Yellowstone Eruption Ash Travel

New Mexico
Supercomputing Challenge
Final Report
April 2, 2014

Team 149
Taos High School

Team Members:
Adrian Hernandez
Ricardo Trujillo
Scott Nevels
Noe Garcia

Teachers:
Ms.Galligan
Mr. Gilroy

Table of Contents

Executive Summary

The Yellowstone Volcano or Yellowstone Caldera last major eruption occurred around 64,000 years ago and is said to have covered a great part of the western half of North America. Yellowstone has an estimated 600,000 year eruption cycle, the last eruption was over 640,000 years ago. Our question is what would happen if the Yellowstone Caldera were to erupt today? Where would the ash go? Our computer model is designed to predict how far the ash would travel if Yellowstone would to erupt. The program could also help predict which cities and towns need to be evacuated first and what cities are in more danger. The programs was made in Net logo and consist of a visual map that shows North and South America the map is used as a visual to show the area covered by the ash.

Problem

Yellowstone caldera is considered one of the largest volcanoes in the world if it ever erupts it is believed that it will impact the whole world. The program that we designed is programed to get an estimate of how bad the eruption will be by simulating how far the ash will travel.If the program is accurate it will be able to help predict which areas are more likely to be affected by the ash. If we know where the ash will go the area which will be affected can be evacuated to a safer area where the ash will not reach.

Research on yellowstone

Yellowstone is a colossal volcano in Wyoming. Yellowstone has a huge reservoir of magma underneath it. This reservoir has the potential of spewing more than 240 cubic miles of magma into America. Yellowstone has the potential to send 2,000 times more Matter into the sky than Mount St. Helens. It can produce an ash cloud that can stretch

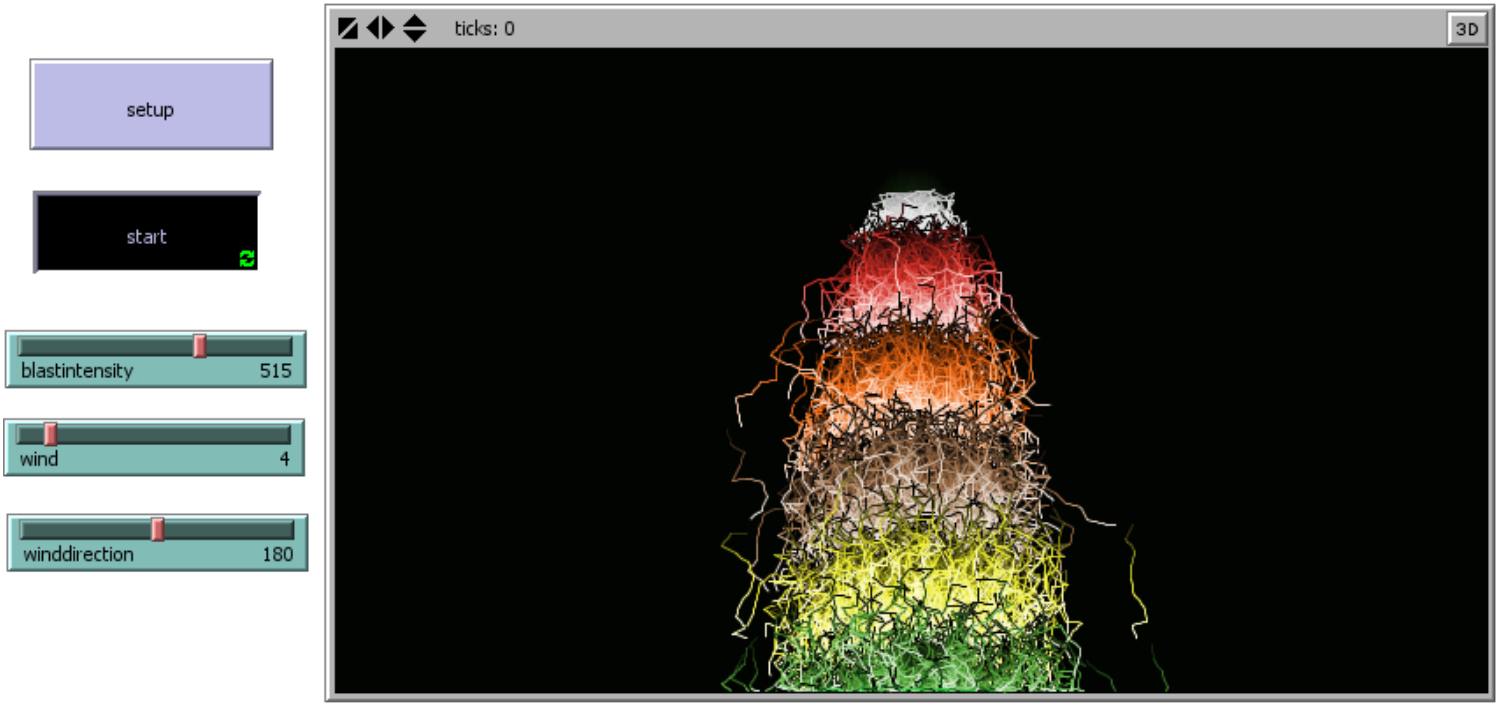
from Wyoming to the east coast. Scientists say large amounts of ash and pulverized rock from the eruption will get lofted into the atmosphere and then fall back slowly to Earth. The Yellowstone volcano's first major eruption occurred 2.06 million years ago, another followed 76 thousand years later. But the fear of an explosion may be a little farfetched according to some of the scientists. Some scientists even think that the giant volcano might even be dying. The last eruption happened 70,000 years ago which some scientists believe was a relative poof compared to the climate-changing, destructive eruptions that happened hundreds of thousands of years ago.

