

Homework 3

Team member: Fan Zhang, Zhiqi Chen

Textbook Problems

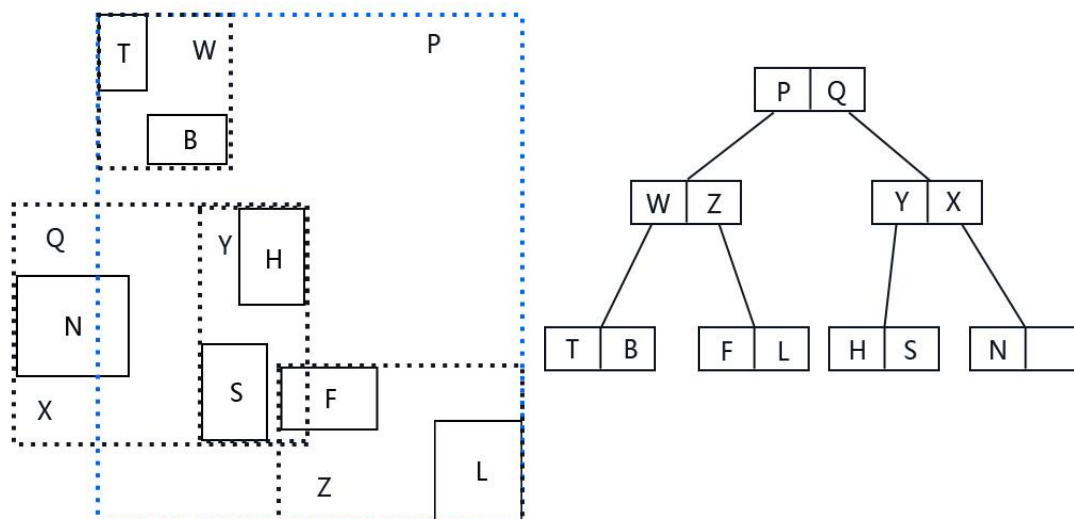
Q 5.2 :

1. Douglas-Pucker algorithm to discretize arcs: $n \cdot \log(n)$
2. Compute area of a simple polygon: linear
3. Compute centroid of a polygon: linear
4. Point in polygon: linear
5. Intersection of a polygon-pair, each with N vertices : polynomial
6. Depth-first or breadth-first graph traversal : polynomial
7. Single-pair shortest path in a graph : polynomial
8. All-pair shortest paths in a graph: polynomial
9. Hamiltonian circuit (or Traveling salesman problem): exponential

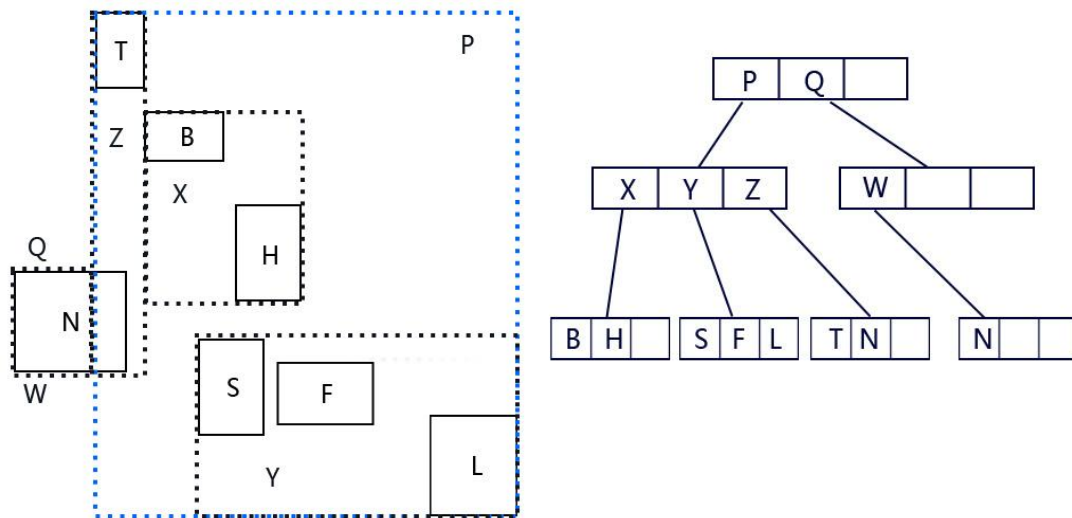
Q 5.4 :

1. NAA is the closest to the GML Simple Features. Because they are both composed by the points and directed arcs. Also, each area is bounded by one or more directed arcs.
2. Spagetti is the closest to ESRI shapefile, since both of them store primitive geometrical data types of points, lines, and polygons.

Q 6.2



For the R+ tree, we find it's very tricky to draw the R+ tree since the R+ tree don't allow overlapping between any non-leaf nodes. The orthogonal bounding rectangle for rectangle T and L just cross the rectangle N. So the minimum bounding box for N become the most tricky part for this problem. In order to keep the tree as height balance, we choose a fan-out ratio of three.



