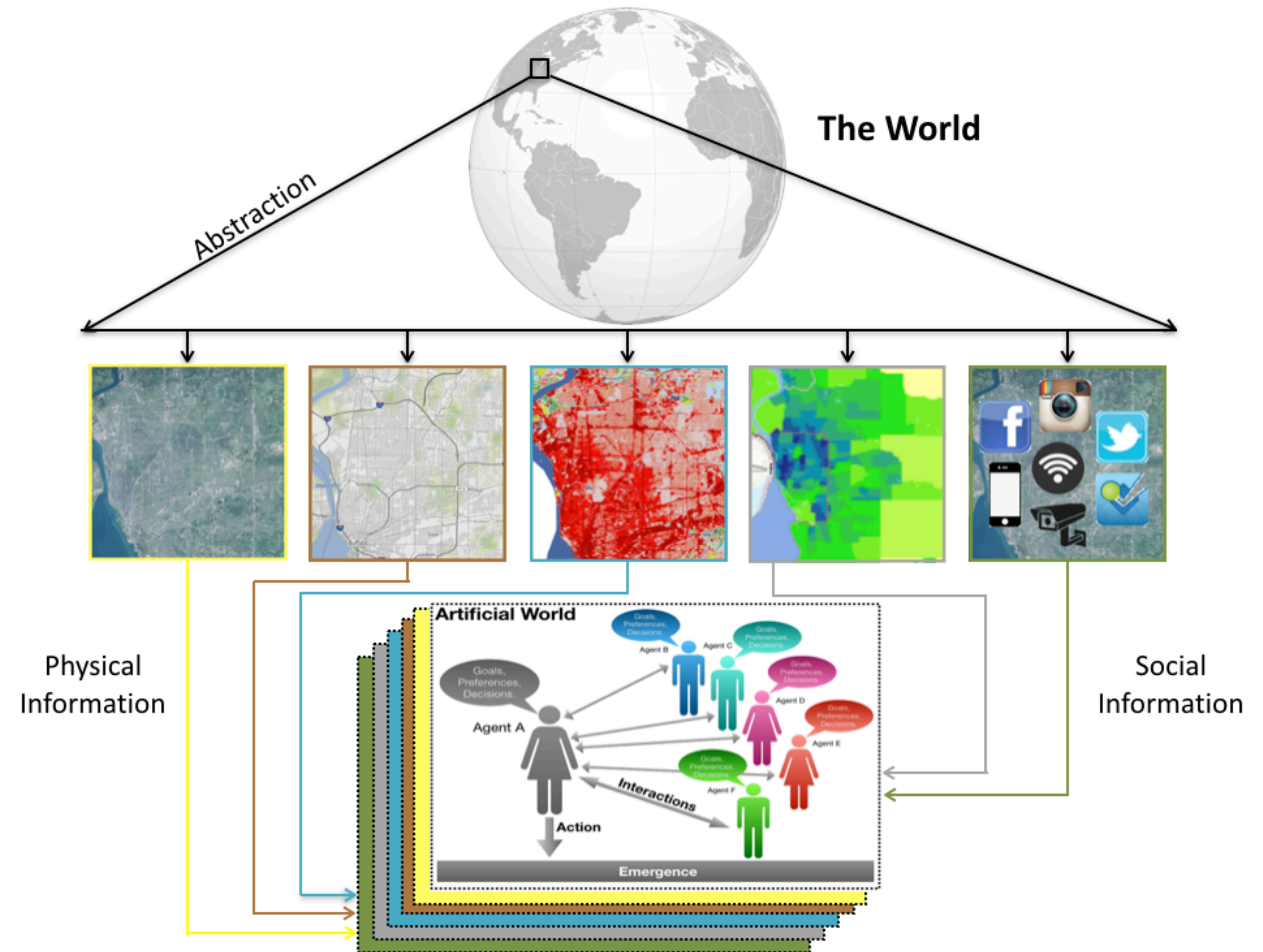


# **Agent Based Modeling and GIS**

**Schelling Segregation Polygon Model**

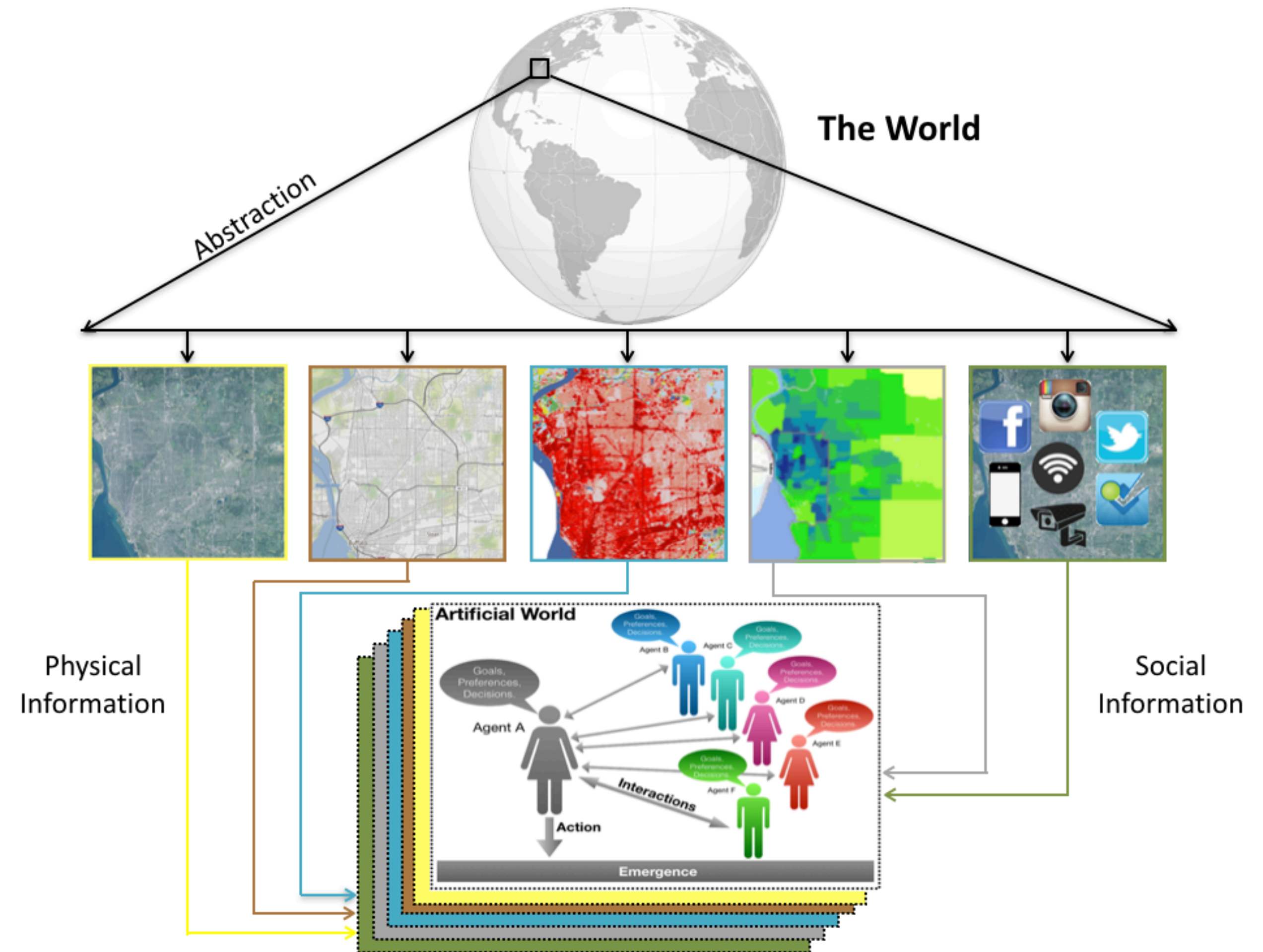
# Why is Space Important?

