MODELING A FOREST FIRE: FIGHTING IT AND ITS EFFECT

NEW MEXICO ADVENTURES IN

SUPERCOMPUTING CHALLENGE

FINAL REPORT

APRIL 7, 2004

TEAM 009

ALBUQUERQUE ACADEMY

Project Member:

Punit Shah

Teacher:

Jim Mims

TABLE OF CONTENTS

EXECUTIVE SUMMERY

Forest fires can devastate communities with incredible speed. They have caused great damage in the past and will forever, unless some course of action is taken. This project aims to take the first steps, model the movement of the fire. By modeling that, fire fighters will be able to better fight the fire.

This project involved a major research portion. There has been a lot of research done in the past, but there still isn't enough information to make a model that would be perfectly accurate. Even the most trivial factors could dramatically change the course of the fire. Intense field research would need to be conducted in order to create the perfect program. Time and resources did not permit such research to be conducted, so I decided to model and create the program using approximate equations and formulas. Later, once field research has been done, they could be replaced with better and more accurate representations of the situation.

Forest fires are very hard to model; this may be why there are not many models of them. Once fires reach a certain size, they are able to create their own winds and weather, which could in turn allow them to expand. This is due in part to the pressure changes occurring in the vicinity as a result of the fire. Such problems as these arose as the mathematical model was developed. Therefore, instead of modeling all of the fire's factors at once, I took only wind and first incorporated it into the project. This way, I would have a starting point and also be able to expand on the original program.

At the beginning of the project, I intended to use Java, but later in the project, I switched to Visual Basic. This allowed for a much better and easier to program GUI.

INTRODUCTION

Fires have devastated regions of the world in terrible ways for hundreds of years. Man has attempted to conquer and overpower this force of nature, but never have we been totally successful. A method that is probably one of the most feasible today would be to model it and create a program that if was modeling a whole forest, would require the usage of a supercomputer.

Fires have proven to lead to very complex models for scientists, which may be one reason no real conclusion has been made in the field regarding how to effectively model the fire. Once a fire becomes of a certain size and the conditions are favorable to the fire, they turn into crown fires, which are the most intense form of fire. These fires are able to generate their own hurricane forced winds and weather due to the pressure changes occurring with the fire. This is the type of fire often associated with the words "forest fire." It is only imaginable how complicated the model of a crown fire is. It is also, unfortunately the main type of fire attempting to be fought.

A fire is quite simply stated, rapid oxidation. Therefore, in the case of fire, for it to sustain, three things must be present; oxygen, heat, and fuel. By removing any one of the necessary components of a fire, the fire is destroyed. There are, therefore, many ways to fight a fire, some of which are more effective than others.

This project attempts to explore some of the different methods used to fight fires. By creating a program in which a user could input parameters concerning a particular forest, he would be able to see what improvements could be made before a fire is sparked. He would also see what tactics would be available to them and how to go about fighting the fire.

METHOD

The first thing I did was greatly simplify the problem. As stated in the introduction, many factors can change the course of the fire. Even the smallest change in a factor could cause the fire to do something much different. The main factor that I determined through our research to have the largest effect on the fire was the wind. Not only does it determine what general direction the fire will go in, it also determines what intensity and velocity the fire burns at. It therefore also determines how high the flames are, which ultimately decides what sorts of barriers will be need to stop the fire.

The other factor that was determined to be important was terrain. In general, a fire will move up hills and other inclines. If it is stuck at the top of a hill, it will usually burn out. Terrain is harder to model compared to many of the other factors as it involves turning a previously two dimensional perception of the fire's world and converting it into three dimensions. This was one of the reasons that it was excluded from the preliminary code.

This project does not have any "true" equations but rather has some equations that seem to be somewhat true. There were two reasons I chose to do this. First, extensive field research would have to have been conducted, which I don't have the resources to do or the time to generate the correct equations. If field research was ever conducted, this method would allow us to replace the current equations with the correct equations. Secondly, many of the models that have been used by certain program that attempt to do some of the things that this project aims at doing involved very high, college level math that I did not have experience in.

Using the method outlined in the previous paragraph, it was easy to determine a starting position for writing our code. Also, after one factor had been modeled into the program, it was simple to add another factor. Each factor is written into its own method, which ultimately can connect up to the driver.

The program was developed using Visual Basic. Due to the programming language's high visual orientation and the programs necessity of graphical outputs for the user, it appeared and proved to be best to use Visual Basic.

At the time that the final report was due, the program had partially incorporated the wind factors into the large model of the whole project. See the pictures in Appendix A for screenshots of the program.

RESULTS

Using the program, I would have been able to determine the best of the fighting tactics in different situations. Every situation has a fighting method that is better fitted for it. By knowing this, the model of the fire and the recommendations to the user regarding fighting tactics would have been more accurate and complete.