Program

The first program that we had was a basic model of ash spreading. The model then needed a variable so we added wind direction which set the heading of the agents to either one of the four directions. Another variable that we added in to our first model was the blast intensity which controlled how many agents or ash particles we were using. The model was looking good but now we wanted a visual such as a map which could show the area that was being covered. First we used a simple elevation code which showed two mountains. The mountains were not a great visual so we then decided to try and import a picture of a map using the image import sample model. The picture was okay but it was not very big and accurate. We finally decided to look further into the samples library and found a GIS map sample which we decided to use. The Map was good because it covered the whole interface net logo screen and was accurate. The program that had the map also had useful stuff like rivers and cities which we could use to see what rivers may be affected by the ash. Our first code of the ash spread was then combined with the GIS map Code then we set the Yellowstone

volcano on a patch that fits with the location of Yellowstone on the GIS map. An approximate location is used to select the patch of the volcano. The patch that is selected will be used as a start from where the ash or agents will begin to spread from. The agents are setup in the patch in the setup section of the code where it commands the agents to set their coordinates to the xy location of where the Yellowstone patch is located .The agents that are used in the model are set to the shape of a cloud which are found in the net logo art library. The agents are set to the cloud shape in the setup part of the code. The agents spread starting from the patch by going in the direction that the wind is going. The ash agents are also given a random size of up to three. The size of the ash is what determines how far the ash particle will go. Smaller ash particles will go farther than bigger ash particles. The ash that is spreading on the map is programed to change the color of the patch which it is on to red. The red patches are then counted and displayed in a monitor to see how many were covered by the ash.

