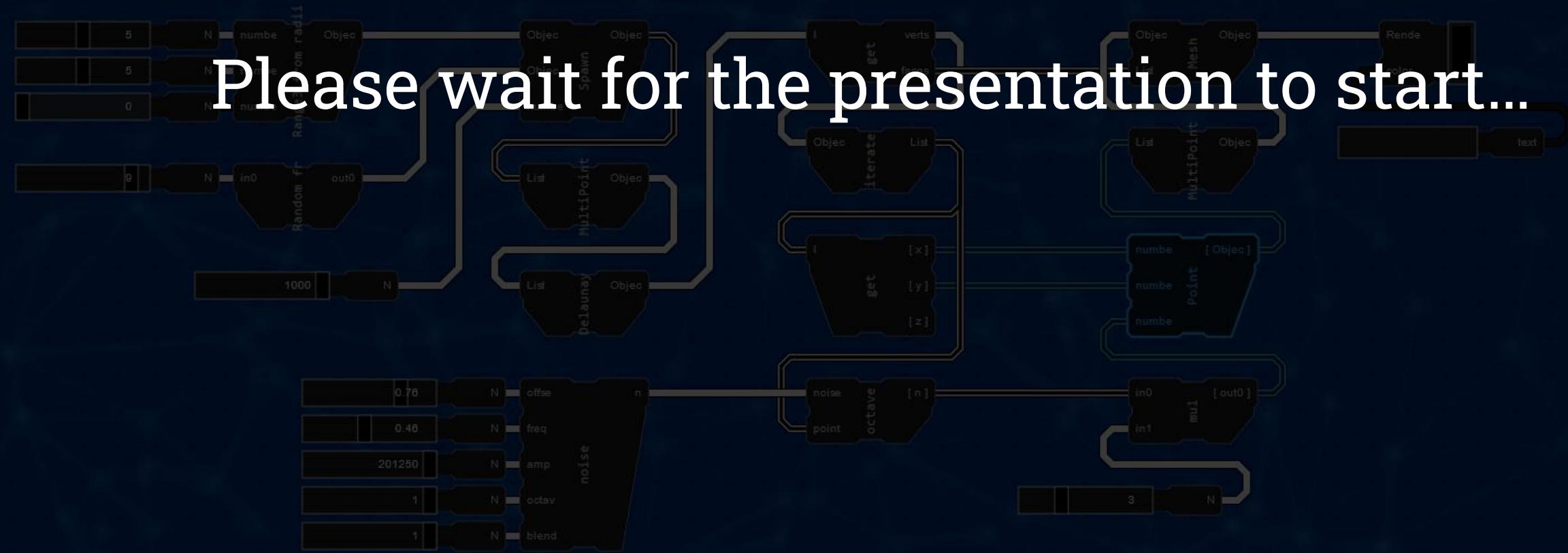


Please wait for the presentation to start...



Geofront:

Directly accessible GIS tools
using
a web-based visual programming language

Master Thesis Geomatics | Final Presentation

Jos Feenstra | November 4th 2022

1. Introduction

2. Objective

3. Background

3. Method

4. Results

5. Conclusion

1. Introduction

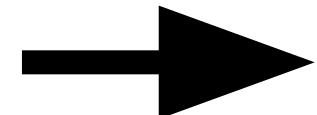
GIS:
Geographical
Information
Science



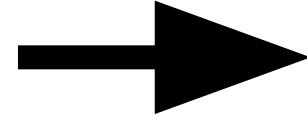
End Users

Climatology
Infrastructure
Urban planning
Agriculture
Governance
Navigation
Military
(...)

Geographical
Information
Science



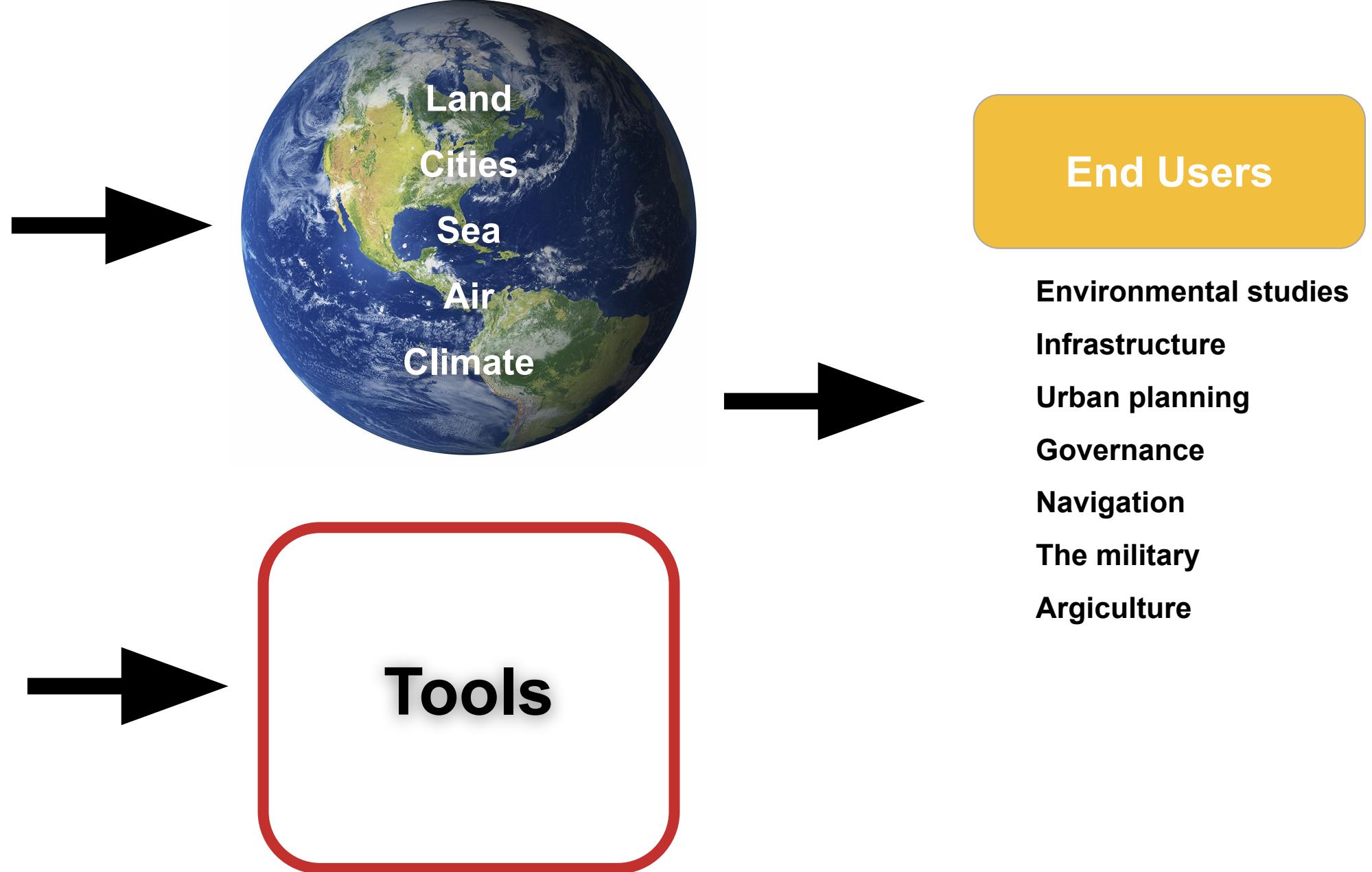
Tools



End Users

Environmental studies
Infrastructure
Urban planning
Governance
Navigation
The military
Agriculture

**Geographical
Information
Science**



Tools: Two forms of software:

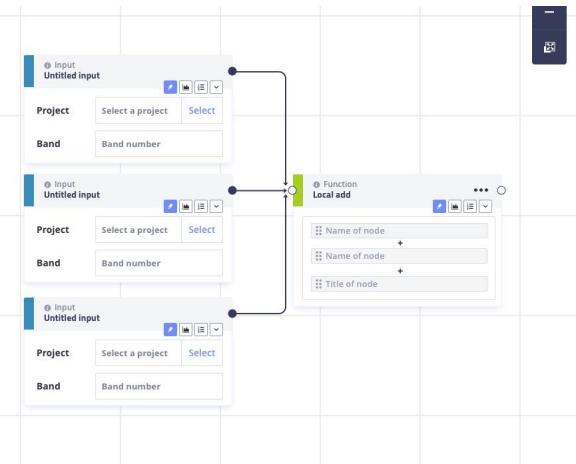
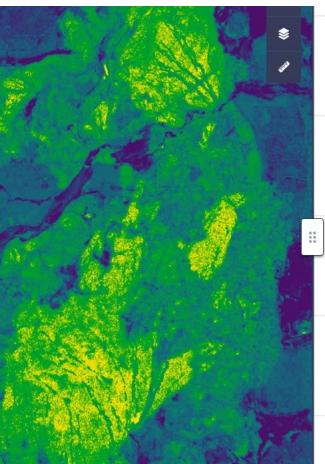
Application

Library

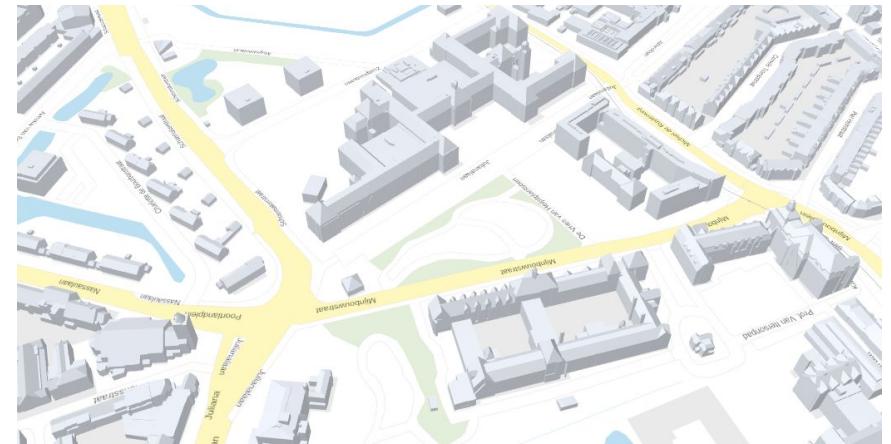
Application

Applications:

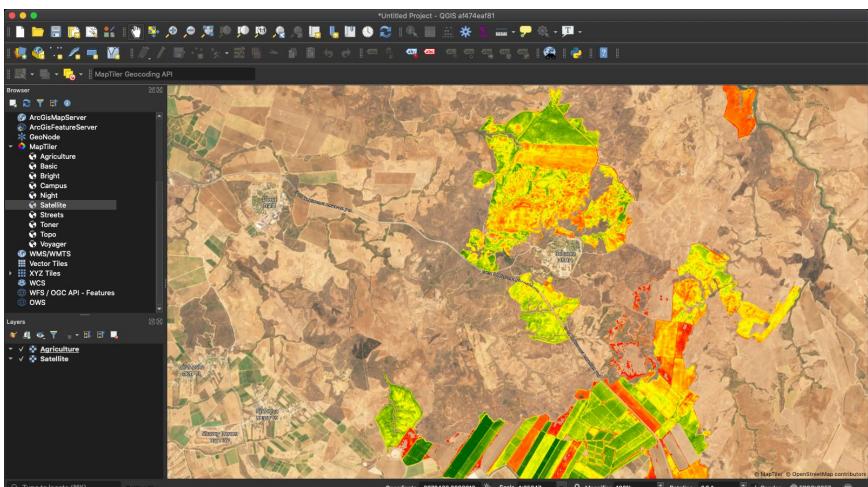
- End users
- Interaction



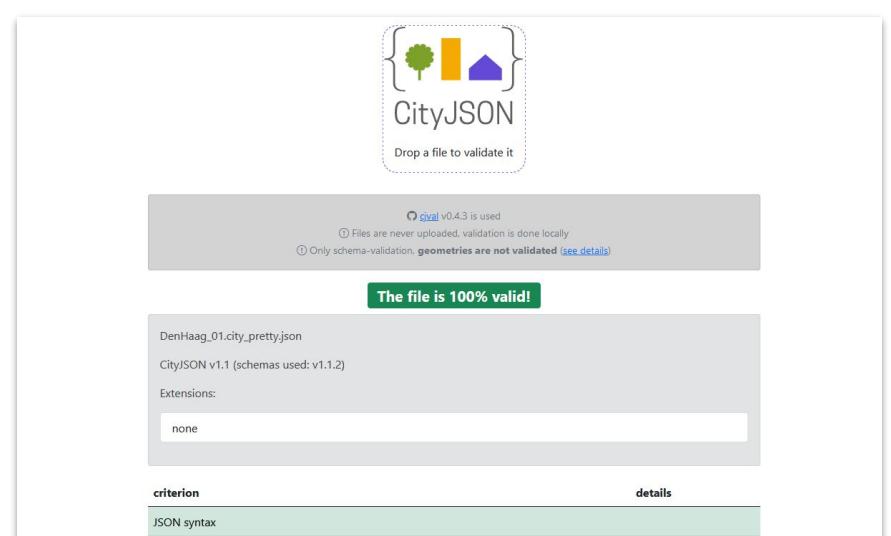
src: Model Lab



src: 3D bag viewer



src: QGIS



src: cjval

Libraries

Libraries:

- Reusable tools for applications (& other libraries)
- Cannot directly be used

3D Geoinformation Libraries

City3D



PolyFit



val3dity



cgval



cjio



PROJ



GEOS



GDAL



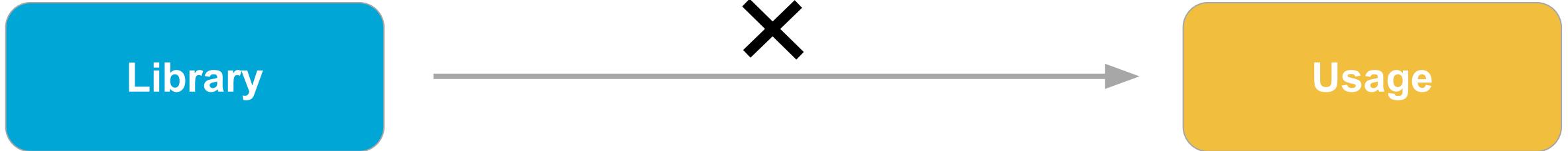
(CGAL)



(OpenCV)

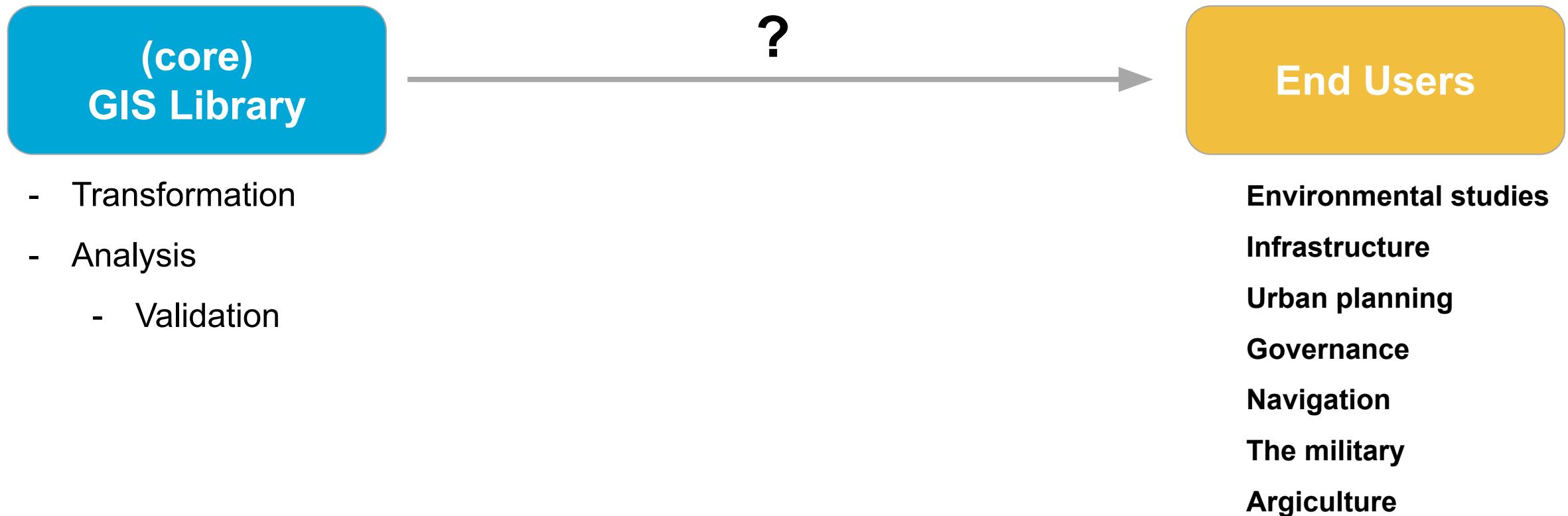


X Usability: Libraries are not directly usable



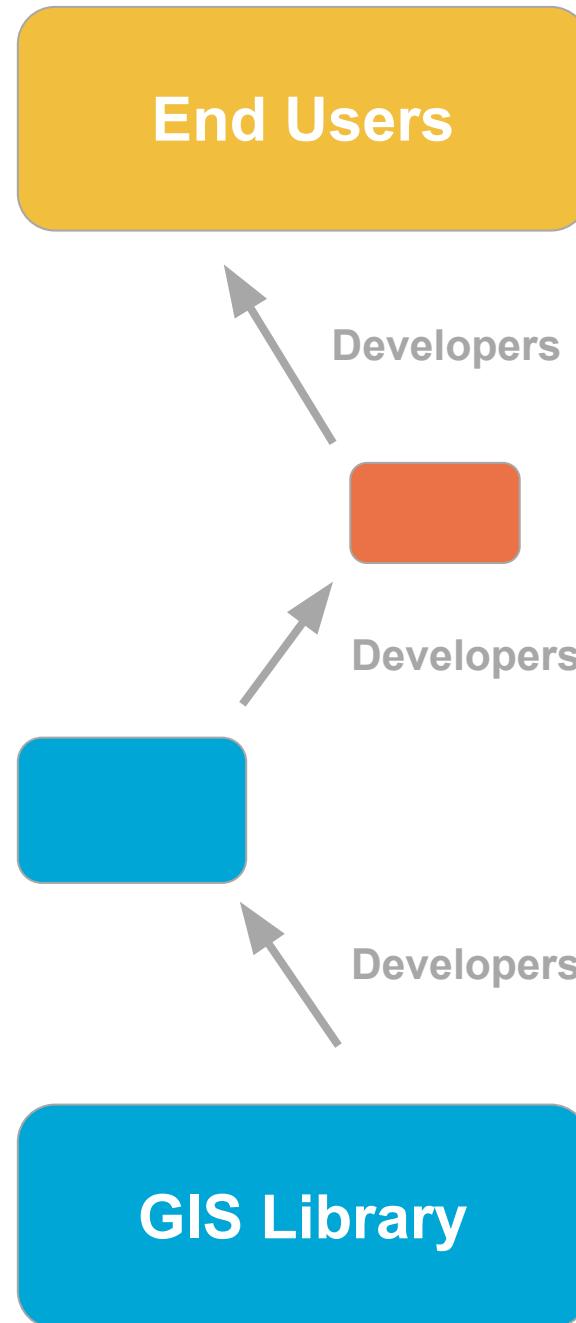


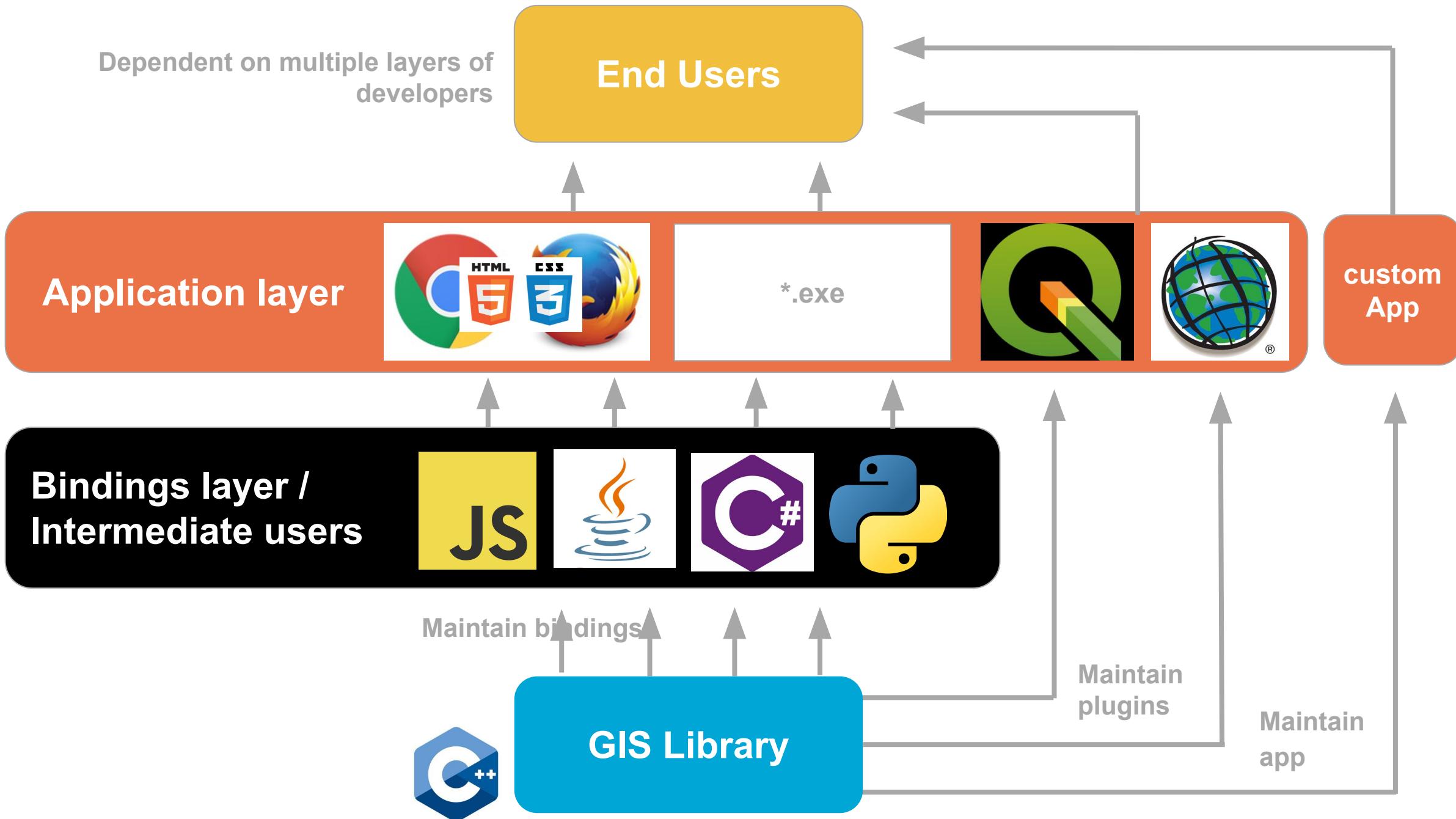
Problem:



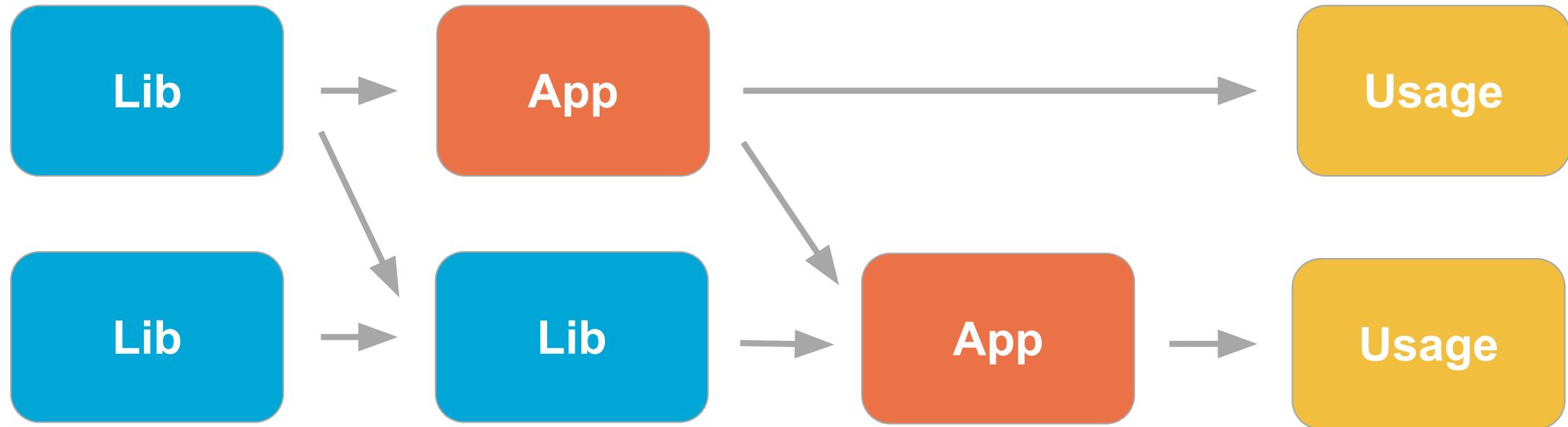
Indirection

- Only indirect access
- Dependent
- Exact?
- New research?





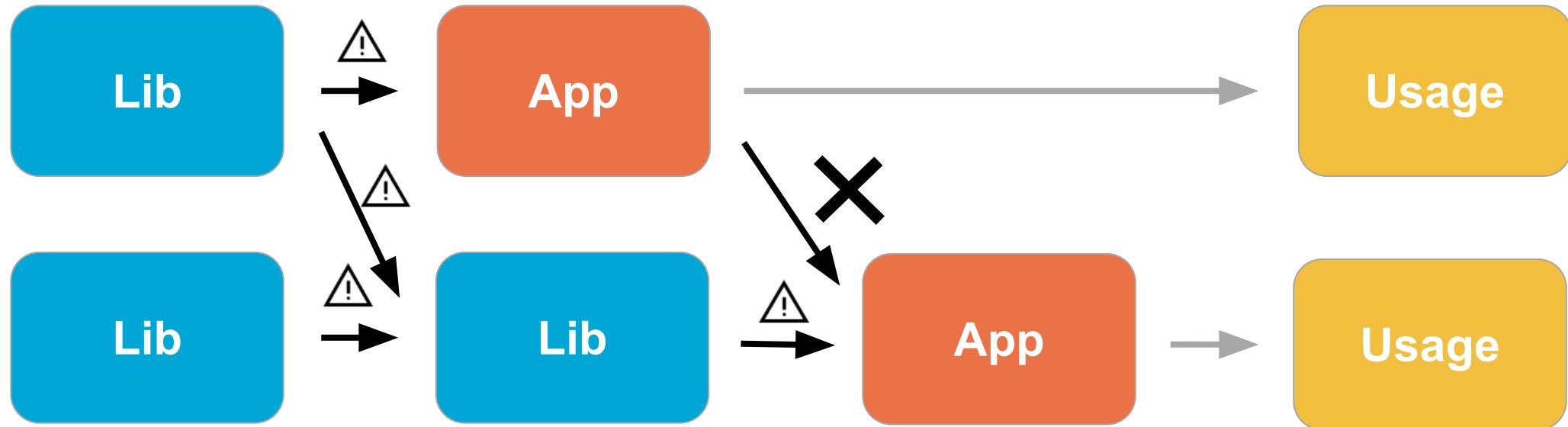
Moreover:



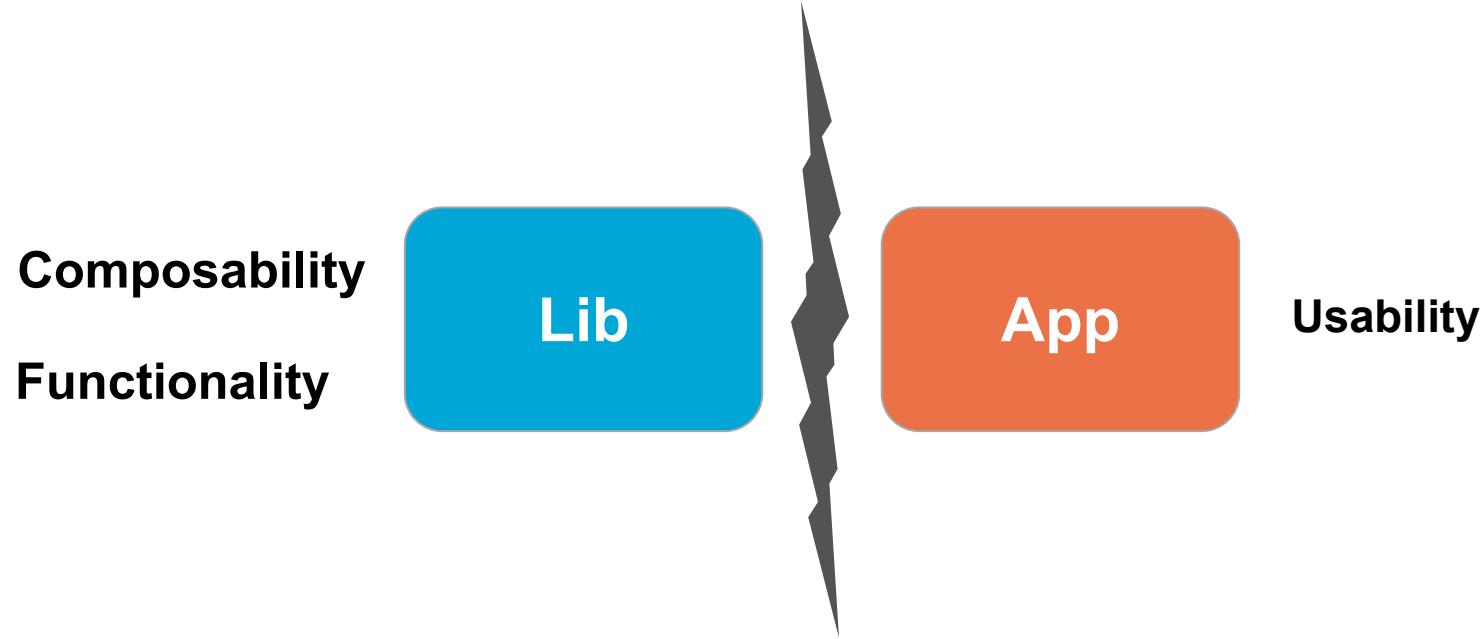
Moreover:

⚠ **Functionality:** capabilities may get lost at every step

✗ **Composability:** **apps** are **not** further composable



Conundrum:



Given this divide, how to achieve **Functionality**,
Composability, & **Usability** at the same time?

Problem statement

End users only have **Indirect access** of GIS libraries, leading to disadvantages...

... for **end users**:

- *At the mercy of in-between software*
- Non-composable applications
- Features getting *lost in translation*

... for **library developers**:

- Synchronizing bindings, plugins, applications.

...for **society**:

- reduced impact of research



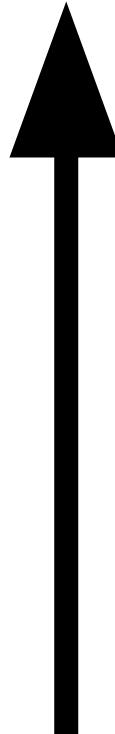
2. Objective

Goal of this study:

Allow GIS practitioners without a background in software development, to access the full potential of core transformation and analysis capabilities found in native GIS libraries.

Goal:

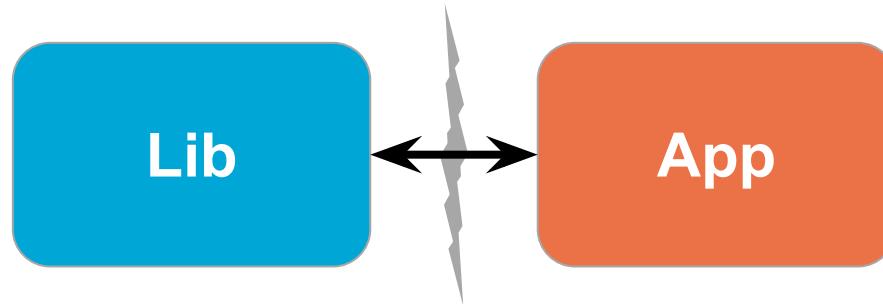
End Users



more direct access

Core
GIS Library

Goal:



apps are an endpoint: Not further composable

→ Add **composability** and **automation** to apps

A **lib** offers no visualization or GUI.

→ Add **usability** and **GUI** to libs

A **lib** must be turned into an **app** before utilization.



Some **Lib** capabilities get lost when used in an **app**

How:

Presenting and prototyping a novel method:

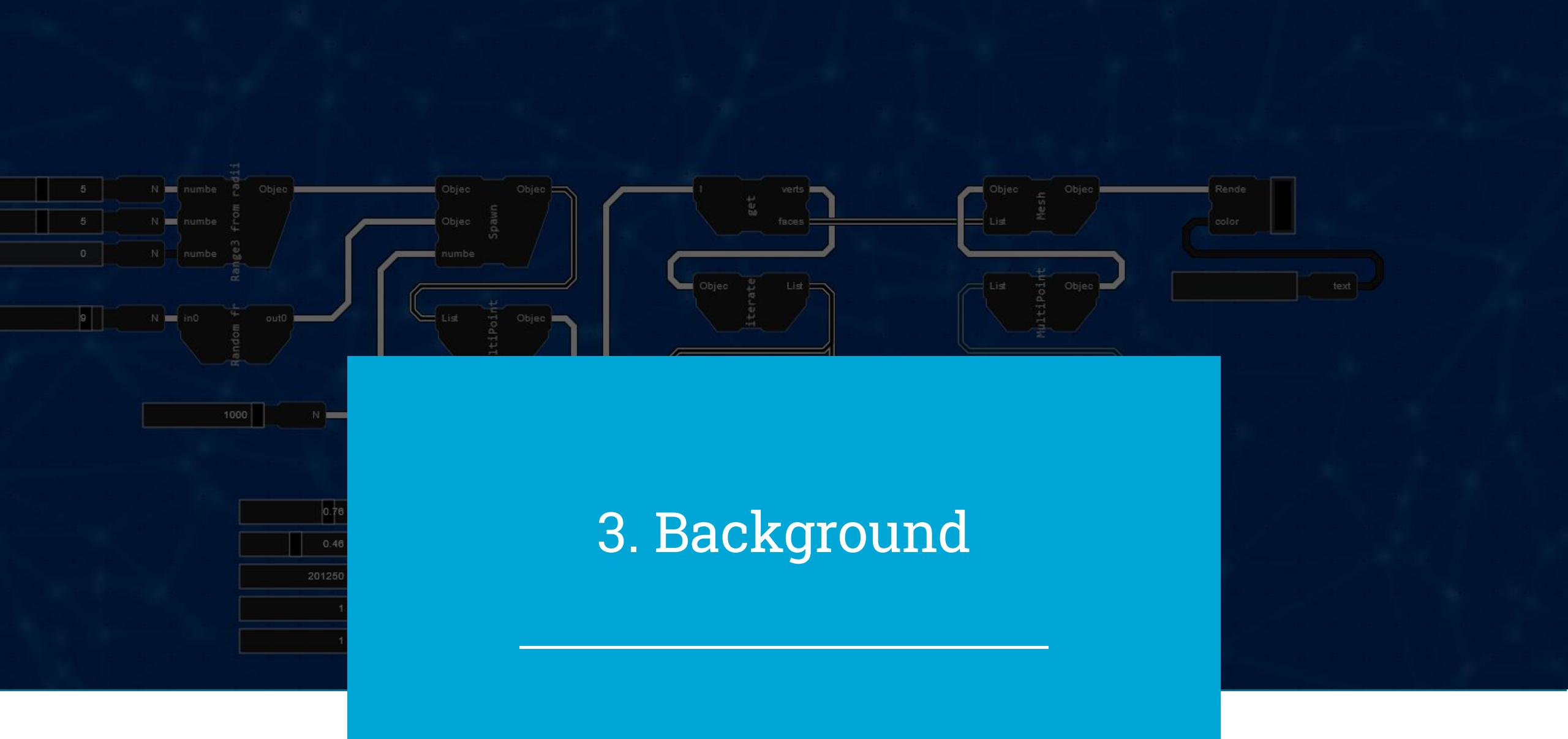
**A Web-based Visual Programming Language (VPL) using
WebAssembly**

Research question:
Is a **web based VPL** a viable method for
directly accessing native GIS libraries
with a composable interface?

Sub Questions

- What **GUI** features are required to facilitate this method, and to what extent does the web platform aid or hurt these features?
- To what extent does this method intent to address the **discrepancies** between **software applications and libraries**, as described by Elliott (2007)? Does it succeed in doing so?
- What are the differences between **compiling** a GIS library written in C++ to WebAssembly, compared to compiling a GIS library written in Rust?
- What measures are taken to make this VPL **scalable** to large geo-datasets, and how effective are these measures?
- How does this method **compare** to existing, alternative VPLs and browser-based geocomputation methods, regarding the properties relevant to the goal of direct accessibility?

3. Background



“Web-based VPL using WebAssembly”:

- 1. Web Application**
- 2. Visual Programming**
- 3. WebAssembly**

1. Web Applications

Possible solution for direct access

Web Application → distribution

- No Installation
- Cross-platform

Static Web Application

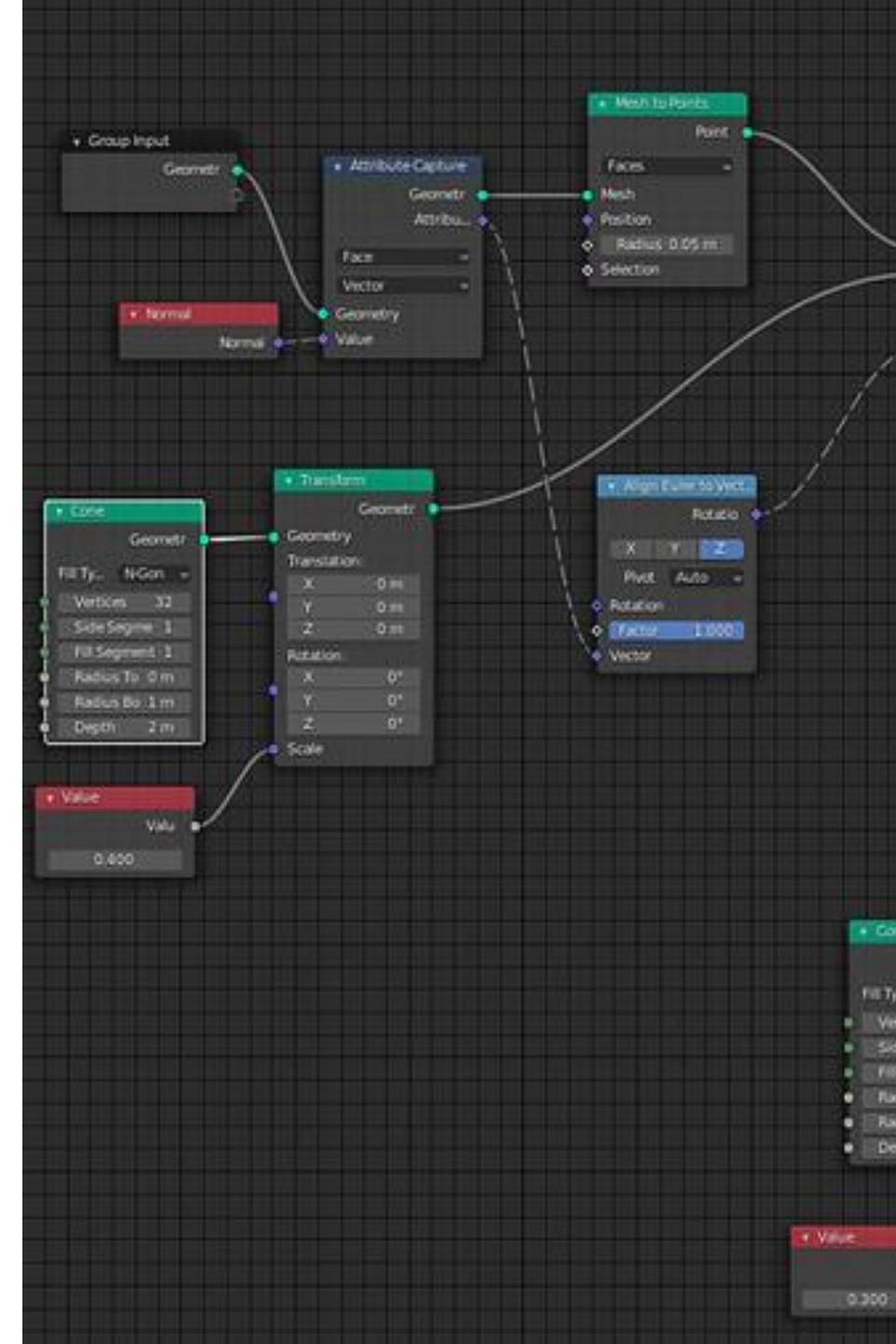
- High traffic
- Cheap
- More composable