- **Space matters:** our perception of the World is naturally spatial.
- GIS allows us to link a place and often a time to agents behaviors within a model
- There are many examples of how agents and their aggregations interact and change in space and time



# GIS and Agent Based Modelling

- Linking models to GIS provide a more realistic representation of the world
- GISystems represent the world in a series of layers and objects of different types
- Layers and objects can be referenced and translated into an ABM

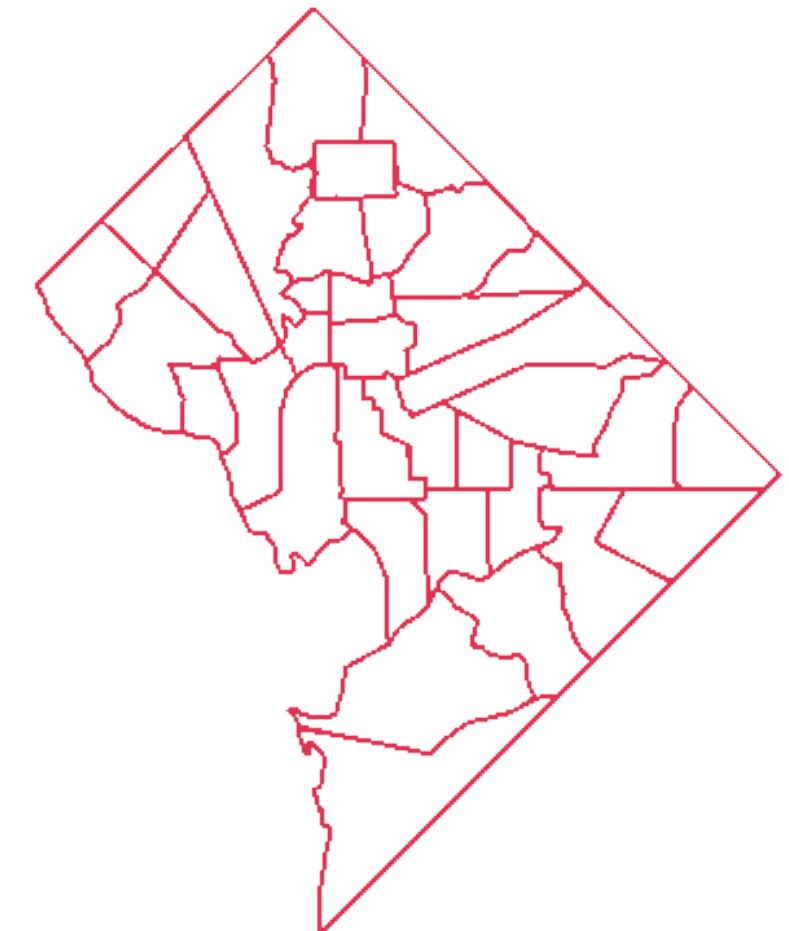
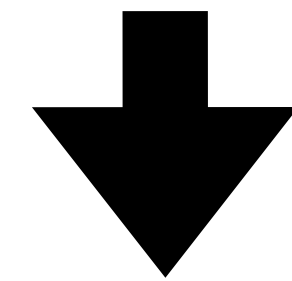
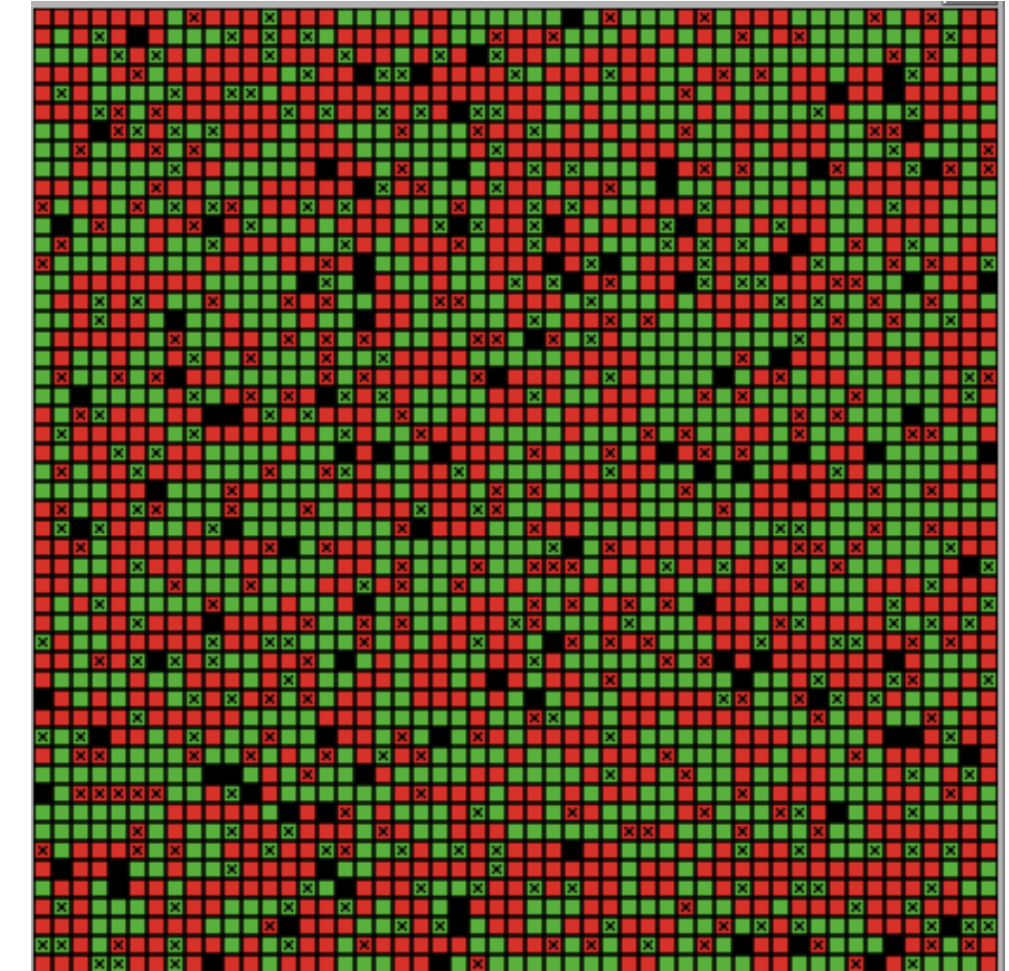




# A Simple GIS & Agent-based Modeling Example

## Schelling Segregation Model

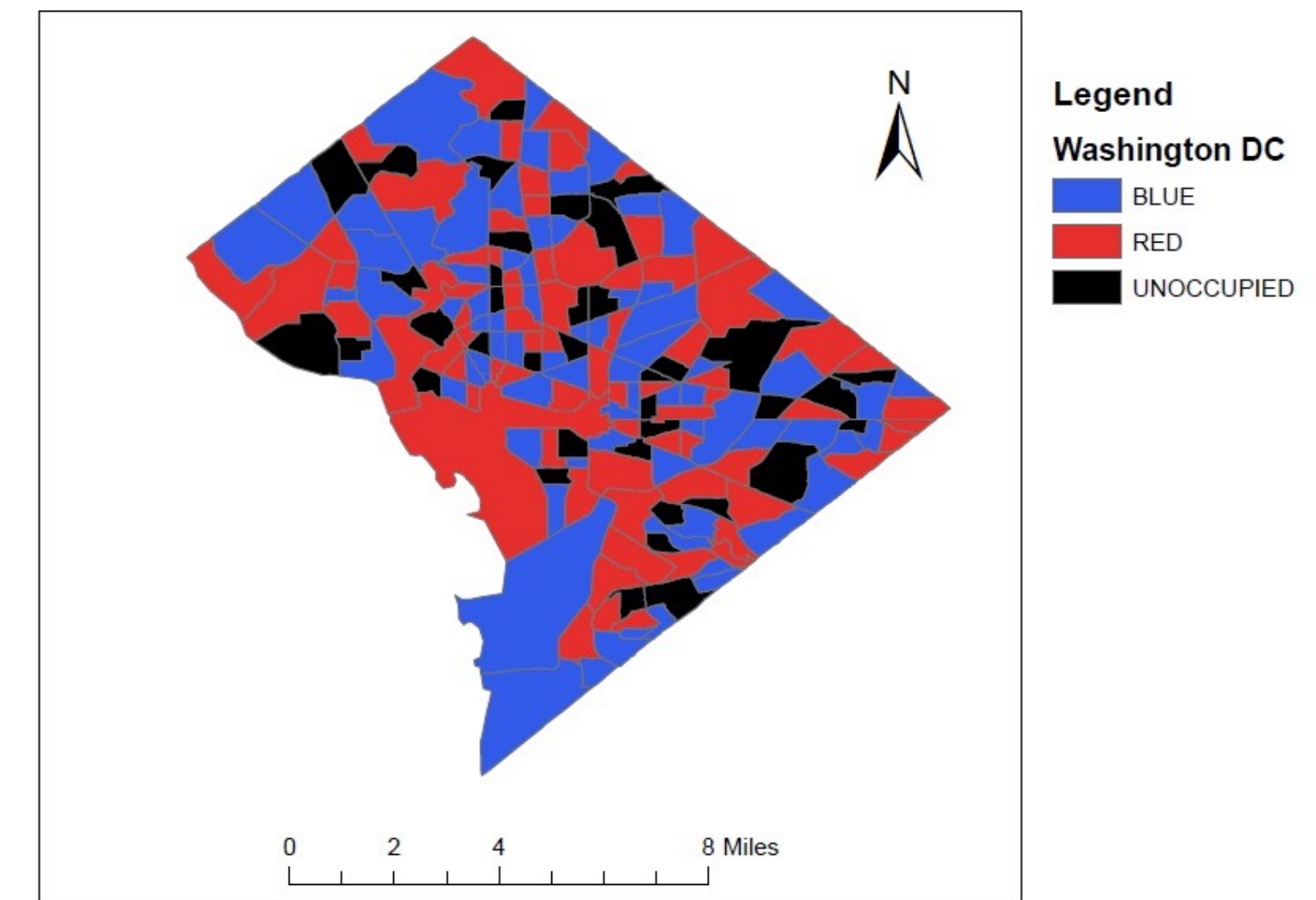
- Model provides an understanding the behavior of two types of agents in a neighborhood over time.
- In this example, a GIS layer allows a realistic representation of neighborhoods (i.e., surrounding census tracts)
- Stylized on Washington DC



# GIS and NetLogo

- NetLogo's GIS extension provides support for reading both vector data (in the form of ESRI shapefiles) and raster data (in the form of ESRI ASCII grid files).
- The [NetLogo documentation](#) states: *'This extension adds GIS (Geographic Information Systems) support to NetLogo. It provides the ability to load vector GIS data (points, lines, and polygons), and raster GIS data (grids) into your model.'*
- As with all NetLogo extensions, the model needs to *'activate'* it with the following line at the beginning of the model code:

extensions[[gis](#)]



# GIS in NetLogo

- Although the Shapefile is a proprietary format, it has become the de facto standard GIS data format for most GIS tools.
- A number of files are associated with the Shapefile format:
  1. the **Shapefile (.shp)**, which stores the geographical information needed to display the feature (x, y, z coordinates of vertexes and edges of the geometric shapes);
  2. the **database file (.dbf)**, which stores the data records for the feature; and
  3. the **index file (.shx)** which links the database file to the Shapefile (via the feature ID).

