Exercise: Working with Graphics and the GraphicsLayer

Graphics are points, lines, or polygons that are drawn on top of your map in a layer that is independent of any other data layers associated with a map service. Most people associate a graphic object with the symbol that is displayed on a map to represent the graphic. However, each graphic in ArcGIS Server can be composed of up to four objects including the geometry of the graphic, the symbology associated with the graphic, attributes that describe the graphic, and an info template that defined the format of the InfoWindow that appears when a graphic is clicked. Although a graphic can be composed of up to four objects it is not always necessary to do so. The objects you choose to associate with your graphic will be dependent upon the needs of the application that you are building. For example, in an application that displays GPS coordinates on a map you may not have a need to associate attributes nor display an InfoWindow for the graphic when clicked. However, in most cases you will be defining the geometry and symbology for a graphic.

In this exercise you will learn how to create and display graphics on a map. We are going to create a thematic map showing population density by county for the State of Colorado. You will also be introduced to query tasks. As you will learn in the next section of this course, tasks can be executed against ArcGIS Server, and include things like spatial and attribute queries, identification of features, geocoding, and more. Finally, you will learn how to attach attributes to your graphic features and display them in an InfoWindow.

Exercise A

Step 1: Add References to ArcGIS Server Resources

- Navigate to the C:\Program Files\Apache Software Foundation\Apache2.2\htdocs\lab\intro_agis_javascript\exercises folder and open graphicexercise.htm in your favorite HTML or text editor. We have pre-written some of the code that you will use in this step in the interests of focusing on the creation of graphic objects.
- Add references to esri.map and esri.tasks.query. Just before the function init( ) line please add the following two lines of code.

```
dojo.require("esri.map");
dojo.require("esri.tasks.query");
```

  We have covered the esri.map resource in past exercises so no additional explanation should be necessary. However, the esri.tasks.query resource is new, and we won't cover this until the next section of the course. However, in order to complete this exercise it is necessary for us to introduce this to you at this point. This resource enables you to perform spatial and attribute queries against a data layer.

Step 2: Create Symbology Object

Inside the init( ) function you will see that we have already created a map object for you and added a layer that points to a map service provided by ArcGIS Online that will serve as the basemap for this exercise. We have also set the initial map extent to display the State of Colorado. In this step you will

create a symbology object that will be used to display the county boundaries. In this exercise we will be displaying polygons representing county boundaries so we'll be using an instance of the SimpleFillSymbol class to symbolize our counties. Initially we're going to create transparent county boundaries.

- Just below the map.add layer line please add the following line of code:

```
defPopSymbol = new esri.symbol.SimpleFillSymbol().setColor(new dojo.Color([255,255,255, 0])); //transparent
```

This creates a new SimpleFillSymbol and assigns it to the variable defPopSymbol. We use RGB values of 255, 255, 255, 0 to ensure that the fill color will be completely transparent. This is accomplished through the "0" value which ensures that our coloring will be fully transparent. Later we will add additional symbol objects so that we can display a color-coded map of county population density. For now though we simply want to create a simple symbol so that you can understand the basic procedure for creating and displaying graphics on a map. At this point your init function should appear as follows:

```
function init() {
      //create map, set initial extent and disable default info window behavior
      map = new esri.Map("map", {
        extent: new esri.geometry.Extent(-109.7897763335139, 36.48523422995126, -101.5466622211952, 41.25539291779116, new esri.SpatialReference({wkid:4326}))});

      map.addLayer(new esri.layers.ArcGISTiledMapServiceLayer("http://server.arcgisonline.com/ArcGIS/rest/services/ESRI_StreetMap_World_2D/MapServer"));

      defPopSymbol = new esri.symbol.SimpleFillSymbol().setColor(new dojo.Color([255,255,255, 0])); //transparent

}
```

Step 3: Create and Execute a Query Task

In this step you're going to get a preview of how the Query task can be used in an application. We'll cover this task in detail in the next section of the course, but for now here is an introduction. The Query task can be used to perform spatial and attribute queries on a data layer in a map service. In this exercise we are going to use a Query task to perform an attribute query against a county boundary layer provided through an ESRI service.

