

```

VERSION 1.0 CLASS
BEGIN
  MultiUse = -1 'True
  Persistable = 0 'NotPersistable
  DataBindingBehavior = 0 'vbNone
  DataSourceBehavior = 0 'vbNone
  MTSTransactionMode = 0 'NotAnMTSObject
END
Attribute VB_Name = "cRouteModel"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = False
Attribute VB_Ext_KEY = "SavedWithClassBuilder6" , "Yes"
Attribute VB_Ext_KEY = "Top_Level" , "Yes"

' -----
' file: cRouteModel.cls
' author: Brendan Kidwell / GeoGraphics Lab
' date: 10 August 2001
'
' Contains the cRouteModel class which is the heart of the
' Bus ETA algorithm
' -----

' Usage Example:
'
' Dim RouteModelObject As New cRouteModel
' RouteModelObject.Zones = 7
' RouteModelObject.RouteName = "test"
'
' [Insert code to aquire data]
' RouteModelObject.NewData BusNum, ObservationTime, Zone
' ...
'
' [When the program wants a prediction:]
' Dim x as Date
' RouteModelObject.Predict 5, rm_INCREASING
' Do
'     x = RouteModel.Prediction
'     If x Then
'         [present x as a prediction time]
'     End If
' Loop While x

```

Option Explicit

```

' [private variables linked to properties of the object]
' how many Zones in this route
Private mZones As Integer
' the number of the route
Private mRouteName As String
' last output of Predict method
Private mPrediction As New Collection
' the current index in mPrediction
Private mPredictionEnum As Integer

```

```

' did the last NewData call move a bus to a new zone?
Private mIsNewZone As Boolean

' [other private variables]

' indicates that the model hasn't been initialized yet
Private ModelNotReady As Boolean

' the path to the data file for saving route data when the object is terminated
Private DataFile As String

' array of most recent times for passes through through each zone
Private ZoneTime() As Date

' collection of information on buses that are currently being tracked on this
' route
Private Buses As New Collection

Private Sub Class_Initialize()
    ' At startup, the model is not ready for input. Zones and RouteName
    ' properties must be set.

    ModelNotReady = True
End Sub

' On termination, the model saves its data for zone times to a file. The file
' name is determined by the RouteName property.
Private Sub Class_Terminate()

    Dim fnum As Integer, zone As Integer, Direction As Integer

    If ModelNotReady Then Exit Sub

    fnum = FreeFile
    Open DataFile For Output As fnum
    For zone = 1 To mZones
        For Direction = 0 To 1
            Print #fnum, Trim(zone) & "|" & Trim(Direction) & _
                "|" & ZoneTime(zone, Direction)
        Next
    Next
    Close fnum
End Sub

' This function is called after the Zones and RouteName properties have both
' been set. This function initializes the model with a saved data set if
' possible.
Public Sub Setup(Zones As Integer, Optional RouteName As String = "1")

    Dim fnum As Integer, fdata As String, farray() As String

    mZones = Zones
    mRouteName = RouteName
    DataFile = App.Path & "\route_" & mRouteName & ".dat"

    ModelNotReady = False
    ResetModel

```

```

' if the data file exists...
If Len(Dir(DataFile)) Then
    fnum = FreeFile
    Open DataFile For Input As fnum
    Do Until EOF(fnum)
        Line Input #fnum, fdata
        farray() = Split(fdata, "|")
        On Error Resume Next
        ZoneTime(farray(0), farray(1)) = farray(2)
        On Error GoTo 0
    Loop
    Close fnum
End If

End Sub

' This function is called by the NewData function when it needs to retrieve the
' appropriate member from the Buses collection. It returns either an existing
' member of the collection, or a newly created member.
Private Function GetBus(BusID As String) As cBus

    Dim b As cBus

    On Error GoTo GetBus_err
    Set GetBus = Buses(BusID)
    On Error GoTo 0
    Exit Function

    GetBus_err:
        Set b = New cBus
        b.BusID = BusID
        Buses.Add b, BusID
        Set GetBus = b
        Resume Next

End Function

' The RouteName property specifies a name for this instance of the class.
' RouteName is used to name the data file that is saved when the object
' terminates.
Public Property Let RouteName(ByVal i As String)
    mRouteName = i
    If mZones Then Setup Zones:=mZones, RouteName:=mRouteName
End Property
Public Property Get RouteName() As String
    RouteName = mRouteName
End Property

' The Zones property informs the object of how many zones there are in this
' route. Zones are numbered starting at 1.
Public Property Let Zones(ByVal i As Integer)
    mZones = i
    If Len(mRouteName) Then Setup Zones:=mZones, RouteName:=mRouteName
End Property
Public Property Get Zones() As Integer
    Zones = mZones
End Property

' The prediction property returns a predictions determined by the last call to
' the Predict method.
'
' Usage: x = RouteModelObject.Prediction