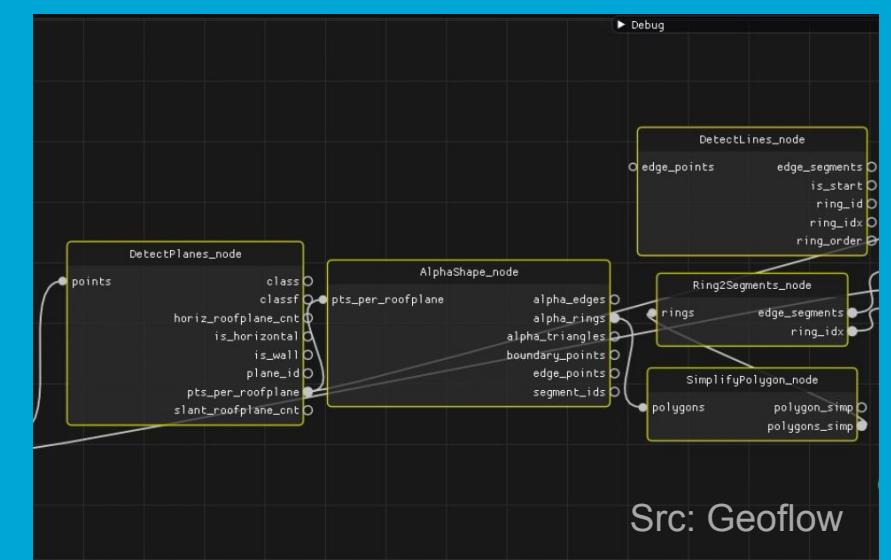
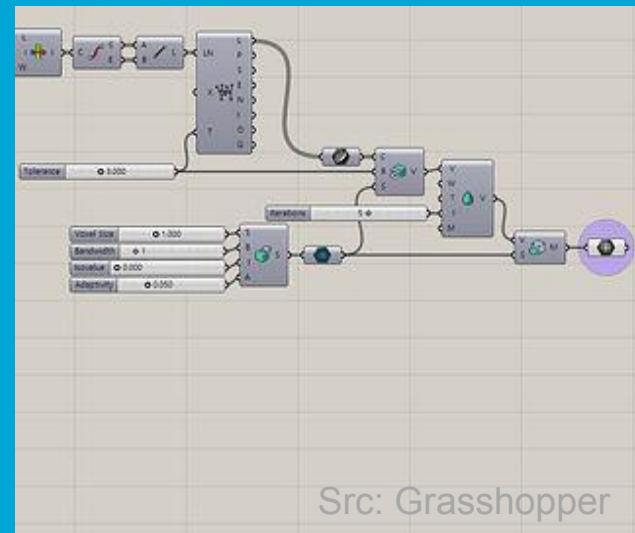
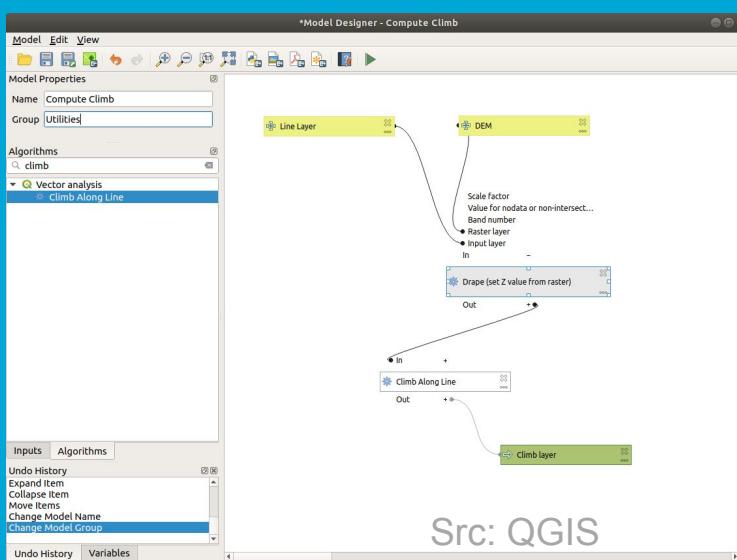
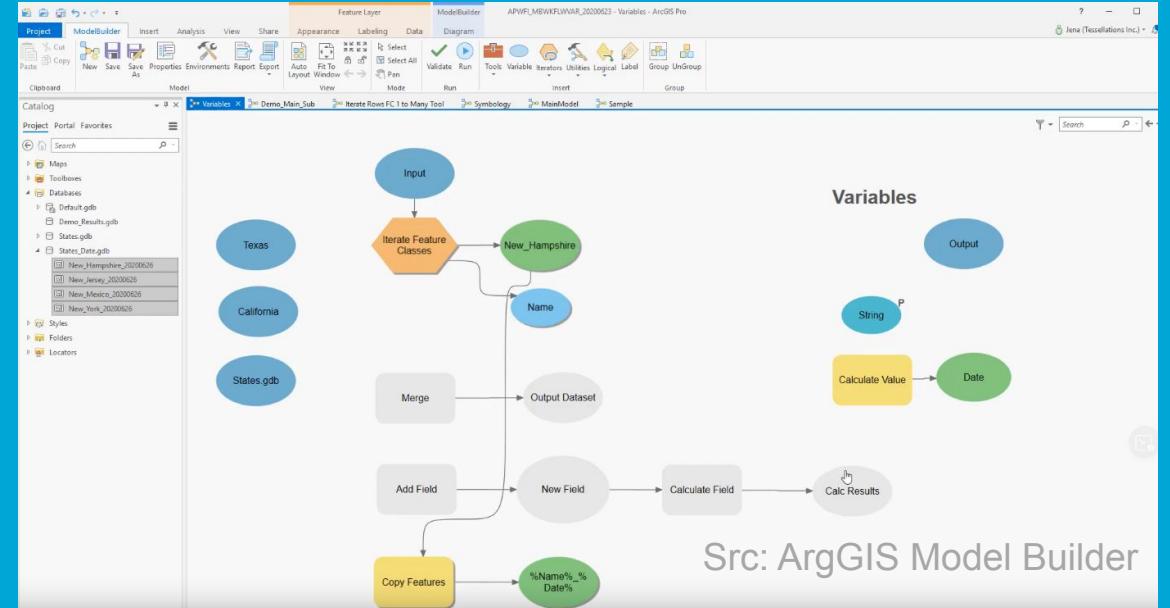
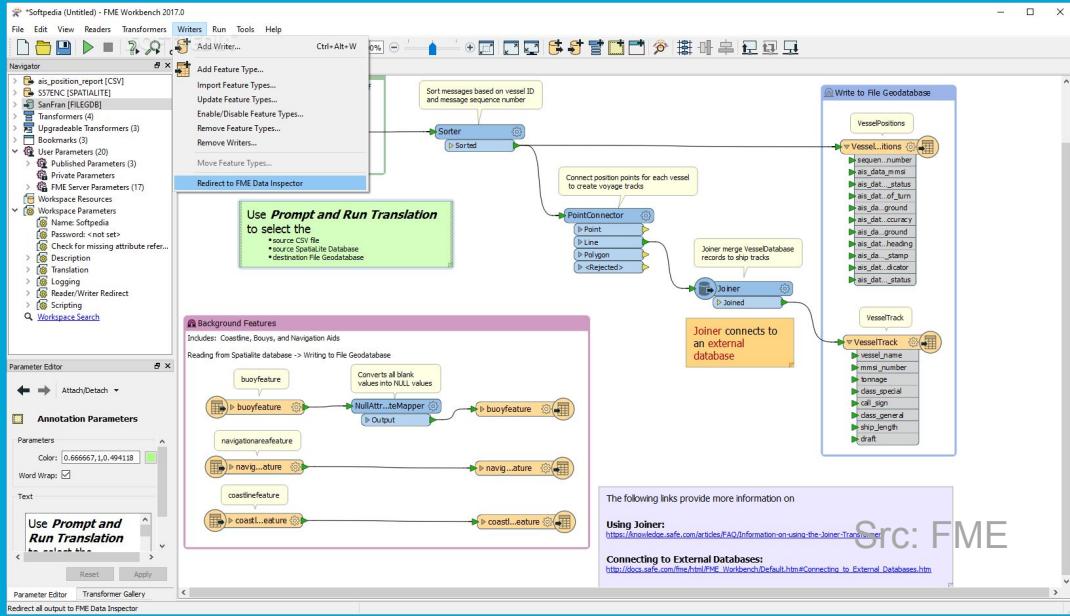
2. Visual Programming

Possible solution for Composable applications

- Visual Programming Language (VPL)
 - Both a scripting language **and** application
 - ‘programming’ by using GUI
 - Composable GUI



VPL within GIS



2. Visual Programming: GIS

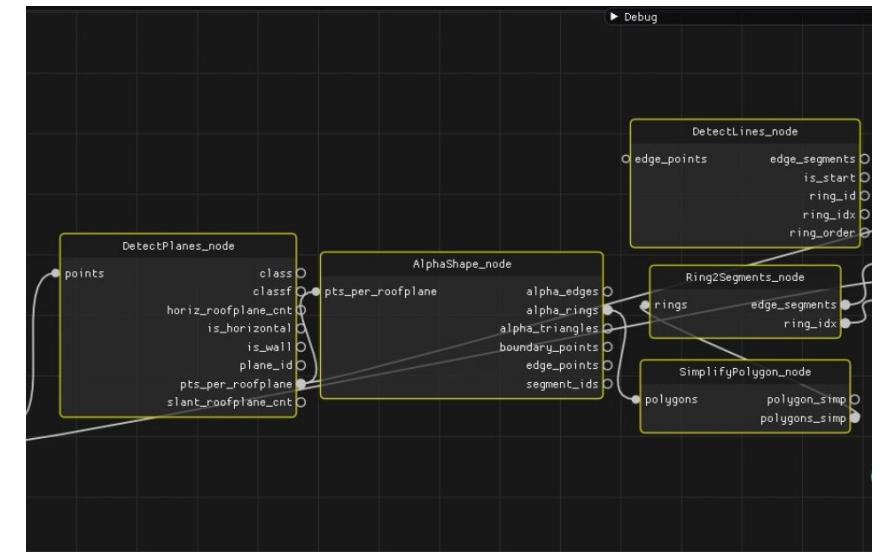
requirement: Scalability

Cloud

Scalability

CLI
Runtime

Use in



3. WebAssembly

Possible solution for more direct access

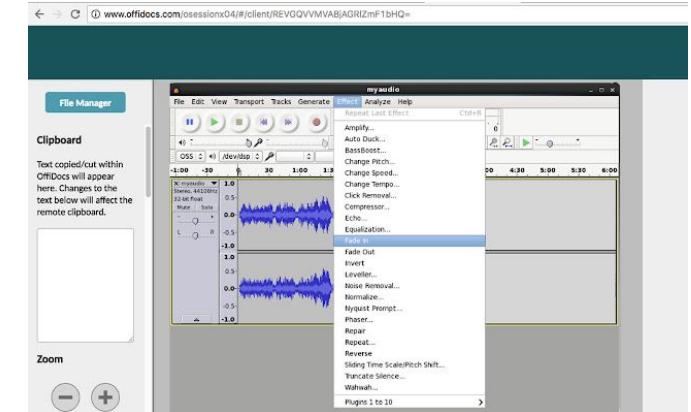
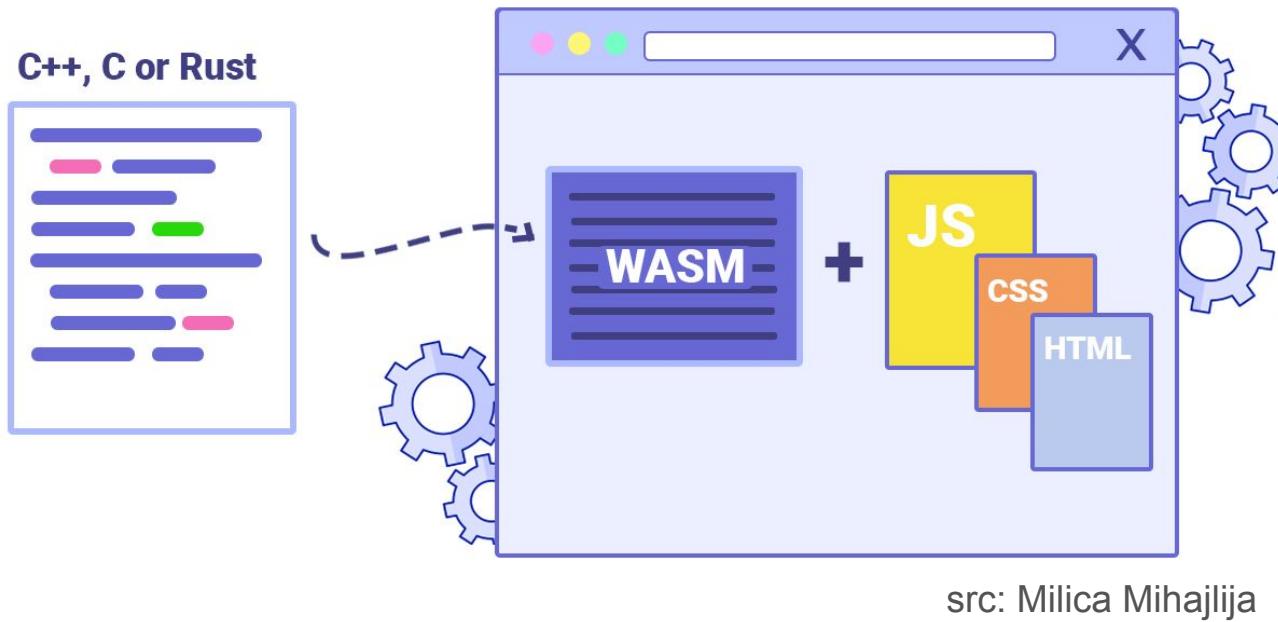
- Exchangeable binaries
- Binary compilation target `library.wasm`
 - From multiple languages
 - To multiple runtimes
- Since 2017 (Haas, 2017)
- In browsers since 2019 (W3C, 2019)

```
1 √ [module
2   (type $t0 (func (param i32 i32)))
3   (type $t1 (func (param i32 i32 i32) (result i32)))
4   (type $t2 (func (param i32 i32) (result i32)))
5   (type $t3 (func (param i32)))
6   (type $t4 (func (param i32) (result i32)))
7   (type $t5 (func))
8   (type $t6 (func (param i32) (result f32)))
9   (type $t7 (func (param i32 f32)))
10  (type $t8 (func (param f32 f32) (result i32)))
11  (type $t9 (func (param i32 i32) (result f32)))
12  (type $t10 (func (param i32 i32 i32)))
13  (type $t11 (func (param i32 i32 i32 i32) (result i32)))
14  (type $t12 (func (param i32) (result i32)))
15  (type $t13 (func (param i32) (result i64)))
16  (type $t14 (func (param i32 i32 i32 i32)))
17  (type $t15 (func (param i32 i32 i32 i32) (result i32)))
18  (type $t16 (func (param i32 i32 i32 i32 i32) (result i32)))
19  (type $t17 (func (param i32 i32 i32 i32 i32) (result i32)))
20  (type $t18 (func (param i64 i32 i32) (result i32)))
21  import "__wbindgen_placeholder__" "__wbindgen_describe" (func $__ZN12wasm_bindgen19__wbindgen_describe17hdb3
22  import "__wbindgen_externref_xform" "__wbindgen_externref_table_grow" (func $__ZN12wasm_bindgen9externref3
23  import "__wbindgen_externref_xform" "__wbindgen_externref_table_set_null" (func $__ZN12wasm_bindgen9extern
24  $t3))
25 √ (import "__wbindgen_placeholder__" "__wbindgen_throw" (func $__ZN12wasm_bindgen16__wbindgen_throw17h7bfcc15cf6
26  (func $add (type $t2) (param $p0 i32) (param $p1 i32) (result i32)
27    local.get $p1
28    local.get $p0
29    i32.add)
30    (func $__ZN12wasm_bindgen_describe_add (type $t5)
31      call $__ZN12wasm_bindgen4__rt19link_mem_intrinsics17h3cc2179cca039a8aE
32      i32.const 11
33      call $__ZN12wasm_bindgen19__wbindgen_describe17hdb3ff320fcac3194E
34      i32.const 0
35      call $__ZN12wasm_bindgen19__wbindgen_describe17hdb3ff320fcac3194E
36      i32.const 2
37      call $__ZN12wasm_bindgen19__wbindgen_describe17hdb3ff320fcac3194E
38      call $__ZN60__LT$u32$u20$as$u20$wasm_bindgen..describe..WasmDescribe$GT$8$describe17h45229e62f39c456cE
39      call $__ZN60__LT$u32$u20$as$u20$wasm_bindgen..describe..WasmDescribe$GT$8$describe17h45229e62f39c456cE
40      call $__ZN60__LT$u32$u20$as$u20$wasm_bindgen..describe..WasmDescribe$GT$8$describe17h45229e62f39c456cE
41    √ (func $__wbg_point_free (type $t3) (param $p0 i32)
42      block $B0
43        block $B1
44          local.get $p0
45          i32.eqz
46          br_if $B1
47          local.get $p0
48          i32.load
49          br_if $B0
50        )
51      )
52    )
53  )
54  )
55  )
56  )
57  )
58  )
59  )
60  )
61  )
62  )
63  )
64  )
65  )
66  )
67  )
68  )
69  )
70  )
71  )
72  )
73  )
74  )
75  )
76  )
77  )
78  )
79  )
80  )
81  )
82  )
83  )
84  )
85  )
86  )
87  )
88  )
89  )
90  )
91  )
92  )
93  )
94  )
95  )
96  )
97  )
98  )
99  )
100  )
101  )
102  )
103  )
104  )
105  )
106  )
107  )
108  )
109  )
110  )
111  )
112  )
113  )
114  )
115  )
116  )
117  )
118  )
119  )
120  )
121  )
122  )
123  )
124  )
125  )
126  )
127  )
128  )
129  )
130  )
131  )
132  )
133  )
134  )
135  )
136  )
137  )
138  )
139  )
140  )
141  )
142  )
143  )
144  )
145  )
146  )
147  )
148  )
149  )
150  )
151  )
152  )
153  )
154  )
155  )
156  )
157  )
158  )
159  )
160  )
161  )
162  )
163  )
164  )
165  )
166  )
167  )
168  )
169  )
170  )
171  )
172  )
173  )
174  )
175  )
176  )
177  )
178  )
179  )
180  )
181  )
182  )
183  )
184  )
185  )
186  )
187  )
188  )
189  )
190  )
191  )
192  )
193  )
194  )
195  )
196  )
197  )
198  )
199  )
200  )
201  )
202  )
203  )
204  )
205  )
206  )
207  )
208  )
209  )
210  )
211  )
212  )
213  )
214  )
215  )
216  )
217  )
218  )
219  )
220  )
221  )
222  )
223  )
224  )
225  )
226  )
227  )
228  )
229  )
229  )
230  )
231  )
232  )
233  )
234  )
235  )
236  )
237  )
238  )
239  )
239  )
240  )
241  )
242  )
243  )
244  )
245  )
246  )
247  )
248  )
249  )
249  )
250  )
251  )
252  )
253  )
254  )
255  )
256  )
257  )
258  )
259  )
259  )
260  )
261  )
262  )
263  )
264  )
265  )
266  )
267  )
268  )
269  )
269  )
270  )
271  )
272  )
273  )
274  )
275  )
276  )
277  )
278  )
279  )
279  )
280  )
281  )
282  )
283  )
284  )
285  )
286  )
287  )
288  )
289  )
289  )
290  )
291  )
292  )
293  )
294  )
295  )
296  )
297  )
298  )
299  )
299  )
300  )
301  )
302  )
303  )
304  )
305  )
306  )
307  )
308  )
309  )
309  )
310  )
311  )
312  )
313  )
314  )
315  )
316  )
317  )
318  )
319  )
319  )
320  )
321  )
322  )
323  )
324  )
325  )
326  )
327  )
328  )
329  )
329  )
330  )
331  )
332  )
333  )
334  )
335  )
336  )
337  )
338  )
339  )
339  )
340  )
341  )
342  )
343  )
344  )
345  )
346  )
347  )
348  )
349  )
349  )
350  )
351  )
352  )
353  )
354  )
355  )
356  )
357  )
358  )
359  )
359  )
360  )
361  )
362  )
363  )
364  )
365  )
366  )
367  )
368  )
369  )
369  )
370  )
371  )
372  )
373  )
374  )
375  )
376  )
377  )
378  )
379  )
379  )
380  )
381  )
382  )
383  )
384  )
385  )
386  )
387  )
388  )
389  )
389  )
390  )
391  )
392  )
393  )
394  )
395  )
396  )
397  )
398  )
398  )
399  )
400  )
401  )
402  )
403  )
404  )
405  )
406  )
407  )
408  )
409  )
409  )
410  )
411  )
412  )
413  )
414  )
415  )
416  )
417  )
418  )
419  )
419  )
420  )
421  )
422  )
423  )
424  )
425  )
426  )
427  )
428  )
429  )
429  )
430  )
431  )
432  )
433  )
434  )
435  )
436  )
437  )
438  )
439  )
439  )
440  )
441  )
442  )
443  )
444  )
445  )
446  )
447  )
448  )
449  )
449  )
450  )
451  )
452  )
453  )
454  )
455  )
456  )
457  )
458  )
459  )
459  )
460  )
461  )
462  )
463  )
464  )
465  )
466  )
467  )
468  )
469  )
469  )
470  )
471  )
472  )
473  )
474  )
475  )
476  )
477  )
478  )
479  )
479  )
480  )
481  )
482  )
483  )
484  )
485  )
486  )
487  )
488  )
489  )
489  )
490  )
491  )
492  )
493  )
494  )
495  )
496  )
497  )
498  )
498  )
499  )
500  )
501  )
502  )
503  )
504  )
505  )
506  )
507  )
508  )
509  )
509  )
510  )
511  )
512  )
513  )
514  )
515  )
516  )
517  )
518  )
519  )
519  )
520  )
521  )
522  )
523  )
524  )
525  )
526  )
527  )
528  )
529  )
529  )
530  )
531  )
532  )
533  )
534  )
535  )
536  )
537  )
538  )
539  )
539  )
540  )
541  )
542  )
543  )
544  )
545  )
546  )
547  )
548  )
549  )
549  )
550  )
551  )
552  )
553  )
554  )
555  )
556  )
557  )
558  )
559  )
559  )
560  )
561  )
562  )
563  )
564  )
565  )
566  )
567  )
568  )
569  )
569  )
570  )
571  )
572  )
573  )
574  )
575  )
576  )
577  )
578  )
579  )
579  )
580  )
581  )
582  )
583  )
584  )
585  )
586  )
587  )
588  )
589  )
589  )
590  )
591  )
592  )
593  )
594  )
595  )
596  )
597  )
598  )
598  )
599  )
600  )
601  )
602  )
603  )
604  )
605  )
606  )
607  )
608  )
609  )
609  )
610  )
611  )
612  )
613  )
614  )
615  )
616  )
617  )
618  )
619  )
619  )
620  )
621  )
622  )
623  )
624  )
625  )
626  )
627  )
628  )
629  )
629  )
630  )
631  )
632  )
633  )
634  )
635  )
636  )
637  )
638  )
639  )
639  )
640  )
641  )
642  )
643  )
644  )
645  )
646  )
647  )
648  )
649  )
649  )
650  )
651  )
652  )
653  )
654  )
655  )
656  )
657  )
658  )
659  )
659  )
660  )
661  )
662  )
663  )
664  )
665  )
666  )
667  )
668  )
669  )
669  )
670  )
671  )
672  )
673  )
674  )
675  )
676  )
677  )
678  )
679  )
679  )
680  )
681  )
682  )
683  )
684  )
685  )
686  )
687  )
688  )
689  )
689  )
690  )
691  )
692  )
693  )
694  )
695  )
696  )
697  )
698  )
698  )
699  )
700  )
701  )
702  )
703  )
704  )
705  )
706  )
707  )
708  )
709  )
709  )
710  )
711  )
712  )
713  )
714  )
715  )
716  )
717  )
718  )
719  )
719  )
720  )
721  )
722  )
723  )
724  )
725  )
726  )
727  )
728  )
729  )
729  )
730  )
731  )
732  )
733  )
734  )
735  )
736  )
737  )
738  )
739  )
739  )
740  )
741  )
742  )
743  )
744  )
745  )
746  )
747  )
748  )
749  )
749  )
750  )
751  )
752  )
753  )
754  )
755  )
756  )
757  )
758  )
759  )
759  )
760  )
761  )
762  )
763  )
764  )
765  )
766  )
767  )
768  )
769  )
769  )
770  )
771  )
772  )
773  )
774  )
775  )
776  )
777  )
778  )
779  )
779  )
780  )
781  )
782  )
783  )
784  )
785  )
786  )
787  )
788  )
789  )
789  )
790  )
791  )
792  )
793  )
794  )
795  )
796  )
797  )
798  )
798  )
799  )
800  )
801  )
802  )
803  )
804  )
805  )
806  )
807  )
808  )
809  )
809  )
810  )
811  )
812  )
813  )
814  )
815  )
816  )
817  )
818  )
819  )
819  )
820  )
821  )
822  )
823  )
824  )
825  )
826  )
827  )
828  )
829  )
829  )
830  )
831  )
832  )
833  )
834  )
835  )
836  )
837  )
838  )
839  )
839  )
840  )
841  )
842  )
843  )
844  )
845  )
846  )
847  )
848  )
849  )
849  )
850  )
851  )
852  )
853  )
854  )
855  )
856  )
857  )
858  )
859  )
859  )
860  )
861  )
862  )
863  )
864  )
865  )
866  )
867  )
868  )
869  )
869  )
870  )
871  )
872  )
873  )
874  )
875  )
876  )
877  )
878  )
879  )
879  )
880  )
881  )
882  )
883  )
884  )
885  )
886  )
887  )
888  )
888  )
889  )
889  )
890  )
891  )
892  )
893  )
894  )
895  )
896  )
897  )
898  )
898  )
899  )
900  )
901  )
902  )
903  )
904  )
905  )
906  )
907  )
908  )
909  )
909  )
910  )
911  )
912  )
913  )
914  )
915  )
916  )
917  )
918  )
919  )
919  )
920  )
921  )
922  )
923  )
924  )
925  )
926  )
927  )
928  )
929  )
929  )
930  )
931  )
932  )
933  )
934  )
935  )
936  )
937  )
938  )
939  )
939  )
940  )
941  )
942  )
943  )
944  )
945  )
946  )
947  )
948  )
949  )
949  )
950  )
951  )
952  )
953  )
954  )
955  )
956  )
957  )
958  )
959  )
959  )
960  )
961  )
962  )
963  )
964  )
965  )
966  )
967  )
968  )
969  )
969  )
970  )
971  )
972  )
973  )
974  )
975  )
976  )
977  )
978  )
978  )
979  )
980  )
981  )
982  )
983  )
984  )
985  )
986  )
987  )
988  )
988  )
989  )
989  )
990  )
991  )
992  )
993  )
994  )
995  )
996  )
997  )
998  )
999  )
999  )
1000  )
```

