The program contains a live GPS of the drone carrying out a task (for every 1000ms of runtime, 1 hour passes in real-time). The drone's location was calculated using different parameters that are found in day-to-day life. This simulation was not part of the bonuses and was a significant challenge to execute. I had a few conceptual ideas as to how a drone could be presented to the end-user:

***Using a 2x2 matrix.***

By using a cartesian coordinate system and assigning values to each “pixel”, the program would be able to color in the pixel which corresponds to the location of the drone. To do this, assign arbitrarily one of the corner’s to be (0,0), the diagonal being (width, height) and do the same for the picture of a real map. The map would have to be the same size and a conversion between map coordinates to matrix coordinates would take place. This was much harder to do and would require a large matrix for accuracy and some way to place a marker on the map (while also drawing the map in the viewport window).

***Google API (would require payment)***

***Pinpoint location with OSM***

*OpenStreetMaps* is open-source and free. This took me some time to figure out, but finally I had the idea once I noticed the URL of the website. The address bar for OSM shows the latitude and longitude of the pin in the map, in turn being extremely useful.

For example:

[*https://www.openstreetmap.org/#map=14/31.7723/35.2181*](https://www.openstreetmap.org/#map=14/31.7723/35.2181)

By using this (with the addition of the pin), the program can send queries to the site to constantly update the map. To view the URL, I used WebView2 which allows for integrating a browser inside a window. I also tweaked some settings which made the browser seem like an actual viewport (not vertical/horizontal scrolling, zoom, scaling, etc.). The only issue is the constant split-second of white between each update (strobing).

**UI/UX Design**

The program underwent many changes, especially with regards to user interface and program design. One of the first iterations of this was a simple *view-box* with some Windows XP style buttons. The current iteration features rounded corners on most objects (including the window), color schemes, custom fonts, better scaling when resizing windows, overall design scheme and more.

**Faux-Chat**

In an effort to parody a life-like imitation of a real “drone company”, I added a fake chat that would allow people to interact between themselves (since the program is more of an *amazonized* eBay, since products are sent between to users; amazon would be more of an online store). I would have also loved to add a homepage with all products up for sale which would allow people to buy things they choose and notify the owner to send that person the item (one purchase is complete).

**Big-Picture Map**

This continues off of the GPS I mentioned above, but is not “real-time” but more static (can be updated after one of the objects’ locations is changed). The user/manager can actually view the location of all the objects of his/her choosing (customers, packages, warehouses, drones etc.). This map will include the specified objects drawn around the map in its exact location. Doing this requires giving the map a coordinate system, asking the program to convert the location to coordinates and also modifying it so that it matches the point on the map.