The program would also show if the correct equations were in place, that it the course of a fire is somewhat predictable. It is known that fires do follow some sort of pattern, but to put it all together into a model that can actually show the progress of the fire is a large advancement. This would be a great advancement because if a perfect model was ever achieved, it would be possible to completely overpower this force of nature.

Some of the other results that might be generated from the program would relate to the effect of certain factors onto the big picture. Using this and comparing it to real data from fires that have really taken place, it would be possible to better perfect the equations already in place and further slim the margin of error.

CONCLUSIONS

By analyzing the results, I was able to see that it is quite hard to model a fire. There is only one thing that a user might want during a situation in which he/she is requesting information regarding a fires path: what should we do next? Therefore, though it may degrade accuracy by a small bit, certain factors can be excluded. The research indicated that though even the smallest change can affect the big picture, there is a point when a user can say that the results are “close enough.”

That is one of the reasons that a precision option was incorporated into the program so if a user wanted the program to be able to detect the status of every square inch or every 50 square feet.

Other conclusions that were made were regarding effective ways to model a fire. Now that I look at the method I used to model the fire, I see that it was probably the best way for this project and it is probably an achievable method for even a larger project regarding the topic. The method allows for the many factors to communicate effectively with each other and the driving method, which is important because many of the factors are related. For example, the amount of rain falling is highly related to the relative humidity.

ACHIEVEMENTS OF THE PROJECT

This project's strongest point was in creating a base for the programmer to work off of without the equations and formulas that are necessary for an accurate picture of what happens during a fire. Even without the data, the many methods related to each factor could effectively communicate with each other. If this project was ever moved to a grander scale with many people working on it, this would be a very productive way to produce the code.

ACKNOWLEDGEMENTS AND CITATIONS

I would like to thank Jim Mims, my coach for all of his time and effort. Before this project, I was not able to effectively manage such a large project and I was not even able to create and handle such a complicated problem in computer science. Now, with his help, I can do many of these actions with much greater speed and effectiveness.

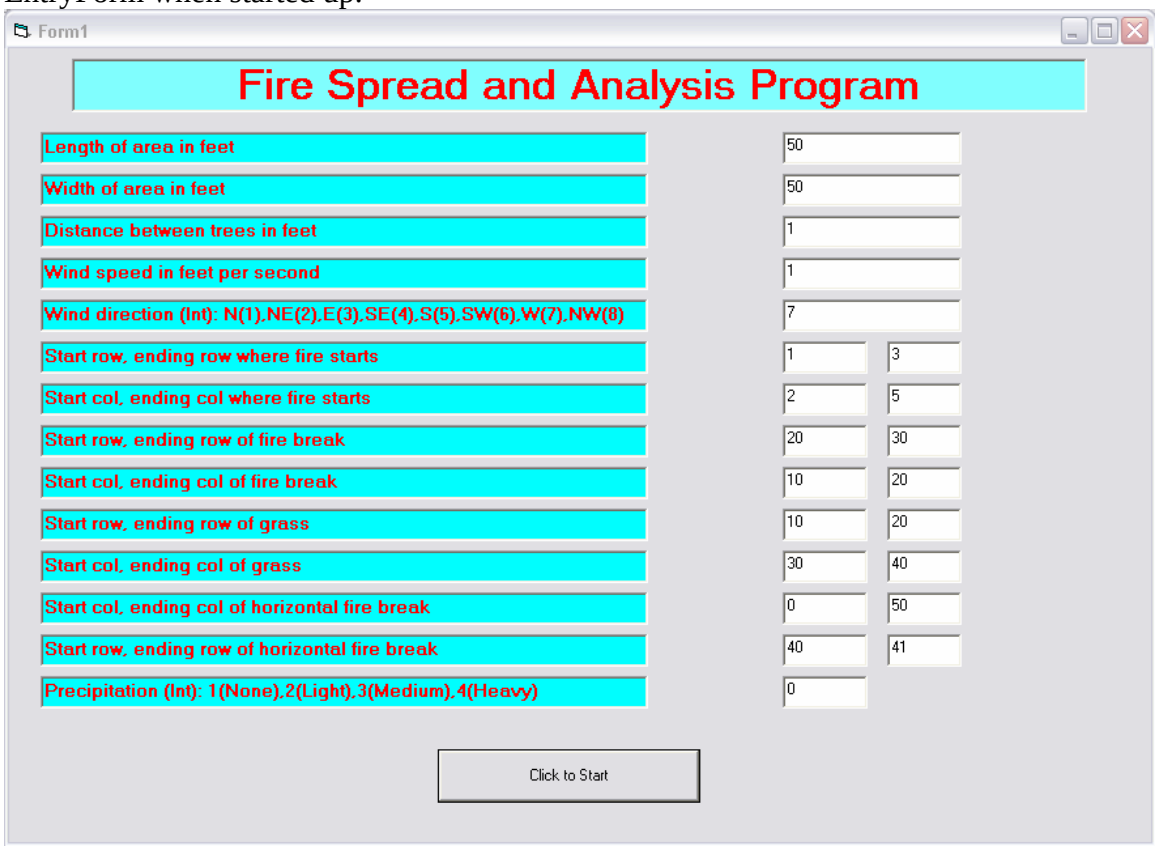
Portions of the code are courtesy of Jim Mims.