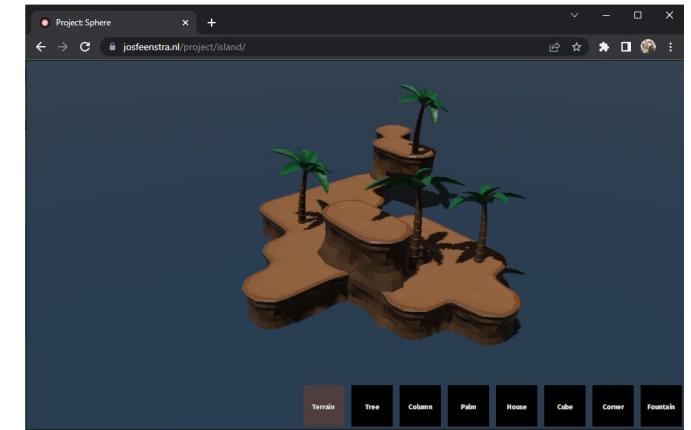
wasm rendered at `wat`
src: author

3. WebAssembly

Use case 1: Run native code on the web



src: audacity

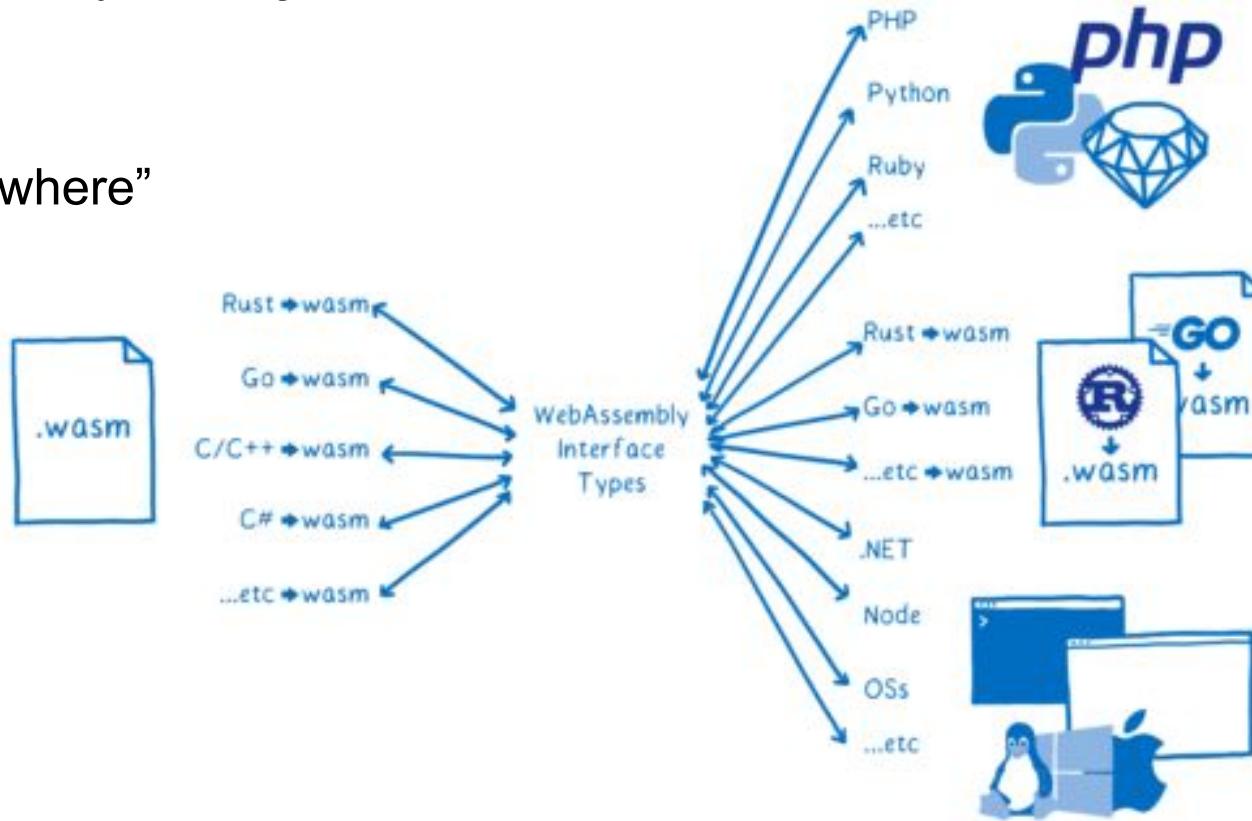


src: author

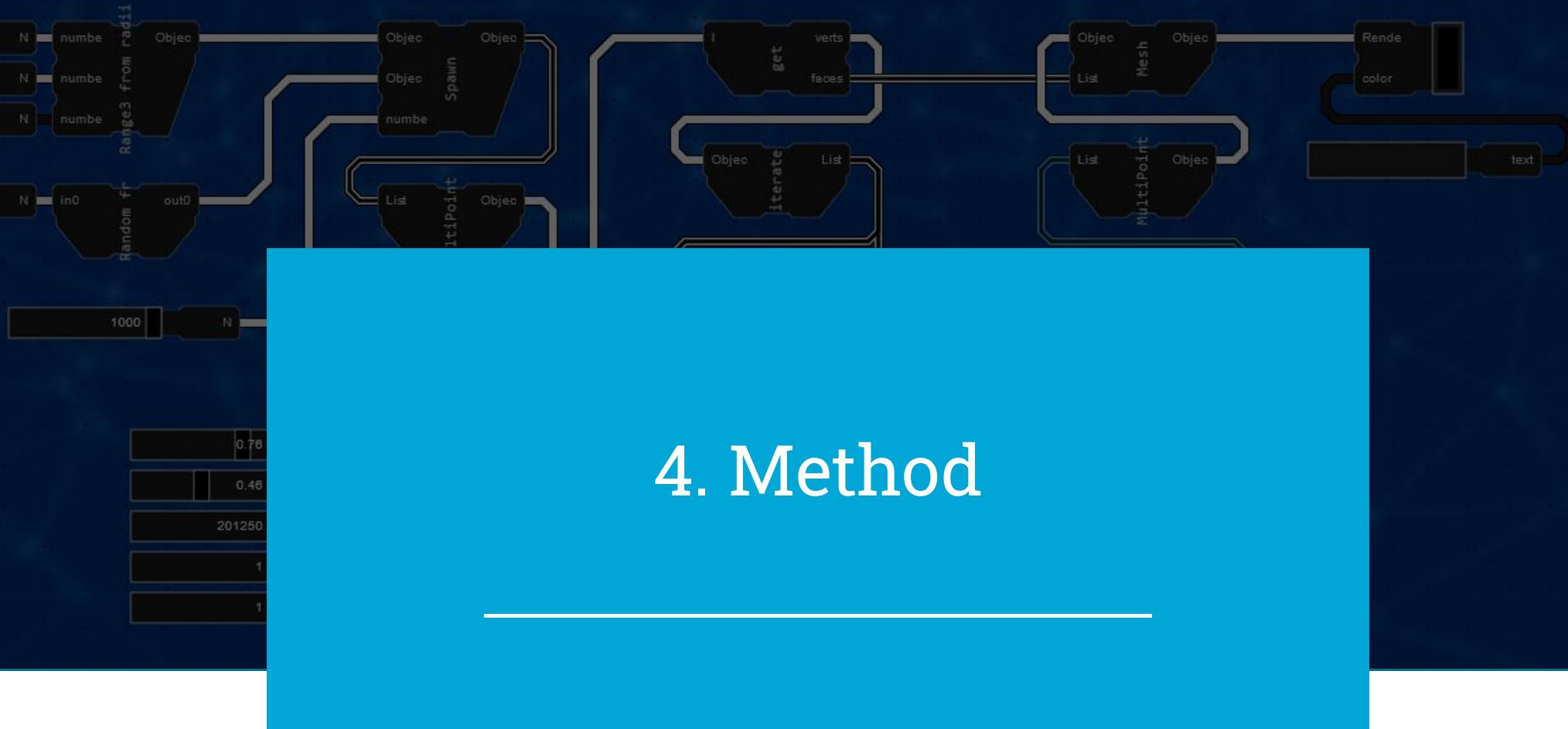
3. WebAssembly

Use case 2: Generic library binding

- Interface Types
- “run anything anywhere”



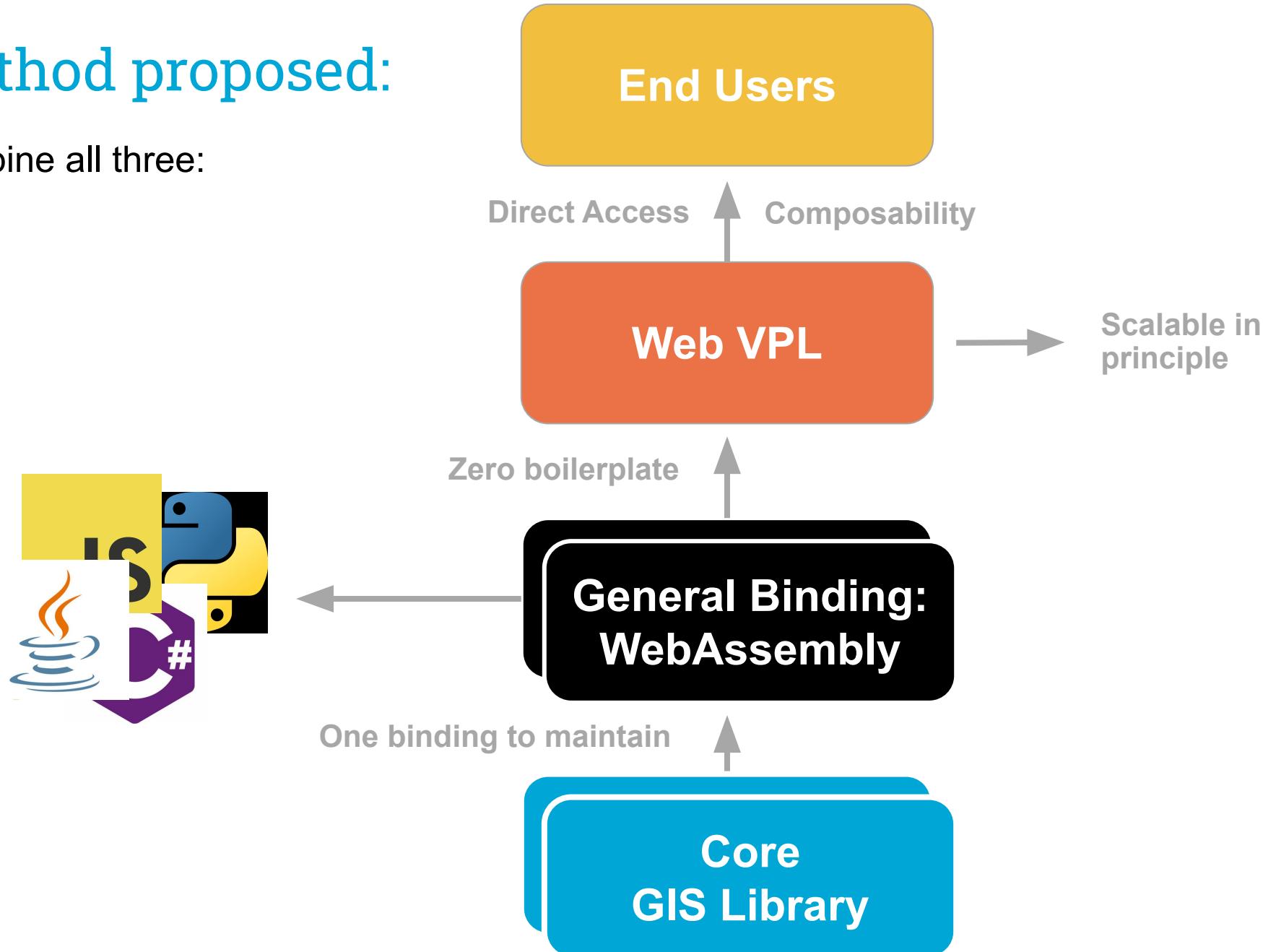
Clark, L. (2019)



4. Method

Method proposed:

Combine all three:

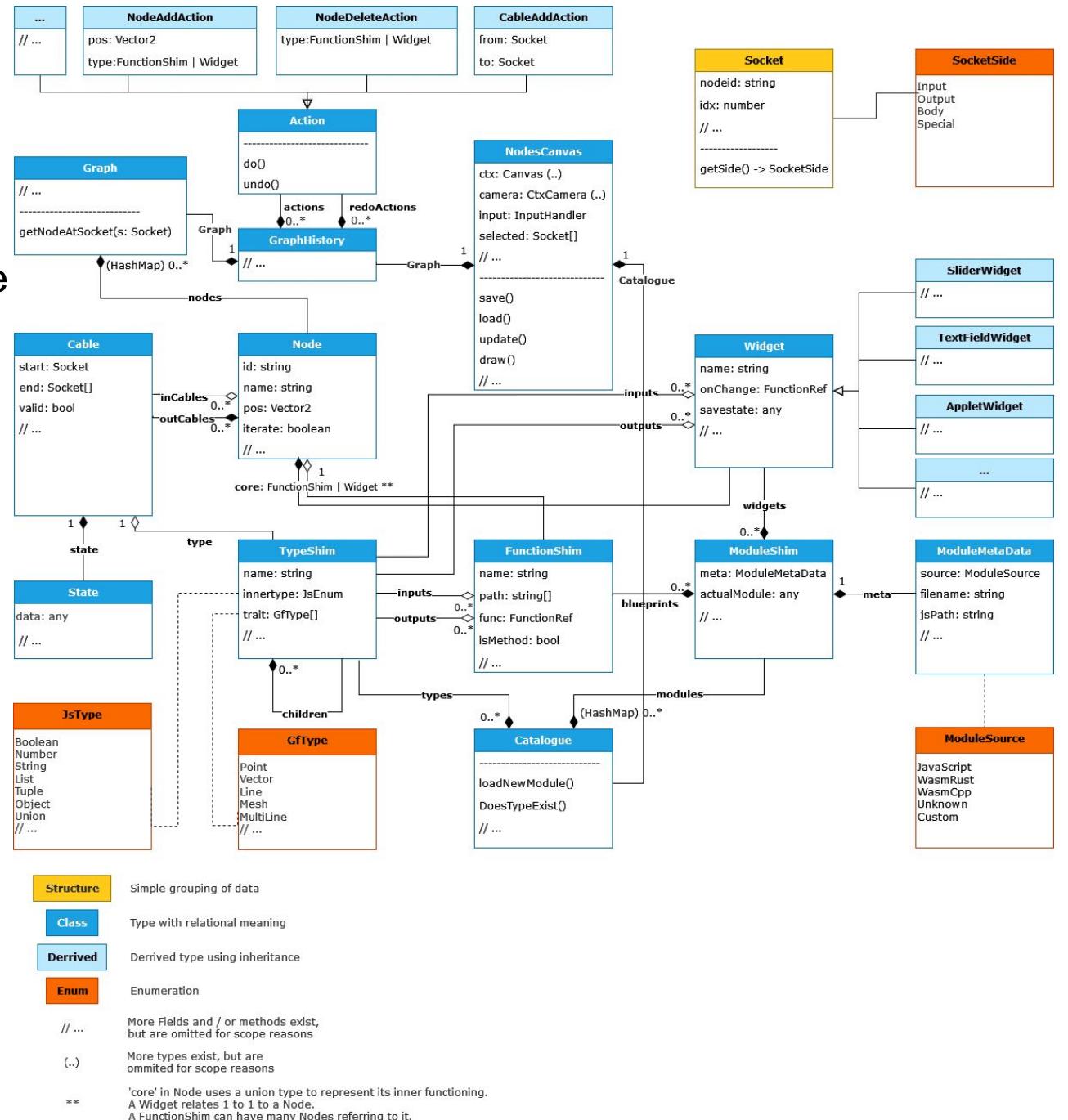


Two components

1. Web VPL
2. Library Plugin system
 - Plugin loader
 - Plugin model

1. Web VPL: Design

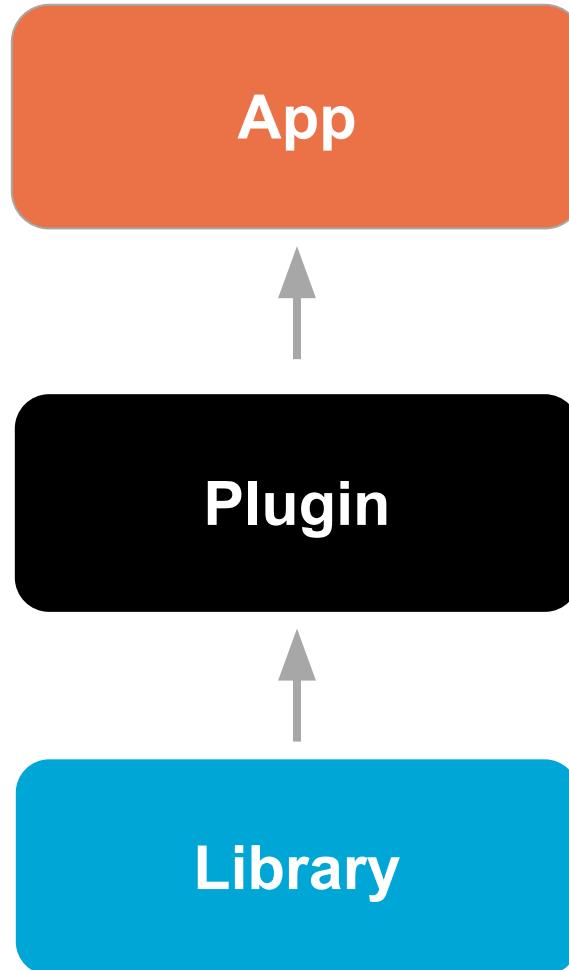
- Essentially, a programming language
- Model View Controller
 - Model : syntax tree



2. Plugin System

Regular case:

- Maintain separate project
- Explicitly state interface
- ‘boilerplate’



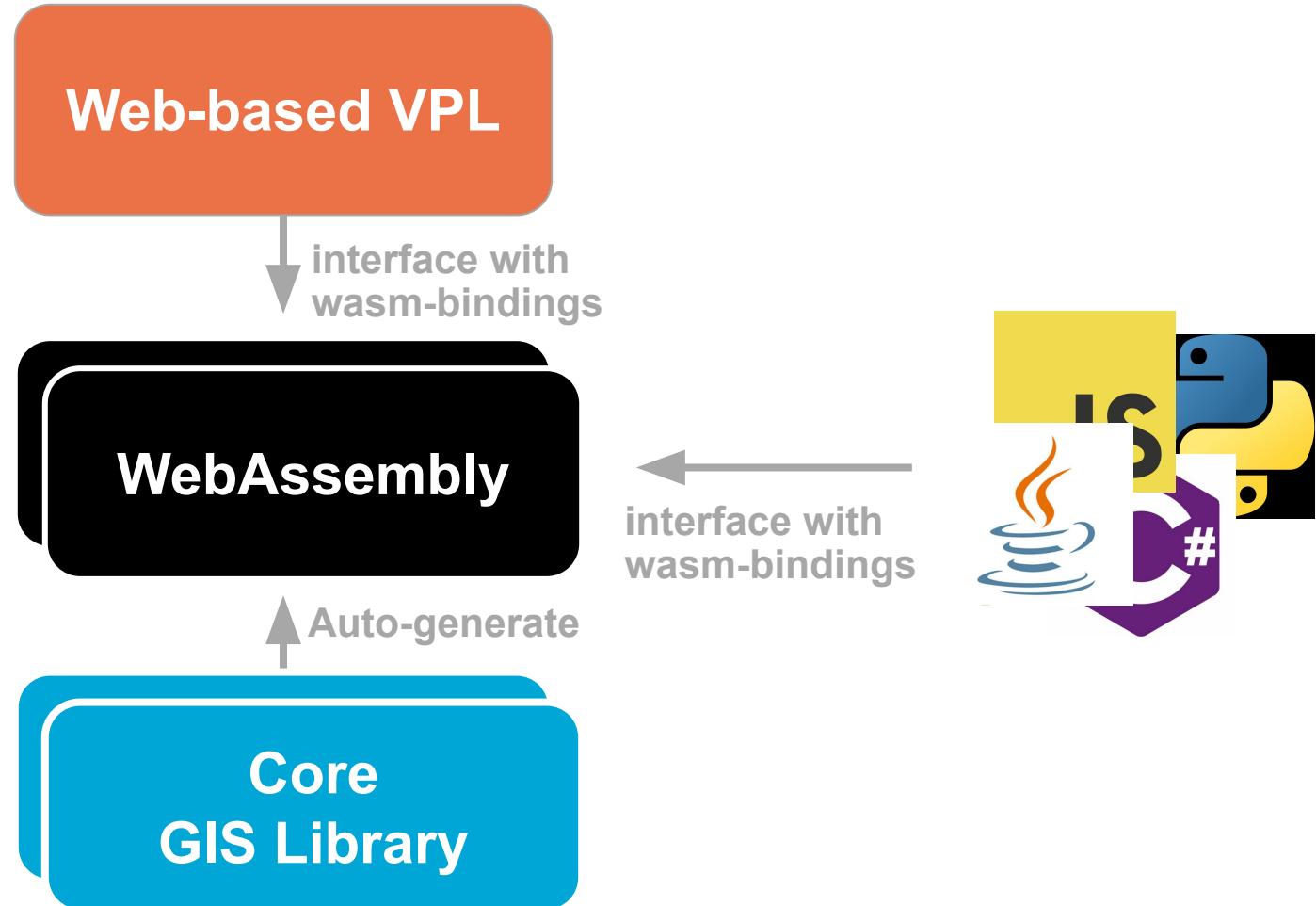
2. Plugin System

Our case:

- Leverage wasm compilers
- Mimic normal language
- Interpret bindings implicitly

Leads to:

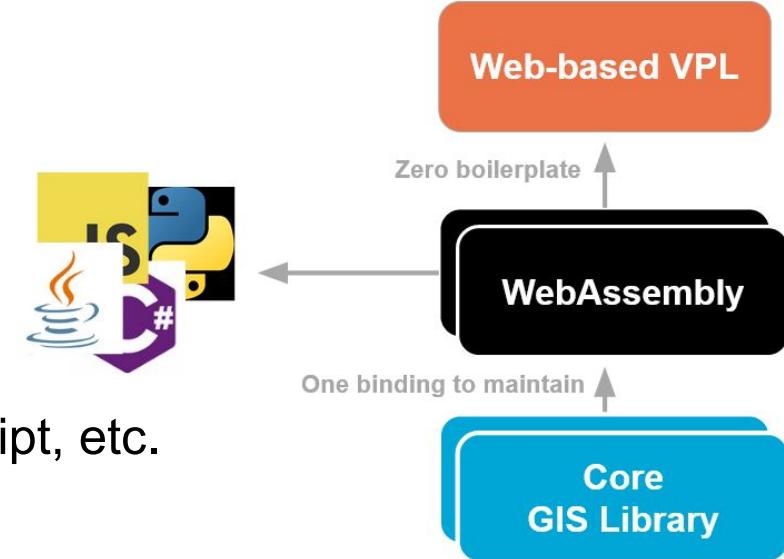
- No boilerplate
- Connect to existing infrastructure



2. Plugin System

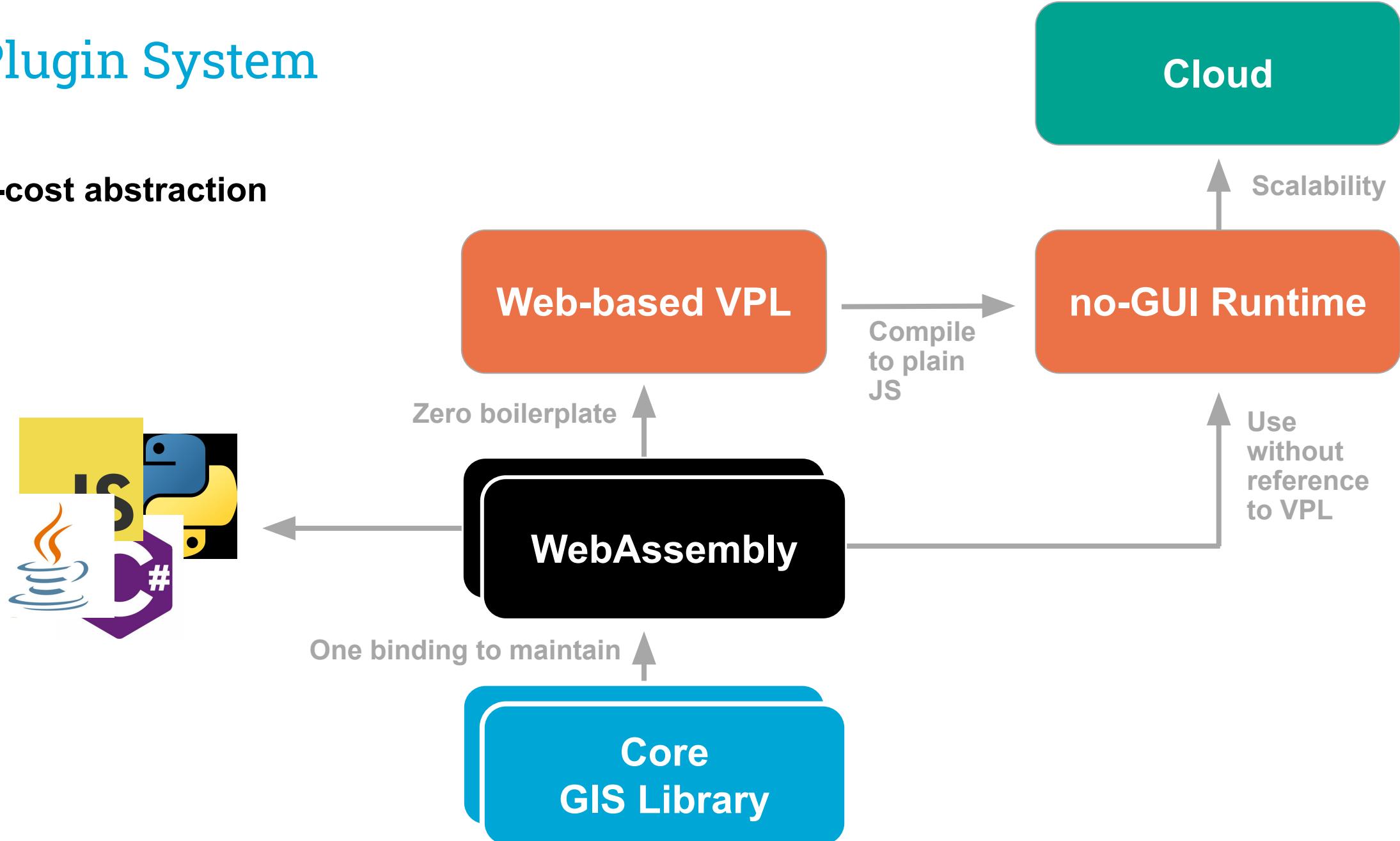
Three elements:

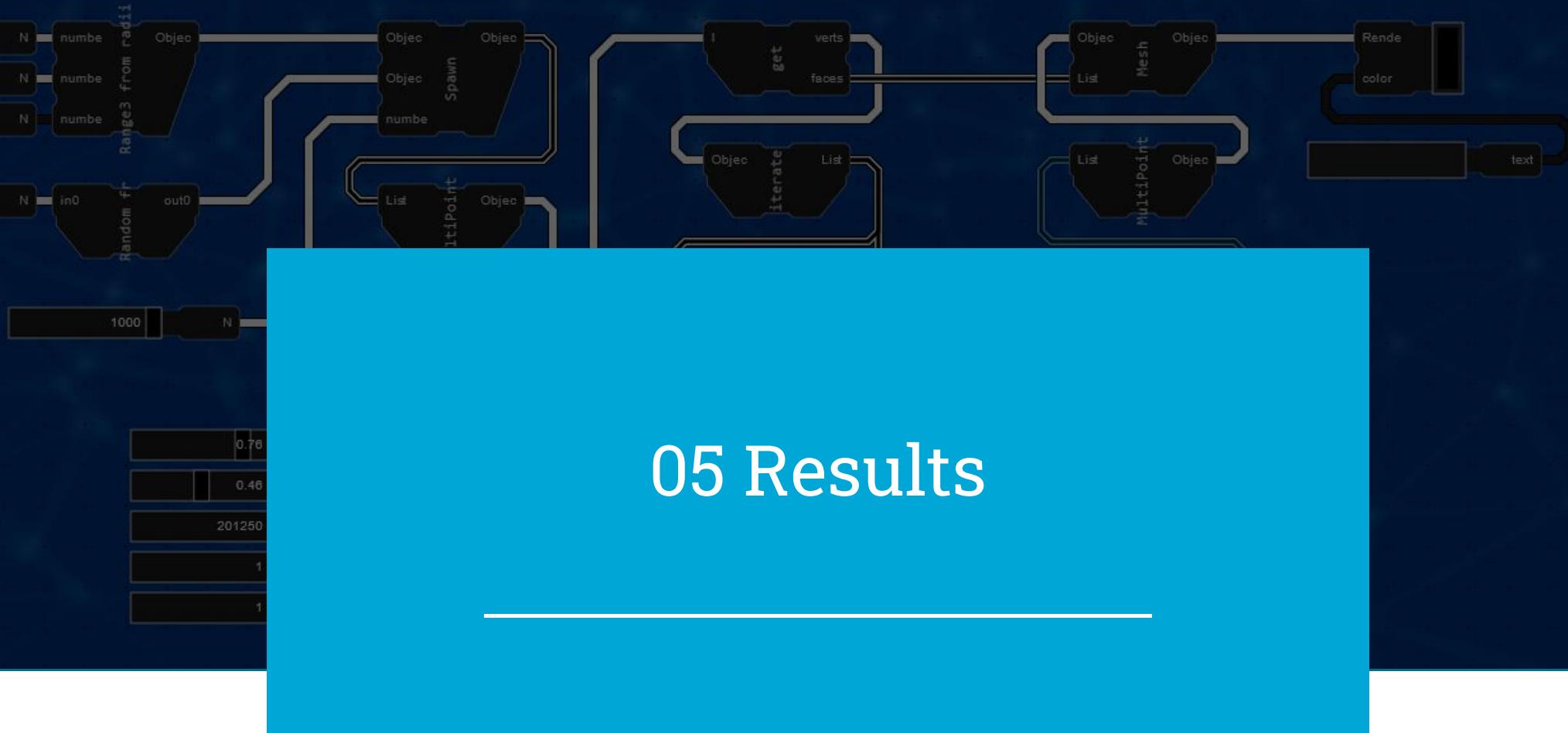
- **Direct utilization** → Zero boilerplate
 - Leverage generic interface properties of WebAssembly
- **Portability**
 - Same behavior within this VPL as in python, C#, JavaScript, etc.
- **Scalability**
 - Zero-cost abstraction



2. Plugin System

Zero-cost abstraction





05 Results

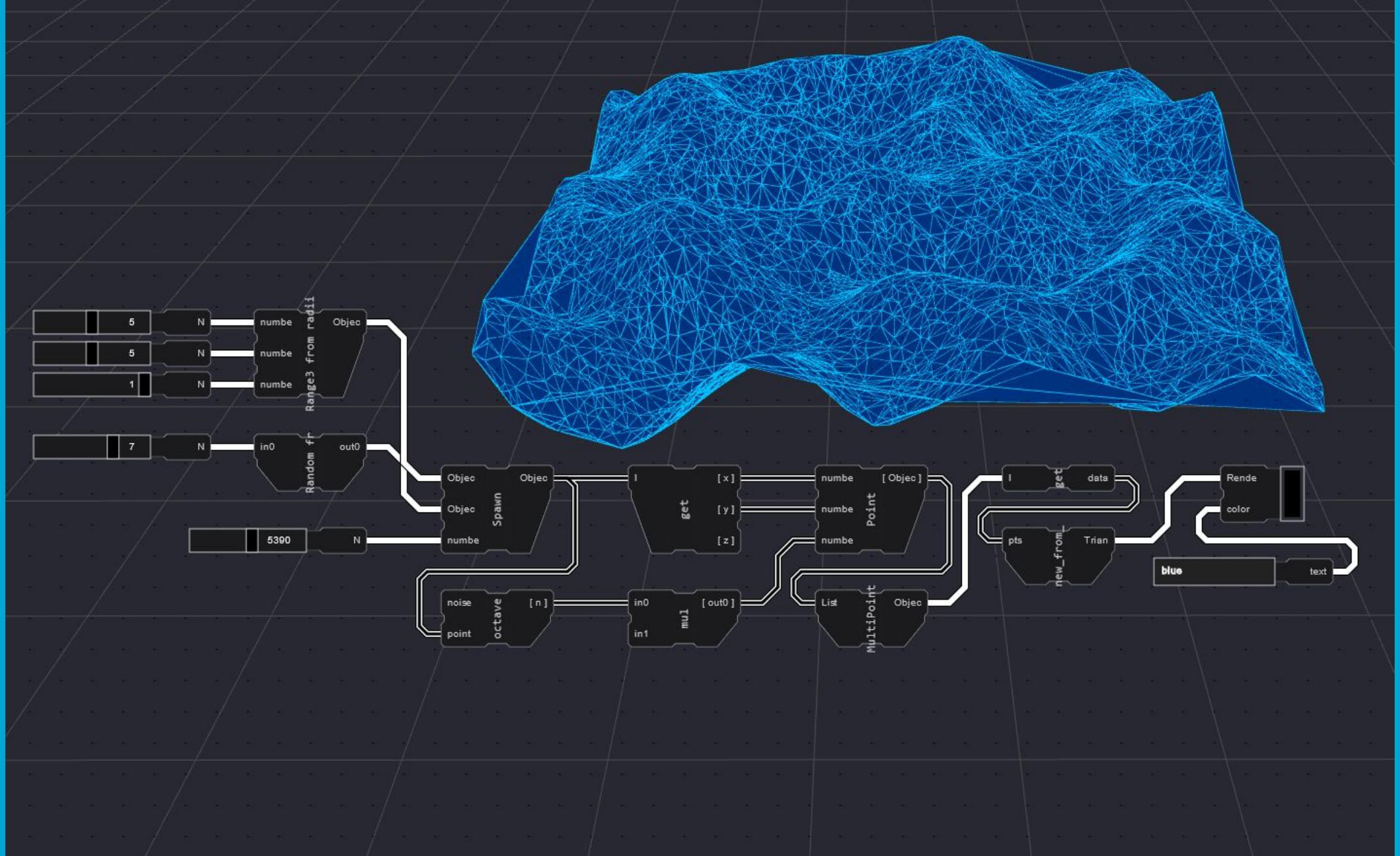
1. Web VPL implementation
2. Library Plugin System implementation

1. Web VPL implementation: Geofront

Web VPL: Geofront

Custom implementation needed to meet all aspects of the method.

