April 10th, 2021

Finding the Perfect Park in Mississauga

AN ANDROID APPLICATION
GGR494H5: OPEN-SOURCE GEOSPATIAL ANALYSIS | DR. DAMIAN MADDALENA

BY: SIMON BEREST (1002523165)

Introduction

Finding the sports amenities of parks can be an inefficient endeavour. Some you discover as you wander your immediate surroundings, some, you discover on an invitation from a friend and some, you never discover. I have found, when deciding on a park to meet with friends, deciding on a centralized park is hard. Sometimes it is hard to know what the most centralized park is to try and balance travel times between an entire group. For this project I wanted to solve these two problems. I wanted to create a tool that could describe parks in the city, let the user filter by different amenities, and let the user find parks that are close to one or more user selected locations. I largely focused on sports amenities, those being baseball diamonds, basketball courts, bocce ball courts, cricket fields, soccer fields, softball fields, and tennis courts. I also included ice-rinks, play structures, and spray pads. The choice of these amenities was entirely due to availability of the data.

I initially intended to study Toronto, even though I found the city's dynamic map[1]. I knew I could create a better user experience than the cities tool. The Toronto map only allows filtering by one type of amenity at a time, and there is nothing about finding a park near you, or in between you and some friends. Unfortunately, the required data for Toronto was not available despite reaching out to the parks department. Turning my sights to Missisauga I also found a dynamic map[2], this one has some better features, as compared to Toronto's map, but also some worse features. I knew I could do better than that as well. I can add more tools for the user to find a perfect park. I have created a similar[3] tool on Carto before but I wanted to add more functionality that Carto does not provide.

Methods

Before I could start working on the tool, I had to do some data munging. The data is available on the Missisauga Open Data Portal[4] in multiple pieces. Predictably that means my study site is confined by the municipal boundaries of Missisauga. The most important piece of data takes the form of a parks point shapefile. The rest of the data takes the form of individual points shapefiles. One for each sports amenity.

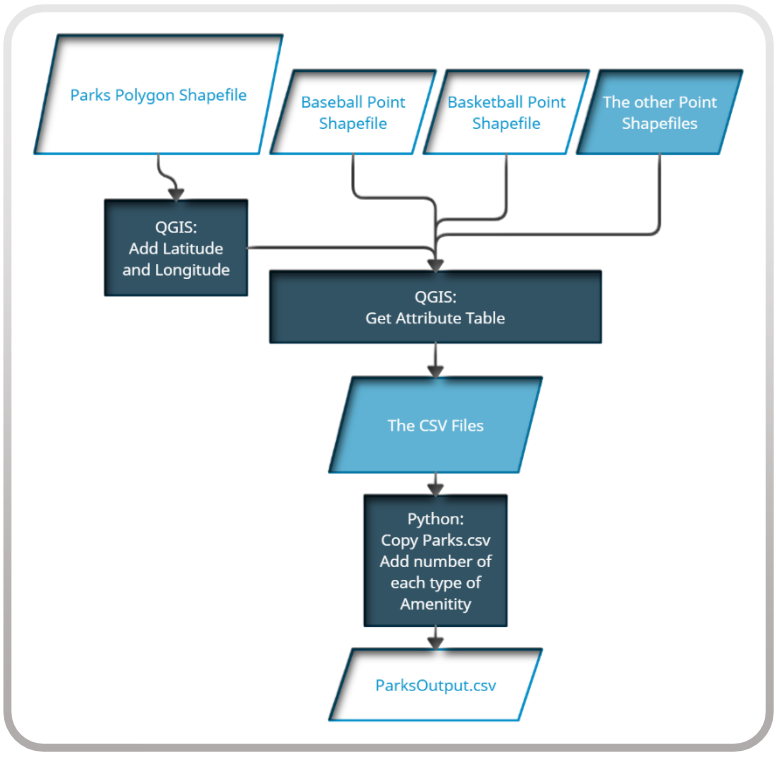

Figure 1. Data Munging Workflow

First, I added latitude and longitude to the parks polygon shapefile using QGIS, and made new columns with the names latitude and longitude as the given x and y locational values were not accurate. I then saved each of the sports amenity shapefiles as well as the parks shape file as a csv file. To create the final CSV I needed to have columns with values that represent what sports amenity that park has. Each column representing one sports amenity with the number representing the amount of that specific amenity a given park had.

With this singular CSV file I was now ready to code the android application. I used Android Studio which is an open-source integrated development environment to write the code, handle the files, and to emulate the phone for testing.

Displaying the map and the parks

To display the map I used the java libraries OSMdroid and OSM Bonus Pack. These libraries are open-source replacements for google-map functionalities. They have built-in tiles for displaying locations at various zoom levels, and built-in marker functionality. The built-in markers have functionality for having custom icons, and for displaying specific information when tapped. I made it so that the parks displayed their name and their number of amenities when tapped.

Clustering of parks

It turns out there are four-hundred and ninety-six parks in the Missisauga Boundaries. This does include 78 not-yet or not-to-be named parks. As such when viewing the entirety of Missisauga there are overlapping icons. To solve this problem I implemented clustering which is

a built-in functionality of OSMBonusPack. The clustering is done such that no park marker is one-hundred pixels from any other. The clusters change depending on the zoom level.

Filtering by amenity type(s)

To filter by amenity I simply created an array to represent how many of each amenity the user has selected to filter by. I then iterated through all the parks and added a park if it had at least as many sports amenity as the user requested.

