SOCIAL GIS: CREATING A MAPPING TOOL TO ENGAGE MIDDLE AND HIGH SCHOOL STUDENTS WITH THEIR NEIGHBORHOODS AND STEM

By

Kathleen Tully

A Thesis Submitted to the Graduate Faculty of Rensselaer Polytechnic Institute

in Partial Fulfillment of the Requirements for the Degree of

MASTER OF SCIENCE

Major Subject: COMPUTER SCIENCE

Examining Committee:

___________________________
Ron Eglash, Thesis Adviser

___________________________
Mukkai Krishnamoorthy, Thesis Adviser

___________________________
Sibel Adah, Thesis Committee Member

Rensselaer Polytechnic Institute
Troy, New York

April 2015
(For Graduation May 2015)
3.2.3 Software Architecture ................................................. 14
  3.2.3.1 views Module .................................................. 14
  3.2.3.2 serializers Module ........................................... 15

4. Use of Social GIS: Activity Outlines ................................. 18
  4.1 Environmental Justice ................................................ 18
    4.1.1 Method of Analysis ............................................. 18
    4.1.2 Extension of Social GIS ....................................... 20
      4.1.2.1 Constructing the Radius ................................. 20
      4.1.2.2 Finding Neighboring Points .............................. 21
      4.1.2.3 Filtering and Sorting Census Tracts .................... 22
    4.1.3 Going Forward .................................................. 24
  4.2 Create an Infographic ............................................. 24
    4.2.1 Explanation of Infographics ................................ 24
    4.2.2 Example Assignment .......................................... 26

5. Conclusion .............................................................. 28
  5.1 Reflection .......................................................... 28
  5.2 Going Forward ........................................................ 28

References ........................................................................ 30

Appendices

A. Project Code ............................................................ 33
LIST OF TABLES

4.1 Selected characteristics of those living near hazardous waste sites . . . . 18
LIST OF FIGURES

2.1 Social GIS interface showing farmers market locations .... 7
3.1 Retail Food Stores marked on the front page of our Ushahidi deployment 10
3.2 Client–server division of duties .................................. 11
3.3 Relational model showing basic SGIS models .......... 12
3.4 Excerpt from JSON object in Retail Food Stores [Open New York 2014] data set .............................................. 13
3.5 street_field, lat_field and lon_field values from JSON object shown in Figure 3.4 ........................................... 13
3.6 reach_field function to parse a JSON element based on a coded field location ............................................. 13
3.7 A simple view ......................................................... 15
3.8 A filtered view ......................................................... 15
3.9 A simple serializer ..................................................... 16
3.10 A more complex serializer ........................................ 17
4.1 Example host tracts .................................................. 19
4.2 Example using 50% areal apportionment ...................... 19
4.3 circle_as_polygon function to approximate a polygon around a point 21
4.4 Example polygon approximation of 1 mile radius .............. 21
4.5 Example of multiple nearby points using 50% areal apportionment ... 22
4.6 Cluster of hazardous waste sites in Los Angeles with polygon boundaries at 1 km, 3 km and 5 km ......................... 23
4.7 “How to Prepare for a Day Hike” infographic .................. 24
4.8 “Top 10 Most Read Books in the World” infographic ........ 25
4.9 “Top 10 Most Read Books in the World” infographic, view of bar graph 26
4.10 Infographic assignment example ................................ 26
Acknowledgments

First, I would like to thank my advisors, Professors Ron Eglash and Mukkai Krishnamoorthy, for their guidance and help with my thesis and throughout my time here at Rensselaer Polytechnic Institute. They brought me into the 3Helix program, allowing me to go into Albany High School to share my software with students. For this opportunity, I am also very appreciative of Vicki Brock’s work, Albany High School for allowing us to work with students, and the National Science Foundation’s support. I would also like to thank Professor Sibel Adalı for joining my thesis committee and for her very helpful input.

I also could not have created this software without early input from Professors Abby Kinchy and Michael Mascarenhas. I am very grateful for Professor Mascarenhas and his Environmental Justice students for being my early guinea pigs and for Professor Kinchy’s input about what educators needed from Social GIS.

Thank you also to Mary Liz Stewart and the Underground Railroad History Project of the Capital Region. In the summer of 2013, I was able to work with the summer youth program and test out the very first iterations of Social GIS.

Without help and input of Gouravjeet Singh, Charles Hathaway and the other 3Helix developers and fellows, Social GIS would not have been possible.
Abstract

Girls and underrepresented ethnic groups tend to disengage from STEM subjects throughout middle and high school. In order to increase their participation in STEM fields, something has to change before they’re deterred. Building on previous research showing individuals in these groups tend to respond more positively to STEM activities that connect to their lives or that can be used to help others, this thesis discusses the development of online GIS software created for use with middle and high school students that blends social justice, mapping and data science. We developed software and corresponding activities to spark an interest in math and computer science in all students. Activities ask students to compile and analyze real-world data to develop their own questions, critiques and conclusions with no wrong answers. The software was developed in a small team using mostly open-source tools, including Django and AngularJS. Over one year, another student and I concurrently developed an interactive Google Maps GUI front end and a RESTful back end.
CHAPTER 1

Background

There is a shortage of STEM graduates. Specifically, the software and engineering industries have complained about the dearth of skilled workers. In recent years, some of the largest of these companies have been called out for the lack of diversity at their companies. Google [2014], Microsoft [2014] and Facebook [Williams 2014] are a few of the most visible companies that have responded by publishing diversity reports. The makeup of their tech workforces are strikingly similar, with whites (53-60%) and Asians (34-41%) representing a combined 92-94% and women making up only 15-17% of tech workers. As Maxine Williams, Global Director of Diversity at Facebook, explains in an interview with Forbes, a diversity of perspectives and thought leads to greater innovation [Bercovici 2014]. She calls this “cognitive diversity” and this is the motivation for industry to foster diversity, but there are other motivations as well. Because there is a high demand for skilled STEM workers, in general, these careers have higher wages than many other career paths.

When the issue of diversity in STEM fields comes up, most jump right to “the pipeline.” Blickenstaff [2005] describes the common metaphor of the “leaky pipeline,” where students begin with an interest in STEM, but are deterred from continuing on to a career at several key points along the way. Many of these “leaks” happen even before high school. By creating software and activities that approach STEM topics, specifically mathematics and computing, differently, I hope to not only recruit new students, but I also aim to widen the perspective of those students who already have an interest in these topics. I also do not aim to push unwilling students into STEM, but to present a more diverse view of what STEM encapsulates.