CITATIONS

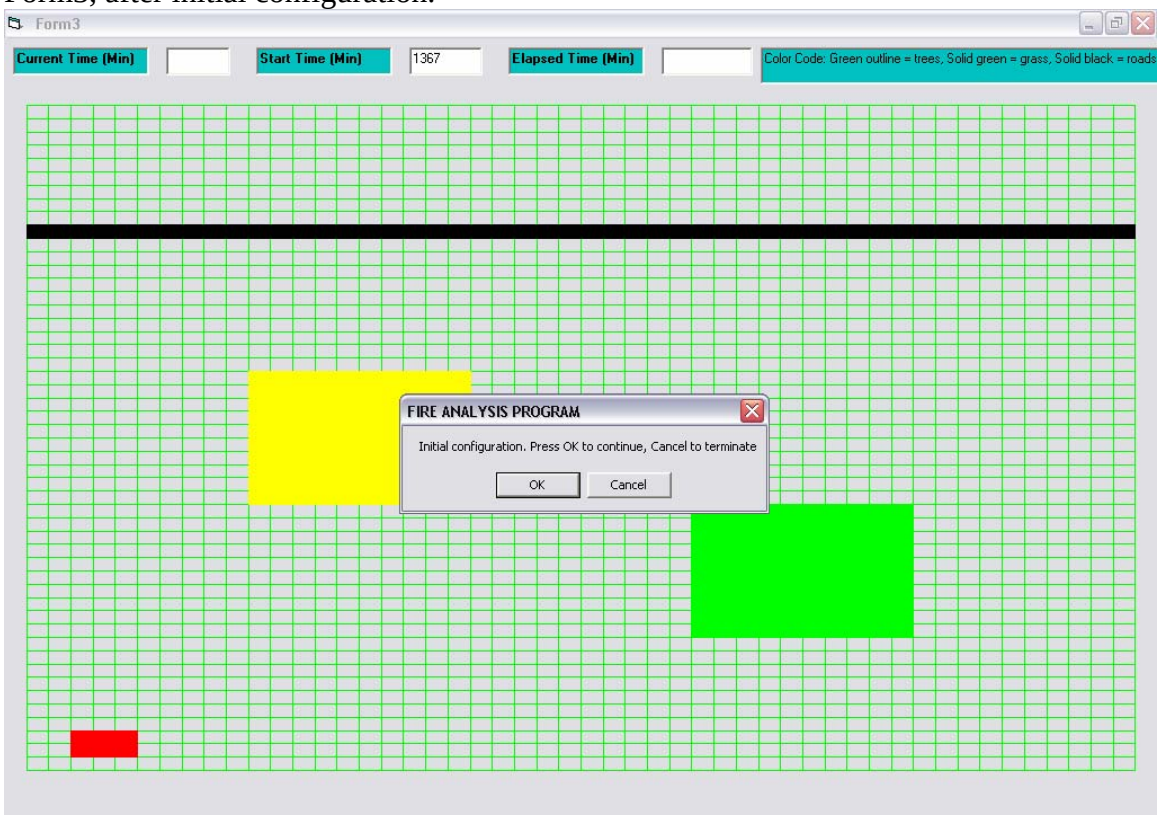
- Bonsor, Kevin. How Wildfires Work. Last edited unknown. Last viewed April 5, 2004. <http://science.howstuffworks.com/wildfire.htm>
- Finney, Mark A. FARSITE: Fire Area Simulator—Model Development and Evaluation. Last edited Nov. 32, 2003. Last Viewed April 5, 2004. <www.firelab.org/fbp/fbppubs/fbppdf/finney/fireareaall.pdf>
- Spidell, Rhonda. Lecture notes from a collection of lectures regarding fires. Late August to early October.

APPENDIX A: APPLICATION SCREEN SHOTS

EntryForm when started up.

