“Adding Qualitative Elements to Visualizing Gentrification in Atlanta Neighborhoods”
Design Statement
Project 3 – Media in the Social and Physical World
LMC 6310
Nick Tippens
12/2/2016

The overall design goal of this project was to expand upon my previous work and add a qualitative element to the quantitative visualization of gentrification in Atlanta’s in-town neighborhoods that I developed in Project 2. In Project 2, I built a location-aware augmented reality mobile application. I chose to build this application on the Argon augmented reality web browsing platform, developed by Georgia Tech Augmented Environments as the world’s first AR web browser based on open web standards for iOS technology. In project 3, I decided to use the Twitter API to incorporate tweets into this application by grabbing their text based on keyword search queries and then displaying the text within the interface of my application.

My design process began with an investigation into how I could pull real world tweets from Twitter. I decided to use the Twitter API to build keyword-based search queries that grab batches of tweets and their data through server-side requests coded in PHP. These requests grab a specified number of tweets and return JSON-encoded associative arrays of their data. This process required me to create a Twitter application, giving me access to Twitter’s data with my own unique authorization keys. After inputting my keys, the call for the aforementioned keyword-based search consisted of the following PHP code:

```
$url = 'https://api.twitter.com/1.1/search/tweets.json';
$requestMethod = 'GET';
$getfield = '?q=atlanta%20gentrification%20-RT%20-#GoodieMarketing%20filter%3Asafe&count=50';

        $twitter = new TwitterAPIExchange($settings);

        $jsonData = $twitter->setGetfield($getfield)
          ->buildOauth($url, $requestMethod)
          ->performRequest();

         $string = json_decode($jsonData, true);
```

I decided to base the Twitter components of my application on two basic types of searches. The first search grabs general tweets regarding gentrification and Atlanta based on a keyword search of, coincidentally, “Atlanta” and “gentrification.” This is the search that can be seen in the code example above. Upon loading the application in Argon, the user is presented with Tweets based on this search. The resulting tweets provide the user with a qualitative overview of what Twitter users are saying about the process and consequences of gentrification in Atlanta. As a search it yields a wide range of tweets, ranging from those of casual citizens expressing their appreciation of or frustration with the realities of gentrification in the city, to those of more professional organizations presenting data or publishing articles about the

process and its effects on housing, urban development, and racial/social issues in the city. It provides for the user a very nice qualitative sample of the reality of gentrification in Atlanta.

The second search is actually a set of searches designed to provide the user with qualitative insight into the cultural "flavor" of specific neighborhoods addressed by the application, beyond simply their relationship to the concept or process of gentrification. Therefore, each of the 26 neighborhoods featured in the application is represented by its own search of keywords unique to the neighborhood, typically the neighborhood's name and "Atlanta" if necessary to differentiate it from neighborhoods of the same name in other places. Examples of these searches include "Midtown" and "Atlanta," "West" "End" and "Atlanta," and "Reynoldstown," for the neighborhoods of Midtown, West End, and Reynoldstown respectively. Determining the most effective wording of these searches took a bit of refinement, as there are more tweets available about some neighborhoods than others. In fact, one of the limiting factors of this application is simply the number of tweets available about a given neighborhood. Research into the functionality of the Twitter API revealed that keyword search queries grab only tweets published within the most recent 7 days. Still, I found my queries to provide satisfactory results for virtually every neighborhood. I wrote each query to grab 50 tweets, and while most neighborhoods do not seem to produce quite this many tweets each 7 days, there are typically at least 8-10 interesting tweets available for any given neighborhood at any given time. This is a large enough selection of tweets to give the user a sense of the culture of the place, accomplishing my design goal.

Another design challenge was writing search queries in a way that efficiently retrieved enough relevant tweets without producing a set of repetitions of the same tweet or an overwhelming batch of promotional tweets from a single organization sent out over a short time span. To solve the first problem, I eliminated retweets from my queries by including "-RT" in each search line. To solve the second problem, I eliminated certain hashtags from my queries by including, for example, "-#GoodieMarketing" to prevent an overwhelming collection of tweets by a single organization from dominating my results. In the case of my primary "Atlanta gentrification" search, my results were flooded with promotional tweets by an innovation organization called Goodie Nation, and by simply removing the hashtag above from the search line, I successfully removed nearly all of the organization's tweets from my results. As a precautionary measure, I also placed a safe filter, "filter safe," on each query to prevent profane and/or inappropriate language from appearing in my application.

Based on the structure of the PHP code in my application, all of the Twitter search queries are performed upon launch of the application. The resulting batches of tweets and their data are stored, as stated above, in JSON encoded PHP associative arrays. I decided to integrate the tweets into the head-up display (HUD) of the Argon application because I wanted it to provide the user with a consistent stream of qualitative information wherever the user is located rather than requiring the user to reach a certain physical location to encounter this aspect of the experience. To accomplish this, I created a hidden div outside of the <ar-scene> in my primary Argon-A-Frame HTML file (which became a PHP file once I sourced the Twitter API code) and placed a css-object, tied to the hidden div by its ID, within the <ar-camera> entity,

similar to the way in which I placed the key image in the scene for Project 2. Placing the css-object to which I would send the Twitter text within the camera entity ensured that it would stay in place regardless of where the user points his or her phone, truly causing it to act as a HUD element. With the hidden div in place, I was able to modify the innerHTML of the div with the text of each tweet from a given search. However, in order to send the text from real world tweets into the div, I had to have access to that text, as well as the ability to cycle through the arrays of different text from different tweets.

