Cinema Spreadsheet Example

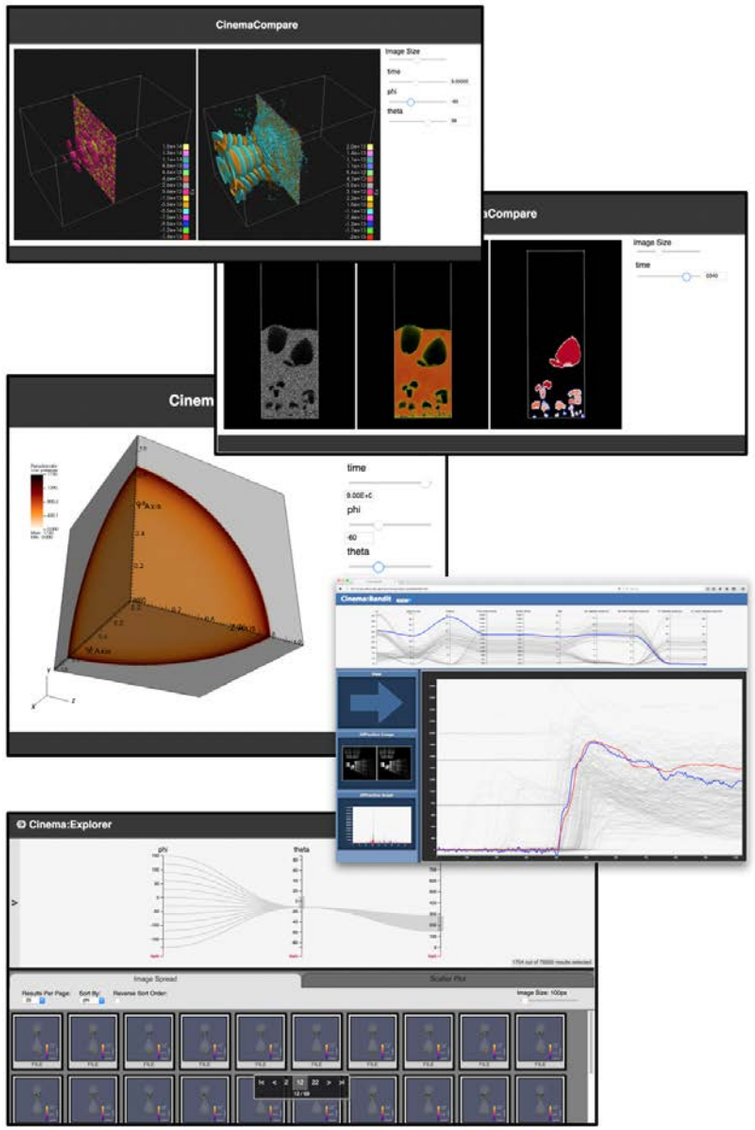
Online Tutorial, 10 March 2020

David Rogers, James Ahrens, Terry Turton, Soumya Dutta,
Divya Banesh, Roxana Bujack, Ethan Stam

LA-UR-19-31442

Example Goals

- **Introduce Cinema**
- **Create a Cinema Spreadsheet Example**
  - Create and inspect a simple Cinema database
- **Further Information**
  - Pointers to more tutorials and instructions

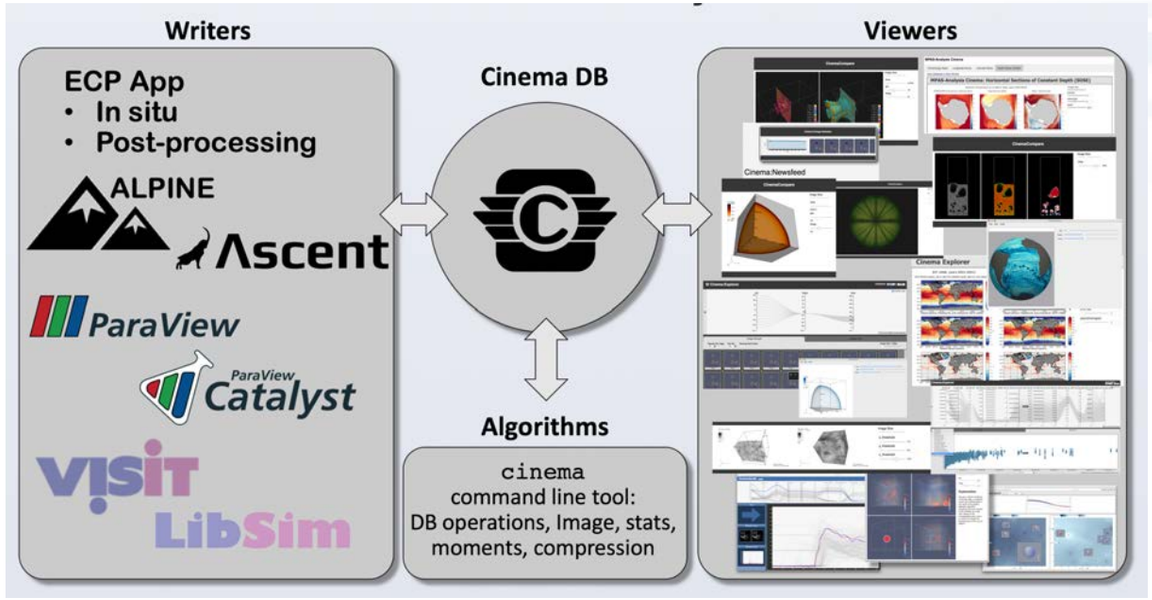


What is Cinema?

Cinema is and ecosystem consisting of:

- database specification,
- database writers
- database viewers
- algorithms

Designed to help explore extreme scale data.



