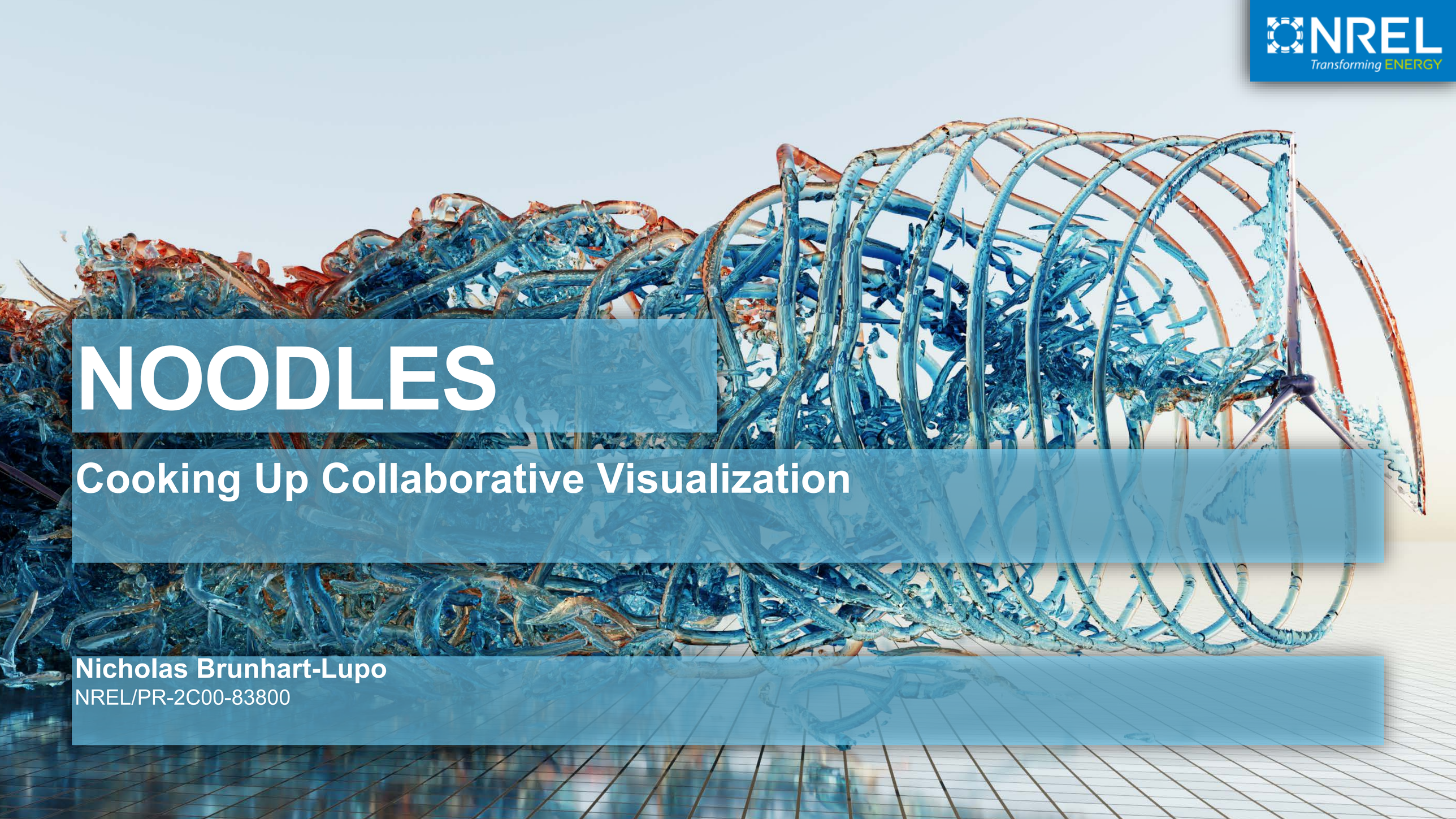


NOODLES

Cooking Up Collaborative Visualization

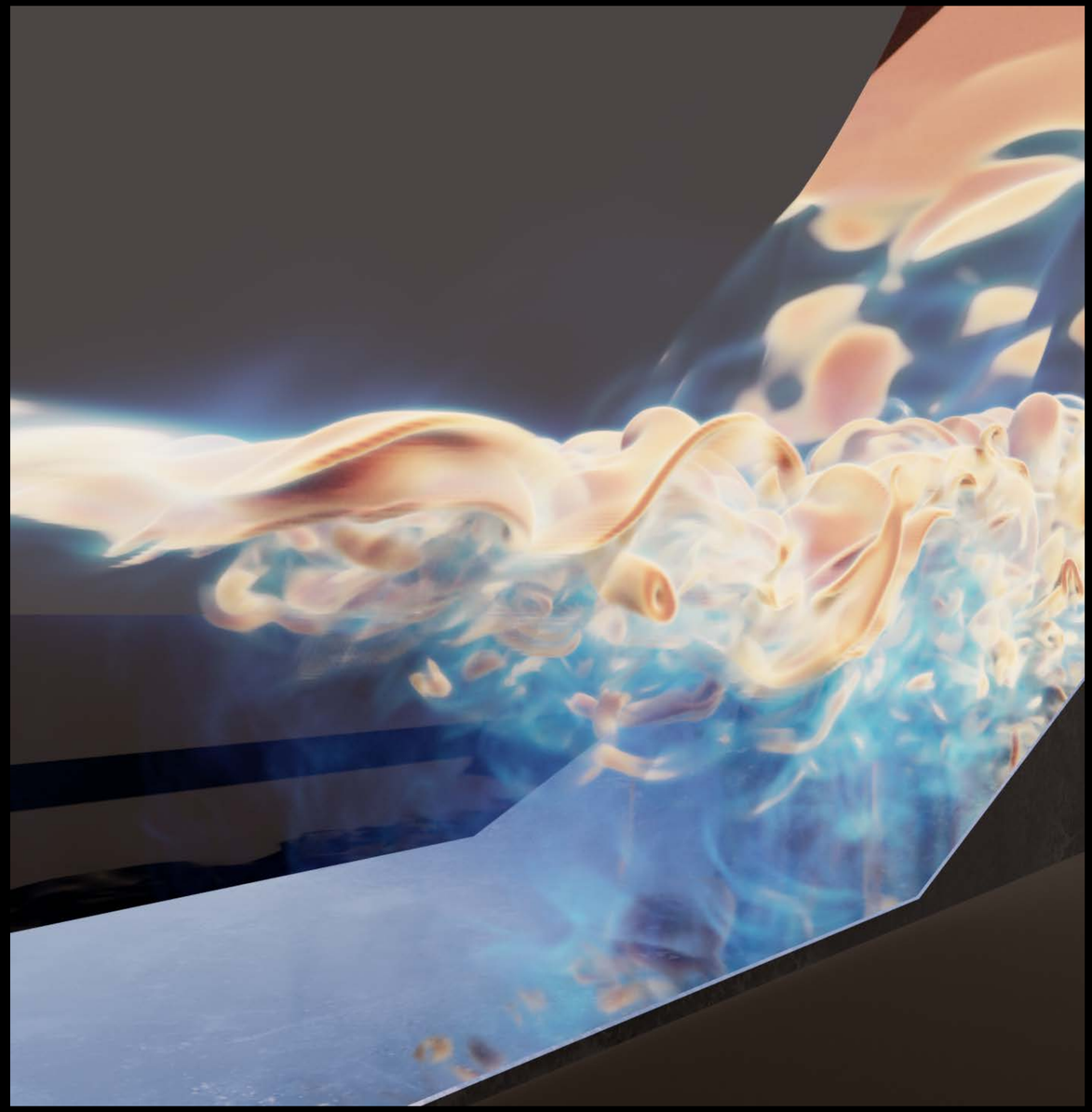
Nicholas Brunhart-Lupo

NREL/PR-2C00-83800

Introduction

Challenges

- Datasets getting larger and more complex
- Teams becoming more diverse
- New hardware available, rarely used
- Result:
  - Waste of resources
  - Questions of analysis defendability