Q6.4

1. File
2. File
3. File
4. R-tree/PM quadtrees
5. R-tree/PM quadtrees
6. Adjacency list
7. Adjacency list
8. Adjacency list
9. Adjacency list

Lab3

PART B:

```
CREATE INDEX roads_idx ON roads_major(geom) INDEXTYPE IS MDSYS.SPATIAL_INDEX;
```

```
CREATE INDEX natural_idx ON natural(geom) INDEXTYPE IS MDSYS.SPATIAL_INDEX;
```

1:

```
select a2.id as A2ID, a1.id as A1ID from activities a1, activities a2 where a2.id = 1 and
mdsys.SDO_NN(a1.geom, a2.geom, 'sdo_num_res=3') = 'TRUE';
```

2:

```
select act.id, road.osm_id from activities act, roads_major road where mdsys.SDO_NN(act.geom,
road.geom, 'sdo_num_res=1') = 'TRUE';
```

3:

```
select road.osm_id, road.name, natural.osm_id from roads_major road, natural where
natural.type = 'water' and mdsys.SDO_OVERLAPS(road.geom, natural.geom) = 'TRUE';
```

4:

```
select n1.osm_id, n2.osm_id from natural n1, natural n2 where n1.type='riverbank' and
n2.type='riverbank' and mdsys.SDO_OVERLAPS(n1.geom, n2.geom) = 'TRUE'and n1.osm_id<
n2.osm_id;
```

5:

```
select n1.osm_id, n1.type, n2.osm_id, n2.type from natural n1, natural n2 where
```

```
n1.type='riverbank' and n2.type='forest' and mdsys.SDO_TOUCH(n1.geom, n2.geom) = 'TRUE';
```

PART C: Impact on Query Response Time

set timing on;

1:

```
select distinct n1.osm_id, n1.type, n2.osm_id, n2.type from natural n1, natural n2,  
user_sdo_geom_metadata m where n1.type='park' and n2.type='park' and n1.osm_id<n2.osm_id  
and mdsys.SDO_GEOM.WITHIN_DISTANCE(n1.geom,m.diminfo,1,n2.geom,m.diminfo) = 'TRUE';
```

Elapsed: 00:00:10.06

2:

```
select distinct n1.osm_id, n1.type, n2.osm_id, n2.type from natural n1, natural n2 where  
n1.type='park' and n2.type='park' and n1.osm_id<n2.osm_id and  
mdsys.SDO_WITHIN_DISTANCE(n1.geom,n2.geom,'distance = 1') = 'TRUE';
```

Elapsed: 00:00:05.13

3:

Q2 has better performance than Q1. It's almost twice faster than Q1. The reason is Q2 use spatial operators, which provide optimum performance since they use the spatial index.

Trend 3

Summary of Chapter 6 Revised

This chapter mainly introduced the database structure and access methods. As a database, the typical organization is as a collection of files, records, and stored on a disk. A file is organized by a sequence of record. A record is a sequence of fields related to a single logical entity. And field is a place for a data item in a record.

For the unordered file, it does not have structure except entry order, so new records are inserted into it in the next physical position on the disk. And when a record deleted, a "hole" will be created. Linear search could be used on the unordered files. Linear search required to scan every record, and the time complexity is $O(n)$. For an ordered file organization, each record is inserted into the file in the order of values in its field. Binary search may be used on ordered fields. B-tree is an index structure that handles the modifications. The B-tree has some operations, such as search insert and delete. From the book, we can get the B-tree has several properties. First of all, the B-tree is completely balanced. Secondly, insertion and deletion have time complexity $O(\log n)$, search has time complexity $O(\log n)$. The last one is each node is guaranteed to be at least half-full at all stages of the tree's evolution.