Displaying user selected locations

I implemented a way for users to select a point on the map. I made it such that when a button is clicked the user can add up to four user locations. The street names of the user locations are found using the Open Street Maps Nominatim[5]. To identify the user locations to the user when they are not seeing the full map.

Filtering by distance from user selected location(s)

To filter by distance was simple. For each user location the user inputs they can select if they want to filter by distance. The default distance is 0.75 kilometres. This was a deliberately chosen value as the US department of Transportation National Survey of Pedestrians found that on average the distance of a walk was 1.3 miles[6]. This distance was for a full walk accounting for the shape of the streets. Thus, I decided that a distance of 0.75 kilometres would be an easy walk for most people so that they still had energy to play sports when they got to their chosen park. After the user selects these factors, I iterate through each park, keeping track of what parks are at most the respective inputted distance away from each user location using Euclidean distance calculation.

Filtering by part of a polygon

To filter as a part of a polygon I used an interesting property of math. For this functionality the user would place user selected location, these locations would then become the vertices of a polygon. For each park, I check how many times a line, that would pass through at least one side of the polygon, passes through any sides of the polygon. If it passes through the polygon an odd number of times, then the point is inside the polygon. If the line passes through the sides of the polygon an odd number of times, then the point is outside the polygon.

Results and discussion

Displaying the map, the parks, and clustering the parks.

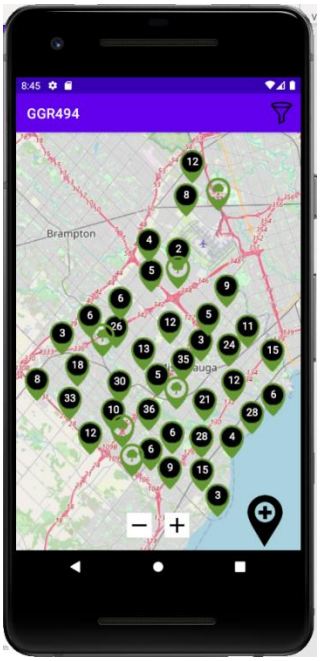

Figure 2. All the parks in Missisauga as displayed in my android app.

The figure on the left is the first page the user sees when the app is launched. Parks are clustered to reduce cluttering on the screen.

In the top right there is the filter button, which brings the user to the filter page. In the bottom right is the user location button. This will be expanded upon later.

The figure on the right is the park information popup. When a singular park marker is tapped the map centers that park in the display and displays this popup which describes the park name and the variety of amenities it has.

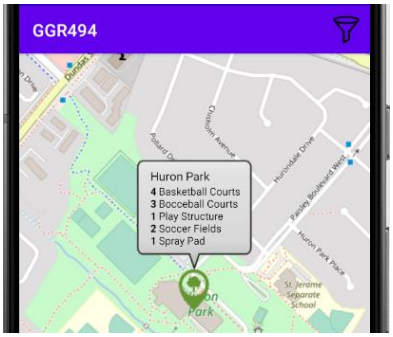

Figure 3. Displaying Huron Park information achieved by clicking on Huron Park.

At the very bottom of the display are the zoom controls. Tapping the minus zooms out, tapping on the plus it zooms in.

Filtering by amenity type

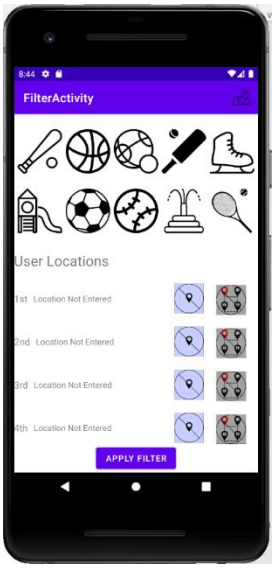

Figure 4. The Filter Screen.

The image on the left is the filter page. The user locations have not been set and as such the related filtering buttons are crossed out.

The paired image on the top right represents the user filtering to all parks that have at least one basketball court and the resulting map when the apply filter is tapped.

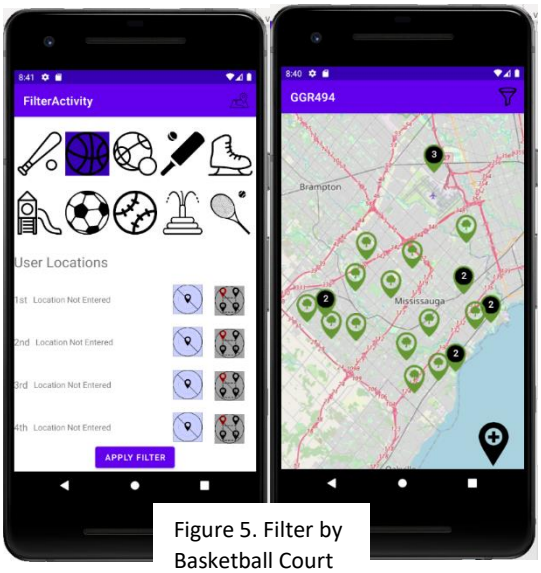

Figure 5. Filter by Basketball Court

Filtering by multiple instances of an amenity

To implement multiplle filtering I added the behaviour to open a seek bar that pops up when an amenity filter is long tapped. The two images on the right display this functionality, only Paul Coffey Park has eight basketball courts.

