

## **Repast Vector GIS integration**

Robert Najlis  
Argonne National Laboratory, Argonne, IL  
rnajlis@anl.gov

Michael J. North  
Argonne National Laboratory, Argonne, IL  
north@anl.gov

Repast now supports the reading, writing, and display of Shapefile data. The current implementation of GIS in Repast is focused on two systems: ESRI ArcMap and OpenMap. This paper will elucidate the creation of agents based on GIS data, as well as how to work with and display those agents. Specific attention is given to the usage of both ArcMap and OpenMap.

### **Contact:**

Robert Najlis  
9700 S. Cass Avenue,  
Argonne, IL 60439  
Tel: 1-917-703-4633  
Email: rnajlis@anl.gov

**Key Words:** Agent-Based Modeling and Simulation, Repast, GIS, Geographic Information Systems, Repast, ArcMap, OpenMap

# Repast Vector GIS integration

Robert Najlis, Michael J. North

Repast now supports the reading, writing, and display of shapefile data. In the Repast-GIS integration, these tasks are generally broken up into two different classes, a data class, and a display class. The data class allows data to be read into Repast from the GIS, and written out from Repast into a GIS format. The current implementation of GIS in Repast is focused on two systems: ESRI ArcMap and OpenMap. This paper will elucidate how to use each of these systems with Repast.

## Getting started: Choosing your data and display classes

Each GIS system used with Repast has its own classes for data operations (reading and writing of data) and display. As can be seen in Table 1, ArcMap uses the GeotoolsData class for data operations, and the EsriDisplay class for display processes. Correspondingly, OpenMap uses the OpenMapData and OpenMapDisplay classes.

Data Class	Display Class	GIS System
GeotoolsData	EsriDisplay	ArcMap
OpenMapData	OpenMapDisplay	OpenMap

Table 1. Data and Display classes for each GIS system

## Creating the Agents

In order to create agents from a shapefile, one first has to define an appropriate agent class. For ArcMap, this can be done by subclassing DefaultGeotoolsAgent, or by creating a class that implements the GeoToolsAgent interface. For OpenMap, the DefaultOpenMapAgent class can be subclassed, or the OpenMapAgent implemented. Furthermore, with OpenMap the MultiLayerOpenMapAgent interface can be implemented to specify agents that can be displayed on more than one layer (displaying agents on more than one layer can also be done using ArcMap, but would be handled through the ArcMap interface, rather than the data class).

While implementing the appropriate interface will result in agents that can be displayed on a map, to make GIS agents truly useful, one would want to assign data to the agent from the relevant data file. To do this, one has to specify functions in the agent class which correspond to the fields in the Shapefile data. For example, if there is a field in the Shapefile data called Landuse, the agent would need functions called setLanduse() and getLanduse() in order to read and write this data field. The setLanduse() function allows the data from the Shapefile to be set in the agent, and the getLanduse() function allows the data from the agent to be used in updating the Shapefile data file.

## Using the Data Classes

After the agent class has been created, you need to get a list of them (one agent for each feature in the Shapefile). This can be done with the createAgents() function

First, get the Data class for the GIS system you are using, either:

```
GeotoolsData gisData = GeotoolsData.getInstance();
```

Or

```
OpenMapData gisData = OpenMapData.getInstance();
```

Then create an ArrayList, and add the Agents to it:

```
ArrayList agentList = new ArrayList();
agentList.addAll(gisData.createAgents(Agent.class, datasource));
//here Agent.class refers to the GisAgent class specified (here in the Agent.java file)
```

Now you have a list of agents. If you specified the get and set Landuse functions, each agent in the list would have had the Landuse value set by the Landuse field in the dbf associated with the Shapefile.

To write out the agents to a data source, you can use the write Agents command, though first, you might want to sort your list of agents (if you have changed the order):

```
gisData.sortGisAgentsByIndex(agentList);
```

Then write the data:

```
gisData.writeAgents(agentList, datasource);
```

To write the data to a new file:

```
gisData.writeAgents(agentList, oldDatasource, newDatasource);
```

Note, that when using ArcMap, it is important that the original Shapefile be overwritten in order for the display to be correctly synchronized with the data. This will be explained more fully in the section on Using the Display Classes.