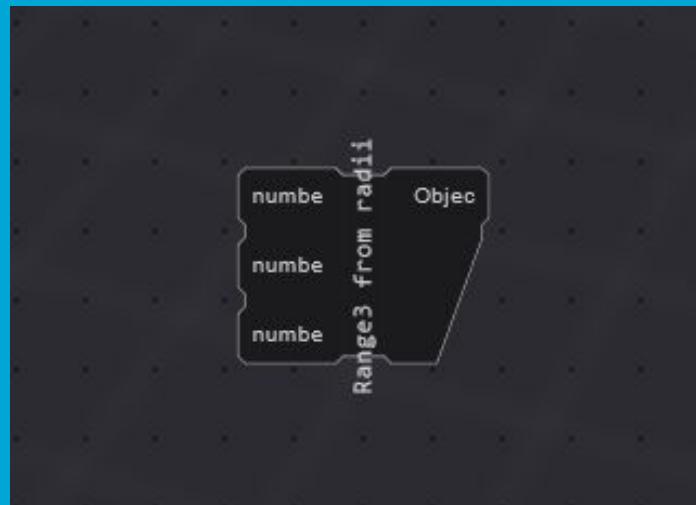
- Application framework
- VPL model implementation
- renderer, Interaction, UI, etc.



Main components

Node

Computation | Function



Cables

Variable | edges



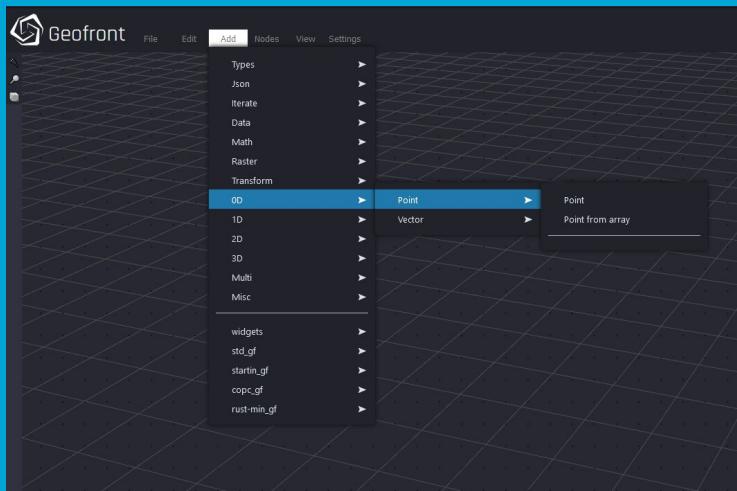
Widget

GUI | Input Output

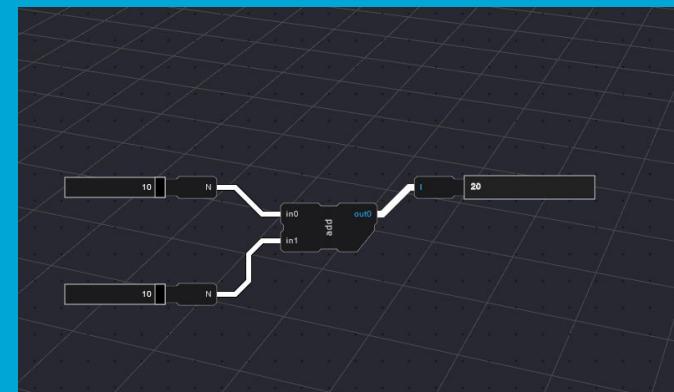


Workflow

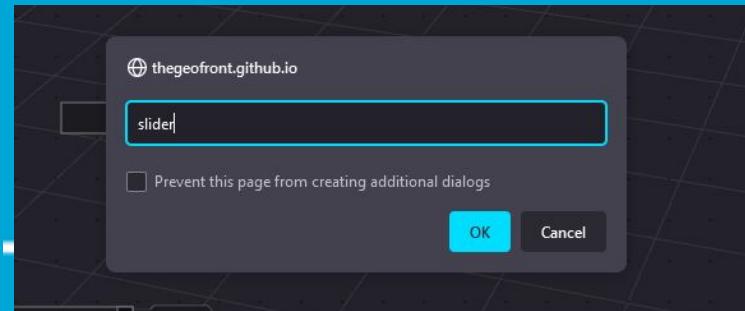
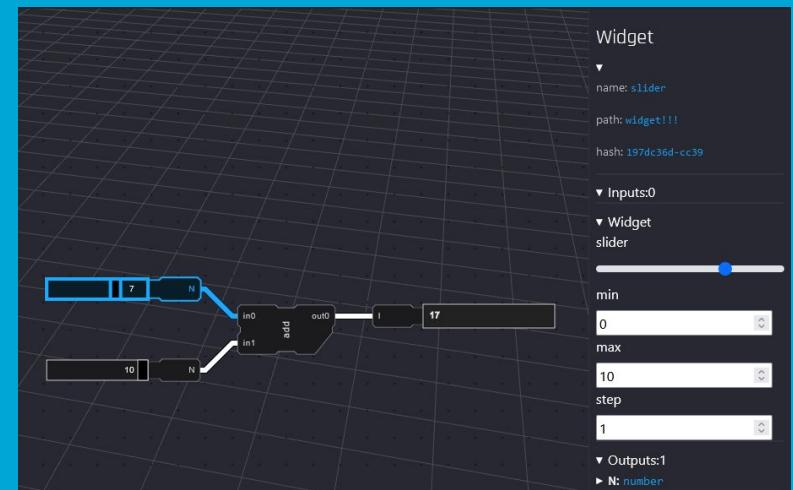
1. Add a node or widget from 'add' dropdown or fuzzy finder



2. Connect nodes by dragging input to output sockets, to form graphs



3. To perform calculations, manipulate the input widgets using the canvas GUI, or a side menu



Widgets: Composable GUI

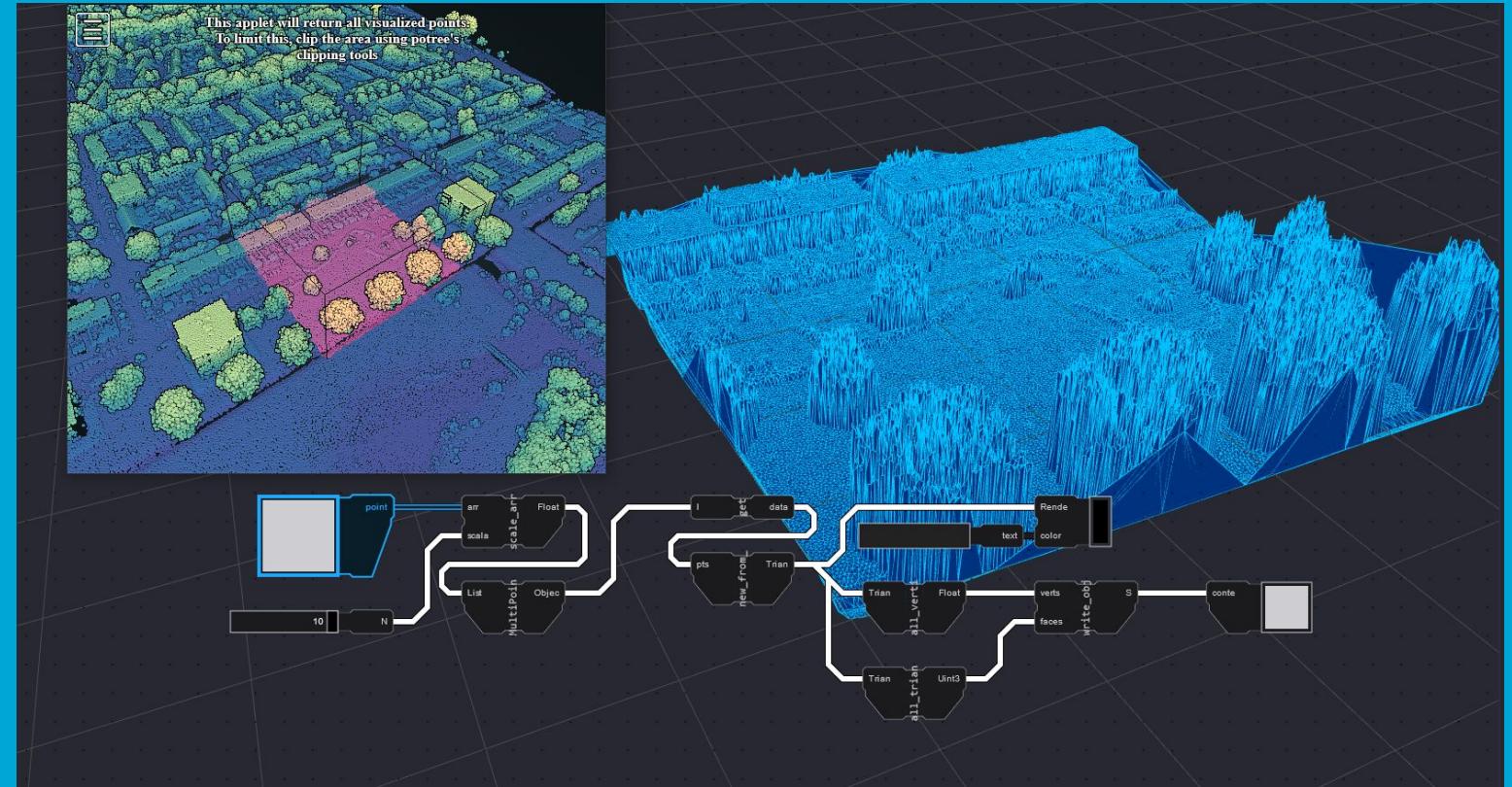
- “Applet”
- Boolean input
- Text field
- Number slider
- Boolean output
- Text output
- Image output
- Renderer



- File save as Blob | String
- File load as Blob | String
- File fetch as Blob | String
- Print to console
- Create list
- Get all properties from object
- Create object from properties

Applet widget: sub-application support

Use output of one application, as input
for Geofront

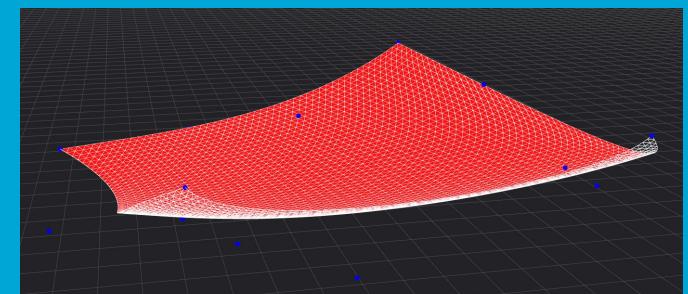
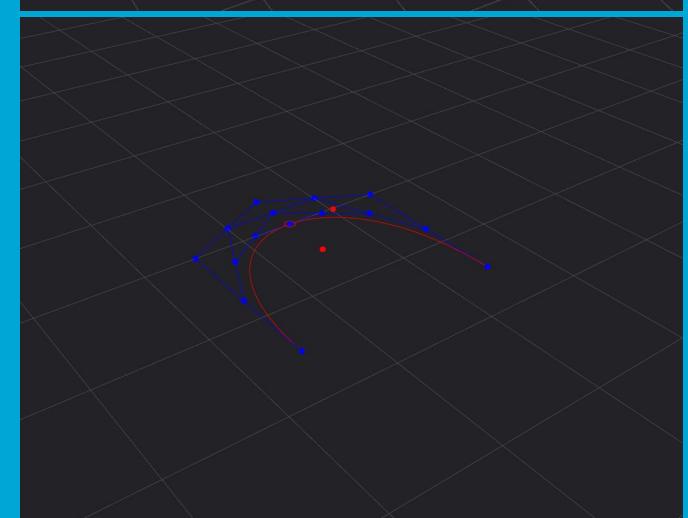
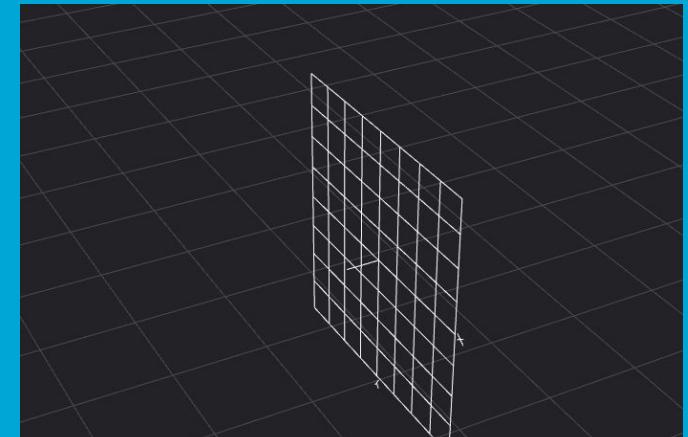
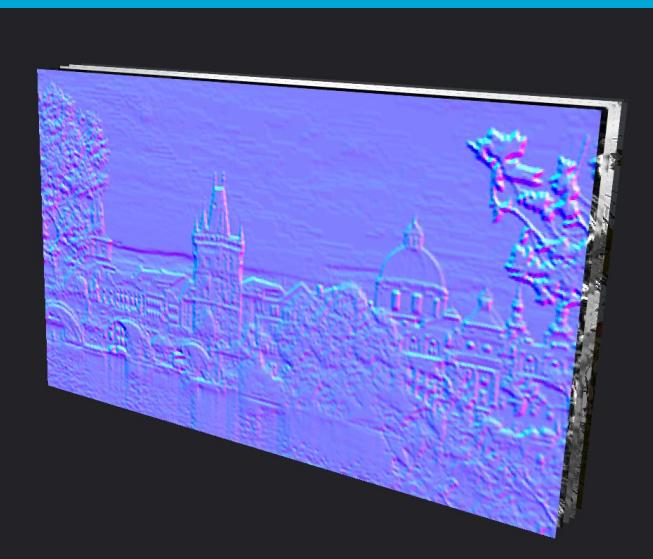
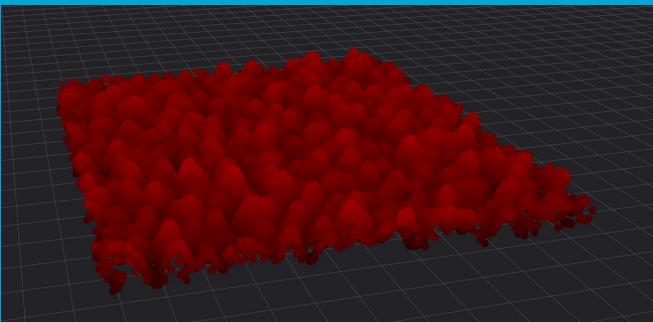
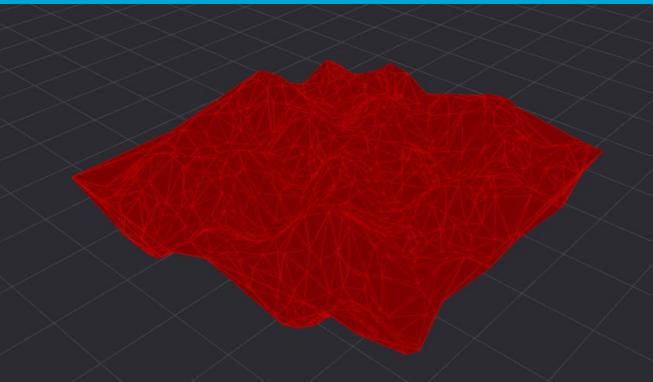


Visualization

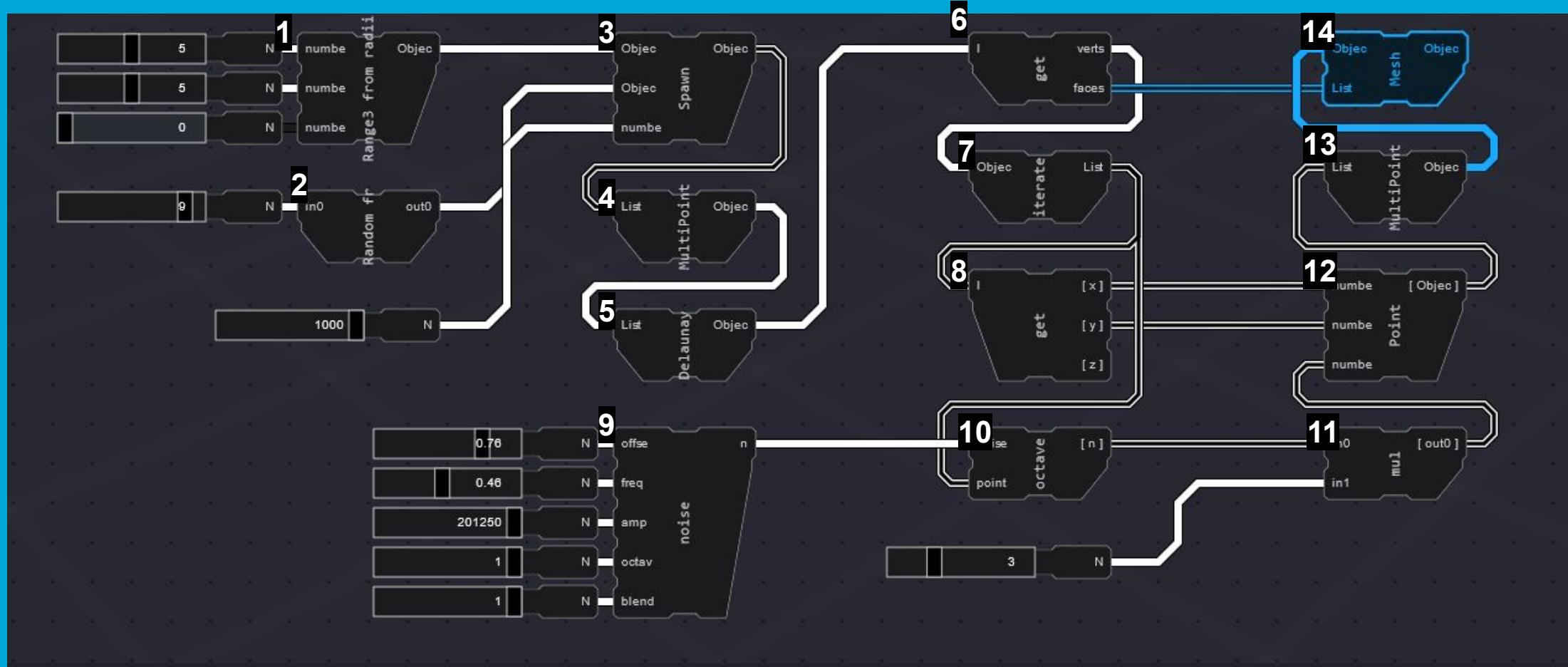
- Custom WebGL implementation

Support for:

- Mesh
- Pointcloud
- Textures (images)
- Plane
- Bezier curve
- Bezier surface



Calculation → Dependency sorting (kahn's algorithm)



