

Creating a Basic GIS Agent Environment

By Karl D. Liebert, modpop@charter.net

With considerable consultation and source code from Nick Collier, nick.collier@gmail.com

This doc is meant to bridge the gap between the Repast S tutorial and the explanation of the Geography projection in the Reference document. A little experience and the advice of Nick Collier, the primary author of the GIS aspects of Repast S, made it clear that Java Classes are the best choice for more complex models as well as GIS models. So if you are trying to construct a complex GIS model, get ready to write some Java code.

This example includes three files. The model.score, the GISContextCreator class, and the GisAgent class.

Create a new Project

If you are in the Eclipse Java Perspective, select

File->New->Other...,

then select

Repast Symphony->Repast Symphony Project.

OR

If you are in the Eclipse Repast Perspective, you can simply select

File->New->Repast Symphony Project.

The Project Name used in this example is GIS_Test. Accept defaults for the remainder of the Project creation dialog boxes and Finish.

Add Necessary Items to the model.score

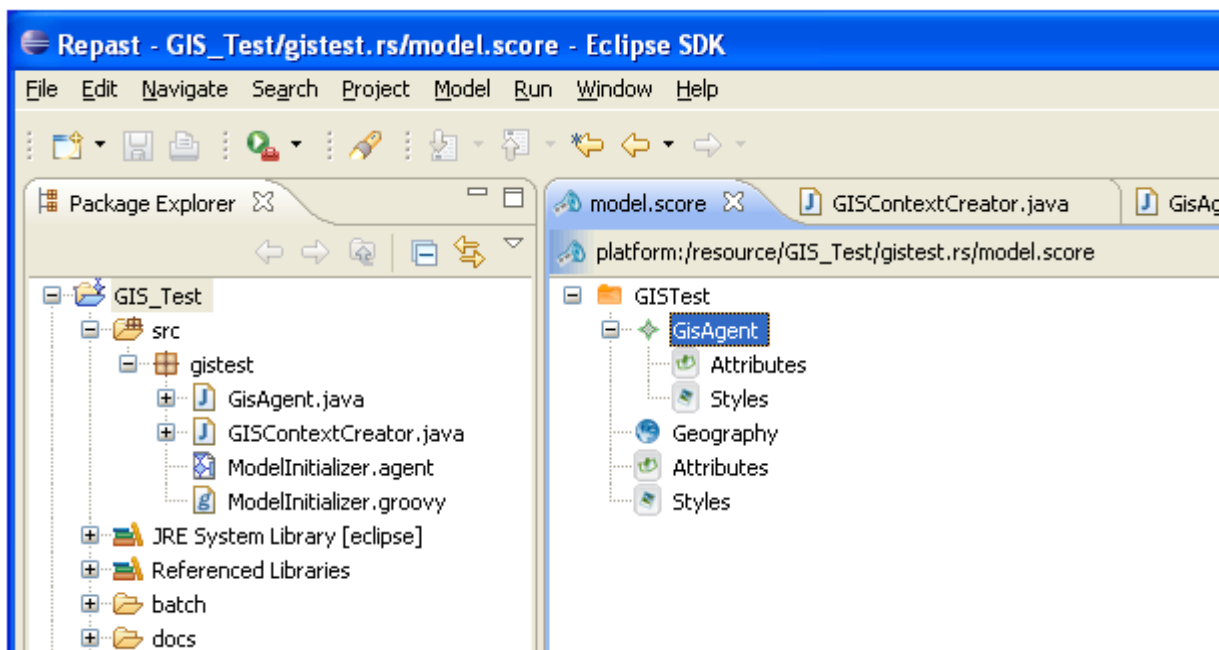
The model.score file is key to providing the Repast Runtime to manage your agents and projections. In the case of Java classes, *the names must match the associated classes you create.*

1. Right click on GISTest in the model.score editor tab then select
Create Member->Projection – Geography.
2. In the Properties tab for the new Geography, change the Label to **Geography** and the Plural Label to **Geographies**. The default name will cause an error when the display initializes. (The names are arbitrary, but must be consistent throughout this example.)
3. Right click on GISTest in the model.score editor tab, then select
Create Member->Agent.
4. In the properties tab for the new Agent, change the label to **GisAgent**.