Important Terms

- **Cinema database:** a set of data in permanent storage, as described by the Cinema Specification, consisting of **artifacts** and parameters.
- **Artifact (Extract):** a logical collection of data in permanent storage
  - example: a set of images
  - example: a set of graphs
  - example: a simulation grid
  - example: `.vti` file

A note about Cinema Viewers and browser permissions

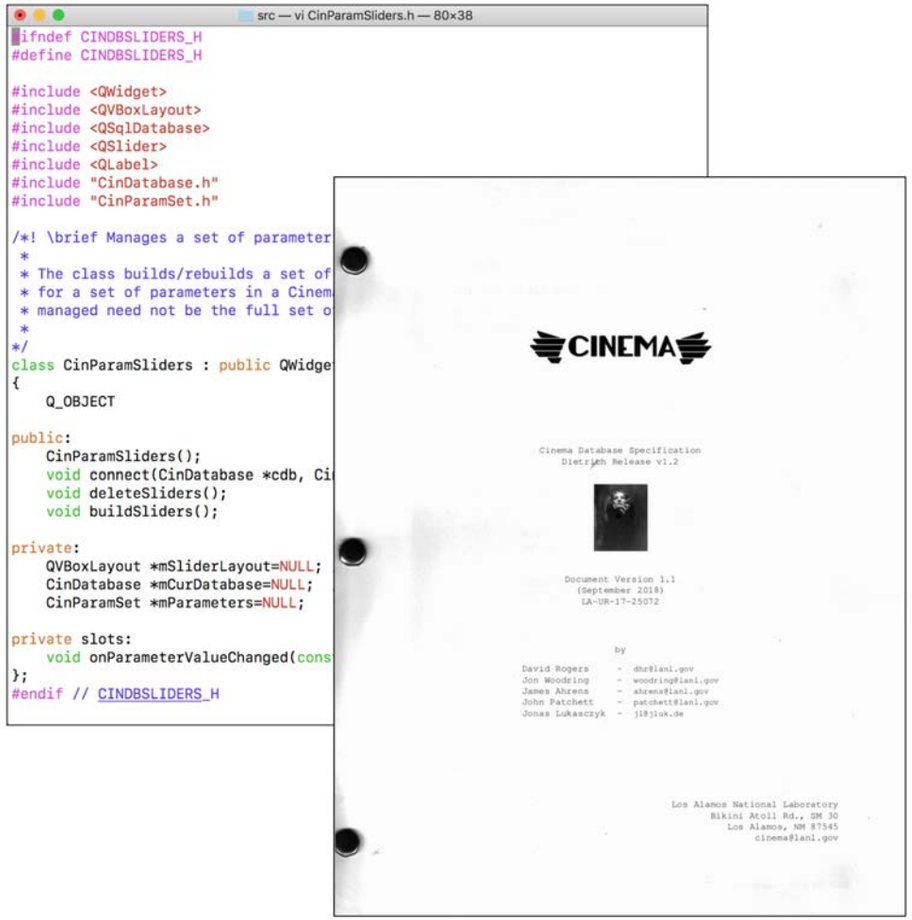
Browsers may not allow access to local files, which is needed by some Cinema viewers. To change this, set the following options for your browser (Firefox and Chrome are Cinema's offically supported browsers, though others may work as well):

- Firefox
  - In address bar, input **about:config:**
    - Change **privacy.file_unique_origin** to **false**
- Chrome (exit Chrome)
  - use --disable-web-security command line option for this session
  - `open_tutorial` file has an example for Mac

Digging into Cinema Databases

A Cinema database is a set of artifacts, parameters and semantics.

- Cinema Specification
  - Describes the format (csv), and a minimal set of restrictions on the data.
  - A set of artifacts, and parameters associated with those files.
  - Note that location of artifacts is not restricted by specification (can be local or remote), but for simplicity, this tutorial covers local examples.



A hand-editable example of a Cinema Database

No matter how complex, all Cinema databases have the same elements.

```
example_01.cdb/
    data.csv (parameters)

example_02.cdb/
    data.csv (parameters)
    01.png   (artifact files)
    02.png
    03.png
    ...
```

Other data can reside in the `<>.cdb` directory, and is not controlled by the spec.

How would you create one from scratch?

- Use Case: you have a set of images defined by some parameters (say, time ...)

```
data/
    x-42.png
    x-43.png
    x-44.png
    x-45.png
    x-46.png
    x-48.png
...
```

Cinema Database

Create a `data.csv` file, according to the specification:

Format is `(parameters)(artifacts)`.

```
time,FILE
42,x-42.png
43,x-43.png
44,x-44.png
45,x-45.png
46,x-46.png
48,x-48.png
49,x-49.png
50,x-50.png
52,x-52.png
53,x-53.png
...
```

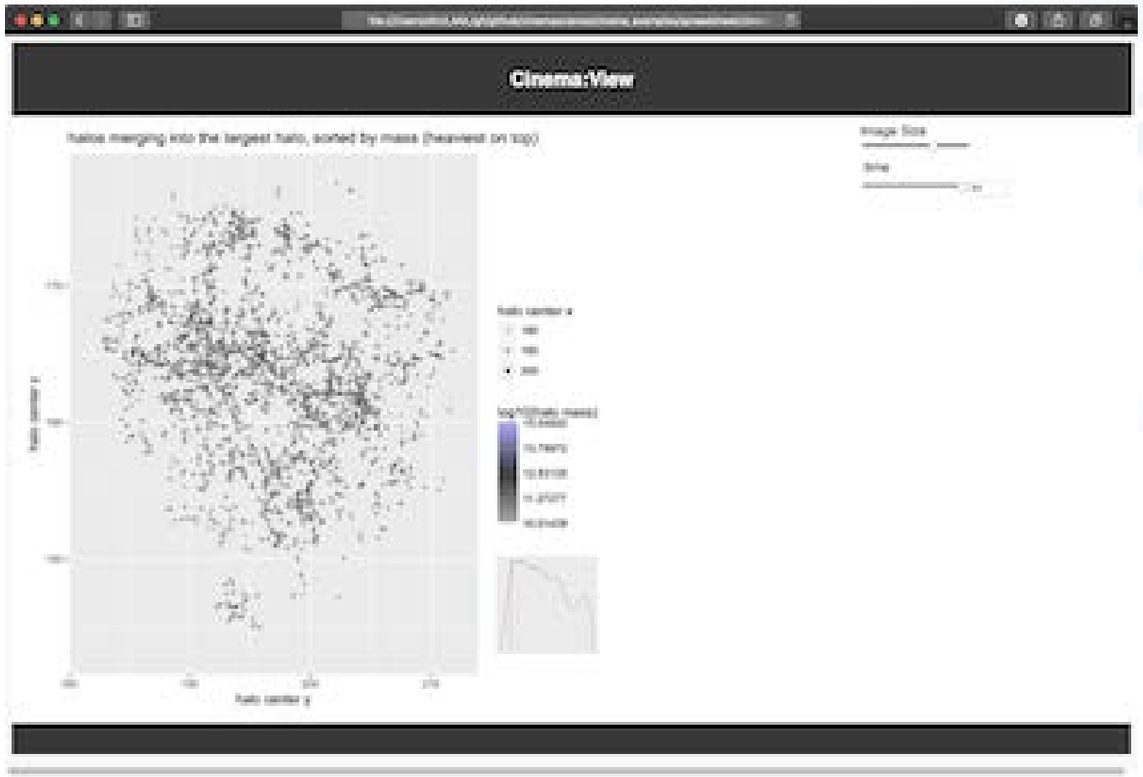
That's it ...

```
data/
    data.csv
    x-42.png
    x-43.png
    x-44.png
    x-45.png
    x-46.png
    x-48.png
...
```