1.1 Challenges Attracting and Retaining Girls STEM

There are many reasons why girls lose interest in STEM throughout middle and high school. Blickenstaff [2005] argues that girls are not just leaking from the pipeline, but are being pushed out. He states, “No one in a position of power along the pipeline has consciously decided to filter women out of the STEM stream, but the cumulative effect of many separate but related factors results in the sex imbalance in STEM that is observed today.” Dean and Fleckenstein [2007] agree that women are “filtered out” of this pipeline and even claims that the metaphor of a pipeline is in itself faulty. Grading bias against girls in STEM classes [Lavy and Sand 2015], a lack of female role models in STEM fields, pressure to conform to gender roles, and even open hostility [Blickenstaff 2005] are just a few examples of the adversity girls face. Later in life, as women begin their STEM careers, they continue to face these same obstacles, but with the added challenges of hostile workplaces and starting a family while also starting a career [Dean and Fleckenstein 2007].

A relatively equal number of male and female high school graduates have taken prerequisite math courses required to enter into STEM programs at the college level,
but many elect not to go into these programs [Metz 2007]. Those that do enter into STEM programs are more likely than their male counterparts to leave their field both before and after graduation from college. The students that do leave STEM programs cite teaching methods as large factor in their choice. These students have similar grades to those who stay, but have notably lower self confidence in the field [Blickenstaff 2005]. This aligns with similar research showing that girls with higher degrees of self-efficacy in the sciences correlated with higher interest levels [Weisgram and Bigler 2006]. Mitts and Haynie [2010] cite research identifying the characteristics of activities girls most preferred in a given technology education program: design or communication and social relevancy. Metz [2007] asserts that emotional and vocational engagement are key to promoting interest in STEM. She defines vocational engagement as viewing their planned career as aligning with their goals and emotional engagement “as a positive reaction to people, content and environment in the academic setting, and a view of the discipline as fun, intriguing and intellectually rewarding.”

Biological sciences do seem to be the exception; females studying these subjects outnumber males [Dean and Fleckenstein 2007]. Interest in biology only reinforces these concepts; females connect with the social aspect of biology inherent in the study of living things [Saraga and Griffiths 1981]. Biological sciences also have a clear altruistic element; research suggests girls may be more interested in scientific professions when altruism plays a key role [Weisgram and Bigler 2006]. On top of that, a girl choosing to go into biological sciences is encouraged and supported because it does not contradict her expected gender role [Saraga and Griffiths 1981].

To attract and retain female STEM students and, at the same time, create a more inclusive STEM program overall, we should examine and adjust, not only the learning environment, but the teaching methods and activities. In a classroom or after school setting, teachers and mentors need to remain conscious of grading and social bias and work to act objectively. In order to foster self-efficacy and confidence, activities should be challenging, but reasonable. Perhaps most importantly, the activities should contain elements to which girls respond more positively: design, communication, social relevancy, and altruism. Chapter 2 explains specific goals for the activities and the software in more detail.

1.2 Challenges Attracting and Retaining Underrepresented Ethnic Groups in STEM

Citing Eccles’ Expectancy Value Model, Zarrett and Malanchuk [2005] explain that all young people are susceptible to outside pressures including cultural, family and the student’s own “expectations for success.” Much like gender status, race, ethnicity and socioeconomic status play a huge role in a student’s achievement in school and in STEM fields in particular. Specifically talking about black students, Terry [2011] argues that the inclusiveness of mathematics “for all” ignores black students’ own heritage as people that strived for education as a means of empowerment and freedom. He explains that counterstories are critical in black students in four key
ways: “Counterstories build community among the marginalized by personalizing educational theory and practice . . . provide a context within which to challenge and transform the hegemonic wisdoms of those at society’s center . . . open new realities to marginalized peoples by helping them envision possible lives . . . teach marginalized people to actualize those new possibilities through synthesizing elements of stories with current realities, thereby producing richer actual lives.” Counterstories are much like the activities we have envisioned for this software tool, some of which are outlined in Chapter 4.

1.3 Diversification of STEM Applications

Asian and white males, groups that are very well represented in STEM and in the computing fields in particular, also stand to benefit from a new approach. As important as it is for underrepresented ethnic groups to understand the systematic roadblocks in their way, it is arguably more important for those who are currently well-represented to be aware. These students can become allies and, in essence, “infiltrate” the industry.

By showing how data and computing can be used to shed light on existing problems, interpret the world or legitimize movements, some of these students will take this to heart. If those students were on track to a STEM career, introducing these concepts enriches their thinking and may steer them towards a truly innovative use of their skills. For those students who are not headed for a STEM career, now that they see that data can act as evidence for some phenomena that had previously only been anecdotal, they may be more attracted to a STEM career. In both cases, students have a new perspective on the role of a person in STEM and on the world itself, which only adds to cognitive diversity, as described at the beginning of this chapter. Even the student who chooses not to pursue STEM is better prepared to be a consumer of data software and media and can even bring new perspective to other fields.

1.4 Introduction to 3Helix

3Helix, read as Triple Helix, is a National Science Foundation supported GK-12 program. GK-12 programs pair graduate students in STEM with K-12 schools to bring advanced material to the classrooms. Graduate students partner with a teach to present lessons and activities in the graduate student’s field of study.

Unlike most GK-12 programs, 3Helix focuses on low-income schools with struggling students. Using a mixture of classroom visits and a daily after school program, we engage students in STEM with hands on and computer-based activities. The variety of software tools created for 3Helix are called Culturally Situated Design Tools (CSDTs). These programs introduce both a cultural concept and a corresponding STEM concept, creating a deeper understanding of both. For example, one of these tools, Cornrow Curves, familiarizes students with the history of African cornrow braiding and the mathematical concept of fractals in the same stroke. Cornrows are
an accessible cultural component for most students, acting as a great jumping off point. A typical student progresses from guided use of the tool to more independent, open-ended activities. By giving students a low entry-point and high ceiling, CSDT activities are accessible, but engaging and flexible. In the end, students gain a deeper understanding of both the cultural history and the STEM concepts [Eglash and Bennett 2009].