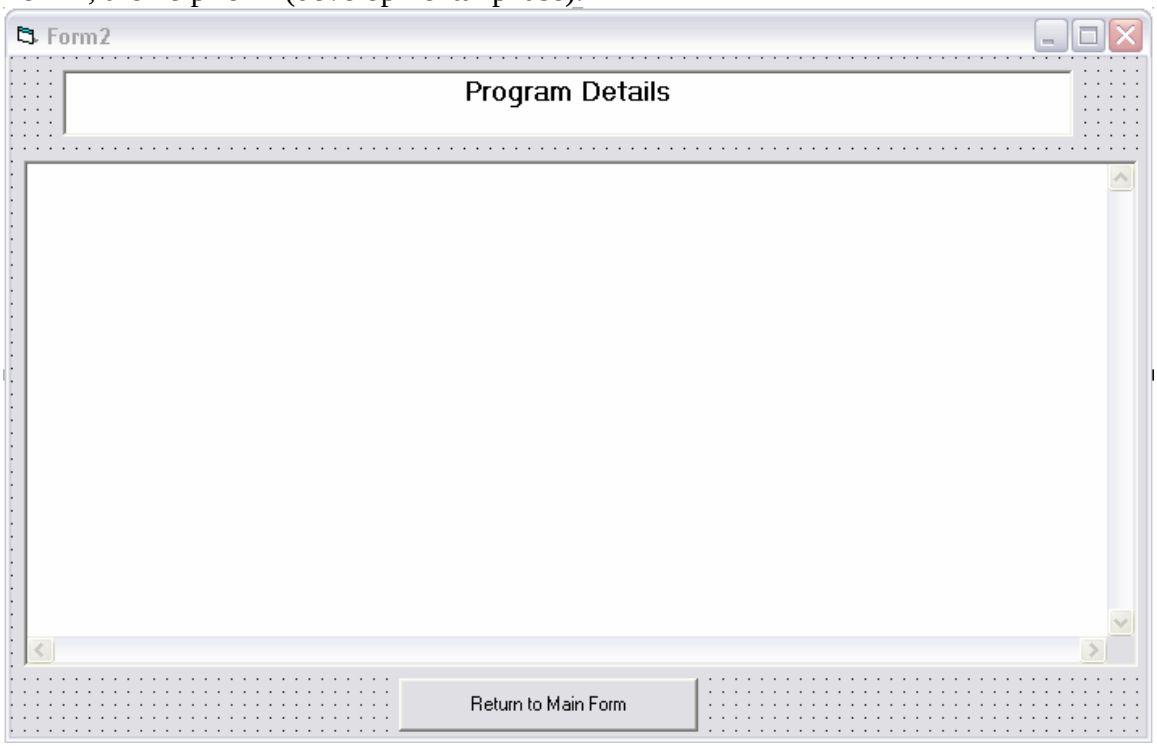


Form3, after initial configuration.



Form3, after run a few times.

Form2, the help form (developmental phase).