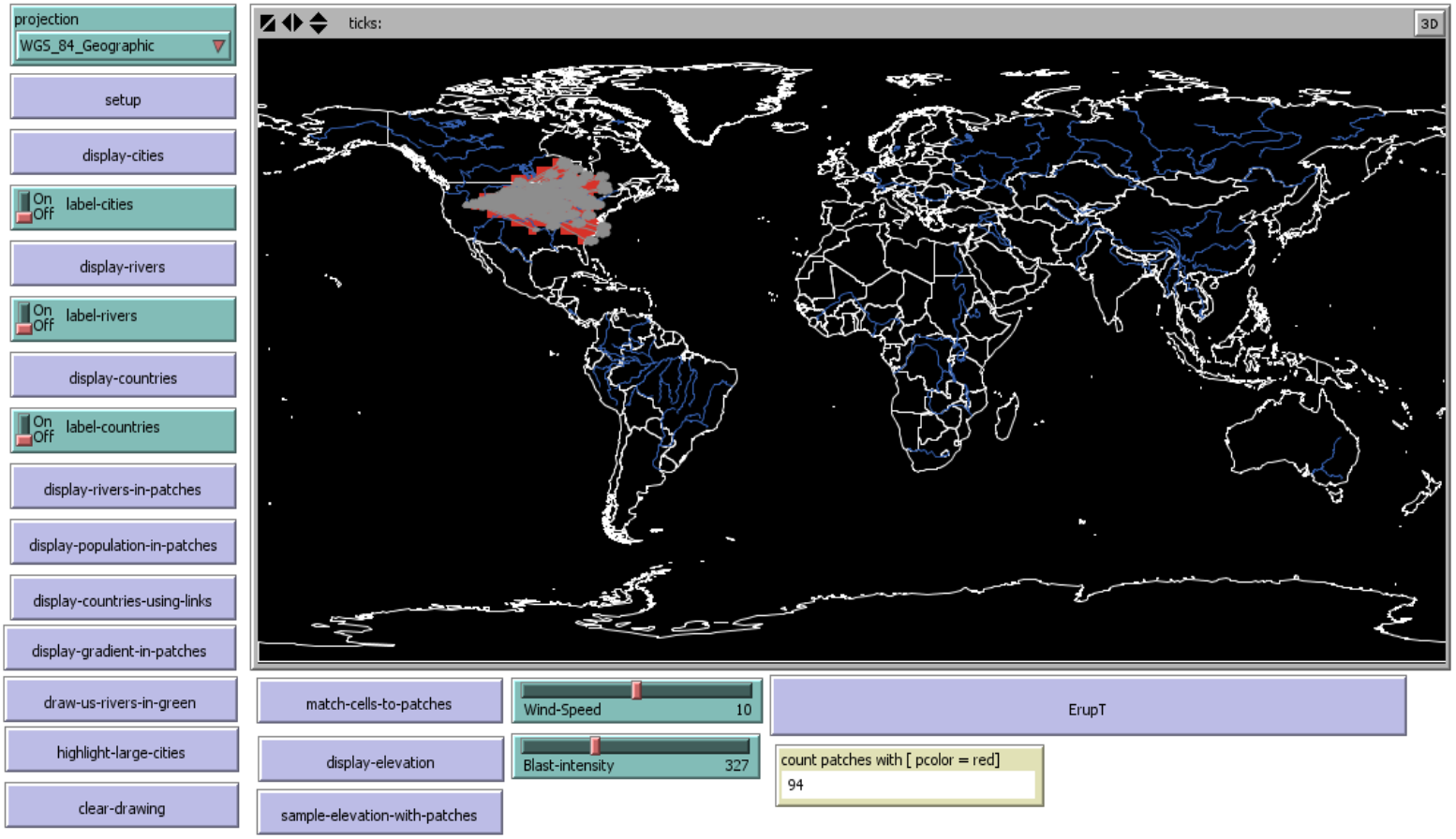
Our 1st Program



1st Program Code

```
patches-own [ elevation ]

to setup
  clear-all
  reset-ticks
  initialize-terrain
  create-turtles blastintensity
  [
    setxy 152.5 126.75
    set color gray
    set shape "cloud"
    set size 5
  ]
end


to go
  ask turtles
  [
    pen-down
    forward wind
    set color color + 1
    set heading winddirection
    right random 120
    left random 120
  if xcor > 305 [ die ]
  if ycor > 169 [ die ]
  if ycor < 0 [ die ]
  ]
end

to initialize-terrain
  ask patches
  [
  let elev1 35 - distancexy 152.5 126.75
  let elev2 30 - distancexy 176 60
  ifelse elev1 > elev2
  [set elevation elev1]
  [set elevation elev2]
  set pcolor scale-color green elevation 23 115
  ]
end
```