1.5 Inception of Social GIS

Within the 3Helix program and alongside the CSDTs, the idea of Social GIS (SGIS) was born. SGIS deviates from the model of the CSDT in a few key ways. First, the software does not directly guide the user through any particular cultural context. It is intended that, at first, teachers will guide younger students with a specific activity. More advanced students have the freedom to explore available data. Second, there is no design element built-in to the software. CSDTs introduce math, science and programming concepts by allowing students to manipulate a visual stage. SGIS does not have this design element, but suggested lessons incorporate graphing and other visualizations as a way to understand the data and convey information.

Initially, our goal was to create a tool to allow students to explore real data. This quickly developed into a GIS application. By building a GIS application, we set out to apply Gutstein’s [2003] work with urban, latino middle school students. He infused his math class with activities and lessons using social data like housing prices, racial makeup and income. Students were asked to analyze this data through the lens of social justice, deepening their understanding of mathematics as well as the world around them. With Gutstein’s work as a model, we developed an example lesson we could use as a target for SGIS. The tool would show cell phone reception as a heat map layered with racial makeup and income. This kind of data is both relevant to the students lives and means to make them aware of disparities that affect their everyday lives. Unfortunately, at the time, the raw data to generate this kind of heat map was not publicly available. Even though this particular data set was unavailable, there are plenty of other data sets available to use in lessons that emphasize critical thinking, data analysis and social justice issues.
CHAPTER 2
Goals

2.1 Students as Active Agents of Social Justice

With this project, my overarching goals were to create a GIS tool that would engage students in mathematics, computing and local social issues. By incorporating a social justice angle, it improves the learning experience for all groups described in Sections 1.1 - 1.3. Girls respond to the real-world applications and the live data. Students in underrepresented ethnic groups are empowered by learning about the systems that affect their place in the world. By giving these students tools and support to explore actual information, it legitimizes the struggles in their families and communities through empirical data. For those students that face less barriers on the way to a STEM career, this curriculum offers an alternative perspective on what exactly that career could entail. Beyond deepening their understanding and appreciation for data, my aim is to cultivate active agents of social justice.

2.1.1 Awareness

Gutstein’s [2003] project asked middle school students to think about the relationship between race/ethnicity and housing prices and determine if they are related. Through discussion, the students learned how the very different data might be interpreted to show the same things:

- One student deduced that if prices were higher for similar homes purchased by ethnic groups at around the same time, this would be racism.
- Another deduced that if whites were purchasing similar homes at prices that were impossible for working class latinos in the area to afford, these prices may have been inflated to keep non-whites out of the neighborhood.
- A third student even described possible negative effects of gentrification on working class neighborhoods without having previously talked about this concept in class.

Students need to be aware of social issues and their possible causes.

2.1.2 Skills

We are bombarded with data all the time. Infographics are passed around on social media. News outlets compile and disseminate graphics and aggregate statistics. Even credit card statements have Google Analytics-like reports. Today, literacy extends beyond the need to read. Using SGIS, students should be able to develop mathematical thought and the ability to really think critically about how data is aggregated and interpreted. If a student imagines she is writing a news article with her interpretation and her classmate is as well, how would these headlines compare? She would observe that it is possible to draw different conclusions using the same data.
2.1.3 Ownership

In Gutstein’s [2003] class, students felt a sense of ownership because it was based on something tangible to them: local data. They could see their roles in the project and how an issue like this might affect their lives, asking questions like: “Might my parents have been overcharged for their home because we’re latino?” With discussions like this, students question the world around them, fostering an appetite for change.

2.2 Software Design

As explained in Chapter 1, in order to engage girls and underrepresented ethnic groups, activities using the software should contain the following activity characteristics which are inherently entwined with the learning outcomes and the design of the software:

- have an altruistic motive,
- be socially relevant,
- apply in the real-world,
- empower students through knowledge and critical thinking,
- foster self-efficacy and self-confidence.

With these characteristics and the goals described in Section 2.1 in mind, I outlined specific targets for the software design.

2.2.1 Low Barrier to Entry

As a GIS tool, it is imperative that students understand the basics of mapping, but most of the challenging GIS and spatial concepts, described in detail in Section 4.1, are handled under the hood. Students do not need to understand spherical geometry, fifty percent areal apportionment (see Section 4.1.1) or even latitude and longitude. Requiring students to handle these details would be tedious and an enormous departure from the types of mapping tools with which they already have experience. It would create a barrier to entry that is too high for most middle and high school students. Instead, we made the front end, shown in Figure 2.1, easy to use and familiar. To re-center the map, users enter an address in a field that has auto-complete populated from Google Maps. The map itself uses the familiar Google Maps interface. As described in Chapter 4, activities can be as simple as counting up the grocery stores in an area or a more advanced analysis of census data. Even with what may appear to be a simple math skill, like counting points, students are forced to consider complex problems with even more complex solutions. For example, if a student finds that one area has more grocery stores and less fast food chains than another student found in a different area, this fosters conversations about issues that have no simple solutions. Would simply adding more grocery stores in an area improve food choices? Could the grocery store survive in competition with fast food restaurants in a working class area that might value cost and convenience over nutritional value? Whose responsibility is it to address this complex issue?
2.2.2 Import a Variety of Data

One very important aspect of SGIS is that the user is analyzing current, real-world data. This reinforces the relevancy of the activities. I aimed to create a universal solution for collecting and presenting geographic data. This does not mean that all geographic data is in the database at any point, but that with slight modification, any geographical data could be imported. This flexibility is important as different needs arise for teachers, mentors and students. This also makes the tool very usable for others that may be looking to download and use this tool in a separate deployment for a currently unknown purpose. Section 3.2.2 describes the design and development of the database in more detail.

2.2.3 Tagging

To allow for more meaningful analysis, we implemented a tagging system in SGIS. There are several benefits of this tagging system. Students are interacting with the data. At the same time, they are, in essence, correcting it. This turns a relatively static set of information into a living, evolving landscape. However, in order to ensure quality and appropriateness, teachers and other site administrators must approve any new tags. Once a tag has been approved once in a particular data set, it is usable for any point in that set.