# Lets Build a Model

- To get the most out of this, refer to Chapters 3, 4 and 6 from
  - **Crooks, A.T., Malleson, N., Manley, E. and Heppenstall, A.J. (2019),**  
*Agent-based Modelling and Geographical Information Systems: A Practical Primer*, Sage, London, UK.

# **Lets First Create the Basic Scaffolding of a Model**



# Setting up the Globals, and Patches

```
extensions [gis] ;; informs NetLogo to call the GIS extension
```

```
globals [dc-dataset  
] ;;this will be the container for the data
```

```
patches-own [ID      ;;patch ID is identical with polygon ID_ID  
             popu    ;;population  
             mycolor  ;;its color  
             myneighbors ;;neighboring polygons' centroid patches  
]
```

# Setting up the Agents

```
turtles-own[tcolor ;;sets the color of the agent
percentage-same ;;percentage of same color turtles on neighbors
happy? ;;happy if neighboring same color agents >= 50%
tneighbors ;;an agentset of its neighbor turtles
rneighbors ;;number of red neighbors
bneighbors ;;number of blue neighbors
tneighborpolygons ;;neighboring polygons' centroid patches
]
```

# Importing the data and Creating Agents

to setup

ca

reset-ticks

set dc-dataset gis:load-dataset "data/DC.shp" ;;loading the vector data of DC

gis:set-world-envelope gis:envelope-of dc-dataset

;;drawing the map

foreach gis:feature-list-of dc-dataset

[ [?1] -> if gis:property-value ?1 "SOC" = "RED" [ gis:set-drawing-color red gis:fill ?1 2.0]

if gis:property-value ?1 "SOC" = "BLUE" [ gis:set-drawing-color blue gis:fill ?1 2.0]

if gis:property-value ?1 "SOC" = "UNOCCUPIED" [ gis:set-drawing-color grey gis:fill ?1 2.0]

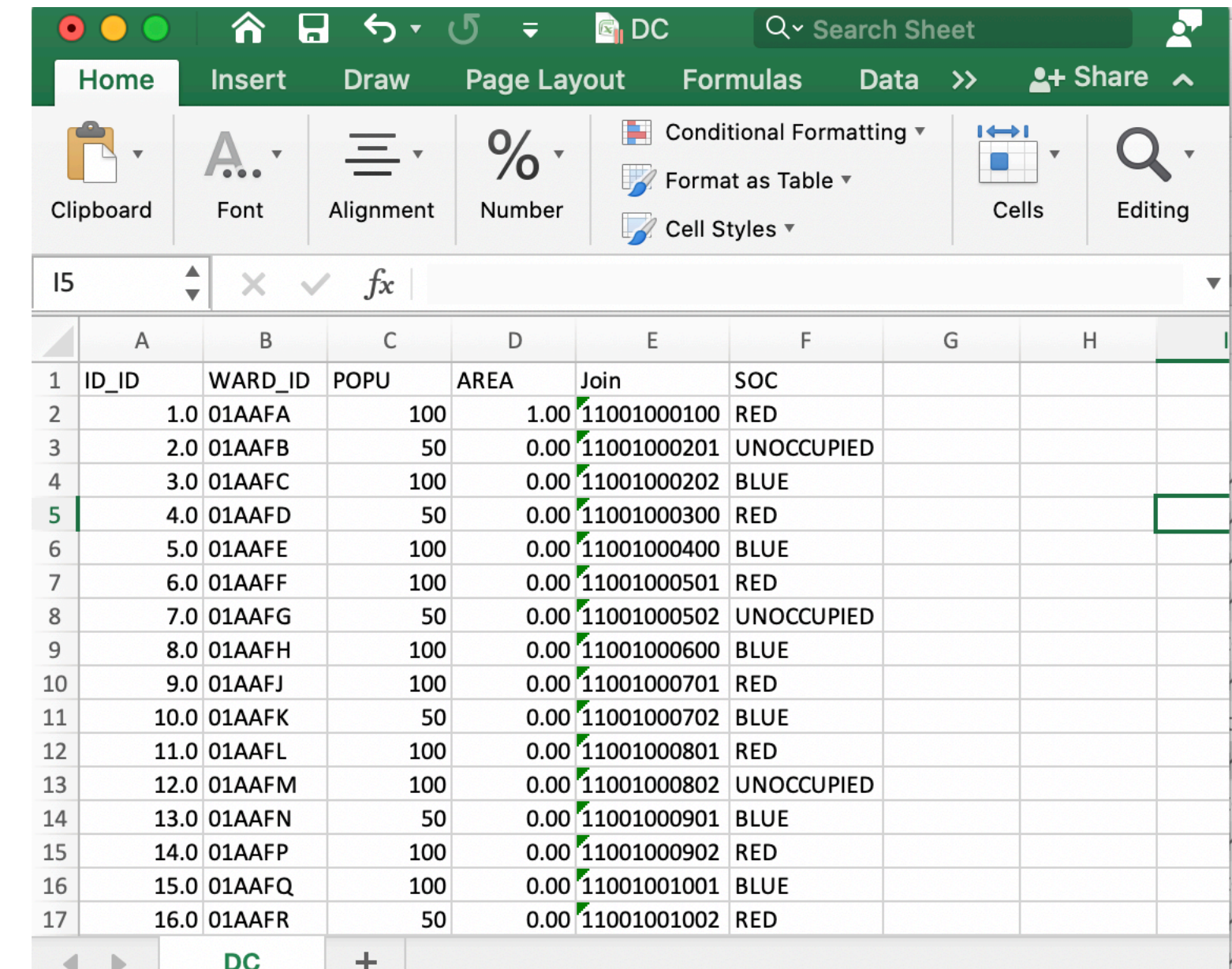
]

;;In the next two lines, we use gis:draw to draw the boundary of polygon data using white color

gis:set-drawing-color white

gis:draw dc-dataset 1

end



	A	B	C	D	E	F	G	H	I
1	ID_ID	WARD_ID	POPU	AREA	Join	SOC			
2	1.0	01AAFA	100	1.00	11001000100	RED			
3	2.0	01AAFB	50	0.00	11001000201	UNOCCUPIED			
4	3.0	01AAFC	100	0.00	11001000202	BLUE			
5	4.0	01AAFD	50	0.00	11001000300	RED			
6	5.0	01AAFE	100	0.00	11001000400	BLUE			
7	6.0	01AAFF	100	0.00	11001000501	RED			
8	7.0	01AAFG	50	0.00	11001000502	UNOCCUPIED			
9	8.0	01AAFH	100	0.00	11001000600	BLUE			
10	9.0	01AAFJ	100	0.00	11001000701	RED			
11	10.0	01AAFK	50	0.00	11001000702	BLUE			
12	11.0	01AAFL	100	0.00	11001000801	RED			
13	12.0	01AAFM	100	0.00	11001000802	UNOCCUPIED			
14	13.0	01AAFN	50	0.00	11001000901	BLUE			
15	14.0	01AAFP	100	0.00	11001000902	RED			
16	15.0	01AAFQ	100	0.00	11001001001	BLUE			
17	16.0	01AAFR	50	0.00	11001001002	RED			

All the agents Properties are held within the .DBF

# Setting the Extent of the World

- It is also necessary to set the extent of the world using the `tis:set-world-envelope` command, which maps the envelope of the NetLogo world to the GIS envelope, maintaining the same x and y scale
- See `gis:set-world-envelope` documentation for more information:
  - <https://ccl.northwestern.edu/netlogo/docs/gis.html#gis:set-world-envelope>



# Importing the data and Creating Agents

```
set dc-dataset gis:load-dataset "data/DC.shp"
```

- The line of code above loads the shapefile (dc.shp) along with the attribute data file (DC.dbf)