Final Program



Final Program Code

```
extensions [ gis ]
globals [ cities-dataset
          rivers-dataset
          countries-dataset
          elevation-dataset ]
breed [ city-labels city-label ]
breed [ country-labels country-label ]
breed [ country-vertices country-vertex ]
breed [ river-labels river-label ]
patches-own [ population country-name elevation ]

breed [ash];; make ask agents

to setup
  reset-ticks

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
  ca
  ; Note that setting the coordinate system here is optional, as
  ; long as all of your datasets use the same coordinate system.
  gis:load-coordinate-system (word "data/" projection ".prj")
  ; Load all of our datasets
  set cities-dataset gis:load-dataset "data/cities.shp"
  set rivers-dataset gis:load-dataset "data/rivers.shp"
  set countries-dataset gis:load-dataset "data/countries.shp"
  set elevation-dataset gis:load-dataset "data/world-elevation.asc"
  ; Set the world envelope to the union of all of our dataset's envelopes
  gis:set-world-envelope (gis:envelope-union-of (gis:envelope-of cities-dataset)
                                                (gis:envelope-of rivers-dataset)
                                                (gis:envelope-of countries-dataset)
                                                (gis:envelope-of elevation-dataset))
end


; Drawing point data from a shapefile, and optionally loading the
; data into turtles, if label-cities is true
to display-cities
  ask city-labels [ die ]
  foreach gis:feature-list-of cities-dataset
  [ gis:set-drawing-color scale-color red (gis:property-value ? "POPULATION") 5000000 1000
    gis:fill ? 2.0
    if label-cities
    [ ; a feature in a point dataset may have multiple points, so we
      ; have a list of lists of points, which is why we need to use
      ; first twice here
      let location gis:location-of (first (first (gis:vertex-lists-of ?)))
      ; location will be an empty list if the point lies outside the
      ; bounds of the current NetLogo world, as defined by our current
      ; coordinate transformation
      if not empty? location
      [ create-city-labels 1
        [ set xcor item 0 location
          set ycor item 1 location
          set size 0
          set label gis:property-value ? "NAME" ] ] ] ]
end
```

```
; Drawing polygon data from a shapefile, and optionally loading some
; of the data into turtles, if label-countries is true
to display-countries
  ask country-labels [ die ]
  gis:set-drawing-color white
  gis:draw countries-dataset 1
  if label-countries
  [ foreach gis:feature-list-of countries-dataset
    [ let centroid gis:location-of gis:centroid-of ?
      ; centroid will be an empty list if it lies outside the bounds
      ; of the current NetLogo world, as defined by our current GIS
      ; coordinate transformation
      if not empty? centroid
      [ create-country-labels 1
        [ set xcor item 0 centroid
          set ycor item 1 centroid
          set size 0
          set label gis:property-value ? "CNTRY_NAME" ] ] ] ]
end

; Loading polygon data into turtles connected by links
to display-countries-using-links
  ask country-vertices [ die ]
  foreach gis:feature-list-of countries-dataset
  [ foreach gis:vertex-lists-of ?
    [ let previous-turtle nobody
      let first-turtle nobody
      ; By convention, the first and last coordinates of polygons
      ; in a shapefile are the same, so we don't create a turtle
      ; on the last vertex of the polygon
      foreach but-last ?
      [ let location gis:location-of ?
        ; location will be an empty list if it lies outside the
        ; bounds of the current NetLogo world, as defined by our
        ; current GIS coordinate transformation
        if not empty? location
        [ create-country-vertices 1
          [ set xcor item 0 location
            set ycor item 1 location
            ifelse previous-turtle = nobody
            [ set first-turtle self ]
            [ create-link-with previous-turtle ]
            set hidden? true
            set previous-turtle self ] ] ]
      ; Link the first turtle to the last turtle to close the polygon
      if first-turtle != nobody and first-turtle != previous-turtle
      [ ask first-turtle
        [ create-link-with previous-turtle ] ] ] ]
end

; Using gis:intersecting to find the set of patches that intersects
; a given vector feature (in this case, a river).
to display-rivers-in-patches
  ask patches [ set pcolor black ]
  ask patches gis:intersecting rivers-dataset
  [ set pcolor cyan ]
end
```