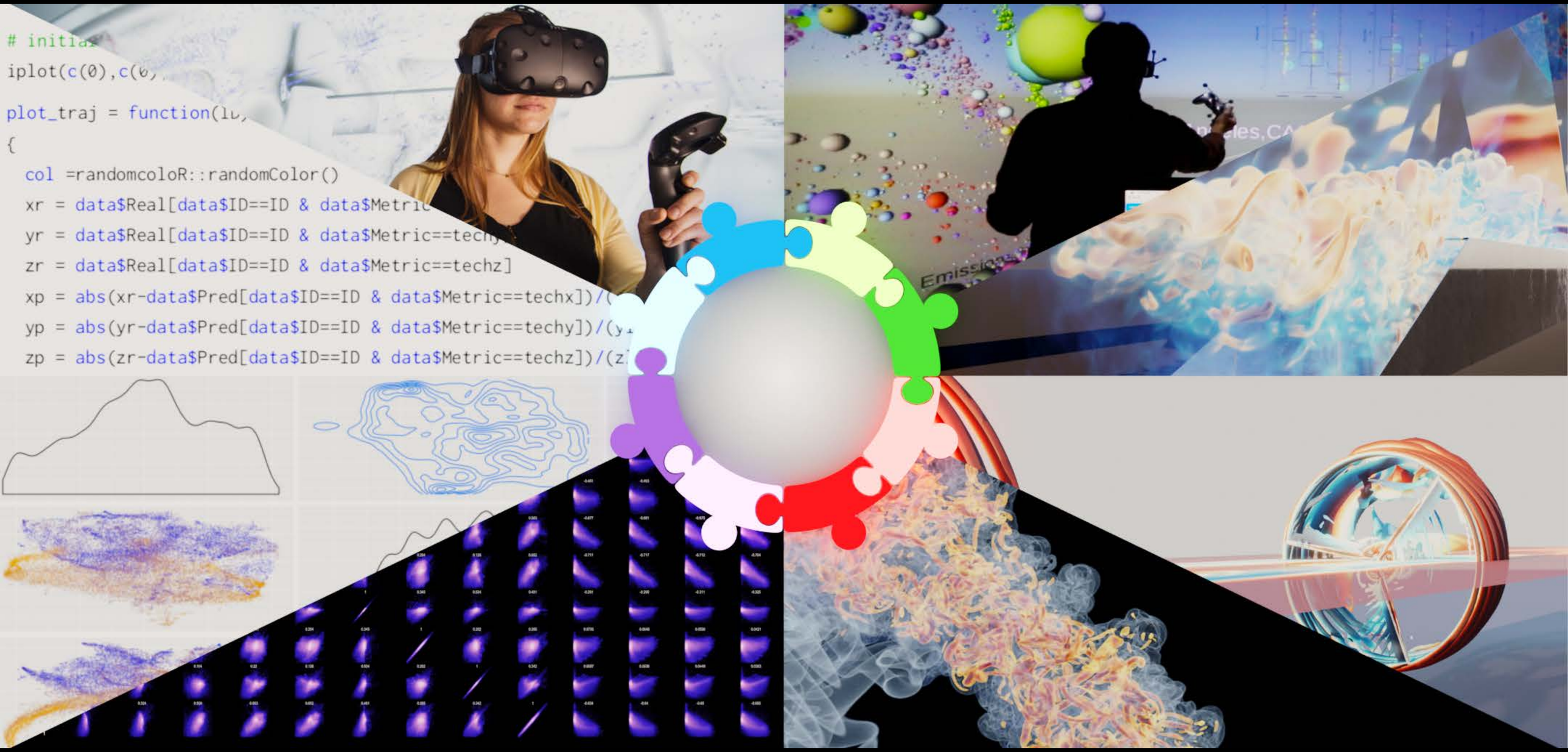
Collaborative Visualization & Analysis

- Collaborative techniques can help
- Bring more eyes to complex problems
- Bring diverse perspectives (both participants and hardware)
- Reduce barriers in team effort

However…

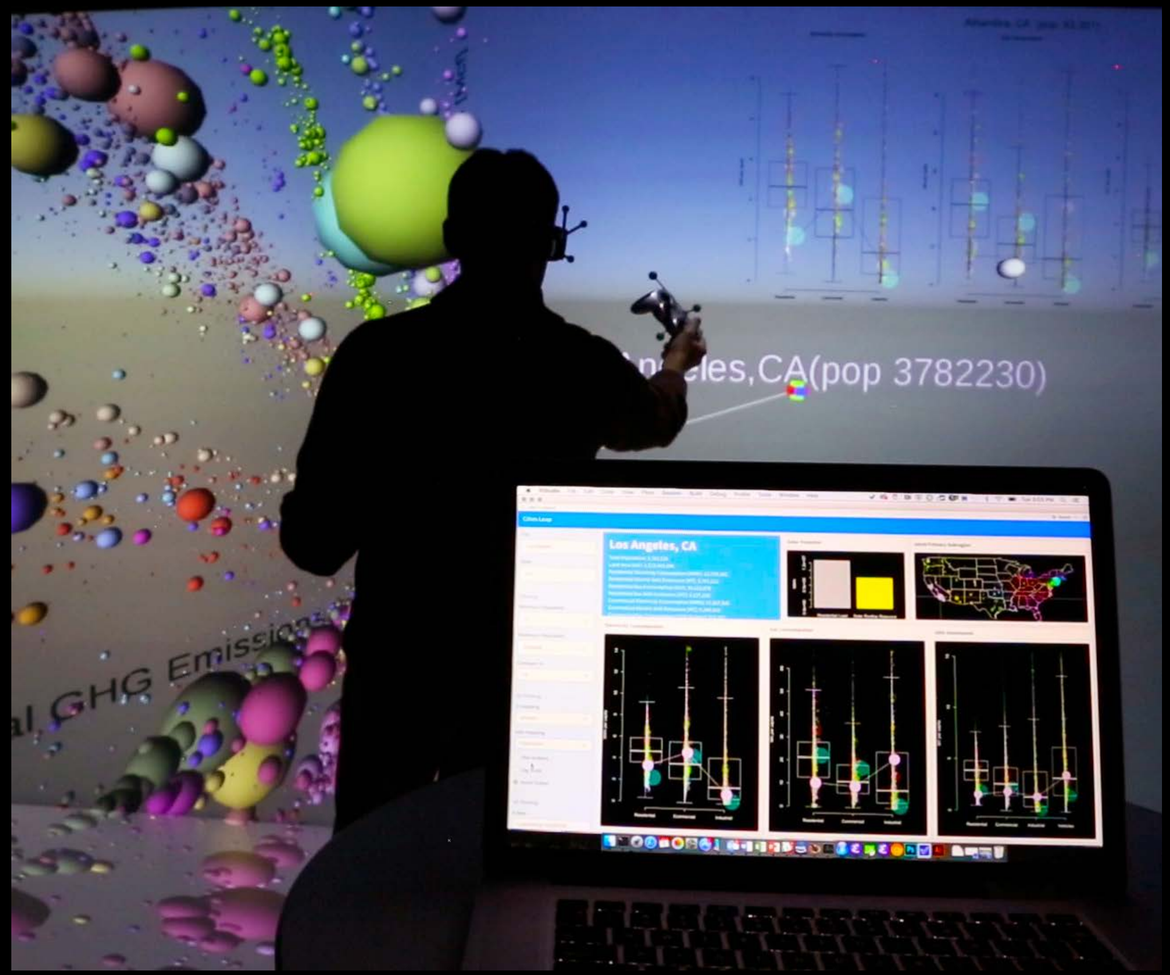
Current collaborative approaches are:

- Stovepiped
- Destructive to workflows
  - Resistance to change
  - Built over years
- Heavy tech requirements
- High friction


iplot(c(0),c(0),
plot_traj = function(lb,
{
col =randomcoloR::randomColor()
xr = data$Real[data$ID==ID & data$Metric
yr = data$Real[data$ID==ID & data$Metric==techy
zr = data$Real[data$ID==ID & data$Metric==techz]
xp = abs(xr-data$Pred[data$ID==ID & data$Metric==techx])/(
yp = abs(yr-data$Pred[data$ID==ID & data$Metric==techy])/(y
zp = abs(zr-data$Pred[data$ID==ID & data$Metric==techz])/(z

Uh, how?

- PlottyVR
- Simple Client <-> Server scatterplot tool
- Iterative project



Principles

- Minimally Invasive
  - Workflows are paramount, disturb them as little as possible
  - Allows composition
- Minimal Dependencies
  - Need to use some technologies, but keep them to a minimum

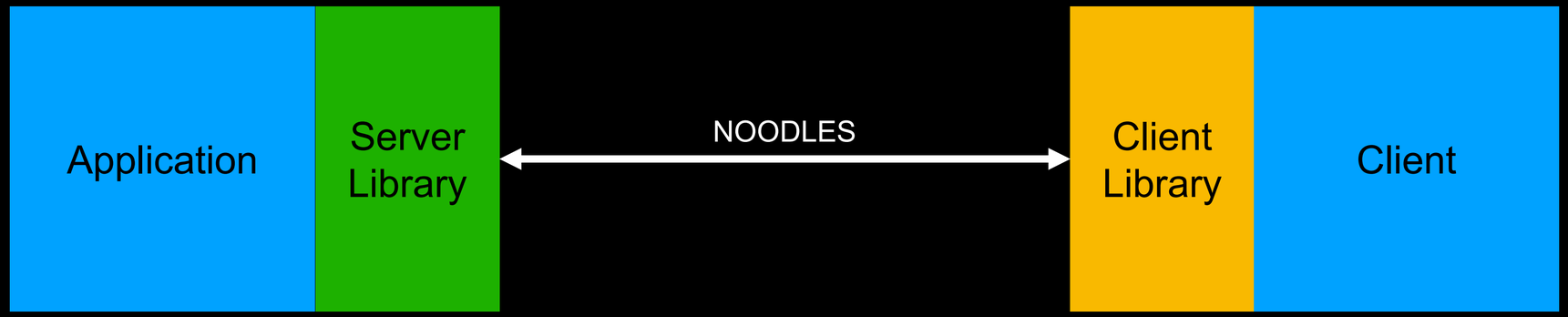
Approach - Protocol

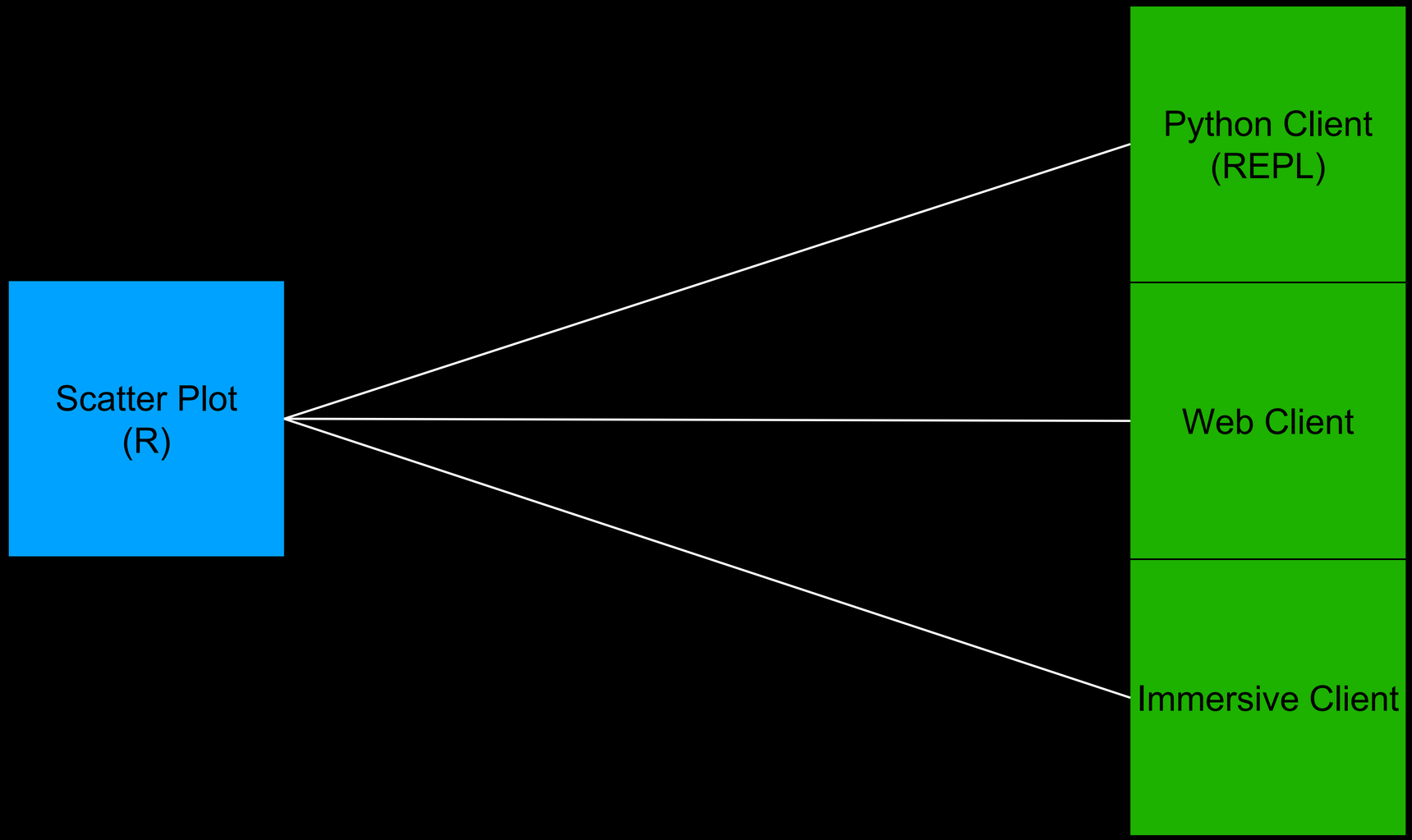
- Any tool that speaks this language can participate
- Supports any transport
- Software stacks die, protocols and formats endure
- Support other implementations!