Now you can explore this in a viewer

Explore this with a Cinema viewer, and scroll through the images using the parameter sliders.

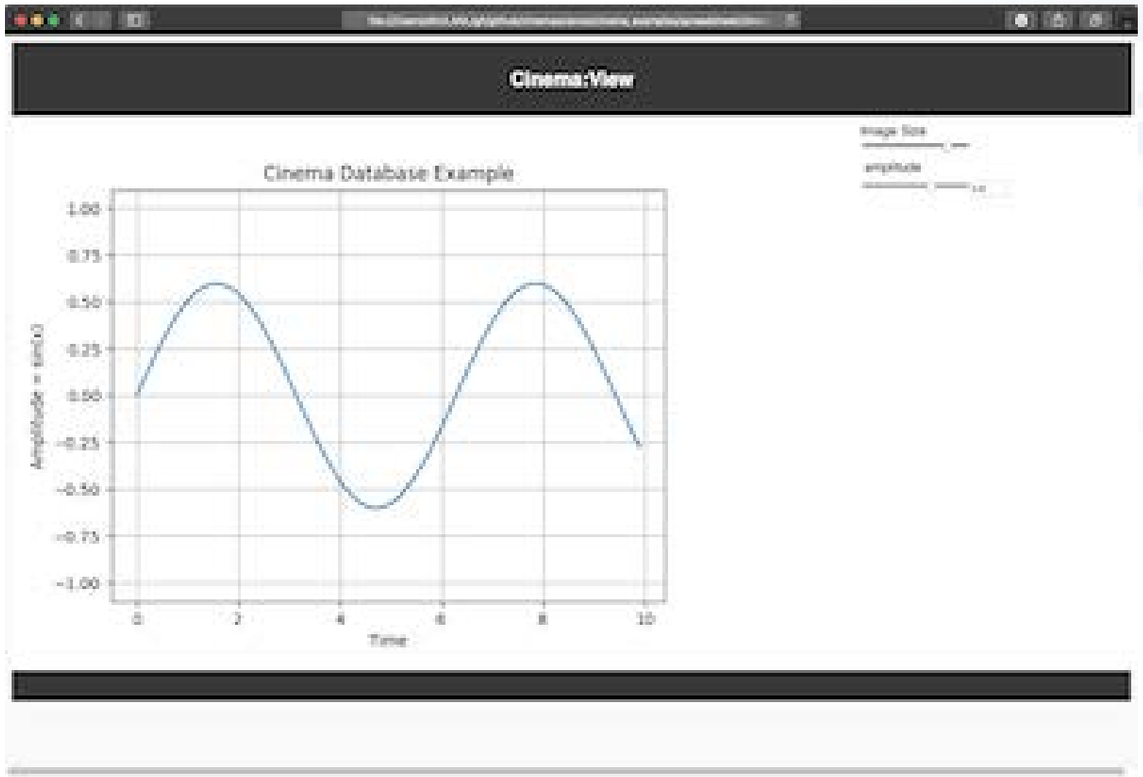
- Halo Dataset link in tutorial page
- Scroll through time
- Change the size of the image



Explore another small dataset, with a single parameter

Explore this with a Cinema viewer, and scroll through the images using the parameter sliders.

- Small Experiment link in tutorial page
- Scroll through time
- Change the size of the image