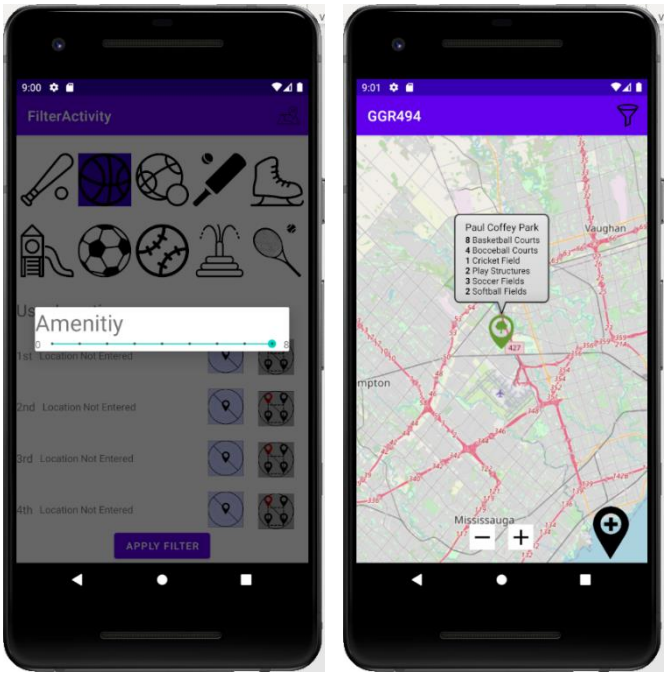

Figure 6. Filtering to all parks that have 8 basketball courts. Only Paul Coffey Park applies.

Filtering by multiple types of a amenities

The image on the right displays the functionality of filtering by multiple types of amenities. It turns out there is only two parks in Missisauga with both a basketball court and a bocce ball court. The first is Paul Coffey Park which was seen in the previous functionality, the second is Huron Park.

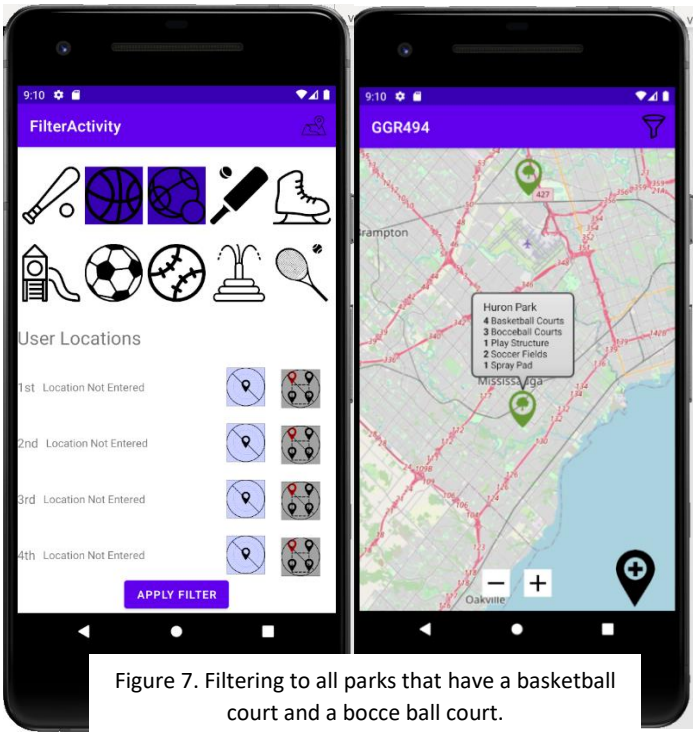

Figure 7. Filtering to all parks that have a basketball court and a bocce ball court.

Displaying user selected locations

As stated earlier the button in the bottom right is the user location button. When it is tapped once it switches to the adding a user location funcitonality. Here I tapped on the cul-de-sac at the end of Irwin Court, The user location street name is then found through geocoding.

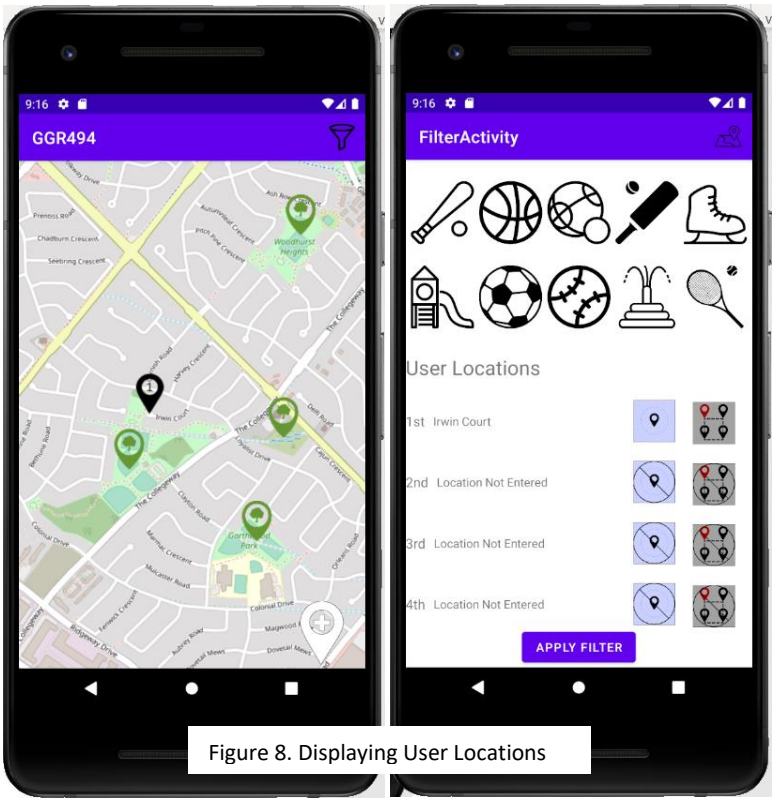

Figure 8. Displaying User Locations

Filtering by a default distance

Using the same location as the previous functionality the user filters by the default distance of 0.75 km from Irwin Court. As you can see that there are three parks within that distance from the Irwin Court cul-de-sac