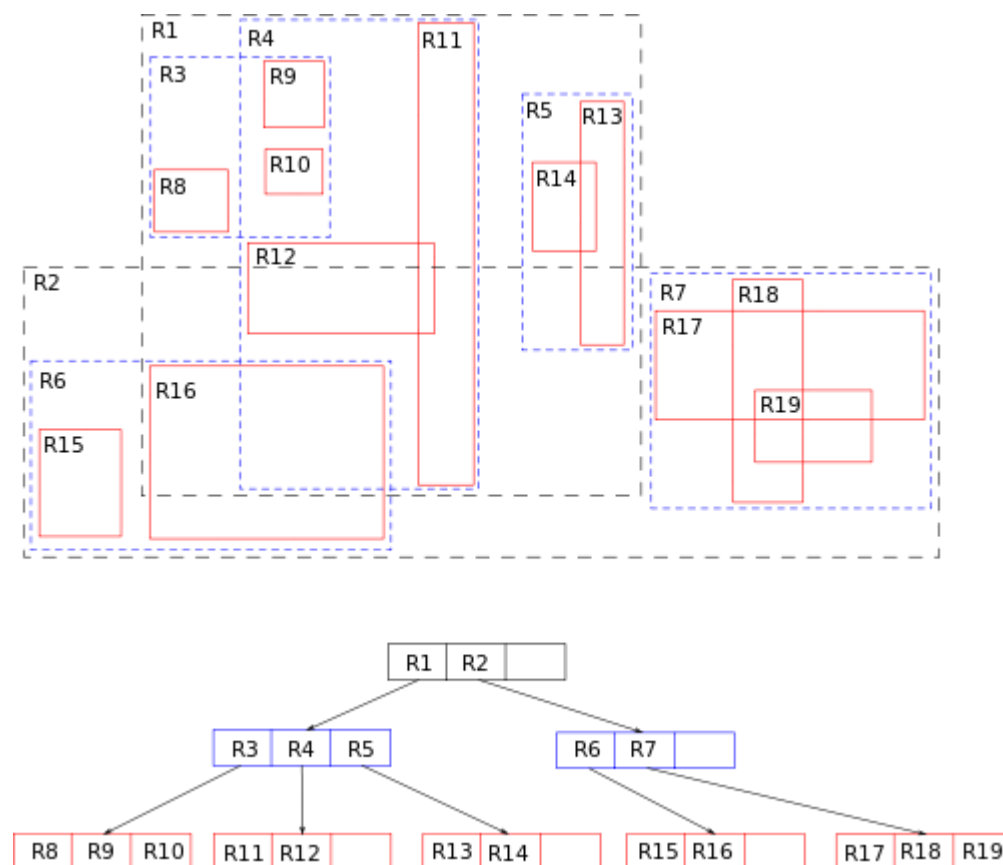
The spatial dimensions and spatial data are orthogonal, but they are depended with each other. There are two types of queries on the planar points: point and range queries. There are six common tile indexes: row order, row-prime order, Cantor-diagonal order, spiral order, Morton order and Peano-Hilbert order. Each tile index has different features. The row order corresponds to the scan of a raster image across the rows of pixels. The row prime order provides a simple modification. The Cantor-diagonal order and spiral order are good at preserving nearness. Morton and Peano-Hilbert orderings are therefore space filing and can be constructed at arbitrary levels of detail.

Raster is able to represent a large range of computable spatial objects. A raster when stored in a

raw state with no compression cannot be efficient in terms of computer storage. In order to represent the raster boundary of a region, chain codes could be used. Run-length encoding (RLE) is an alternative representation that counts the length of “run” of consecutive cells. Block encoding is a generalization of run-length encoding to two dimensions. Region quadtree is used for holding planar areal raster data, and its principle is a recursive subdivision of a nonhomogeneous square array of the cells into four equal-sized quadrants.

The point object structures are considered as grid files, point quadtrees and 2D-trees. The grid structure allows binary search methods to be applied in more than one dimension. The point quadtree combines the grid approach with a multidimensional generalization of the binary search tree. The 2D-tree solves the exponential increase in the number of descendant of non-leaf nodes as the dimension of the embedding space increases. PM quadtree is a variant of region quadtree designed to storing linear objects. It has several types of PM quadtree, including PM1, PM2 and PM3 quadtrees. Efficient indexes for rectangles are important because rectangles can be used to get bounded planar spatial objects. Each geometric object is enclosed in its MMB (minimum bounding box). The R-tree is a rooted tree, and its node represents a rectangle. The leaf nodes represent containers for the actual rectangles to be indexed. R-tree has the time complexity $O(n)$ on the worst case, and $O(\log n)$ on the best case.

One major reason for using R-tree based index structure is the requirement to index not only point data but also extended spatial data, and R-tree based index structures are well suited for both data type.



The R-tree range search algorithm

RangeSearch(TypeNode RN, TypeRegion Q)

/* Finds all rectangles that are stored in an R-tree with root node RN, which intersect with a query rectangle Q. Answers are stored in set A. */

if RN is not a leaf node

 examine each entry e of RN to find e.mbb that intersect Q;

 foreach such entry e call RangeSearch(e.p, Q);

else // RN is a leaf node

 examine each entry e to find e.mbb that intersects Q;

 add these entries to answer set A;

endif

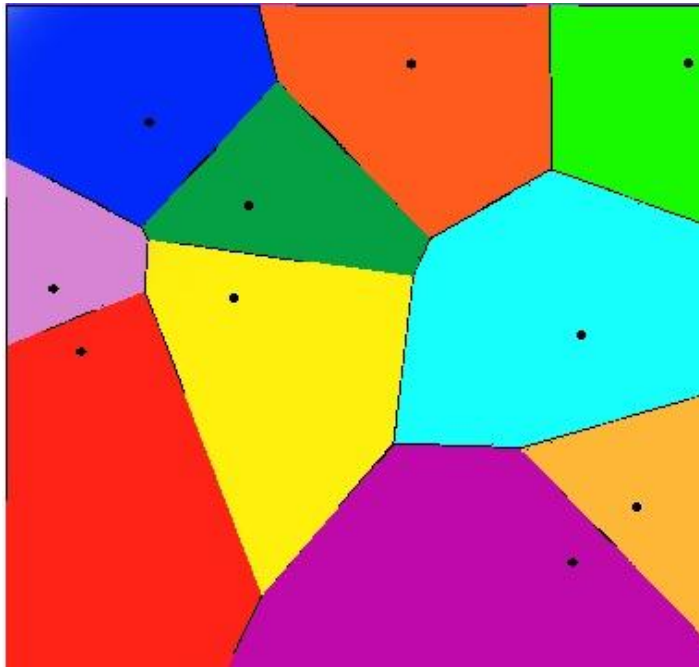
The search starts from the root node of the tree. For every rectangle in a node, it has to be decided if it overlaps the search rectangle or not. If yes, the corresponding child node also has to be searched. Searching is done like this in a recursive manner until all overlapping nodes have been traversed. When a leaf node is reached, the contained bounding boxes are tested against the search rectangle and their objects are put into the result set if they lie within the search rectangle.