A	B	C	D	E	F	G
ID_ID	WARD_ID	POPU	AREA	Join	SOC	
1.0	01AAFA	100	1.00	11001000100	RED	
2.0	01AAFB	50	0.00	11001000201	UNOCCUPIED	
3.0	01AAFC	100	0.00	11001000202	BLUE	
4.0	01AAFD	50	0.00	11001000300	RED	
5.0	01AAFE	100	0.00	11001000400	BLUE	
6.0	01AAFF	100	0.00	11001000501	RED	
7.0	01AAFG	50	0.00	11001000502	UNOCCUPIED	
8.0	01AAFH	100	0.00	11001000600	BLUE	
9.0	01AAFI	100	0.00	11001000701	RED	

# Copying Attributes to Patches

- Make one patch represent one polygon (one color) by copy the color attributes to the patch centroid (add this to setup before the end)

```
let n 1
```

```
foreach gis:feature-list-of dc-dataset
```

```
[ [?1] -> let center-point gis:location-of gis:centroid-of ?1
```

```
ask patch item 0 center-point item 1 center-point [
```

```
set ID n
```

```
set mycolor gis:property-value ?1 "SOC"
```

```
if mycolor != "UNOCCUPIED" [sprout 1 [ ht set tcolor [mycolor] of myself setcolor]
```

```
]]
```

```
set n n + 1 ]
```

# Setup Color of Patches

```
to setcolor
```

```
  if tcolor = "RED" [set color red]
```

```
  if tcolor = "BLUE" [set color blue]
```

```
end
```

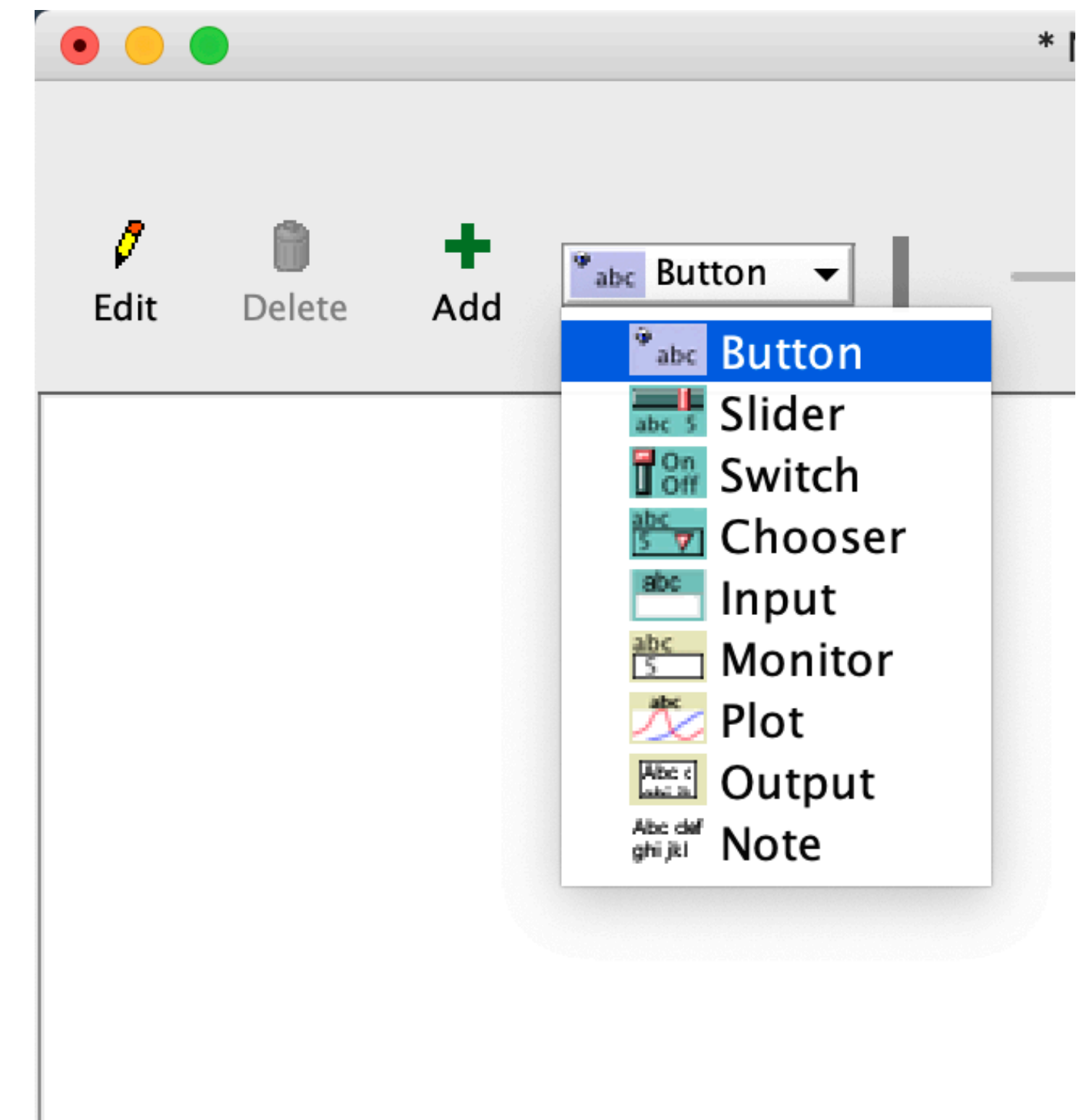
# So far we have:

- Set the bounding rectangle to shape envelope of the GIS space
- Color the polygons with either red or blue or grey and borderline of 2 points
- Go back to Interface and add a set up button



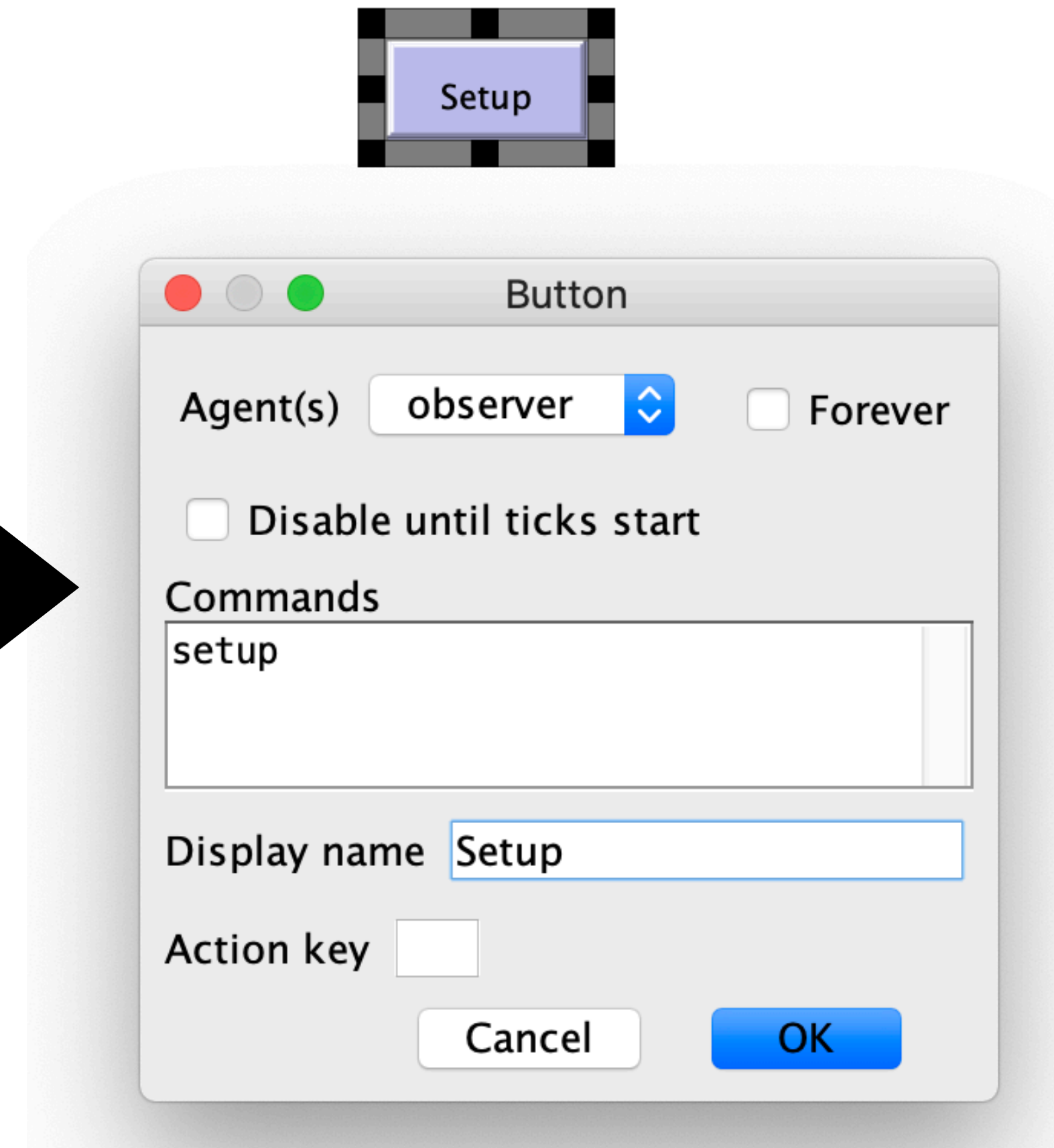
# Interface Objects

- Add some **user interface** control objects
  - To improve the usability, we'll add some interface control objects to the Interface tab.
  - Click on the Interface tab, and click on the pulldown selector labeled "**button**". Highlight the "**button**" type as shown to the right.