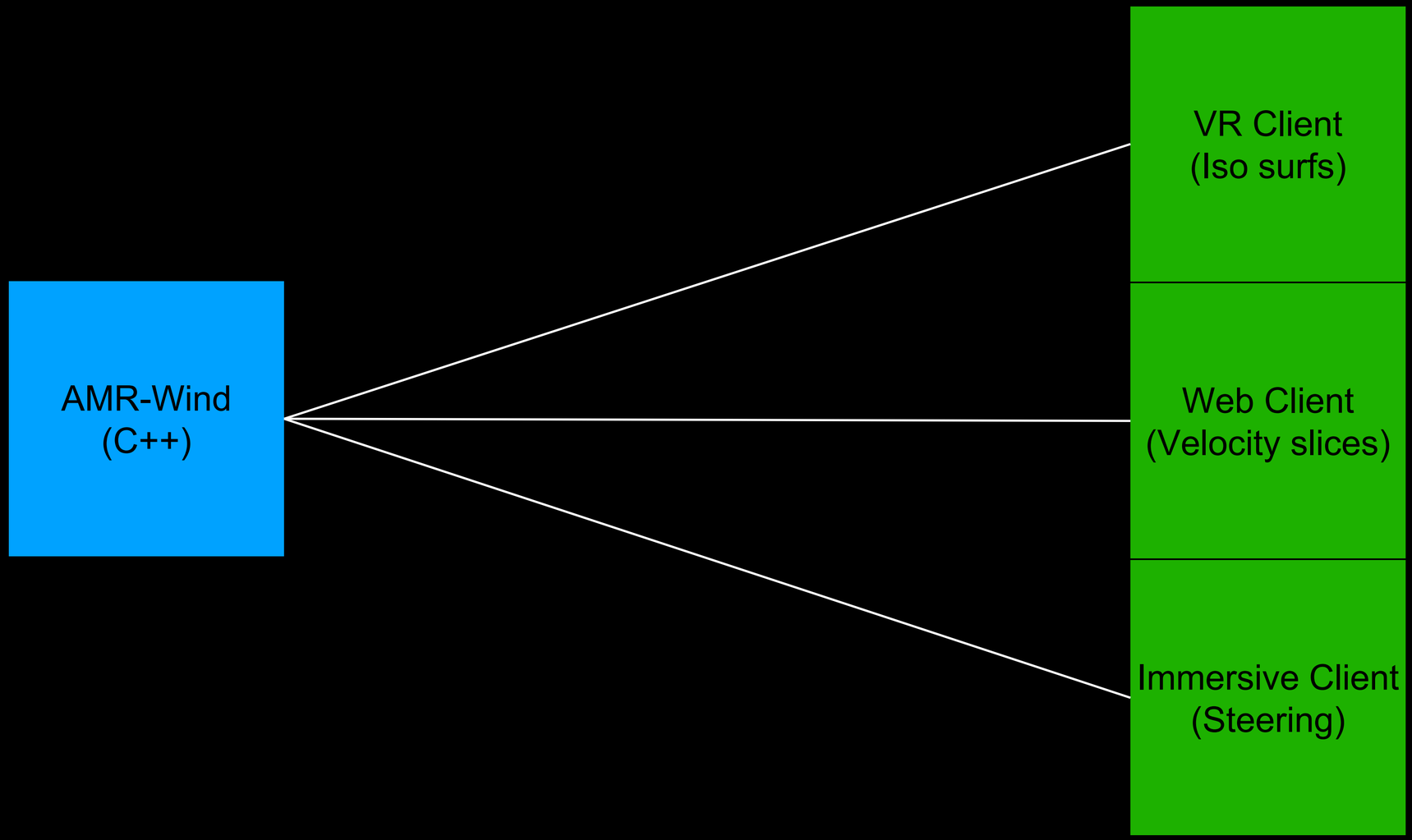
Next Level

- Support multiple clients
- Operate on a lower level for generalization
  - Entity component model
- CBOR-based messages

NOODLES


Application
Server Library
NOODLES
Client Library
Client

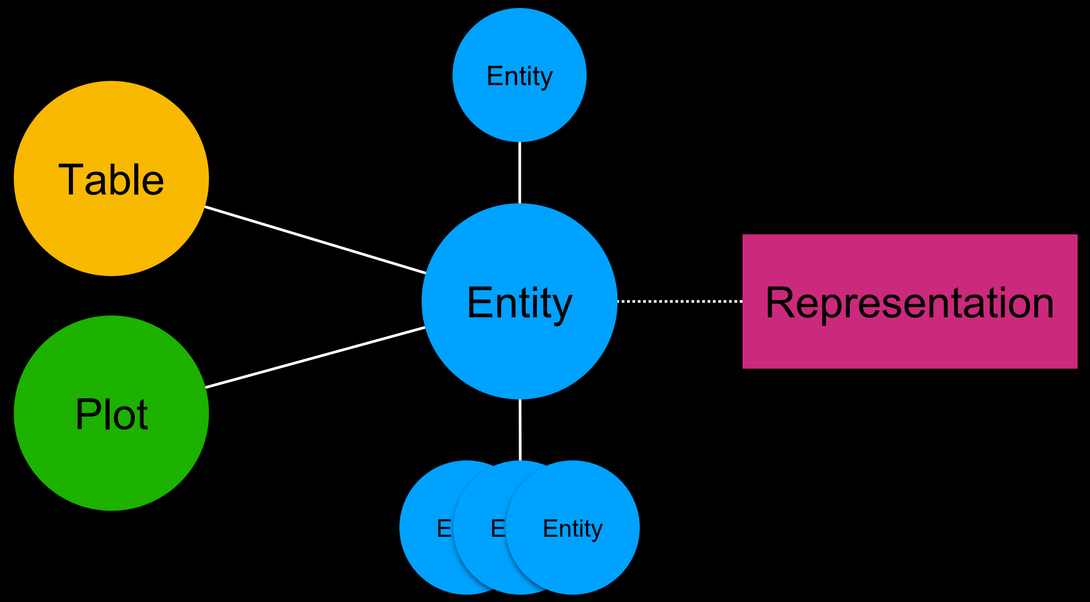

Scatter Plot
(R)
Python Client
(REPL)
Web Client
Immersive Client


AMR-Wind
(C++)
VR Client
(Iso surfs)
Web Client
(Velocity slices)
Immersive Client
(Steering)