```

```

'         repeated calls return all the predictions in sequence, and a 0
'         value at the end
'
'         x = RouteModelObject.Prediction(y)
'         returns prediction number y out of the collection of predictions
'
' Predictions are returned as a time interval: How long will it be before
' the bus gets here?
Property Get Prediction(Optional ByVal i As Integer = 0) As Date
    Prediction = 0
    If mPrediction.Count > 0 Then
        If i = 0 Then
            If mPredictionEnum = 0 Then mPredictionEnum = 1
        ElseIf (i > 0) And (i <= mPrediction.Count) Then
            mPredictionEnum = i
        End If
        If mPredictionEnum <= mPrediction.Count Then
            Prediction = mPrediction(mPredictionEnum)
        End If
    End If
    mPredictionEnum = mPredictionEnum + 1
    If mPredictionEnum > mPrediction.Count + 1 Then mPredictionEnum = 0
End Property

' The PredictionCount property returns the number of predictions that were
' determined by the last call to the Predict method.
Property Get PredictionCount()
    PredictionCount = mPrediction.Count
End Property

' The IsNewZone property returns True if a recent call to the NewData method
' resulted in a bus moving from one zone to the next. The parent program must
' explicitly reset IsNewZone to False before it calls NewData and checks
' IsNewZone to see if a zone border was passed. This way, a batch of data can
' be fed into NewData and then the flag can be tested.
Property Let IsNewZone(i As Boolean)
    mIsNewZone = i
End Property
Property Get IsNewZone() As Boolean
    IsNewZone = mIsNewZone
End Property

' The ResetModel method clears known zone transit times.
Public Sub ResetModel()
    If ModelNotReady Then Exit Sub

    ReDim ZoneTime(1 To mZones, 0 To 1)
End Sub

' The NewData method is used to feed data into the model.
'
' Usage: RouteModelObject.NewData BusID, dtm, zone
'
'     BusID - The unique ID of this vehicle
'     dtm   - The time of this observation
'     zone  - The zone number associated with this position
'
' Zone must be computed by the parent program. A call to GIS software, such as
' Maptitude, with a geographic layer of all the zones loaded could find in
' which zone a particular latitude-longitude ordered pair belongs.
'
Public Sub NewData(BusID As String, dtm As Date, zone As Integer)

```

```

Dim Duration As Date, b As cBus

Debug.Print "NewData", BusID, dtm, zone

' ModelNotReady is True until Setup is called
If ModelNotReady Then Err.Raise vbObjectError + 1002, "prjPredict.cRouteModel", _
    "call to NewData and object not initialized."

' raise error if Zone parameter is out of bounds
If (zone < 1) Or (zone > mZones) Then
    Err.Raise vbObjectError + 1001, "prjPredict.cRouteModel", _
        "zone parameter of NewData out of bounds."
End If

' Find an existing or new member of the Buses collection
Set b = GetBus(BusID)

' take note of current zone for this bus and time of last known position
b.CurrZone = zone
b.LastPosTime = dtm

' initialize PrevZone if this is the first event for this bus
If b.PrevZone = 0 Then
    b.PrevZone = b.CurrZone
' if the zone has changed since last event for this bus, do some work:
ElseIf b.PrevZone <> b.CurrZone Then
    mIsNewZone = True
    ' if StartTime hasn't yet been set skip this bit. StartTime will be empty
    ' the first time a bus moves into a new zone.
    If b.StartTime Then
        b.EndTime = dtm
        Duration = b.EndTime - b.StartTime
        If (b.CurrZone - b.PrevZone) <= -1 Then
            b.CurrDirection = rm DECREASING ' zone decreasing
        ElseIf (b.CurrZone - b.PrevZone) >= 1 Then
            b.CurrDirection = rm INCREASING ' zone increasing
        End If
        If b.PrevZone = mZones Then
            b.CurrDirection = 1 ' turn around at far end of run
        End If
        If b.PrevZone = 1 Then
            b.CurrDirection = 0 ' turn around at near end of run
        End If

        ' Only record this zone time if it was less than 45 minutes
        If (Duration < #12:45:00 AM#) And (Duration > 0) Then
            ZoneTime(b.PrevZone, b.CurrDirection) = Duration
        End If

    End If
    b.PrevZone = b.CurrZone
    b.StartTime = dtm
End If

' If no new data comes in for this bus in 45 minutes, it will be dropped
' from memory.
b.Expires = Now + #12:45:00 AM#

' collect trash
TrashCollect

End Sub