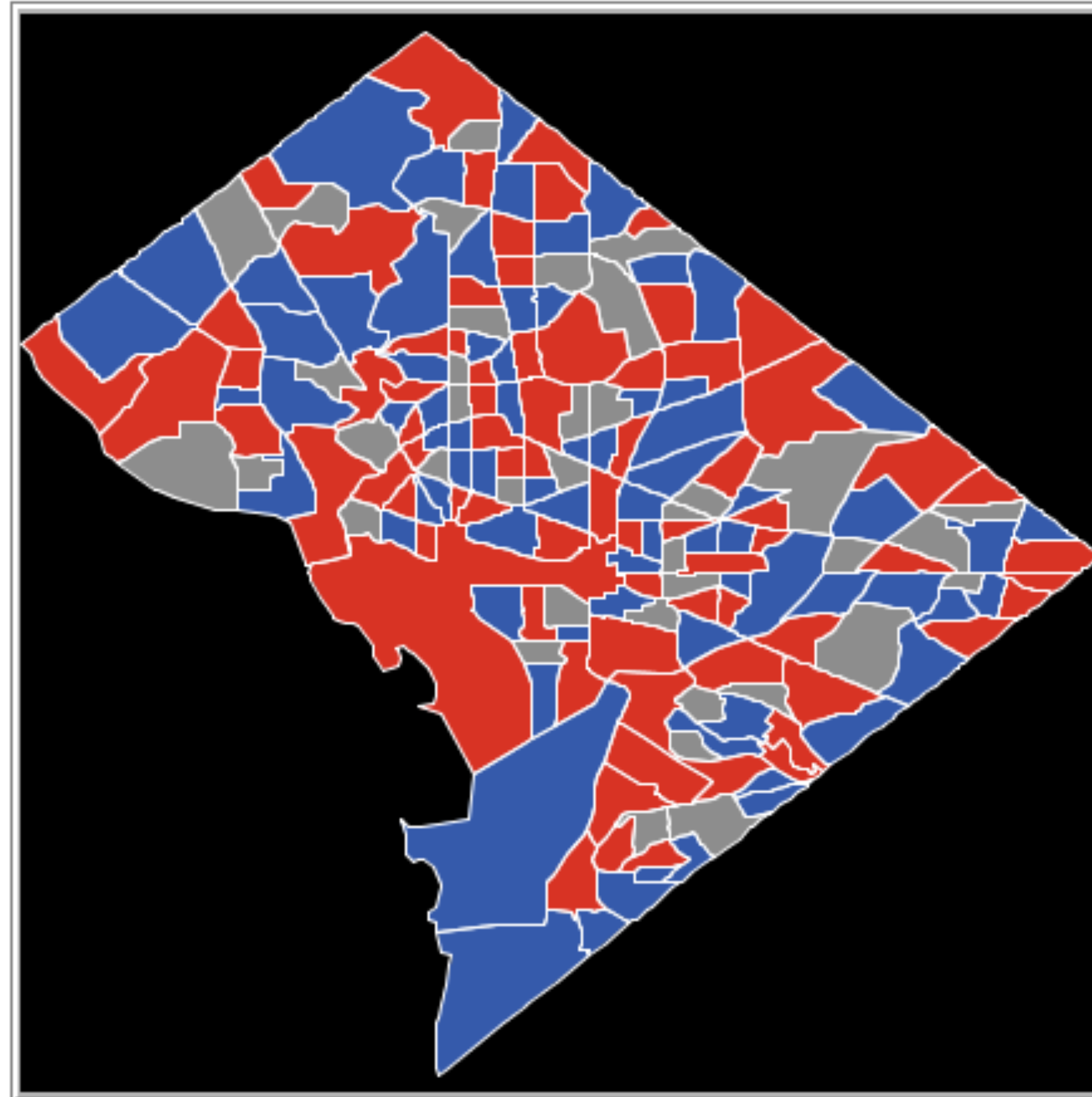
# Interface Objects

- In the "Commands" field of the dialog box, type "**setup**". In the "Display name" field, type "**Setup**". Your dialog box should now look like this.
- Press OK, and the dialog box will close. To reposition your button, place the cursor on the button, Right-click, and choose "select". You may now reposition the button on the screen.



# Draw Vector Data

setup



# Set the Neighbors for Each Patch (in setup)

- We have data about polygons with coinciding edges.
- Polygon 1 has 3, 5, 6, 62, 81, 82 and 91 as neighbor
- Load this from neighbors.txt

```
ask patches with [ID > 0] [ set myneighbors n-of 0 patches ;;empty agentset
]

;;find neighbors using a txt file produced by ArcGIS Polygon Neighbors. Please read NETLOGO FEATURES in info tab for more
information.
file-close
file-open "data/neighbors.txt"

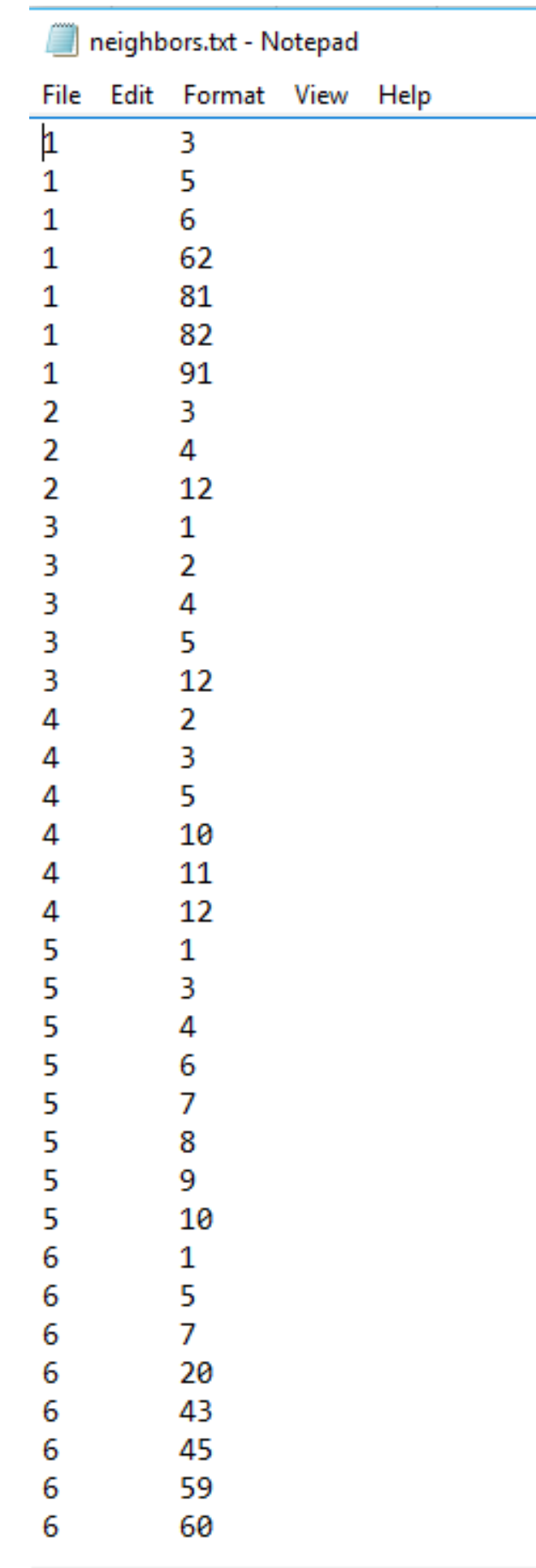
while [not file-at-end?] [
  let x file-read let y file-read
  ask patches with [ID = x] [set myneighbors (patch-set myneighbors patches with [ID = y]) ]
]
file-close
```

neighbors.txt - Notepad	
File	Edit Format View Help
1	3
1	5
1	6
1	62
1	81
1	82
1	91
2	3
2	4
2	12
3	1
3	2
3	4
3	5
3	12
4	2
4	3
4	5
4	10
4	11
4	12
5	1
5	3
5	4
5	6
5	7
5	8
5	9
5	10
6	1
6	5
6	7
6	20
6	43
6	45
6	59
6	60



# Calculating Polygon Neighbors

- While it is possible to calculating polygon neighbours via the model it is easier to do this outside of the model.
- In this example, the neighbours are calculated separately in a GIS (using the 'Polygon Neighbors' function in ArcGIS in this case) and stored in a plain text (.txt) file.