The tagging system allows the user to mark a point in an existing data set as one thing or another. For example, were using a data set of Retail Food Stores in New York State that includes supermarkets, small grocery stores, convenience stores and bodegas. If a teacher wants students to examine food access in a neighborhood, the differences between these types of stores is enormous. By giving this responsibility to the students, they gain a sense of ownership over the data.
2.2.4 Open-Source

Students working with the 3Helix program have direct access to the developers and can request any changes or additional analysis tools they can imagine. Very advanced students in the 3Helix program or elsewhere can even modify the tool on their own as the project is completely open source and freely available on GitHub. In addition, the very concept of open-source software is, at its core, an altruistic notion. In this way, students see real computer scientists doing programming in a tangible way with the greater good in mind, rather than the interest of corporations. This aspect of altruism attracts girls in particular who tend toward fulfilling careers, like service jobs helping others, rather than money-driven careers [Saraga and Griffiths 1981].
CHAPTER 3
Social GIS Development

Development of the first incarnation of the Social GIS tool began in the summer of 2013. At this time, when I first joined the 3Helix program, Ron Eglash, along with students Michael Lachney and Laquana Cooke, had started to discuss a few activity ideas involving maps, but didn’t have a tool to use with students.

3.1 Trial and Error

In the summer of 2013, my first attempt at creating an online GIS tool was based on the open source project Ushahidi. Because it’s an existing project with a community of users and developers, we had an opportunity to submit SGIS as an extension that another person in the community would be able to use. In addition to being open source, this project also has a strong social justice basis as it was originally created in 2008 to track violence in Kenya following a turbulent election. A Ushahidi deployment allows any user to add an report to a live database via the web or simple SMS text messaging [Ushahidi 2015]. Each report is required to have at least a title, a description, a date and time, a category, and a location [Ushahidi 2014]. Reports are mapped and presented on the front page of the site (in most deployments, although this is easily customized). Most deployments also allow users to filter the visible reports by category or time frame.

Ushahidi was an excellent starting point for us because we were up and running quickly with an online GIS tool, a database to hold locations and a user interface for creating these points. As we set up an off-the-shelf deployment of Ushahidi, we discussed a few possible lessons and use cases we could work towards building into Ushahidi. Food availability immediately came to mind. New York State’s Open Data website provides Retail Food Stores [Open New York 2014] and Farmers Markets [Open New York 2015a] data sets as raw data. I imported these data sets directly into the database to get us started. Figure 3.1 shows Ushahidi with Retail Food Stores visible.

In order to import 2010 census data and access key information about income and race, I needed to also display the census tracts which are polygons that overlay on a map and are associated with demographics information. Census tracts will be described in more detail in Section 4.1.1. The only way to import polygons to Ushahidi was to use KML (Keyhole Markup Language), which is an XML format used to describe map features. Unfortunately, Ushahidi was not designed to accommodate the details nor the number of polygons of the KML file. It was extremely slow and often did not finish loading. Because these polygons were not stored in the same way as the markers, it was very difficult to algorithmically determine if a particular point was inside a particular polygon.

In addition, because Ushahidi was built for another purpose, each point on the map required a place and a time. In our food availability activity, we didn’t need
Figure 3.1: Retail Food Stores marked on the front page of our Ushahidi deployment

a time, just a place. I had put dummy times in place of an actual time, but this was not an ideal solution. In all the activity examples the 3Helix development team brainstormed, having a time associated with each point on the map was simply not important. Because having a time and a place for each point is so central to the original purpose of Ushahidi, many of the features of Ushahidi were, at best, not useful for our purposes and at worst, hindrances. At the end of the summer, I had spent a great deal of time trying to fit a square peg in a round hole, but didn’t have a lot to show for it. Most importantly, the census tract polygons were still unusable which was crucial to examine the demographics of an area.

3.2 Software Design

I revisited the (not yet named) Social GIS project in the summer of 2014. After reviewing the experiences I had the summer before with Ushahidi, the 3Helix development team and I agreed that we would start over. This time I had a better idea of our priorities and what I really wanted to get out of the software. The Ushahidi deployment acted as a prototype that was easy to get up and running so the we could determine our goals. In addition, for the summer, we had a research assistant, Gouravjeet Singh, to help with the development of SGIS. He had experience working with Google Maps and JavaScript. With Gouravjeet’s help, we could develop faster than I could alone.

At this time, the community site for all 3Helix participants to share their work was about a year old and just starting to be used regularly by mentors and students. This site had been created by Charles Hathaway, another 3Helix fellow, in Django, a Python web framework. Because Gouravjeet and I would be starting from scratch and we knew that users of SGIS would need accounts to manage tagging, we decided to develop SGIS as a Django app. Django has an optional extension, called
GeoDjango, that adds geographic database functions. GeoDjango works with several
database management systems, but the greatest number of database functions are
available with PostgreSQL with the PostGIS extender [Foundation 2015b]. Django
is modular so this would allow us to develop SGIS separately from the community
site, but integrate it easily when we wanted the site to go live. Also, developing it
separately means that other Django sites are free to implement and improve upon
our Django app as it is freely available on GitHub.

3.2.1 Client–Server Separation

We also decided to develop the back end and the front end separately to
allow for easier concurrent development. We designated clear duties for both ends.
The back end, the portion written using Django, would, of course, interact with
the database. Most of the heavy lifting is done on this end as well, including any
geographic calculations and account or permissions management. This was the
portion I developed in the summer and fall semesters of 2014. During this time,
Gouravjeet was developing the front end using AngularJS. This front end is shown
in Figure 2.1. To communicate between the front and the back ends, I used a Django
app called Django REST Framework to create a RESTful API. Figure 3.2 shows a
division of client and server duties.

![Diagram 3.2: Client–server division of duties](image)

As shown in Figure 3.2, GET and POST are the only functions allowed. UP-
DATE and DELETE functionality are not included as this type of editing is only
allowed through Django’s built-in admin system. The interactions between client
and server is mostly through GET calls, as the main purpose of the UI is to display
the data. POST is used only to add new tags to points. Theoretically, the back end
could be accessed by a different front end in a different instance of SGIS, but in our
case, we have not set up cross–origin resource sharing to allow any other website to
access the API.

3.2.2 Designing and Developing a Flexible Database

Because we wanted the software to be flexible enough to accept and present
data from many different sources, I spent a lot of time working on the design of
the database where this information is stored. To implement a database design in
Django, one must create classes, called models in Django, that inherit from Django’s Model object. Django then creates database tables in the image of these classes when the application is installed.