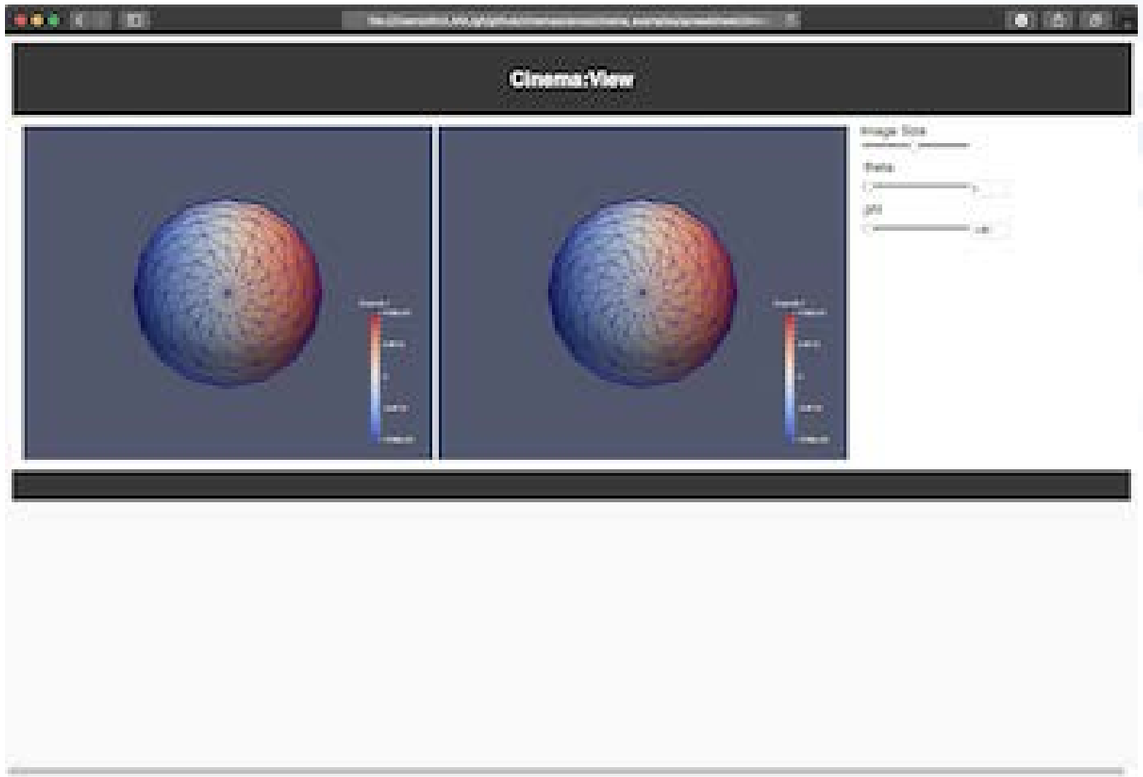


Try a database with a few more parameters

- Volume Dataset link in tutorial page
- Scroll through time, camera angle, and slice plane position
- Change the size of the image

Finally, compare two datasets side-by-side

- Comparison Example link in tutorial page
- Scroll through camera angle
- Change the size of the image



But ... Data Is 'Messy'

- Smoothly varying databases are nice, and typically come from simulations
  - Typically have all values expected - entire `data.csv` is populated
- But real world data can be messy, and this is valid per the spec
  - `NaN`, empty values are fine
  - viewers, algs are expected to properly deal with this

Example of a messy `data.csv` file:

```
temp,pressure,vel,time,FILE
3.0,1.0,,1.0,results/1.csv
3.1,2.0,45.6,1.0,results/2.csv
3.3,3.0,45.6,1.0,results/3.csv
4.0,NaN,45.6,1.0,results/4.csv
...
```

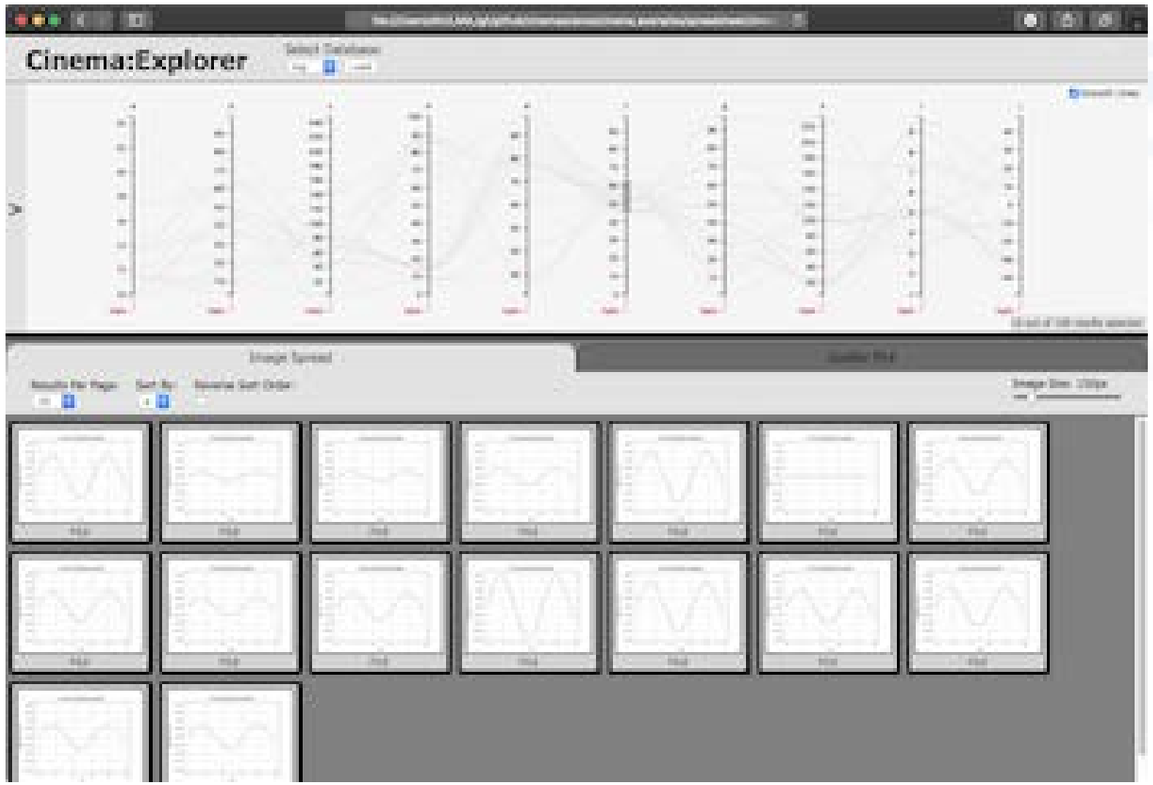
A multi-artifact Cinema database

- A database can have multiple artifacts
  - Example: `materials/data/multi_artifact.cdb`
  - The database has the same organization as before:
  - `(set of parameters)(set of artifacts)`

```
theta,phi,vti x-radius,pdb,FILE,FILE_VTI,FILE_PDB
0,-180,10,good,image/-180/0.png,wavelet/10.vti,good.pdb
0,-162,20,N/A,image/-162/0.png,wavelet/20.vti,
0,-144,30,N/A,image/-144/0.png,wavelet/30.vti,
0,-126,40,N/A,image/-126/0.png,,
...
```