APPENDIX B: ENTRYFORM CODE

THE INFORMATION INPUT SCREEN

```
'GENERAL DECLARATIONS
Option Explicit                 'Forces declaration of all variables
Public FireBreak As New Forest
Public Road As New Forest
Public Grass As New Forest
Public Trees As New Forest
Public Fire As New Forest
Public WSpeed As Integer
Public WDirection As Integer

Dim FireTracking() As Forest
'NOTE: THIS VERSION USES A DUMB SWEEP, ALL THE WAY THROUGH
EACH TIME. CHANGE TO FIND AND
'FOCUS ON THE FIRE. THEN SWEEP OUTWARD FROM THE FIRE

'THE MAIN METHOD - EVENT DRIVEN
Private Sub Start_Click()

  'DIMENSION ALL VARIABLES USED
   'Note that input values are for the plotting grid reference. They refer to lines
   'With lines starting at 0 in both directions
   Dim i As Integer
   Dim j As Integer
   Dim length As Integer            'Length of the area, feet
   Dim width As Integer             'Width of the area, feet
   Dim Spacing As Integer           'Width of a cell, resolution, feet
   Dim WindSpeed As Integer         'Speed of the wind, FPS
   Dim FireStartCol As Integer      'Left boundary column where fire starts
   Dim FireEndCol As Integer        'Right boundary column where fire starts
   Dim FireStartRow As Integer      'Bottom boundary row where fire starts
   Dim FireEndRow As Integer        'Top boundary row where fire starts
   Dim WindDirection As Integer     'Direction of the wind, 8 possibilities
   Dim FBStartRow As Integer        'Start row of fire break
   Dim FBEndRow As Integer          'End row of fire break
   Dim FBStartCol As Integer        'Start col of fire break
   Dim FBEndCol As Integer          'End col of fire break
   Dim GrassStartRow As Integer     'Start row of grass
   Dim GrassEndRow As Integer       'End row of grass
   Dim GrassStartCol As Integer     'Start col of grass
   Dim GrassEndCol As Integer       'End col of grass
   Dim HFBStartCol As Integer       'Start col of horizontal fire break
```

```
    Dim HFBEndCol As Integer              'End col of horizontal fire break
    Dim HFBStartRow As Integer            'Start row of horizontal fire break
    Dim HFBEndRow As Integer              'End row of horizontal fire break
    Dim LeftColRoad As Integer            'Left column of vertical road
    Dim Precipitation As Integer          'Rain

    'CAPTURE VALUES OF THE VARIABLES FROM USER INPUT
    length = Val(LengthIn.Text)
    width = Val(WidthIn.Text)
    Spacing = Val(SpacingIn.Text)
    WindSpeed = Val(WindSpeedIn.Text)
    WSpeed = Val(WindSpeedIn.Text)
    FireStartRow = Val(FireStartRowIn.Text)
    FireEndRow = Val(FireEndRowIn.Text)
    FireStartCol = Val(FireStartColIn.Text)
    FireEndCol = Val(FireEndColIn.Text)
    WindDirection = Val(WindDirectionIn.Text)
    WDirection = Val(WindDirectionIn.Text)
    FBStartRow = Val(SRFB.Text)
    FBEndRow = Val(ERFB.Text)
    FBStartCol = Val(SCFB.Text)
    FBEndCol = Val(ECFB.Text)
    GrassStartRow = Val(SRGrass.Text)
    GrassEndRow = Val(ERGrass.Text)
    GrassStartCol = Val(SCGrass.Text)
    GrassEndCol = Val(ECGrass.Text)
    HFBStartRow = Val(HFBStartRowIn.Text)
    HFBEndRow = Val(HFBEndRowIn.Text)
    HFBStartCol = Val(HFBStartColIn.Text)
    HFBEndCol = Val(HFBEndColIn.Text)
    Precipitation = Val(PrecipitationIn.Text)       'Rain

    Form3.AutoRedraw = True

    'CALL METHOD TO CREATE GRID FOR VISUAL REPRESENTATION
    Call CreateGrid(length, width, Spacing, FireStartCol, FireStartRow, FireEndCol, _
    FireEndRow, FBStartRow, FBEndRow, FBStartCol, FBEndCol, GrassStartRow,
GrassEndRow, _
    GrassStartCol, GrassEndCol, LeftColRoad, HFBStartRow, HFBEndRow,
HFBStartCol, _
    HFBEndCol)

    'THE MSGBOX METHOD PAUSES THE PROGRAM SO THE THAT USER CAN
SEE THE INITIAL CONFIGURATION
    'BEFORE CONTINUING
    Dim response As Integer
```

```
        response = MsgBox("Initial configuration. Press OK to continue, Cancel to
terminate", _
        vbOK, "FIRE ANALYSIS PROGRAM")
        Select Case response
           Case vbCancel
             End
           Case vbOK
        End Select

    'CALL METHOD TO INITIALIZE OBJECTS
    Call InitializeObjects(FireBreak, Road, Grass, Trees, Fire, WindDirection,
FBStartRow, _
    FBEndRow, FBStartCol, FBEndCol, GrassStartRow, GrassEndRow, GrassStartCol, _
    GrassEndCol, Precipitation, HFBStartRow, HFBEndRow, HFBStartCol, HFBEndCol)

    'CALL METHOD TO INITIALIZE FIRE TRACKING ARRAY WITH
APPROPRIATE OBJECTS
    Call FireTrackingArray(FireBreak, Road, Grass, Trees, Fire, length, width,
FireStartCol, _
    FireEndCol, FireStartRow, FireEndRow, GrassStartRow, GrassEndRow,
GrassStartCol, _
    GrassEndCol, FBStartRow, FBEndRow, FBStartCol, FBEndCol, HFBStartRow,
HFBEndRow, _
    HFBStartCol, HFBEndCol)


End Sub

