direction
is adjacent to the square of interest. Xdiff and Ydiff are the differences in the
'two squares' centroid coordinates

Public Function SEAdjacent(Xdiff, Ydiff As Double) As Boolean
    If Xdiff = 2 * glbSquareSize And Ydiff = -(2 * glbSquareSize) Then
        SEAdjacent = True
    Else
        SEAdjacent = False
    End If
End Function

This function tells whether a development square in a particular direction
is adjacent to the square of interest. Xdiff and Ydiff are the differences in the
'two squares' centroid coordinates

Public Function SWAdjacent(Xdiff, Ydiff As Double) As Boolean
    If Xdiff = -(2 * glbSquareSize) And Ydiff = -(2 * glbSquareSize) Then
        SWAdjacent = True
    Else
        SWAdjacent = False
    End If
End Function
Else
    SWAdjacent = False
End If
End Function

**HydroOperations Module**

Option Explicit
' ******************************************************************************
' * The HydroOperations module contains a collection of subroutines     *
' * that provide modeling steps for hydrology                          *
' * Created February 29, 1999                                         *
' ******************************************************************************

'This subroutine creates files to pass land use information to HSPF
'and shows the user the values in the two files. The file of original
'land use values is OldLandUse.prn, and the file of scenario values
'is NewLandUse.prn
Public Sub MakeHSPFFile()
    Dim dcHydro As New MapObjects2.DataConnection
dcHydro.Database = App.Path & \"BC_Data\"

    'Set up the dataset information for the grid squares
    Dim gdGrid As MapObjects2.GeoDataset
    Set gdGrid = dcHydro.FindGeoDataset("grid_pattern")
    If gdGrid Is Nothing Then 'invalid file
        MsgBox "Couldn't find the data. Please try again."
        Exit Sub
    End If
    Dim lyrGrid As New MapObjects2.MapLayer
    Set lyrGrid.GeoDataset = gdGrid
    Dim rsGrid As New MapObjects2.Recordset
    Set rsGrid = lyrGrid.Records

    'Set up the dataset information for the combined types of development squares
    Dim gdDevels As MapObjects2.GeoDataset
    Set gdDevels = dcHydro.FindGeoDataset(glbCombinedDevLayerFilename)
    If gdDevels Is Nothing Then 'invalid file
        MsgBox "Couldn't find the data. Please try again."
        Exit Sub
    End If
    Dim lyrDevels As New MapObjects2.MapLayer
    Set lyrDevels.GeoDataset = gdDevels
    Dim rsDevels As New MapObjects2.Recordset
    Set rsDevels = lyrDevels.Records

    'Initialize the total amount of land use for each land segment to 0
    Dim i As Integer
    For i = 1 To numLandSegs
        glbTotOldForest(i) = 0
        glbTotOldHerbAg(i) = 0
        glbTotOldWater(i) = 0
        glbTotOldDisturb(i) = 0
        glbTotOldMixed(i) = 0
        glbTotAllLandUses(i) = 0
    Next i

    'This variable is the land segment for the current shape
    Dim LandSegID As Integer

    'Loop through all the records in the grid pattern to add up the amount
of area for each land segment and land use type
rsGrid.MoveFirst
Do While Not rsGrid.EOF
    LandSegID = rsGrid.Fields("Landseg").Value 'Set the land segment id
    glbTotOldForest(LandSegID) = glbTotOldForest(LandSegID) +
    rsGrid.Fields("Oldforest").Value
    glbTotOldHerbAg(LandSegID) = glbTotOldHerbAg(LandSegID) +
    rsGrid.Fields("Oldherbag").Value
    glbTotOldWater(LandSegID) = glbTotOldWater(LandSegID) +
    rsGrid.Fields("Oldwater").Value
    glbTotOldDisturb(LandSegID) = glbTotOldDisturb(LandSegID) +
    rsGrid.Fields("Olddisturb").Value
    glbTotOldMixed(LandSegID) = glbTotOldMixed(LandSegID) +
    rsGrid.Fields("Oldmixed").Value
    rsGrid.MoveNext 'Move on to the next record
Loop

'Set up output file information for storing data to pass to HSPF
Dim FreeFileNum, oldHSPFFileNum, newHSPFFileNum As Integer
FreeFileNum = 1
Dim oldHSPFFileName, newHSPFFileName As String
oldHSPFFileNum = FreeFileNum
Open App.Path & "\OutputFiles\OrigLandUse.prn" For Output As #oldHSPFFileNum

'Loop through each land segment
For i = 1 To numLandSegs
    'Add up the total area of all land uses for each land segment
    glbTotAllLandUses(i) = glbTotAllLandUses(i) + glbTotOldForest(i) +
                        glbTotOldHerbAg(i) + glbTotOldWater(i) + glbTotOldDisturb(i) +
                        glbTotOldMixed(i)
    'Calculate percentage of each land segment that is a particular land use
    glbPctOldForest(i) = glbTotOldForest(i) / glbTotAllLandUses(i)
    glbPctOldHerbAg(i) = glbTotOldHerbAg(i) / glbTotAllLandUses(i)
    glbPctOldWater(i) = glbTotOldWater(i) / glbTotAllLandUses(i)
    glbPctOldDisturb(i) = glbTotOldDisturb(i) / glbTotAllLandUses(i)
    glbPctOldMixed(i) = glbTotOldMixed(i) / glbTotAllLandUses(i)
    'Print the percentage values to a file that can be used by HSPF
    'Note -- Ignoring water as a land use because it is insignificant
    'Rounding values to 2 decimal places to avoid false precision
    Print #oldHSPFFileNum, Tab(0); i; Tab(7); _
        Round(glbPctOldForest(i), 2); Tab(21); _
        Round(glbPctOldHerbAg(i), 2); Tab(35); _
        Round(glbPctOldDisturb(i), 2); Tab(49); _
        Round(glbPctOldMixed(i), 2)
Next i

'Close the file that was just written and increment the free file number
Close #oldHSPFFileNum
FreeFileNum = FreeFileNum + 1

'Initialize the values of land use areas to old values
For i = 1 To numLandSegs
    glbTotNewForest(i) = glbTotOldForest(i)
    glbTotNewHerbAg(i) = glbTotOldHerbAg(i)
    glbTotNewWater(i) = glbTotOldWater(i)
    glbTotNewDisturb(i) = glbTotOldDisturb(i)
    glbTotNewMixed(i) = glbTotOldMixed(i)
Next i

'Loop through all the records in the developments and add up the amount of
'area for each land segment and land use type. do this by adding the new land use 'areas and subtracting the old areas.

Do While Not rsDevels.EOF
    LandSegID = rsDevels.Fields("Landseg").Value 'Set the land segment id
    glbTotNewForest(LandSegID) = glbTotNewForest(LandSegID) + _
        rsDevels.Fields("Newforest").Value - rsDevels.Fields("Oldforest").Value
    glbTotNewHerbAg(LandSegID) = glbTotNewHerbAg(LandSegID) + _
        rsDevels.Fields("Newherbag").Value - rsDevels.Fields("Oldherbag").Value
    glbTotNewWater(LandSegID) = glbTotNewWater(LandSegID) + _
        rsDevels.Fields("Newwater").Value - rsDevels.Fields("Oldwater").Value
    glbTotNewDisturb(LandSegID) = glbTotNewDisturb(LandSegID) + _
        rsDevels.Fields("Newdisturb").Value - rsDevels.Fields("Olddisturb").Value
    glbTotNewMixed(LandSegID) = glbTotNewMixed(LandSegID) + _
        rsDevels.Fields("Newmixed").Value - rsDevels.Fields("Oldmixed").Value
    rsDevels.MoveNext 'Move on to the next record
Loop

'Present the form that shows the user the values stored in the two output files
frmHydroReport.Show

'Add labels for the user to see original land use values
frmHydroReport.lstOldLandSeg.AddItem ("Land Seg")
frmHydroReport.lstOldForest.AddItem (" Forest")
frmHydroReport.lstOldHerbAg.AddItem (" Herb/Ag")
frmHydroReport.lstOldDisturb.AddItem (" Disturbed")
frmHydroReport.lstOldMixed.AddItem (" Mixed")

'Add labels for the user to see scenario land use values
frmHydroReport.lstNewLandSeg.AddItem ("Land Seg")
frmHydroReport.lstNewForest.AddItem (" Forest")
frmHydroReport.lstNewHerbAg.AddItem (" Herb/Ag")
frmHydroReport.lstNewDisturb.AddItem (" Disturbed")
frmHydroReport.lstNewMixed.AddItem (" Mixed")

'Loop through each land segment
For i = 1 To numLandSegs
    'Calculate percentage of each land segment that is a particular land use
    glbPctNewForest(i) = glbTotNewForest(i) / glbTotAllLandUses(i)
    glbPctNewHerbAg(i) = glbTotNewHerbAg(i) / glbTotAllLandUses(i)
    glbPctNewWater(i) = glbTotNewWater(i) / glbTotAllLandUses(i)
    glbPctNewDisturb(i) = glbTotNewDisturb(i) / glbTotAllLandUses(i)
    glbPctNewMixed(i) = glbTotNewMixed(i) / glbTotAllLandUses(i)

    'Print the percentage values to a file that can be used by HSPF
    Note -- Ignoring water as a land use because it is insignificant
    'Rounding values to 2 decimal places to avoid false precision
    Print #newHSPFFileNum, Tab(0); i; Tab(7); _
        Round(glbPctNewForest(i), 2); Tab(21); _
        Round(glbPctNewHerbAg(i), 2); Tab(35); _
Round(glbPctNewDisturb(i), 2); Tab(49); _
Round(glbPctNewMixed(i), 2)

'Put the values in a list box for the user to see
frmHydroReport.lstOldLandSeg.AddItem (" " & i)
frmHydroReport.lstOldForest.AddItem (" " & Round(glbPctOldForest(i), 2))
frmHydroReport.lstOldDisturb.AddItem (" " & Round(glbPctOldDisturb(i), 2))
frmHydroReport.lstOldHerbAg.AddItem (" " & Round(glbPctOldHerbAg(i), 2))
frmHydroReport.lstOldMixed.AddItem (" " & Round(glbPctOldMixed(i), 2))

frmHydroReport.lstNewLandSeg.AddItem (" " & i)
frmHydroReport.lstNewForest.AddItem (" " & Round(glbPctNewForest(i), 2))
frmHydroReport.lstNewDisturb.AddItem (" " & Round(glbPctNewDisturb(i), 2))
frmHydroReport.lstNewHerbAg.AddItem (" " & Round(glbPctNewHerbAg(i), 2))
frmHydroReport.lstNewMixed.AddItem (" " & Round(glbPctNewMixed(i), 2))

Next i

'Close the file that was just written
Close #newHSPFFileNum

dcHydro.Disconnect

End Sub

EconOperations Module

Option Explicit
Option Base 0

' ******************************************************************************
' * The EconOperations module contains a collection of subroutines that provide modeling procedures for calculating changes in local government tax revenues and expenditures. *
' * Created March 7, 1999 *
' ******************************************************************************

'This subroutine writes out changes from baseline gov't revs and expenditures to an excel spreadsheet
Public Sub MakeEconExcelFile()

'*******************************************************************************
'This part calculates values that will be put in the spreadsheet
Dim DatConnect As New MapObjects2.DataConnection
DatConnect.Database = App.Path & "\BC_Data"

'These are variables for the development squares data
Dim developGD As MapObjects2.GeoDataset
Set developGD = DatConnect.FindGeoDataset(glbCombinedDevLayerFilename)
If developGD Is Nothing Then
    MsgBox "Couldn't find data for your development scenario!"
    Exit Sub
End If
Dim lyrDevelop As New MapObjects2.MapLayer
Set lyrDevelop.GeoDataset = developGD
Dim developRecSet As MapObjects2.Recordset 'This is the set of records for the developments
Set developRecSet = lyrDevelop.Records