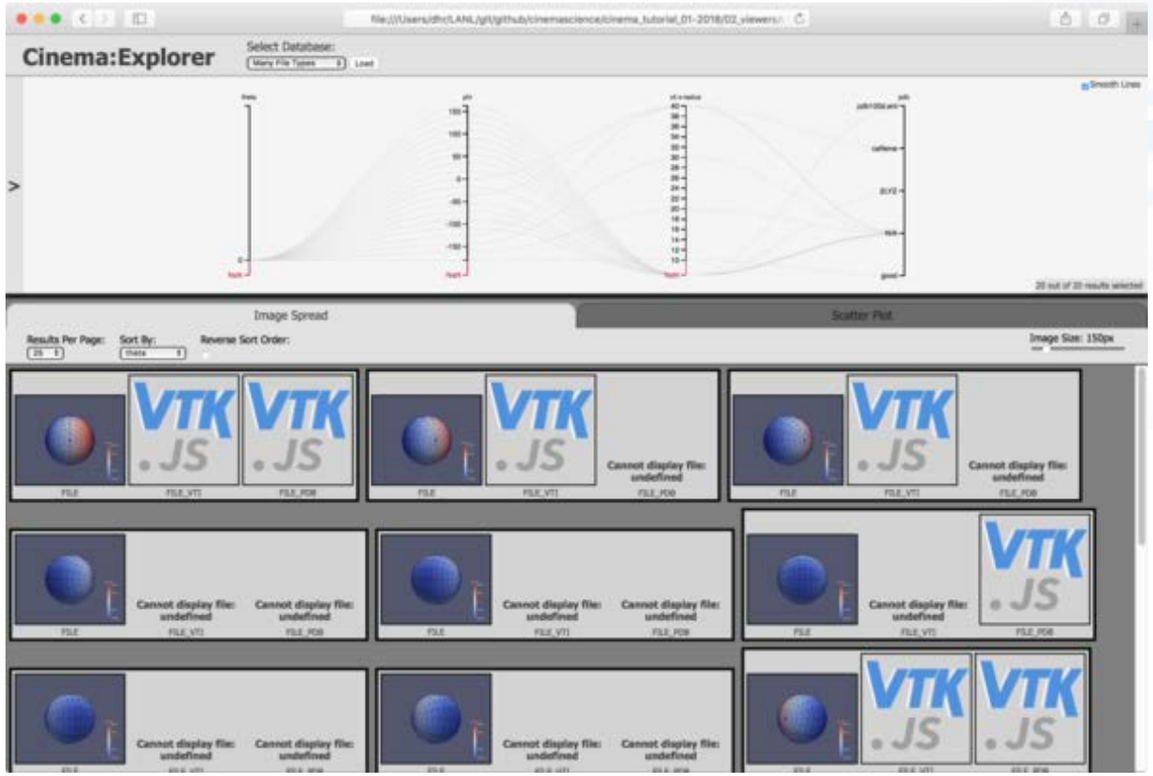
An example of Cinema:Explorer viewer on more complex data

- *Big Experiment* link in **tutorial home page**
- select "Multiple Artifacts" from the database list at the top, press `load` button



Cinema Explorer viewer supports multiple data types

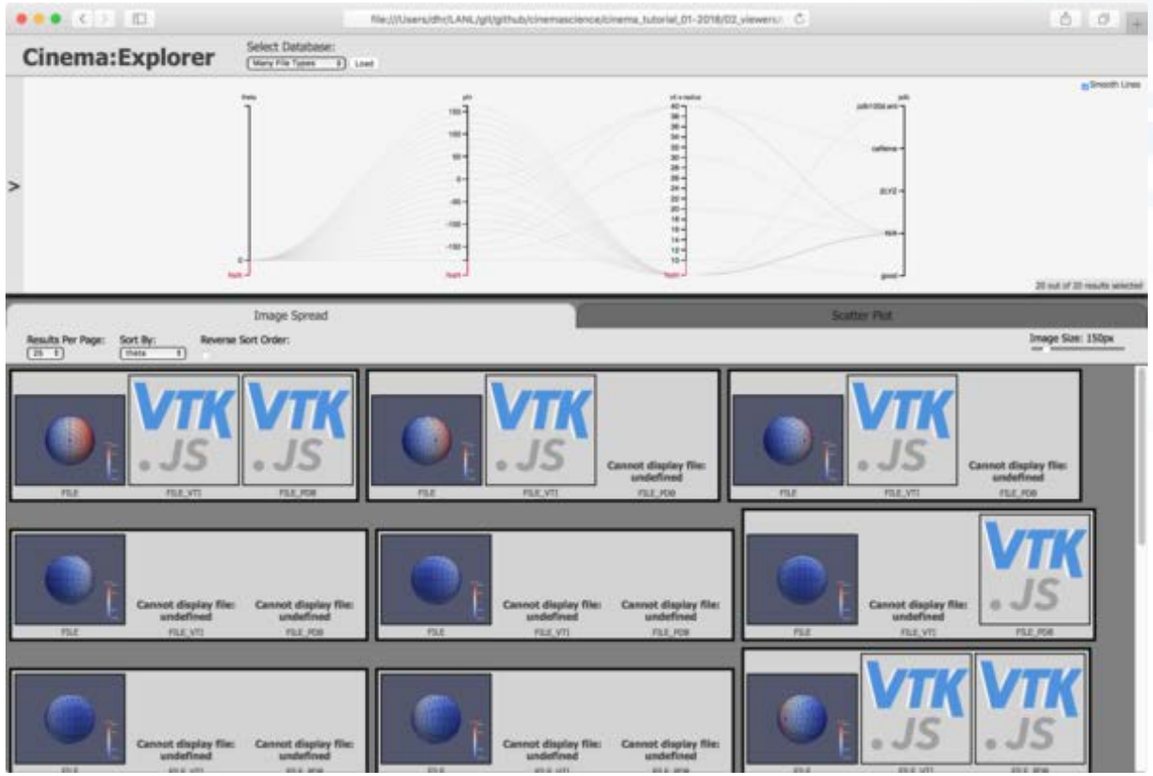
- *Cinema Explorer* link in **tutorial home page**
- select "Multiple Artifacts" from the database list at the top, press `load` button