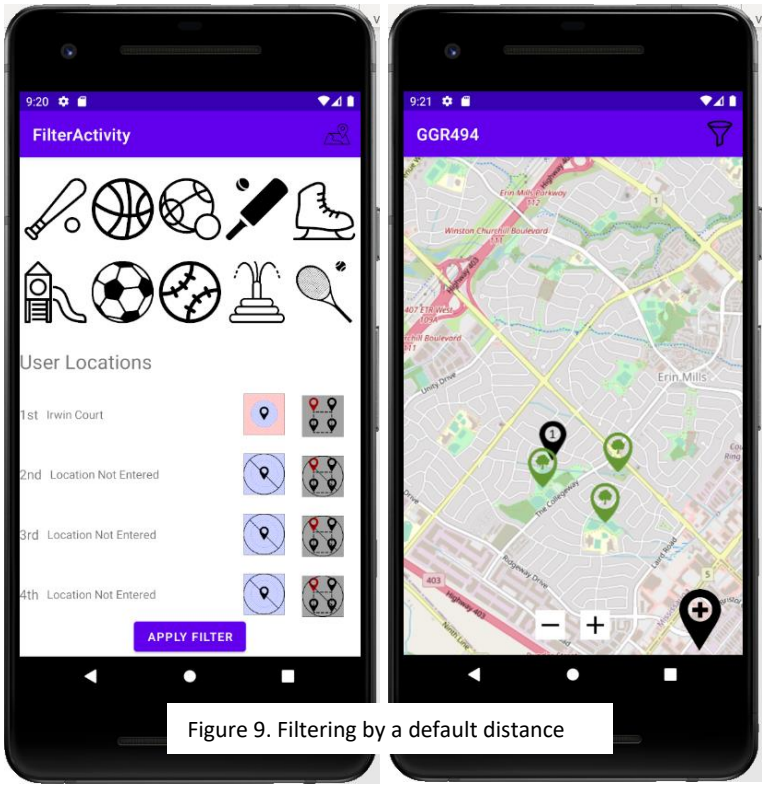

Figure 9. Filtering by a default distance

Filtering by a user given distance.

Using the same location as the previous functionalities the first image shows the popup. Here the user inputs 3 for a three-kilometre range. The second image shows that the parks are filtered down to a three-kilometre radius.

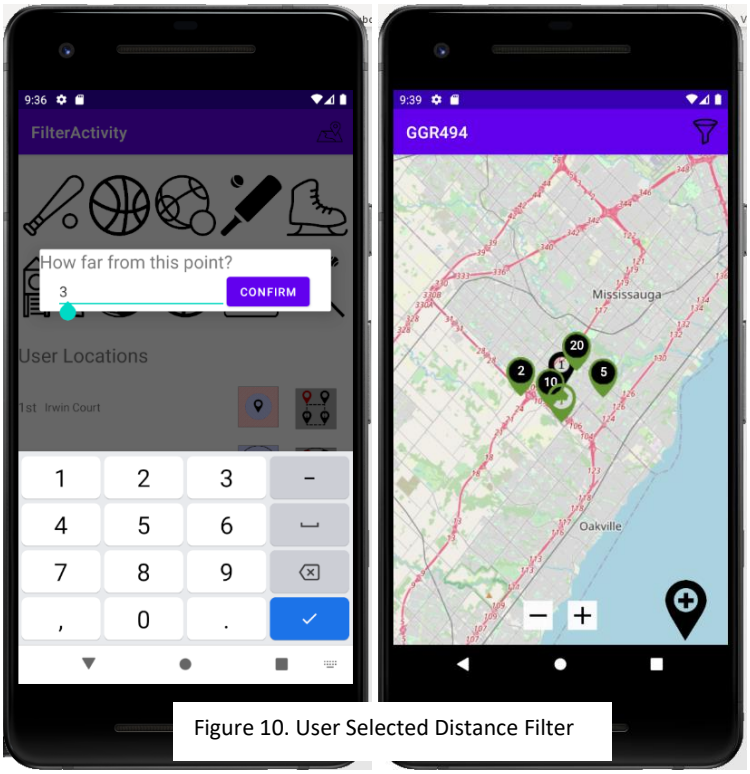

Figure 10. User Selected Distance Filter

Filtering as part of a polygon.

This functionality requires the user to place more than one location. The first image shows the user having placed four locations. The second shows the user having enabled the "filter as part of a polygon" button. The third image shows that the parks were limited to only be the parks within the created polygon.

