Phase 1. Compile

What is the most effective methodology of compiling sizable C++

- Try to compile a small appliation to wasm.
- Try to compile CGAL to wasm.
- Try to compile GDAL to wasm.
- Experiment with different compilation methods and settings, and compare results
- Define a guide of compiling large (C++) libraries to Wasm

Phase 2. Develop

How to design and create

- Develop the basics of a browser-based visual programming language
- Add general geometry processing & visualization support
- Add core GIS functionalities including WFS and WMS
- Add auxiliary functionalities
- Examine the state of this environment as it is right now

Phase 3. Synthesize

How can wasm-compiled geoprocessing libraries be used in

- Add wasm plugin support to the geo web vpl
- Add the wasm-compiled C++ libraries to the geo web vpl as plugins or dependencies
- Experiment with different distribution strategies and considerations
- Choose an effective method, and explain why
- R
  A "geo-
  GDAL &

Phase 4. Benchmark

What are t

client-side geoprocessing

- Setup a 'benchmark test suite'
- Perform Benchmarks
- Conclude the performance penaltiy compared to native
- Result:
  A more performant environment

Phase 5. Utilize

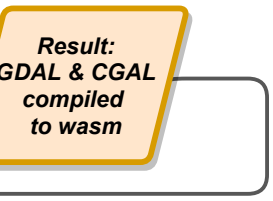
What are the advantages and d

using a client-side geoprocessing

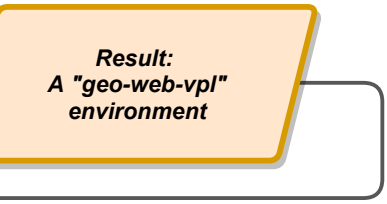
- Create an application using the environment
- State advantages and disadvantages
- Discuss what this means for the geospatial community
- Result:
  Example projects utilizing the vpl environment

- Completed Preliminary work
- Development
- Assessment

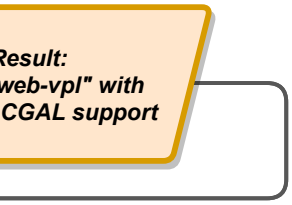
geoprocessing libraries to WebAssembly?



a client-side geoprocessing environment?



a client-side geoprocessing environment?



the performance considerations of using a
g environment powered by WebAssembly?

disadvantages of GIS applications created
g environment powered by WebAssembly?


Conclusion
Products