Cinema Explorer viewer supports multiple data types

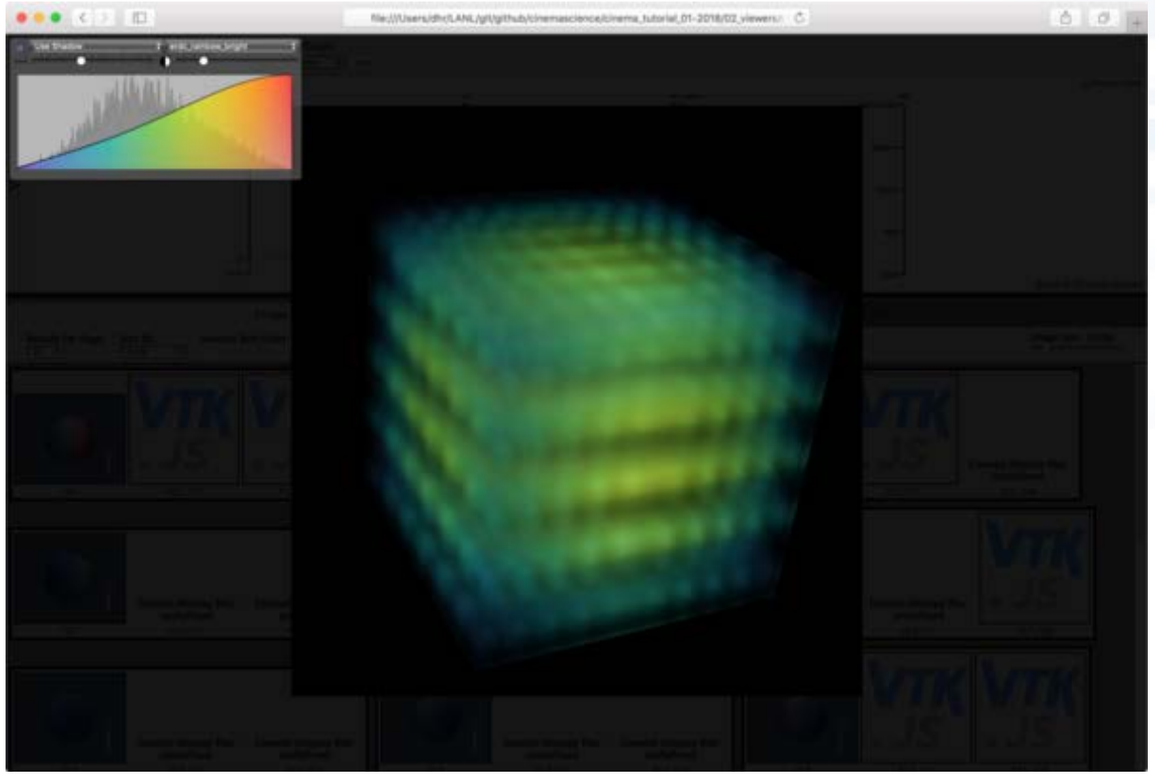
When viewing this data, `Cinema:Explorer` shows thumbnails for each type of data that it can view.

- Clicking on the thumbnail brings up an appropriate viewer for that type of data, if possible.
- **NOTE:** `Cinema:Explorer` indicates which types of files it doesn't have viewers for.



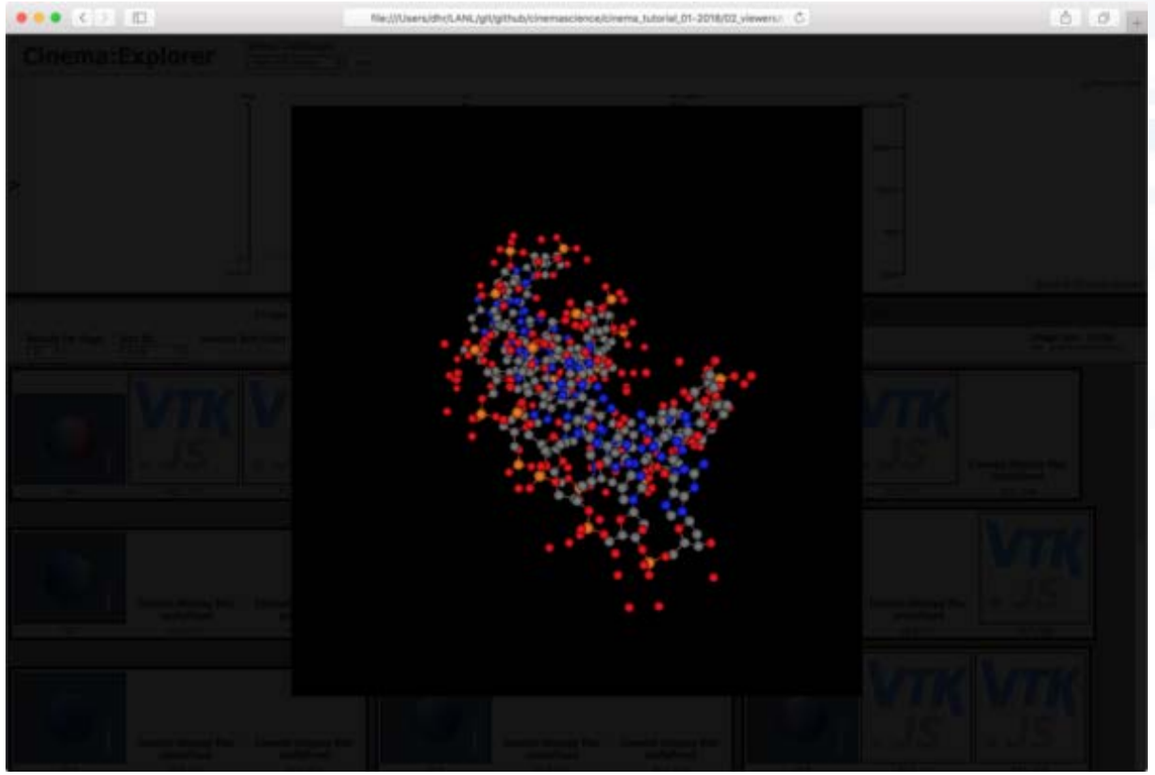
Cinema Explorer viewer supports multiple data types

ParaView/VTK `.vti` files can be interactively viewed ...



