

**AT&T FACULTY-STAFF AWARDS IN INSTRUCTIONAL TECHNOLOGY**  
**2012-2013 Faculty-Staff Competition**  
**Course APPLICATION FORM**

Course Identifier: (e.g. TLC801) GEO826

Course Name: Geocomputation

Department: Geography College: Social Science

Primary contact name, phone number, and email (*normally this will be the lead instructor*)

Joseph P. Messina 353-1715 jpm@msu.edu

Faculty and Staff Involved in Developing and Offering the Course *please list full name, position at MSU, email address, and project role for each person*

NAME	MSU Affiliation	PROJECT ROLE
Joseph P. Messina	GEO	Designed and delivered the course

Which Competition Are You Entering (select one):

FULLY ONLINE COURSE (no required face to face component)

BLENDED/HYBRID OR FLIPPED COURSE (some face to face learning is replaced by online learning)

TECHNOLOGY-ENHANCED LEARNING INNOVATION (one specific technology innovation in a face-to-face, blended, flipped, or online course)

Semester(s) offered in 2012-2013 and number of students enrolled:

SEMESTER	# STUDENTS
FS 2012	5

Please address these categories:

I. Course Description (400 word limit)

Geocomputation is a course designed to expose graduate students to the theoretical and methodological constructs supporting geographic information systems (GIS) science. Most students entering the course have moderate levels of prior experience with commercial GIS software packages, but the theoretical underpinnings behind the development of these packages and the vast array of spatial data supporting them are missing from their experience set. Students

are assessed primarily through a series of exercises. These exercises account for at least 60% of their grade. All but the first exercise is technology enhanced. The first exercise is used in the second exercise, which is technology enhanced. Most of the materials produced for FS 2012 were also dynamically web-enabled. This includes all but one lecture.

## II. Learning and Interaction Goals of the Course or Technology-enhanced Innovation

I offer this course every fall. For all previous instantiations of the course, I used the programming language “IDL” for all the exercises. There is no programming prerequisite for the course, so I spend some time each semester devising methods for the students without any programming experience to rapidly gain the proficiency needed to complete the assignments. Although IDL is an interpreted language and easier to learn than C, for example, I would always find a student(s) who would simply give up by the middle of the semester. The combination of math, computer science, and GIScience simply proved to be too much. I struggled with balancing what I thought should be taught and what would challenge the best students, with crafting a class that would lift the least confident or prepared. I knew from reviews and post-course discussions that the programming language was a contributing factor to all who fell behind.

I am a non-traditional programmer as well, and I selected IDL to maximize the learning opportunities. IDL is a traditional geographic information software. However, in the fall of 2011 I learned of new natural language processing within Mathematica 8.0. I decided to try and learn the Mathematica language and apply it in GEO826. During the spring and summer of 2012, I read a number of books on the language and prepared the course. Virtually everything is new. It is quite likely that I am the only person in the country teaching a geography course with Mathematica.

I converted every PowerPoint lecture (except one) into a Mathematica web enabled notebook. What this means is that every embedded programmatic example is now live using computable document (CDF) technology. Instead of providing a limited set of examples in text form, the student now can dynamically alter the programs or run the demonstration projects from any CDF enabled (free-download) browser. The course transitioned from static and traditional to dynamic and web-enabled. This dramatically enhanced programming fundamentals comprehension.

The natural language processor also proved very useful for student learning. In prior years, students would have to learn formal programming language while learning core GIScience concepts. For some, this “two unknowns” problem was a substantial learning barrier. Following is a screenshot natural language example. Creating a list of random numbers is a common task in GEO826. Using the natural language processor I simply can say I want the students to create a list of  $n$  uniform distribution random numbers, and the students having learned the uniform distribution can simply type a “normal” i.e. natural language request. At the orange = sign you will find a simple example. Further down the page (Input), you will see the actual programming language request. The great thing is that the interpreter not only returns the result, but also returns the correct programming language solution. The student not only clearly sees the results, but also quickly learns the programming language through the process.

In[1]:= `plot 10 random numbers from 0 to 1` »

↳ `10 random numbers from 0 to 1` ?

↳ Result

`{0.849456, 0.817408, 0.773538, 0.35952, 0.278511, 0.259178, 0.386649, 0.0350346, 0.915439, 0.743818}`

↳ Using closest WolframAlpha interpretation: `10 random numbers from 0 to 1` ?