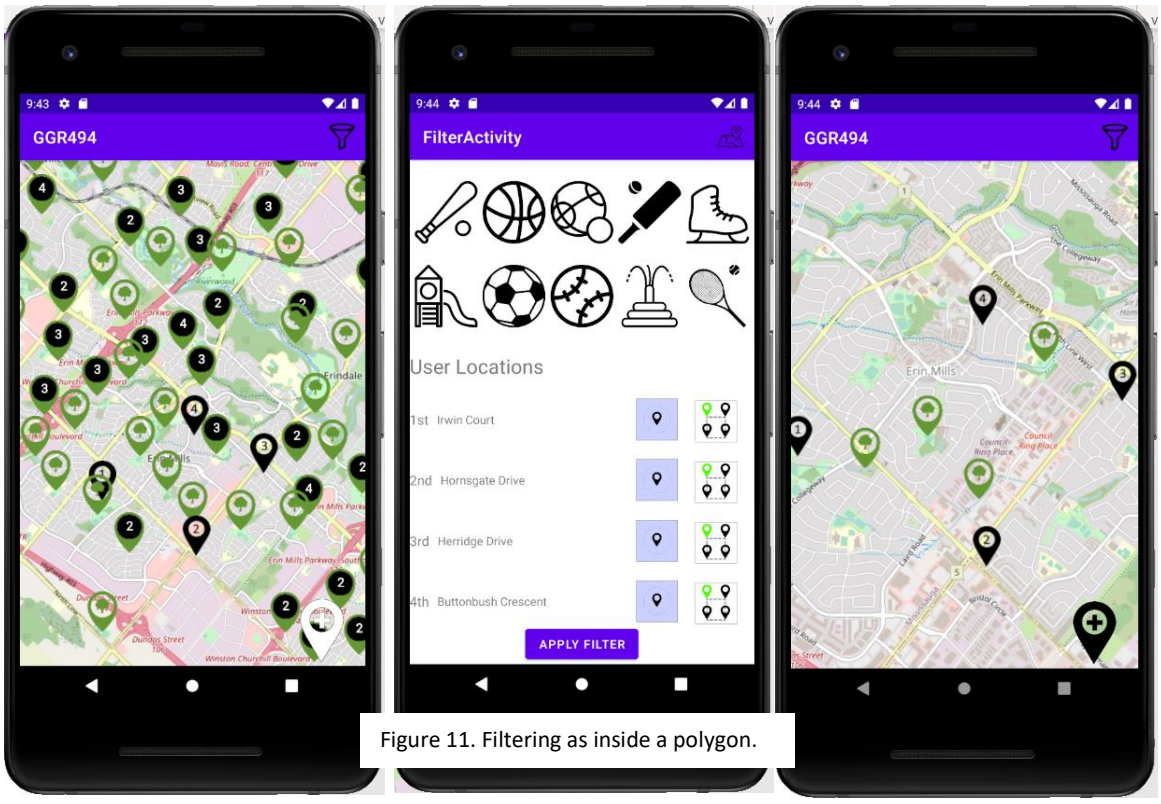

Figure 11. Filtering as inside a polygon.

Combination of filters.

Finally, I will demonstrate the functionality of combining the filters. There are almost infinite amounts of ways to combine the different filtering types especially given different combination of user locations. Here the user shows off the functionality of filtering by play structure and as inside a polygon. As you can see two of the previous four parks have play structures.

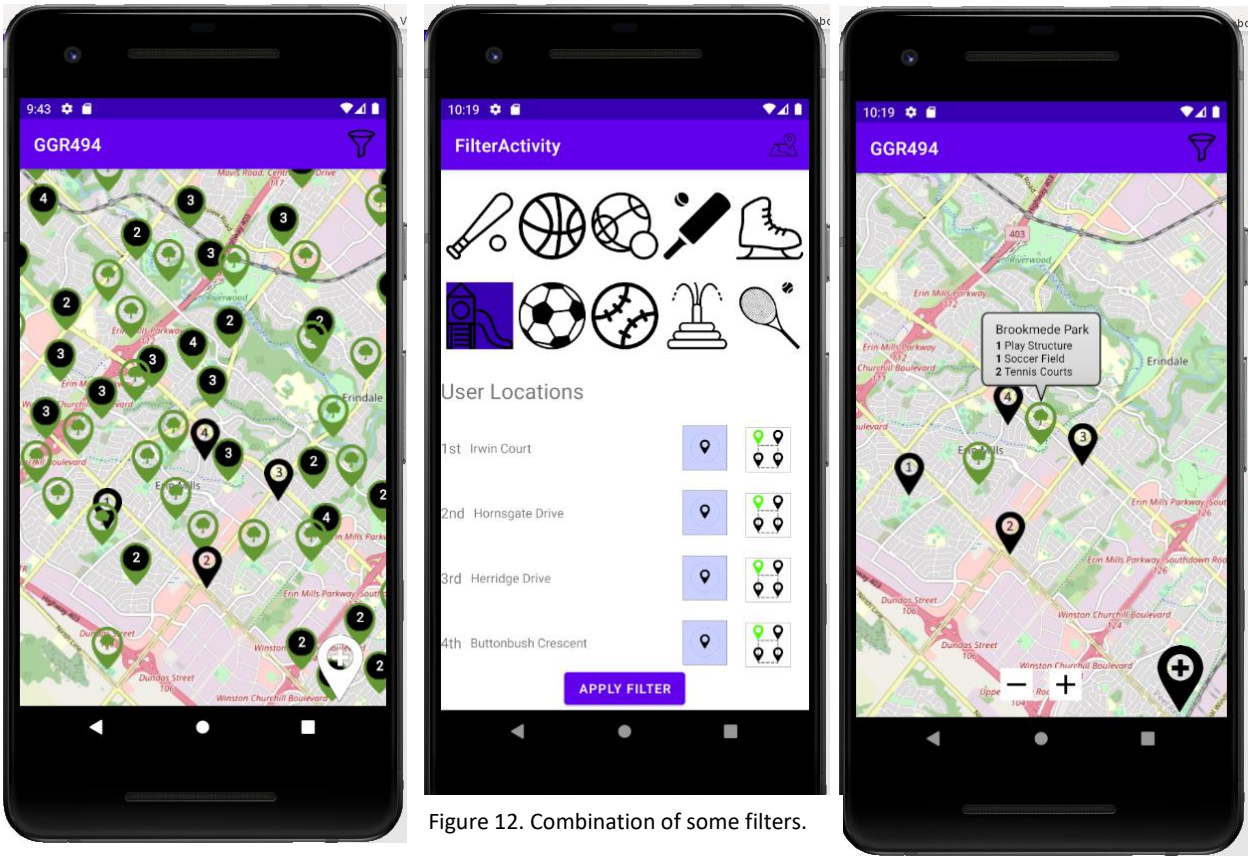

Figure 12. Combination of some filters.