Figure 3.3 shows a database schema of the central models I created for SGIS.

![Database Schema](image)

**Figure 3.3: Relational model showing basic SGIS models**

### 3.2.2.1 Dataset

The Dataset model stores meta data and provenance about each data set. This table has a reverse relationship with every other table shown, meaning a Dataset is completely unaware of entities related to it as it has no foreign keys. These data elements relate the SGIS data model back to the original data source. If the original data source has an API, this information can be used to update the data set at a scheduled time (cache_max_age) by accessing the url. All data elements ending in _field store the column name of the given field in the source data. To allow for concatenation or nesting of fields, these fields have been encoded as shown in Figures 3.4 and 3.5. reach_field(), shown in Figure 3.6, interprets the coded data. Once the MapElements of a given Dataset have been loaded or updated, each point is geocoded if necessary. This is done because many data sets have simply been geocoded to a latitude and longitude based on zip code. This is not accurate enough for our purposes. The geocoding is done against a Google Maps API until the daily quota is met. If there are still MapElements to be coded, this is done the next time the job is run, without the entire data set being re-cached.

### 3.2.2.2 MapElement

Many MapElements are associated with a single Dataset. In Django/Python terms, MapElement is a base class for all other elements that may appear on a map. In the database, this means that MapElement has a one-to-one relationship with any child model. MapElement may be used alone, but a need for this implementation has not yet arisen. The following models are children of MapElement:
"street_name" : "CHURCH ST #242",
"address_line_2" : "",
"street_number" : "240",
"location" : {
"needs_recoding" : false,
"longitude" : "-73.75540780899968",
"latitude" : "42.63540995700049",
"human_address" : "{"address":"240 CHURCH ST
#242","city":"ALBANY","state":"NY","zip":"12202"}"
}

Figure 3.4: Excerpt from JSON object in Retail Food Stores [Open New York 2014] data set

street_field = "street_number+street_name+address_line_2"
lon_field = "location,longitude"
lat_field = "location,latitude"

Figure 3.5: street_field, lat_field and lon_field values from JSON object shown in Figure 3.4

def reach_field(self, json_item, location):
    result = ''
    if len(location) > 1:
        for field in location[0]:
            if field in json_item:
                result += self.reach_field(json_item[field],location[1:]) + ' ' 
    elif len(location) == 1:
        for field in location[0]:
            if field in json_item:
                result += json_item[field].strip() + ' ' 
    return result.strip()

Figure 3.6: reach_field function to parse a JSON element based on a coded field location

A MapPoint is any single point on a map. All of the MapElements in the Retail Food Stores data set, for example, are MapPoints.

A MapPolygon is any polygon or group of polygons (multipolygon) on a map. All of the MapElements in the 2010 Census Tracts data set, for example, are MapPolygons.

There have been discussions about adding an additional model to accommodate data from buoyed sensors that another 3Helix fellow, James Davis, is creating. However,
this has not yet been implemented. If a need arises for more data fields in the future, any of these models could be extended to create a new model appropriate for that use.

### 3.2.2.3 Tag

The Tag model relates a Dataset with the text of a tag. In SGIS, a tag is always lower case, free of leading or trailing spaces and does not contain a comma. Anyone may submit a new tag and the API will accept it and create a new record, but it requires an administrator to approve this. Until approved, this tag will not appear anywhere in a response to a call to the API. Once approved, however, the tag will be available to all to tag individual points for that particular Dataset.

### 3.2.2.4 TagIndiv

The TagIndiv model relates an individual MapElement to a Tag in a many-to-many relationship. However, each TagIndiv must contain a unique combination of MapElement and Tag. In addition, because the API actually hides this model from users and it is created using the dataset_id, there is no way that the dataset_id of the paired MapElement and Tag could contradict.

### 3.2.3 Software Architecture

In Django’s documentation [Foundation 2015a], the framework is reluctantly labeled as a “model-template-view framework” in lieu of a model-view-controller (MVC) architecture. Furthermore, because I am using Django REST Framework, templates are not required. Serializers format the external JSON interfaces. Again, comparing this to an MVC framework, models in SGIS act as models in any MVC framework, storing information; serializers act almost as controllers would, editing and updating models; and views determine which information is shown. Django takes cares of the database calls and manages specific calls depending on which database is used. However, SGIS is only going to work accurately with PostGIS as it uses many GeoDjango functions that are only available on a PostGIS database.

With the direct database calls taken care of by Django and the rendering of the REST API managed by Django REST Framework, the bulk of the software is in the models, as described in Section 3.2.2, the views and the serializers. The urls module simply points a given URL format to a given view.

#### 3.2.3.1 views Module

In the simplest cases, a view simply defines which results should be shown in the REST API and which serializer will control the input and output for that view. An example is shown in Figure 3.7. Given this code, Django REST Framework will return a single Dataset object if requested by id in a url formatted as `<base dataset URL>/<id number>`, as is customary for REST frameworks.

For more complex situations, the filter_tools and geometry_tools modules contain helper functions used by the views module. For example, Figure 3.8 shows
class DatasetViewSet(PaginatedReadOnlyModelViewSet):
    queryset = Dataset.objects.all()
    serializer_class = DatasetSerializer

Figure 3.7: A simple view

a view using filter_request() of filter_tools. filter_request() accepts query

class MapPointViewSet(PaginatedReadOnlyModelViewSet):
    serializer_class = MapPointSerializer
    model = MapPoint
    queryset = filter_request(self.request.QUERY_PARAMS, 'mappoint')

Figure 3.8: A filtered view

parameters and filters the queryset of any MapElement type based on those parameters. Parameters in this function include data elements like city, state, zip code, and dataset id. Geometry–based filters are also used here. In particular, parameters defining a bounding box are key in ensuring that only the required geometries are returned. Because these parameters are mostly the same across several views, it only makes sense to have this work done by one function. This also means that changes to these parameters are uniformly implemented across all views.

3.2.3.2 serializers Module

As in Figure 3.7, DatasetSerializer, shown in Figure 3.9, is very simple. Because this inherits from the Django REST Framework's ModelSerializer, only fields in the JSON that are not part of the model are defined here. Fields that are part of the model can also be renamed or reformatted in other ways. In this example, the tags have a many-to-one relationship with the dataset, so the get_tags() function creates a nested list of all approved tags related to the dataset.