Usage 1: Basic interaction

<https://thegeofront.github.io/presentation/videos/geofront-1.mp4>

Usage 2: Basic composition & data inspection

<https://thegeofront.github.io/presentation/videos/geofront-2.mp4>

Usage 3: A larger setup & parametrization

<https://thegeofront.github.io/presentation/videos/geofront-3.mp4>

Usage 4: Geodata input → Obj output

<https://thegeofront.github.io/presentation/videos/geofront-4.mp4>

Implementation: results

- + All major requirements able to be implemented on the web.
- + Does provide application composability
- Limited STD
- Types not interoperable
- Limited performance

Geofront: Feature comparison

Unique combination

	<i>Grasshopper</i>	<i>Blender</i>	<i>Mobius</i>	<i>Geoflow</i>	<i>Geofront</i>
Plugin support	Yes	No*	No	Yes	Yes
Plugin language	C#	-	-	C++	Rust/Js/Ts**
Plugin types	Partially	No	No	Unknown	Yes
Headless runtime	No	No	No	Yes	No
Web based	No	No	Yes	No	Yes
Base GIS Nodes	No	No	Yes	Yes	No
GUI nodes	Yes	Yes	No	No	Yes

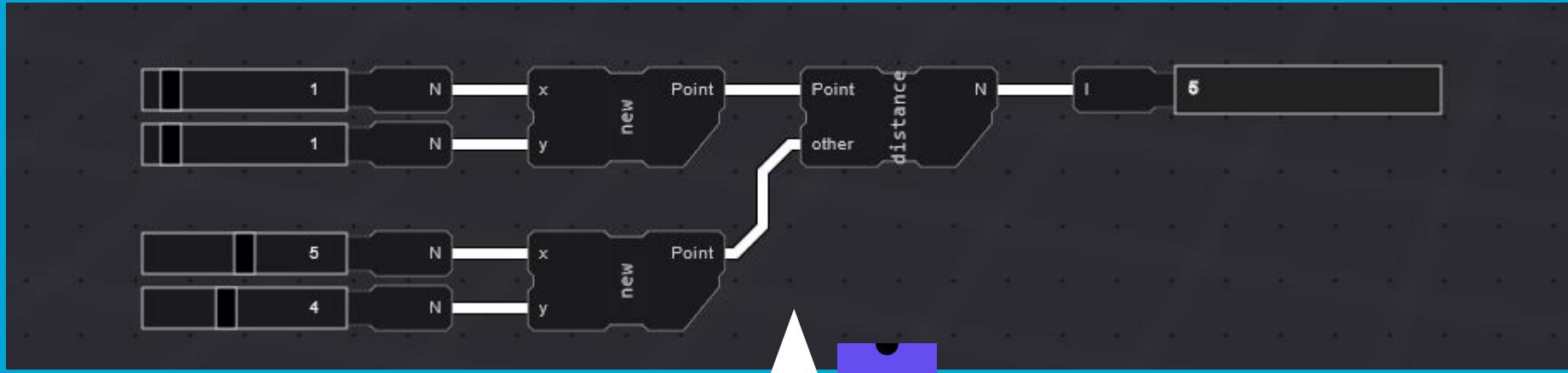
2. Library Plugin System implementation

Plugin System: Implementation

Automated extraction of:

- A list of all functions present in the library
- A list of all custom types (structs / classes) present in the library
- Per function:
 - A list of all input parameters, name and type
 - An output type

Plugin System: Results



```
8 #[wasm_bindgen]
9 pub struct Point {
10     x: f32,
11     y: f32,
12 }
13
14 #[wasm_bindgen]
15 impl Point {
16
17     pub fn new(x: f32, y: f32) -> Self {
18         Self { x, y }
19     }
20
21     pub fn distance(&self, other: &Self) -> f32 {
22         ((self.x - other.x).powi(2) + (self.y - other.y).powi(2)).powf(0.5)
23     }
24 }
```



Plugin System: Comparison

```
1 namespace MyPlugin
2 {
3     public class AdderNode : GH_Component
4     {
5         public ComponentNodeFromString()
6             : base("Add Integers",
7                 "Add",
8                 "This component adds two integer values",
9                 "My Plugin",
10                "My Plugin Category")
11        {
12        }
13
14        protected override void RegisterInputParams(GH_Component.GH_
15        InputParamManager pManager)
16        {
17            pManager.AddIntegerParameter("a", "value A", GH_ParamAccess.item);
18
19            pManager.AddIntegerParameter("b", "value B", GH_ParamAccess.item);
20
21        }
22
23        protected override void RegisterOutputParams(GH_Component.GH_
24        OutputParamManager pManager)
25        {
26            pManager.AddIntegerParameter("R", "result", GH_ParamAccess.item);
27        }
28
29        protected override void SolveInstance(IGH_DataAccess DA)
30        {
31            int a;
32            int b;
33            DA.GetData(0, ref a);
34            DA.GetData(1, ref b);
35            int c = a + b;
36            DA.SetData(0, c);
37
38        }
39
40    }
```

```
1 class AddNode : public Node
2 {
3     public:
4         using Node::Node;
5
6         void init()
7         {
8             add_input("a", typeid(int));
9             add_input("b", typeid(int));
10            add_output("result", typeid(int));
11        }
12
13        std::string info()
14        {
15            std::string s;
16            if (output("result").has_data())
17                s = std::to_string(output("result").get<int>());
18            return s;
19        }
20
21        void process()
22        {
23            auto in1 = input("a").get<int>();
24            auto in2 = input("b").get<int>();
25            std::this_thread::sleep_for(std::chrono::microseconds(200));
26            output("result").set(int(in1 + in2));
27        }
28};
```

Figure 55: Geoflow plugin

```
1 #[wasm_bindgen]
2 fn add(a: i32, b: i32) -> i32 {
3     a + b
4 }
```

Figure 56: Geofront plugin



Plugin System: Tests

C++ → emscripten → WebAssembly

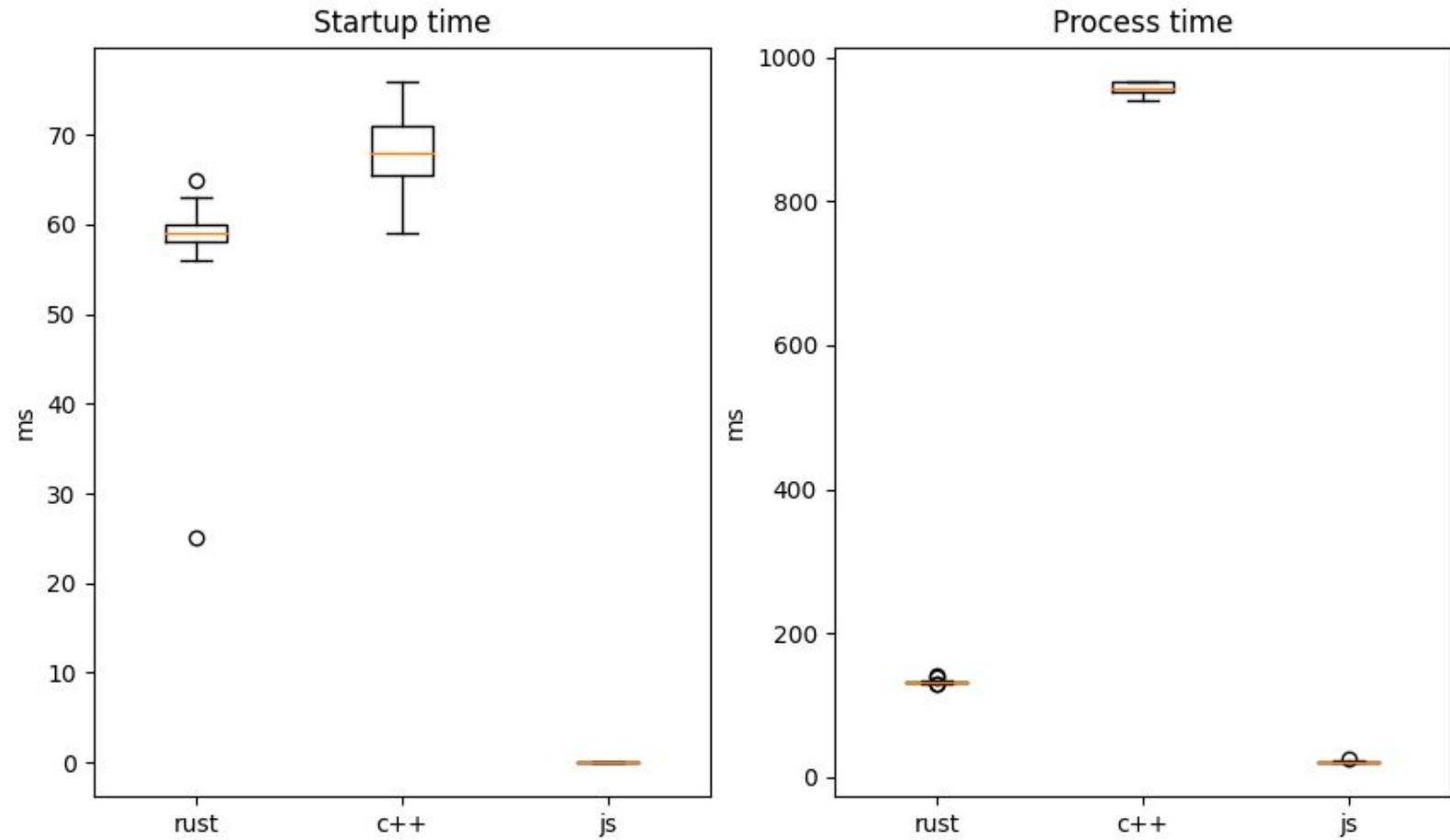
```
lib.cpp M x
plugins > cpp-min-gf > src > lib.cpp
1 // quick_example.cpp
2 #include <emscripten/bind.h>
3 #include <cmath>
4
5 using namespace emscripten;
6
7 float add(float left, float right) {
8     return left + right;
9 }
10
11 class Point {
12 public:
13     double x;
14     double y;
15
16     Point(double x, double y) :
17         x(x),
18         y(y) {}
19
20     double distance(Point& other) {
21         return std::pow(
22             std::pow(x - other.x, 2) + std::pow(y - other.y, 2),
23             0.5);
24     }
25 };
26
27 EMSCRIPTEN_BINDINGS(cpp_min) {
28     function("add", &add);
29     class_<Point>("Point")
30         .constructor<double, double>()
31         .function("distance", &Point::distance)
32         .property("x", &Point::x)
33         .property("y", &Point::y);
34 }
```

Rust → wasm-pack → WebAssembly

```
lib.rs M x
plugins > rust-min-gf > src > lib.rs > {} impl Point
1 use wasm_bindgen::prelude::*;
2
3 #[wasm_bindgen]
4 pub fn add(left: usize, right: usize) -> usize {
5     left + right
6 }
7
8 #[wasm_bindgen]
9 pub struct Point {
10     x: f32,
11     y: f32,
12 }
13
14 #[wasm_bindgen]
15 impl Point {
16     pub fn new(x: f32, y: f32) -> Self {
17         Self { x, y }
18     }
19
20     pub fn distance(&self, other: &Self) -> f32 {
21         ((self.x - other.x).powi(2) + (self.y - other.y).powi(2)).powf(0.5)
22     }
23 }
24 }
```

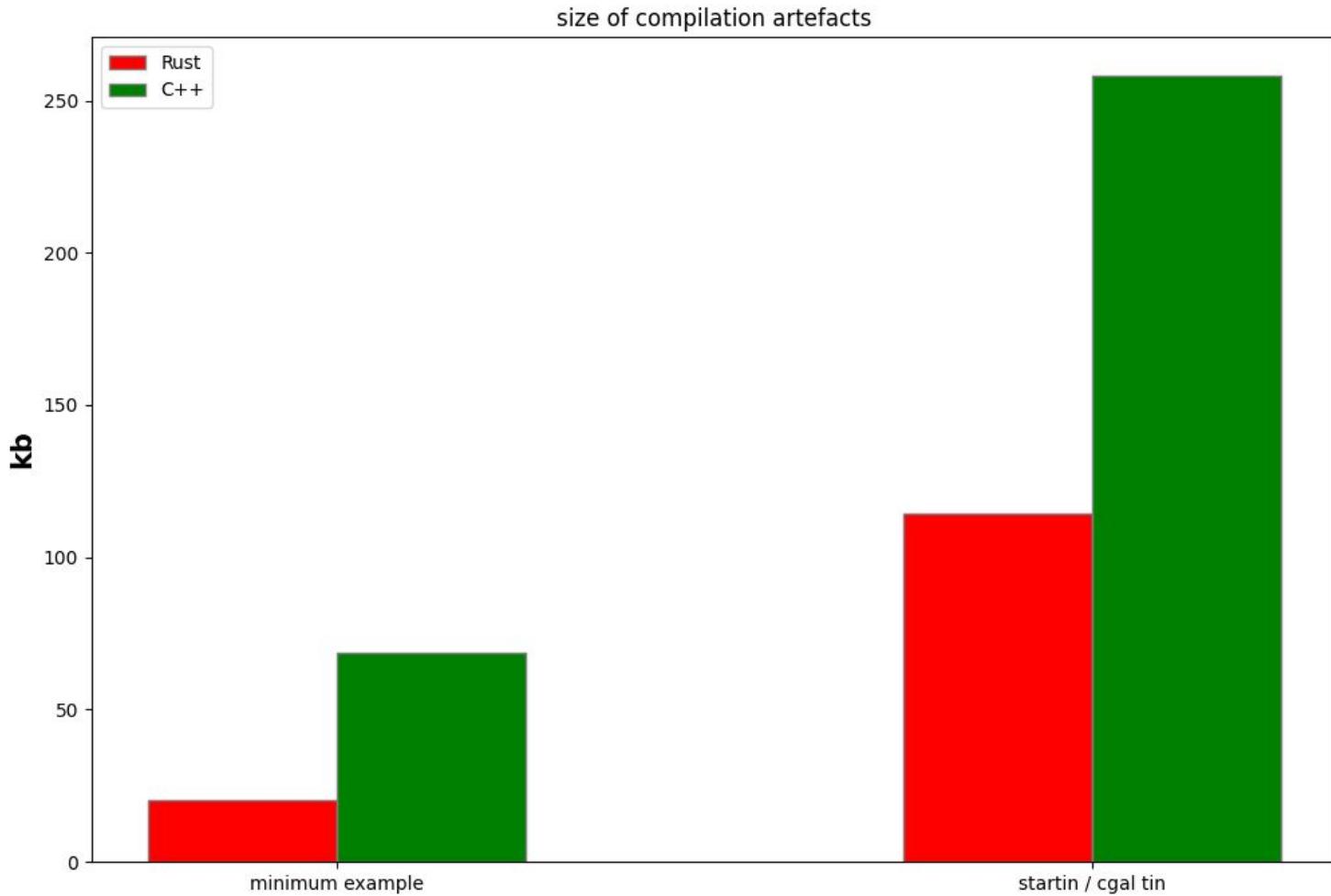
Plugin System: Tests

Interfacing this binary with JavaScript was around **six** times as slow compared to the rust equivalent.



Plugin System: Tests

the emscripten compiler produced a binary which requires more than **three** times the size of the same functionality compiled with wasm-pack.



Plugin System: Test Results

Rust

Worked almost immediately for almost any library

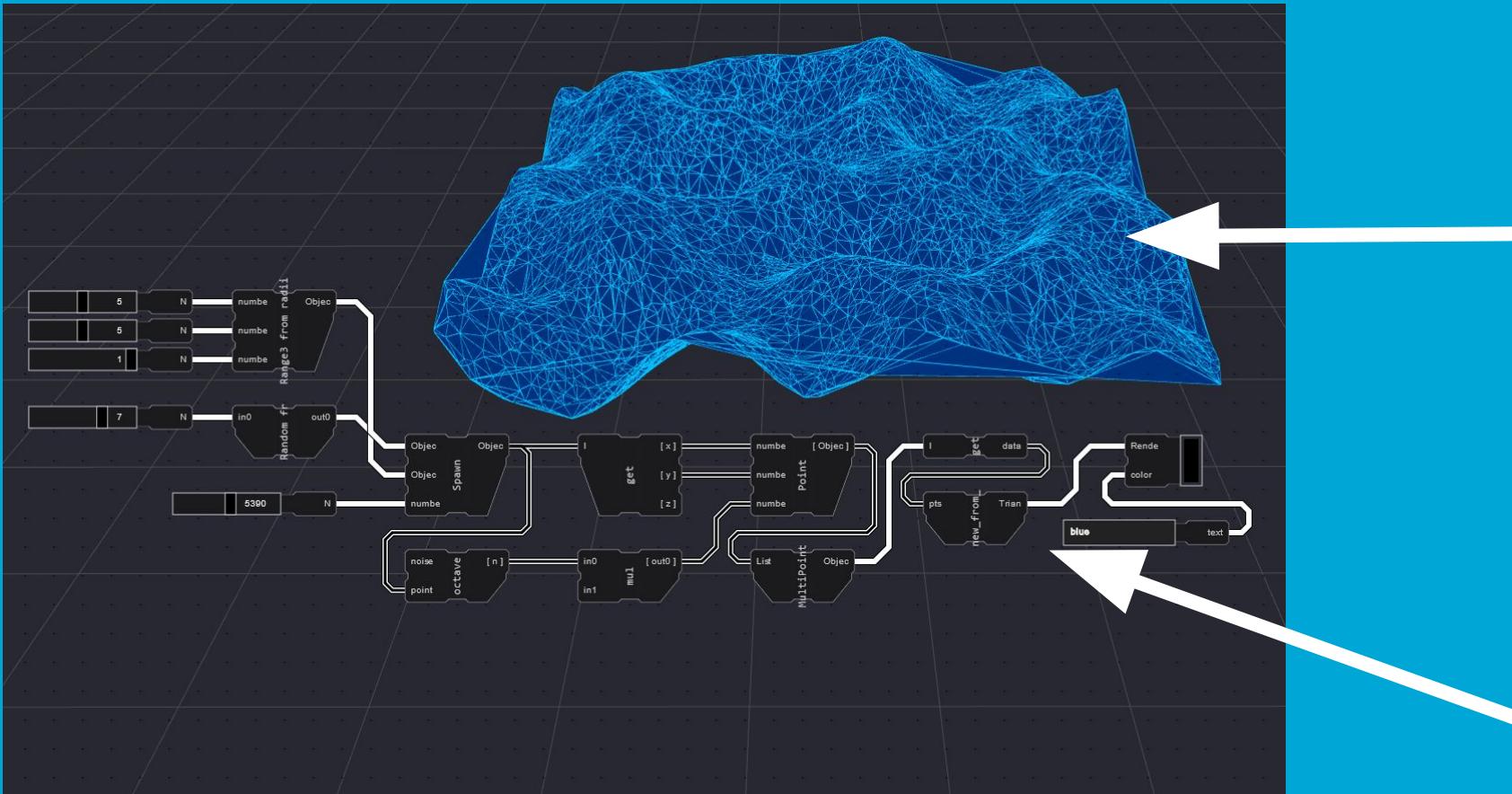
- + Expressive bindings allow complex data types to be exchanged in a simple manner.
- *Still some runtime overhead due to wrappers*

C++

Multiple workarounds eventually allowed some parts of CGAL to be run in geofront, if included in the source code

- *Requires many workarounds*
- *More wrapper overhead than rust*
- *Larger binaries than rust*
- *Sub-optimal support for bindings*
 - + Interface Types will most likely be added in the future to emscripten

Plugin System: Tests: startin



```
// impl Renderable for Triangulation
#[wasm_bindgen]
impl Triangulation {

    pub fn gf_has_trait_renderable() -> bool {
        true
    }

    pub fn gf_get_shader_type() -> GeoShaderType {
        GeoShaderType::Mesh
    }

    pub fn gf_get_buffers(&self) -> JsValue {
        let buffer = MeshBuffer {
            verts: self.all_vertices(),
            cells: self.all_triangles(),
        };
        serde_wasm_bindgen::to_value(&buffer).unwrap()
    }
}
```

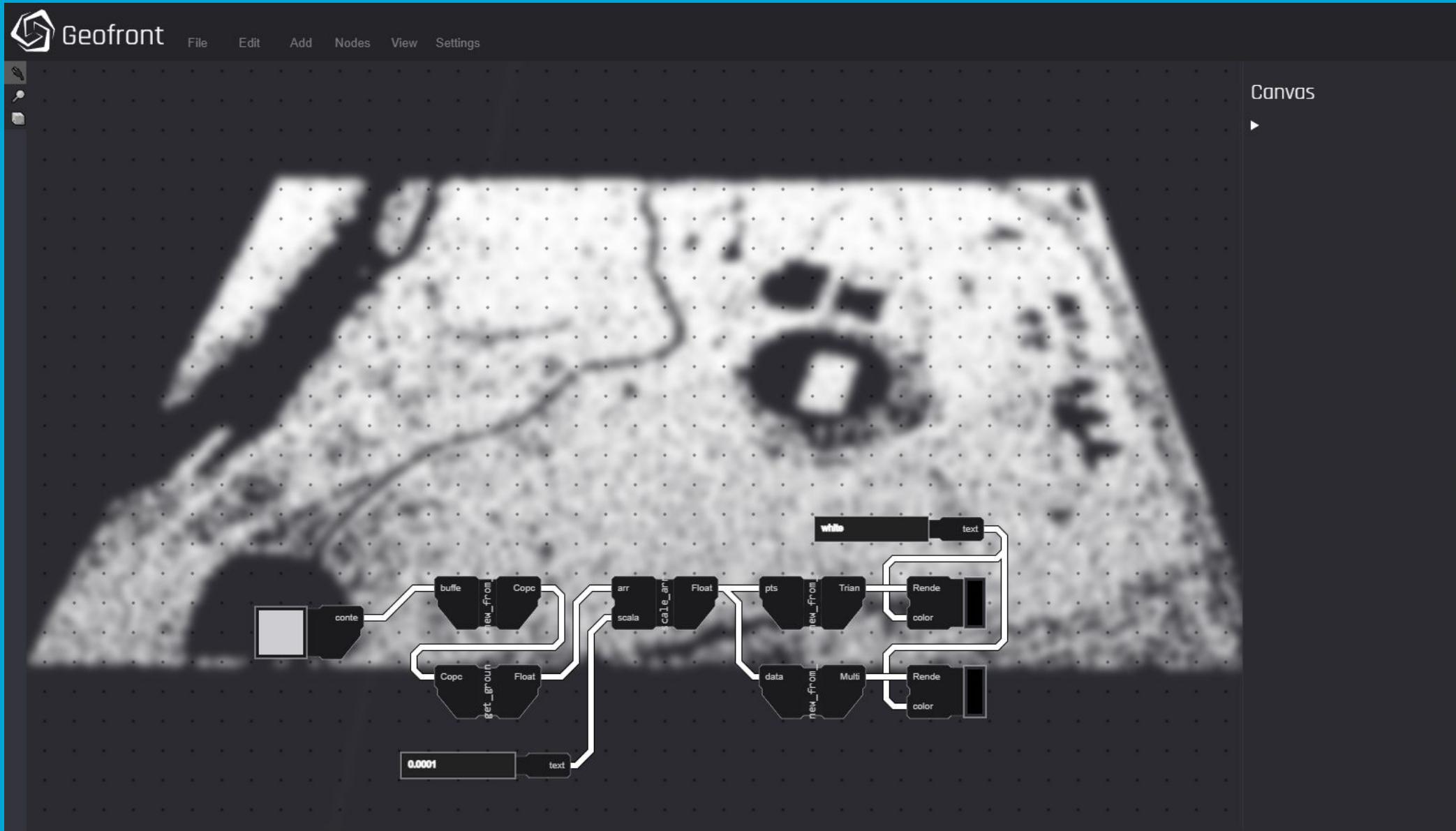
```
#[wasm_bindgen]
pub struct Triangulation {
    dt: startin::Triangulation,
}

#[wasm_bindgen]
impl Triangulation {

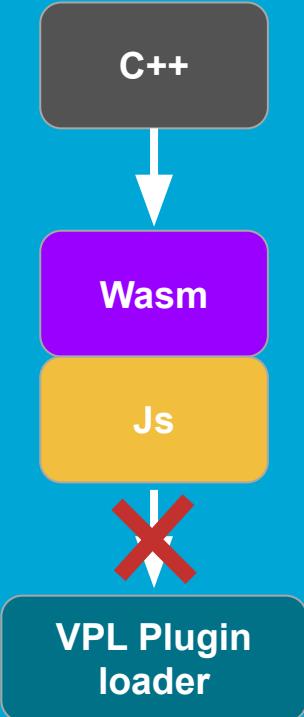
    pub fn new_from_vec(pts: Vec<f64>) -> Triangulation {
        let mut tri = Triangulation::new();
        tri.insert(pts);
        tri
    }

    pub fn new() -> Triangulation {
        let dt = startin::Triangulation::new();
        Triangulation { dt }
    }
}
```

Rust Library: copc-rs (Point cloud loader)

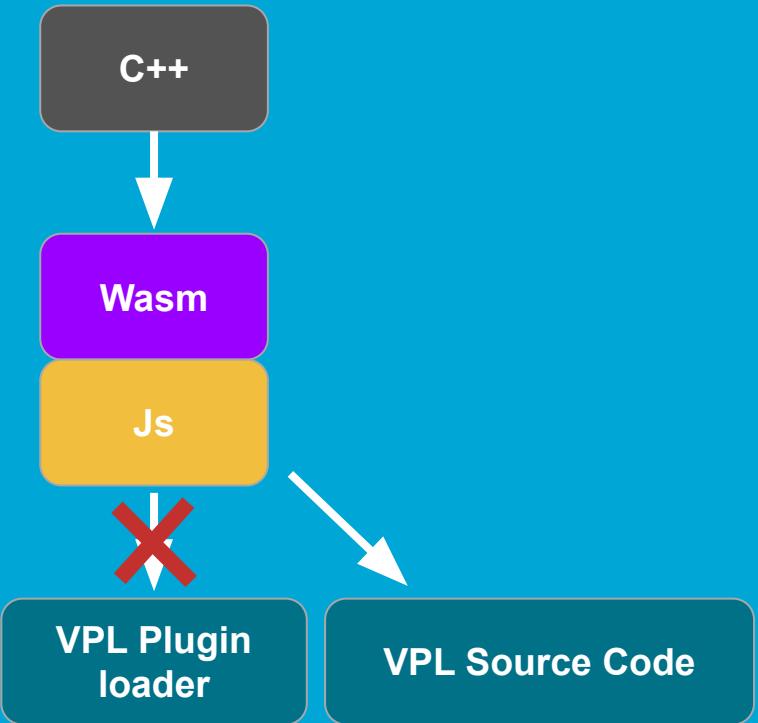


C++ Library: CGAL

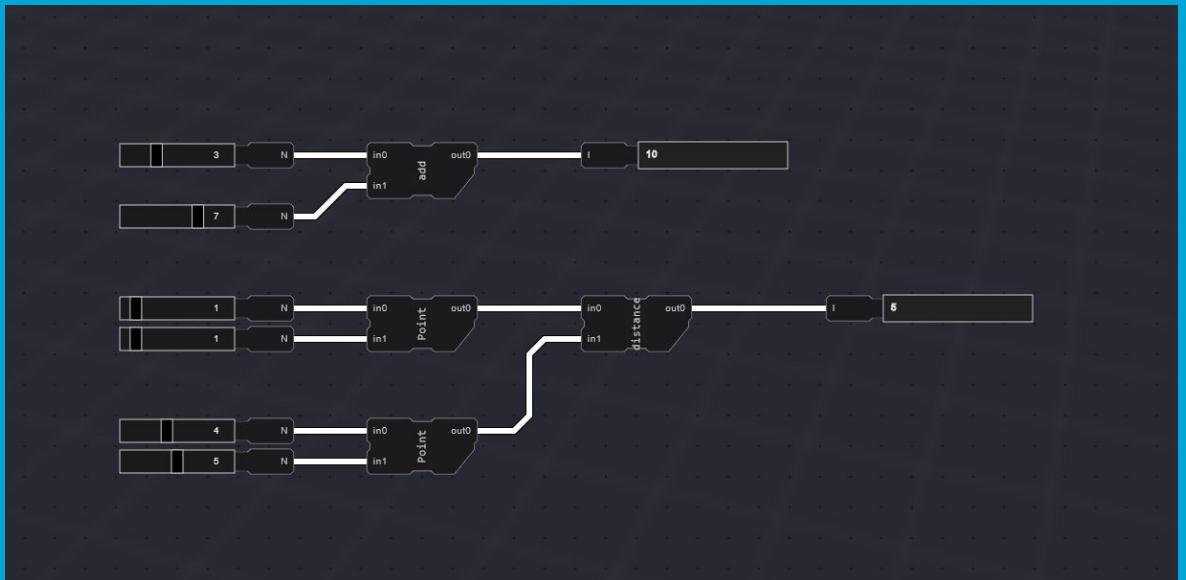


```
lib.cpp M X
plugins > cpp-minify > src > lib.cpp
1 // quick_example.cpp
2 #include <emscripten/bind.h>
3 #include <cmath>
4
5 using namespace emscripten;
6
7 float add(float left, float right) {
8     return left + right;
9 }
10
11 class Point {
12 public:
13     double x;
14     double y;
15
16     Point(double x, double y) :
17         x(x),
18         y(y) {}
19
20     double distance(Point& other) {
21         return std::sqrt(
22             std::pow(x - other.x, 2) + std::pow(y - other.y, 2),
23             0.5);
24     }
25 }
26
27 EMSCRIPTEN_BINDINGS(cpp_min) {
28     function("add", &add);
29     class<Point>"Point"
30         .constructor<double, double>()
31         .function("distance", &Point::distance)
32         .property("x", &Point::x)
33         .property("y", &Point::y);
34 }
```

C++ Library: CGAL

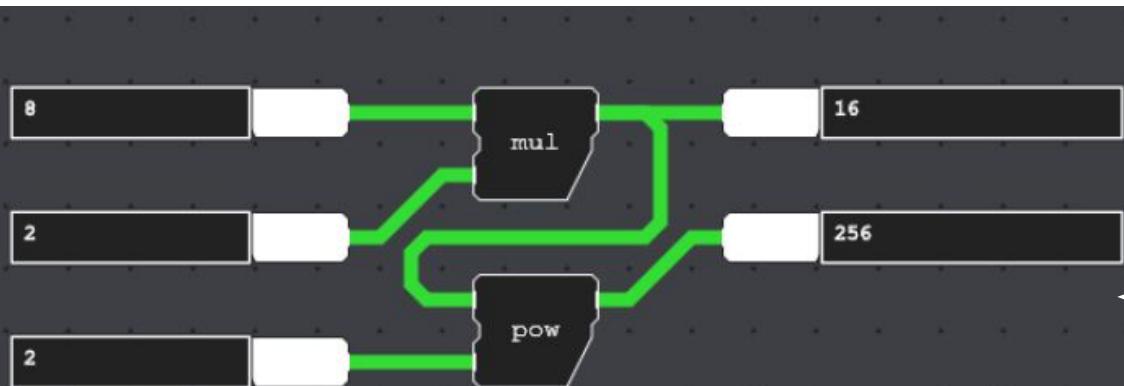


```
lib.cpp M X
plugins > cpp-min glfw > src > lib.cpp
1 // quick_example.cpp
2 #include <emscripten/bind.h>
3 #include <cmath>
4
5 using namespace emscripten;
6
7 float add(float left, float right) {
8     return left + right;
9 }
10
11 class Point {
12 public:
13     double x;
14     double y;
15
16     Point(double x, double y) :
17         x(x),
18         y(y) {}
19
20     double distance(Point& other) {
21         return std::sqrt(
22             std::pow(x - other.x, 2) + std::pow(y - other.y, 2),
23             0.5);
24     }
25 }
26
27 EMSCRIPTEN_BINDINGS(pp_min) {
28     function("add", &Add);
29     class<Point>"Point"
30         .constructor<double, double>()
31         .function("distance", &Point::distance)
32         .property("x", &Point::x)
33         .property("y", &Point::y);
34 }
```



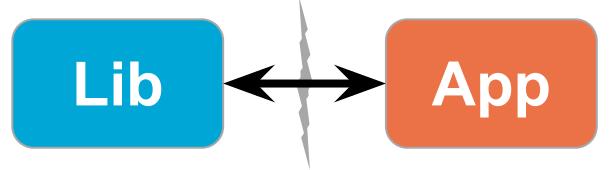
Plugin System: Zero cost abstraction runtime

Currently Incomplete, but promising



```
1 function anonymous(
2   a /* "widget": "text" | "state": "2" | "x": 2 | "y": -10 */,
3   b /* "widget": "text" | "state": "2" | "x": 2 | "y": -12 */,
4   c /* "widget": "text" | "state": "8" | "x": 2 | "y": -14 */
5 ) {
6   let [d] = math.mul(c, b) /* "x": 6 | "y": -14 */;
7   let [e] = math.pow(d, a) /* "x": 6 | "y": -11 */;
8   return [
9     e /* "widget": "console" | "x": 10 | "y": -11 */,
10    d /* "widget": "console" | "x": 10 | "y": -14 */
11  ];
12 }
13
14
```





sub Q: library & application divide

- To what extent does this method intent to address the **discrepancies** between **software applications and libraries**, as described by Elliott (2007)?

Does it succeed in doing so?

1. Libraries cannot be directly used, end users are dependent on in-between applications →

- + VPL acts as “a custom GUI for any library”
- + Only dependent on Wasm-bindings
- Exception: C++

2. Applications are not further composable →

- + VPL: Use tools in a composable manner
- + Potree demo: further composable web applications

3. Capabilities get lost in in-between steps →

- + Plugin system: Minimum in-between steps
- + Wasm-bindings only limiting factor

A: All aspects were able to be addressed to a certain extent.

Main research question:

Q: Is a web based VPL a viable method for directly accessing native GIS libraries with a composable interface?

Yes

- Provides solutions for app / lib divide
- Successfully implemented and combined:
 - no-boilerplate plugin system
 - Composable GUI
 - Web-based

No

- More research is required to proof full feasibility:
 - C++ → Interface Types
 - GUI-less runtime → Scalability

A: Yes, but with exceptions

Future work

- Improved VPL model:
 - Improved type support
 - Validated Dataflow VPL
 - Concurrency
 - Compile to WebAssembly
- Deployment & scalability
 - Cloud-based execution
 - “Deploy as application”
- Effects of Rust as replacement for C++ in GIS or any scientific endeavor
 - Less error-prone, improved library management, improved wasm support → distribution

Sources:

- Elliott, C. (2007). Tangible functional programming. International Conference on Functional Programming. <http://conal.net/papers/Eros/> Accessed 2022-09-27. Related Talk: <https://www.youtube.com/watch?v=faJ8N0giqzw>.
- Haas, A., Rossberg, A., Schuff, D. L., Titzer, B. L., Holman, M., Gohman, D., Wagner, L., Zakai, A., and Bastien, J. (2017). Bringing the web up to speed with WebAssembly. In Proceedings of the 38th ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI 2017, pages 185–200, New York, NY, USA. Association for Computing Machinery.
- w3c (2019). World Wide Web Consortium brings a new language to the Web as WebAssembly becomes a W3C Recommendation. <https://www.w3.org/2019/12/pressrelease-wasm-rec.html.en>
- Clark, L. (2019). WebAssembly Interface Types: Interoperate with All the Things. Mozilla Hacks: the Web developer blog. <https://hacks.mozilla.org/2019/08/webassembly-interface-types/>
- Kuhail, M. A., Farooq, S., Hammad, R., and Bahja, M. (2021). Characterizing Visual Programming Approaches for End-User Developers: A Systematic Review. IEEE Access, 9:14181–14202.
- Sousa, T. (2012). Dataflow Programming: Concept, Languages and Applications. Unpublished.

Thank you for your attention!



Questions



Discussion:



Why can't an existing, generic VPL's be used? In other words, what is a "GIS VPL" in relationship to 'any' computer graphics VPL?

- In one word: scalability.
- The scale of geodata is an important characteristic of GIS.
- To a large extend, a GIS VPL will require the same features as a VPL for other fields which can be considered 'applied CG'.
 - Construct and view geometry, large collection of transformation components, UI components, etc.
- However, A stark difference can be found in the need for scalability.
 - While development of GIS tools and geo-pipelines often occur on a small scale, the eventual execution of these tools often needs to happen on sizable datasets.
 - This means that any VPL which claims to be 'aimed at GIS', will need to at least provide an answer on how it can be scaled to handle massive datasets.

Why does the field of GIS need a(nother) VPL?

- In GIS, VPLs are almost exclusively used as server-side or native ETL tools.
- However, VPLs can serve many other purposes, which are currently not being explored.
- One possible application of a VPL in GIS which has not been explored, is to use a VPL for web-based, serverless geocomputation.
- Such an application would contribute to making geocomputation libraries more directly usable, and web applications more composable.

Is this science?

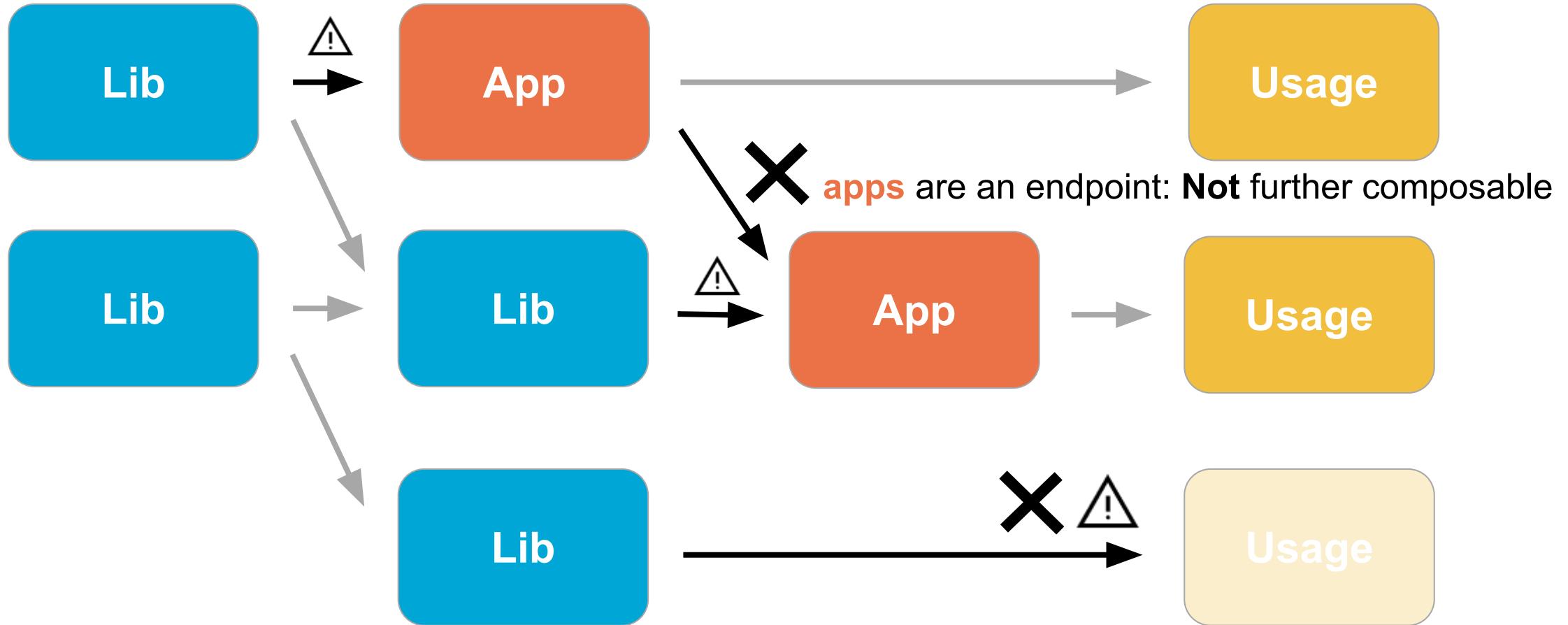
I think It is valuable to be alert of all ‘trivia’ within scientific research:

- If trivia becomes overwhelming component of a study, it should be solved
- whether or not that counts as science, I don’t know.
- I do know it is in the interest of science to solve those problems.
- Distribution & Accessibility are the trivial problems I tried to solve, once and for all, for all subsequent research. That was the goal I set out to accomplish at least.
- BME example (lies)

Leftovers:



⚠ Some **lib** capabilities get lost when used in an **app**



⚠ A **lib** offers no visualization or GUI.

✗ A **lib** must be turned into an **app** before utilization.