'Note: Variables of type typEconTractDataHolder hold values for each kind of development tract.
'Variables to calculate regression results
Dim RegressPart1 As Double, RegressPart2 As Double, RegressPart3 As Double
Dim TypeOfDevelop As String 'Identifies kind of tract
Dim ParcSizeHec As typEconTractDataHolder 'Parcel size in hectares
Dim NumberOfParcelsPerTract As typEconTractDataHolder
Dim NumberOfPeoplePerTract As typEconTractDataHolder
Dim TotNumberOfParcels As typEconTractDataHolder
Dim TotNumberOfPeople As typEconTractDataHolder
'Estimated value (from regression) of individual parcel
Dim NewIndividParcelValue As typEconTractDataHolder
'Estimated value (from adding up individual parcels) of all parcels of tract type
Dim NewAggParcelValue As typEconTractDataHolder
'Holds fraction of open space for each kind of tract
Dim OpenSpaceFraction As typEconTractDataHolder
'Fractions come from Stephenson report -- only mid density tracts provide open space
OpenSpaceFraction.Low = 0
OpenSpaceFraction.MidConv = 0.23
OpenSpaceFraction.MidClus = 0.45
OpenSpaceFraction.High = 0
OpenSpaceFraction.Comm = 0
'Fractions come from Stephenson report -- Fraction of land occupied by infrastructure
Dim InfrastructureFraction As typEconTractDataHolder
  InfrastructureFraction.Low = 0.08
  InfrastructureFraction.MidConv = 0.11
  InfrastructureFraction.MidClus = 0.07
  InfrastructureFraction.High = 0.17
  InfrastructureFraction.Comm = 0.17 'assumed to be the same as high density
'Fractions come from Stephenson report -- Fraction of land occupied by privately owned parcels
Dim PrivateSpaceFraction As typEconTractDataHolder
  PrivateSpaceFraction.Low = 1 - (OpenSpaceFraction.Low + InfrastructureFraction.Low)
  PrivateSpaceFraction.MidConv = 1 - (OpenSpaceFraction.MidConv + InfrastructureFraction.MidConv)
  PrivateSpaceFraction.MidClus = 1 - (OpenSpaceFraction.MidClus + InfrastructureFraction.MidClus)
  PrivateSpaceFraction.High = 1 - (OpenSpaceFraction.High + InfrastructureFraction.High)
Dim NumberOfTracts As typEconTractDataHolder
Dim TractSizeHec As Integer 'Tract size in hectares
'Tract size in hectares = square meters * a conversion factor of 0.0001
TractSizeHec = ((glbSquareSize * 2) ^ 2) * 0.0001
Dim BaselineTractValue As typEconTractDataHolder 'Baseline value of tract
Dim AggBaselineTractValue As typEconTractDataHolder 'Baseline value of all tracts of each type
Dim difBareLandValue As typEconTractDataHolder 'land value difference between new and baseline
Dim IndividHouseCost As typEconTractDataHolder 'Construction cost for houses on each type of tract
IndividHouseCost.Low = 215355.8 'Given in Speir report as custom 2.5 story Roanoke
IndividHouseCost.MidConv = 161200.8 'Given in Speir report as custom 1 story Roanoke
IndividHouseCost.MidClus = 161200.8 'Given in Speir report as custom 1 story Roanoke
IndividHouseCost.High = 161200.8 'Given in Speir report as custom 1 story Roanoke
'No information for commercial buildings
Dim AggHouseCost As typEconTractDataHolder 'Aggregated construction cost for houses
'Road, sidewalk, and storm drain construction cost for each kind of tract (Roanoke)
'From Speir report
Dim IndividRoadCost As typEconTractDataHolder
IndividRoadCost.Low = 13828.5 + 2933.52 + 598.4448
IndividRoadCost.MidConv = 3802.838 + 806.443 + 1346.501
IndividRoadCost.MidClus = 2854.598 + 659.817 + 807.9005
IndividRoadCost.High = 2854.598 + 659.817 + 987.4339
Dim AggRoadCost As typEconTractDataHolder 'Aggregated construction cost for roads, etc.
'Sewer construction cost for each kind of tract (Roanoke)
'From Speir report
Dim IndividSewerCost As typEconTractDataHolder
IndividSewerCost.Low = 0
IndividSewerCost.MidConv = 829.1393
IndividSewerCost.MidClus = 678.3867
IndividSewerCost.High = 678.3867
Dim AggSewerCost As typEconTractDataHolder 'Aggregated construction cost for sewer/septic
'Water supply construction cost for each kind of tract (Roanoke)
'From Speir report
Dim IndividWaterCost As typEconTractDataHolder
IndividWaterCost.Low = 0
IndividWaterCost.MidConv = 1466.567
IndividWaterCost.MidClus = 1199.919
IndividWaterCost.High = 1199.919
Dim AggWaterCost As typEconTractDataHolder 'Aggregated construction cost for water
Dim TotDevelopmentCost As typEconTractDataHolder 'Sum of house, road, sewer, and water
dim TotTractCost As typEconTractDataHolder 'Sum of bare land value and development costs
'These variables grab information from the econ user input form
Dim UseAssessRateAg As Double 'The rate ($/hec) at which ag land is assessed
UseAssessRateAg = Val(FrmEconInput.txtUseAssessAg)
Dim UseAssessRateFor As Double 'The rate ($/hec) at which forest land is assessed
UseAssessRateFor = Val(FrmEconInput.txtUseAssessFor)
Dim UseAssessRateOS As Double 'The rate ($/hec) at which land in open space is assessed
UseAssessRateOS = Val(FrmEconInput.txtUseAssessOS)
Dim MktAssessRate As Double 'Percentage multiplied by market value of land to calculate assessed value of land
MktAssessRate = Val(FrmEconInput.txtMktAssess) / 100
Dim TaxRate As Double 'Rate at which assessed values are taxed
TaxRate = Val(FrmEconInput.txtTaxRate) / 100
Dim PortionUseValBaseline As Double 'The portion of baseline land that is assessed at use value
PortionUseValBaseline = Val(FrmEconInput.txtPortionUseBL) / 100
'Amount of predevelopment land in various land use types
Dim AgLandArea As typEconTractDataHolder
Dim ForLandArea As typEconTractDataHolder
Dim WaterLandArea As typEconTractDataHolder
Dim DistLandArea As typEconTractDataHolder
Dim MixedLandArea As typEconTractDataHolder
'Baseline use assessment values for individual development squares
Dim IndividUseAssessBL As typEconTractDataHolder
Dim IndividUseAssessBLAg As typEconTractDataHolder
Dim IndividUseAssessBLFor As typEconTractDataHolder
Dim AggUseAssessBL As typEconTractDataHolder
'Aggregate use assessment for new development squares
Dim UseAssessNew As typEconTractDataHolder
'Difference between use assesment for new development vs. baseline
Dim difUseAssessment As typEconTractDataHolder

'Baseline mkt assessment values for individual development squares
Dim IndividMktAssessBL As typEconTractDataHolder
Dim AggMktAssessBL As typEconTractDataHolder

'Aggregate mkt assessment for new development squares
Dim MktAssessNew As typEconTractDataHolder

'Difference between mkt assessment for new development vs. baseline
Dim difMktAssessment As typEconTractDataHolder

Dim difTotAssessedValue As typEconTractDataHolder 'difference in total assessed value (mkt + use)

Dim SewerConstructCostPerFoot As Double
' $25.66 per foot for materials, labor, equipment, excavation, and bedding.
' Cost for a 15" reinforced concrete culvert
' $1545 per manhole. Manholes are laid every 400 feet.
SewerConstructCostPerFoot = 25.66 + (1545 / 400)
Dim SewerConstructCostPerMeter As Double 'convert to meters to work with GIS data
SewerConstructCostPerMeter = SewerConstructCostPerFoot / 0.3048
Dim IndividSewerConnectCost As typEconTractDataHolder 'Connection cost for each tract
Dim AggSewerConnectCost As typEconTractDataHolder 'Aggregate cost for tracts of each type

Dim WaterConstructCostPerFoot As Double
' $31.45 per foot for materials, labor, equipment, excavation, and bedding.
' Cost for a 12" ductile iron pipe
' $2.52 per foot for valve and $3.76 per foot for hydrants.
WaterConstructCostPerFoot = 31.45 + 2.52 + 3.76
Dim WaterConstructCostPerMeter As Double 'convert to meters to work with GIS data
WaterConstructCostPerMeter = WaterConstructCostPerFoot / 0.3048
Dim IndividWaterConnectCost As typEconTractDataHolder 'Connection cost for each tract
Dim AggWaterConnectCost As typEconTractDataHolder 'Aggregate cost for tracts of each type

Dim i As Integer
Dim RecordNumber As Integer
RecordNumber = 0

'Loop thru the set of development squares to calculate the land value for each square
developRecSet.MoveFirst
Do While Not developRecSet.EOF
  'Log of Land Value price per hectare is equal to the sum of 3 components:
  '1. log of partial value price per hectare
  '2. (-0.527739 * log of parcel size in hectares)
  '3. (-0.023488 * log of parcel size squared in hectares)
  'Count which record the loop is on
  RecordNumber = RecordNumber + 1
  'Set up the first component
  RegressPart1 = developRecSet.Fields("LogPVHec").Value
  'The second and third component depend on what type of development square is clicked
  TypeOfDevelop = developRecSet.Fields("DevType").Value

  '...
Select Case TypeOfDevelop
'Note: square feet given by Kurt Stephenson report, based on Schuler (1998) and Arendt (1994). 0.404687 is a factor to convert acres to hectares. The square footages are for 50 dwelling units and 125 people. There are 9 hec in a development square.

Case "LDR"
'For low density, 50 parcels and 125 people fit on 150 acres or 60.70305 hec
NumberOfParcelsPerTract.Low = 50 * 9 / 60.70305
NumberOfPeoplePerTract.Low = 125 * 9 / 60.70305
'3 acres * conversion factor
ParcSizeHec.Low = 3 * 0.404687

'Calculate the total number of low parcels and people
TotNumberOfParcels.Low = TotNumberOfParcels.Low + Round(NumberOfParcelsPerTract.Low, 2)
TotNumberOfPeople.Low = TotNumberOfPeople.Low + Round(NumberOfPeoplePerTract.Low, 2)

'Add low density values to the aggregate
TotNumberOfParcels.Agg = TotNumberOfParcels.Agg + Round(NumberOfParcelsPerTract.Low, 2)
TotNumberOfPeople.Agg = TotNumberOfPeople.Agg + Round(NumberOfPeoplePerTract.Low, 2)

'Log in VB means natural log, so need to convert to base 10 log
RegressPart2 = -0.527739 * (Log(ParcSizeHec.Low) / Log(10))
RegressPart3 = -0.023488 * ((Log(ParcSizeHec.Low) / Log(10)) ^ 2)

'The regression value is a log of price, so take the antilog to get back to dollars.
NewIndividParcelValue.Low = 10 ^ (RegressPart1 + RegressPart2 + RegressPart3)

'Count number of low density tracts
NumberOfTracts.Low = NumberOfTracts.Low + 1

'Get baseline tract value from the database
BaselineTractValue.Low = developRecSet.Fields("BlParcVal").Value

'Get the predevelopment land use areas (square meters)
AgLandArea.Low = developRecSet.Fields("OldHerbAg").Value
ForLandArea.Low = developRecSet.Fields("OldForest").Value
WaterLandArea.Low = developRecSet.Fields("OldWater").Value
DistLandArea.Low = developRecSet.Fields("OldDisturb").Value
MixedLandArea.Low = developRecSet.Fields("OldMixed").Value

'The ag baseline use assessment value for a development square is equal to
'the amount of area in ag, water, and disturbed + half of the mixed
times the ag use assessment rate, the portion of predevelopment land
'that is assessed at use value, and a conversion to hectares
IndividUseAssessBLAg.Low = (AgLandArea.Low + WaterLandArea.Low + _
   DistLandArea.Low + (0.5 * MixedLandArea.Low)) * (0.0001) * _
   UseAssessRateAg * PortionUseValBaseline

'The forest baseline use assessment value for a development square is equal to
'the amount of area in forest + half of the mixed
'times the forest use assessment rate, the portion of predevelopment land
'that is assessed at use value, and a conversion to hectares
IndividUseAssessBLFor.Low = (ForLandArea.Low + (0.5 * MixedLandArea.Low)) _
   * 0.0001 * UseAssessRateFor * PortionUseValBaseline
'Add up the ag and forest baseline use assessments to get total use assessment
IndividUseAssessBL.Low = IndividUseAssessBLAg.Low + IndividUseAssessBLFor.Low

'Calculate the baseline market assessment -- Multiply the market baseline value of
'the tract (note: this includes no house or other development costs -- assuming land
'was bare before development) by the portion that the user wants to assess at market value.
IndividMktAssessBL.Low = (1 - PortionUseValBaseline) * BaselineTractValue.Low

'Calculate the cost of the sewer hook up
IndividSewerConnectCost.Low = developRecSet.Fields("SewDist").Value * SewerConstructCostPerMeter

'Calculate the cost of the Water hook up
IndividWaterConnectCost.Low = developRecSet.Fields("WatDist").Value * WaterConstructCostPerMeter

Case "MDRConv"
'For MidConv density, 50 parcels and 125 people fit on 21 acres or _
8.498427 hec
NumberOfParcelsPerTract.MidConv = 50 * 9 / 8.498427
NumberOfPeoplePerTract.MidConv = 125 * 9 / 8.498427
'1/5 of an acre * conversion to hec
ParcSizeHec.MidConv = (1 / 5) * 0.404687

'Calculate the total number of midConv parcels and people
TotNumberOfParcels.MidConv = TotNumberOfParcels.MidConv +
Round(NumberOfParcelsPerTract.MidConv, 2)
TotNumberOfPeople.MidConv = TotNumberOfPeople.MidConv +
Round(NumberOfPeoplePerTract.MidConv, 2)

'Add midConv density values to the aggregate
TotNumberOfParcels.Agg = TotNumberOfParcels.Agg +
Round(NumberOfParcelsPerTract.MidConv, 2)
TotNumberOfPeople.Agg = TotNumberOfPeople.Agg +
Round(NumberOfPeoplePerTract.MidConv, 2)

'Log in VB means natural log, so need to convert to base 10 log
RegressPart2 = -0.527739 * (Log(ParcSizeHec.MidConv) / Log(10))
RegressPart3 = -0.023488 * ((Log(ParcSizeHec.MidConv) / Log(10)) ^ 2)

'The regression value is a log of price, so take the antilog to get back to dollars.
NewIndividParcelValue.MidConv = 10 ^ (RegressPart1 + RegressPart2 +
RegressPart3)

'Count number of midConv tracts
NumberOfTracts.MidConv = NumberOfTracts.MidConv + 1

'Get baseline tract value from the database
BaselineTractValue.MidConv = developRecSet.Fields("BlParcVal").Value

'Get the predevelopment land use areas (square meters)
AgLandArea.MidConv = developRecSet.Fields("OldHerbAg").Value
ForLandArea.MidConv = developRecSet.Fields("OldForest").Value
WaterLandArea.MidConv = developRecSet.Fields("OldWater").Value
DistLandArea.MidConv = developRecSet.Fields("OldDisturb").Value
MixedLandArea.MidConv = developRecSet.Fields("OldMixed").Value
The ag baseline use assessment value for a development square is equal to the amount of area in ag, water, and disturbed + half of the mixed times the ag use assessment rate, the portion of predevelopment land that is assessed at use value, and a conversion to hectares

\[
\text{IndividUseAssessBLAg.MidConv} = (\text{AgLandArea.MidConv} + \text{WaterLandArea.MidConv} + \text{DistLandArea.MidConv} + (0.5 \times \text{MixedLandArea.MidConv})) \times (0.0001) \times \text{UseAssessRateAg} \times \text{PortionUseValBaseline}
\]

The forest baseline use assessment value for a development square is equal to the amount of area in forest + half of the mixed times the forest use assessment rate, the portion of predevelopment land that is assessed at use value, and a conversion to hectares

\[
\text{IndividUseAssessBLFor.MidConv} = (\text{ForLandArea.MidConv} + (0.5 \times \text{MixedLandArea.MidConv})) \times 0.0001 \times \text{UseAssessRateFor} \times \text{PortionUseValBaseline}
\]

Add up the ag and forest baseline use assessments to get total use assessment

\[
\text{IndividUseAssessBL.MidConv} = \text{IndividUseAssessBLAg.MidConv} + \text{IndividUseAssessBLFor.MidConv}
\]

Calculate the baseline market assessment -- Multiply the market baseline value of the tract (note: this includes no house or other development costs -- assuming land was bare before development) by the portion that the user wants to assess at market value.

\[
\text{IndividMktAssessBL.MidConv} = (1 - \text{PortionUseValBaseline}) \times \text{BaselineTractValue.MidConv}
\]

Calculate the cost of the sewer hook up

\[
\text{IndividSewerConnectCost.MidConv} = \text{developRecSet.Fields("SewDist").Value} \times \text{SewerConstructCostPerMeter}
\]

Calculate the cost of the Water hook up

\[
\text{IndividWaterConnectCost.MidConv} = \text{developRecSet.Fields("WatDist").Value} \times \text{WaterConstructCostPerMeter}
\]

Case "MDRClus"

For MidClus density, 50 parcels and 125 people fit on 21 acres or 8.498427 hec

\[
\begin{align*}
\text{NumberOfParcelsPerTract.MidClus} & = 50 \times 9 / 8.498427 \\
\text{NumberOfPeoplePerTract.MidClus} & = 125 \times 9 / 8.498427 \\
\end{align*}
\]