- Let's first examine the map service and layer that we will use in our query. Open a web browser and point to: http://sampleserver1.arcgisonline.com/ArcGIS/rest/services/Specialty/ESRI_StateCityHighway_USA/MapServer
  This map service provides census information for U.S. states and counties and also includes a highway layer. In this exercise we are interested in the counties layer which has an index number of 2. Click the counties link to get detailed information about this layer. There are a lot of fields in this layer, but we are really only interested in a field that will allow us to query by state name and a field that gives us population density information. The STATE_NAME field gives us the state name for each county, and the POP90_SQMI field gives us population density for each county.
- Return to your code editor. Below the line of code where we created our symbol, initialize a new Query task by adding the following line of code:

```
var queryTask = new esri.tasks.QueryTask
("http://sampleserver1.arcgisonline.com/ArcGIS/rest/services/Specialty/ESRI_StateCityHighway_USA/MapServer/2");
```

What this line has done is create a new Query task object that points to the ESRI_StateCityHighway_USA map service that we just examined in our browser, and specifically points to layer index 2 which is our county layer.

- All Query task objects need input parameters so that they will know what to execute against the layer. This is accomplished through a Query object. Add the following line of code just below the line you just entered:

```
var query = new esri.tasks.Query();
```

- Now, we'll define some of the properties on our new Query object that will enable us to perform an attribute query. Add the following three lines of code:

```
query.where = "STATE_NAME = 'Colorado'";
query.returnGeometry = true;
query.outFields = ["POP90_SQMI"];
```

The where property is used to create a SQL statement that will be executed against the layer. In this case, we're stating that we'd like to return only county records that have a state name of 'Colorado'. Setting the returnGeometry property to 'true' indicates that we'd like for ArcGIS Server to return the geometric definition of all features that matched our query. This is necessary because we need to plot these features as graphics on top of the map. Finally, the outFields property is used to define which fields we'd like returned along with the geometry. This information will be used later when we create the color coded map of county population density.

- Finally, we'll use the execute method on QueryTask to perform the query against the layer we have indicated (counties) using the parameters defined on our Query object. Add the following line of code:

```
queryTask.execute(query, addPolysToMap);
```

In addition to passing the Query object into ArcGIS Server, we have also indicated that addPolysToMap will serve as the callback function. This function will be executed after ArcGIS Server has performed the query and returned the results. It is up to the addPolysToMap function to plot the records using the FeatureSet returned to it. At this point your init( ) function should appear as follows:

```
function init() {
        //create map, set initial extent and disable default info window behavior
        map = new esri.Map("map", {
          extent: new esri.geometry.Extent(-109.7897763335139, 36.48523422995126, -101.5466822211952, 41.25539291779116, new esri.SpatialReference({wkid:4326}))});

        map.addLayer(new esri.layers.ArcGISTiledMapServiceLayer("http://server.arcgisonline.com/ArcGIS/rest/services/ESRI_StreetMap_World_2D/MapServer"));

        defPopSymbol = new esri.symbol.SimpleFillSymbol().setColor(new dojo.Color([255,255,255, 0])); //transparent

            //initialize & execute query
        var queryTask = new esri.tasks.QueryTask("http://sampleserver1.arcgisonline.com/ArcGIS/rest/services/Specialty/ESRI_StateCityHighway_USA/MapServer/2");
        var query = new esri.tasks.Query();
        query.where = "STATE_NAME = 'Colorado'";
        query.returnGeometry = true;
        query.outFields = ["POP90_SQMI"];
        queryTask.execute(query, addPolysToMap);
}
```