More interpretations: [plot 10 random numbers](#) | [0 to 1 plot](#)

Assuming "random number" is `RandomReal` | Use `RandomInteger` instead

Input:

`RandomReal[{0, 1}, 10]`

`RandomReal[{0, 1}, 10]`

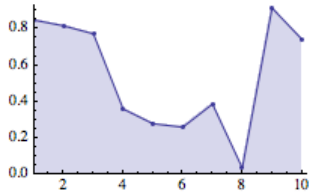
Result:

↳ `{0.849456, 0.817408, 0.773538, 0.35952, 0.278511, 0.259178, 0.386649, 0.0350346, 0.915439, 0.743818}`

`{0.849456, 0.817408, 0.773538, 0.35952, 0.278511, 0.259178, 0.386649, 0.0350346, 0.915439, 0.743818}`

Plot:

`ListLinePlot[{0.849456, 0.817408, 0.773538, 0.35952, 0.278511, 0.259178, 0.386649, 0.0350346, 0.915439, 0.743818}, Mesh -> All, Filling -> Axis, AxesOrigin -> {1, 0}]`



The third primary advantage of this transition was that I was able to increase the sophistication and total content of the course. I added new lectures, notably one on graph theory and another on data mining, that I simply did not have the time to cover in the past. I display some graph theory lecture highlights in section 3.

Finally, the most important outcome, though, is that for the first time no student fell significantly behind. Despite being a small class, the performance gains by the students, none of whom had any prior Mathematica experience, were substantial.

### III. Points of Interest and Innovation

The computable document format developed with rich and dynamic content allowed for truly interactive learning objects and direct student interaction. I have embedded below three distinct forms of dynamic content used in the class (beyond the natural language processor described earlier). These snapshots are extracted from the actual lectures and web deliverables. Recall, I moved my entire course from PowerPoint into these dynamic “notebooks.”

This first example “Different ways of clustering” is the simplest form – the student can dynamically replace the numbers after the commands and observe the output by experimenting with, for example, plotting options. During lecture this example is presented as a single element – i.e. a bit like a single PowerPoint slide in a traditional lecture model.

#### Different ways of clustering

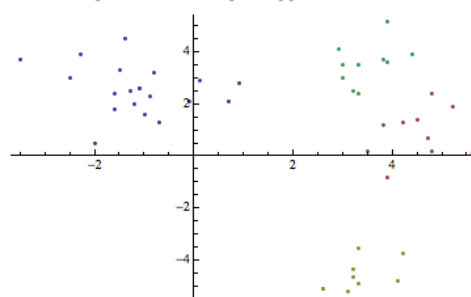
```
FindClusters[{1, 2, 10, 12, 3, 1, 13, 25}]
{{1, 2, 3, 1}, {10, 12, 13, 25}}
```

```
FindClusters[{1, 2, 10, 12, 3, 1, 13, 25}, 4]
{{1, 1}, {2, 3}, {10}, {12, 13, 25}}
```

```
FindClusters[{1 → a, 2 → b, 10 → c, 12 → d, 3 → e, 1 → f, 13 → g, 25 → h}]
{{a, b, e, f}, {c, d, g, h}}
```

```
data = {{-1.1, 2.6}, {3.9, -0.8}, {4.2, -3.7}, {3.3, 3.5}, {3.9, 5.2}, {4.1, -4.8}, {3.8, 3.7}, {5.6, 0.1},
{3.1, -5.2}, {-0.9, 2.3}, {2.9, 4.1}, {-2.3, 3.9}, {-2.5, 3.}, {2.6, -5.5}, {5.2, 1.9}, {-0.7, 1.3}, {0.9, 2.8},
{-1.5, 3.3}, {3.8, 1.2}, {2.6, -5.1}, {-0.8, 3.2}, {4.7, 0.7}, {3., 3.}, {3.9, 3.6}, {4.5, 1.4}, {4.2, 1.3},
{-1.1, 2.6}, {4.8, 2.4}, {3.3, -3.5}, {3.2, -4.6}, {3.3, -4.9}, {3., 3.5}, {0.7, 2.1}, {3.2, -4.3}, {-2., 0.5},
{-1.2, 2.}, {-1.6, 1.8}, {-3.5, 3.7}, {4.8, 0.2}, {3.3, 2.4}, {-0.1, 2.1}, {-1.3, 2.5}, {4.4, 3.9}, {3.5, 0.2},
{0.1, 2.9}, {-1., 1.6}, {-1.4, 4.5}, {3.2, 2.5}, {-1.6, 2.4}, {2.6, -5.1}};
```

```
ListPlot[FindClusters[data]]
```



```
ClusteringComponents[{1, 2, 3, 7, 8}, 2]
{1, 1, 1, 2, 2}
```

```
ArrayComponents[
$$\begin{pmatrix} 0 & a & a & a \\ 0 & a & a & a \\ 0 & a & a & a \\ b & b & b & b \end{pmatrix}$$
] // MatrixForm
```