Concepts

Concepts

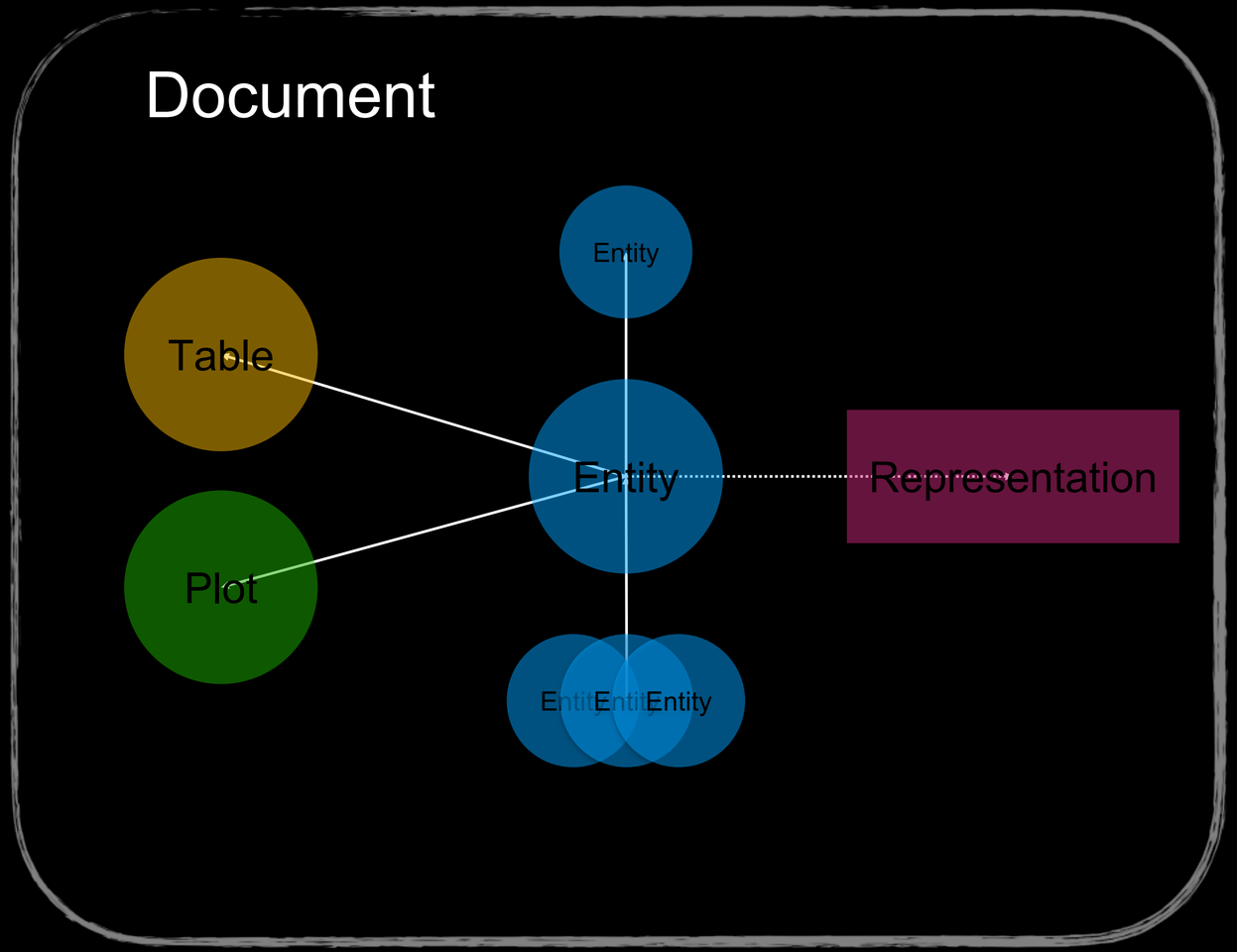
Document



Concepts

Document

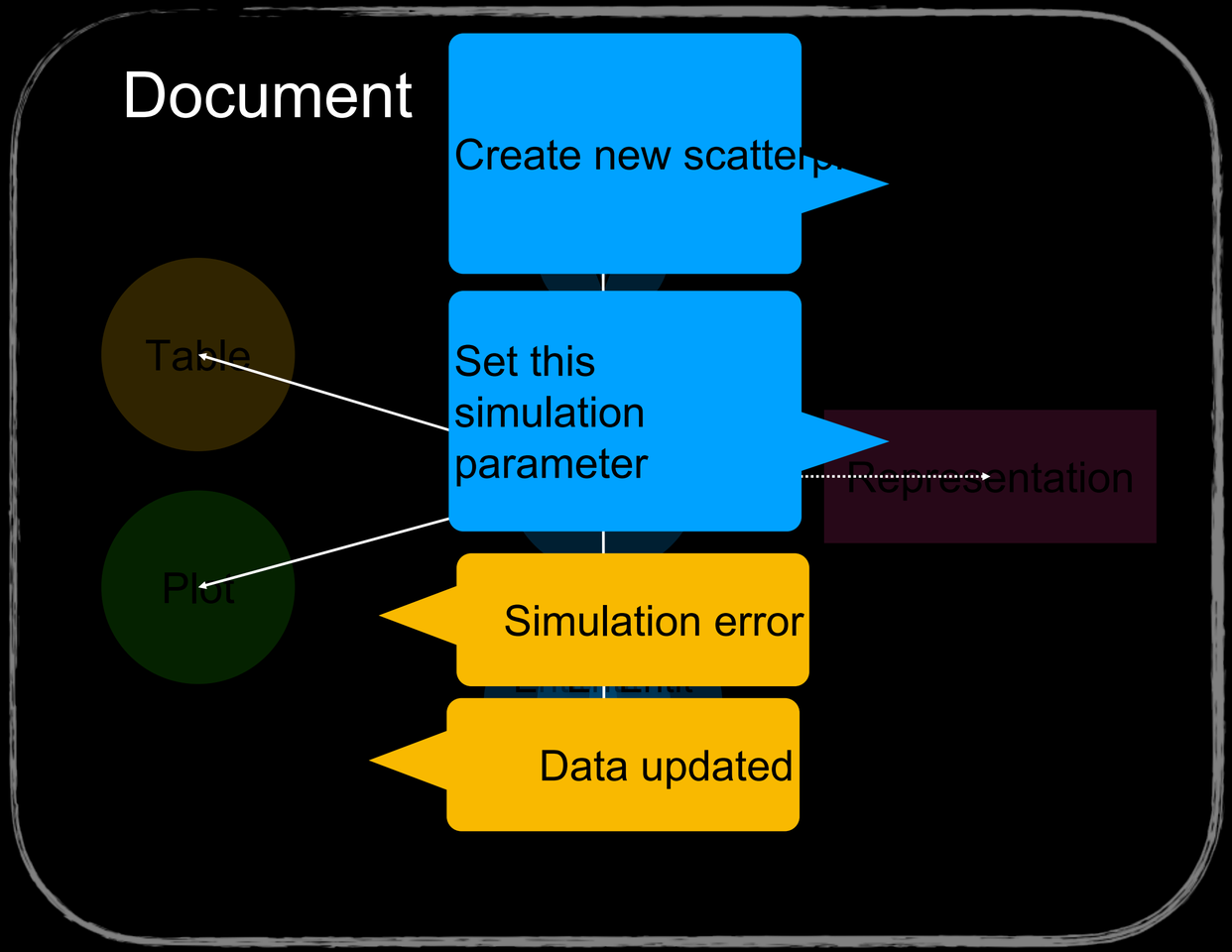
- The visualization scene
- Contains all ‘things’
- Can be reset
- Has methods and signals



