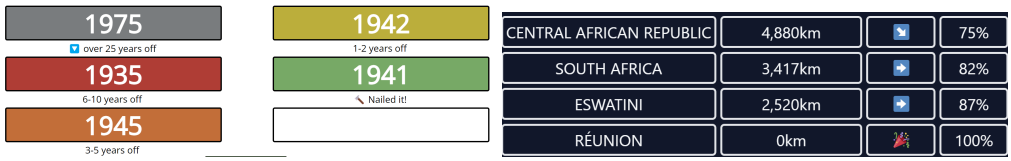
📍 Where On Earth Is That?!

Think of it as a reverse Geoguessr. Given a map of a city (e.g. your hometown), you're given the name of a point of interest (like a restaurant, a museum, a record store, a park) and you have a certain amount of guesses to pinpoint that location on a map.

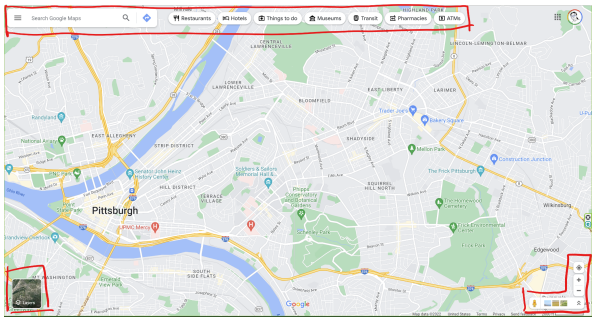
🎮 Similar games

My main inspiration for this game is not just GeoGuessr, but also the various Wordle variants out there– especially Yeardle, Semantle, and Worldle. What I liked about these variants are the clues they give for each guess. Yeardle (guess a year in history) uses color coding, and Worldle (guess a country) uses distance and compass directions. In my project, I'm hoping to use all of these clues in my game to guide the player to the right answer for a place they may not be familiar with.



Left: Yeardle's use of color coding as hints; right: Worldle's use of distance and direction as hints.

For this project, I also studied Google Maps. The most intuitive part of Google Maps' user experience on desktop was the mouse shortcuts– the user can click anywhere to drop a pin; click and drag to pan the map; and scroll the mouse to zoom in and out. I'd like to give players a similar experience of using the map in my game, but two-dimensional zooming and detecting dragging might be the hardest thing to implement for the game. If I run out of time, I can probably add a button, key shortcut, or a slider for zooming in.



Buttons are placed on the edges and corners of the page in Google Maps.

One other influence that I found was from a friend's game that she made for 15112 which also involved a map; she explained that to cut down her map's rendering time, she loaded the data into a giant array and read data from there, which was a lot faster than re-rendering everything

including data that wouldn't be shown on screen. I implemented something similar to that in my tech demo using Pandas.

⚙️ Algorithmic plan

- Displaying the map will be a bit difficult, especially with 112 graphics, since I'll have to display a ton of tiny buildings, converting from geographic coordinates to canvas coordinates and back. Storing and getting data for the buildings will be especially challenging, since there are so many buildings to render. For this, I'll only render necessary buildings within certain boundaries and use fast NumPy operations to scale every building down to size. I'm also storing building data in a Pandas dataframe for easy filtering.
- Displaying labels for buildings, streets, and other geographical features might be a little difficult. I'll have to do calculations on how the angles are tilted to line up with the directions of each street. They might be an extra feature if I run out of time.
- For better user experience, I'll try to make the game respond smoothly to mouse movements. I'll have to use mousePressed and mouseReleased and detect different mouse buttons with Tkinter. That's a little ambitious, though, and if I don't have time, I might need to use keys or use buttons instead.

🌳 Structural Plan

Functions to implement

- ☐ toMapCoords()
  - converts latitude and longitude coordinates to canvas coordinates for drawing
  - arguments: a 2 by n list of coords to convert, plus a bounding box of the map
- ☐ toCanvasCoords()
  - converts x and y coordinates on the canvas (for example, a mouse clicking on the canvas) to approximate latitude and longitude
  - same arguments as toMapCoords()
- ☐ loadData()
  - extracts data from a .pbf file filled with OpenStreetMap data
  - creates an .csv file that's a dataframe of locations and what they're categorized as (an amphitheater? a fast food restaurant? a university?)

Modes

(I will implement these if I have time.)