'1/7 of an acre * conversion to hec 

\[
\text{ParcSizeHec.MidClus} = (1 / 7) \times 0.404687
\]

Calculate the total number of MidClus parcels and people

\[
\begin{align*}
\text{TotNumberOfParcels.MidClus} & = \text{TotNumberOfParcels.MidClus} + \text{Round(NumberOfParcelsPerTract.MidClus, 2)} \\
\text{TotNumberOfPeople.MidClus} & = \text{TotNumberOfPeople.MidClus} + \text{Round(NumberOfPeoplePerTract.MidClus, 2)} \\
\end{align*}
\]

Add midClus density values to the aggregate

\[
\begin{align*}
\text{TotNumberOfParcels.Agg} & = \text{TotNumberOfParcels.Agg} + \text{Round(NumberOfParcelsPerTract.MidClus, 2)} \\
\text{TotNumberOfPeople.Agg} & = \text{TotNumberOfPeople.Agg} + \text{Round(NumberOfPeoplePerTract.MidClus, 2)} \\
\end{align*}
\]

Log in VB means natural log, so need to convert to base 10 log

\[
\begin{align*}
\text{RegressPart2} & = -0.527739 \times (\log(\text{ParcSizeHec.MidClus}) / \log(10)) \\
\text{RegressPart3} & = -0.023488 \times ((\log(\text{ParcSizeHec.MidClus}) / \log(10))^2)
\end{align*}
\]
'The regression value is a log of price, so take the antilog to get back to dollars.

\[ \text{NewIndividParcelValue.MidClus} = 10^{(\text{RegressPart1} + \text{RegressPart2} + \text{RegressPart3})} \]

'Count number of midClus tracts
\[ \text{NumberOfTracts.MidClus} = \text{NumberOfTracts.MidClus} + 1 \]

'Get baseline tract value from the database
\[ \text{BaselineTractValue.MidClus} = \text{developRecSet.Fields("BlParcVal").Value} \]

'Get the predevelopment land use areas (square meters)
\[ \begin{align*}
\text{AgLandArea.MidClus} &= \text{developRecSet.Fields("OldHerbAg").Value} \\
\text{ForLandArea.MidClus} &= \text{developRecSet.Fields("OldForest").Value} \\
\text{WaterLandArea.MidClus} &= \text{developRecSet.Fields("OldWater").Value} \\
\text{DistLandArea.MidClus} &= \text{developRecSet.Fields("OldDisturb").Value} \\
\text{MixedLandArea.MidClus} &= \text{developRecSet.Fields("OldMixed").Value}
\end{align*} \]

'The ag baseline use assessment value for a development square is equal to
'the amount of area in ag, water, and disturbed + half of the mixed
'times the ag use assessment rate, the portion of predevelopment land
'that is assessed at use value, and a conversion to hectares
\[ \text{IndividUseAssessBLAg.MidClus} = (\text{AgLandArea.MidClus} + \text{WaterLandArea.MidClus} + \text{DistLandArea.MidClus} + (0.5 \times \text{MixedLandArea.MidClus})) \times (0.0001) \times (\text{UseAssessRateAg} \times \text{PortionUseValBaseline}) \]

'The forest baseline use assessment value for a development square is equal to
'the amount of area in forest + half of the mixed
'times the forest use assessment rate, the portion of predevelopment land
'that is assessed at use value, and a conversion to hectares
\[ \text{IndividUseAssessBLFor.MidClus} = (\text{ForLandArea.MidClus} + (0.5 \times \text{MixedLandArea.MidClus})) \times 0.0001 \times \text{UseAssessRateFor} \times \text{PortionUseValBaseline} \]

'Add up the ag and forest baseline use assessments to get total use assessment
\[ \text{IndividUseAssessBL.MidClus} = \text{IndividUseAssessBLAg.MidClus} + \text{IndividUseAssessBLFor.MidClus} \]

'Calculate the baseline market assessment -- Multiply the market baseline value of
'the tract (note: this includes no house or other development costs --
'assuming land
'was bare before development) by the portion that the user wants to assess at market value.
\[ \text{IndividMktAssessBL.MidClus} = (1 - \text{PortionUseValBaseline}) \times \text{BaselineTractValue.MidClus} \]

'Calculate the cost of the sewer hook up
\[ \text{IndividSewerConnectCost.MidClus} = \text{developRecSet.Fields("SewDist").Value} \times \text{SewerConstructCostPerMeter} \]

'Calculate the cost of the Water hook up
\[ \text{IndividWaterConnectCost.MidClus} = \text{developRecSet.Fields("WatDist").Value} \times \text{WaterConstructCostPerMeter} \]

Case "HDR"
'For high density, 50 parcels and 125 people fit on 12 acres or 4.856244 hec
\[ \begin{align*}
\text{NumberOfParcelsPerTract.High} &= 50 \times 9 / 4.856244 \\
\text{NumberOfPeoplePerTract.High} &= 125 \times 9 / 4.856244
\end{align*} \]
'1/7 of an acre * conversion to hec
ParcSizeHec.High = (1 / 7) * 0.404687

'Calculate the total number of High parcels and people
TotNumberOfParcels.High = TotNumberOfParcels.High +
Round(NumberOfParcelsPerTract.High, 2)
TotNumberOfPeople.High = TotNumberOfPeople.High +
Round(NumberOfPeoplePerTract.High, 2)

'Add high density values to the aggregate
TotNumberOfParcels.Agg = TotNumberOfParcels.Agg +
Round(NumberOfParcelsPerTract.High, 2)
TotNumberOfPeople.Agg = TotNumberOfPeople.Agg +
Round(NumberOfPeoplePerTract.High, 2)

'Log in VB means natural log, so need to convert to base 10 log
RegressPart2 = -0.527739 * (Log(ParcSizeHec.High) / Log(10))
RegressPart3 = -0.023488 * ((Log(ParcSizeHec.High) / Log(10)) ^ 2)

'The regression value is a log of price, so take the antilog to get back to
dollars.
NewIndividParcelValue.High = 10 ^ (RegressPart1 + RegressPart2 +
RegressPart3)

'Count number of high tracts
NumberOfTracts.High = NumberOfTracts.High + 1

'Get baseline tract value from the database
BaselineTractValue.High = developRecSet.Fields("BlParcVal").Value

'Get the predevelopment land use areas (square meters)
AgLandArea.High = developRecSet.Fields("OldHerbAg").Value
ForLandArea.High = developRecSet.Fields("OldForest").Value
WaterLandArea.High = developRecSet.Fields("OldWater").Value
DistLandArea.High = developRecSet.Fields("OldDisturb").Value
MixedLandArea.High = developRecSet.Fields("OldMixed").Value

'The ag baseline use assessment value for a development square is equal to
'the amount of area in ag, water, and disturbed + half of the mixed
'times the ag use assessment rate, the portion of predevelopment land
'that is assessed at use value, and a conversion to hectares
IndividUseAssessBLAg.High = (AgLandArea.High + WaterLandArea.High + 
  DistLandArea.High + (0.5 * MixedLandArea.High)) * (0.0001) * _
  UseAssessRateAg * PortionUseValBaseline

'The forest baseline use assessment value for a development square is equal to
'the amount of area in forest + half of the mixed
'times the forest use assessment rate, the portion of predevelopment land
'that is assessed at use value, and a conversion to hectares
IndividUseAssessBLFor.High = (ForLandArea.High + (0.5 * MixedLandArea.High))
  * 0.0001 * UseAssessRateFor * PortionUseValBaseline

'Add up the ag and forest baseline use assessments to get total use
assessment
IndividUseAssessBL.High = IndividUseAssessBLAg.High +
IndividUseAssessBLFor.High

'Calculate the baseline market assessment -- Multiply the market baseline
value of
'the tract (note: this includes no house or other development costs --
assuming land
'was bare before development) by the portion that the user wants to assess at market value.
IndividMktAssessBL.High = (1 - PortionUseValBaseline) * BaselineTractValue.High

'Calculate the cost of the sewer hook up
IndividSewerConnectCost.High = developRecSet.Fields("SewDist").Value * SewerConstructCostPerMeter

'Calculate the cost of the Water hook up
IndividWaterConnectCost.High = developRecSet.Fields("WatDist").Value * WaterConstructCostPerMeter

Case "COM"
'For commercial, the number and size of parcels is assumed to be the same as high density, and the number of people is assumed to be 0.
NumberOfParcelsPerTract.Comm = 50 * 9 / 4.856244
NumberOfPeoplePerTract.Comm = 0
'1/7 of an acre * conversion to hec
ParcSizeHec.Comm = (1 / 7) * 0.404687

'Calculate the total number of commercial parcels and people
TotNumberOfParcels.Comm = TotNumberOfParcels.Comm + Round(NumberOfParcelsPerTract.Comm, 2)
TotNumberOfPeople.Comm = TotNumberOfPeople.Comm + Round(NumberOfPeoplePerTract.Comm, 2)

'Add commercial values to the aggregate
TotNumberOfParcels.Agg = TotNumberOfParcels.Agg + Round(NumberOfParcelsPerTract.Comm, 2)
TotNumberOfPeople.Agg = TotNumberOfPeople.Agg + Round(NumberOfPeoplePerTract.Comm, 2)

'Log in VB means natural log, so need to convert to base 10 log
RegressPart2 = -0.527739 * (Log(ParcSizeHec.Comm) / Log(10))
RegressPart3 = -0.023488 * ((Log(ParcSizeHec.Comm) / Log(10)) ^ 2)

'The regression value is a log of price, so take the antilog to get back to dollars.
NewIndividParcelValue.Comm = 10 ^ (RegressPart1 + RegressPart2 + RegressPart3)

'Count number of midConv tracts
NumberOfTracts.Comm = NumberOfTracts.Comm + 1

'Get baseline tract value from the database
BaselineTractValue.Comm = developRecSet.Fields("BlParcVal").Value

'Get the predevelopment land use areas (square meters)
AgLandArea.Comm = developRecSet.Fields("OldHerbAg").Value
ForLandArea.Comm = developRecSet.Fields("OldForest").Value
WaterLandArea.Comm = developRecSet.Fields("OldWater").Value
DistLandArea.Comm = developRecSet.Fields("OldDisturb").Value
MixedLandArea.Comm = developRecSet.Fields("OldMixed").Value

'The ag baseline use assessment value for a development square is equal to the amount of area in ag, water, and disturbed + half of the mixed times the ag use assessment rate, the portion of predevelopment land that is assessed at use value, and a conversion to hectares
IndividUseAssessBLAg.Comm = (AgLandArea.Comm + WaterLandArea.Comm + _
    DistLandArea.Comm + (0.5 * MixedLandArea.Comm)) * (0.0001) * _
    UseAssessRateAg * PortionUseValBaseline
The forest baseline use assessment value for a development square is equal to
the amount of area in forest + half of the mixed
times the forest use assessment rate, the portion of predevelopment land
that is assessed at use value, and a conversion to hectares
\[
\text{IndividUseAssessBLFor.Comm} = (\text{ForLandArea.Comm} + (0.5 \times \text{MixedLandArea.Comm})) \times 0.0001 \times \text{UseAssessRateFor} \times \text{PortionUseValBaseline}
\]

Add up the ag and forest baseline use assessments to get total use assessment
\[
\text{IndividUseAssessBL.Comm} = \text{IndividUseAssessBLAg.Comm} + \text{IndividUseAssessBLFor.Comm}
\]

Calculate the baseline market assessment -- Multiply the market baseline value of the tract (note: this includes no house or other development costs -- assuming land was bare before development) by the portion that the user wants to assess at market value.
\[
\text{IndividMktAssessBL.Comm} = (1 - \text{PortionUseValBaseline}) \times \text{BaselineTractValue.Comm}
\]

Calculate the cost of the sewer hook up
\[
\text{IndividSewerConnectCost.Comm} = \text{developRecSet.Fields("SewDist").Value} \times \text{SewerConstructCostPerMeter}
\]

Calculate the cost of the Water hook up
\[
\text{IndividWaterConnectCost.Comm} = \text{developRecSet.Fields("WatDist").Value} \times \text{WaterConstructCostPerMeter}
\]

End Select

Add up the total value of all parcels of each kind of tract
\[
\text{NewAggParcelValue.Low} = \text{NewAggParcelValue.Low} + (\text{NewIndividParcelValue.Low} \times \text{NumberOfParcelsPerTract.Low})
\]
\[
\text{NewAggParcelValue.MidConv} = \text{NewAggParcelValue.MidConv} + (\text{NewIndividParcelValue.MidConv} \times \text{NumberOfParcelsPerTract.MidConv})
\]
\[
\text{NewAggParcelValue.MidClus} = \text{NewAggParcelValue.MidClus} + (\text{NewIndividParcelValue.MidClus} \times \text{NumberOfParcelsPerTract.MidClus})
\]
\[
\text{NewAggParcelValue.High} = \text{NewAggParcelValue.High} + (\text{NewIndividParcelValue.High} \times \text{NumberOfParcelsPerTract.High})
\]
\[
\]

Add up the total baseline value of all tracts of each type
\[
\text{AggBaselineTractValue.Low} = \text{AggBaselineTractValue.Low} + \text{BaselineTractValue.Low}
\]
\[
\text{AggBaselineTractValue.MidConv} = \text{AggBaselineTractValue.MidConv} + \text{BaselineTractValue.MidConv}
\]
\[
\text{AggBaselineTractValue.MidClus} = \text{AggBaselineTractValue.MidClus} + \text{BaselineTractValue.MidClus}
\]
\[
\text{AggBaselineTractValue.High} = \text{AggBaselineTractValue.High} + \text{BaselineTractValue.High}
\]
\[
\text{AggBaselineTractValue.Comm} = \text{AggBaselineTractValue.Comm} + \text{BaselineTractValue.Comm}
\]

Add up the total baseline use value assessments
\[
\text{AggUseAssessBLLow} = \text{AggUseAssessBLLow} + \text{IndividUseAssessBLLow}
\]
\[
\text{AggUseAssessBLMidConv} = \text{AggUseAssessBLMidConv} + \text{IndividUseAssessBLMidConv}
\]
\[
\text{AggUseAssessBLMidClus} = \text{AggUseAssessBLMidClus} + \text{IndividUseAssessBLMidClus}
\]
\[
\text{AggUseAssessBLHigh} = \text{AggUseAssessBLHigh} + \text{IndividUseAssessBLHigh}
\]
\[
\text{AggUseAssessBLComm} = \text{AggUseAssessBLComm} + \text{IndividUseAssessBLComm}
\]