Step 4: Create Function to Add Graphics to the Map

As I mentioned in the last step, the callback function addPolysToMap will be executed when ArcGIS Server returns the FeatureSet containing the records that matched our attribute query. Most of this function has already been created for you, but let's examine what the code actually does before continuing. The addPolysToMap function takes a single parameter: featureSet. When a QueryTask is executed, ArcGIS Server returns a FeatureSet object back to your code. A FeatureSet object contains the graphic objects returned by the query. Inside the addPolysToMap function you will see the line "var features = featureSet.features; " . The features property returns an array with all the graphics contained within. After defining a new "feature" variable, we create a "for" loop that we will use to loop through each of these graphics and plot the graphic to the map. Inside this "for" loop you will need to add the following line of code just below "feature = feature[i];".

```
map.graphics.add(features[i].setSymbol(defPopSymbol));
```

As I mentioned in the lecture you have to add each graphic that you create to the GraphicsLayer object. This is done through the add( ) method as you can see above. You will also notice that we are attaching the symbol we created earlier to each of the graphics (county boundaries). Your addPolysToMap function should appear as follows:

```
function addPolysToMap(featureSet) {
        var features = featureSet.features;
            var feature;
        for (var i=0, il=features.length; i<il; i++) {
          feature = features[i];
              map.graphics.add(features[i].setSymbol(defPopSymbol));
}
```

Step 5: View Map

- Save the file
- Open a web browser and point to: http://localhost/lab/intro_agis_javascript/exercises/graphicexercise.htm
- You should see the map below. If not, you may need to recheck your code for accuracy. Notice that all the county boundaries for the State of Colorado have been place on top of our basemap. Each county boundary is actually a graphic with a simple fill symbol set to fully transparent. In Exercise B we are going to expand on this by creating additional symbols to each of our counties based on population densities. The end result will be a thematic map of county population

density.

Exercise B

Step 1: Add New Symbol Objects

- You will be using the same graphicexercise.htm file that you used in Exercise A. If it isn't already open in your editor, please navigate to C:\Program Files\Apache Software Foundation\Apache2.2\htdocs\lab\agis_javascript\exercises and open the file graphicexercise.htm.
- Comment out the defPopSymbol variable inside the init( ) function and add five new symbols as follows:

```
//defPopSymbol = new esri.symbol.SimpleFillSymbol().setColor(new dojo.Color([255,255,255, 0])); //transparent
 onePopSymbol = new esri.symbol.SimpleFillSymbol().setColor(new dojo.Color([255,255,128, .85])); //yellow
 twoPopSymbol = new esri.symbol.SimpleFillSymbol().setColor(new dojo.Color([250,209,85, .85]));
 threePopSymbol = new esri.symbol.SimpleFillSymbol().setColor(new dojo.Color([242,167,46, .85])); //orange
 fourPopSymbol = new esri.symbol.SimpleFillSymbol().setColor(new dojo.Color([173,83,19, .85]));
 fivePopSymbol = new esri.symbol.SimpleFillSymbol().setColor(new dojo.Color([107,0,0, .85])); //maroon
```

What we're doing here is basically creating a color ramp of symbols that will be assigned to each county based upon population density. We are also applying a transparency value of .85 to each symbol so that we will be able to see through each of the counties somewhat. This will enable us to see the base map below containing city names.

Step 2: Review Query Properties

Recall that in Exercise A we created QueryTask and Query objects, and that we defined an outFields property on Query to return the "POP90_SQMI" field. This will now come into play as we use the values

returned in this field to determine the symbol applied to each county based on the population density of that county.

Step 3: Update addPolysToMap Function

Update the addPolysToMap function as you see below and then we'll discuss what we've done.