- Game Mode (main mode)
- Menu Mode (the main menu, where you download data to play games)
- Tutorial Mode (to teach you how to play the game, if I even have time)

Classes to implement

- **Button**
    - properties:
        - value (changeable for TextBoxes)
        - width & height
        - coordinates (northwest anchor)
        - outline and fill colors (default and when clicked)
        - id
    - methods:
        - draw (change with scrolling? will there even be scrolling at all???)
        - reaction when clicked (call function? idk how)
- **Map**
    - properties:
        - location (which city/place it's displaying, determines which file to take from)
        - bounds (an array in the form of [x0, y0, x1, y1], just like a canvas rectangle)
        - buildings/streets to be drawn? (taken from the .npy file?)
            - base, buildings, greenery, water
    - methods:
        - findFeaturesWithinBounds (search for latitudes and longitudes within certain bounds in the array)
        - redraw (draw map using polygons)
        - zoom (adjust bound box based on zoom factor and center of mouse)
        - pan (add/subtract from boundaries)
- **GuessPin**
    - properties:
        - map coordinates
        - distance from actual location + angle
            - calculate distance from actual location
        - guess #
        - color
            - calculate color based on distance from actual location
    - methods:
        - redrawPin
        - redrawClues (an arrow pointing in the direction of the mystery location, and some text hinting at how far away it is)
- **Dashboard**
    - properties:
        - point of interest to look for
        - timer
        - guess
        - other buttons (a list of buttons for changing settings)
    - methods:

- game-play
- decreasing the timer
- gameOver function
- choosing a mystery location

⏳ Timeline

TP0:

- Tech demo of the Pyrosm module– it should be just a basic static map, and that's probably what I'm going to turn in for TP0. It's just a proof of concept that things are working.

TP1:

- Get some of the basic map interactivity working.
- Start elaborating on some of the UI design.
- If possible, a main menu where you can download the data for a city or country so you can play a game based in that place.

TP2:

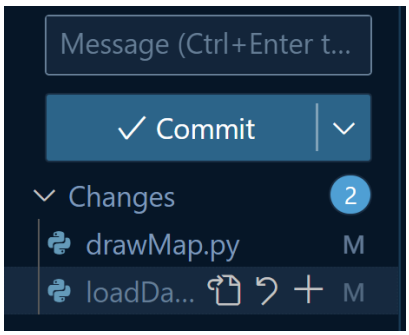
- Implement most of the game mechanics, even if they are buggy.
- I'm anticipating that the labeling for streets and maps will be *horrible* at this point.

TP3:

- Polish the UI and if possible, add extra settings.
- Perhaps add a tutorial for new players.
- Improve the labeling for streets and maps; maybe make the map a bit more detailed.
- Get the video done.

🗃️ Version Control Plan

I'm using a GitHub repository to store all my work. My setup on VSCode is connected to GitHub Desktop, which is connected to a remote repository.



💾 Module List

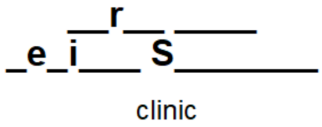
- I'm using **NumPy and Pandas** because they are *way more efficient* than the lists I used initially to store map data. Creating a large list of building coordinates in San Francisco and turning it into a NumPy array took about an hour. Simply adding them to the existing Pandas dataframe that I loaded took one minute.
- I want to make use of OpenStreetMap data because it's free and I don't need to pay money to make use of it (unlike the Google Maps API). I tried out several Python modules for downloading OpenStreetMap data, and the fastest I've found is **Pyrosm**. The only problem is that it can only download data from a select number of cities, many of which are European. For more information, check out the link below: https://pyrosm.readthedocs.io/en/latest/index.html

1️⃣ TP1 Update:

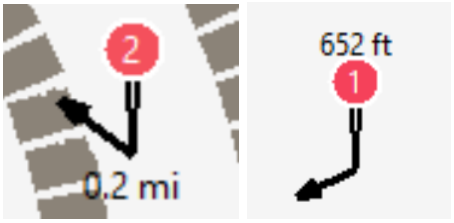
- Because the game lags a lot if it needs to draw a lot of buildings, I'll probably limit the maximum amount the player can zoom out, and for each round, I'm going to limit the player to certain boundaries in the city within a ~0.5 mile (or less) radius around the mystery location.
- I realized early on that the player could cheat in the game by searching up the name of the mystery location on Google Maps, so I'm going to make it more challenging by only revealing certain words in the mystery location's name but mentioning what kind of amenity the mystery location is (e.g., a university, a salon, or a supermarket). To find the mystery location, the player will have to rely on their own instincts (Would it be a super big building? Would it have a weird shape? Would it be close to a river?)
- I'm having difficulties figuring out how to pan the map smoothly with the mouse. I think I'm *very close* to solving the problem, but if I can't figure out the panning problem, I'll probably switch to key pressing.