Essentially, for a read-only GET request, a view first filters what information is to be shown in the outgoing JSON, then the serializer arranges the information and adds outside information from other models. In the case of a POST request, the serializer does a little bit more work, as is the case in Figure 3.10.

As described in Sections 3.2.2.3 and 3.2.2.4, the tag information is separated over two models. Therefore, to POST this information and to keep it simple for the front end, the information is accessed through the API as if the model were simply one that contained only the text of the tag and the MapElement it belongs to. restore_object() does the work to rearrange this information into two models.
class DatasetSerializer(serializers.ModelSerializer):
    tags = serializers.SerializerMethodField(get_function('get_tags'))
    count = serializers.SerializerMethodField(get_function('get_count'))

    def get_tags(self, dataset):
        #build nested distinct list
        return Tag.objects.filter(approved=True,
                                  dataset=dataset).order_by('-count').values('id','tag')

    def get_count(self, dataset):
        #build nested distinct list
        return MapElement.objects.filter(dataset=dataset).count()

class Meta:
    model = Dataset
    fields =
        ('id','name','cached','count','field1_en','field2_en','field3_en',
         'tags')

Figure 3.9: A simple serializer
class NewTagSerializer(serializers.ModelSerializer):
    mappoint = serializers.IntegerField(source='mapelement_id')
    tag = serializers.CharField()

class Meta:
    model = TagIndiv
    fields = ('mappoint', 'tag')

def restore_object(self, attrs, instance=None):
    #find the mappoint
    try:
        mp = MapElement.objects.get(id=attrs['mapelement_id'])
    except:
        raise exceptions.ParseError()

    #don't except lists in the tag parameter or tags with commas in it
    if ',' in attrs['tag']:
        return None

    #tags have no leading or trailing spaces and are always lower case in this implementation
    attrs['tag'] = attrs['tag'].strip().lower()

    #find if this tag is already being used with the dataset
    tags = Tag.objects.filter(dataset=mp.dataset, tag=attrs['tag'])

    len_tags = len(tags)
    if len_tags == 0:
        tag = Tag(dataset=mp.dataset, tag=attrs['tag'])
        tag.save()
    elif len_tags == 1:
        tag = tags[0]
    else:
        approved_tags = tags.filter(approved=True)
        if len(approved_tags) > 0:
            tag = approved_tags[0]
        else:
            tag = tags[0]

    return TagIndiv(mapelement=mp, tag=tag)

Figure 3.10: A more complex serializer
CHAPTER 4
Use of Social GIS: Activity Outlines

4.1 Environmental Justice

Early in the fall semester, I met with Professor Michael Mascarenhas of the Department of Science and Technology Studies. He had heard about this project at the Generative Justice Conference over the summer and was interested in using the tool with the undergraduate students in his Environmental Justice class. The students were given an assignment to, as a class, create an updated report based on the report by Bullard et al. [2007], “Toxic Wastes and Race at Twenty 1987–2007: A Report Prepared for the United Church of Christ Justice & Witness Ministries”. In particular, the tool would be used to update the third chapter of the report that analyzes the demographics of the people living close to hazardous waste sites. Because this report looks at this data up until 2007, Professor Mascarenhas wanted the students to update the analysis using demographics from the 2010 census.

The report by Bullard et al. [2007] analyzes the population in four groups: those living within one, three and five kilometers of the hazardous waste sites and those living beyond five kilometers. The report essentially showed with the data in Table 4.1 that the demographics closer to the site were poorer and less white.

Table 4.1: Selected characteristics of those living near hazardous waste sites

<table>
<thead>
<tr>
<th></th>
<th>&lt;1 km</th>
<th>1-3 km</th>
<th>3-5 km</th>
<th>&gt;5 km</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Population</td>
<td>845</td>
<td>7,828</td>
<td>14,101</td>
<td>225,936</td>
</tr>
<tr>
<td>(1000s)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Population Density</td>
<td>690</td>
<td>840</td>
<td>810</td>
<td>24</td>
</tr>
<tr>
<td>(persons per square kilometer)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Percent People of Color</td>
<td>47.7%</td>
<td>46.1%</td>
<td>35.7%</td>
<td>22.2%</td>
</tr>
<tr>
<td>Poverty Rate</td>
<td>20.1%</td>
<td>18.3%</td>
<td>16.9%</td>
<td>12.7%</td>
</tr>
</tbody>
</table>

4.1.1 Method of Analysis

For this analysis, the report by Bullard et al. was a departure from many previous reports looking at the same type of information. First, it is important to note that census information is only available in aggregate form. For most data points in the census, the most granular aggregate data is available for a census tract. Census tracts are defined by state, county and other municipal lines, as well as being roughly equal in population. Because of this, census tracts vary significantly in land area [United States Census Bureau 2012]. Some reports have simply found the census tract that the hazardous waste site is in and then used the demographics of that tract for analysis. This, as Bullard et al. [2007] show in Figure 4.1, misrepresents
the actual demographics of who lives nearby. There are uncounted people living very close to both hazardous waste sites and, in the case of the larger tract, a large number of people counted who do not even live close by.

Figure 4.1: Example host tracts

Citing Mohai and Saha, Bullard et al. [2007] describe the method of “50% areal apportionment” to more accurately determine the demographics of those living closest to the hazardous waste sites. To analyze the area around a point, sum up the raw aggregate data of each census tract for which 50% of the area is within a given distance of the point, creating an approximation that looks similar to a Riemann’s sum. This method, as Bullard et al. show in Figure 4.2, gives a more realistic estimate of the population.

Figure 4.2: Example using 50% areal apportionment


4.1.2 Extension of Social GIS

After Professor Mascarenhas approached me about this assignment, I discussed with other members of the 3Helix developer’s team how this kind of functionality would fit into the scope of the project. We quickly agreed that this method would be useful for almost any of the activities we had imagined up to this point. Students could analyze the area surrounding their home or school more accurately. Once decided, I needed to add functionality for this type of analysis to the back end. Very little development was required on the front end because the calculations are left to the back end. Retrieving this information and displaying circles on a map are both relatively simple.

