Navigating the Future: Mobile Geolocation Search with Spatial Freedom

Ole Kr. Aamot

7 January 2024 (Initial Draft)

Contents

List of Figures

List of Tables

Listings

Chapter 1

Navigating the Future: Geopher Mobile Geolocation Search with Spatial Freedom

Introduction:

In the rapidly evolving landscape of technology, mobile geolocation search has become an integral part of our daily lives, transforming the way we navigate physical and digital spaces. This article explores the dynamic realm of mobile geolocation search, emphasizing spatial freedom, and the convergence of latitude, longitude, and altitude within the World Wide Web and Electronic Mail.

The Essence of Spatial Freedom:

Traditional geolocation systems primarily focus on latitude and longitude, providing coordinates that define a point on the Earth's surface. However, the future of mobile geolocation search involves expanding this paradigm to include altitude and embrace the concept of spatial freedom. Spatial freedom implies the ability to move in any direction without constraints, breaking away from traditional parallel systems and allowing for more nuanced and accurate positioning.

No-Paralleity: Redefining the Geolocation Paradigm:

The concept of no-paralleity challenges the traditional notion of parallel systems in geolocation. Instead of adhering strictly to latitude and longitude lines, no-paralleity envisions a dynamic, three-dimensional space where users can move freely without being confined to predefined axes. This approach enables more precise location tracking and opens the door to innovative applications across various industries.

Free Movements in Space:

The integration of altitude into geolocation systems enriches the user experience by providing information about vertical positioning. This is particularly valuable in scenarios where altitude plays a crucial role, such as indoor navigation, aviation, and augmented reality applications. Free movements in space encompass not only lateral movement but also vertical mobility, creating a comprehensive understanding of a user's position in three-dimensional space.

World Wide Web Integration:

The World Wide Web has become an indispensable platform for information access and communication. Integrating spatial freedom into the web environment enhances location-based services. Whether it's finding nearby businesses, navigating complex structures, or interacting with augmented reality elements, the marriage of spatial freedom and the World Wide Web creates a seamless and immersive user experience.

Electronic Mail and Geolocation:

Email communication has transcended its traditional text-based format, incorporating rich media and interactive elements. Geolocation in electronic mail opens up new possibilities for context-aware communication. Imagine receiving an email that not only contains textual information but also includes location-based data, allowing users to visualize the sender's location and associated spatial context.

Conclusion:

The future of mobile geolocation search is marked by spatial freedom, no-paralleity, and the integration of latitude, longitude, and altitude in the World Wide Web and Electronic Mail. This paradigm shift holds immense potential for revolutionizing location-based services, navigation systems, and communication platforms. As technology continues to advance, embracing spatial freedom ensures that our digital experiences seamlessly align with our movements in the physical world.

Appendix A

Haversine Computation of geopher.com Geolocations: geopher-location-computation-search.py

The following Python program is a resolver for piperpal.com

```
# Written by Ole Aamot, 20240107

from cgi import parse_qs, escape
from urllib import quote_plus
import cgi
import urllib3

import hashlib
from math import radians, cos, sin, asin, sqrt
import textwrap
from cgi import parse_qs, escape
from bs4 import BeautifulSoup
import cgi
import time

def piperpal_resolver(l,n,lat,lon):
    p = http.request('GET', l)
    o = p.data
    y = BeautifulSoup(o, "lxml")
    m = hashlib.sha256(p.data).hexdigest()
    locationtags = y.find_all("location")
    i = 0
    print locationtags
    while (i < len(locationtags)):
        notbefore = y.findAll("location")[i]["notbefore"]
        notafter = y.findAll("location")[i]["notafter"]
        name = y.findAll("location")[i]["name"]
        glat = y.findAll("location")[i]["lat"]
        glon = y.findAll("location")[i]["lon"]
service = y.findAll("location")[i]["service"]
        data = y.findAll("location")[i]
        href = y.findAll("location")[i]["href"]
        r = haversine(float(glat),float(glon),float(lat),float(lon))
#        print map
        print(gindex(r,name,m,href,data,glat,glon,lat,lon))
        for x in range(-90, 91):
           for y in range(-180, 181):
               print x,y,lat,lon,q,l,0
               print x,y,lat,lon,q,l,1
               for z in range(1, 11):
                     print x,y,lat,lon,q,l,z
                     linker = 'https://api.piperpal.com/location/rob
                     insert = http.request('GET', linker)
                     print linker
                     print insert
#linker = 'https://api.piperpal.com/location/robot.php?name=' + quot
# insert = http.request('GET', linker)
# print linker
# print insert
# object = insert.data
# tester = BeautifulSoup(object, "lxml")
# macron = hashlib.sha256(insert.data).hexdigest()
i = i + 1
        piperpal_resolver(href,n,lat,lon)
    return (y,i)

def haversine(lat1, lon1, lon2, lat2):
    """
    Calculate the great circle distance between two points
    on the earth (specified in decimal degrees)
    """
    # convert decimal degrees to radians
    lon1, lat1, lon2, lat2 = map(radians, [lon1, lat1, lon2, lat2])

    # haversine formula
    dlon = lon2 - lon1
    dlat = lat2 - lat1
    a = sin(dlat/2)**2 + cos(lat1) * cos(lat2) * sin(dlon/2)**2
    c = 2 * asin(sqrt(a))
    r = 6371 # Radius of earth in kilometers. Use 3956 for miles
    return c * r

def gindex(radius,n,d,l,query,my_lat,my_lon,lat,lon):
    return radius,n,d,l,query,my_lat,my_lon,lat,lon
maps = dict()
maps['Books','Food'] = dict()
http = urllib3.PoolManager()
q = 'Wikipedia'
p = 1
l = 'https://piperpal.com/piperpal.xml';
lat = '37.42242500'
lon = '-122.08755550'
notbefore = '2024-01-01'
notafter = '2024-12-31'
service = 'Books'
piperpal_resolver(l,q,lat,lon)
```