Tools and Discussion

**Pros:**
- Uniquely grounded in physical world
- Easy to create and process image targets
- Unity Support

**Cons:**
- Inconsistent user experience
- Pricing tiers
- Limited customizations at runtime
- Limited community help resources

**Other tools to consider:**
- D3.js - a more flexible library that can handle more than just maps however expect a sharper learning curve
- p5.js - inspired by goals of processing language to make coding more accessible. Three tends to let you get more niddy-gritty but if that’s a non-issue use p5.js

**Pros:**
- Similar to WebGL/OpenGL
- Highly Customizable
- Flexible visualization and data processing
- Similar to WebGL/OpenGL
- Outdated community help resources
- A bit resource intensive

**Cons:**
- Reliance on map tile providers
- Requires some form of location data
- Awkward to use outside of mapping
- Requires attribution

**Other tools to consider:**
- ARToolKit - Free and open source with a Unity plugin, ARToolKit covers most of the bases but some functionality, such as creating image targets and image tracking, are worse. Probably best for smaller projects.

**Pros:**
- Uniquely grounded in physical world
- Easy to create and process image targets
- Unity Support

**Cons:**
- Inconsistent user experience
- Pricing tiers
- Limited customizations at runtime
- Limited community help resources

**Other tools to consider:**
- ARToolKit - Free and open source with a Unity plugin, ARToolKit covers most of the bases but some functionality, such as creating image targets and image tracking, are worse. Probably best for smaller projects.

**Pros:**
- Similar to WebGL/OpenGL
- Highly Customizable
- Flexible visualization and data processing

**Cons:**
- Similar to WebGL/OpenGL
- Outdated community help resources
- A bit resource intensive

**Other tools to consider:**
- p5.js - inspired by goals of processing language to make coding more accessible. Three tends to let you get more niddy-gritty but if that’s a non-issue use p5.js

**Pros:**
- Quick to learn, quick to code in
- Easy to change look of overlay
- Fleshed-out tutorials and examples

**Cons:**
- Requires attribution

**Other tools to consider:**
- D3.js - a more flexible library that can handle more than just maps however expect a sharper learning curve

Conclusion

Leaflet was the easiest to learn but that came at the cost of being overly specialized. There’s not much Leaflet has to offer and ultimately, if time isn’t a major constraint, it would be better to learn D3.js than Leaflet.

three.js is a highly versatile library that could handle all functionality of Leaflet with enough blood, sweat, and tears. If you’re a die-hard OpenGL/WebGL fan but you just like the concepts instead of the code then three.js is good for you. If you want the complete WebGL experience then you might not appreciate how three.js abstracts away some of the quirks of WebGL. If you’d rather not splinter any hairs and have a pleasant afternoon then p5.js is waiting, just be aware that you’re giving up some more niche functionality that you might want on a more specific project.

Vuforia makes AR fairly easy to set up, just pray you don’t have to modify some of the default behavior since that ramps up the difficulty significantly. If you are using Vuforia in Unity, you will want something to edit C# code. Vuforia can have hiccups when asked to recognize an image in various lighting conditions. Vuforia really struggles when your image target is not one of the default supported shapes. If this is the case then it’s likely that your model will jump all over target image. If you are looking for Vuforia alternatives be aware that Vuforia is one of the more friendly and approachable AR tool. Do your research when you select your AR tools since the AR field is changing rapidly and what’s good today may not be tomorrow.

Future Work

- Connect visualizations to database running MongoDB
- Make 2D traffic visualization with D3.js
- Cohesive color scheme for crime visualizations
- Look into more efficient 3D visualization using p5.js
- Simplify customization for visualizations
- Improve usability of tools for visualizations
- Incorporate CMS into webpage

Acknowledgments

This work was supported by the National Science Foundation (NSF) award #ACI-1560410. This project wouldn’t have been possible without these people:
- Shayan Shams for the database assistance and advice
- Chris Cano and Tim Wright for the technical support.
- Brandon Wheeler for keeping the REU running smoothly and making all the events wonderful
- Vuforia team for the image used in the vuforia table