Lucy Hu

David Kosbie

15112

23 November 2021

Term Project Proposal

**Project Description**

The project, named Mini CMU, illustrates a virtual CMU campus where the user can use a pathfinder to navigate to different spots and meet random students with different majors doing different things.

**Competitive Analysis**

This project does include features that can be found in some previous projects. For instance, spring 2020’s “15-112 Simulator” simulates the life of a typical 112 student, who engage in activities such as lining up for the OHQ, grabbing some food at Entropy, or sleeping through; another project of spring 2020, “CMU Life Simulator”, features a pathfinder for paths basing off buildings.

This project creates a more comprehensive CMU campus combining the features mentioned above. It includes a pathfinder for a 2D map of CMU, where the user can click to enter buildings and see other students engaging in various activities.

**Structural Plan**

* File: main
  + - Function: appStarted
    - Function: keyPressed
    - Function: mousePressed
    - Function: timerFired
    - Function: redrawAll
* File: map
  + Object: map
    - Function: init
    - Function: drawMap
  + Object: vertex
    - Function: init
    - Function: addNeighbor
  + Object: graph
    - Function: addVertex
    - Function: addEdge
    - Function: getDistance
    - Function: findPath
    - Function: findPathHelper
    - Function: printPath
    - Function: drawGraph
* File: student
  + Object: user
    - Function: init
    - Function: getUserInfo
    - Function: drawUser
  + Object: student
    - Function: init
    - Function: getMajorsDict
    - Function: getMajorsList
    - Function: getClasses
    - Function: createDataframe
    - Function: getActivities
    - Function: getBuildings
    - Function: drawStudents

**Algorithmic Plan**

* Pathfinding uses Depth First Search (DFS) and Dijkstra's algorithm. I keep track of three things: a distance dictionary (with each vertex as the key and its distance to the starting point as a value), a path dictionary (with a vertex as a key and its previous vertex of the path as the value), and a set of unvisited points (in the same direction as the end point). From the starting point, I recursively check the start point and its neighbors—every time a neighbor’s distance to the original starting point is less than the distance from the dictionary, I update the distance, the path dictionary, and remove the point from the unvisited set.
* A dataframe of 100 randomly generated students is kept. Their ‘Major’ is randomly generated from all majors from the websites of CMU’s different colleges, and their ‘College’ is matched to the major. Their ‘Year’ is randomly generated from 1 to 4, with the exception of Architecture majors from 1 to 5. Their ‘Class’ is a class that they are probably taking (generated out of a list of all classes at CMU, with a greater weight on the classes from that student’s college, and an even greater weight on the classes from their major).

**Timeline Plan**

**Timeline

Description automatically generated**

**Version Control Plan**

* I commit and push to Git at the end of the day.
  + Table

    Description automatically generated
* I also upload my files to Google Drive whenever I made big changes.
  + Graphical user interface, website

    Description automatically generated

**Module List**

* Pandas
* Beautiful Soup
* Pillow (from 112 website)
* (I don’t think I am using Numpy anymore)

**TP2 Update**

* I updated my pathfinding function because its previous version was not returning the optimal path. I added minDistance, minPath, and currentDistance as new parameters of the helper function, so that I can keep track of a best distance (and its corresponding path) and compare it with my current distance every time I run the function.
* I also separated my student.py file into two files: student.py and webscraping.py. My codes used to run slowly because every time I ran the main file (which imports student.py), I web scrapped from many websites to create a new dataframe. After I separate the files, webscraping.py can only be ran once to create the excel, and then I read data from the excel.
* For the dataframe, I also added ‘Building’ and ‘ing…’ to represent the building a student is at and what they are doing. The building is determined by their college, and what they are doing is determined by which building they are in.

**TP3 Update**

No changes made.