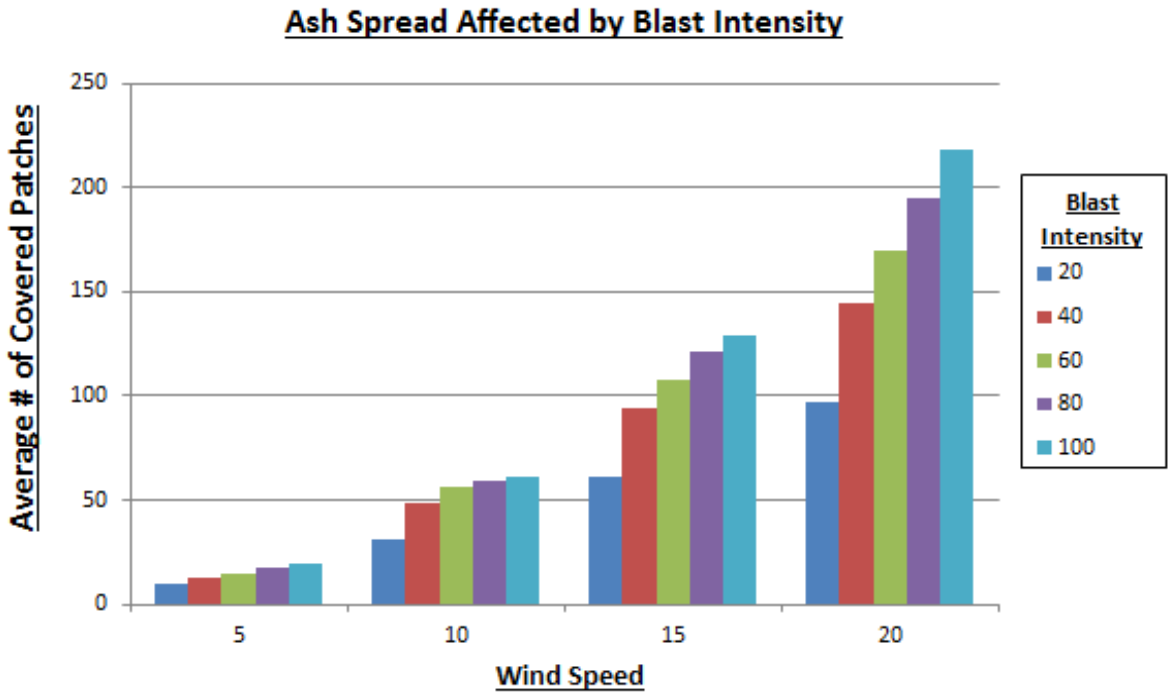
```
; Drawing polyline data from a shapefile, and optionally loading some
; of the data into turtles, if label-rivers is true
to display-rivers
  ask river-labels [ die ]
  gis:set-drawing-color blue
  gis:draw rivers-dataset 1
  if label-rivers
  [ foreach gis:feature-list-of rivers-dataset
    [ let centroid gis:location-of gis:centroid-of ?
      ; centroid will be an empty list if it lies outside the bounds
      ; of the current NetLogo world, as defined by our current GIS
      ; coordinate transformation
      if not empty? centroid
      [ create-river-labels 1
          [ set xcor item 0 centroid
            set ycor item 1 centroid
            set size 0
            set label gis:property-value ? "NAME" ] ] ] ]
end

; Drawing polygon data from a shapefile, and optionally loading some
; of the data into turtles, if label-countries is true
to display-countries
  ask country-labels [ die ]
  gis:set-drawing-color white
  gis:draw countries-dataset 1
  if label-countries
  [ foreach gis:feature-list-of countries-dataset
    [ let centroid gis:location-of gis:centroid-of ?
      ; centroid will be an empty list if it lies outside the bounds
      ; of the current NetLogo world, as defined by our current GIS
      ; coordinate transformation
      if not empty? centroid
      [ create-country-labels 1
        [ set xcor item 0 centroid
          set ycor item 1 centroid
          set size 0
          set label gis:property-value ? "CNTRY_NAME" ] ] ] ]
end

; Loading polygon data into turtles connected by links
to display-countries-using-links
  ask country-vertices [ die ]
  foreach gis:feature-list-of countries-dataset
  [ foreach gis:vertex-lists-of ?
    [ let previous-turtle nobody
      let first-turtle nobody
      ; By convention, the first and last coordinates of polygons
      ; in a shapefile are the same, so we don't create a turtle
      ; on the last vertex of the polygon
      foreach but-last ?
      [ let location gis:location-of ?
        ; location will be an empty list if it lies outside the
        ; bounds of the current NetLogo world, as defined by our
        ; current GIS coordinate transformation
        if not empty? location
```