2️⃣ TP2 Update:

- Instead of revealing the entire name of the location (making it easy to search up the mystery location on Google Maps), the game reveals one letter at a time, but also gives the category of the location as a clue. If the player fails to place a pin within 100 feet of the right location, the name of the location is revealed. If I have time, I might add a feature soon where the app takes a screenshot of the mystery location to show where the answer really is.

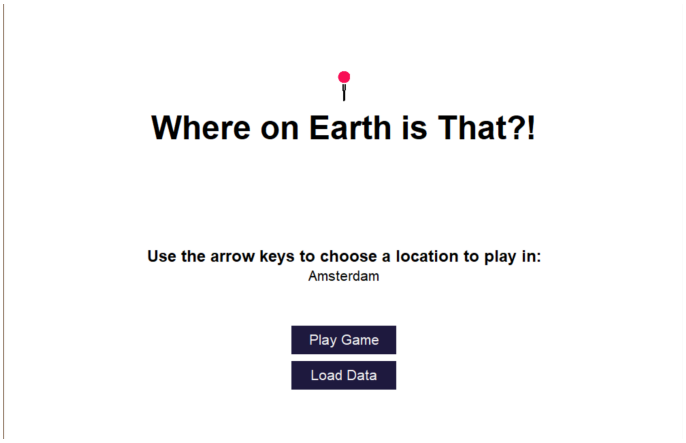


- Instead of redrawing the buildings over and over again, the app first draws the buildings using PIL and then saves the map as an image that moves around. The map, though, is *still so laggy* because the image is very large. If I want to speed it up, I might have to sacrifice image quality, making it more pixelated.
- Currently, the player is unable to drag the map with a mouse. It's been difficult trying to implement that, but if I have time before TP3 (which is unlikely) I'll try doing so.
- The pins are different colors based on how far away you are from the answer! The more orange/yellow you are, the farther away you are, and the closer the color gets to red, the closer you are. If you hover over the pins, they display the distance from the answer and an arrow pointing from the pin to the answer. The arrow's length grows longer the farther away you are from the answer.

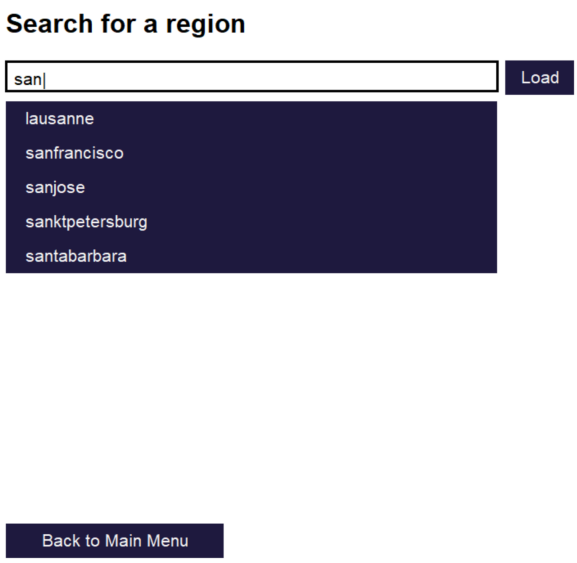


3️⃣ TP3 Update:

- I added a main menu and switched up the colors!



- If you click on "Load Data" in the main menu, I also added a section where you can load map data using pyrosm, complete with a search bar.



- Unfortunately, loading the data freezes up the app for a bit, and I don't have sufficient time to fix that problem; instead, I added some user interactivity in the python console.
- If you go to the main menu again, you can use the arrow keys to switch between different datasets that you've downloaded so you can play the game in different locations.
- I've successfully implemented a click and drag interaction! It's laggy, but it works. I didn't have the time to make the game detect double-clicks, so you just have to move your cursor to a specific location and press the spacebar. That can be a little glitchy at times because of how Tkinter's mouseMoved works.
- During the user testing on Tuesday, I realized that the game wasn't as intuitive to people as I thought, so I added a fun little tutorial for the first game.
- I've gotten rid of zooming and the settings menu. I don't have time to implement those.

Hope you enjoy the final product!!