date	2022-07-06 - Wednesday
present	Giorgio, Ken, Stelios, Jos
Location	Cyberspace

0. Introduction

1. What has been done?

"I've been asked to focus on the 'use case' and experiments section, and to see how far I can come with the ideas that I had. I've interpreted this as writing out the applications intended for these particular use-cases, writing down key criteria on which these applications should be judged, and then attempting to make these applications a reality "

Writing:

I have written sketches of three different use-case applications. I have made some first attempt at what these applications should be judged upon. This is followed up by a hypothesized assessment.

Use-case applications

So far, I have not been able to start on the actual use-case applications, since lots of aspects of the environment prevented me from starting working on them. I have managed to fix some of these aspects, more on this in de 'coding section'. I think I need another week to actually build one of the use-case applications.

However, by developing this, I Think I can more clearly flush out the story.

Coding:

Feature: Plugins can now 'actually' be shared:

- The plugins used can now be edited using the plugin menu.
- This configuration is saved within the savefile the graphs (json).
- Both local plugins (local path) and online plugins can be used like this (npm CDN url)
- When someone else loads a graph using an online plugin, they request these plugin at that adress.

Feature: (Rust) Plugins now use a 'magic method' system to communicate more intimately with geofront:

- a 'Data' Trait now makes objects within a plugin compose and decompose into a 'normal' json.
- a 'Renderable' Trait now tells geofront how to fill a shader with this objects information. We can now directly visualize things within a plugin
- a 'Explain' Trait can be used to give useful meta-info about a class.

    - it also can give descriptions, so that all expose functions can easily be used.
- All these features are optional.

Feature: Rust based standard library

- Very incomplete as it stands
- Added a rust based 'lastools' dependency to the project
    - Planning to use it during the use-cases
    - We can now load and manipulate 'LAS' files.
    - 'LAZ' should also work, but I've run into errors
- Trying to recreate the javascript based STD.
    - Would be a very nice thing to compare.
        - It would tell us something about if jumping between javascript and rust is more effective than just using javascript.

Feature: bugfixes

- Make Sliders Save & Load correctly.
- Make it so huge jsons can not be inspected, for performance reasons.

```
Notes during meeting...
```

2. Where are we currently?

- I like the current state of the plugins. I can now practically take any rust library, and quite easily convert it into a geofront-integrated plugin.

- The LAS support is nice. This is a good feature to build the 'experiments section around' to test and build demo's around.

- I'm worried about the interaction between the software project & the written thesis, as usual.

    - Every hour spend coding feels like time I should be spending writing.
    - When I'm writing, the whole thesis feels directionless and unfinished.

- Sidenote: I've noticed that my motivation & clarity on the project drops to the floor as soon as I start writing the thesis. It feels like the written thesis has had to switch perspective many times, and this is demotivating. This is the complete opposite when I work on the software itself.

-> this is the main point, the main thing I want to ask you all: How to prevent this demotivation? How to make meaningful use-case applications which the written thesis can use to build a story around, while not diving too far into the code? How to make the written thesis clear & concise? I know these are really open and difficult to answer questions, and it basically comes down to me trying to find that balance as I go forward. Still, if you have any tips, much appreciated 😃.

- What are some gaping issues? / things that are missing?

```
Notes during meeting...
```

3. What needs to be done for next meeting?

- I think I want to finish '*A*' chapter of the thesis. As it stands, no chapter is truly done.
  - Implementation is the easiest chapter to finish for me.
- I would love nothing more than to continue development on interesting plugins now that the plugin integration is more or less done. I could try and incorporate:
  - cjval
  - DXF export
  - LAS export
  - The old CGAL demo
  - Delaunay triangulation with isocurves We are *JUST* at the point where we can try and truly use this environment. However, this takes time away from writing the thesis. What is wisdom?

```
Notes during meeting...
```

Planning for coming weeks

- Do we have an actual P4 date pinned yet? should we request one in advance??
- (for yourself) How much time do I still need to get P4 / P5 ready?

WVTTK

Next Meeting

Appendix: Notes

"These are notes written during this period. It explains my thought process, but are not very coherent or well-edited."

on the one hand: I dislike 'easy programming languages' to an extend

- I'm a huge fan of the 'rust' idea: certain things must become more complicated again, to do them right and securely
- BUT: my thesis is about creating an 'accessible' visual programming language. How do I square these two things?
- SO: This must not be about 'making programming / geoprocessing as a whole easier'
    - This must be about: ravi peter's argument: visual geo-programming just to test, visualize, and play with parameters.
    - A vpl is just a useful interface in its own right
    - Geofront's addition: the ability to share this online.
    - MAYBE this makes programming easier for some people, but this will be a 'side product' of the other pursuits.

SO: Geofront is intended as a Computational Geometry Sandbox environment. Meant for:

- publication
- sharing ideas
- trying things out
- debugging code
- learning

NOT for making programming as a whole 'easier'

Does it succeed in doing that?

Yes:

- Using the environment, you can take a rust-based geo-computation function or library, and without very many steps, use it within a visual programming environment. The environment can then be used to:
    - Visually debug,
    - Play around with parameters,
    - Compare it to similar libraries,
    - And, unique to this environment, do this all online, in a 'published' format: the full configuration can be shared using a URL.

  The combination of these aspects makes this environment unique.
- These libraries can be used with a minimum of configurations. Any Rust library with `wasm-bindgen` annotations, in other words, any rust library intended for javascript consumption, automatically works in

'geofront', albeit with some edge-case exceptions.

No:

- While it is indeed possible to use and run any rust library with `wasm-bindgen` annotations, in order to properly communicate, visualize, and make data interoperable, special 'config' functions and methods are needed.

- For now, only 'Rust' and 'JavaScript / TypeScript' can be properly used as libraries. However, most libraries relevant to geo-computation are C++ based. While C++ has excellent support for compiling full, self contained applications to WebAssembly using the 'emscripten' toolset, it lacks rust's level of support in compiling existing libraries. `embind` can be used for this, but compilation using embind is a more tedious, error-prone process. Additionally, any dependencies

- the environment uses browser-based calculations, so it cannot be used properly for big data, or other expensive processes. Future work: compile the flowchart, run it headless on a server for large datasets.

- the flowchart can only represent linear processes. Many geoprocessing algorithms are iterative and make use of conditionals. These cannot easily be expressed on the canvas. As such, these processes must happen within the context of a function, within a 'computational node' ...

(Low code only works in very constraint environments)

(Learning is a cornerstone within software development. For example: Engineers go to great lengths writing compilers which tell the users precisely what is wrong with the code.)

....

The problem is, is that "Why I made geofront" and "What I can proof about geofront" are two different phenomena. I can proof that it can in fact be used to take existing libraries, and turn these libraries into an interactive, sharable vpl in a web browser with a 3d visualizer attached. I can proof things about the performance hit this takes.

- The thesis shows the possibilities and limitations of the current methodology.

WHY DOES SOMEONE WANT TO TAKE EXISTING LIBRARIES AND TURN THEM INTO A SHARABLE VPL FORMAT?

- interactive, visual debugging
  - explaining the behavior of algorithms to yourself and others
  - why web: collaborative debugging
- reproducability of results
  - lowering the delta between "it works for me" and "it works for someone else"
- improving the 'online presence' of research.
  - making research results 'usable' via a link
  - interdisciplinary exchange of ideas

HOW DOES THIS TRANSLATE INTO AN INTRODUCTION? -> ... -> "... Most software developed during and for research does not leave the machines of the researches involved ..."