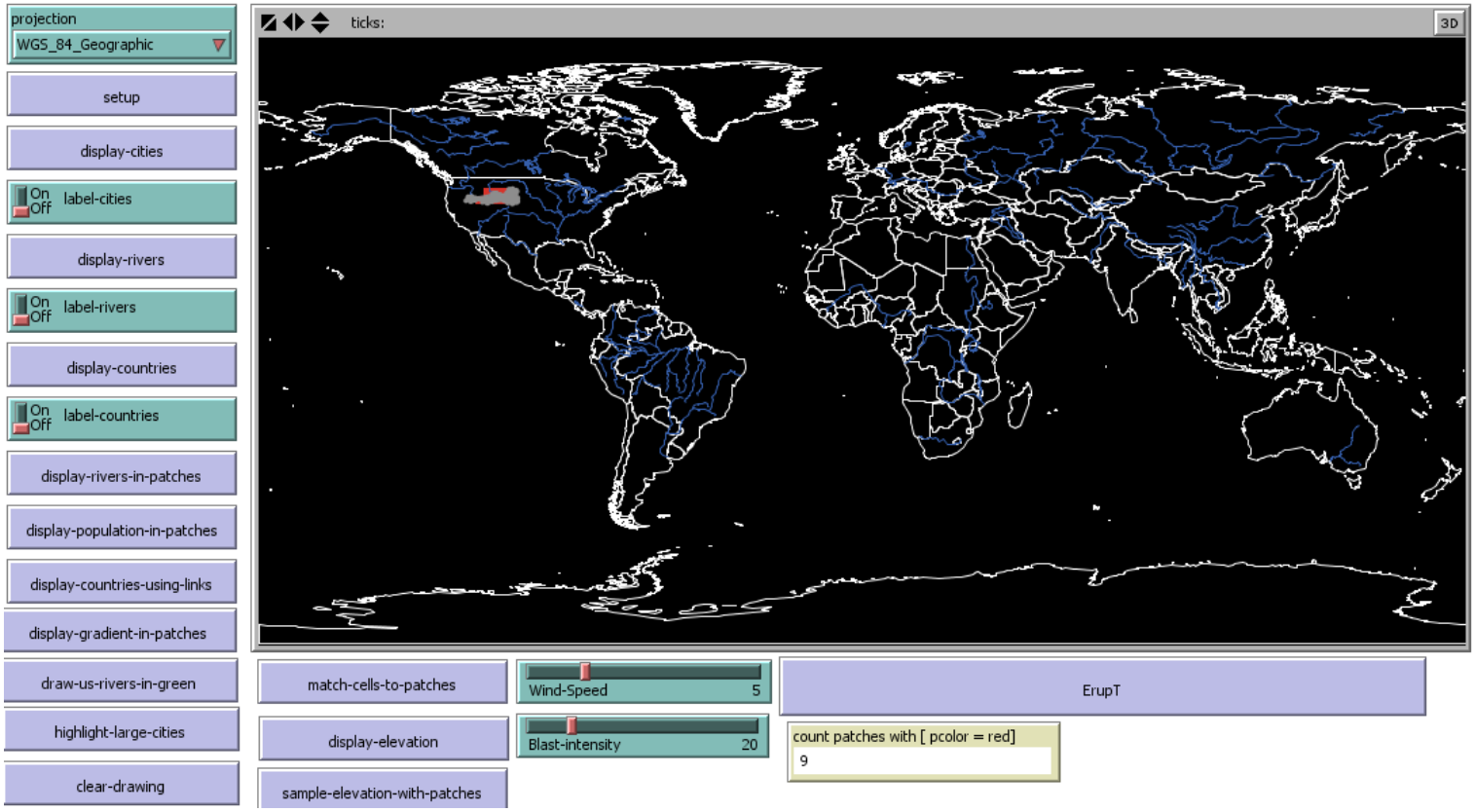
```
        [ create-country-vertices 1
          [ set xcor item 0 location
            set ycor item 1 location
            ifelse previous-turtle = nobody
            [ set first-turtle self ]
            [ create-link-with previous-turtle ]
            set hidden? true
            set previous-turtle self ] ] ]
      ; Link the first turtle to the last turtle to close the polygon
      if first-turtle != nobody and first-turtle != previous-turtle
      [ ask first-turtle
        [ create-link-with previous-turtle ] ] ] ]
end

; Using gis:intersecting to find the set of patches that intersects
; a given vector feature (in this case, a river).
to display-rivers-in-patches
  ask patches [ set pcolor black ]
  ask patches gis:intersecting rivers-dataset
  [ set pcolor cyan ]
end

; Using gis:apply-coverage to copy values from a polygon dataset
; to a patch variable
to display-population-in-patches
  gis:apply-coverage countries-dataset "POP_CNTRY" population
  ask patches
  [ ifelse (population > 0)
    [ set pcolor scale-color red population 500000000 100000 ]
    [ set pcolor blue ] ]
end

; Using find-one-of to find a particular VectorFeature, then using
; gis:intersects? to do something with all the features from another
; dataset that intersect that feature.
to draw-us-rivers-in-green
  let united-states gis:find-one-feature countries-dataset "CNTRY_NAME" "United States"
  gis:set-drawing-color green
  foreach gis:feature-list-of rivers-dataset
  [ if gis:intersects? ? united-states
    [ gis:draw ? 1 ] ]
end

; Using find-greater-than to find a list of VectorFeatures by value.
to highlight-large-cities
  let united-states gis:find-one-feature countries-dataset "CNTRY_NAME" "United States"
  gis:set-drawing-color yellow
  foreach gis:find-greater-than cities-dataset "POPULATION" 10000000
  [ gis:draw ? 3 ]
end

; Drawing a raster dataset to the NetLogo drawing layer, which sits
; on top of (and obscures) the patches.
to display-elevation
  gis:paint elevation-dataset 0
end
```

```
to display-elevation-in-patches
  ; This is the preferred way of copying values from a raster dataset
  ; into a patch variable: in one step, using gis:apply-raster.
  gis:apply-raster elevation-dataset elevation
  ; Now, just to make sure it worked, we'll color each patch by its
  ; elevation value.
  let min-elevation gis:minimum-of elevation-dataset
  let max-elevation gis:maximum-of elevation-dataset
  ask patches
  [ ; note the use of the "<= 0 or >= 0" technique to filter out
    ; "not a number" values, as discussed in the documentation.
    if (elevation <= 0) or (elevation >= 0)
    [ set pcolor scale-color black elevation min-elevation max-elevation ] ]
end

; This is a second way of copying values from a raster dataset into
; patches, by asking for a rectangular sample of the raster at each
; patch. This is somewhat slower, but it does produce smoother
; subsampling, which is desirable for some types of data.
to sample-elevation-with-patches
  let min-elevation gis:minimum-of elevation-dataset
  let max-elevation gis:maximum-of elevation-dataset
  ask patches