Concepts

Methods and Signals

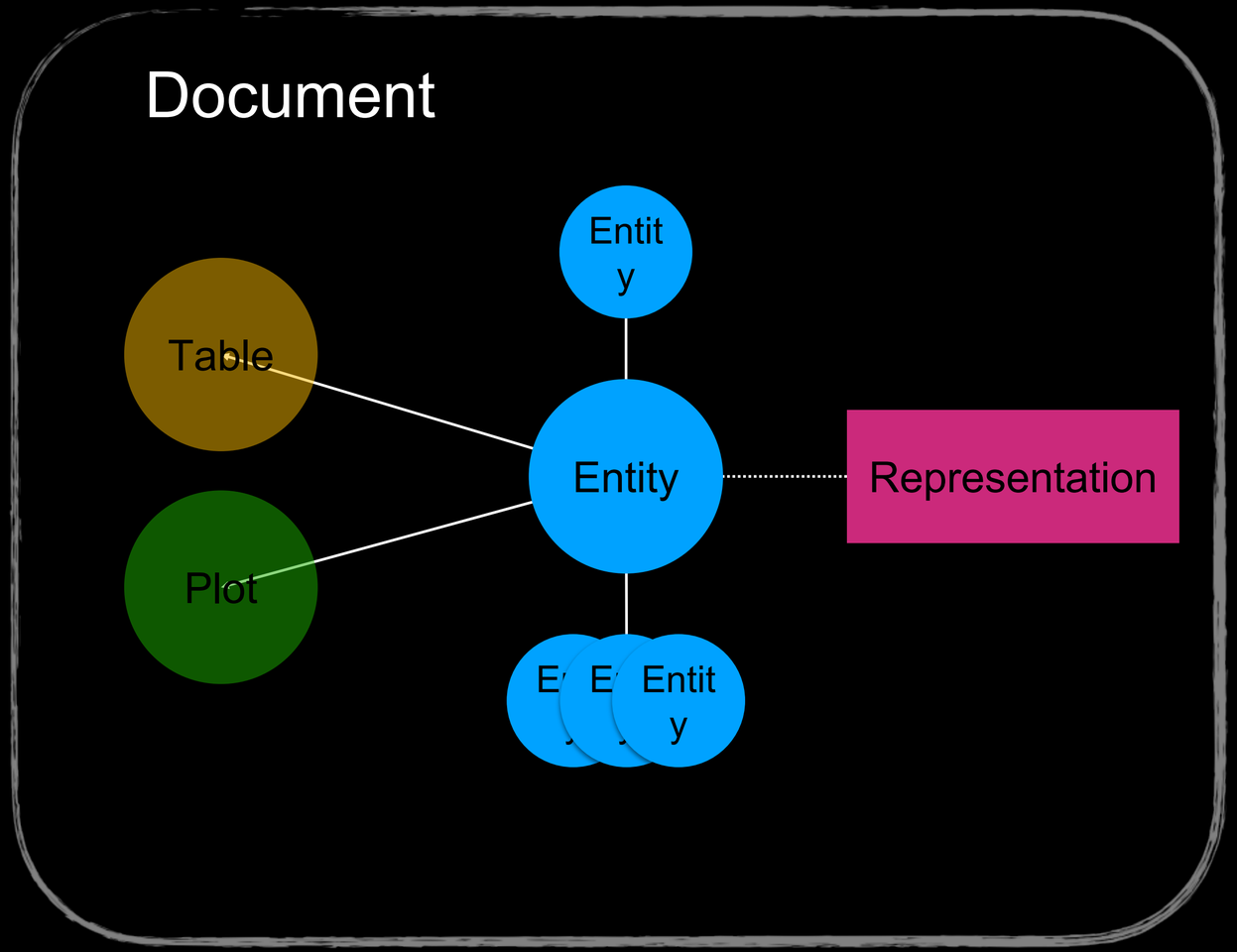
- Methods are remote procedures
  - Clients ask to invoke
  - Server executes some function
  - Some defined by the spec
  - Most application specific
- Signals are notifications



Concepts

Entity

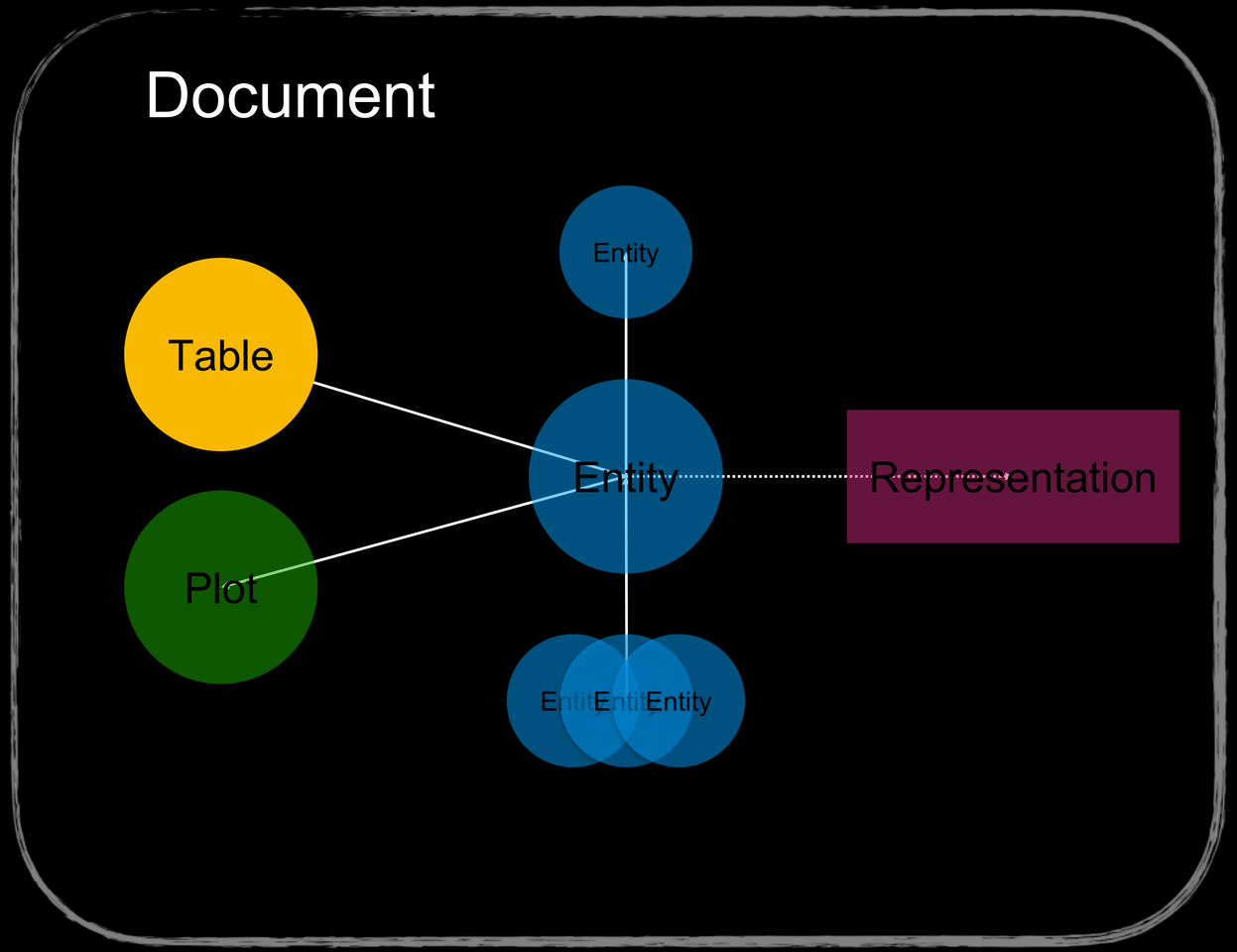
- Organized in a tree
- Has 3D position, rotation, scale
- May have a representation
  - Geometry, text, or webpage
  - Geometry may be instanced
- Have methods and signals