The crux of this task was to identify which census tracts are at least fifty percent inside a circle about a given point. These census tracts could be any polygons; for example, there is childhood obesity data available for New York State by school district, which are a completely independent set of polygons [Open New York 2015b]. However, for clarity, I am going to refer to these polygons as census tracts. Initially, I intended to tackle this problem using PostGIS functions. Any database functions are going to run faster than any algorithm I could come up with. Unfortunately, though there are many database functions to determine geometric collisions and containment, there are no functions out of the box that can determine polygons with fifty percent areal containment within another polygon, let alone a circle. Without a magic database function to solve this task for me, I broke the problem into smaller steps.

4.1.2.1 Constructing the Radius

Because I could not use any database functions to determine whether at least fifty percent of a given census tract was within a given distance of a given point, but I could use the database functions to define the polygon that makes up the intersection of two polygons and to determine the area of a polygon.

First, I need to determine a polygon that represents the circle around the point. This is complicated for two reasons: with just latitude and longitude, creating a circle on a sphere is tricky and if the polygon has too many sides, calculations will be overly complex. Ignoring the complications caused by the world not being flat, the polygon would be oblong at the equator or at the poles, depending on how I calibrated the calculation. Luckily, there is a Python library called Geopy that includes functions for calculating geometric distances. Using the destination function of the Vincenty distance object, Geopy can calculate the latitude and longitude of a new point given a starting point, a bearing and a distance [geopy/geopy - GitHub 2014]. This distance uses the Vincenty algorithm for determining an equal distance from the latitude and longitude. Using this function, I create the radius polygon around a point with twelve sides, as shown in Figure 4.3.

The return value of this function is a Polygon object that can be used within GeoDjango to query against the PostGIS database. An example polygon approximation of a circle is shown in Figure 4.4.
def circle_as_polygon(lat, lon, n = 12, distance = Distance(km=1)):
    #this can be done all in degrees because thats what geopy uses!
    points = []
    #build points, loop around circle
    for angle in [360.*i/n for i in range(0,n)]:
        curr = VincentyDistance(distance).destination(gpPoint(lat, lon),
        angle)
        points.append((curr.longitude, curr.latitude))
    #loop back to the first point for the n+1th point in the list
    points.append(points[0])
    return Polygon(tuple(points))

Figure 4.3: circle_as_polygon function to approximate a polygon around a point

Figure 4.4: Example polygon approximation of 1 mile radius

4.1.2.2 Finding Neighboring Points

If we’re looking at a point that exists within a data set, like the hazardous waste sites, neighboring points must be accounted for in order to not double count populations that may live within a certain distance of multiple sites. In the general case, this is not needed for ad hoc points, like a student’s home or school, but is needed to accurately count populations near any set of points. The report by Bullard et al. grouped nearby points together, as shown in Figure 4.5 [Bullard et al. 2007].

To achieve this result, I find the largest distance to analyze. In the case of the analysis by Bullard et al., this would be five kilometers. Then, I use the GeoDjango’s distance_lte filter, which calls PostGIS’s ST_Distance() function, to determine if
any other points are within twice the radius of both the original point and any other points found to be nearby the original point. In this way, the algorithm is potentially very computationally expensive if many points are clustered close together. Fortunately, in the case of the hazardous waste sites, clusters were never more than ten sites using a maximum radius of five kilometers. Because other uses may push the server to its limits, I set a hard maximum as to the number of sites per cluster. If the request is for the entire data set at once, clustering is completely bypassed. All points are run as if they are one large cluster.

The polygons for each distance level are created for all the points in a cluster. I create a multipolygon from the union of the polygons for each distance. Figure 4.6 shows a cluster of hazardous waste sites, as entered by the students of Professor Mascarenhas’ class, in Los Angeles and the one, three and five kilometer boundary multipolygons. As will be described in Section 4.1.2.3, the fifty percent areal appor- tionment is applied using these joined polygons. This means that, for example, an intersecting census tract that passes through the one kilometer boundary polygons for two hazardous waste sites and has at least fifty percent of its area between the two boundaries is in the one kilometer cohort. This is true, even if it would not be included if either of the points were alone.

4.1.2.3 Filtering and Sorting Census Tracts

With a set of boundary polygons, the next step is to determine which census tracts are at least fifty percent contained within each distance level. As the census tract set includes all census tracts in the United States, the group must be culled very quickly before any significant computation can be done. Any filters that were
Figure 4.6: Cluster of hazardous waste sites in Los Angeles with polygon boundaries at 1 km, 3 km and 5 km

passed as an argument are applied first, though the actual database call is done all at once, when the individual census tracts are accessed. I do this using a combination of GeoDjango’s \texttt{intersects} filter and \texttt{covers} filter which are both database functions. Starting with the smallest radius boundary, census tracts are sorted into two lists: those completely inside the boundary and those intersecting with the boundary, but not completely inside the boundary. Unfortunately, sorting through the second list is not a simple task.

The second list, our potential census tracts, must be tested for fifty percent areal intersection with the boundary. One by one, each census tract is intersected with the boundary polygon. The area of this resulting polygon is compared to the area if the original census tract polygon. Those with at least fifty percent of the area are added to the list of census tracts within the boundary. To reduce redundancy, as the list of boundary polygons are traversed, from smallest to largest, those tracts that have already been counted in smaller boundaries are excluded off the bat. This means in the very unlikely case where exactly fifty percent of the tract is within one boundary and the other fifty percent is within another boundary, this tract is counted in the smaller boundary only. Once the census tracts for a particular boundary level have been identified, the demographic information is summed. After all census tracts have been analyzed, the demographic information can be organized as in Table 4.1.
4.1.3 Going Forward

Professor Mascarenhas’ undergraduate students were able to overlay their hazardous waste site data with 2010 census demographics information to analyze it in the same way as the report produced by Bullard et al. [2007]. While high school or middle school students may have trouble completing an assignment as in depth as the Environmental Justice students, they are capable of interpreting the data, similar to that of Table 4.1, that SGIS can now output. This data lends itself well to discussions about social justice and the rights of the people living closest to these sites. It is also information that can be graphed in many different ways or even be presented in a more creative format, like an infographic, as described below in Section 4.2.

4.2 Create an Infographic
4.2.1 Explanation of Infographics

Infographics, as the name suggests, are a popular way to represent information graphically, often using a lot of color and creative representations of data. They are used all over the Internet to explain complex concepts, promote products and publicize social injustices and are used in news articles, blogs and social media campaigns. In this activity, students are asked to look at some existing infographics to get a feel for what an infographic is and to determine what are good or bad elements in an infographic. Figure 4.7 shows an infographic I created to give students some idea of the types of infographics they might find on the Internet.