*Reset inividuseassessBL to 0*
IndividUseAssessBL.Low = 0
IndividUseAssessBL.MidConv = 0
IndividUseAssessBL.MidClus = 0
IndividUseAssessBL.High = 0
IndividUseAssessBL.Comm = 0

'Add up the total baseline mkt value assessments
AggMktAssessBL.Low = AggMktAssessBL.Low + IndividMktAssessBL.Low
AggMktAssessBL.MidConv = AggMktAssessBL.MidConv + IndividMktAssessBL.MidConv
AggMktAssessBL.MidClus = AggMktAssessBL.MidClus + IndividMktAssessBL.MidClus
AggMktAssessBL.High = AggMktAssessBL.High + IndividMktAssessBL.High
AggMktAssessBL.Comm = AggMktAssessBL.Comm + IndividMktAssessBL.Comm

'Reset individMktassessBL to 0
IndividMktAssessBL.Low = 0
IndividMktAssessBL.MidConv = 0
IndividMktAssessBL.MidClus = 0
IndividMktAssessBL.High = 0
IndividMktAssessBL.Comm = 0

'Add up the sewer connection costs of all tracts of each type
AggSewerConnectCost.Low = AggSewerConnectCost.Low + IndividSewerConnectCost.Low
AggSewerConnectCost.MidConv = AggSewerConnectCost.MidConv + IndividSewerConnectCost.MidConv +
IndividSewerConnectCost.MidClus = AggSewerConnectCost.MidClus +
IndividSewerConnectCost.MidClus = AggSewerConnectCost.MidClus +
IndividSewerConnectCost.High = AggSewerConnectCost.High +
IndividSewerConnectCost.High = AggSewerConnectCost.High +
IndividSewerConnectCost.Comm = AggSewerConnectCost.Comm +
IndividSewerConnectCost.Comm

'Reset IndividSewerConnectCost to 0
IndividSewerConnectCost.Low = 0
IndividSewerConnectCost.MidConv = 0
IndividSewerConnectCost.MidClus = 0
IndividSewerConnectCost.High = 0
IndividSewerConnectCost.Comm = 0

'Add up the Water connection costs of all tracts of each type
AggWaterConnectCost.Low = AggWaterConnectCost.Low + IndividWaterConnectCost.Low
AggWaterConnectCost.MidConv = AggWaterConnectCost.MidConv + IndividWaterConnectCost.MidConv +
IndividWaterConnectCost.MidClus = AggWaterConnectCost.MidClus +
IndividWaterConnectCost.MidClus = AggWaterConnectCost.MidClus +
IndividWaterConnectCost.High = AggWaterConnectCost.High +
IndividWaterConnectCost.High = AggWaterConnectCost.High +
IndividWaterConnectCost.Comm

'Reset IndividWaterConnectCost to 0
IndividWaterConnectCost.Low = 0
IndividWaterConnectCost.MidConv = 0
IndividWaterConnectCost.MidClus = 0
IndividWaterConnectCost.High = 0
IndividWaterConnectCost.Comm = 0

developRecSet.MoveNext
Loop

'Count total number of tracts
NumberOfTracts.Agg = RecordNumber

'Find the change in bare land value (value of new development - baseline value)
difBareLandValue.Low = NewAggParcelValue.Low - AggBaselineTractValue.Low
difBareLandValue.MidConv = NewAggParcelValue.MidConv - AggBaselineTractValue.MidConv
difBareLandValue.MidClus = NewAggParcelValue.MidClus - AggBaselineTractValue.MidClus
difBareLandValue.High = NewAggParcelValue.High - AggBaselineTractValue.High

'Add up the cost of all houses for each type of tract - no commercial info
AggHouseCost.Low = IndividHouseCost.Low * TotNumberOfParcels.Low
AggHouseCost.MidConv = IndividHouseCost.MidConv * TotNumberOfParcels.MidConv
AggHouseCost.MidClus = IndividHouseCost.MidClus * TotNumberOfParcels.MidClus
AggHouseCost.High = IndividHouseCost.High * TotNumberOfParcels.High

'Add up the cost of all roads, sidewalks, and storm drains for each type of tract -- no commercial info
AggRoadCost.Low = IndividRoadCost.Low * TotNumberOfParcels.Low
AggRoadCost.MidConv = IndividRoadCost.MidConv * TotNumberOfParcels.MidConv
AggRoadCost.MidClus = IndividRoadCost.MidClus * TotNumberOfParcels.MidClus
AggRoadCost.High = IndividRoadCost.High * TotNumberOfParcels.High

'Add up the cost of all inside-tract sewer for each type of tract - no commercial info
AggSewerCost.Low = IndividSewerCost.Low * TotNumberOfParcels.Low
AggSewerCost.MidConv = IndividSewerCost.MidConv * TotNumberOfParcels.MidConv
AggSewerCost.MidClus = IndividSewerCost.MidClus * TotNumberOfParcels.MidClus
AggSewerCost.High = IndividSewerCost.High * TotNumberOfParcels.High

'Add up all tract development costs
TotDevelopmentCost.MidConv = AggHouseCost.MidConv + AggRoadCost.MidConv + AggSewerCost.MidConv + AggWaterCost.MidConv

'Totall up change in tract cost
TotTractCost.Low = difBareLandValue.Low + TotDevelopmentCost.Low
TotTractCost.MidConv = difBareLandValue.MidConv + TotDevelopmentCost.MidConv
TotTractCost.MidClus = difBareLandValue.MidClus + TotDevelopmentCost.MidClus
TotTractCost.High = difBareLandValue.High + TotDevelopmentCost.High

'Calculate the use value assessment for new development squares
equal to fraction of openspace * tract area * assessment rate
UseAssessNew.Low = OpenSpaceFraction.Low * TractSizeHec * NumberOfTracts.Low * UseAssessRateOS
UseAssessNew.MidConv = OpenSpaceFraction.MidConv * TractSizeHec * NumberOfTracts.MidConv * UseAssessRateOS
UseAssessNew.MidClus = OpenSpaceFraction.MidClus * TractSizeHec * NumberOfTracts.MidClus * UseAssessRateOS
UseAssessNew.High = OpenSpaceFraction.High * TractSizeHec * NumberOfTracts.High * UseAssessRateOS
UseAssessNew.Comm = OpenSpaceFraction.Comm * TractSizeHec * NumberOfTracts.Comm * UseAssessRateOS

'Calculate the change in use value assessment from the baseline to new development
difUseAssessment.Low = UseAssessNew.Low - AggUseAssessBL.Low
difUseAssessment.MidConv = UseAssessNew.MidConv - AggUseAssessBL.MidConv
difUseAssessment.MidClus = UseAssessNew.MidClus - AggUseAssessBL.MidClus
difUseAssessment.High = UseAssessNew.High - AggUseAssessBL.High

'Calculate the mkt value assessment for new development squares
'equal to total tract value times the % of the tract that is private space times _
'the mkt assessment rate
MktAssessNew.Low = PrivateSpaceFraction.Low * TotTractCost.Low * MktAssessRate
MktAssessNew.MidConv = PrivateSpaceFraction.MidConv * TotTractCost.MidConv * MktAssessRate
MktAssessNew.MidClus = PrivateSpaceFraction.MidClus * TotTractCost.MidClus * MktAssessRate

'diffMktAssessment.Low = MktAssessNew.Low - AggMktAssessBL.Low
MktAssessNew.MidConv = MktAssessNew.MidConv - AggMktAssessBL.MidConv
MktAssessNew.MidClus = MktAssessNew.MidClus - AggMktAssessBL.MidClus
MktAssessNew.High = MktAssessNew.High - AggMktAssessBL.High

'diffTotAssessedValue.Low = difUseAssessment.Low + difMktAssessment.Low
MktAssessNew.MidConv = MktAssessNew.MidConv + difMktAssessment.MidConv
MktAssessNew.MidClus = MktAssessNew.MidClus + difMktAssessment.MidClus

DatConnect.Disconnect

*****************************************************************************

'This part populates the spreadsheet

'Declare object variables for Microsoft Excel, _
'application workbook, and worksheet objects.
Dim xlApp As Excel.Application
Dim xlBook As Excel.Workbook
Dim xlSheet As Excel.Worksheet

'Assign object references to the variables. Use _
'Add methods to create new workbook and worksheet objects.
Set xlApp = New Excel.Application
Set xlBook = xlApp.Workbooks.Add
Set xlSheet = xlBook.Worksheets.Add

'This with block puts all the textual items into the spreadsheet
With xlSheet
'Make font Times New Roman
.Cells.Font.Name = "Times New Roman"
.Cells.Font.Size = 10

'Put a table name in the first Excel cell.
.Cells(1, 1).Value = "Tax Revenue and Fiscal Costs (" & App.Path & 
"\OutputFiles\GovtRevsExps.xls)"
.Cells(1, 1).Font.Bold = True
.Cells(1, 1).Font.Size = 14
.Range("A1:F1").Merge
.Range("A1:F1").HorizontalAlignment = xlCenter

'Set the column widths of the spreadsheet
.Columns("A:A").ColumnWidth = 36
.Columns("B:F").ColumnWidth = 17
'Add and format row and column headings
<table>
<thead>
<tr>
<th>Land reserved for open space (hectares)</th>
<th>Land occupied by housing/commer. lots (hectares)</th>
<th>Land occupied by infrastructure (hectares)</th>
<th>Total land (hectares)</th>
<th>Total number of housing/commercial lots</th>
<th>Total number of people living on the lots</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Development Land Uses</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Land reserved for open space (hectares)&quot;</td>
</tr>
<tr>
<td>&quot;Land occupied by housing/commer. lots (hectares)&quot;</td>
</tr>
<tr>
<td>&quot;Land occupied by infrastructure (hectares)&quot;</td>
</tr>
<tr>
<td>&quot;Total land (hectares)&quot;</td>
</tr>
<tr>
<td>&quot;Total number of housing/commercial lots&quot;</td>
</tr>
<tr>
<td>&quot;Total number of people living on the lots&quot;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Development Tract Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Low Density&quot;</td>
</tr>
<tr>
<td>&quot;Mid Standard&quot;</td>
</tr>
<tr>
<td>&quot;Mid Cluster&quot;</td>
</tr>
<tr>
<td>&quot;High Density&quot;</td>
</tr>
<tr>
<td>&quot;Commercial&quot;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dollar Change Relative to Predevelopment Baseline</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Development cost&quot;</td>
</tr>
<tr>
<td>1. New house construction cost</td>
</tr>
<tr>
<td>2. Road/sidewalk/drain cost (within tract)</td>
</tr>
<tr>
<td>3. Sewer cost (within tract)</td>
</tr>
<tr>
<td>4. Water cost (within tract)</td>
</tr>
<tr>
<td>B. Total tract development cost (1+2+3+4)</td>
</tr>
<tr>
<td>C. Estimated total tract value (A+B)</td>
</tr>
<tr>
<td>Assessed value</td>
</tr>
<tr>
<td>3. Use value assessment</td>
</tr>
<tr>
<td>4. Market value assessment</td>
</tr>
<tr>
<td>D. Total value of assessed property (3+4)</td>
</tr>
<tr>
<td>E. Tax revenue (D*Tax Rate)</td>
</tr>
<tr>
<td>Cost to local government</td>
</tr>
<tr>
<td>1. Total Sewer connection cost</td>
</tr>
<tr>
<td>2. Total Water connection cost</td>
</tr>
<tr>
<td>Total sewer and water cost to localities (1+2)</td>
</tr>
<tr>
<td>Annualized sewer and water cost to localities</td>
</tr>
</tbody>
</table>

143
.Cells(37, 1).Font.Bold = True

'Set decimal places for cells
.Range("B4:F9").Style = "Comma"
.Range("B4:F9").NumberFormat = "_(* #,##0.00_);_(* (#,##0.00);_(* ""??_);_(@_"
.Range("B11:F36").Style = "Comma"
.Range("B11:F36").NumberFormat = "_(* #,##0_);_(* #00);_(* ""??_);_(@_"

'Set Cell Alignment
.Range("F4:F36").HorizontalAlignment = xlRight
End With

Dim row As Integer
row = 4
'This with block puts the data into the spreadsheet
With xlSheet
  'Calculate open space for each kind of tract
  .Cells(row, 2).Value = OpenSpaceFraction.Low * NumberOfTracts.Low * TractSizeHec
  .Cells(row, 3).Value = OpenSpaceFraction.MidConv * NumberOfTracts.MidConv * TractSizeHec
  .Cells(row, 4).Value = OpenSpaceFraction.MidClus * NumberOfTracts.MidClus * TractSizeHec
  .Cells(row, 5).Value = OpenSpaceFraction.High * NumberOfTracts.High * TractSizeHec
  row = row + 1

  'Calculate land occupied by privately owned parcels for each kind of tract
  .Cells(row, 2).Value = PrivateSpaceFraction.Low * NumberOfTracts.Low * TractSizeHec
  .Cells(row, 3).Value = PrivateSpaceFraction.MidConv * NumberOfTracts.MidConv * TractSizeHec
  .Cells(row, 4).Value = PrivateSpaceFraction.MidClus * NumberOfTracts.MidClus * TractSizeHec
  .Cells(row, 5).Value = PrivateSpaceFraction.High * NumberOfTracts.High * TractSizeHec
  row = row + 1

  'Calculate infrastructure space for each kind of tract
  .Cells(row, 2).Value = InfrastructureFraction.Low * NumberOfTracts.Low * TractSizeHec
  .Cells(row, 3).Value = InfrastructureFraction.MidConv * NumberOfTracts.MidConv * TractSizeHec
  .Cells(row, 4).Value = InfrastructureFraction.MidClus * NumberOfTracts.MidClus * TractSizeHec
  .Cells(row, 5).Value = InfrastructureFraction.High * NumberOfTracts.High * TractSizeHec
  row = row + 1

  'Sum land in open space, land in parcels, and infra. land for each kind of tract
  .Cells(row, 2).Formula = "=B4+B5+B6"
  .Cells(row, 3).Formula = "=C4+C5+C6"
  .Cells(row, 4).Formula = "=D4+D5+D6"
  .Cells(row, 5).Formula = "=E4+E5+E6"
  .Cells(row, 6).Formula = "=F4+F5+F6"
  row = row + 1
End With
'Insert number of housing/commercial lots
.Cells(row, 2).Value = TotNumberOfParcels.Low
.Cells(row, 3).Value = TotNumberOfParcels.MidConv
.Cells(row, 4).Value = TotNumberOfParcels.MidClus
.Cells(row, 5).Value = TotNumberOfParcels.High
.Cells(row, 6).Value = TotNumberOfParcels.Comm
row = row + 1