Concepts

Table

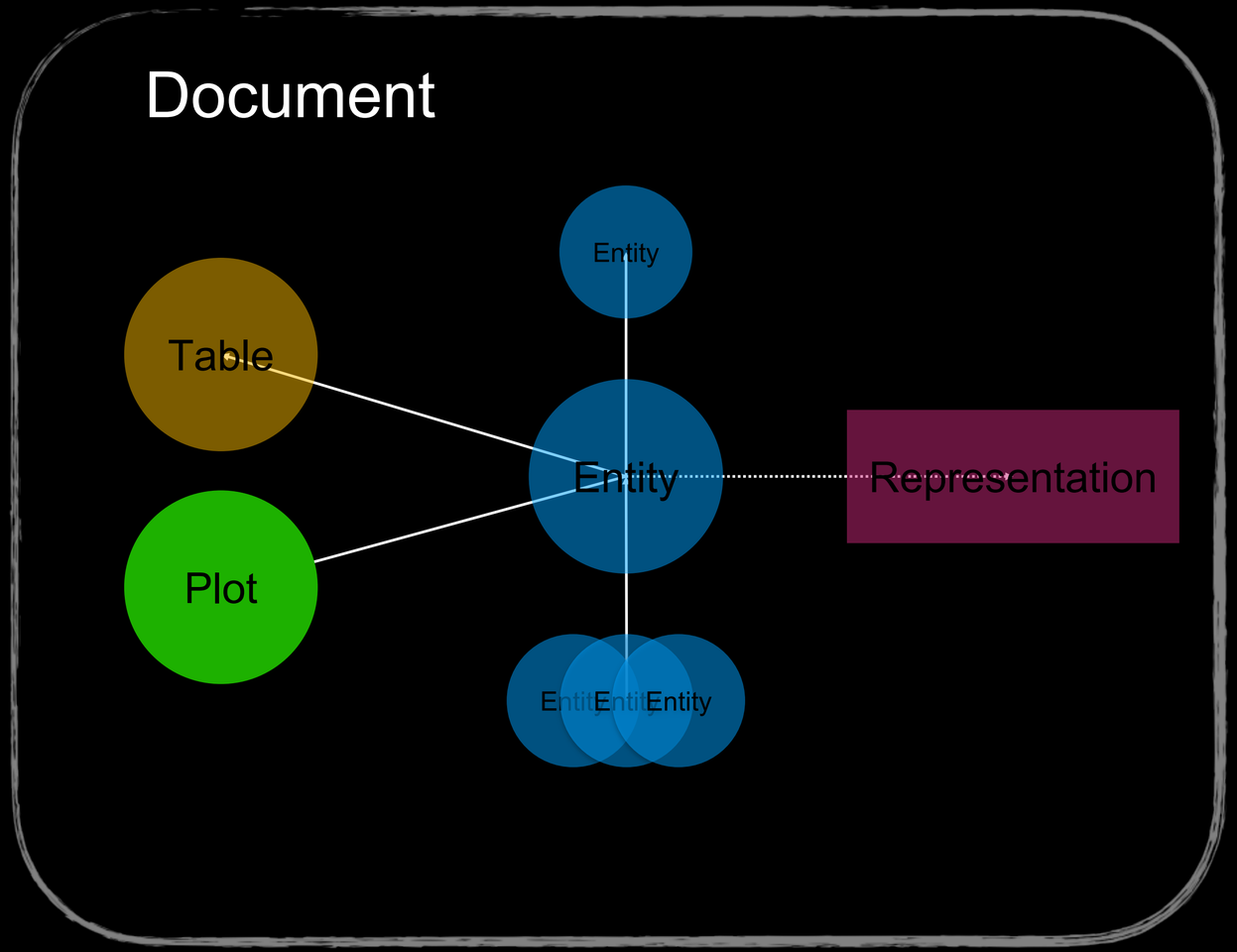
- Provide structured access to records data
- Clients may be able to modify
  - Subscribers updated
- May be linked to entities
- May link to database for smarter clients
- Have methods and signals



Concepts

Plot

- Provide additional or alternative view
- Can be linked with entity
- Can be simple, complex, or webpage
  - Webpage allows nesting!
  - Up for revision
- Have methods and signals



Under the Hood

Communication

- Sequence of well-defined messages
- Approx. three kinds:
  - Create, Update, Delete
- Encoded in CBOR
- Rest of semantics defined over signals and methods
- Specified in .CDDL format for verification

```
MethodArg = {
   name: tstr,
   ? doc: tstr,
   ? editor_hint: tstr
}

MsgMethodCreate = {
   id: MethodID,
   name: tstr,
   ? doc: tstr,
   ? return_doc: tstr,
   arg_doc: [ * MethodArg ]
}

MsgMethodDelete = {
   id: MethodID
}
```

Under the Hood

CBOR

- Looks and tastes like JSON
- Is a superset of JSON
- Schema free
- Designed for IoT
- More concise than BSON, etc
- Lots of support, trivial codec

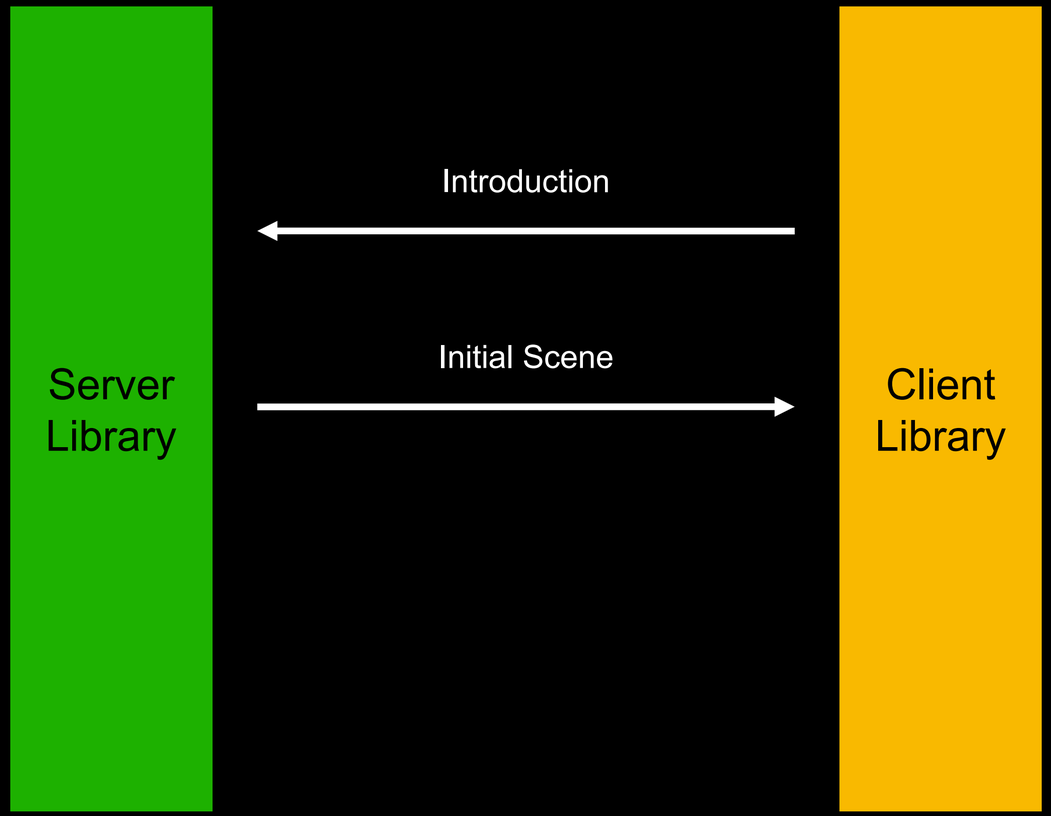
```
MsgMethodCreate = {
   id: MethodID,
   name: tstr,
   ? doc: tstr,
   ? return_doc: tstr,
   arg_doc: [ * MethodArg ]
}

MsgMethodDelete = {
   id: MethodID
}
```

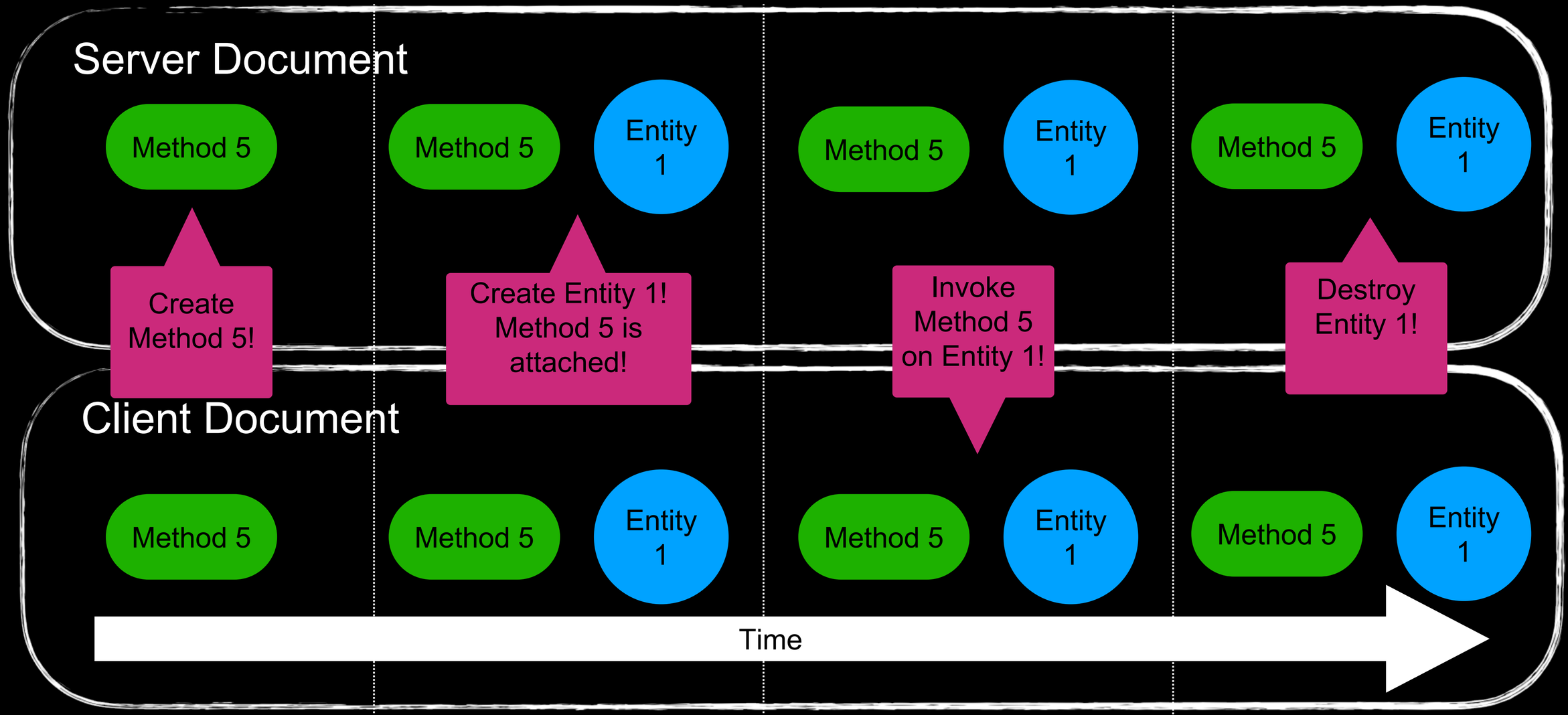
Communication

Semantics

Connection



Semantics


Server Document
Method 5
Method 5
Entity 1
Method 5
Entity 1
Method 5
Entity 1
Create Method 5!
Create Entity 1! Method 5 is attached!
Invoke Method 5 on Entity 1!
Destroy Entity 1!
Client Document
Method 5
Method 5
Entity 1
Method 5
Entity 1
Method 5
Entity 1
Time

Demo: Scatter Plot

Demo: Isosurfaces


Noodles Viewer
/Users/nbrunhar/Downloads/whole_sc
DirectionalLight_1055531218044
ROOT
DirectionalLight_1055531218048
DirectionalLight_1055531218051
DirectionalLight_1055531218054
DirectionalLight_1055531218057
Object 8|0
Object 10|0
Object 12|0
Object 13|0
Object 14|0
Object 15|0
Object 17|0
Object 18|0
Tables
Methods

Table Semantics

Table Methods

TblInit noo::tbl_subscribe()
void noo::tbl_insert([TableRow])
void noo::tbl_update([Key], [TableRow])
void noo::tbl_remove([Key])
void noo::tbl_clear()
void noo::tbl_update_selection(Selection)

Table Signals

void noo::tbl_reset(TblInit)
void noo::tbl_updated([Key], [TableRow])
void noo::tbl_rows_removed([Key])
void noo::tbl_selection_updated(Selection)

Object Semantics

- Activation
- Per-Object Variables
- Constrained Options
- Movability
- Selection
- Probing
- Attention
- Client View

Current Status

- Spec is reasonably mature
- Libraries
  - C++: 100%
  - Javascript: 90%
  - Python: 50%
- Plugins
  - Blender: 10%
  - Paraview/VTK: 10%
- Applications:
  - Scatterplot
  - Playground
  - NOODLES + three.js

Future Work

- Animation
- Lighting
- Plots
- Volume rendering
- Compression
- Combine scenes from multiple servers (MPI compositions, or overlays)
- More clients!
- Data and service discovery
- Recording
- Remote rendering
  - Noodles as uniform interface
  - Federation

Message Specification and Libraries