File	Edit	Format	View	Help
1		3		
1		5		
1		6		
1		62		
1		81		
1		82		
1		91		
2		3		
2		4		
2		12		
3		1		
3		2		
3		4		
3		5		
3		12		
4		2		
4		3		
4		5		
4		10		
4		11		
4		12		
5		1		
5		3		
5		4		
5		6		
5		7		
5		8		
5		9		
5		10		
6		1		
6		5		
6		7		
6		20		
6		43		
6		45		
6		59		
6		60		

# Now Lets make the model run....

to go

move

ask patches with [ID > 0] [if count turtles-here > 1 [print "ERROR"]] ;;verification

update-colors

tick

if count turtles with [happy? = true] = count turtles [stop];;stops the model when all agents are happy

end

# **Don't Forget to Add the “Go” Button**

# In “Go” we ask the agents to “Move”:

to move

;;count the number of red or blue turtles in neighboring polygons

ask turtles [set tneighbors turtles-on [myneighbors] of patch-here

set tneighborpolygons [myneighbors] of patch-here

set bneighbors count tneighbors with [tcolor = "BLUE"]

set rneighbors count tneighbors with [tcolor = "RED"]]

;;calculate the percentage of same color turtles

ask turtles [ ifelse rneighbors + bneighbors = 0 [set percentage-same 1][

if tcolor = "RED" [set percentage-same rneighbors / (rneighbors + bneighbors)]

if tcolor = "BLUE" [set percentage-same bneighbors / (rneighbors + bneighbors)]]

ifelse percentage-same < (Percentage-same-to-be-happy / 100) [set happy? false][set happy? true]]

;;move to an unoccupied polygon if not happy. change the color here.

ask turtles [if happy? = false and count tneighborpolygons with [mycolor = "UNOCCUPIED"] > 0

[ask patch-here [set mycolor "UNOCCUPIED"]

move-to one-of tneighborpolygons with [mycolor = "UNOCCUPIED"]

ask patch-here [set mycolor [tcolor] of myself]]

]

end



# In “Go” we also asked the agents to “update-colors”: Update Colors of the Patches

```
;if agents move patch colors need to be updated
```

```
to update-colors ;;update polygon colors
```

```
  ask patches with [ID > 0][
```

```
    if mycolor = "RED" [ gis:set-drawing-color red gis:fill item (ID - 1) gis:feature-list-of dc-dataset 2.0]
```

```
    if mycolor = "BLUE" [ gis:set-drawing-color blue gis:fill item (ID - 1) gis:feature-list-of dc-dataset 2.0]
```

```
    if mycolor = "UNOCCUPIED" [ gis:set-drawing-color grey gis:fill item (ID - 1) gis:feature-list-of dc-dataset 2.0]
```

```
  ]
```

```
  gis:set-drawing-color white
```

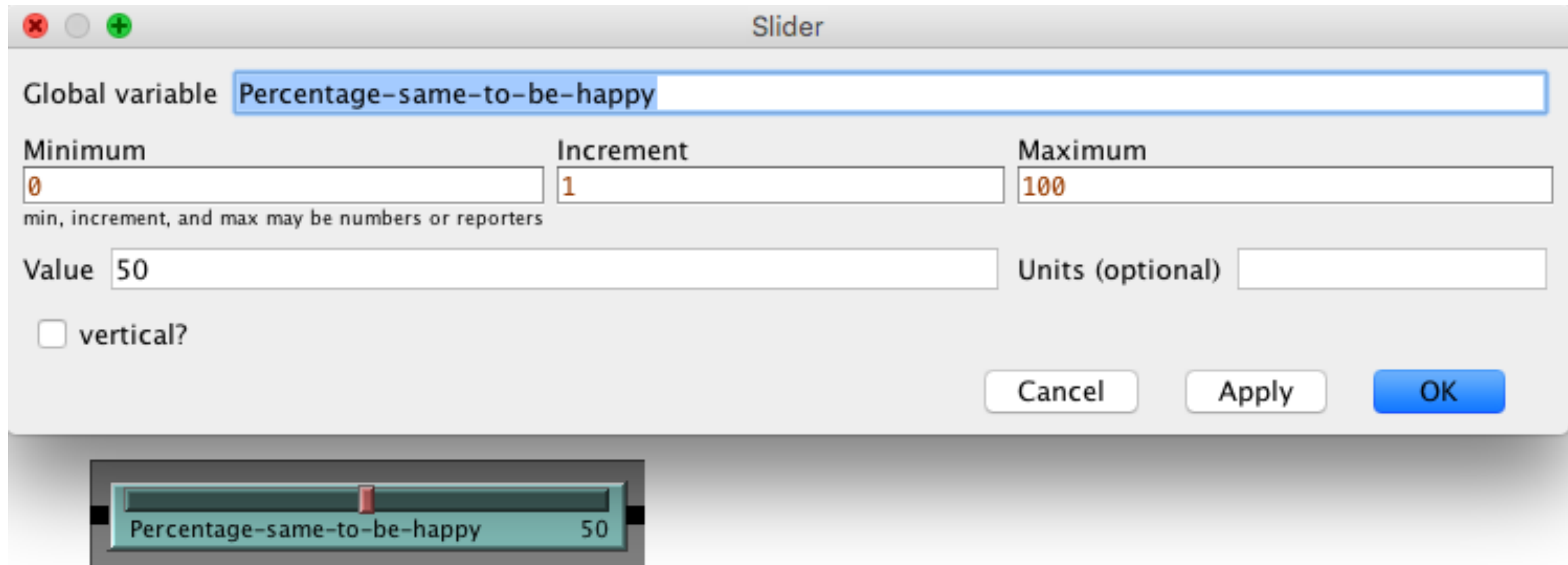
```
  gis:draw dc-dataset 1
```

```
end
```

