Derek Choi (derekchoi@college.harvard.edu) Ramsey Fahs (cfahs@college.harvard.edu)
Alice Tang (alicetang@college.harvard.edu) Jessica Zhu (jessicazhu@college.harvard.edu)

SUGAR, SPICE, AND EVERYTHING NICE: CRIME-FIGHTING CS51 PROJECT *OR "MOOOOOPQUEST" OR "MAP-COWST"*

A Treatife and Examination
Into
The Most Specific Pathf
That Enable Mankind to Traverfe
Without Error or Regard to His Safety
The Dangerouf World of Governor Stuyvefant's City of New York

"I see myself as a fish in a stream; deflected;
held in place;
but cannot describe the stream."

— Virginia Woolf on refs and streams

Brief Overview

Navigational tools do not address the safety of a given route. When walking in New York City, a tool like Google Maps will provide the most efficient route, but it will not account for the safety of the given path.

To fix this problem, we plan on integrating historical crime data with open source map navigation technology to generate a route-calculating algorithm that will account for path safety. We hope to develop an effective navigation tool for New York City that will take parameters for safety in determining efficient walking directions.

Feature List

Core Features: Manhattan between 100th and 120th Streets

switched to Python, did not factor out info

- Implement the OpenStreetMap (OSM) data (https://www.openstreetmap.org) for walking by factoring out unnecessary qualifications (potentially using an OCaml OSM parser: http://wiki.openstreetmap.org/wiki/OSM_ocaml_parser).
- Implement an algorithm that will manipulate OSM data by adding parameters for crime levels, potentially.
- Manipulate OpenStreetMap data to include crime history data from 2014-2015.
- Manipulate PyRoute algorithm to take into account our manipulation of the OSM: https://wiki.openstreetmap.org/wiki/Pyroute. The new algorithm will generate a walking route that avoids high crime blocks, thus using the new weights from the input OSM data.

Cool Extensions

- User preferences, manipulating ways (http://wiki.openstreetmap.org/wiki/Way)
    - Generate an OSM manipulating algorithm that will allow user specification of type of crime (particularly violent vs. nonviolent).
    - Generate an OSM manipulating algorithm that will allow user specification of time and safety parameters by factoring these parameters (which would potentially add up to 1) into the calculation of the edge weights.
    - Generate an OSM manipulating algorithm that will allow user specification of limitations of duration of trip/extra distance willing to walk, thus changing the importance of avoiding high crime areas based on input.
- Expand coverage area to all of Manhattan (as opposed to the local area mentioned in the "Core Features" section).
- Avoid high crime areas a certain radius from a crime location (as opposed to only avoiding blocks of previous crimes).
- Implement a comparison of routes that do take crime into account and routes that don't.

switched to Python, did not factor out info

Technical Specification

It will be possible to separate into one module the data pulling and merging. After that, another module can calculate the route using a modified version of the PyRoute algorithm that considers the relative safety of different paths.

Module One: Extract map data and modify it to take crime into account.

- Extract map data from OpenStreetMap, focusing solely on New York City.
    - We will initially focus on Manhattan from 110th to 120th streets.
- Extract crime data from http://maps.nyc.gov/crime/ using method outlined in http://thomaslevine.com/!/nyc-crime-map/.

- Add a crime index tag to OSM nodes/ways where crime has occurred.
    - Compare latitude and longitude of crime data with latitude and longitude of nodes to identify which street (edge/way) is relevant to each crime.

Module Two: Manipulate routing algorithm to account for this new crime index weight field.

- Disincentivize routing engine from using less safe (or "more weighted") paths in favor of safer (or "less weighted") ones.
- The new OSM data generated from Module One will be inputs to the modified PyRoute algorithm.

Module Three: Implement the "front-end" so to speak

- Take in arguments (current location and desired location).
- Call updated routing algorithm and display route.
- PyRoute has a GUI implementation that will hopefully still work with our changes.

Cool Extensions

- Modify crime index tags based on user sensitivity to crime. (*Module One*)
    - Change crime index weights by a factor that depends on user sensitivity.
- Modify crime index tags based on crime history (accounting for recent/older crimes). (*Module One*)
    - Only use data from the past *x* months when determining crime index weights.
- Modify crime index tags based on crime type (what the user wishes to avoid). (*Module One*)
    - Only use data from certain crimes when determining crime index weights.
- Add crime index tag to OSM nodes/ways in *vicinity* of crime (each crime has a "radius" when determining which streets are affected). (*Module One*)
    - When comparing latitude/longitude, allow greater distance between crime and nodes, and determine crime index weights accordingly.

- Manipulate routing algorithm to take in parameters for time (maximum duration of trip), distance (extra distance willing to walk). (*Module Two*)
    - If time and distance restrict the algorithm's original output, maximize crime index weight (greatest sensitivity) within the given time or distance.

Next Steps

- We really need to delve into the OpenStreetMap documentation to understand the nuances of the API (they format it as XML nodes, tags, and ways).
- Set up coding environment: linked into GitHub (see below). Since PyRoute is written in Python, it is important that we also interpret the PyRoute code so that we can manipulate it and add safety weightings.
    - We need to set up the correct environment for PyRoute to run properly - a virtualenv with pycairo so that the GUI works.
- We've set up GitHub accounts and a repository for our project! Once we start doing things, each of us needs to clone this repository on our machine so we can work on the project and then push/pull.