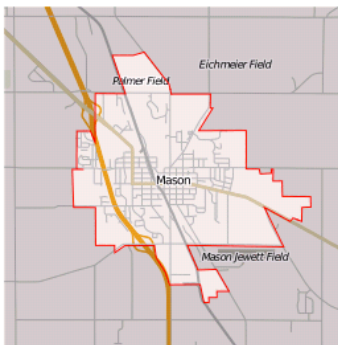
$$\begin{pmatrix} 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 \end{pmatrix}$$

This proved a very effective tool for learning basic programming constructs while teaching geospatial foundations. Each of the commands above is fully dynamic. A student can change any parameter and instantly visualize the impact.

The “Road Network” example is a bit more dynamic. This example extracts map image data from the web and then extracts distinct spatial objects. The student then explores and extracts information from the modeled object. This effort combines multiple learning objectives. I’ve placed a second example with a different place name (I recomputed this as I was writing this section – it is that simple and dynamic). As expected, though this example unlike the prior clustering example does require Internet access.

## Road Networks

```
image = WolframAlpha["Mason MI", {"Map:CityData", 1}, "Image"];
graph = MorphologicalGraph[Binarize[ColorNegate[image]], EdgeStyle -> Gray];
Row[{image, Spacer[20], HighlightGraph[graph, NeighborhoodGraph[graph, 180, 10], ImageSize -> {256, 278}, VertexSize -> {180 -> 10}]}
```



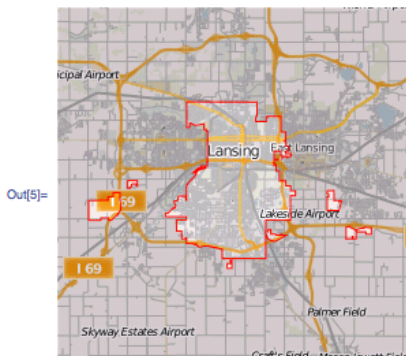
(based on current OpenStreetMap data)



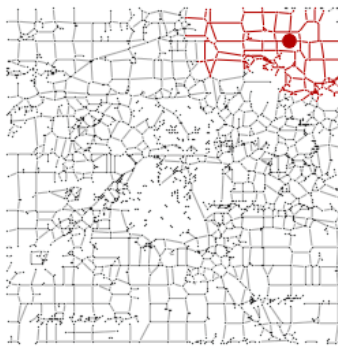
Graph representation of the road network in Mason, Michigan.

## Road Networks

```
In[3]:= image = WolframAlpha["Lansing MI", {"Map:CityData", 1}, "Image"];
In[4]:= graph = MorphologicalGraph[Binarize[ColorNegate[image]], EdgeStyle -> Gray];
In[5]:= Row[{image, Spacer[20], HighlightGraph[graph, NeighborhoodGraph[graph, 180, 10], ImageSize -> {256, 278}, VertexSize -> {180 -> 10}]}
```



(based on current OpenStreetMap data)



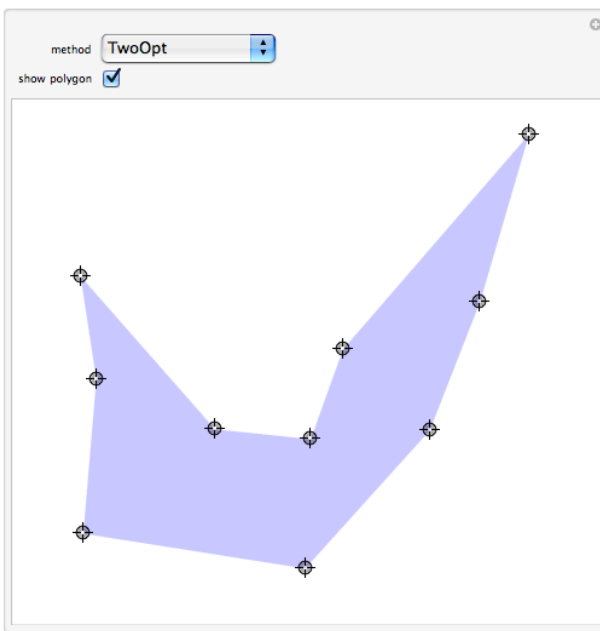
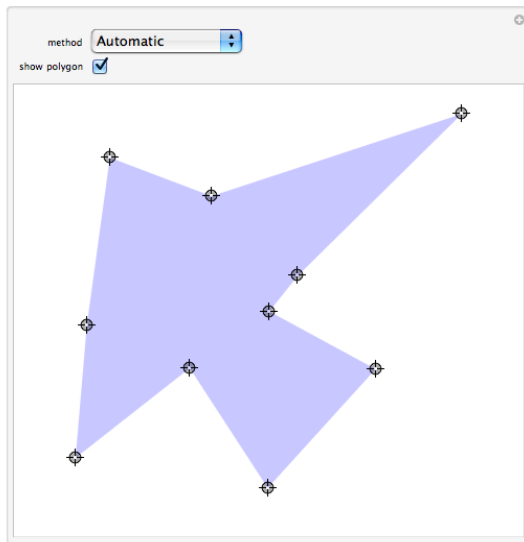
Graph representation of the road network in Lansing, Michigan.