5. Verify the xml code for the model.score file by opening it in Notepad. It's path will be something like **C:\RepastS\workspace\GIS_Test\gistest.rs\model.score**. It should look something like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<score:SContext xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:score="http://scoreabm.org/score" label="GISTest" ID="gISTest"
pluralLabel="GISTests">
  <implementation package="gistest" className=""
basePath="../GIS_Test" mode="AUTO"/>
  <agents label="GisAgent" ID="gisAgent" pluralLabel="GisAgents">
    <implementation className="GisAgent"/>
  </agents>
  <projections xsi:type="score:SGeography" label="Geography"
ID="geography" pluralLabel="Geographies"/>
</score:SContext>
```

6. Meanwhile, your model.score editor should look like this, minus the java classes in the Package Explorer (We haven't gotten there yet.):



Add the Agent and ContextCreator Classes

The agent can be a very simple class. I tried to keep this as simple as possible, but found the agent needed to have, minimally, one attribute and the associated Getter and Setter to avoid an error.

1. To create the GisAgent, right click on the **gistest** package in the Package Explorer. Select **New->Class**.
2. Enter **GisAgent** in the Name field and select Finish.
3. Complete the GisAgent as below:

```

package gistest;

public class GisAgent {
    int someData;

    public GisAgent() {
        someData = 0;
    }

    public int getSomeData() {
        return someData;
    }

    public void setSomeData(int someData) {
        this.someData = someData;
    }
}

```

4. To create the GISContextCreator, right click on the **gistest** package in the Package Explorer. Select **New->Class**.
5. Enter **GISContextCreator** in the Name field.
6. In the Interfaces list, select the **Add...** button.
7. In the field under Choose Interfaces, enter **ContextBuilder**. By the time you get to 'B', the Matching Items list will have narrowed it down to right class in repast.simphony.dataLoader. Select **ContextBuilder** from the Matching Items list and hit the **Add** button. Finally hit the Finish button.
8. In the editor tab for GISContextBuilder, complete the class as follows:

```

package gistest;

import com.vividsolutions.jts.geom.Coordinate;
import com.vividsolutions.jts.geom.GeometryFactory;
import com.vividsolutions.jts.geom.Point;

import repast.simphony.context.Context;
import repast.simphony.context.space.gis.GeographyFactoryFinder;
import repast.simphony.dataLoader.ContextBuilder;
import repast.simphony.space.gis.Geography;
import repast.simphony.space.gis.GeographyParameters;

public class GISContextCreator implements ContextBuilder<GisAgent> {

    public Context<GisAgent> build(Context<GisAgent> context) {
        // Because the Geography acts as a container for the
        // GisAgent and its
        // associated Coordinate, you need to tell the Factory
        // what it will be
        // storing, a <GisAgent>.
        GeographyParameters<GisAgent> geoParams = new
            GeographyParameters<GisAgent>();
        Geography<GisAgent> geography =

```

```

GeographyFactoryFinder.createGeographyFactory(null).createGeography("Geog
raphy", context, geoParams);

    // fac creates Coordinates to hold agent positions.
    GeometryFactory fac = new GeometryFactory();
    for (int i = 0; i < 10; i++) {
        // Added the agent to the context.
        GisAgent agent = new GisAgent();
        context.add(agent);
        // Then position the agent.
        // Since an agent is a POJO and its
        // position is held in the geometry
        // rather than the object itself, point
        // agent's position doesn't need
        // to be an attribute of the agent itself.
        Coordinate coord = new Coordinate(-103.823 +
            i / 100.0, 44.373);
        Point geom = fac.createPoint(coord);
        geography.move(agent, geom);
    }

    return context;
}
}
}

```

Launch the Runtime and Edit the Scenario

There are two tasks in this section, adding the ContextBuilder as a Data Loader and defining the display.

1. Launch the runtime window by selecting the little down arrow next to the green play button, the selecting **Run GIS_Test** from the menu.
2. To add the data n the Repast Symphony runtime control window, right click on Data Loaders and select **Set Data Loader**.
3. In the “Select Data Source Type” list, select “**A Specific Java Class**”. Then hit Next.
4. In the “Class Name” list, all of the Data Loaders cted thus far will be listed in the pull-down menu. Select **gistest.GISContextCreator** from the list. Hit the Next button and then the Finish button.
5. Hit the Save button (the diskette).
6. To create the Display, right-click on “Displays” in the Scenario Tree and select **Add Display**.
7. In the “Display Details” dialog box under “Projections and Value Layers”, Select **Geography** in the left side list and hit the arrow button to move it to the right side list. Then hit the Next button.
8. In the “Agent Style” dialog, GisAgent should be selected in the Layer order list. The Agent Name field should be gistest.GisAgent. And the Geometry should be POINT. Hit the Edit button. In the agent style editor that pops up change the Fill Color and select the OK button. (I think it is necessary currently to change some attribute of the Agent Style to avoid a bug.) Select the Next button.

9. In the Schedule Details, accept the defaults and hit the Finish button.
10. Save the scenario.

Results

OK. We have not added a step function, so nothing moves and there is not behavior. But procedures for those are covered in the Tutorial and should be straightforward.

1. Hit the Initialize button (round blue button with little man figure in it).
2. You should observe that the GISTest:A Display is shown with a row of squares in whatever color you chose for the agent style. As shown here:

