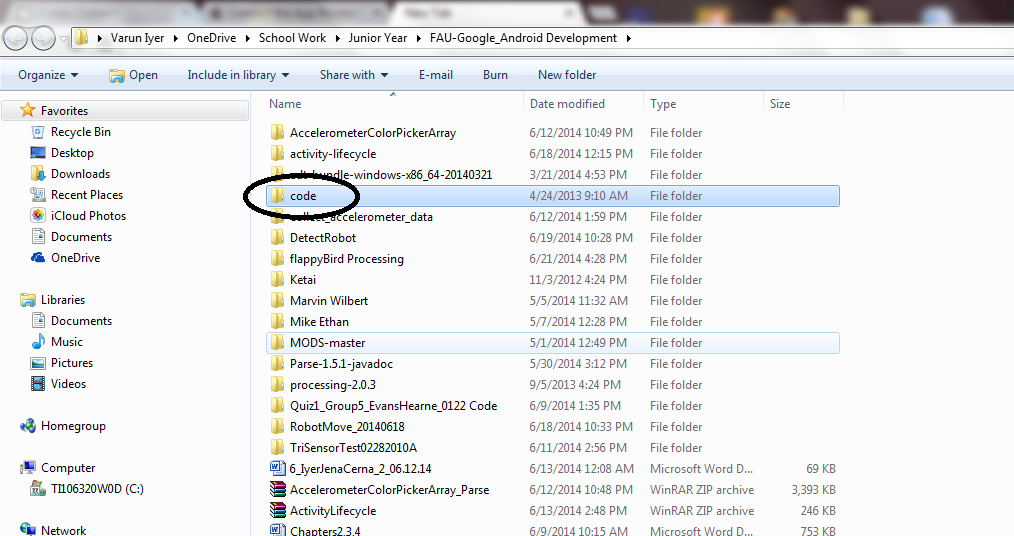
**Tutorial by Varun Iyer**

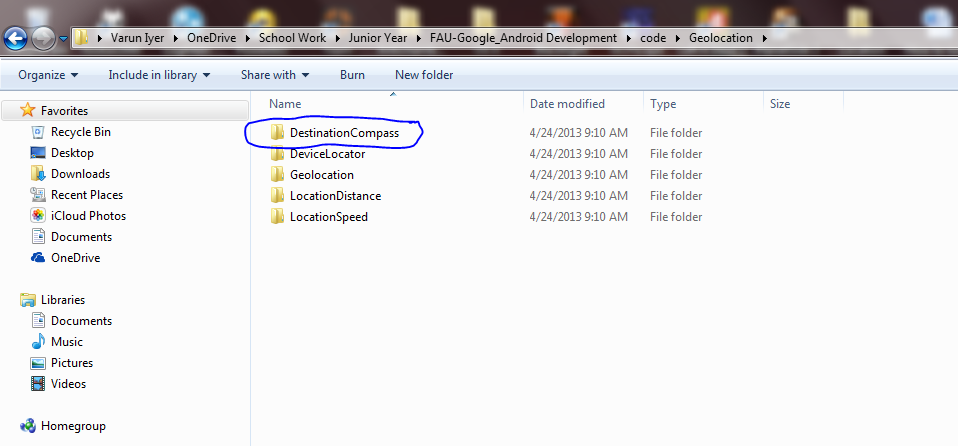
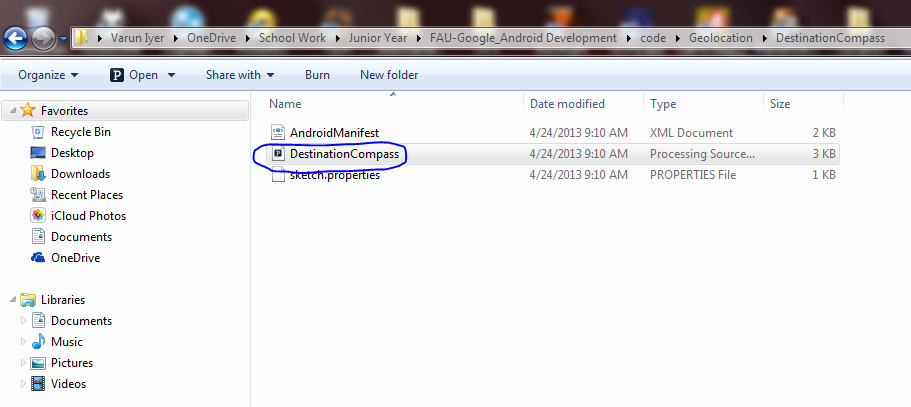
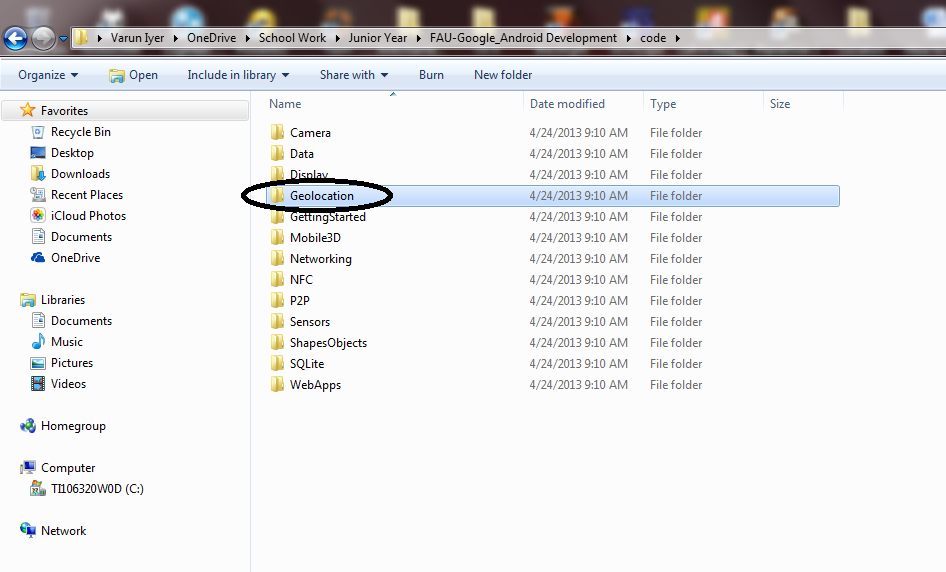
How to turn your nexus into a navigator!