Private Sub InitializeObjects(ByRef FireBreak, ByRef Road, ByRef Grass, ByRef Trees,
_
ByRef Fire, ByVal WindDirection As Integer, ByVal FBStartRow As Integer, ByVal
FBEndRow As Integer, _
ByVal FBStartCol As Integer, ByVal FBEndCol As Integer, ByVal GrassStartRow As
Integer, _
ByVal GrassEndRow, ByVal GrassStartCol, ByVal GrassEndCol, ByVal Precipitation
As Integer, _
HFBStartRow, HFBEndRow, HFBStartCol, HFBEndCol)

    'Dimension the parameters
    'Fire Status in a cell, True or False
    Dim F As Boolean
    'Status of vegetation: 1: Trees, 2: Grass, 3: Fire Break
    Dim V As Integer
    'Precipitation: 0: No rain, 1: light rain, 2: medium rain, 3: heavy rain
    Dim P As Integer
    'Moisture: 0: Dry, 1: slightly moist, 2: moderately moist, 3: very moist
```

```
Dim M As Integer
'Slope of terrain: 1: level, 2: slightly sloping, 3: steep slope
Dim S As Integer
'Wind speed, integer between 0 and 100 FPS
Dim W As Integer
'Wind direction: 1: N, 2: NE, 3: E, 4: SE, 5: S, 6: SW, 7: W, 8: NW
Dim WD As Integer
'Can Burn or not: True or False
Dim CB As Boolean

'Initialize the FireBreak object
F = False
V = 1
P = 1
M = 1
S = 1
W = 10
WD = 3
CB = False
Call FireBreak.SetData(F, V, P, M, S, W, WD, CB)

'Initialize the Trees object
F = False
V = 1
P = 1
M = 1
S = 1
W = 10
WD = 3
CB = True
Call Trees.SetData(F, V, P, M, S, W, WD, CB)

'Initialize the Grass object
F = False
V = 1
P = 1
M = 1
S = 1
W = 10
WD = 3
CB = True
Call Grass.SetData(F, V, P, M, S, W, WD, CB)

'Initialize the Roads object
F = False
V = 1
```

```
    P = 1
    M = 1
    S = 1
    W = 10
    WD = 3
    CB = False
    Call Road.SetData(F, V, P, M, S, W, WD, CB)

    'Initialize the Fire object
    F = True
    V = 1
    P = 1
    M = 1
    S = 1
    W = 10
    WD = 3
    CB = True
    Call Fire.SetData(F, V, P, M, S, W, WD, CB)

End Sub


'METHOD TO CREATE GRID FOR VISUAL REPRESENTATION
Private Sub CreateGrid(ByVal length As Integer, ByVal width As Integer, _
ByVal Spacing As Integer, ByVal FireStartCol As Integer, ByVal FireStartRow As
Integer, _
ByVal FireEndCol As Integer, ByVal FireEndRow As Integer, ByVal FBStartRow As
Integer, _
ByVal FBEndRow As Integer, ByVal FBStartCol As Integer, ByVal FBEndCol As
Integer, _
ByVal GrassStartRow As Integer, ByVal GrassEndRow As Integer, ByVal GrassStartCol
As Integer, _
ByVal GrassEndCol As Integer, ByVal LeftColRoad As Integer, HFBStartRow,
HFBEndRow, _
HFBStartCol, HFBEndCol)

    Dim i As Integer                    'Loop variable
    Dim j As Integer                    'Loop variable
    Form3.WindowState = 2                   'Maximize the window
    StartTime = Hour(Time) * 60 + Minute(Time)      'Calculate starting time from system
clock
    Form3.Text4.Text = StartTime                'Place start time on form
    Form3.Show                          'Show the form

    'Scale output form, boundaries (equal to spacing) added for readability
    Form3.Scale (-Spacing, width + 5 * Spacing)-(length + Spacing, -5 * Spacing)
```

```
    'Initialize all cells to green, occupied by trees
    Dim count As Integer
    count = 0
    For i = 0 To width - 1 Step Spacing
        count = count + 1
        For j = 0 To length Step Spacing
            Form3.Line (i, j)-(i + Spacing, count * Spacing), vbGreen, B
        Next j
    Next i

    'Change fire start cell to solid red
    Form3.Line (FireStartCol * Spacing, FireStartRow * Spacing)-(FireEndCol * Spacing,
_
    FireEndRow * Spacing), vbRed, BF

    'Draw the fire break cells solid yellow
    Form3.Line (FBStartCol * Spacing, FBStartRow * Spacing)-(FBEndCol * Spacing, _
    FBEndRow * Spacing), vbYellow, BF

    'Draw the grass cells solid green
    Form3.Line (GrassStartCol * Spacing, GrassStartRow * Spacing)-(GrassEndCol *
Spacing, _
    GrassEndRow * Spacing), vbGreen, BF

    'Draw the horizontal road in solid black
    Form3.Line (HFBStartCol * Spacing, HFBStartRow * Spacing)- _
    (HFBEndCol * Spacing, HFBEndRow * Spacing), vbBlack, BF

End Sub


'METHOD TO CREATE ARRAY FOR STATUS TRACKING
Public Sub FireTrackingArray(FireBreak, Road, Grass, Trees, Fire, length, width,
FireStartCol, _
    FireEndCol, FireStartRow, FireEndRow, GrassStartRow, GrassEndRow,
GrassStartCol, _
    GrassEndCol, FBStartRow, FBEndRow, FBStartCol, FBEndCol, HFBStartRow,
HFBEndRow, _
    HFBStartCol, HFBEndCol)

Dim i As Integer
Dim j As Integer

ReDim FireTracking(width, length) As Forest     'Creates array of type Forest
```

```
'Initialize each element of array as reference to Forest objects
For i = 0 To width - 1
   For j = 0 To length - 1
      Set FireTracking(i, j) = New Forest
   Next j
Next i


'Set each element to Tree object, will be modified subsequently below
For i = 0 To width - 1
   For j = 0 To length - 1