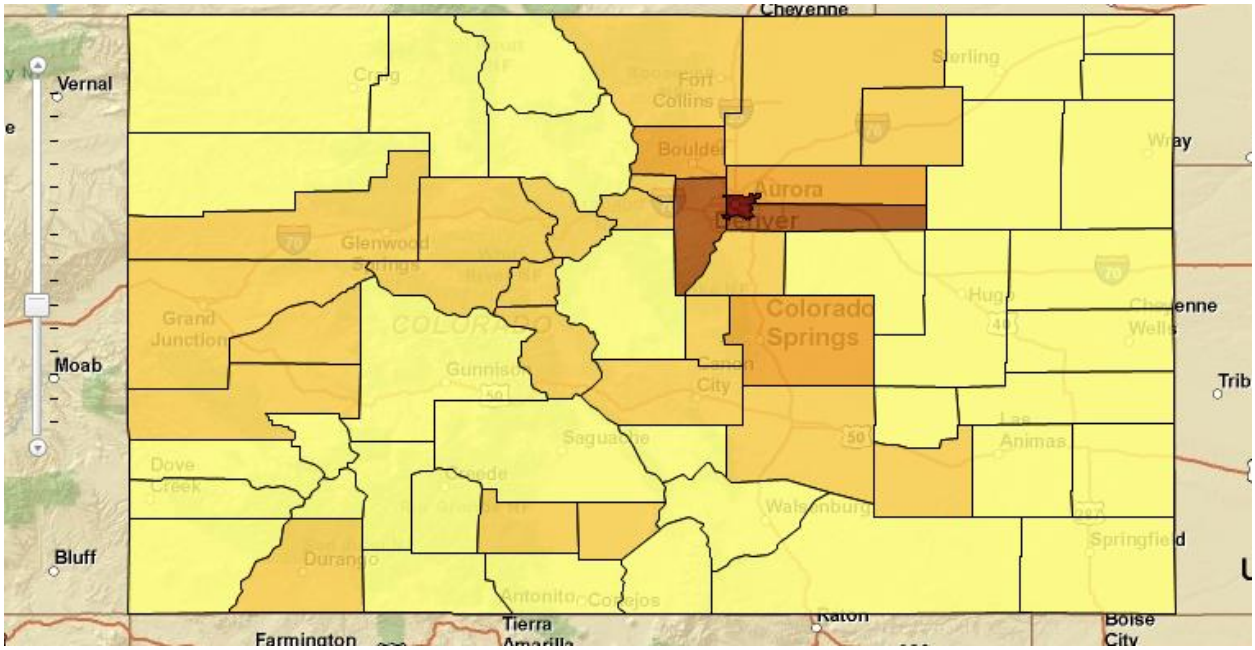
```
function addPolysToMap(featureSet) {
        var features = featureSet.features;
            var feature;
        for (var i=0, il=features.length; i<il; i++) {
                feature = features[i];
                attributes = feature.attributes;
                pop = attributes.POP90_SQMI;

                if (pop < 10)
                {
                        map.graphics.add(features[i].setSymbol(onePopSymbol));
                }
                else if (pop >= 10 && pop < 95)
                {
                        map.graphics.add(features[i].setSymbol(twoPopSymbol));
                }
                else if (pop >= 95 && pop < 365)
                {
                        map.graphics.add(features[i].setSymbol(threePopSymbol));
                }
                else if (pop >= 365 && pop < 1100)
                {
                        map.graphics.add(features[i].setSymbol(fourPopSymbol));
                }
                else
                {
                        map.graphics.add(features[i].setSymbol(fivePopSymbol));
                }

        }
}
```

I've highlighted the changes that you have made to your addPolysToMap function. What we've done with this code block is obtain the population density information from each graphic and save it to a variable called "pop". An if/else if/else code block is then used to assign a symbol to the graphic based on the population density of that county. For example, a county with a population density (as defined in the POP90_SQMI field) of 400 would be assigned the symbol defined by "fourPopSymbol". Because we are in a "for" loop that examines every county in Colorado, each county graphic will be assigned a symbol.