Conclusion

The application is technically complete now, all my planned functionalities either work, or a suitable alternative was found that expand on the use case for the user. That said there are still changes I would make. On the visual side I would change the filter page to be more visually appealing, the symbol icons are not obviously buttons, and the text is not an easy-to-read colour. I would add some form of in app help feature that would tell the user what different buttons do. I would display the user location markers more prominently so that they are more obviously different than the park markers, I would add a reset filter button on the main map page. I would add more amenities as they are released by the city of Missisauga, and I would add more filtering capabilities for lighting which is included in the data. As they say, “perfect is the enemy of good.” As such I am happy with the results.

References

1. City of Toronto. “Parks Map.” City of Toronto, March 6, 2017. Accessed February 11, 2021. https://www.toronto.ca/data/parks/maps/index.html.
2. City of Missisauga. “Parks in the CIty of Missisauga.” eParks. Accessed April 12, 2021. http://www.eparks.ca/.
3. Berest, Simon. “Location of Sports Infrastructure in Parks in Missisauga.” CARTO, December 2020. Accessed February 11, 2021. https://sberest.carto.com/builder/f3a0f4fb-a971-41b4-93ba-cfb33e46dd61/embed.
4. City of Mississauga. City of Mississauga Open Data Catalogue. Accessed February 11, 2021. https://data.mississauga.ca/.
5. OpenStreetMap Contributors. Nominatim Demo. Open Street Map. Accessed April 12, 2021. https://nominatim.openstreetmap.org/ui/search.html.
6. U.S. Department of Transportation's National Highway Traffic Safety Administration and the Bureau of Transportation Statistics (2002). National Survey of Pedestrian and Bicyclist Attitudes and Behaviors. Accessed Freburary 10, 2021. http://www.nhtsa.gov/DOT/NHTSA/Traffic%20Injury%20Control/Articles/Associated%20Files/810971.pdf

Appendices

Workflow Documentation

Data Munging Workflow is figure 1.