'Insert number of people
.Cells(row, 2).Value = TotNumberOfPeople.Low
.Cells(row, 3).Value = TotNumberOfPeople.MidConv
.Cells(row, 4).Value = TotNumberOfPeople.MidClus
.Cells(row, 5).Value = TotNumberOfPeople.High
.Cells(row, 6).Value = TotNumberOfPeople.Comm
row = row + 2

'Insert bare land values (Estimated new land values - estimated baseline values)
.Cells(row, 2).Value = difBareLandValue.Low
.Cells(row, 3).Value = difBareLandValue.MidConv
.Cells(row, 4).Value = difBareLandValue.MidClus
.Cells(row, 5).Value = difBareLandValue.High
.Cells(row, 6).Value = difBareLandValue.Comm
row = row + 3

'Insert house construction cost
.Cells(row, 2).Value = AggHouseCost.Low
.Cells(row, 3).Value = AggHouseCost.MidConv
.Cells(row, 6).Value = "N/A" 'no info on commercial buildings
row = row + 1

'Insert Road/sidewalk/storm drain cost
.Cells(row, 2).Value = AggRoadCost.Low
.Cells(row, 3).Value = AggRoadCost.MidConv
.Cells(row, 4).Value = AggRoadCost.MidClus
.Cells(row, 5).Value = AggRoadCost.High
.Cells(row, 6).Value = "N/A" 'no info on commercial
row = row + 1

'Insert sewer pipe cost
.Cells(row, 2).Value = AggSewerCost.Low
.Cells(row, 3).Value = AggSewerCost.MidConv
.Cells(row, 4).Value = AggSewerCost.MidClus
.Cells(row, 5).Value = AggSewerCost.High
.Cells(row, 6).Value = "N/A" 'no info on commercial
row = row + 1

'Insert water pipe cost
.Cells(row, 2).Value = AggWaterCost.Low
.Cells(row, 3).Value = AggWaterCost.MidConv
.Cells(row, 6).Value = "N/A" 'no info on commercial
row = row + 2

'Sum up inside-the-tract development costs
.Cells(row, 2).Formula = "=B14+B15+B16+B17"
.Cells(row, 3).Formula = "=C14+C15+C16+C17"
.Cells(row, 4).Formula = "=D14+D15+D16+D17"
.Cells(row, 5).Formula = "=E14+E15+E16+E17"
.Cells(row, 6).Value = "N/A" 'no commercial information
row = row + 2

'Sum up the total tract value
.Cells(row, 2).Formula = "=B11+B19"
.Cells(row, 3).Formula = "=C11+C19"
.Cells(row, 4).Formula = "=D11+D19"
.Cells(row, 5).Formula = "=E11+E19"
.Cells(row, 6).Value = "N/A" 'no commercial information
row = row + 3

'Insert the use value assessment
.Cells(row, 2).Value = difUseAssessment.Low
.Cells(row, 3).Value = difUseAssessment.MidConv
.Cells(row, 4).Value = difUseAssessment.MidClus
.Cells(row, 5).Value = difUseAssessment.High
.Cells(row, 6).Value = "N/A" 'no commercial information
row = row + 1

'Insert the mkt value assessment
.Cells(row, 2).Value = difMktAssessment.Low
.Cells(row, 3).Value = difMktAssessment.MidConv
.Cells(row, 4).Value = difMktAssessment.MidClus
.Cells(row, 5).Value = difMktAssessment.High
.Cells(row, 6).Value = "N/A" 'no commercial information\nrow = row + 2

'Insert the total value of assessed property
.Cells(row, 2).Value = difTotAssessedValue.Low
.Cells(row, 3).Value = difTotAssessedValue.MidConv
.Cells(row, 4).Value = difTotAssessedValue.MidClus
.Cells(row, 5).Value = difTotAssessedValue.High
.Cells(row, 6).Value = "N/A" 'no commercial information
row = row + 2

'Insert the Tax revenue
.Cells(row, 2).Value = difTotAssessedValue.Low * TaxRate
.Cells(row, 3).Value = difTotAssessedValue.MidConv * TaxRate
.Cells(row, 4).Value = difTotAssessedValue.MidClus * TaxRate
.Cells(row, 5).Value = difTotAssessedValue.High * TaxRate
.Cells(row, 6).Value = "N/A" 'no commercial information
row = row + 3

'Insert the Sewer Connection cost
.Cells(row, 2).Value = AggSewerConnectCost.Low
.Cells(row, 3).Value = AggSewerConnectCost.MidConv
.Cells(row, 4).Value = AggSewerConnectCost.MidClus
.Cells(row, 5).Value = AggSewerConnectCost.High
row = row + 1

'Insert the Water Connection cost
.Cells(row, 2).Value = AggWaterConnectCost.Low
.Cells(row, 3).Value = AggWaterConnectCost.MidConv
row = row + 2

'Insert the Total of Sewer and Water combined
.Cells(row, 2).Formula = "=B32+B33"
.Cells(row, 3).Formula = "=C32+C33"
.Cells(row, 4).Formula = "=D32+D33"
.Cells(row, 5).Formula = "=E32+E33"
Annualize the total sewer and water costs at a 7% rate over 30 years.
.Cells(row, 2).Formula = "=-PMT(0.07,30,B32+B33)"
.Cells(row, 3).Formula = "=-PMT(0.07,30,C32+C33)"
.Cells(row, 4).Formula = "=-PMT(0.07,30,D32+D33)"
.Cells(row, 5).Formula = "=-PMT(0.07,30,E32+E33)"
.Cells(row, 6).Formula = "=-PMT(0.07,30,F32+F33)"
End With

End Sub

'This subroutine calculates and stores the distance from a neighborhood to a sewer
'Note: Current sewer line layer is for test purposes only -- it is not a real layer
Public Sub CalcSewerDist()

Dim DatConn As New MapObjects2.DataConnection
DatConn.Database = App.Path & "\BC_Data"

' These are variables for the development squares data
Dim develGD As MapObjects2.GeoDataset
Set develGD = DatConn.FindGeoDataset(glbCombinedDevLayerFilename)
If develGD Is Nothing Then
    MsgBox "Couldn't find data for your development scenario!"
    Exit Sub
End If
Dim lyrDevel As New MapObjects2.MapLayer
Set lyrDevel.GeoDataset = develGD
Dim develRecSet As MapObjects2.Recordset 'This is the set of records for the developments
Set develRecSet = lyrDevel.Records
Dim develField As MapObjects2.Field
Dim develPoly As MapObjects2.Polygon

' These are variables for the sewer data
Dim sewGD As MapObjects2.GeoDataset
Set sewGD = DatConn.FindGeoDataset("sewertest")
If sewGD Is Nothing Then
    MsgBox "Couldn't find data for the sewer lines!"
    Exit Sub
End If
Dim lyrSew As New MapObjects2.MapLayer
Set lyrSew.GeoDataset = sewGD
Dim sewRecSet As MapObjects2.Recordset 'This is the set of records for the sewer layer
Set sewRecSet = lyrSew.Records
Dim sewField As MapObjects2.Field
Dim SewLine As MapObjects2.Line
Dim MinDist As Double
Dim distance As Double

'CountNumber is used to figure out which of the development square records needs to be edited
Dim CountNumber As Integer
Dim j As Integer

If develRecSet.Count <> 0 Then 'Make sure records exist in the set of developments
develRecSet.MoveFirst
CountNumber = 0

Do While Not develRecSet.EOF 'Loop thru each of the development polygons
CountNumber = CountNumber + 1 'Count which record the loop is on
MinDist = 999999999999# 'set an impossibly high min. sewer distance for initialization
Set develField = develRecSet.Fields("shape")
Set develPoly = develField.Value

'This section finds the distance to sewers
'Loop thru each line in the sewer layer
sewRecSet.MoveFirst
Do While Not sewRecSet.EOF
Set sewField = sewRecSet.Fields("shape")
Set SewLine = sewField.Value
distance = develPoly.DistanceTo(SewLine) 'The DistanceTo functions works between 2 shapes
If distance < MinDist Then MinDist = distance 'Track the minimum distance
sewRecSet.MoveNext
Loop

' Edit values in the development layer's SewDist field
' Update method causes a jump back to the first record in the recordset
' Use a loop to move back down again to the appropriate record
With lyrDevel.Records
 .MoveFirst
 .For j = 2 To CountNumber
 .MoveNext
 Next j
 .Edit
 .Fields("SewDist").Value = MinDist
 .Update
 End With

End If

'Development squares in a cluster store 0 as a distance. This section adjusts for supergroups of development squares --
Dim i As Integer
Dim MinAdjustDistance As Double, AdjustDistance As Double
Dim j As Integer
Dim MinDist As Double
Dim AdjustDistance As Double

If develRecSet.Count <> 0 Then 'Make sure records exist in the set of developments
develRecSet.MoveFirst
CountNumber = 0

Do While Not develRecSet.EOF 'Loop thru each of the development polygons
CountNumber = CountNumber + 1 'Count which record the loop is on
MinDist = 999999999999# 'set an impossibly high min. sewer distance for initialization
Set develField = develRecSet.Fields("shape")
Set develPoly = develField.Value

'This section finds the distance to sewers
'Loop thru each line in the sewer layer
sewRecSet.MoveFirst
Do While Not sewRecSet.EOF
Set sewField = sewRecSet.Fields("shape")
Set SewLine = sewField.Value
distance = develPoly.DistanceTo(SewLine) 'The DistanceTo functions works between 2 shapes
If distance < MinDist Then MinDist = distance 'Track the minimum distance
sewRecSet.MoveNext
Loop

' Edit values in the development layer's SewDist field
' Update method causes a jump back to the first record in the recordset
' Use a loop to move back down again to the appropriate record
With lyrDevel.Records
 .MoveFirst
 .For j = 2 To CountNumber
 .MoveNext
 Next j
 .Edit
 .Fields("SewDist").Value = MinDist
 .Update
 End With

End If

End If

'Run a loop from 1 to the number of development squares to cover all the potential supergrp numbers that have been assigned (the number of supergrps must be lower than the number of squares)
Dim i As Integer
Dim MinAdjustDistance As Double, AdjustDistance As Double
For i = 1 To develRecSet.Count
   'Set an incredibly high minimum distance initially
   MinAdjustDistance = 1E+19
   develRecSet.MoveFirst
   Do While Not develRecSet.EOF 'loop thru each of the development squares
      'If the development square belongs to supergroup(i) then keep only the
      'minimum distance in the database for the minimum-distance tract. Set
      'the distance of all other tracts in the supergrp to 0
      If develRecSet.Fields("SuperGrp").Value = i Then
         AdjustDistance = develRecSet.Fields("SewDist").Value
         'If the distance for a square is less than the minimum, then reset the min.
         If AdjustDistance < MinAdjustDistance Then MinAdjustDistance = AdjustDistance
         End If
      DevelRecSet.MoveNext
   Loop

   'loop thru the development squares a second time to actually write the values
   develRecSet.MoveFirst
   CountNumber = 0
   Do While Not develRecSet.EOF 'loop thru each of the development squares
      CountNumber = CountNumber + 1 'Track which record the loop is on
      'For SuperGrp(i), if the distance is not equal to the minimum, then set it to 0.
      If develRecSet.Fields("SuperGrp").Value = i And _
         develRecSet.Fields("SewDist").Value <> MinAdjustDistance Then
         With lyrDevel.Records
            .MoveFirst
            For j = 2 To CountNumber
               .MoveNext
               .Edit
               .Fields("SewDist").Value = 0
               .Update
            Next j
            End With
         End If
      End If
      develRecSet.MoveNext
   Loop
   Next i

   'Readjust sewer distance data to account for cases where no sewer is needed
   'i.e., Low density tracts use septic, so sewer distance becomes 0
   develRecSet.MoveFirst
   CountNumber = 0
   Do While Not develRecSet.EOF 'Loop thru each of the development polygons
      CountNumber = CountNumber + 1
      If develRecSet.Fields("DevType").Value = "LDR" Then
         'Edit values in the development layer's SewDist field
         'Update method causes a jump back to the first record in the recordset
         'Use a loop to move back down again to the appropriate record
         With lyrDevel.Records
            .MoveFirst
            For j = 2 To CountNumber
               .MoveNext
               .Edit
               .Fields("SewDist").Value = 0
               .Update
            Next j
            End With
         End If
      End If
'This subroutine calculates and stores the distance from a neighborhood to a water line.
'Note: Current water line layer is for test purposes only -- it is not a real layer.
Public Sub CalcWaterDist()

    Dim dcWater As New MapObjects2.DataConnection
    dcWater.Database = App.Path & "\BC_Data"

    'These are variables for the development squares data
    Dim develsGD As MapObjects2.GeoDataset
    Set develsGD = dcWater.FindGeoDataset(glbCombinedDevLayerFilename)
    If develsGD Is Nothing Then
        MsgBox "Couldn't find data for your development scenario!"
        Exit Sub
    End If

    Dim lyrDevels As New MapObjects2.MapLayer
    Set lyrDevels.GeoDataset = develsGD
    Dim develsRecSet As MapObjects2.Recordset 'This is the set of records for the developments
    Set develsRecSet = lyrDevels.Records
    Dim develsField As MapObjects2.Field
    Dim develsPoly As MapObjects2.Polygon

    'These are variables for the water data
    Dim watGD As MapObjects2.GeoDataset
    Set watGD = dcWater.FindGeoDataset("watertest")
    If watGD Is Nothing Then
        MsgBox "Couldn't find data for the water lines!"
        Exit Sub
    End If

    Dim lyrWat As New MapObjects2.MapLayer
    Set lyrWat.GeoDataset = watGD
    Dim watRecSet As MapObjects2.Recordset 'This is the set of records for the water layer
    Set watRecSet = lyrWat.Records
    Dim watField As MapObjects2.Field
    Dim watLine As MapObjects2.Line
    Dim MinDist As Double
    Dim distance As Double

    'CountNumber is used to figure out which of the development square records needs to be edited
    Dim CountNumber As Integer
    Dim j As Integer

    If develsRecSet.Count <> 0 Then 'Make sure records exist in the set of developments
        develsRecSet.MoveFirst
        CountNumber = 0

        Do While Not develsRecSet.EOF 'Loop thru each of the development polygons
            CountNumber = CountNumber + 1 'Count which record the loop is on
            MinDist = 999999999999# 'set an impossibly high min. water distance for initialization
            Set develsField = develsRecSet.Fields("shape")

           -develRecSet.MoveNext
            Loop

            lyrDevel.Records.StopEditing
            DatConn.Disconnect
        End Sub
    End If

    Do While Not watRecSet.EOF 'Loop thru each of the water lines
        distance = 0
        Do While Not develsRecSet.EOF
            If distance <= MinDist Then
                MinDist = distance
            End If
            distance = distance + 1
        Loop
        If Distance < MinDist Then
            Distance = Distance + 1
        End If
    Loop

    MsgBox "Min. distance is " & MinDist
End Sub
Set develsPoly = develsField.Value

'This section finds the distance to waters
'Loop thru each line in the water layer
watRecSet.MoveFirst
Do While Not watRecSet.EOF
    Set watField = watRecSet.Fields("shape")
    Set watLine = watField.Value
    distance = develsPoly.DistanceTo(watLine) 'The DistanceTo functions works between 2 shapes
    If distance < MinDist Then MinDist = distance 'Track the minimum distance
    watRecSet.MoveNext
Loop

' Edit values in the development layer's WatDist field
' Update method causes a jump back to the first record in the recordset
' Use a loop to move back down again to the appropriate record
With lyrDevels.Records
    .MoveFirst
    For j = 2 To CountNumber
        .MoveNext
        Next j
    .Edit
    .Fields("WatDist").Value = MinDist
    .Update
End With

develsRecSet.MoveNext
Loop
lyrDevels.Records.StopEditing

End If

'Readjust water distance data to account for supergroups of development squares --
'If a square is in a cluster, distance to water line only needs to be measured from
'the nearest of those squares -- other squares in the cluster store 0 as a distance.