The R+-tree is a refinement of the R-tree that does not permit over-lapping rectangles associated with non-leaf nodes. For R+-tree, the entries of any internal node do not overlap and the nodes are not guaranteed to be at least half filled like R-tree. R+-tree could be larger than an R-tree built on same data set and R+-tree is more complex than the construction and maintenance of R-tree.

Spherical tessellations provide some methods to surface of the Earth; the quaternary triangular mesh (QTM) recursively can get locations on the surface of a sphere by nested collection of equilateral triangles.

Summary of GIS Article revised

The article Voronoi Diagram talked about the definition, created steps, key applications, and future direction of Voronoi Diagram. And, the author used several examples to explain Voronoi Diagram. In general, this method decomposes a set of objects in a spatial space to a set of polygonal partitions. There are many variants to the Voronoi Diagram such as the k-th order, the Farthest-Site. In order to create a Voronoi Diagram, we need to find all the points that are the nearest to the generated object and the points that may have more than one closest generated point at first. Then, we also need to find the polygonal shapes around each object. Thus, the polygonal shapes comprise the completed first order Voronoi Diagram. Voronoi Diagram had already been used in many domains, such as science, astronomy, biology, nearest neighbor problem, and route planning. Voronoi Diagram can be widely used in everything that represented spatially. For example, marketing researchers can use them to create geographical representations of sales. And zoologists can use them to identify habitats of various species.



The article R-Trees talked about the R-Tree, which has been show on the textbook from page 250 to 253. The R-tree structure is the most important access methods in the area of Spatial Data Management. The R-tree comprises a generalization of the B+-tree structure for multiple dimensions. The splitting criteria are as the following: Linear Split, Quadratic Split, and Exponential Split. R-tree is diverse. Base on it, some other structure derived, such as R+-tree, R*-Tree, Static R-tree, Packed R-tree, Hilbert Packed R-tree, STR R-tree and Time-Evolving R-tree. R-tree has been used in many areas, such as spatial data management, p2p system, data streams, and all aspects concerning a database system. For example, it can be used for processing spatial queries. It also can be used in spatio-temporal database, multimedia database and data warehousing and data mining. Based on the research, it covers almost every aspect of concerning a database. R-tree will be a ubiquitous data structure and it will be important in modern system and application.

R*-tree is an improvement of R-tree and it is an efficiently access method for points and rectangles. R*-tree supports point and spatial data at the same time and its implementation cost is a little higher than of other R-tree variants. In the article, the author introduces the optimization criteria for R-trees and how to choose the subtree. Then, the author analysis how to split a node and forced reinsert. Now, R*-tree has been used in geographic information system (GIS), digital mock-up (DMU), and multidimensional feature vectors. The R*-tree is a popular access method in database systems organizing both multidimensional point data and spatial data. R*-tree will be more suitable for the more complex object representation. The insertion strategy to the R*-tree is with $O(N \log N)$.

The article Mobile Object Indexing introduces the definition of mobile object indexing and the way for indexing in one dimension and two dimensions. The traditional way to record the position of moving objects in one and two dimensions is storing the location, start time, and velocity vector. Once the object's motion changed, the motion information will be updated. The drawback of this

way is it can't be applied when object moves to infinity. Mobile object indexing is pretty useful in "location-aware applications such as traffic monitoring, intelligent navigation and mobile communications management require the storage and retrieval of the locations of continuously moving objects" (Kollios, Tsotras, Gunopulos, "Encyclopedia of GIS", 670). For example, in an air-traffic control system, the locations of the airplanes that are flying close to an airport or a city must be continuously monitored. In that case, a predictive range query can be issued every few seconds. This index will speed up the search and allow for multiple queries to be executed each minute.

T3-Revisions for narrative

Based on the reviewers' suggestion, we improved our narratives by adding some details and diagrams. First of all, some algorithms' time complexities had been talked in the narratives, such as R-tree, R+-tree and R*-tree. Secondly, the R-tree's searching diagram and algorithm has been added. The search starts from the root node of the tree. For every rectangle in a node, it has to be decided if it overlaps the search rectangle or not. If yes, the corresponding child node also has to be searched. Searching is done like this in a recursive manner until all overlapping nodes have been traversed. When a leaf node is reached, the contained bounding boxes are tested against the search rectangle and their objects are put into the result set if they lie within the search rectangle. Thirdly, some details about R-tree have been added. One major reason for using R-tree based index structure is the requirement to index not only point data but also extended spatial data, and R-tree based index structures are well suited for both data type. Fourth, societal motivation had been added. For Voronoi diagrams, anything that can be represented spatially can use it. For example, marketing researchers can use them to create geographical representations of sales. And zoologists can use them to identify habitats of various species. R-tree has been used in many areas, such as spatial data management, p2p system, data streams, and all aspects concerning a database system. R*-tree is used in GIS system and digital mock-up. Mobile object indexing is used in traffic monitoring, intelligent navigation and mobile communication. At last, we modified some detail about the computer science motivation. R* tree is an improvement of R-tree, it is a popular access methods for points and rectangles and R* tree supports both of them at the same time. R* tree modify R-tree's insert and split algorithms. The implement cost is a litter higher than that of other r-tree variants.