# Error

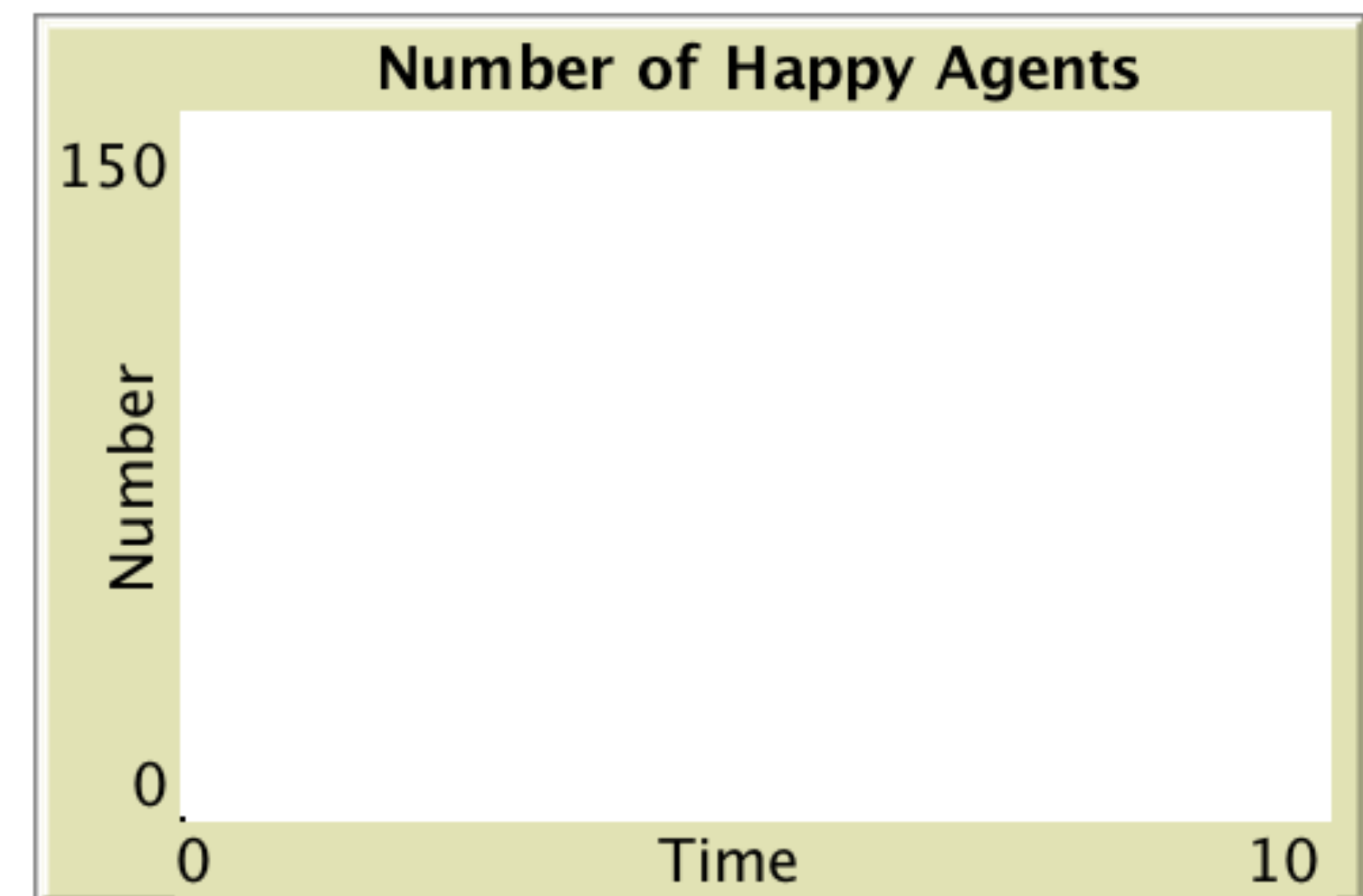
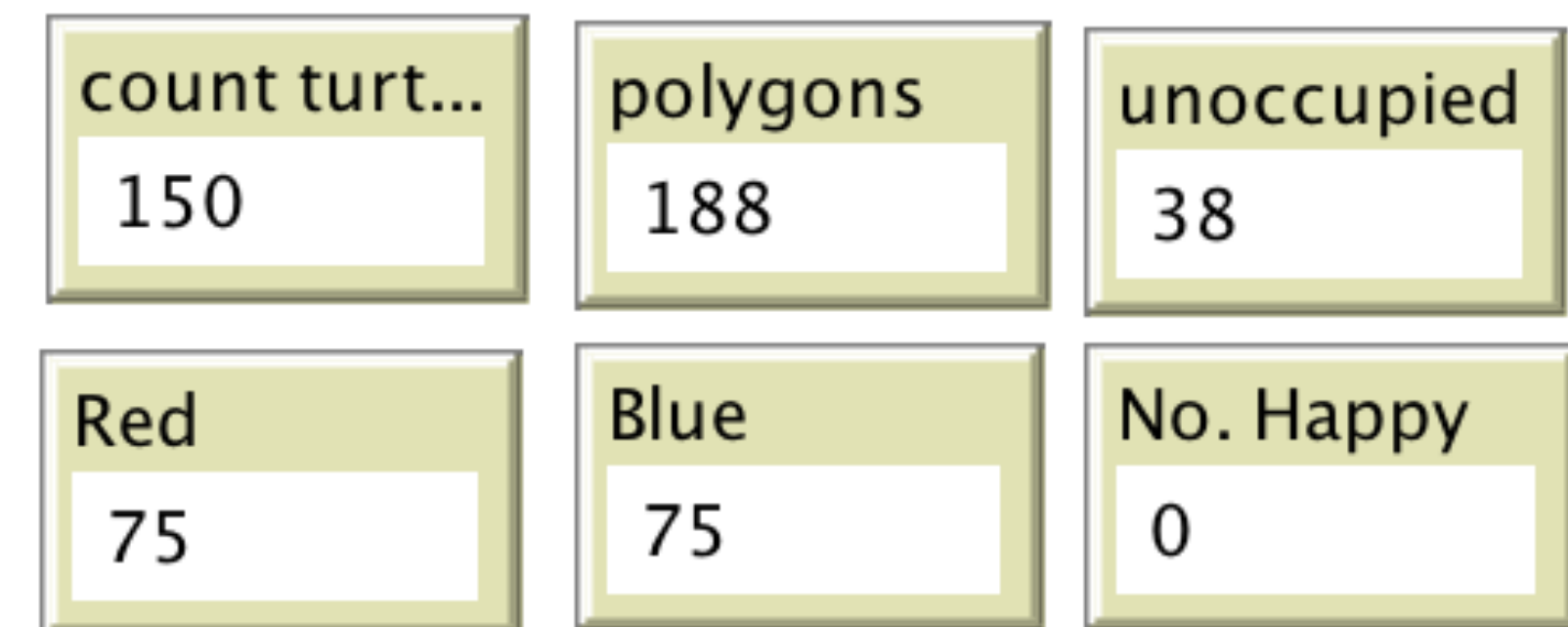
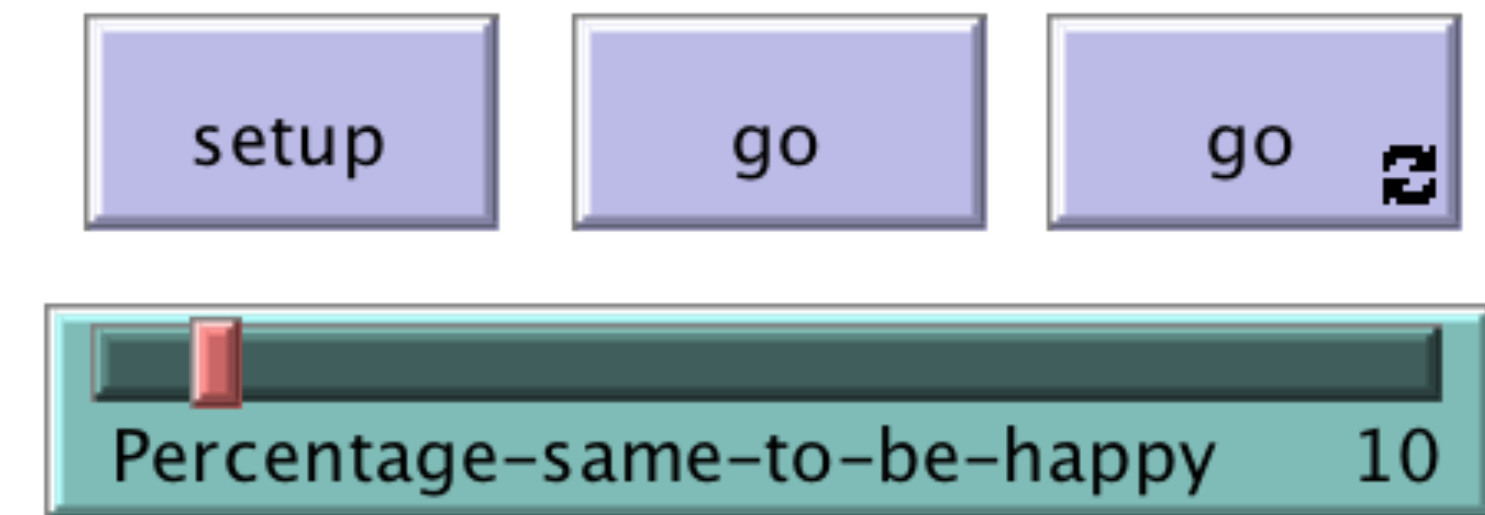
## You should of Noticed an Error

- Need to add a global parameter called “Percentage-same-to-be-happy”