'Rerun a loop from 1 to the number of development squares to cover all the potential
'supgrp numbers that have been assigned (the number of supergrps must be lower
'than the number of squares)
Dim i As Integer
Dim MinAdjustDistance As Double, AdjustDistance As Double
For i = 1 To develsRecSet.Count
    'Set an incredibly high minimum distance initially
    MinAdjustDistance = 1E+19
    develsRecSet.MoveFirst
    Do While Not develsRecSet.EOF 'loop thru each of the development squares
        'If the development square belongs to supergroup(i) then keep only the
        'minimum distance in the database for the minimum-distance tract. Set
        'the distance of all other tracts in the supergrp to 0
        If develsRecSet.Fields("SuperGrp").Value = i Then
            AdjustDistance = develsRecSet.Fields("WatDist").Value
            'If the distance for a square is less than the minimum, then reset the min.
            If AdjustDistance < MinAdjustDistance Then MinAdjustDistance = AdjustDistance
        End If
        develsRecSet.MoveNext
    Loop

    'loop thru the development squares a second time to actually write the values
    develsRecSet.MoveFirst
CountNumber = 0
Do While Not develsRecSet.EOF 'loop thru each of the development squares
  CountNumber = CountNumber + 1 'Track which record the loop is on
  If develsRecSet.Fields("SuperGrp").Value = i And _
     develsRecSet.Fields("Watdist").Value <> MinAdjustDistance Then
    With lyrDevels.Records
      .MoveFirst
      For j = 2 To CountNumber
        .MoveNext
        Next j
      .Edit
      .Fields("WatDist").Value = 0
      .Update
    End With
  End If
  develsRecSet.MoveNext
Loop
Next i

'Readjust water distance data to account for cases where no water is needed
'i.e., Low density tracts use wells, so water distance becomes 0
DevelsRecSet.MoveNext
CountNumber = 0
Do While Not develsRecSet.EOF 'Loop thru each of the development polygons
  CountNumber = CountNumber + 1
  If develsRecSet.Fields("DevType").Value = "LDR" Then
    ' Edit values in the development layer's watDist field
    ' Update method causes a jump back to the first record in the recordset
    ' Use a loop to move back down again to the appropriate record
    With lyrDevels.Records
      .MoveFirst
      For j = 2 To CountNumber
        .MoveNext
        Next j
      .Edit
      .Fields("WatDist").Value = 0
      .Update
    End With
  End If
  develsRecSet.MoveNext
Loop
lyrDevels.Records.StopEditing
dcWater.Disconnect
End Sub

FisheriesOperations Module

Option Explicit

'*******************************************************************************
' * The FisheriesOperations module contains a collection of subroutines          *
' * that provide modeling of fish health                                       *
' * Created March 24, 1999                                                   *
*******************************************************************************
'This subroutine calculates mean metric scores for fish health and reports them 'to the user in a table and on the map.
Public Sub EstimateFishHealth()

' Set up dataset/layer variables
Dim dcFish As New MapObjects2.DataConnection
    dcFish.Database = App.Path & \"BC_Data\" ' Tells where to look for data
Dim gdFish As MapObjects2.GeoDataset
    Set gdFish = dcFish.FindGeoDataset("fish_outlets") ' Find outlet points shapefile
Dim lyrFish As New MapObjects2.MapLayer
    lyrFish.GeoDataset = gdFish
Dim rsFish As New MapObjects2.Recordset
    Set rsFish = lyrFish.Records

' Set up variables related to the mean metric score for fish health
' MMS before the user adds a scenario
Dim BaseMMS(numLandSegs) As Double
' MMS after the user runs a scenario
Dim NewMMS(numLandSegs) As Double
' Constant in the MMS regression equation
Dim MMSRegressConst As Double
    MMSRegressConst = 0.3158
' Coefficient for the fraction of disturbed land
Dim MMSCoeff As Double
    MMSCoeff = -0.0426
Dim i As Integer
Dim j As Integer
Dim aggBaseDisturbedArea(numLandSegs) As Double
Dim aggNewDisturbedArea(numLandSegs) As Double
Dim aggBaseDisturbedPct(numLandSegs) As Double
Dim aggNewDisturbedPct(numLandSegs) As Double
Dim aggAllArea(numLandSegs) As Double
Dim MMSUpper As Double, MMSLower As Double
    MMSUpper = 0.569
    MMSLower = 0.4
Dim MMSBaseCategory(numLandSegs) As String
Dim MMSNewCategory(numLandSegs) As String

' Initialize aggregation variables to 0
For i = 1 To numLandSegs
    aggBaseDisturbedArea(i) = 0
    aggNewDisturbedArea(i) = 0
    aggAllArea(i) = 0
Next i

'Determine the baseline and new mean metric scores (MMS) for each land segment.
'MMS is based on a regression from the 43 fish sampling sites throughout the Upper Roanoke River Watershed. The simple regression equation is
'MMS = -0.0426ln(FractionDisturbed) + 0.3158
For i = 1 To numLandSegs ' Loop through each subwatershed
' The first two subwatersheds have no upstream subwatersheds, so their
' land areas do not need to be aggregated -- their mean metric scores
' can be calculated directly
    If i = 1 Or i = 2 Then
        ' Calculate the regression equation (baseline and with scenario)
        BaseMMS(i) = MMSCoeff * Log(glbPctOldDisturb(i)) + MMSRegressConst
        NewMMS(i) = MMSCoeff * Log(glbPctNewDisturb(i)) + MMSRegressConst
    Else
        ' The remaining subwatersheds (3-10) have upstream subwatersheds,
        ' so their land areas need to be aggregated.
        Else
For j = 1 To numLandSegs 'Loop thru subwatersheds
    'If the inner loop (j) subwatershed is upstream of the outer loop (i)
    'subwatershed or the loops are on the same subwatershed,
    'then add its disturbed land area to the total. Also add up the
    'total area
    If j <= i Then
        aggBaseDisturbedArea(i) = aggBaseDisturbedArea(i) + glbTotOldDisturb(j)
        aggNewDisturbedArea(i) = aggNewDisturbedArea(i) + glbTotNewDisturb(j)
        aggAllArea(i) = aggAllArea(i) + glbTotAllLandUses(j)
    End If
Next j

'Calculate proportions of disturbed area to plug into the regression
aggBaseDisturbedPct(i) = aggBaseDisturbedArea(i) / aggAllArea(i)
aggNewDisturbedPct(i) = aggNewDisturbedArea(i) / aggAllArea(i)

'Calculate the regression equation (baseline and with scenario)
BaseMMS(i) = MMSCoeff * Log(aggBaseDisturbedPct(i)) + MMSRegressConst
NewMMS(i) = MMSCoeff * Log(aggNewDisturbedPct(i)) + MMSRegressConst
End If

'Determine the MMS category of the base mean metric score
If BaseMMS(i) > MMSUpper Then
    MMSBaseCategory(i) = "Green" 'high integrity sites
    frmFishReport.picBaseMMS(i - 1).BackColor = RGB(0, 255, 0) 'Color a box green
ElseIf BaseMMS(i) < MMSLower Then
    MMSBaseCategory(i) = "Red" 'low integrity sites
    frmFishReport.picBaseMMS(i - 1).BackColor = RGB(255, 0, 0) 'Color a box red
Else:
    MMSBaseCategory(i) = "Yellow" 'mid integrity sites
    frmFishReport.picBaseMMS(i - 1).BackColor = RGB(255, 255, 0) 'Color a box yellow
End If

'Determine the MMS category of the new mean metric score
If NewMMS(i) > MMSUpper Then
    MMSNewCategory(i) = "Green" 'high integrity sites
    frmFishReport.picNewMMS(i - 1).BackColor = RGB(0, 255, 0) 'Color a box green
ElseIf NewMMS(i) < MMSLower Then
    MMSNewCategory(i) = "Red" 'low integrity sites
    frmFishReport.picNewMMS(i - 1).BackColor = RGB(255, 0, 0) 'Color a box red
Else:
    MMSNewCategory(i) = "Yellow" 'mid integrity sites
    frmFishReport.picNewMMS(i - 1).BackColor = RGB(255, 255, 0) 'Color a box yellow
End If
Next i

'Write the base and new MMS Categories in the outlet point shapefiles' databases
'Remember to scroll through the recordset to the correct record
Dim recCount, k As Integer
recCount = 0
Do While Not rsFish.EOF
    recCount = recCount + 1
    With lyrFish.Records
        .MoveFirst
        For k = 2 To recCount
            .MoveNext
        Next k
        .Edit
        .Fields("MMSCat").Value = MMSBaseCategory(recCount)
        .Update
    End With
    rsFish.MoveNext
Loop

Set rsFish = Nothing
'lyrFish.Records.StopEditing

'Put the results in a table and show them on a map
frmFishReport.Show
'insert titles for list boxes
With frmFishReport
  .lstBaseSubwat.AddItem ("Watershed")
  .lstBaseSubwat.AddItem ("--------------------------")
  .lstNewSubwat.AddItem ("Watershed")
  .lstNewSubwat.AddItem ("--------------------------")
  .lstBaseDist.AddItem ("Portion Disturbed")
  .lstBaseDist.AddItem ("--------------------------")
  .lstNewDist.AddItem ("Portion Disturbed")
  .lstNewDist.AddItem ("--------------------------")
  .lstBaseMMS.AddItem ("Mean Met. Score")
  .lstBaseMMS.AddItem ("--------------------------------")
  .lstNewMMS.AddItem ("Mean Met. Score")
  .lstNewMMS.AddItem ("--------------------------------")
End With

For i = 1 To numLandSegs
  With frmFishReport
    'insert subwatershed numbers in list boxes
    .lstBaseSubwat.AddItem (" " & i)
    .lstNewSubwat.AddItem (" " & i)
    'insert fraction of disturbed area in list boxes
    If i = 1 Or i = 2 Then
      .lstBaseDist.AddItem (" " & Round(glbPctOldDisturb(i), 4))
      .lstNewDist.AddItem (" " & Round(glbPctNewDisturb(i), 4))
    Else
      .lstBaseDist.AddItem (" " & Round(aggBaseDisturbedPct(i), 4))
      .lstNewDist.AddItem (" " & Round(aggNewDisturbedPct(i), 4))
    End If
    'insert mean metric score in list boxes
    .lstBaseMMS.AddItem (" " & Round(BaseMMS(i), 4))
    .lstNewMMS.AddItem (" " & Round(NewMMS(i), 4))
  End With
Next i

'Draw results on the two maps
'Set up rendering variables
Dim rndFishOutlet As New MapObjects2.ValueMapRenderer
Dim rndSubwat As New MapObjects2.LabelRenderer

'Set up the Base Fish Outlets (circle polygons) Layer with symbol styles and colors
With lyrFish
  .Name = "Fish Outlets" 'Set Name property of new MapLayer
  .Symbol.Style = moSolidFill 'Set fill style
  .Symbol.OutlineColor = moBlack 'set outline color
End With

Set lyrFish.Renderer = rndFishOutlet 'Set the renderer for the layer to code sites 'as red, yellow, or green

'Set up the renderer for the fish outlet sites
With lyrFish.Renderer

.ValueCount = 3 '3 possibilities -- red, yellow, or green
.Field = "MMSCat" 'Use the BaseMMS field to choose the appropriate color
  .Value(0) = "Green"
  .Symbol(0).Color = moGreen
  .Value(1) = "Yellow"
  .Symbol(1).Color = moYellow
  .Value(2) = "Red"
  .Symbol(2).Color = moRed
End With

frmFishReport.mapFishBase.Layers.Add lyrFish 'Add subwatersheds to base map layer collection

Set lyrFish = Nothing 'Reset layer to add the next file

'Set up the New Fish Outlets (circle polygons) Layer
Set gdFish = dcFish.FindGeoDataset("New_Fish_Outlets") 'Find subwatershed shapefile
lyrFish.GeoDataset = gdFish
Set rsFish = lyrFish.Records

'Write values in the shapefile
recCount = 0
Do While Not rsFish.EOF
  recCount = recCount + 1
  With lyrFish.Records
    .MoveFirst
    For k = 2 To recCount
      .MoveNext
      Next k
    .Edit
    .Fields("MMSCat").Value = MMSNewCategory(recCount)
    .Update
  End With
  rsFish.MoveNext
Loop

With lyrFish
  .GeoDataset = gdFish 'Set GeoDataset property of new map layer
  .Name = "New Fish Outlets" 'Set Name property of new MapLayer
  .Symbol.Style = moSolidFill 'Set fill style
  .Symbol.OutlineColor = moBlack 'set outline color
End With

Set lyrFish.Renderer = rndFishOutlet 'Apply the renderer to the outlets
frmFishReport.mapFishNew.Layers.Add lyrFish 'Add subwatersheds to new map layer collection

Set lyrFish = Nothing 'Reset layer to add the next file

'Set up the Subwatersheds Layer with symbol styles, colors, and labels
Set gdFish = dcFish.FindGeoDataset("subwatersheds") 'Find subwatershed shapefile
With lyrFish
  .GeoDataset = gdFish 'Set GeoDataset property of new map layer
  .Name = "Subwatersheds" 'Set Name property of new MapLayer
  .Symbol.OutlineColor = moBlack
  .Symbol.SymbolType = moFillSymbol
  .Symbol.Style = moTransparentFill
End With

Set lyrFish.Renderer = rndSubwat
lyrFish.Renderer.Field = "Wshid" 'Label the watersheds with their id numbers

frmFishReport.mapFishBase.Layers.Add lyrFish  'Add subwatersheds to base map layer
collection
frmFishReport.mapFishNew.Layers.Add lyrFish  'Add subwatersheds to new map layer
collection