The students were particularly pleased with this example. All selected their respective hometowns and explored the “neighborhood graph” models using familiar places.

The third example “Traveling Salesman” is the most complex and remains fully dynamic. Using either the free browser plug-in or the Mathematica software, the student can manipulate the figure by dragging points or changing parameterization with the dropdown menu. The student also is exposed to more complex programming tools (e.g., Manipulate) beyond what is required for the class. These are examples of demonstration projects used in many of the lectures.

### Traveling Salesman Problem

```
Manipulate[
Graphics[{Blue, Opacity[.2], Switch[poly, True, Polygon, False, Line][p[Last[FindShortestTour[p, Method -> method]]]}],
PlotRange -> {{-1, 1}, {-1, 1}}, ImageSize -> {450, 400}},
{method, {Automatic, "CCA", "TwoOpt", "SpaceFillingCurve", "OrOpt", "OrZweig", "Greedy", "GreedyCycle"}},
{{p, PadRight[{{0, 0}}, 10, RandomReal[{-1, 1}], {15, 2}]}}, {-1, -1}, {1, 1}, Locator, LocatorAutoCreate -> {1, ∞}},
{{poly, True, "show polygon"}, {True, False}},
ControllerLinking -> False]
```



#### IV. Accessibility

During the design phase of this class, I did not fully consider this issue, but there are two positive accessibility outcomes. The first is that the natural language processor allows more people to explore computational methods than traditional programming alone. The second benefit is to students who require visual aids while learning new materials. The fact that a student can repeatedly and dynamically recast spatial models, like the traveling salesman problem from earlier, will likely improve comprehension and retention.

#### V. Evidence of Effectiveness with Students

Admittedly, FS 2012 was a small class. However, my summary SIR score was the best I have received at 1.26. The students unanimously liked the software and mechanism for delivery. Since they had no experience with prior versions of the class they would not be able to offer comparative differences. However, the students never had to rely on group work to complete assignments, the amount of time I spent teaching programming was reduced by ~50%, and all of the students meaningfully advanced during the course of the semester.

One significant change was that the students were now expected to write their entire lab assignments within the dynamic Mathematica notebook structure. This afforded me an immediate test of successful completion and easy assessment of embedded work.

I use the SIRs form as a mechanism to find out what lectures people like, which to replace, which to enhance. In most prior course evaluation sets someone always complained of IDL or the lack of dedicated programming lectures. In those prior classes, students never selected the hard lectures for expansion. This time three students requested enhancements of the difficult lectures at the end of the course (graph theory and Cellular Automata Modeling). Nobody complained of the programming language or the difficulty learning it. To summarize, I increased the difficulty and total content in the course and yet the students asked for more.

#### VI. Plans for Sustainability

Fall semester 2013 will be the second iteration of the course in this form. Since I teach it once a year, it is updated with some frequency. Content aside, I expect to continue to enhance the technology component of the course with more dynamic content (of the third example type). I only learned the programming language this past year, so I am sure that the style of deliverable dynamic web content will improve. I always change my lectures based on student comments, and remove one lecture and add another, and this fall will be no different. One request that I intend to adopt is peer review of weekly programming solutions. Finally, my ultimate goal with this course and software combination is to make it largely self-contained and modular where a student, particularly a non-traditional learner, might be able to reconfigure the course elements and still gain equivalent proficiency and learning outcomes as traditionally trained students.