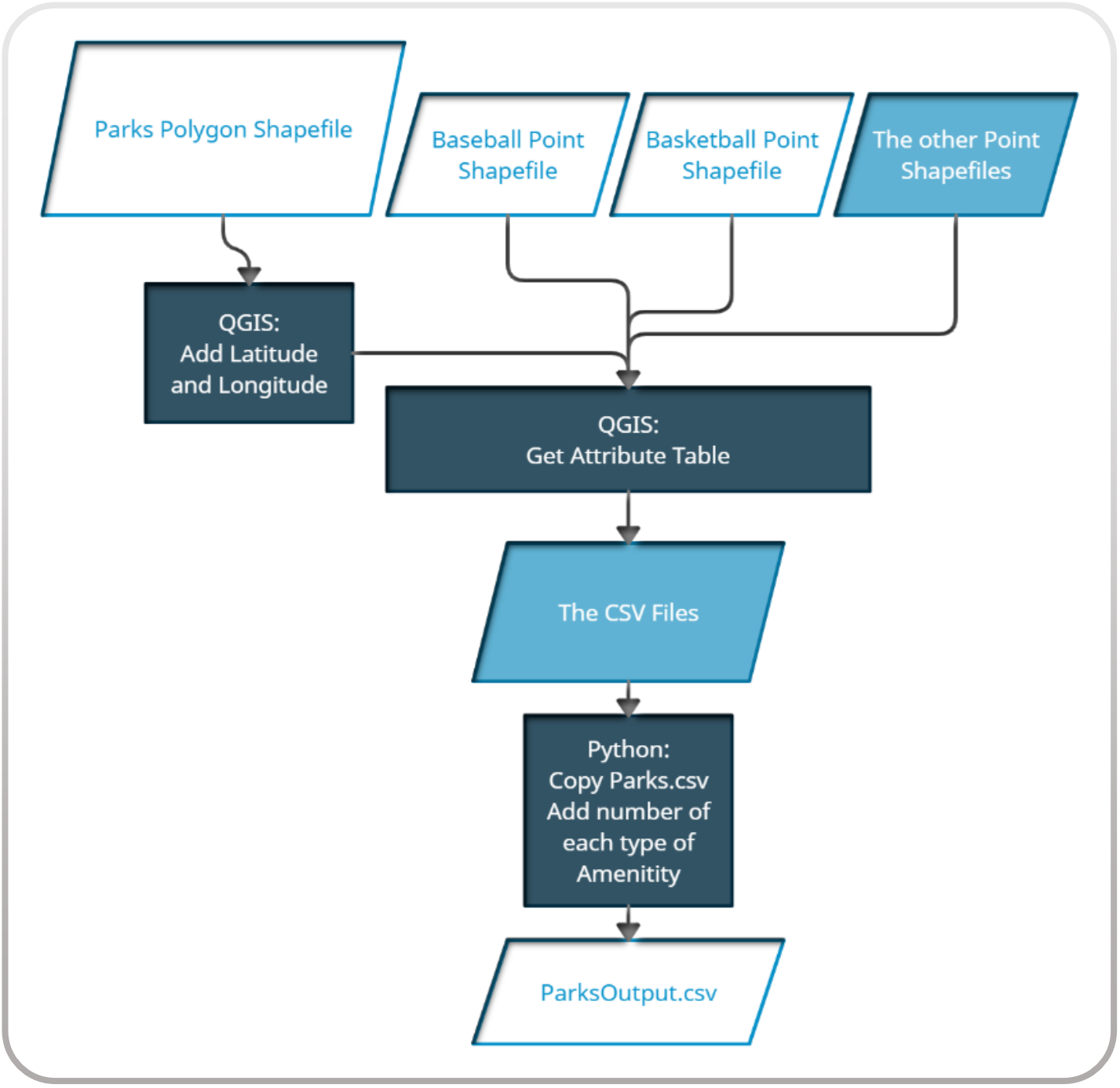

Figure 1. Data Munging Workflow

The python code to extend the parks.csv.

```
import csv
#Manually set filenames and column/field names for latter
filesToDictionary =
['Baseball_Diamonds.csv','Basketball_Courts.csv','Bocce_Courts.csv','Cricket_Pitches.csv','Municpal_Pools.csv','Natural_Ice_Rinks.csv','Play_Sites.csv','Soccer_Fields.csv','Softball_Diamonds.csv','Spray_Pads.csv','Tennis_Courts.csv']
colNames =
['nBaseball','nBasket','nBocce','nCricket','nPool','nIceRinks','nPlaySites','nSoccer','nSoftball','nSprayPads','nTennis']

#read in Parks
f_in = open('Parks.csv','r')
parks_reader = csv.reader(f_in)

#Set up dictionary of park id to list of 11 values for amount then reset the reader
parkDict = {}
for row in parks_reader:
        parkDict[row[13]] = [0,0,0,0,0,0,0,0,0,0,0]
f_in.seek(0)

#Fill out the dictionary based on the files inputted
i = 0
for path in filesToDictionary:
        f = open(path)
        csv_reader = csv.reader(f)

        for row in csv_reader:
                if row[-3] != 'PARENTID':
                        if row[-3] in parkDict:
                                parkDict[row[-3]][i] = parkDict[row[-3]][i] + 1
                        else:
                                print(f"{path}:{row[-3]} not found")
        i = i + 1
        f.close()

#Set up our output file
f_out = open("ParksOut.csv",'w')
```

```
csv_writer = csv.writer(f_out)

headerFlag = 1
for row in parks_reader:
        if headerFlag == 1:
                row.extend(colNames)
                print(row)
                headerFlag = 0
                csv_writer.writerow(row)
        else:
                row.extend(parkDict[row[13]])
                print(row)
                csv_writer.writerow(row)
```

Code Block A1. Python code I used to add columns representing how many of each type of an amenity a park had.

The Source Code

The full source Code is located at https://github.com/SBerest/GGR494/tree/master

Meta Data

File Name	Description	Type	Projection	Extent	Source
City Parks	Point data of parks including name, address, locational data, park id, and more.	Points Shape File	WGS84	City of Missisauga	Open Data Missisauga
City Baseball Diamonds	Point data of baseball diamonds including the park id that it is in, locational data, and more.	Points Shape File	WGS84	City of Missisauga	Open Data Missisauga
City Basketball Courts	Point data of basketball courts including the park id that it is in, locational data, and more.	Points Shape File	WGS84	City of Missisauga	Open Data Missisauga
City Bocce Courts	Point data of bocce courts including the park id that it is in, locational data, and more.	Points Shape File	WGS84	City of Missisauga	Open Data Missisauga
City Cricket Pitches	Point data of cricket pitches including the park id that it is in, locational data, and more.	Points Shape File	WGS84	City of Missisauga	Open Data Missisauga
City Natural Ice Rinks	Point data of natural ice rinks including the park id that it is in, locational data, and more.	Points Shape File	WGS84	City of Missisauga	Open Data Missisauga
City Play Sites	Point data of play sites including the park id that it is in, locational data, and more.	Points Shape File	WGS84	City of Missisauga	Open Data Missisauga

City Soccer Fields	Point data of soccer fields including the park id that it is in, locational data, and more.	Points Shape File	WGS84	City of Missisauga	Open Data Missisauga
City Softball Diamonds	Point data of softball diamonds including the park id that it is in, locational data, and more.	Points Shape File	WGS84	City of Missisauga	Open Data Missisauga
City Spray Pads	Point data of spray pads including the park id that it is in, locational data, and more.	Points Shape File	WGS84	City of Missisauga	Open Data Missisauga
City Tennis Courts	Point data of tennis courts including the park id that it is in, locational data, and more.	Points Shape File	WGS84	City of Missisauga	Open Data Missisauga

Table A1. Meta data of all the pieces of data used.