      Set FireTracking(i, j) = Trees
   Next j
Next i

'Set horizontal fire break elements
For i = HFBStartRow To HFBEndRow
   For j = 0 To length - 1
      Set FireTracking(i, j) = Road
   Next j
Next i


'Set Fire Break elements
For i = FBStartRow To FBEndRow
   For j = FBStartCol To FBEndCol
      Set FireTracking(i, j) = FireBreak
   Next j
Next i

'Set Grass elements
For i = GrassStartRow To GrassEndRow
   For j = GrassStartCol To GrassEndCol
      Set FireTracking(i, j) = Grass
   Next j
Next i

 'Set Fire Start Location.
For i = FireStartRow To FireEndRow - 1
   For j = FireStartCol To FireEndCol - 1
      Set FireTracking(i, j) = Fire
   Next j
Next i

Dim count As Integer
Dim output As Integer
```

```
    output = 0
    count = 0
    For i = 0 To width - 1
        For j = 0 To length - 1
            If FireTracking(i, j).FireStatus = True Then
                count = count + 1
            End If
        Next j
    Next i

End Sub


'METHOD TO SWEEP THROUGH FIRETRACKING ARRAY TO CHECK FOR FIRE
AND UPDATE PLOT
'THIS METHOD IS CALLED AND CONTROLLED BY A TIMER
'IT IS IN THE TIMER CODE AND CONTAINS A LOOP

Public Sub StatusCheck()
Dim width As Integer
Dim length As Integer
Dim Spacing As Integer
width = Val(WidthIn.Text)       'cannot be passed with timer control
length = Val(LengthIn.Text)     'cannot be passed with timer control
Spacing = Val(SpacingIn.Text)
Dim r As Integer
Dim i As Integer
Dim j As Integer


'LEFT TOP BOUNDARY CELL
    For i = 0 To 0
        For j = 0 To 0
            If FireTracking(i, j).FireStatus = False And _
                FireTracking(i, j).CanBurn = True And _
                FireTracking(i, j + 1).FireStatus = True Or _
                FireTracking(i + 1, j).FireStatus = True Or _
                FireTracking(i + 1, j + 1).FireStatus = True _
                Then
                    Call UpDateGrid(i, j, width, length, Spacing)
                    Let FireTracking(i, j).FireStatus = True
            End If
        Next j
    Next i

'TOP BOUNDARY CELLS LESS THE 2 TOP BORDER CELLS
```

```
   For i = 0 To 0
      For j = 1 To length - 2
         If FireTracking(i, j).FireStatus = False And _
            FireTracking(i, j).CanBurn = True And _
            FireTracking(i, j - 1).FireStatus = True Or _
            FireTracking(i + 1, j - 1).FireStatus = True Or _
            FireTracking(i + 1, j).FireStatus = True Or _
            FireTracking(i + 1, j + 1).FireStatus = True Or _
            FireTracking(i, j + 1).FireStatus = True _
            Then
               Call UpDateGrid(i, j, width, length, Spacing)
               Let FireTracking(i, j).FireStatus = True
         End If
      Next j
   Next i

'RIGHT TOP BOUNDARY CELL
   For i = 0 To 0
      For j = length - 1 To length - 1
         If FireTracking(i, j).FireStatus = False And _
            FireTracking(i, j).CanBurn = True And _
            FireTracking(i, j - 1).FireStatus = True Or _
            FireTracking(i + 1, j - 1).FireStatus = True Or _
            FireTracking(i + 1, j).FireStatus = True _
            Then
               Call UpDateGrid(i, j, width, length, Spacing)
               Let FireTracking(i, j).FireStatus = True
         End If
      Next j
   Next i

'LEFT BOUNDARY CELLS LESS THE CORNERS
   For i = 1 To width - 2
      For j = 0 To 0
         If FireTracking(i, j).FireStatus = False And _
            FireTracking(i, j).CanBurn = True And _
            FireTracking(i - 1, j).FireStatus = True Or _
            FireTracking(i + 1, j).FireStatus = True Or _
            FireTracking(i, j + 1).FireStatus = True Or _
            FireTracking(i - 1, j + 1).FireStatus = True Or _
            FireTracking(i + 1, j + 1).FireStatus = True _
            Then
               Call UpDateGrid(i, j, width, length, Spacing)
               Let FireTracking(i, j).FireStatus = True
         End If
      Next j
```

```
   Next i

'NON-BOUNDARY CELLS: All cells i, j, except cells along a boundary have 8
neighbors

   For i = 1 To (width - 2)            'non-boundary cells
      For j = 1 To (length - 2)        'non-boundary cells

         If FireTracking(i, j).FireStatus = False And _
            FireTracking(i, j).CanBurn = True And _
            FireTracking(i + 1, j).FireStatus = True Or _
            FireTracking(i - 1, j).FireStatus = True Or _
            FireTracking(i, j + 1).FireStatus = True Or _
            FireTracking(i, j - 1).FireStatus = True Or _
            FireTracking(i - 1, j - 1).FireStatus = True Or _
            FireTracking(i + 1, j + 1).FireStatus = True Or _
            FireTracking(i - 1, j + 1).FireStatus = True Or _
            FireTracking(i + 1, j - 1).FireStatus = True _
            Then
