CSc 317
Maps

Benjamin Dicken

Announcements

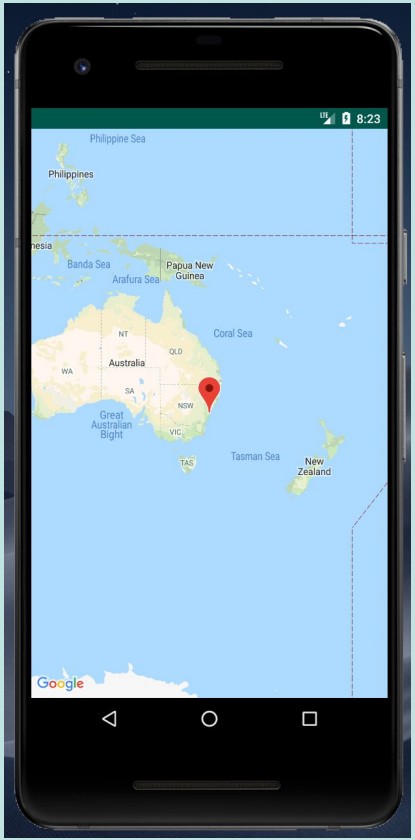
- The SCS
  - If response rate >= 80%, will drop a PA
  - 39.53% as of this morning
  - ***Please leave thorough, honest feedback!***
- Project check-in meetings on Monday

Android Maps API

- An Android API that allows you to utilize google maps info within your app!
  - “The API automatically handles access to Google Maps servers, data downloading, map display, and response to map gestures.”
- You can customize by . . .
  - Adding markers, zooming, moving the view, adding lines, polygons, etc

Maps and Location

- Create a new map project
- Download Google Play Services API, via Android Studio preferences
- Get an API key
- Add API key to XML file
- Run the application

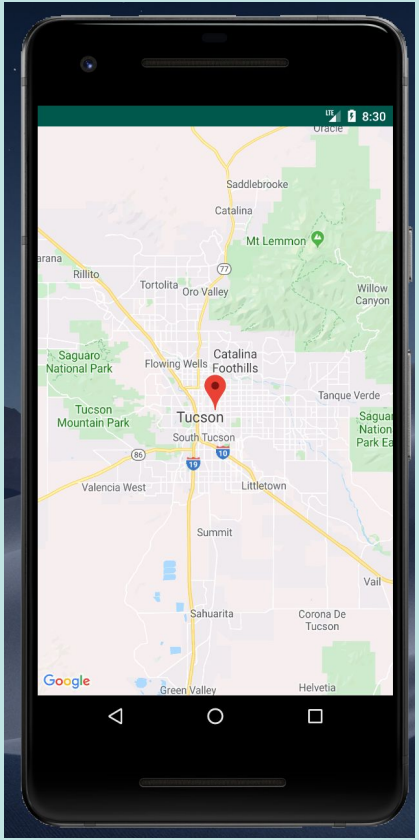


The GoogleMaps object

- To use a map in your application, use the **SupportMapFragment** class
  - Add the fragment to the activity of your choice
- Once the fragment is ready, update the map using the **GoogleMap** object

Move the marker, view

- Go to **MainActivity**, and edit **onMapReady**
- Change the code so that the marker and view show the location of Tucson, rather than Sydney
  - How would you figure that out?
- Also, try to make it more zoomed in to tucson, without the user having to manually zoom in

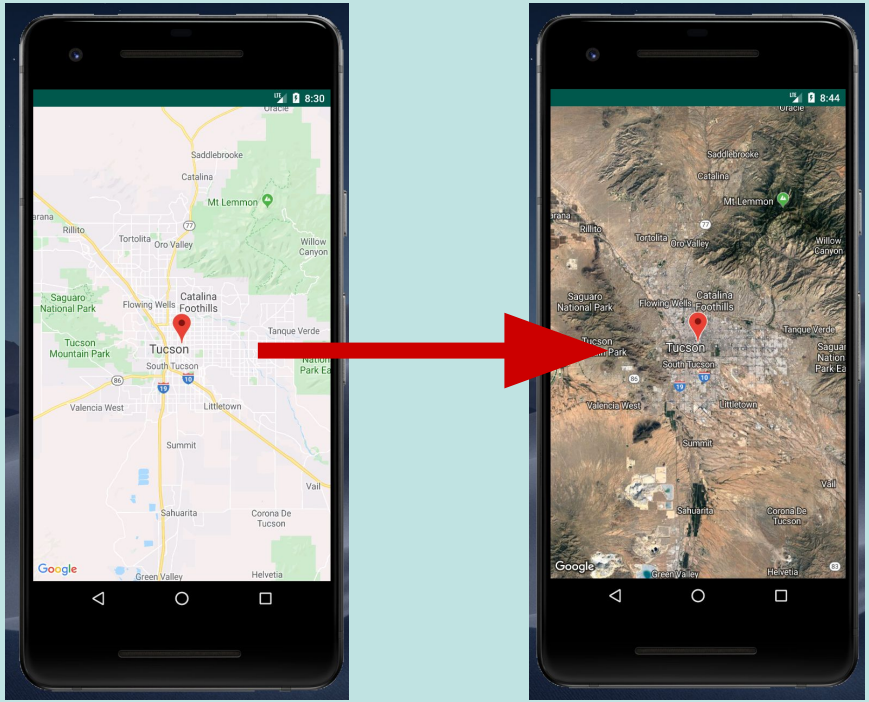


Setting the map type

- **`GoogleMap.MAP_TYPE_NORMAL`** - Display a “normal” looking google map
- **`GoogleMap.MAP_TYPE_SATELLITE`** - Display satellite imagery
- **`GoogleMap.MAP_TYPE_HYBRID`** - Display satellite imagery with additional information included
- . . .