I wanted to display the tweets in a way that temporally allowed the user to operate the app and experience the visual, quantitative data visualization without distraction while the qualitative information contained within the tweets occasionally updated itself automatically and unobtrusively. I quickly discovered through PHP documentation that this would be very difficult to accomplish with PHP code, as timed intervals are not a functionality native to the language. To solve this problem, I successfully converted the encoded JSON PHP associative arrays of my Twitter search result data into JavaScript associative arrays with the same indexes using the following line of code, repeated for each neighborhood's array:

```
var tweetsArray = <?php echo json_encode($string, JSON_PRETTY_PRINT) ?>;
```

This code proved to be crucial to the expansion of my application in terms of it functioning the way that I had envisioned and cycling through a selection of tweets by displaying a new tweet in the same location every 10 seconds. With the JSON data encoded into a JavaScript array, I was able to write cycling functions for both the tweet results of the intial "Atlanta" and "gentrification" search as well as for the searches of all the specific neighborhoods.

As can be seen below, the general "Atlanta" and "gentrification" search's resulting tweets begin cycling through on the HUD display of the application as soon as it loads. As you can also see, I access the text of the tweets within the JavaScript associative JSON array by iterating through its index at the third level down, the first level being "Statuses," the second level being each tweet's index number "i," and the third level being the text of the tweet selected by the given index number. By replacing the innerHTML of the aforementioned hidden div with the text of the next tweet in the array every 10 seconds, as defined by the timing interval at the bottom of the code above, I am able to successfully display to the user the series of tweets grabbed by the Twitter search.

```
window.onload = function(){

 var mainText = document.getElementById("tweet");
 var i = 0;

 function nextTweet() {

  mainText.innerHTML = tweetsArray["statuses"][i]["text"];
  i++ % tweetsArray.length;
```

```
 }

 nextTweet();

 currentIntervalId= window.setInterval(nextTweet, 10000);

}
```

To update the cycle of tweets to reflect the results of a search about a specific neighborhood, and stop displaying the general "Atlanta" and "gentrification" search results, I combined the fuse-click functionality already built into my application from Project 2 with another cycling function that also resets the timing interval mentioned above. From a design standpoint, I wanted this transition to take place when the user fuse-clicked on a particular neighborhood's 3D tower (box). I had already created a variable called boxNumber that was populated with the ID of the given neighborhood's box when the box was fuse-clicked upon. In Project 2, this variable made it possible to trigger the text of a neighborhood's tower to animate when the tower was clicked upon. I used this same boxNumber variable to trigger the cycle of tweets to update and begin displaying tweets from the Twitter search about a given neighborhood based on its box being fuse-clicked. As can be seen below, the neighborhoodTweets array is an array of the JSON arrays of tweet data from the various neighborhood searches. I access a particular neighborhood's array with the boxNumber variable as the index of the larger neighborhoodTweets array, and once inside of it, access and cycle through the text of each tweet and ultimately replace the innerHTML of the hidden div with that text in the same way as before. At the bottom of the function, the timing interval is reset, the HUD element stops cycling through tweets from whatever search it had been cycling through, and begins cycling through the tweets from the search related to the neighborhood that has had its box fuse-clicked upon.

```
function changeTweetStream(){

 var mainText = document.getElementById("tweet");
 var k = 0;

 function nextTweet() {

  mainText.innerHTML = neighborhoodTweets[boxNumber]["statuses"][k]["text"];
  k++ % neighborhoodTweets[boxNumber].length;

 }

 nextTweet();

 clearInterval(currentIntervalId);
 currentIntervalId= window.setInterval(nextTweet, 10000);

}
```

All of this is triggered by a second cursor-listener component that I created, registered as an A-Frame component, and added to each box element as part of updating Project 2 for Project 3.

```
AFRAME.registerComponent('cursor-listener2', {
    init: function () {

     this.el.addEventListener('click', changeTweetStream);

    }
});
```

Ultimately, I believe that I accomplished my design goal of adding a qualitative element to my quantitative visualization of gentrification in Atlanta neighborhoods. I find the final product to be engaging, particularly in its portrayal of the cultural "flavor" of the different neighborhoods that the user encounters. The application is a truly interesting environment in which to experience recent, unbiased, real world tweets about different Atlanta neighborhoods while also engaging with the visualization of those neighborhoods' respective degrees of gentrification experienced the last 15 years. That, along with the exposure to the general attitudes toward gentrification in Atlanta on Twitter that the user experiences when he or she loads the application, combine to create a uniquely interesting and informative augmented reality experience.

Instructions for loading the application:

1. Download the Argon 4 application from the App Store on an iOS device.
2. Copy and paste the following link into the browser: http://6310.lmc.gatech.edu/~ntippens3/ATL%20Gentrification%20Twitter%20Update/ATLgentrificationTwittAR.php