Proof of Open Source

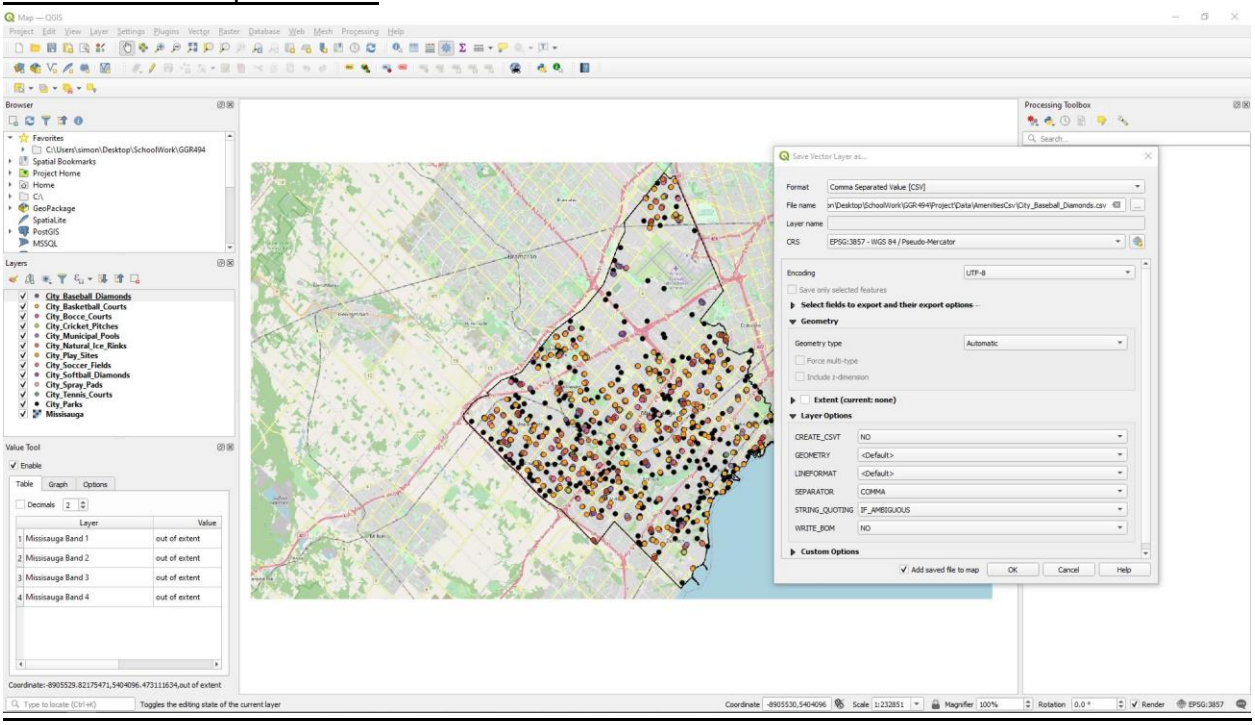

Figure A1. Proof I used QGIS to extract the attribute tables from the points data files.

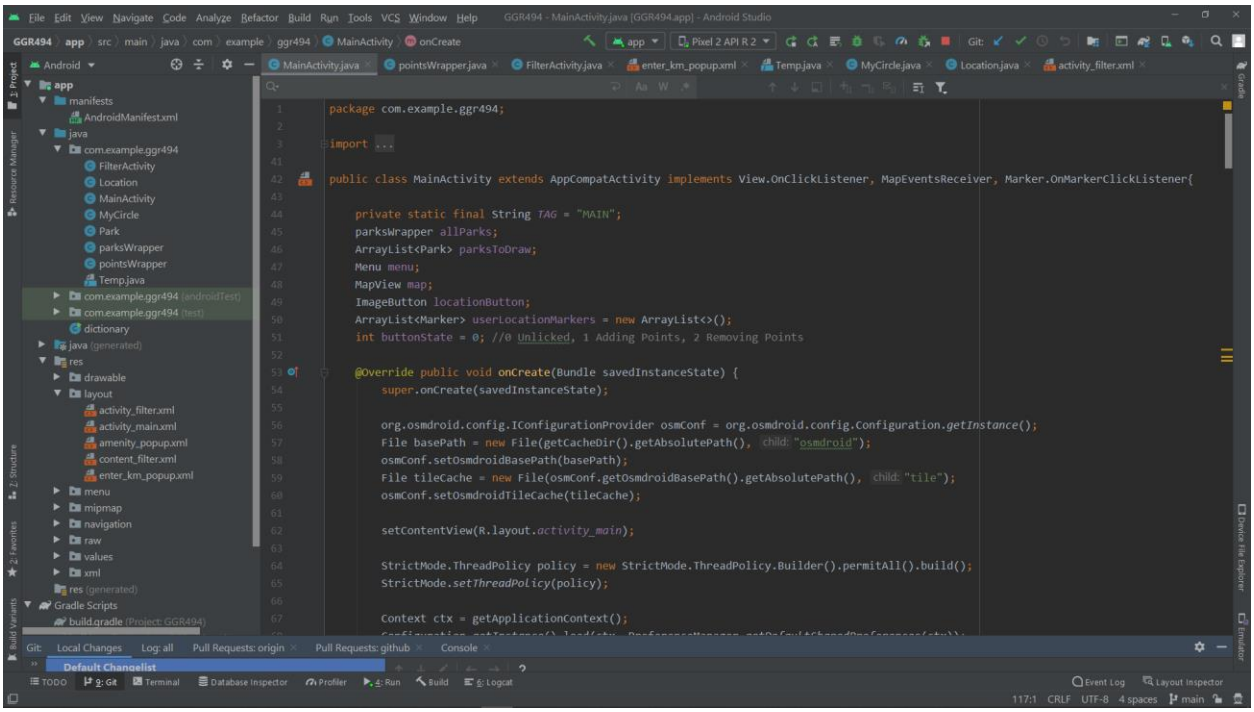

Figure A2. Screen Shot of some of the source code in Android Studio, along with the file layouts.