               Call UpDateGrid(i, j, width, length, Spacing)
               Let FireTracking(i, j).FireStatus = True
            End If
      Next j
   Next i

'RIGHT BOUNDARY CELLS LESS THE CORNERS
   For i = 1 To width - 2
      For j = length - 1 To length - 1
         If FireTracking(i, j).FireStatus = False And _
            FireTracking(i, j).CanBurn = True And _
            FireTracking(i - 1, j - 1).FireStatus = True Or _
            FireTracking(i, j - 1).FireStatus = True Or _
            FireTracking(i + 1, j - 1).FireStatus = True Or _
            FireTracking(i + 1, j).FireStatus = True Or _
            FireTracking(i - 1, j).FireStatus = True _
            Then
               Call UpDateGrid(i, j, width, length, Spacing)
               Let FireTracking(i, j).FireStatus = True
         End If
      Next j
   Next i

'LEFT BOTTOM BOUNDARY CELL
   For i = width - 1 To width - 1
      For j = 0 To 0
         If FireTracking(i, j).FireStatus = False And _
```

```
            FireTracking(i, j).CanBurn = True And _
            FireTracking(i - 1, j).FireStatus = True Or _
            FireTracking(i - 1, j + 1).FireStatus = True Or _
            FireTracking(i, j + 1).FireStatus = True _
            Then
               Call UpDateGrid(i, j, width, length, Spacing)
               Let FireTracking(i, j).FireStatus = True
         End If
      Next j
   Next i

'BOTTOM BOUNDARY CELLS LESS THE LEFT AND RIGHT CORNERS
   For i = width - 1 To width - 1
      For j = 1 To length - 2
         If FireTracking(i, j).FireStatus = False And _
            FireTracking(i, j).CanBurn = True And _
            FireTracking(i, j - 1).FireStatus = True Or _
            FireTracking(i - 1, j - 1).FireStatus = True Or _
            FireTracking(i - 1, j).FireStatus = True Or _
            FireTracking(i - 1, j + 1).FireStatus = True Or _
            FireTracking(i, j + 1).FireStatus = True _
            Then
               Call UpDateGrid(i, j, width, length, Spacing)
               Let FireTracking(i, j).FireStatus = True
         End If
      Next j
   Next i

'RIGHT BOTTOM BOUNDARY CELL
   For i = width - 1 To width - 1
      For j = length - 1 To length - 1
         If FireTracking(i, j).FireStatus = False And _
            FireTracking(i, j).CanBurn = True And _
            FireTracking(i - 1, j - 1).FireStatus = True Or _
            FireTracking(i, j - 1).FireStatus = True Or _
            FireTracking(i - 1, j).FireStatus = True _
            Then
               Call UpDateGrid(i, j, width, length, Spacing)
               Let FireTracking(i, j).FireStatus = True
         End If
      Next j
   Next i
End Sub


'METHOD TO UPDATE THE FIRE PLOTTING GRID
```

```
'Conversions have to be made to account for fact that array goes from 0 to length - 1 and
width - 1, starting
'upper left and the plotting goes from 0 to Length and Width starting bottom left.

Private Sub UpDateGrid(ByVal i As Integer, ByVal j As Integer, ByVal width As
Integer, _
ByVal length As Integer, ByVal Spacing As Integer)

Dim response As Integer
      response = MsgBox("Fire progression shown. Press OK to continue, Cancel to
terminate", _
      vbOK, "FIRE ANALYSIS PROGRAM")
      Select Case response
         Case vbCancel
           End
         Case vbOK
      End Select

   Dim x As Integer
   Dim y As Integer
   Dim ISet As Integer
   Dim JSet As Integer


   'Start x or col: x = j * Spacing
   'End x or col: x = (j + 1) * Spacing

   'Start y or row: y = i * Spacing
   'End y or row: y = (i + 1) * Spacing
   'Dim WindDirection As Integer



   If (WDirection = 1) Then 'North
   ISet = i - 1
   JSet = j
   End If

   If (WDirection = 2) Then 'Northeast
   ISet = i - 1
   JSet = j + 1
   End If

   If (WDirection = 3) Then 'east
   ISet = i
   JSet = j + 1
```

```
    End If

    If (WDirection = 4) Then 'se
    ISet = i + 1
    JSet = j + 1
    End If

    If (WDirection = 5) Then 's
    ISet = i + 1
    JSet = j
    End If

    If (WDirection = 6) Then 'sw
    ISet = i + 1
    JSet = j - 1
    End If

    If (WDirection = 7) Then 'w
    ISet = i
    JSet = j - 1
    End If

    If (WDirection = 8) Then 'nw
    ISet = i - 1
    JSet = j - 1
    End If



    Form3.Line (i * Spacing, j * Spacing)-((JSet) * Spacing, ((ISet) * Spacing)), vbRed,
BF


End Sub
```

APPENDIX C: FORM2 CODE

THE HELP FORM

```
Private Sub Command1_Click()
   EntryForm.Show
End Sub
```

APPENDIX D: FORM3 CODE

THE VISUALIZATION ENVIRONMENT

```
Private Sub Timer1_Timer()
   Form3.Text1.Text = Hour(Time) * 60 + Minute(Time)
   Form3.Text6.Text = Hour(Time) * 60 + Minute(Time) - StartTime
End Sub


'TIMER TO CALL STATUSCHECK METHOD AT 1 SECOND INTERVALS
Private Sub Timer2_Timer()
   EntryForm.StatusCheck
End Sub
```