Step 4: View Map

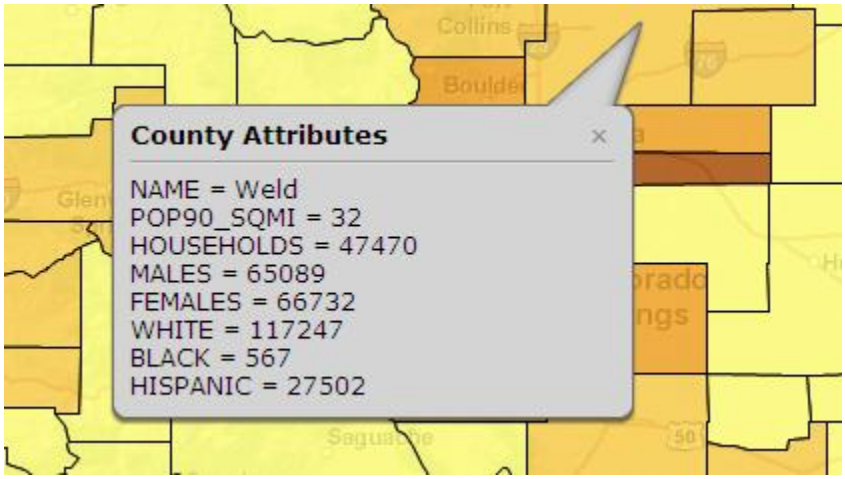
- Save the file

- Open a web browser and point to:
  http://localhost/lab/intro_agis_javascript/exercises/graphicexercise.htm
  You may need to hit the Refresh button on your browser if you still have the browser open from Exercise A.
- You should see the map shown in the figure below. If not, please double check your code for accuracy.



Step 5: Display Graphic Attributes in InfoWindow

In this step you will learn how to attach attributes to a graphic and display them in an InfoWindow when the graphic is clicked.



An InfoWindow is an HTML popup window that displays when you click a graphic on the map. Normally it contains the attributes of the clicked graphic, but it can also contain custom content that you specify

as a developer. The content of these windows is specified through an InfoTemplate object which specifies a title for the window and the content to display in the window. You can create these InfoTemplate objects using one of the three constructors seen below.

esri.InfoTemplate()	Creates a new empty InfoTemplate object.
esri.InfoTemplate(title, content)	Creates a new InfoTemplate object. All parameters are required and must be specified in the order given.
esri.InfoTemplate(json)	Creates a new InfoTemplate object using a JSON object.

The easiest way to create an InfoTemplate object is to use a wildcard for the content that will automatically insert all the fields of a dataset into the InfoWindow. We are going to add some additional output fields so that more content can be displayed in the InfoWindow.

- Alter the Query.outFields line to include the following fields:

```
query.outFields = ["NAME", "POP90_SQMI", "HOUSEHOLDS", "MALES", "FEMALES", "WHITE", "BLACK", "HISPANIC"];
```

- Now we're going to create a new instance of InfoTemplate through a constructor on this class. Add the following line of code just below the QueryTask.execute line.

```
resultTemplate = new esri.InfoTemplate("County Attributes", "${*}");
```

The first parameter passed into the constructor ("County Attributes") is the title for the window. The second parameter is a wildcard indicating that all the attribute's name value pairs should be printed to the window. Therefore, the new fields we added to Query.outFields should all be included in the InfoWindow when a graphic is clicked.

- Finally, we use the Graphic.setInfoTemplate( ) method to assign the newly created InfoTemplate to a graphic. Alter your if/else if/else statement as follows

```
if (pop < 10)
{
   map.graphics.add(features[i].setSymbol(onePopSymbol).setInfoTemplate(resultTemplate));
}
else if (pop >= 10 && pop < 95)
{
   map.graphics.add(features[i].setSymbol(twoPopSymbol).setInfoTemplate(resultTemplate));
}
else if (pop >= 95 && pop < 365)
{
   map.graphics.add(features[i].setSymbol(threePopSymbol).setInfoTemplate(resultTemplate));
}
else if (pop >= 365 && pop < 1100)
{
   map.graphics.add(features[i].setSymbol(fourPopSymbol).setInfoTemplate(resultTemplate));
}
else
{
   map.graphics.add(features[i].setSymbol(fivePopSymbol).setInfoTemplate(resultTemplate));
}
```

Step 6: View Map and Display InfoWindow

- Save the file

- Open a web browser and point to:
  http://localhost/lab/intro_agis_javascript/exercises/graphicexercise.htm
  You may need to hit the Refresh button on your browser if you still have the browser open from Exercise A.
- Click any of the counties in the map and you should see an InfoWindow similar to the figure below.