Set lyrFish = Nothing 'Reset layer to add the next file

'Set up the Stream Reaches Layer with symbol styles and colors
Set gdFish = dcFish.FindGeoDataset("reaches") 'Find the shapefile of stream reaches
With lyrFish
  .GeoDataset = gdFish 'Set GeoDataset property of new map layer
  .Name = "Stream Reaches" 'Set Name property of new MapLayer
  .Symbol.Color = moBlue
  .Symbol.SymbolType = moLineSymbol
End With

frmFishReport.mapFishBase.Layers.Add lyrFish  'Add subwatersheds to base map layer
collection
frmFishReport.mapFishNew.Layers.Add lyrFish  'Add subwatersheds to new map layer
collection

'Zoom the maps to the full extent
frmFishReport.mapFishBase.Extent = frmFishReport.mapFishBase.FullExtent
frmFishReport.mapFishNew.Extent = frmFishReport.mapFishNew.FullExtent

dcFish.Disconnect

End Sub

frmMain Form

Option Explicit
Option Base 0

Private Sub cmdInputComplete_Click()
  'Make the status frame with its message label appear to the user
  lblStatusMessage.Caption = "Compiling attributes of your scenario. Please wait..."
  fraStatusMessage.Visible = True
  frmMain.Refresh
  'Disable the inputComplete button
  cmdInputComplete.Enabled = False
  'Disable the map, so the user can't click any more developments on it
  mapDisplay.Enabled = False

  Call SaveSquaresAsDatabase 'In the AddDevelopmentsOperations Module
  Call FindSuperGroupsOfSquares 'In the AddDevelopmentsOperations Module
  frmModel.Show
  'Provide a new message for the user in the status frame to tell him/her
  'how to input a new scenario if necessary
  lblStatusMessage.Caption = "Hitting the CLEAR ALL button will allow you to input a
  new scenario."
End Sub

Private Sub Form_Load()
  fraStatusMessage.Visible = False
'Set the position of the form on the screen
frmMain.Left = Screen.Width / 10
frmMain.Top = Screen.Height / 20

'First set some variable values to define the input grid pattern (like a SimCity grid)
glbSquareSize = 150 'Set the development square radius (half the length or width) size to 150 meters
glbGridMaxX = 601260.843644
glbGridMinX = 576060.843644
glbGridMaxY = 4121901.383057
glbGridMinY = 4110501.383057

Call AutoLoadDataLayer 'In the BasicOperations Module -- Loads base data
Call LinkLegend 'In the BasicOperations Module -- links legend control to the map

'This block sets the addedpreviously array to false, since the user has added no data to the map at this point. This gets used in frmMoreData where the user adds extra data.
Dim n As Integer
For n = 0 To 49
    glbAddedPreviously(n) = False
Next n

End Sub

'Updates map when user turns layers on or off
Private Sub legMapDisplay_AfterSetLayerVisible(Index As Integer, isVisible As Boolean)
    frmMain.mapDisplay.Refresh
End Sub

'This subroutine completes actions that the user wants to do by clicking the map
Private Sub mapDisplay_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
    Dim clickedPt As New MapObjects2.Point
    Dim ClickX As Double, clickY As Double

    'Zoom in button was pushed
    If tbrMapControl.Buttons("ZoomIn").Value = 1 Then
        Call ZoomIn 'In the BasicOperations Module
    End If

    'Zoom out button was pushed
    ElseIf tbrMapControl.Buttons("ZoomOut").Value = 1 Then
        Call ZoomOut 'In the BasicOperations Module
    ElseIf tbrMapControl.Buttons("Pan").Value = 1 Then
        Call Pan 'In the BasicOperations Module
    ElseIf tbrMapControl.Buttons("LowDensityRes").Value = 1 Then
        Set clickedPt = mapDisplay.ToMapPoint(X, Y)
        ClickX = clickedPt.X
        clickY = clickedPt.Y
        clickedPt.X = SnapXToGrid(ClickX) 'In the AddDevelopmentsOperations module -- changes x coordinate to fit the grid
clickedPt.Y = SnapYToGrid(clickY) 'In the AddDevelopmentsOperations module -- changes y coordinate to fit the grid

If SeeThatSquareIsInGrid(clickedPt) = 1 Then
    Call AddDevelopmentSquareL(clickedPt) 'In AddDevelopments Operations Module
Else
    MsgBox "Your Development must be on the grid!"
End If

'Drop-Mid-Density-Residential-Conventional-Development button was pushed
ElseIf tbrMapControl.Buttons("MidDensityResConv").Value = 1 Then

Set clickedPt = mapDisplay.ToMapPoint(X, Y)

ClickX = clickedPt.X
clickY = clickedPt.Y
clickedPt.X = SnapXToGrid(ClickX) 'In the AddDevelopmentsOperations module -- changes x coordinate to fit the grid
clickedPt.Y = SnapYToGrid(clickY) 'In the AddDevelopmentsOperations module -- changes y coordinate to fit the grid

If SeeThatSquareIsInGrid(clickedPt) = 1 Then
    Call AddDevelopmentSquareMC(clickedPt) 'In AddDevelopments Operations Module
Else
    MsgBox "Your Development must be on the grid!"
End If

'Drop-Mid-Density-Residential-Cluster-Development button was pushed
ElseIf tbrMapControl.Buttons("MidDensityResCluster").Value = 1 Then

Set clickedPt = mapDisplay.ToMapPoint(X, Y)

ClickX = clickedPt.X
clickY = clickedPt.Y
clickedPt.X = SnapXToGrid(ClickX) 'In the AddDevelopmentsOperations module -- changes x coordinate to fit the grid
clickedPt.Y = SnapYToGrid(clickY) 'In the AddDevelopmentsOperations module -- changes y coordinate to fit the grid

If SeeThatSquareIsInGrid(clickedPt) = 1 Then
    Call AddDevelopmentSquareMCL(clickedPt) 'In AddDevelopments Operations Module
Else
    MsgBox "Your Development must be on the grid!"
End If

'Drop-High-Density-Residential-Development button was pushed
ElseIf tbrMapControl.Buttons("HighDensityRes").Value = 1 Then

Set clickedPt = mapDisplay.ToMapPoint(X, Y)

ClickX = clickedPt.X
clickY = clickedPt.Y
clickedPt.X = SnapXToGrid(ClickX) 'In the AddDevelopmentsOperations module -- changes x coordinate to fit the grid
clickedPt.Y = SnapYToGrid(clickY) 'In the AddDevelopmentsOperations module -- changes y coordinate to fit the grid

If SeeThatSquareIsInGrid(clickedPt) = 1 Then
    Call AddDevelopmentSquareH(clickedPt) 'In AddDevelopments Operations Module
Else
    MsgBox "Your Development must be on the grid!"
End If
'Drop-Commercial-Development button was pushed 
ElseIf tbrMapControl.Buttons("Commercial").Value = 1 Then

    Set clickedPt = mapDisplay.ToMapPoint(X, Y)

    ClickX = clickedPt.X
    clickY = clickedPt.Y
    clickedPt.X = SnapXToGrid(ClickX) 'In the AddDevelopmentsOperations module -- changes x coordinate to fit the grid
    clickedPt.Y = SnapYToGrid(clickY) 'In the AddDevelopmentsOperations module -- changes y coordinate to fit the grid

    If SeeThatSquareIsInGrid(clickedPt) = 1 Then
        Call AddDevelopmentSquareC(clickedPt) 'In AddDevelopments Operations Module
    Else
        MsgBox "Your Development must be on the grid!"
    End If

'Remove-a-development-square button was pushed 
'This allows a user to remove unwanted development squares
ElseIf tbrMapControl.Buttons("Remove").Value = 1 Then

    Dim RemoveClickedPt As New MapObjects2.Point

    Set RemoveClickedPt = mapDisplay.ToMapPoint(X, Y)
    Call RemoveSingleSquare(RemoveClickedPt) 'In the AddDevelopmentsOperations module -- removes a clicked square

End If

End Sub

'This subroutine draws user-clicked squares on the map
Private Sub mapDisplay_afterTrackingLayerDraw(ByVal hDC As Stdole.OLE_HANDLE)

    Dim polyL As MapObjects2.Polygon 'this is the low density development square
    Dim polyMC As MapObjects2.Polygon 'this is the mid density conv. square
    Dim polyMCL As MapObjects2.Polygon 'this is the mid density cluster square
    Dim polyH As MapObjects2.Polygon 'this is the high density square
    Dim polyC As MapObjects2.Polygon 'this is the commercial square

    Dim DevelSquareSymL As New MapObjects2.Symbol 'this is the symbol for a low density square
    Dim DevelSquareSymMC As New MapObjects2.Symbol 'this is the symbol for the mid conv square
    Dim DevelSquareSymMCL As New MapObjects2.Symbol 'this is the symbol for the mid cluster square
    Dim DevelSquareSymH As New MapObjects2.Symbol 'this is the symbol for the high density square
    Dim DevelSquareSymC As New MapObjects2.Symbol 'this is the symbol for the commercial square

    With DevelSquareSymL 'This with block sets the symbol properties
        .SymbolType = moFillSymbol
        .Style = moSolidFill
        .Color = moRed
    End With

    With DevelSquareSymMC 'This with block sets the symbol properties
        .SymbolType = moFillSymbol
        .Style = moSolidFill
        .Color = moBlue
    End With

    With DevelSquareSymMCL 'This with block sets the symbol properties
        .SymbolType = moFillSymbol
        .Style = moSolidFill
        .Color = moRed
    End With
.SymbolType = moFillSymbol
.Style = moSolidFill
.Color = moGreen
End With

With DevelSquareSymH 'This with block sets the symbol properties
.SymbolType = moFillSymbol
.Style = moSolidFill
.Color = moYellow
End With

With DevelSquareSymC 'This with block sets the symbol properties
.SymbolType = moFillSymbol
.Style = moSolidFill
.Color = moPurple
End With

' If there are polygons in the collection (colDevelSquares), then draw them on map
' Draw polygons in different colors depending on what type of development the user put down
' Low density res
If colDevelSquaresL.Count <> 0 Then
    For Each polyL In colDevelSquaresL
        frmMain.mapDisplay.DrawShape polyL, DevelSquareSymL
    Next polyL
End If

'Mid density conventional res
If colDevelSquaresMC.Count <> 0 Then
    For Each polyMC In colDevelSquaresMC
        frmMain.mapDisplay.DrawShape polyMC, DevelSquareSymMC
    Next polyMC
End If

'Mid density cluster res
If colDevelSquaresMCL.Count <> 0 Then
    For Each polyMCL In colDevelSquaresMCL
        frmMain.mapDisplay.DrawShape polyMCL, DevelSquareSymMCL
    Next polyMCL
End If

'High density res
If colDevelSquaresH.Count <> 0 Then
    For Each polyH In colDevelSquaresH
        frmMain.mapDisplay.DrawShape polyH, DevelSquareSymH
    Next polyH
End If

'Commercial
If colDevelSquaresC.Count <> 0 Then
    For Each polyC In colDevelSquaresC
        frmMain.mapDisplay.DrawShape polyC, DevelSquareSymC
    Next polyC
End If

End Sub

Private Sub mnuAbout_Click()
    frmAbout.Show
End Sub

Private Sub mnuAddData_Click()
'_frmMain.tbrMapControl.Buttons(5).Value = tbrPressed
Call tbrMapControl_ButtonClick(tbrMapControl.Buttons(5))
End Sub

Private Sub mnuFilePrint_Click()
    frmMain.mapDisplay.PrintMap "Back Creek", ",", True
    MsgBox "Your map is being printed on your Windows Default Printer."
End Sub

Private Sub mnuFileQuit_Click()
    Call QuitProgram 'In the BasicOperations Module
End Sub

Private Sub mnuHelpDevCosts_Click()
    frmHelp.Show
    frmHelp.webHelp.Navigate App.Path & "\HelpFiles\DevelCostHelp.html"
End Sub

Private Sub mnuHelpEcon_Click()
    dlgDialog.ShowHelp
End Sub

Private Sub mnuHelpFish_Click()
    frmHelp.Show
    frmHelp.webHelp.Navigate App.Path & "\HelpFiles\FishHelp.html"
End Sub

Private Sub mnuHelpHSPF_Click()
    frmHelp.Show
    frmHelp.webHelp.Navigate App.Path & "\HelpFiles\HSPFHelp.html"
End Sub

Private Sub mnuHelpLandVal_Click()
    frmHelp.Show
    frmHelp.webHelp.Navigate App.Path & "\HelpFiles\LandValuesHelp.html"
End Sub

Private Sub mnuInputScenario_Click()
    frmHelp.Show
    frmHelp.webHelp.Navigate App.Path & "\HelpFiles\InputScenariosHelp.html"
End Sub

Private Sub tbrMapControl_ButtonClick(ByVal Button As MSComctlLib.Button)
    Dim ButtKey As String
    ButtKey = Button.Key
    Call MapToolBarTask(ButtKey) 'In the BasicOperations Module
End Sub

_frmAbout Form

Private Sub cmdOK_Click()
    frmAbout.Hide
End Sub

Private Sub Form_Load()
    frmAbout.Top = Screen.Height / 20
    frmAbout.Left = Screen.Width / 5
End Sub
frmMoreData Form

Option Explicit
Option Base 0

'Use the load event to populate the lists of data
Private Sub Form_Load()
' Set the form's position on the screen
frmMoreData.Left = frmMain.Left
frmMoreData.Top = frmMain.Top

' Put the names of data sets in the general data list
With lstGeneralData
    .AddItem "Land Use", 0
End With

' Put the names of data sets in the hydrology data list
With lstHydroData
    .AddItem "Rain Gages", 0
    .AddItem "Thiessen Polygons", 1
    .AddItem "Main Stem Reaches", 2
    .AddItem "Reach Outlet Points", 3
    .AddItem "Stream Gages", 4
    .AddItem "Tinker Creek WS", 5
    .AddItem "Upper Roanoke WS", 6
End With

' Put the names of data sets in the economics data list
With lstEconData
    .AddItem "Roads", 0
    .AddItem "Sewer Lines", 1
    .AddItem "Water Supply Lines", 2
End With

' Put the names of data sets in the ecology data list
With lstEcolData
    .AddItem "Fish Sampling Sites", 0
End With

End Sub

' Add the data highlighted in the list boxes to the map
Private Sub cmdAddData_Click()
    Dim i As Integer

    i = 0

    ' Check to see what GENERAL layers the user wants to add to the map and specify properties
    If lstGeneralData.Selected(0) = True And glbAddedPreviously(0) = False Then
        ' Define properties for land use layer
        glbLayersNewData(i).Name = "land_use"
        glbLayersNewData(i).Renderer = True
        i = i + 1
        glbAddedPreviously(0) = True
    End If