For mobile Object Indexing, an object's movement can be presented through a linear function of time with their initial location, a starting time instant and a velocity vector.

Project Introduction

The limited accessed website based on specified geo-location is one implementation of geo-code Internet. It limited the website access permission based on the geo-location. By using this technique, the geo-location application for the web can be written directly in the browser. The limited accessed website means the setting website only can be browse in the setting location. This technique will improve the website's security and convenience. At the same time, the limited accessed website will help user to control the administrative privileges much easily than before. For example, the website administrators specify the people who are in UMN campus can browse the UMN website.

The limited accessed website will improve the security. This technique can be implemented in some top classified organizations that require the user's location in order to reduce hacker's attacks. For example, a company website usually includes lots of business documents, trading record and trade secrets. Maybe this information could attract the hackers to attack in order to get some business information. If the company use the limited accessed website, only people in the company have the permission to browse the company's website, it will reduce the attack risk. The company's business documents and secrets will be much more safe.

Using the limited accessed website will be much more convenience. It can share the resource with people in the same geo-location much easily. For example, UMN libraries has lots of ebooks, the limited accessed website can set the people in UMN campus automatically have the permission to use these ebooks. There is another example, as we know every year, UMN will spend lot of money to buy the IEEE account. If we can set the geo-location as the UMN campus, and the people in this location can use the UMN account to browse the IEEE website, it will be much more convenient for professors and students.

The limited access website also can control the administrative privileges much effective. People can set different administrative privileges based on the different geo-location. For example, in a company, different department has their own hardware and schedule. Usually, there are some printers in a company. User can set each printer to different departments. Based on the limited website, different department have their own limited access online print system. From online printer system, the different department can use different printers based by their access permission. There is another example that everyday people need to punch in. The different department can have their own punch in system. This system could be a limited accessed website base on the every department's geo-location. From these two examples above, we can find that the limited access websites can control the administrative privileges much effective.

Codified Information

To implement our project, it involves several techniques. First of all, HTML 5 Geolocation is used to locate a user's position. We want to restrict our area to University of Minnesota, Twin city campus. Shapefiles for twin city campus buildings are converted into KML file in order to display in Google Maps as KmlLayer. To improve user's experience, jQuery is used to get the coordinates of certain area or building which is selected by website administrator. Then when users try to access the website, the website will get user's current location and determine whether it is inside of the area which is selected by website administrator. To simplify our code, we want to implement function built in Google Map API. The codified documents are listed below:

W3C Geolocation API: <http://dev.w3.org/geo/api/spec-source.html>

Shapefile introduction: <http://en.wikipedia.org/wiki/Shapefile>

KML tutorial: https://developers.google.com/kml/documentation/kml_tut

KML reference: <https://developers.google.com/kml/documentation/kmlreference>

KmlLayer example: <https://google-developers.appspot.com/maps/documentation/javascript/examples/layer-kml>

jQuery API: <http://api.jquery.com/>

Geometry library documentation: <https://developers.google.com/maps/documentation/javascript/reference#poly>

Project Milestone

- Implement Geolocation with HTML5 to locate user's location, shown in figure 1.
- Convert shapefile into KML and display in Google Maps, shown in figure 2.