GPS Location is a technology that has been in development since the 20th century. One of its many useful applications is using your device as a GPS to navigate to exotic locations. It has been a useful feature used by many people. In this tutorial you will learn how to turn your android device into a navigator using GPS technology. You will be given instant directions to the location that you desire and will be navigated towards it.

**Step 1:** Download the folder from this site: <https://www.dropbox.com/s/9nh1sc6csq67ykp/code.zip>

Or, if you have the “code” folder you downloaded from **Blackboard**, go to **code\GeoLocation\DestinationCompass** and open the processing file.

I have also posted detailed steps for opening it:



**Step 2:** View the Code

For the purposes of this tutorial, I will briefly cover the code written in this file to aid your understanding in how this code works. If you have any questions, feel free to contact me at vi0210@yahoo.com.

The descriptions for some of the methods in the code are written as comments. If you wish to understand the functionality of this code, it is requested that you look at it.

import ketai.sensors.\*;

import android.location.Location;

KetaiLocation location;

KetaiSensor sensor;

Location destination; //Location object that stores your destination

PVector locationVector = new PVector();

float compass; //This creates a compass variable to store the rotation around the z-axis

void setup() {

destination = new Location("uic");

destination.setLatitude(41.824698); //Latitude coordinate of your destination

destination.setLongitude(-87.658777); //Longitude coordinate of your destination

location = new KetaiLocation(this);

sensor = new KetaiSensor(this);

sensor.start();

orientation(PORTRAIT);

textAlign(CENTER, CENTER);

textSize(28);

smooth();

}

void draw() {

background(78, 93, 75);

float bearing = location.getLocation().bearingTo(destination); //returns the direction of the destination pointer

float distance = location.getLocation().distanceTo(destination);

if (mousePressed) {

if (location.getProvider() == "none")

text("Location data is unavailable. \n" +

"Please check your location settings.", 0, 0, width, height);

else

text("Location:\n" +

"Latitude: " + locationVector.x + "\n" +

"Longitude: " + locationVector.y + "\n" +

"Compass: "+ round(compass) + " deg.\n" +

"Destination:\n" +

"Bearing: " + bearing + "\n" +

"Distance: "+ distance + " m\n" +

"Provider: " + location.getProvider(), 20, 0, width, height);