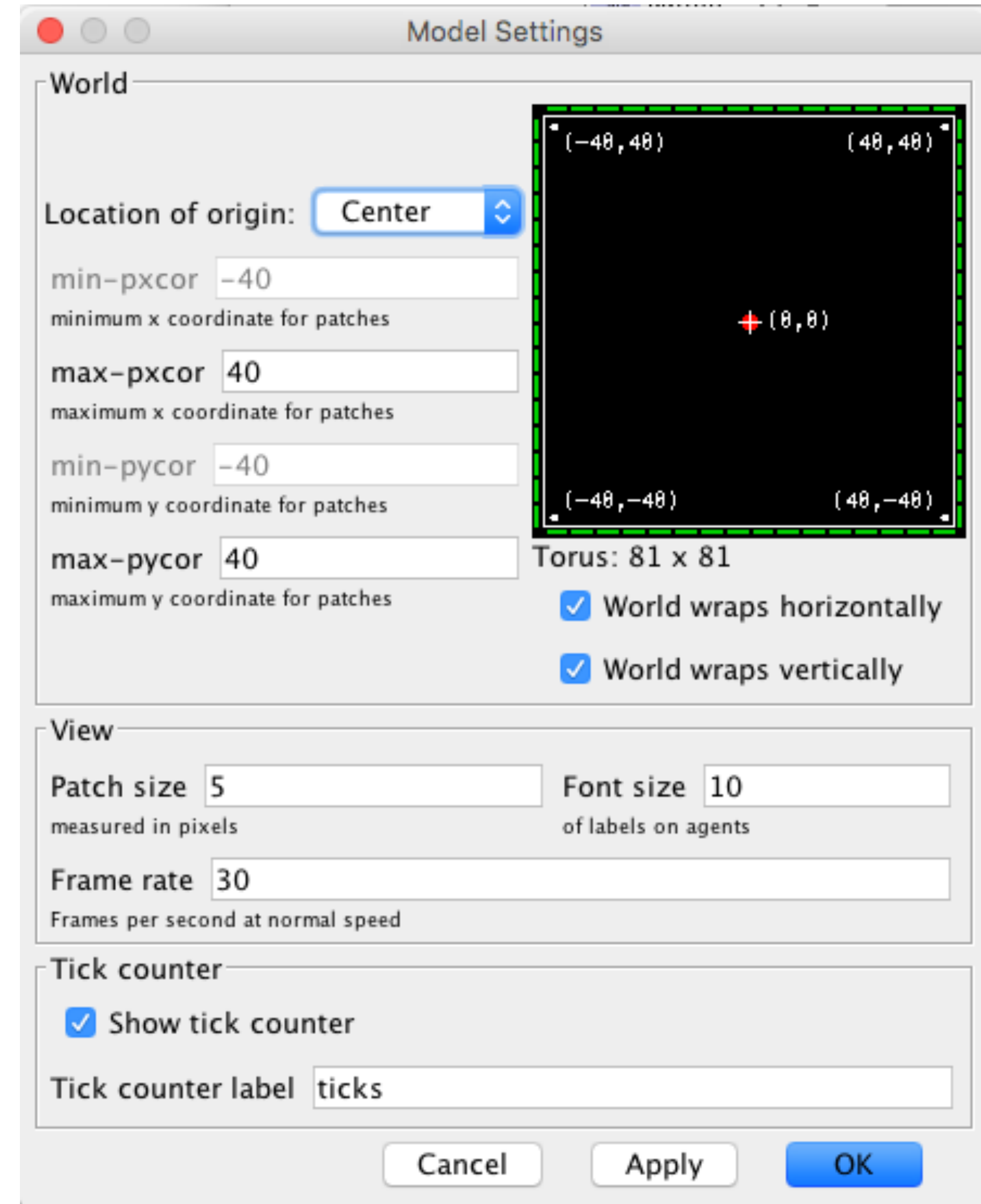
# Adding GUI Elements

- Add monitors so we can see what is going on:
- count turtles,
- Count polygons :
  - count patches with [ID > 0]
- UNOCCUPIED areas:
  - count patches with [mycolor = "UNOCCUPIED"]
- Count of Red Agents:
  - count turtles with [mycolor = "RED"]
- Count of Blue Agents:
  - count turtles with [mycolor = "BLUE"]
- Count Number of happy agents:
  - count turtles with [happy? = true]
- Add Plot of Happy Agents
  - plot count turtles with [happy? = true]



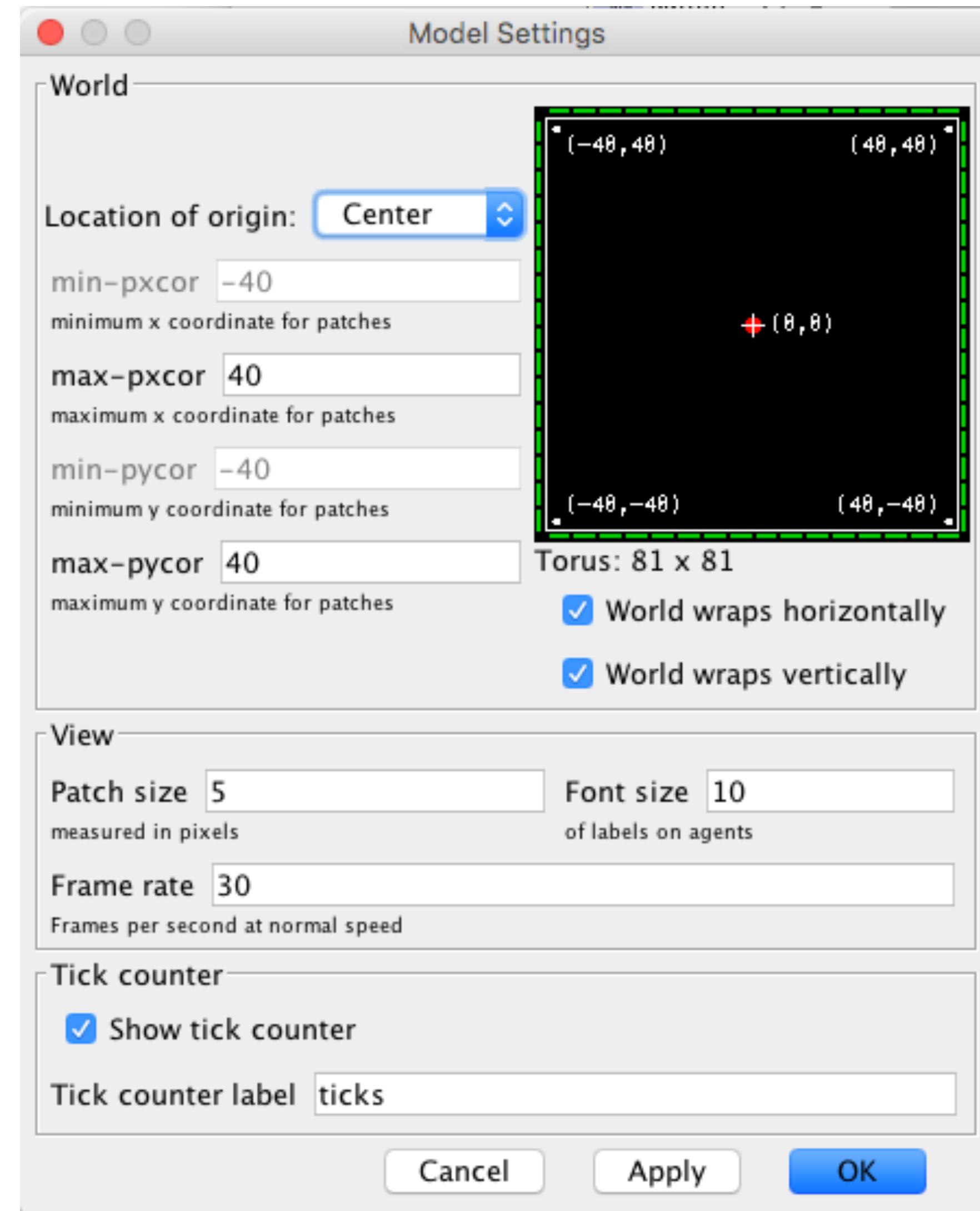
# Resize the world

- How many Polygons and Agents do you have?



# Resize the world

- How many Polygons and Agents do you have?
- But the DBF file has:
  - **188** wards, **75** are Red, **75** are Blue; and **38** are Unoccupied
- Why?
  - Some of the wards are smaller than the patches, therefore multiple centroids are place on top of each other.
- Therefore we need to resize the world.
- Now check the monitors





# Running the Model

- Verify the code for errors
- Go back to Interface tab
  - Click on Setup
    - Run the model with 25% Percentage-same to be happy
    - Reset and run the model with 50% Percentage-same to be happy
    - Reset and run the model with 75% Percentage-same to be happy

# Recap

- Why is GIS is important for ABM ?
- How to load GIS vector data into NetLogo
- How to integrate GIS attributes to NetLogo Elements

# Find out More

## NetLogo Gis Extension

### Using

This extension adds GIS (Geographic Information Systems) support to NetLogo. It provides the ability to load vector GIS data (points, lines, and polygons), and raster GIS data (grids) into your model.

The extension supports vector data in the form of ESRI shapefiles and GeoJSON files. The shapefile (.shp) and GeoJSON (.geojson) formats are the most common format for storing and exchanging vector GIS data. The extension supports raster data in the form of ESRI ASCII Grid files. The ASCII grid file (.asc or .grd) is not as common as the shapefile, but is supported as an interchange format by most GIS platforms.

### How to use

In general, you first define a transformation between GIS data space and NetLogo space, then load datasets and perform various operations on them. The easiest way to define a transformation between GIS space and NetLogo space is to take the union of the “envelopes” or bounding rectangles of all of your datasets in GIS space and map that directly to the bounds of the NetLogo world. See GIS General Examples for an example of this technique.

You may also optionally define a projection for the GIS space, in which case datasets will be re-projected to match that projection as they are loaded, as long as each of your data files has an associated .prj file that describes the projection or geographic coordinate system of the data. If no associated .prj file is found, the extension will assume that the dataset already uses the current projection, regardless of what that projection is.

Once the coordinate system is defined, you can load datasets using [gis:load-dataset](#). This primitive reports either a VectorDataset or a RasterDataset, depending on what type of file you pass it.

A VectorDataset consists of a collection of VectorFeatures, each one of which is a point, line, or polygon, along with a set of property values. A single VectorDataset may contain only one of the three possible types of features.

There are several things you can do with a VectorDataset: ask it for the names of the properties of its features, ask it for its “envelope” (bounding rectangle), ask for a list of all VectorFeatures in the dataset, search for a single VectorFeature or list of VectorFeatures whose value for a particular property is less than or greater than a particular value, or lies within a given range, or matches a given string using wildcard matching (“\*”, which matches any number of occurrences of any characters). If the VectorFeatures are polygons, you can also apply the values of a particular property of the dataset’s features to a given patch variable.

There are also several things you can do with a VectorFeature from a VectorDataset: ask it for a list of vertex lists, ask it for a property value by name, ask it for its centroid (center of gravity), and ask for a subset of a given agentset whose agents intersect the given VectorFeature. For point data, each vertex list will be a one-element list. For line data, each vertex list will represent the vertices of a line that makes up that feature. For polygon data, each vertex list will represent one “ring” of the polygon, and the first and last vertex of the list will be the same. The vertex lists are made up of values of type Vertex, and the centroid will be a value of type Vertex as well.

There are a number of operations defined for RasterDatasets as well. Mostly these involve sampling the values in the dataset, or re-sampling a raster to a different resolution. You can also apply a raster to a given patch variable, and convolve a raster using an arbitrary convolution matrix.

**Code Example:** GIS General Examples has general examples of how to use the extension

**Code Example:** GIS Gradient Example is a more advanced example of raster dataset analysis.

<https://ccl.northwestern.edu/netlogo/docs/gis.html>