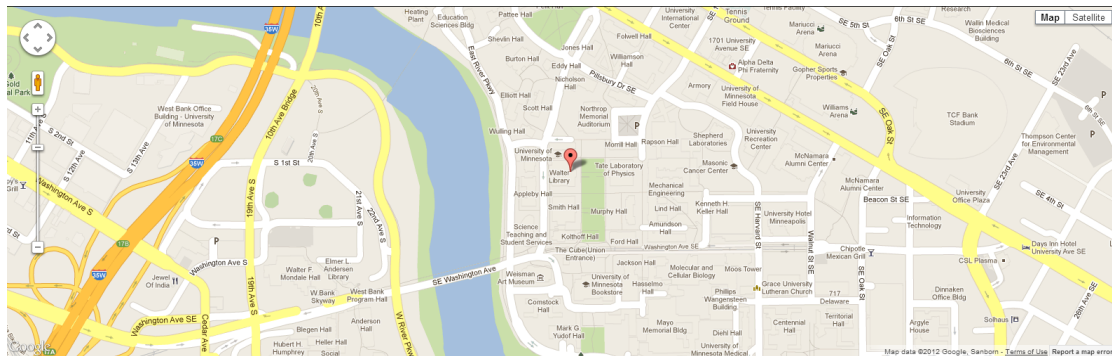


Figure 1. Geolocation in HTML5 to get user's current location

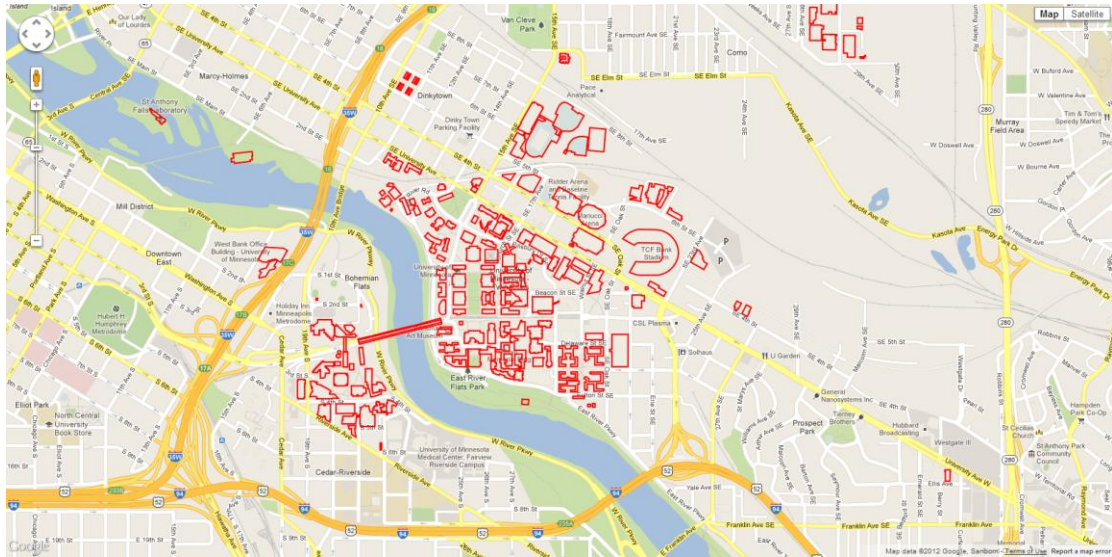


Figure 2. KmlLayer on Google Maps

Novel Information

For use the KmlLayer in Google Maps, the KML file must be located on a server which can be reached by Google. If the KML is on the local host machine, the KmlLayer will not show up since Google could not read the file. A trick here is we can put the KML file in the dropbox's public folder during the coding process. By doing this, we can still develop our website on local machine, no need to synchronize to the server side.

Current limitations

Our objective is the website master can specify a certain area, which allows users in this certain area can get access to the website, while users outside of the area will not be able to access the website. For right now, we have the shapefile for University of Minnesota Twin city area. This shapefile contains both buildings in Como area, East Bank, West Bank, and St. Paul area. In order to display this shapefile into Google Maps, the shapefile need to change its format to KML file. KML file format is supported by Google Maps and Google Earth. We have already converted the shapefile into KML file successfully. However, all the buildings in Twin city campus are shown in Google Maps since all the data are stored in a single KML file. In order to fulfill our project's objective, we need to manipulate KML via Google Maps API. There are three limitations currently: 1. Retrieve coordinates from KML file. 2. Determine if a coordinates are inside a KML region. 3. The accuracy of geolocation.

1. Retrieve coordinates from KML file.

In order to create a user-centered UI, we want to build functions to highlight the buildings user selected. However, it seems there is no way to access the Javascript

objects Google Maps creates when we add a KmlLayer. Because Google Maps renders layer overlayed tiles that get rendered on the server side.

2. Determine if a coordinates are inside a KML region.

After reviewing the Google Maps API about KmlLayer, it found out that there is no functions like `containsLocation(point: LatLng, polygon: Polygon)` on KmlLayer.

3. The accuracy of geolocation

The W3C Geolocation API is intentionally agnostic to the method the browser users to locate the device. A browser on a mobile device or a laptop will likely give an accurate result. A device connected via Ethernet will only give a result as accurate as the IP address can reveal.

Solution Strategies

1. Retrieve coordinates from KML file.

There are some possible ways to overcome the limitation.

- Create multiple KML files, one for each set of placemarks. Once user choose a certain area or building, it is easy to show the corresponding KmlLayer.
- Use jQuery to extract the coordinates of the user specified area or building. We can store the coordinates and placenames in a local array and render the shape as polygon.

2. Determine if a coordinates are inside a KML region.

Since we need to extract the coordinates for the area or building, we can create a polygon use the Geometry library of Google Maps API and use function `containsLocation()` to compute whether the given coordinates lies inside the specified polygon.

3. The accuracy of geolocation

Although we can set `enableHighAccuracy` attribute as true, it still hard to satisfy the performance in some case, such as if a user stands close to the boundary of an area or a building.