}

else {

translate(width/2, height/2); //(3)

rotate(radians(bearing) - radians(compass)); //rotates the triangle at an angle relative to orientation of the screen.

stroke(255);

triangle(-width/4, 0, width/4, 0, 0, -width/2);

text((int)distance + " m", 0, 50);

text(nf(distance\*0.000621, 0, 2) + " miles", 0, 100); //converts the distance from meters to miles, rounding to the hundredths place (2 decimals places after the 0)

}

}

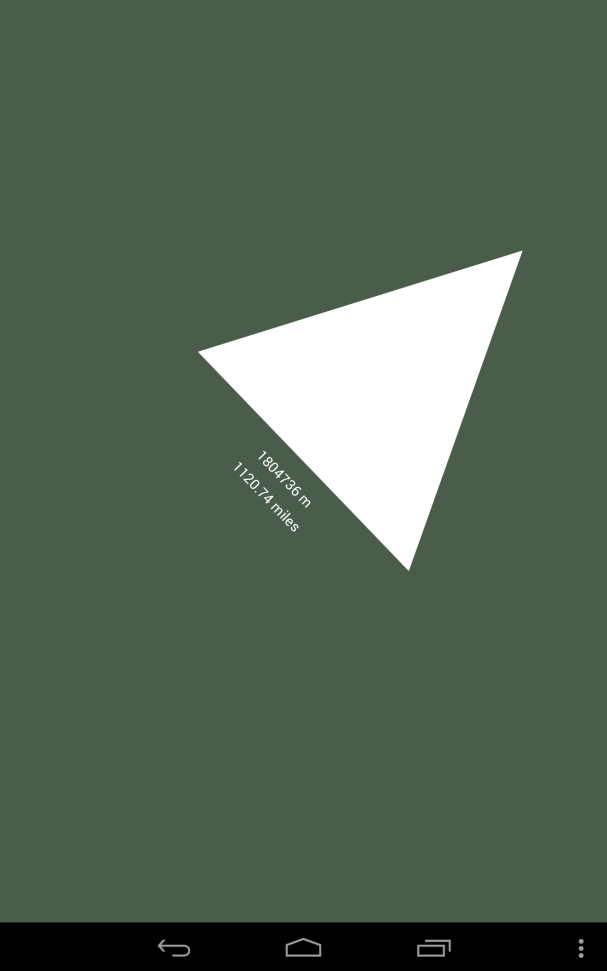
void onLocationEvent(Location \_location) {

println("onLocation event: " + \_location.toString());

locationVector.x = (float)\_location.getLatitude(); //Stores the device latitude

locationVector.y = (float)\_location.getLongitude(); //Stores the device latitude

}

void onOrientationEvent(float x, float y, float z, long time, int accuracy) { //returns the x, y, z and z values from the bearing.

compass = x;

// Azimuth angle between magnetic north and device y-axis, around z-axis.

// Range: 0 to 359 degrees

// 0=North, 90=East, 180=South, 270=West

}

**If you would like further clarification on this material, please read section 4.8 (pg 83-87) in the book “Rapid Android Development.”**

**Step 3:** Run the Project!

Once you run the project, you should get a screenshot similar to this:

Also please note that in order to keep the directions as accurate as possible, make sure you keep your device as level as possible.

Tilting the device up and down changes the direction of the triangle.



If you tap (mouse press), you should be seeing this as well:

**Step 4:** Where do we go from here?

After doing this tutorial, you should have a firm understanding of how to use the Ketai library to grab the device’s location and generate navigation using a given location from its latitude and longitude points. This feature can have many hopeful and interesting applications, and I hope you will take what you have learned here and apply it to future projects. I wish you all good luck in the future.

As I stated before, if you have any questions, comments or concerns, please feel free to contact me at [vi0210@yahoo.com](mailto:vi0210@yahoo.com). Thank you for taking the time to read and understand this tutorial.