Show satellite imagery

- Change the view so that it shows actual satellite imagery



Placing markers

```
LatLng location = new LatLng(latitude, longitude);
Marker m = googleMap.addMarker(
        new MarkerOptions()
            .position(location)
            .title(someTitleString));
m.setVisible(true);
m.showInfoWindow();
```

Add markers

- Increase the zoom to show the UofA, and make sure **Tucson** marker is within UofA borders
- Add markers, with titles, for **Old Main** and **McKale Center**

```
LatLng location = new LatLng? ?);
Marker m = googleMap.addMarker(
        new MarkerOptions()
            .position(location)
            .title(?));
m.setVisible(true);
m.showInfoWindow();
```

Drawing polygons (shapes)

```
Polygon polygon = googleMap.addPolygon(new PolygonOptions()
        .add(new LatLng(?, ?),
             new LatLng(?, ?),
             new LatLng(?, ?)));
polygon.setFillColor(0x4400ee55);
```

Add a polygon for UofA

- Add a green polygon around UofA
- Use google maps to figure out the lat/long

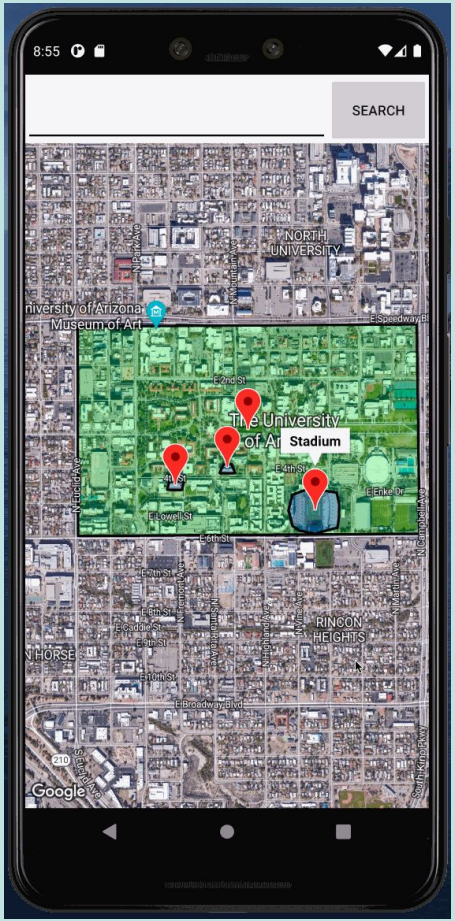
```
Polygon polygon = googleMap.addPolygon(
    new PolygonOptions()
        .add(new LatLng(?, ?),
            . . .
              new LatLng(?, ?)));
polygon.setFillColor(0x4400ee55);
```

Other Features and APIs

- Go to https://console.developers.google.com
- Can change settings, add other APIs
  - Places API
  - Directions API
  - Even other non-map APIs

The goal

- Reads in location info from a file named polygons.txt
  - Places polygons and markers at those locations
- Allows the user to search for places near those locations



Places API search results


```
{  "html_attributions" : [],
   "next_page_token" : "...???...",
   "results" : [
      { "geometry" : {
            "location" : {
               "lat" : 32.2352107,
               "lng" : -110.9569159
            }, … }
         },
         "icon" : "https://maps.gstatic.com/mapfiles/place_api/icons/cafe-71.png",
         "id" : "74c90d7eca76a864bdc00cd4081cfb2278eb6c99",
         "name" : "Teaholic",
   ],
   . . .
```

Write the code in

- First, **DrawLocations**
- Then, **getAverage**
- Then, **onPostExecute**

Getting user Location

- Use a **LocationListener** object
- Implement required callbacks
  - To be notified when the device location changes, override the **onLocationChanged** function

```
private final LocationListener mLocationListener = new LocationListener() {
    @Override
    public void onLocationChanged(final Location location) {
        double currLat = location.getLatitude();
        double currLng = location.getLongitude();
        // Do things with the lat and lng
    }

    @Override
    public void onStatusChanged(String s, int i, Bundle bundle) { }
    @Override
    public void onProviderEnabled(String s) { }
    @Override
    public void onProviderDisabled(String s) { }
};
```

```
mLocationManager.requestLocationUpdates(
        LocationManager.GPS_PROVIDER,
        100 /*LOCATION_REFRESH_TIME*/,
        1 /*LOCATION_REFRESH_DISTANCE*/,
        mLocationListener);
```

The goal

- Create an application that tracks a user's path
- Draws a merker when it gets a location update, and a path from last known location to updated location
- Generate your own key and place in **`google_maps_api.xml`**
- Use the **PathTracker** starter code