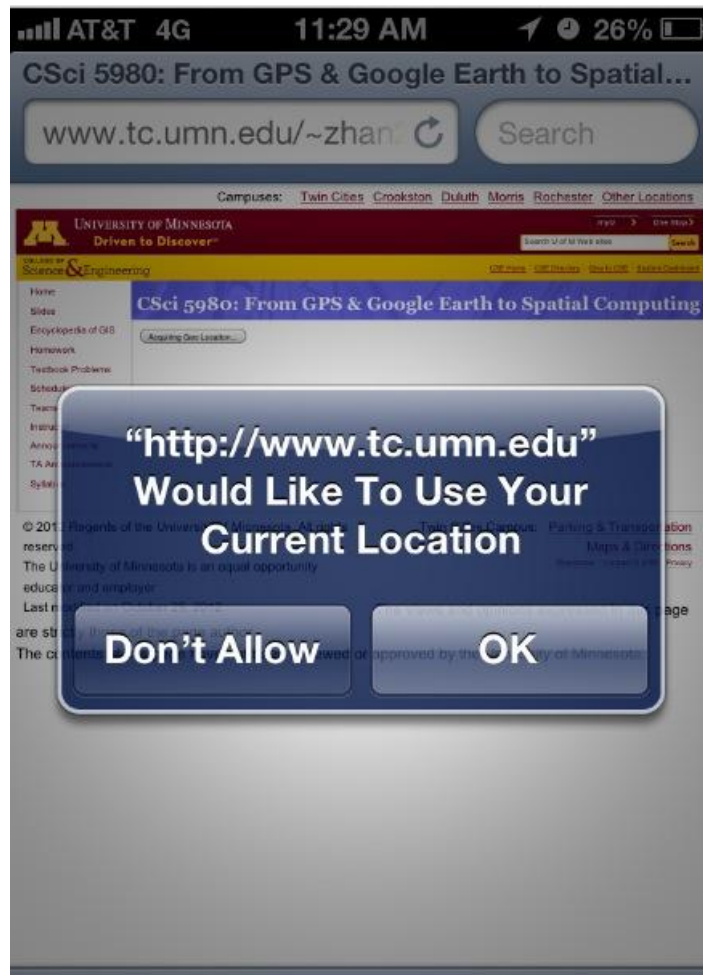
Reference

- Localizing the Internet: Implications of and Challenges in Geo-locating Everything Digital, Michael R. Evans, Chintan Patel
- Geolocation API Specification, <http://dev.w3.org/geo/api/spec-source.html>
- Google Maps API, <https://developers.google.com/maps/documentation/javascript/reference>
- jQuery API: <http://api.jquery.com/>
- KML reference: <https://developers.google.com/kml/documentation/kmlreference>





Web 3

http://www.tc.umn.edu/~zhan2191/o3_gps.html






Require for permission to access the current location



In building

 **AT&T 4G** **11:50 AM**   **68%** 

Current position: lat=44.965348878348344, long=-93.26934389858225
(accuracy 69.86230626512928m)
last reading taken at: 11:49:21
Current position: lat=44.96602667471171, long=-93.2706098116779 (accuracy
67.51592509871858m)
last reading taken at: 11:49:28
Current position: lat=44.96654524250703, long=-93.27112025514774
(accuracy 65m)
last reading taken at: 11:49:32
Current position: lat=44.966670018699695, long=-93.2719909948938
(accuracy 65m)
last reading taken at: 11:49:38
Current position: lat=44.96644325099665, long=-93.27363112968987
(accuracy 66.8200045061757m)
last reading taken at: 11:49:44
Current position: lat=44.96637853894469, long=-93.27431980142403
(accuracy 65m)
last reading taken at: 11:49:47
Current position: lat=44.96640597924433, long=-93.27648194461631
(accuracy 65m)
last reading taken at: 11:49:53
Current position: lat=44.966384471267645, long=-93.27816149994676
(accuracy 65m)
last reading taken at: 11:50:00
Current position: lat=44.966454832777636, long=-93.27859713562091
(accuracy 65m)
last reading taken at: 11:50:04
Current position: lat=44.96591247627291, long=-93.27885989748647
(accuracy 65m)
last reading taken at: 11:50:11
Current position: lat=44.965797401608214, long=-93.27888978096115
(accuracy 65m)
last reading taken at: 11:50:15
Current position: lat=44.96547554273524, long=-93.28066976615831
(accuracy 65m)
last reading taken at: 11:50:22
Current position: lat=44.965220946245644, long=-93.28191024745401

Outside the building

AT&T 4G 11:30 AM 26%

CSci 5980: From GPS & Google Earth to Spatial...

www.tc.umn.edu/~zhan Search

Campuses: Twin Cities Crookston Duluth Morris Rochester Other Locations

UNIVERSITY OF MINNESOTA
Driven to Discover™

myU One Stop

Search U of M Web sites

COLLEGE OF Science & Engineering

CSE Home CSE Directory Goe to CSE Student Dashboard

Home
Slides
Encyclopedia of GIS
Homework
Textbook Problems
Schedule
Teams
Instructor
Announcements
TA Announcements
Syllabus