  [ set elevation gis:raster-sample elevation-dataset self
    if (elevation <= 0) or (elevation >= 0)
    [ set pcolor scale-color black elevation min-elevation max-elevation ] ]
end

; This is an example of how to select a subset of a raster dataset
; whose size and shape matches the dimensions of the NetLogo world.
; It doesn't actually draw anything; it just modifies the coordinate
; transformation to line up patch boundaries with raster cell
; boundaries. You need to call one of the other commands after calling
; this one to see its effect.
to match-cells-to-patches
  gis:set-world-envelope gis:raster-world-envelope elevation-dataset 0 0
  cd
  ct
end

; This command also demonstrates the technique of creating a new, empty
; raster dataset and filling it with values from a calculation.
;
; This command uses the gis:convolve primitive to compute the horizontal
; and vertical Sobel gradients of the elevation dataset, then combines
; them using the square root of the sum of their squares to compute an
; overall "image gradient". This is really more of an image-processing
; technique than a GIS technique, but I've included it here to show how
; it can be easily done using the GIS extension.
to display-gradient-in-patches
  let horizontal-gradient gis:convolve elevation-dataset 3 3 [ 1 0 -1 2 0 -2 1 0 -1 ] 1 1
  let vertical-gradient gis:convolve elevation-dataset 3 3 [ 1 2 1 0 0 0 -1 -2 -1 ] 1 1
  let gradient gis:create-raster gis:width-of elevation-dataset gis:height-of elevation-dataset gis:envelope-of elevation-dataset
  let x 0
  repeat (gis:width-of gradient)
  [ let y 0
    repeat (gis:height-of gradient)
```

```
    [ let gx gis:raster-value horizontal-gradient x y
      let gy gis:raster-value vertical-gradient x y
      if ((gx <= 0) or (gx >= 0)) and ((gy <= 0) or (gy >= 0))
      [ gis:set-raster-value gradient x y sqrt ((gx * gx) + (gy * gy)) ]
      set y y + 1 ]
    set x x + 1 ]
  let min-g gis:minimum-of gradient
  let max-g gis:maximum-of gradient
  gis:apply-raster gradient elevation
  ask patches
  [ if (elevation <= 0) or (elevation >= 0)
    [ set pcolor scale-color black elevation min-g max-g ] ]
end



to ErupT
  create-ash Blast-intensity ;; start eruption
  ask ash ;; ask the ash to...

  [
    setxy -47 17 ;; set spawning location to yellowstone on the map
    set shape "cloud" ;;; set the agent shap to cloud
    set color grey ;; set the agent color to grey
    set size random 3 ;; set the agents size to a random 3
   set heading 0
   pen-down ;; set the pen down


   if ycor < -7
  [
    set heading -90
  ]

  if ycor > 7
  [
    set heading 90 - random 10
  ]

  if size >= 2  ;; if the size of the ash is more than 2 go random 5 forward
     [
        repeat random wind-speed + random 3 [ left random 10 right random 10 forward 1 set pcolor red]
     ]


        if size <= 2  ;;;; if the size of the ash is less than 2 go random 10 forward
    [
    repeat random wind-speed - random 3 [ left random 10 right random 10 forward 1 set pcolor red]
    ]
  ]

end


; Public Domain:
; To the extent possible under law, Uri Wilensky has waived all
; copyright and related or neighboring rights to this model.
```