It is also possible to interrogate data, which means to look into a shapefile and see what fields it contains. The interrogate function returns an array of type `FieldNameAndType`

```
FieldNameAndType[] nameTypes = gisData.interrogate( SluGISOM.datasource);
for (int i=0; i<nameTypes.length; i++) {
    System.out.println("interrogate " + i + " field name: " + nameTypes[i].getFieldName() + "
    type : " + nameTypes[i].getFieldType());
}
```

Neighborhood information can also be set for each agent. In order to do this, a neighborhood data file must first be prepared. The neighborhood file can be computed using Geoda. Once the file is calculated, you can use the `readNeighborhoodInfo` function to set the neighborhood information for each agent:

```
gisData.readNeighborhoodInfo(neighborhoodFile.GAL, agentList);
```

### **Creating Shapefile data from Agents**

There are a few things that can be done from within the `OpenMapData` class that cannot be done with the `GeotoolsData` class. One of the most useful may be the ability to create Shapefile data from GIS Agents that were created from within Repast rather than from a Shapefile. This can be done using the `OpenMapData#writeAgentsNoShp` function

### **Using the Display Classes**

The ArcMap and OpenMap display classes work somewhat differently. This is due in part to the different type of integration they have with Repast. Nonetheless, they both allow Repast to update the GIS display so that the display of the GIS can correspond to the data of the ABMS.

Repast has Shapefile integration with ArcMap. Shapefile integration is similar a looser coupling based on sharing of files, with Repast having the ability to tell the GIS to update based on the files that Repast has written out. This means that while Repast and the GIS use the same Shapefile, they have very

limited interaction with each other. The Shapefile is loaded into the GIS, then Repast is used to read in the Shapefile data. Agents are created using this data, and the data is updated from the agents, as described in the previous section. In order to update the GIS display, the data must first be written out to file, and then the GIS can be notified to update its display by reading the newly written (updated) Shapefile data. With Shapefile integration, the display class only allows for the display to be updated. It does not provide a means for Repast to interact with the API of the GIS being used for display. However, since GeoTools is used from within Repast to represent the GIS data of the agents, the GeoTools API is available for use with Repast-ArcMap models.

With Openmap, Repast has Java integration. Native Java integration is a tighter coupling. Both systems are written in Java, and the source code has been integrated, so there is full access to the GIS code from within Repast. This means that a Repast program can have full interaction with the GIS. The Shapefile is loaded into Repast. The GIS is launched from within Repast. Shapefiles are added to the GIS from Repast. Layers are added to the GIS based on agent definitions. Updating of layers is also based on agent definitions. Agents are created from and written to Shapefiles in the same manner as with Shapefile integration. Of course layers can also be added and based on the Shapefile as well, and in this way be used in the same manner as Shapefile integration. Thus, while the data integration remains the same, there is a much tighter integration on the display end. Repast users can have full access to the GIS being used for display, thus allowing them to get information about agents on the map, such as location, distance from other geographic objects (including geographically represented agents), and more.

### **Displaying Agents with ArcMap**

Displaying GIS agents in ArcMap requires that the agent classes implement the GeotoolsAgent interface. In order to display the agents within ESRI ArcMap:

1. Load the data into ArcMap, adding the agents to the appropriate layers, and setting the symbology for those layers from within ArcMap
2. Run your model, either through AgentAnalyst, or Repast For Java.
3. Write the data. Make sure you rewrite the file to the same location, as ArcMap will look for the same file when it refreshes (you can also save to alternate locations if you wish)
4. After the data has been written, you can tell Esri ArcMap to refresh

```
EsriDisplay esriDisplay = ESRIDisplay.getInstance();
esriDisplay.updateDisplay();
```

### **Displaying Agents with OpenMap**

In order to work with the OpenMap display, agents need to implement the OpenMapAgent interface. The following steps should be followed in order to display the agents with OpenMap:

1. The Shapefile data must be in decimal degree coordinates, OpenMap does not handle pre-projected data.
2. Create the agents and add them to a list (as shown in the section on Data)
3. Create a new instance of OpenMap Display:

```
OpenMapDisplay omDisplay = new OpenMapDisplay();
```