CSci 5980: From GPS & Google Earth to Spatial Computing

Current position: lat=44.9742121531978, long=-93.23178727944058 (accuracy 65m)
last reading taken at: 11:29:59
Current position: lat=44.97420705219831, long=-93.2316020152842 (accuracy 65m)
last reading taken at: 11:30:05
Current position: lat=44.97420972178871, long=-93.23144202036397 (accuracy 65.37772686445884m)
last reading taken at: 11:30:12
Current position: lat=44.97424394906653, long=-93.23150543621762 (accuracy 65m)
last reading taken at: 11:30:18
Current position: lat=44.97413760986374, long=-93.23135211626483 (accuracy 69.79919149029084m)
last reading taken at: 11:30:25
Current position: lat=44.97418491244713, long=-93.23177477395333 (accuracy 65m)
last reading taken at: 11:30:31
Current position: lat=44.974345550462246, long=-93.23165531605665 (accuracy 65m)
last reading taken at: 11:30:38

Stop

© 2012 Regents of the University of Minnesota. All rights reserved.
The University of Minnesota is an equal opportunity educator and employer
Last modified on October 29, 2012

Twin Cities Campus: [Parking & Transportation](#)
[Maps & Directions](#)
[Directories](#) [Contact U of M](#) [Privacy](#)

The views and opinions expressed in this page are strictly those of the page author.

◀ ▶ ↺ 📖 7

In the Car

AT&T

7:40 PM

100%

CSci 5980: From GPS & Google Earth to Spatial...

www.tc.umn.edu/~zhan

Search

Campuses: [Twin Cities](#) [Crookston](#) [Duluth](#) [Morris](#) [Rochester](#) [Other Locations](#)

UNIVERSITY OF MINNESOTA
Driven to Discover™

myU Search U of M Web sites Search

COLLEGE OF
Science & Engineering CSE Home CSE Directory Give to CSE Student Dashboard

Home
Slides
Encyclopedia of GIS
Homework
Textbook Problems
Schedule
Teams
Instructor
Announcements
TA Announcements
Syllabus

CSci 5980: From GPS & Google Earth to Spatial Computing

Current position: lat=44.97076077856586, long=-93.22336077698806 (accuracy 10m)
last reading taken at: 19:40:15Accuracy not sufficient (1414m vs 150m) - last reading taken at: 19:40:15Accuracy not sufficient (1414m vs 150m) - last reading taken at: 19:40:15
Current position: lat=44.97127687084941, long=-93.22343769583279 (accuracy 65m)
last reading taken at: 19:40:15
Current position: lat=44.971175346402596, long=-93.22346940004206 (accuracy 65m)
last reading taken at: 19:40:16
Current position: lat=44.97109280174991, long=-93.22351969241335 (accuracy 65m)
last reading taken at: 19:40:20
Current position: lat=44.97099559758321, long=-93.22383234286049 (accuracy 50m)
last reading taken at: 19:40:21
Current position: lat=44.97095435861961, long=-93.22373704062143 (accuracy 30m)
last reading taken at: 19:40:22
Current position: lat=44.970968943131126, long=-93.22377048441508 (accuracy 30m)
last reading taken at: 19:40:23
Current position: lat=44.97095494535283, long=-93.22372346193829 (accuracy 10m)
last reading taken at: 19:40:24

Stop

© 2012 Regents of the University of Minnesota. All rights

Twin Cities Campus: [Parking & Transportation](#)

7

AT&T

7:41 PM

100%

Homework

Textbook Problems

Schedule

Teams

Instructor

Announcements

TA Announcements

Syllabus

Current position: lat=44.97076077856586, long=-93.22336077698806 (accuracy 10m)
last reading taken at: 19:40:15Accuracy not sufficient (1414m vs 150m) - last reading taken at: 19:40:15Accuracy not sufficient (1414m vs 150m) - last reading taken at: 19:40:15
Current position: lat=44.97127687084941, long=-93.22343769583279 (accuracy 65m)
last reading taken at: 19:40:15
Current position: lat=44.971175346402596, long=-93.22346940004206 (accuracy 65m)
last reading taken at: 19:40:16
Current position: lat=44.97109280174991, long=-93.22351969241335 (accuracy 65m)
last reading taken at: 19:40:20
Current position: lat=44.97099559758321, long=-93.22383234286049 (accuracy 50m)
last reading taken at: 19:40:21
Current position: lat=44.97095435861961, long=-93.22373704062143 (accuracy 30m)
last reading taken at: 19:40:22
Current position: lat=44.970968943131126, long=-93.22377048441508 (accuracy 30m)
last reading taken at: 19:40:23
Current position: lat=44.97095494535283, long=-93.22372346193829 (accuracy 10m)
last reading taken at: 19:40:24
Current position: lat=44.9709987407969, long=-93.22376579054931 (accuracy 10m)
last reading taken at: 19:40:25
Current position: lat=44.97100200973914, long=-93.22377953687051 (accuracy 10m)
last reading taken at: 19:40:26
Current position: lat=44.97096529700325, long=-93.22374718272427 (accuracy 10m)
last reading taken at: 19:40:27
Current position: lat=44.97095431671009, long=-93.2237388008211 (accuracy 5m)
last reading taken at: 19:40:30
Current position: lat=44.97095431671009, long=-93.2237388008211 (accuracy 5m)
last reading taken at: 19:40:30
Current position: lat=44.97090813242362, long=-93.223704183561 (accuracy

7