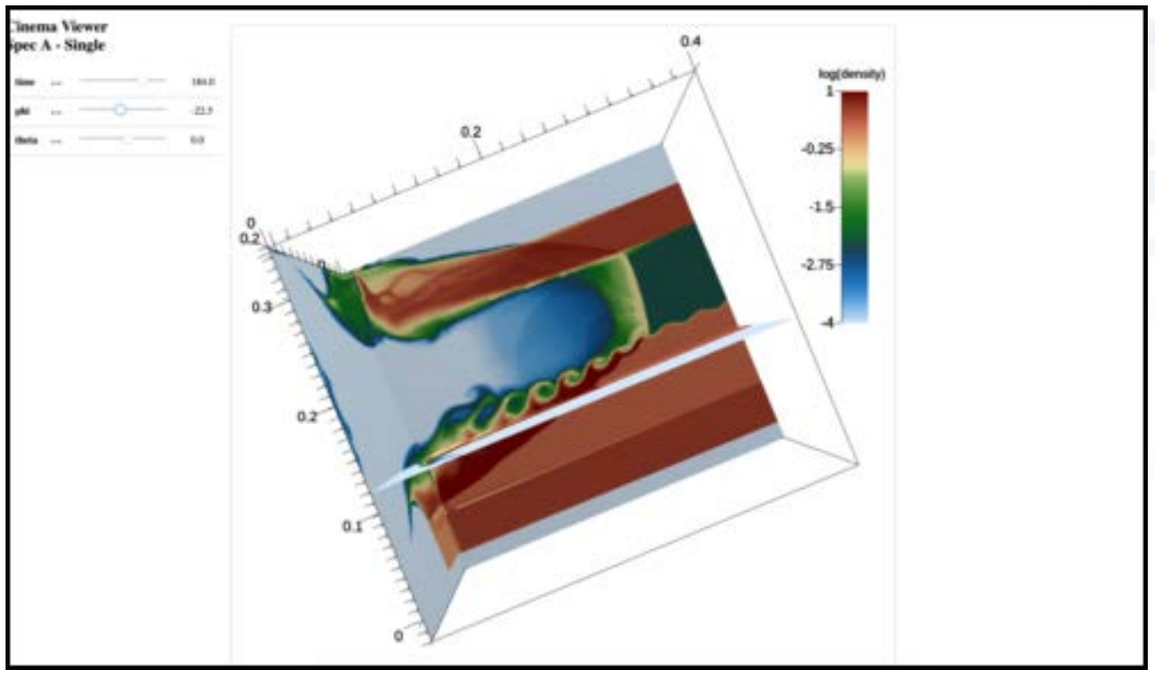
Cinema Explorer viewer supports multiple data types

ParaView/VTK `.pdb` files can be interactively viewed ...



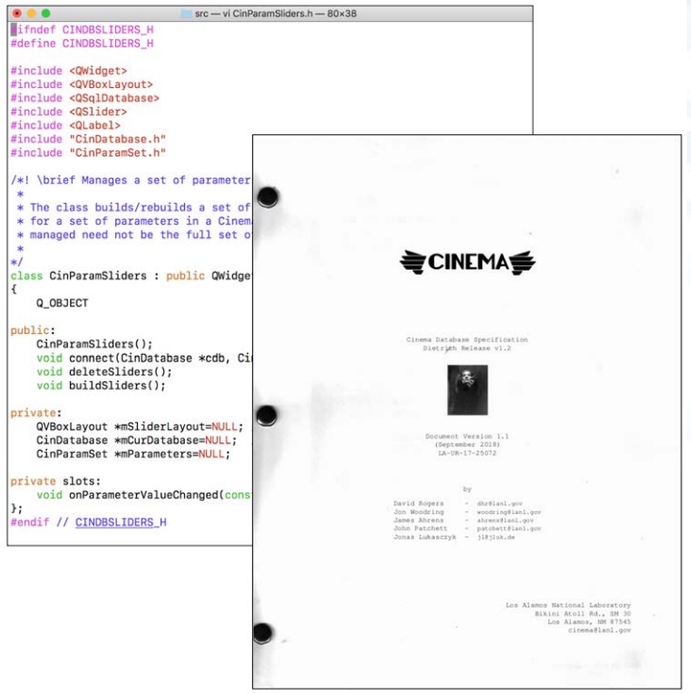
How else can we create a Cinema database?

- Directly output Cinema from your code
  - Refer to the specification
- ASCENT
  - **in-situ** output
- ParaView
  - application
  - **in-situ:** ParaView catalyst
    (instructions on cinema website)
- VisIt
  - application (instructions on cinema website)



Cinema Viewers

`cinemascience` github organization includes several viewer repositories

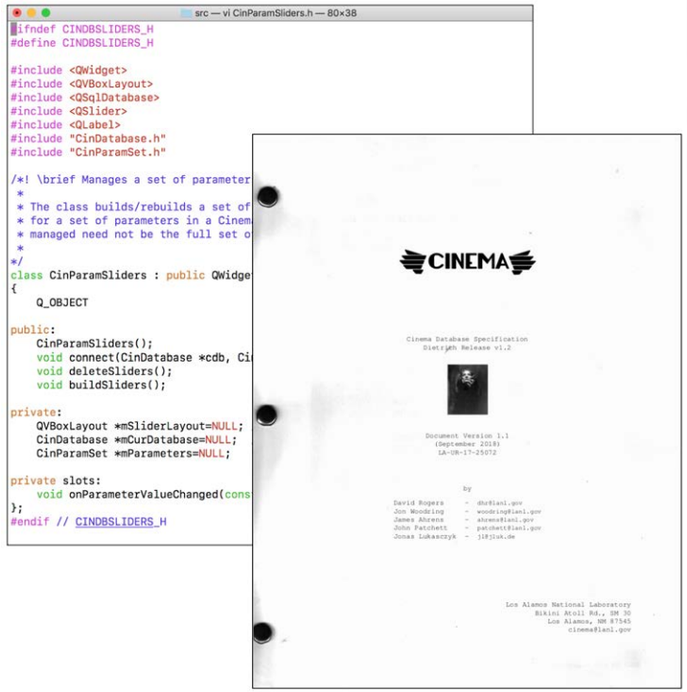


- **Cinema:Components**, reusable browser-based components for viewers.
- **Cinema:View**, a basic viewer to explore one or more databases.
- **Cinema:Explorer**, a more full-featured viewer to explore databases.
- **Cinema:Scope**, a cross-platform viewer application under development.