2. Add the agents to a layer:

```
omDisplay.addlayer(gisAgents, "AgentLayer");
```

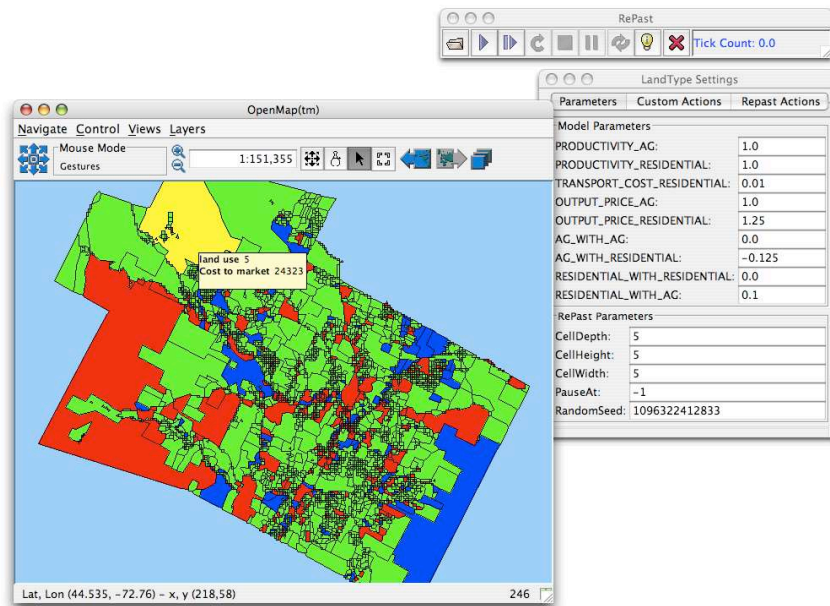
3. Update Agents. After changes have been made to the agents, you can update the layer containing them:

```
omDisplay.updateLayer(gisAgents, "AgentLayer");
```

### **Using The OpenMap Display**

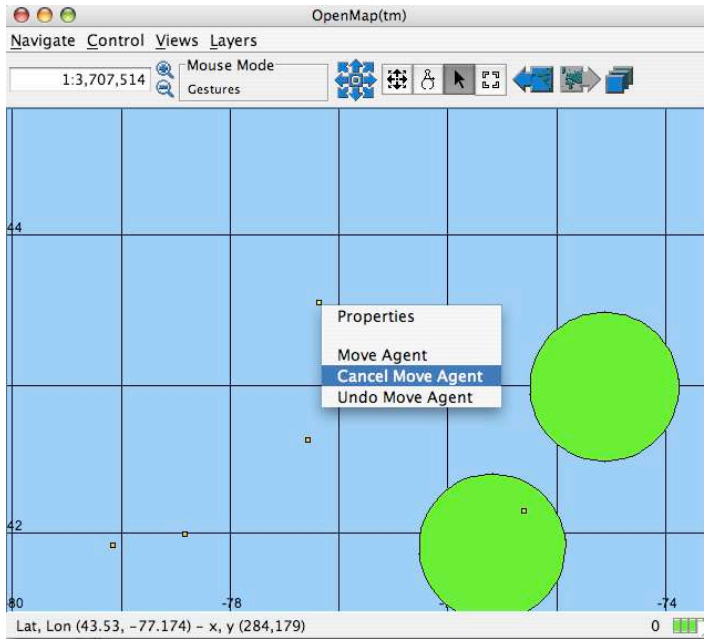
Agents displayed in OpenMap will display a tooltip when the mouse is moved over it. The tooltip text

is based on the `gisPropertyList` returned from the GIS agent from the function `OpenMapAgent#gisPropertyList`. The `gisPropertyList` is used here as follows: the list is expected to consist of a pair of strings for each entry. The first element is the name to be display ("land use" in the example below) and the second element is the name of a function ("getLanduse")



**Image 1: This image shows a number of different agent types: The smaller ones use EsriPoints for their graphics, the larger use OMCircles. The EsriPoints can be written to a shapfile, while the OMCircles cannot.**

Agents can be moved in the display. To do this, make sure that the cursor is an arrow. Also, make sure that the layer you want to modify is on top (use the Edit layers option under the layers menu to change the relative position - top to bottom - of layers). Right click on the agent you want to move, you will get the popup menu. Choose move agent. Then left click wherever you want to move it. Note, that polygons cannot be moved. You can also view the properties of an agent in a separate window (these are the same properties as shown in the tooltip).



**Image 2: The pop-up window shows options for moving agents, as well as showing agent properties.**

### References

ESRI – The GIS software leader, <<http://esri.com>>, last access date: Sep 15, 2004

GeoTools – Home, <<http://geotools.codehaus.org/>>, last access date: Sep 15, 2004

OpenMap, <<http://openmap.bbn.com/>>, last access date: Sep 15, 2004