    ' Check to see what HYDROLOGY layers the user wants to add to the map and specify properties
    If lstHydroData.Selected(0) = True And glbAddedPreviously(10) = False Then
        ' Define properties for rain gages layer
        glbLayersNewData(i).Name = "rain_gages"
glbLayersNewData(i).SymbolColor = moDarkGreen
glbLayersNewData(i).OutlineColor = moDarkGreen
glbLayersNewData(i).PointStyle = moCircleMarker
glbLayersNewData(i).Renderer = False
i = i + 1
glbAddedPreviously(10) = True
End If

If lstHydroData.Selected(1) = True And glbAddedPreviously(11) = False Then
  'Define properties for Thiessen Polygons layer
  glbLayersNewData(i).Name = "thiessen"
  glbLayersNewData(i).FillStyle = moTransparentFill
  glbLayersNewData(i).OutlineColor = moBlack
  glbLayersNewData(i).Renderer = False
  i = i + 1
  glbAddedPreviously(11) = True
End If

If lstHydroData.Selected(2) = True And glbAddedPreviously(12) = False Then
  'Define properties for Main Stem Reaches layer
  glbLayersNewData(i).Name = "reaches"
  glbLayersNewData(i).SymbolColor = moNavy
  glbLayersNewData(i).Renderer = False
  i = i + 1
  glbAddedPreviously(12) = True
End If

If lstHydroData.Selected(3) = True And glbAddedPreviously(13) = False Then
  'Define properties for outlet points layer
  glbLayersNewData(i).Name = "outlet_points"
  glbLayersNewData(i).SymbolColor = moMaroon
  glbLayersNewData(i).OutlineColor = moMaroon
  glbLayersNewData(i).PointStyle = moTriangleMarker
  glbLayersNewData(i).Renderer = False
  i = i + 1
  glbAddedPreviously(13) = True
End If

If lstHydroData.Selected(4) = True And glbAddedPreviously(14) = False Then
  'Define properties for stream gages layer
  glbLayersNewData(i).Name = "stream_gages"
  glbLayersNewData(i).SymbolColor = moBrown
  glbLayersNewData(i).OutlineColor = moBrown
  glbLayersNewData(i).PointStyle = moCrossMarker
  glbLayersNewData(i).Renderer = False
  i = i + 1
  glbAddedPreviously(14) = True
End If

If lstHydroData.Selected(5) = True And glbAddedPreviously(15) = False Then
  'Define properties for Tinker Creek Watershed layer
  glbLayersNewData(i).Name = "tinker_ws"
  glbLayersNewData(i).FillStyle = moTransparentFill
  glbLayersNewData(i).OutlineColor = moRed
  glbLayersNewData(i).Renderer = False
  i = i + 1
  glbAddedPreviously(15) = True
End If

If lstHydroData.Selected(6) = True And glbAddedPreviously(16) = False Then
  'Define properties for Upper Roanoke Watershed layer
  glbLayersNewData(i).Name = "up_roan_ws"
glbLayersNewData(i).FillStyle = moTransparentFill
glbLayersNewData(i).OutlineColor = moRed
glbLayersNewData(i).Renderer = False
i = i + 1
  glbAddedPreviously(16) = True
End If

'Check to see what ECONOMICS layers the user wants to add to the map and specify properties
If lstEconData.Selected(0) = True And glbAddedPreviously(20) = False Then
  'Define properties for the roads
  glbLayersNewData(i).Name = "roads"
glbLayersNewData(i).SymbolColor = moBlack
glbLayersNewData(i).Renderer = False
  i = i + 1
  glbAddedPreviously(20) = True
End If

If lstEconData.Selected(1) = True And glbAddedPreviously(21) = False Then
  'Define properties for the sewer lines
  glbLayersNewData(i).Name = "SewerTest"
glbLayersNewData(i).SymbolColor = moBrown
glbLayersNewData(i).Renderer = False
  i = i + 1
  glbAddedPreviously(21) = True
End If

If lstEconData.Selected(2) = True And glbAddedPreviously(22) = False Then
  'Define properties for the water supply lines
  glbLayersNewData(i).Name = "WaterTest"
glbLayersNewData(i).SymbolColor = moCyan
glbLayersNewData(i).Renderer = False
  i = i + 1
  glbAddedPreviously(22) = True
End If

'Check to see what ECOLOGY layers the user wants to add to the map and specify properties
If lstEcolData.Selected(0) = True And glbAddedPreviously(30) = False Then
  'Define properties for fish sampling sites
  glbLayersNewData(i).Name = "fish_sites"
glbLayersNewData(i).SymbolColor = moNavy
glbLayersNewData(i).PointStyle = moCircleMarker
glbLayersNewData(i).Renderer = False
  i = i + 1
  glbAddedPreviously(30) = True
End If

glbLayersCountNewData = i

**********************************************************************************
'Draw the layers defined above on the map by adding layers to the
'MapObjects Layers collection
**********************************************************************************

Dim DataConnectNewData As New MapObjects2.DataConnection
Dim geoSetNewData As GeoDataset
Dim newDataLayer As New MapLayer
Dim j As Integer
Dim k As Integer
Dim MapRendererNewData As New MapObjects2.ValueMapRenderer
DataConnectNewData.Database = App.Path & "\BC_Data" 'Tells where to look for data
If DataConnectNewData.Connect = True Then 'If the data connection path is ok
   For j = 0 To glbLayersCountNewData - 1 'Loop through all new layers
      Set geoSetNewData = DataConnectNewData.FindGeoDataset(glbLayersNewData(j).Name)
      'Find shapefile as GeoDataset in DataConnectNewData
      newDataLayer.GeoDataset = geoSetNewData 'Set GeoDataset property of new MapLayer
      newDataLayer.Name = glbLayersNewData(j).Name 'Set Name property of new MapLayer
      If newDataLayer.Symbol.SymbolType = moFillSymbol Then 'Fill style only applies to polygons
         newDataLayer.Symbol.Style = glbLayersNewData(j).FillStyle 'Set fill of new MapLayer
      End If
      If newDataLayer.Symbol.SymbolType = moPointSymbol Then 'Point style only applies to points
         newDataLayer.Symbol.Style = glbLayersNewData(j).PointStyle 'set point style of new MapLayer
      End If
      If newDataLayer.Symbol.SymbolType <> moLineSymbol Then 'Outline color does not apply to lines
      End If
      If glbLayersNewData(j).Renderer = True Then 'If the layer is land use
         Set newDataLayer.Renderer = MapRendererNewData
         With newDataLayer.Renderer
            .ValueCount = 5
            .Field = "Type"
            'Set values for forest
            .Value(0) = "Forest"
            .Symbol(0).OutlineColor = moDarkGreen
            .Symbol(0).Style = moSolidFill
            .Symbol(0).Color = moDarkGreen
            'Set values for herbaceous
            .Value(1) = "Herbaceous"
            .Symbol(1).OutlineColor = moYellow
            .Symbol(1).Style = moSolidFill
            .Symbol(1).Color = moYellow
            'Set values for open water
            .Value(2) = "Open Water"
            .Symbol(2).OutlineColor = moBlue
            .Symbol(2).Style = moSolidFill
            .Symbol(2).Color = moBlue
            'Set values for disturbed
            .Value(3) = "Disturbed"
            .Symbol(3).OutlineColor = moRed
            .Symbol(3).Style = moSolidFill
            .Symbol(3).Color = moRed
            'Set values for edge/mixed
            .Value(4) = "Edge/Mixed"
            .Symbol(4).OutlineColor = moGray
            .Symbol(4).Style = moSolidFill
            .Symbol(4).Color = moGray
         End With
      End If
   Next j
End If
frmMain.mapDisplay.Layers.Add newDataLayer 'Add MapLayer to Layers collection
Set newDataLayer = Nothing 'Reset newDataLayer to add the next file
Next j
Else 'geoSetNewData Is Nothing
MsgBox "Error connecting to data", vbCritical
Exit Sub
End If

'Refresh link with the map legend
Call LinkLegend 'In the BasicOperations Module -- links legend control to the map

'Zoom the Map Display to the full extent
frmMain.mapDisplay.Extent = frmMain.mapDisplay.FullExtent

'Get rid of the form that adds data
frmMoreData.Hide

'Make it so that the Add Data button is not still pressed
frmMain.tbrMapControl.Buttons(5).Value = tbrUnpressed

DataConnectNewData.Disconnect
End Sub

frmModel Form

Option Explicit
Option Base 0

Private Sub Form_Load()

' Set the form's position on the screen
frmModel.Left = frmMain.Left
frmModel.Top = frmMain.Top

' Set "none" as the default models selected
chkHydroNone.Value = 1
chkEconNone.Value = 1
chkEcolNone.Value = 1
End Sub

' If the user selects a model, remove the check from "none"
Private Sub chkHydroHSPF_Click()
If chkHydroHSPF.Value = 1 Or chkHydroModflow.Value = 1 Or _
chkHydroHydraulics.Value = 1 Then
chkHydroNone.Value = 0
End If
End Sub

' If the user selects a model, remove the check from "none"
Private Sub chkHydroModflow_Click()
If chkHydroHSPF.Value = 1 Or chkHydroModflow.Value = 1 Or _
chkHydroHydraulics.Value = 1 Then
chkHydroNone.Value = 0
End If
End Sub

' If the user selects "none" for a model, remove checks from other models
Private Sub chkHydroNone_Click()
    If chkHydroNone.Value = 1 Then
        chkHydroHSPF.Value = 0
        chkHydroModflow.Value = 0
        chkHydroHydraulics.Value = 0
        chkHydroNone.Value = 1
    End If
End Sub
'If the user selects a model, remove the check from "none"
Private Sub chkEconTaxChg_Click()
    If chkEconTaxChg.Value = 1 Then
        chkEconNone.Value = 0
    End If
End Sub
'If the user selects a model, remove the check from "none"
Private Sub chkEconLandVal_Click()
    If chkEconTaxChg.Value = 1 Then
        chkEconNone.Value = 0
    End If
End Sub
'If the user selects a model, remove the check from "none"
Private Sub chkEconNone_Click()
    If chkEconNone.Value = 1 Then
        chkEconTaxChg.Value = 0
c    chkEconNone.Value = 1
    End If
End Sub
'If the user selects "none" for a model, remove checks from other models
Private Sub chkEcolFish_Click()
    If chkEcolFish.Value = 1 Or chkEcolBugs.Value = 1 Then
        chkEcolNone.Value = 0
    End If
    'if the user selects the fish model, then select the hydro model automatically.
    If chkEcolFish.Value = 1 Then chkHydroHSPF.Value = 1
End Sub
'If the user selects a model, remove the check from "none"
Private Sub chkEcolBugs_Click()
    If chkEcolFish.Value = 1 Or chkEcolBugs.Value = 1 Then
        chkEcolNone.Value = 0
    End If
End Sub
'If the user selects "none" for a model, remove checks from other models
Private Sub chkEcolNone_Click()
    If chkEcolNone.Value = 1 Then
        chkEcolFish.Value = 0
        chkEcolBugs.Value = 0
        chkEcolNone.Value = 1
    End If
End Sub
'When the user has decided what models to run
Private Sub cmdModelingSet_Click()
    If chkHydroHSPF.Value = 1 Then
        frmModel.Hide
        Call MakeHSPFFile
End If

If chkEconTaxChg.Value = 1 Then
    Call CalcSewerDist
    Call CalcWaterDist
    frmModel.Hide
    FrmEconInput.Show
    'The ok button in frmEconInput calls other Econ modeling subroutines
End If

If chkEcolFish.Value = 1 Then
    Call EstimateFishHealth
End If

Unload frmModel
End Sub

frmHelp Form

Private Sub Form_Load()
    'Set the form's position on the screen
    FrmEconInput.Left = Screen.Width / 20
    FrmEconInput.Top = Screen.Height / 20
End Sub

frmHydroReport Form

Private Sub Form_Load()
    'Set the form's position on the screen
    frmHydroReport.Left = frmMain.Left
    frmHydroReport.Top = frmMain.Top
End Sub

frmEconInput Form

Option Explicit

'This subroutine sends the user out of the Econ Input form and calls subroutines _
to produce economic results

Private Sub cmdHelp_Click()
    frmHelp.Show
    frmHelp.webHelp.Navigate App.Path & "\HelpFiles\EconInputHelp.html"
End Sub

Private Sub cmdOK_Click()
    fraEconInputs.Visible = False
    fraMakingSpreadsheet.Visible = True
    Call MakeEconExcelFile 'In EconOperations Module
    fraMakingSpreadsheet.Visible = False
    fraEconInputs.Visible = True
    FrmEconInput.Hide
    frmEconExcelView.Show
    frmEconExcelView.webViewExcel.Navigate App.Path & "\OutputFiles\GovtRevsExps.xls"
End Sub
Private Sub Form_Load()
    'Set the form's position on the screen
    FrmEconInput.Left = frmMain.Left
    FrmEconInput.Top = frmMain.Top
    'make the status message invisible
    fraMakingSpreadsheet.Visible = False
    '$220/acre, converted to per hectare value. Source: Use Value Methodology (Van Eenoo et al.)
    txtUseAssessAg.Text = Int(220 / 0.404687)
    '$120/acre, converted to per hectare value. Source: Use Value Methodology (Van Eenoo et al.)
    txtUseAssessFor.Text = Int(120 / 0.404687)
    '$2500/acre, converted to per hectare value. Source: Use Value Methodology (Van Eenoo et al.)
    txtUseAssessOS.Text = Int(2500 / 0.404687)
    'Roanoke County Treasurer says mkt value assessment is 100%
    txtMktAssess.Text = 100
    'Roanoke County Treasurer says tax rate is 1.13%
    txtTaxRate = 1.13
    'Default for predevelopment land is 100% assessed at use value
    txtPortionUseBL.Text = 100
End Sub

frmEconExcelView Form

Private Sub cmdHelp_Click()
    frmHelp.Show
    frmHelp.webHelp.Navigate App.Path & "\HelpFiles\EconSpreadsheetHelp.html"
End Sub

Private Sub Form_Load()
    'Set the position of the form on the screen
    frmEconExcelView.Left = Screen.Width / 60
    frmEconExcelView.Top = Screen.Height / 60
End Sub

frmFishReport Form

Private Sub cmdHelp_Click()
    frmHelp.Show
    frmHelp.webHelp.Navigate App.Path & "\HelpFiles\FishHelp.html"
End Sub

Private Sub Form_Load()
    'Set the form's position on the screen
    frmFishReport.Left = Screen.Width / 20
    frmFishReport.Top = Screen.Height / 20
End Sub