Results


Ash Spread Affected by Blast Intensity
Average # of Covered Patches
250
200
150
100
50
0
5
10
15
20
Wind Speed
Blast Intensity
20
40
60
80
100


projection
WGS_84_Geographic
setup
display-cities
On Off label-cities
display-rivers
On Off label-rivers
display-countries
On Off label-countries
display-rivers-in-patches
display-population-in-patches
display-countries-using-links
display-gradient-in-patches
draw-us-rivers-in-green
highlight-large-cities
clear-drawing
ticks:
3D
match-cells-to-patches
display-elevation
sample-elevation-with-patches
Wind-Speed 5
Blast-intensity 20
ErupT
count patches with [pcolor = red]
9

Conclusion

From the data that we got from our program we can see that the ash spreads more as the blast intensity goes up. We can also conclude that the average number of patches covered increases as the wind speed goes up. We also discovered that the blast intensity can be really high but it will not spread and cover as much patches without a powerful wind speed. We also concluded that the number of patches covered is higher if the wind speed is higher and. We also discovered that when the wind speed is high and the blast intensity is low the ash will spread almost the same distance and direction as when the blast intensity is high but will not cover as much land. The Program also tells us that the ash will not affect much of the west coast of the United States because of the wind currents we believe that the ash will spread to the east coast.

Future Work

Our team would like to keep working on the model and improve the wind currents on the map. In addition we would also like to be able to add more variables such as time of year and mountains.

Acknowledgments

We would like to thank our sponsor teacher Ms. Galligan who helped us with the programing. We would like to thank Mr. Gilroy for helping us with the scientific part of our project and also thank Sue Gibbs for looking at our program and giving us advice. The team would also like to give credit to Uri Wilensky for his GIS General Examples program which we used to help make our program.

Works Cited

"How High Can Explosive Eruptions Go and How Far Can the Debris and Ash Be Spread? | Volcano World." *How High Can Explosive Eruptions Go and How Far Can the Debris and Ash Be Spread?* | *Volcano World*. N.p., n.d. Web. 02 Apr. 2014.

"NASA Satellites Help Track Volcanic Ash Affecting Air Travel." *UPI*. N.p., n.d. Web. 02 Apr. 2014.

Than, Ker. "Huge Magma Pocket Lurks Beneath Yellowstone Supervolcano." *National Geographic*. National Geographic Society, 18 Dec. 2013. Web. 02 Apr. 2014.

"What Would An Eruption Of The Yellowstone Supervolcano Look Like?" *Infowars What Would An Eruption Of The Yellowstone Supervolcano Look Like Comments*. N.p., n.d. Web. 02 Apr. 2014.

"Yellowstone Geology." *Yellowstone Geology*. N.p., n.d. Web. 31 Mar. 2014.

"Yellowstone Volcano Is the World's Largest, but Is It Dying?" *Missoulian.com*. N.p., 24 Mar. 2014. Web. 02 Apr. 2014.

Uri,Wilensky. Gis General Examples,