```

```

' The Predict method takes an imaginary walk from a target zone, back along the
' path of the route, to all the buses that are running. Each time it passes
' a bus, it makes a note of how far away in time the bus is.
'
' Usage: RouteModelObject.Predict TargetZone, TargetDirection
'
' TargetZone - the zone where the user might be waiting
' TargetDirection - the direction the user wants to travel
'
' The method returns the number of predictions it found. See the Prediction
' property for a way to retrieve these predictions.
'
Public Function Predict(tZone As Integer, ByVal tDirection As rm_DIRECTION, _
    Optional PredictFor As Date = 0) As Integer

    Dim Direction As rm_DIRECTION, zone As Integer
    Dim b As cBus, bTime As Date, bPredict As Boolean, bI As Integer

    ' Clear mPrediction collection
    Do Until mPrediction.Count = 0
        mPrediction.Remove 1
    Loop

    ' ModelNotReady is True until Setup is called
    If ModelNotReady Then Err.Raise vbObjectError + 1002, _
        "prjPredict.cRouteModel", "call to Predict and object not initialized."

    ' assume prediction is false until we find otherwise
    Predict = 0

    ' ensure we have valid (Zone, Direction) pair
    If (tZone = mZones) And (tDirection = rm DECREASING) Then
        tDirection = rm INCREASING
    ElseIf (tZone = 1) And (tDirection = rm INCREASING) Then
        tDirection = rm DECREASING
    End If

    ' Default time to predict for Now.
    ' Normally, you only specify PredictFor if you're running the model on
    ' historical data.
    If PredictFor = 0 Then PredictFor = Now

    ' make a prediction for each bus in the Buses collection
    For Each b In Buses

        ' assume the prediction for this bus is good
        bPredict = True

        ' if it's been too long, prediction is flagged as bad
        If PredictFor - b.LastPosTime > #12:45:00 AM# Then bPredict = False

        ' start counting at the zone the bus is in now
        Direction = b.CurrDirection: zone = b.CurrZone

        ' add on half of the time for the first zone (the one the bus is in)
        If Not ((tZone = b.CurrZone) And (tDirection = b.CurrDirection)) Then
            If ZoneTime(zone, Direction) = 0 Then bPredict = False
            bTime = ZoneTime(zone, Direction) / 2
            Select Case Direction
                Case rm DECREASING
                    zone = zone - 1
            End Select
        End If
    Next b
End Function

```

```

        If zone < 1 Then
            zone = 2: Direction = rm_INCREASING
        End If
    Case rm_INCREASING
        zone = zone + 1
        If zone > mZones Then
            zone = mZones - 1: Direction = rm DECREASING
        End If
    End Select
End If
' add on the time of each intervening zone
Do Until (tZone = zone) And (tDirection = Direction)
    If ZoneTime(zone, Direction) = 0 Then bPredict = False
    bTime = bTime + ZoneTime(zone, Direction)
    Select Case Direction
        Case rm DECREASING
            zone = zone - 1
            If zone < 1 Then
                zone = 2: Direction = rm_INCREASING
            End If
        Case rm_INCREASING
            zone = zone + 1
            If zone > mZones Then
                zone = mZones - 1: Direction = rm DECREASING
            End If
        End Select
    Loop
' add on half of the time of the last zone (target zone)
If ZoneTime(zone, Direction) = 0 Then bPredict = False
bTime = bTime + ZoneTime(zone, Direction) / 2
' if we have a prediction for this bus, add to collection
If bPredict Then
    If mPrediction.Count = 0 Then
        mPrediction.Add bTime
    Else
        For bI = 1 To mPrediction.Count
            ' add predictions in ascending order
            If bTime < mPrediction(bI) Then
                mPrediction.Add bTime, , bI
                bPredict = False ' remember that the prediction was
                                ' added
            Exit For
        End If
    Next
    ' if the prediction hasn't been inserted somewhere, stick it
    ' on the end.
    If bPredict Then mPrediction.Add bTime, , mPrediction.Count
End If
End If
Next

' Initialize enumerator of the mPrediction collection, so a call to the
' Prediction property will start at the first prediction.
mPredictionEnum = 0
' Return the number of predictions found.
Predict = mPrediction.Count

```

End Function

```

' The TrashCollect method is called automatically at the end of a call to the
' NewData method. It looks through the Buses collection for members that

```

```
' haven't seen any new data for a long time, and discards them. This is to  
' facilitate new buses joining a route and old buses leaving a route through  
' the course of the day.  
Public Sub TrashCollect()
```

```
    Dim b As cBus
```

```
    For Each b In Buses
```

```
        If Now > b.Expires Then
```

```
            Buses.Remove b.BusID
```

```
        End If
```

```
    Next
```

```
End Sub
```