![Image](image.png)

Figure 4.7: “How to Prepare for a Day Hike” infographic

While it may be informative and helpful for a particular audience, this infographic can be used to clarify what is not expected of the student’s work in this activity. In particular, where did the information come from? Are any of these tips proven science? Does this infographic actually contain any data? For our purposes, we are looking to present data. This graphic is, however, a good example of using...
an illustration to help make your point and create an easily “sharable” Internet artifact. The discussion might also touch on the audience of the image and how the design and colors used in the image are playful and cartoon like. For which audiences would this be appropriate?

Figure 4.8, as Fanning [2012] published on his personal portfolio, shows another example of an infographic. Initially, most students would see the contrast of this image against the previous example. This image uses attention grabbing colors, represents actual data and cites a source in the image as well as an explanation of the data.

Figure 4.8: “Top 10 Most Read Books in the World” infographic

The students identifying what is wrong with this infographic may be more difficult as these details are more nuanced. The first thing the students may notice is that the colors are too bright, as a projector washes out the image. This leads to a discussion of context; if the image was meant for websites and social media where individual users would see it, is the appear on a projector important? Next, the students may or may not notice that the scale of the bar graph does not match the numbers shown. A close up of the bar graph is shown in Figure 4.9. Looking at the details of the graph, the scale is true, but excludes the body of the books below the zero line. In addition, there are many details to signal this to the reader, but are they made visible enough? This image could also spark a discussion about the quality of the data, as it did when I presented this to the 3Helix teachers. The Bible wins out by a huge margin, but is it fair to assume this many Bibles have been read just because they were purchased? Bibles are purchased by many groups to hand out and place in hotel rooms. Even home Bibles sometimes sit unread. Conversations like this encourage students to think about bias in data that is interpreted for them or presented to them in media.
4.2.2 Example Assignment

With these two examples in mind and after receiving instructions on using SGIS, students can begin creating their own infographics. Depending on the intent of the lesson, students may be directed to research a specific topic or a specific neighborhood, but the tool itself, as described in Chapter 2, is intended to be open ended. One example of a directed activity could be as simple as asking students to examine the types of food available near their home. Figure 4.10 shows an infographic I created as an example for the students.

Figure 4.10: Infographic assignment example

To create this, I took advantage of the tagging feature. I made sure that all food stores and food service locations within a mile of this fictional student’s home, which is actually located at Albany High School, were tagged properly, especially...
grocery stores, pizza shops and fast food restaurant chains. Then I showed the 1 mile radius around that point and counted the points that matched each tag filter. Depending on the level of the students or the amount of time that can be devoted to this activity, the assignment could be more complex, or even as complex as the activity described in Section 4.1.


5.1 Reflection

Along the way, I faced many challenges. First, I struggled with our initial attempt, as I described in Section 3.1, for a few reasons. Ushahidi is written using the Kohana framework, which is in PHP and is an HMVC framework. I had never worked on a large project written in only PHP before, so there was a steep learning curve. Again, as I said in Section 3.1, Ushahidi was created for a very different purpose than we tried to use it. However, I believe that Ushahidi gave us a prototype with which we could experiment and determine what we wanted out of our software.

I also grappled with the challenges that come with concurrently, but separately developing the front end and the back end with another student. Progressively adding the next feature meant that both ends worked towards that goal at the same time. The front end in particular had no back end to test against. This meant it was very important for me to explain how I expected the back end interface to work and stick to it. Gouravjeet, the student working on the front end, would then interact with dummy data based on these requirements. In most cases, this worked, but it was difficult to test POST features in this way without a working API. Also, very often, the features were easier to implement in either the front or the back end, so that person would move on to the next task ahead of the other. In part because Gouravjeet was helping only as a three credit independent study during the fall semester and could devote less time to the project, the back end was feature rich and the front end was lagging behind toward the end of the semester. Even so, at the end of the fall semester, we managed to have a working front and back end that could display points in a data set, post new tags and filter displayed points by tags. This gave us a starting point for working with students.

Because we didn’t have a working product until the end of the fall semester and we still don’t have a live deployment, I did not have time to try the software out with high school students, the intended audience. I did, however, work with the Environmental Justice class as discussed in Section 4.1. The class had a very specific need that happened to fit with our overall goals for the software. This feature was added into the back end, but unfortunately hasn’t made it into the main front end yet. I was able to work with the students and give them the data they needed, but with the software running only on a test server and with a rudimentary front end interface, they could not interact directly with the software.

5.2 Going Forward

Social GIS is an ongoing project that has a lot of potential. As I explained in Section 1.5, SGIS is very different from existing tools created and used in the 3Helix program. It addresses data head-on, unlike the other tools, most of which address
graphing skills and other visual mathematical concepts. Data has become such a huge part of our lives and it is important to empower students to become analytical consumers of data as well as active investigators.

Developing SGIS has been a great experience, but it’s far from done. SGIS was created with expansion in mind and, as in Section 4.1, it has already begun to evolve. There are three things on the immediate horizon for me and SGIS. First, very soon, I intend to integrate the analysis used with the Environmental Justice class into the main front end so that all can use it. Second, by the end of this semester, I will get SGIS into a classroom where it can be tested with a full class of students over a few class periods. My teacher pair in the 3Helix program at Albany High School, Mike Justice, has an AP Stats class that will have extra class time at the end of the year. After the AP exam in early May, any classroom activities are not going to interrupt the curriculum. This offers a certain amount of free time and flexibility to work on a larger project with the students. Second, as I mentioned briefly in Section 3.2.2.2, I have been working with James Davis on finding a data storage solution for his buoyed water sensors. He is developing a device that floats in the river and periodically takes measurements. This device will be solar powered and partially self-sustaining. The only required human intervention will be to pick up batches of data from the devices wirelessly from a smartphone. Then, via the smartphone, upload the data to SGIS. This would open up new, interdisciplinary learning opportunities. We have already laid out a plan to extend MapElement to accommodate this information, but while James works on engineering the sensor, I am working implementing the planned database schema and the corresponding REST API calls.
References


APPENDIX A
Project Code

Back end  https://github.com/GK-12/SGIS-backend/
Front end  https://github.com/GK-12/SGIS-frontend/