Cinema Viewers

Viewers and components can be used in a variety of ways:

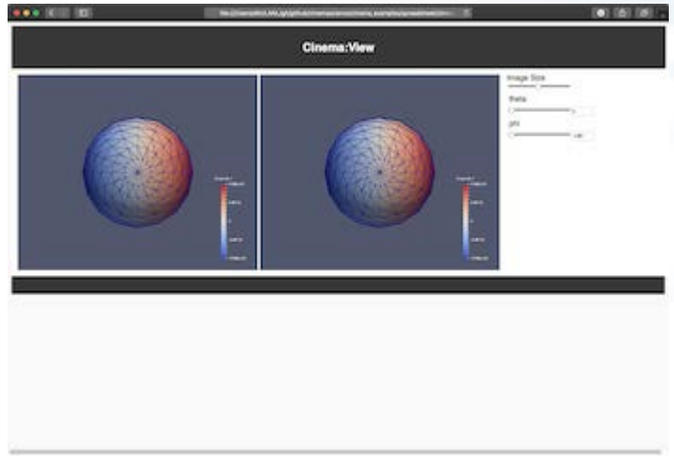
- Small files saved along with Cinema databases
  - Useful when web access is restricted
  - Requires no infrastructure support
    - Ease of support
    - Long shelf life
- Online, through web server
  - Common use case
  - Cinema will be supported natively on ECP Concur



Viewer `Cinema:View`

Browser-based basic viewer for image-based output

- Basic viewer, can be easily edited to view multiple databases
- Sliders control all windows
- `Cinema:Compare` will do its best to show images as the parameters are manipulated (missing images will not cause problems).



Viewer `Cinema:View`

To modify this application, edit the `index.html` file to point to a list of databases you'd like to view:

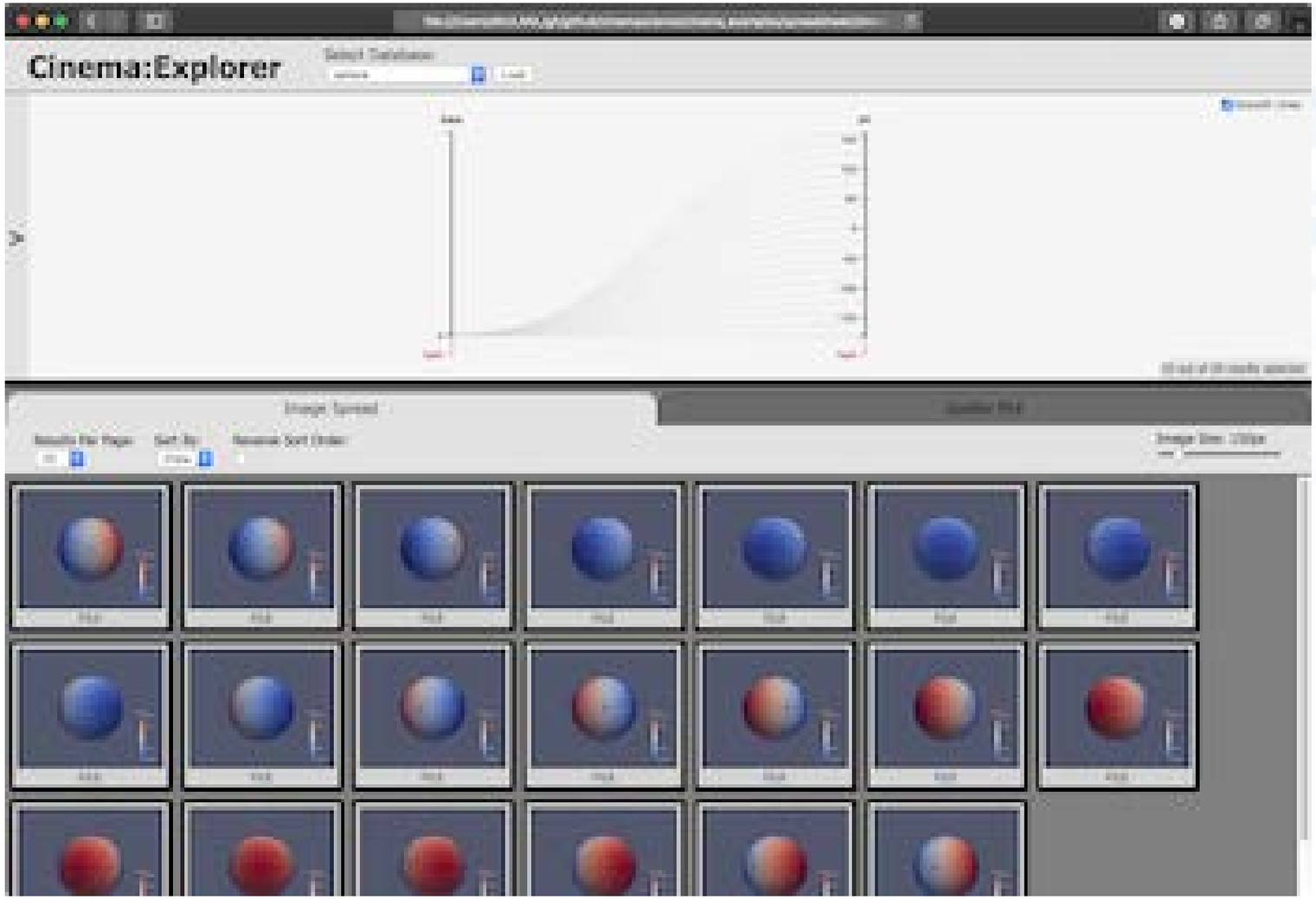
- tutorial file: `materials/databases.json`

```
<!doctype html>
<html>

<head>
    <meta charset="utf-8">
    <link rel='stylesheet' href='cinema/compare/1.0/css/compare.css'>
    <link rel='stylesheet' href='cinema/compare/1.0/css/common.css'>
    <script src='https://d3js.org/d3.v4.min.js'></script>
</head>
...
    // START: Array of databases to view
    var dataSets = [ "data/sphere.cdb", "data/sphere.cdb" ];
    // END : Array of databases to view
...
```

Viewer `Cinema:Explorer`

Browser-based Explorer for general databases



Cinema:Explorer

To use this application, edit the `databases.json` file referenced in the html:

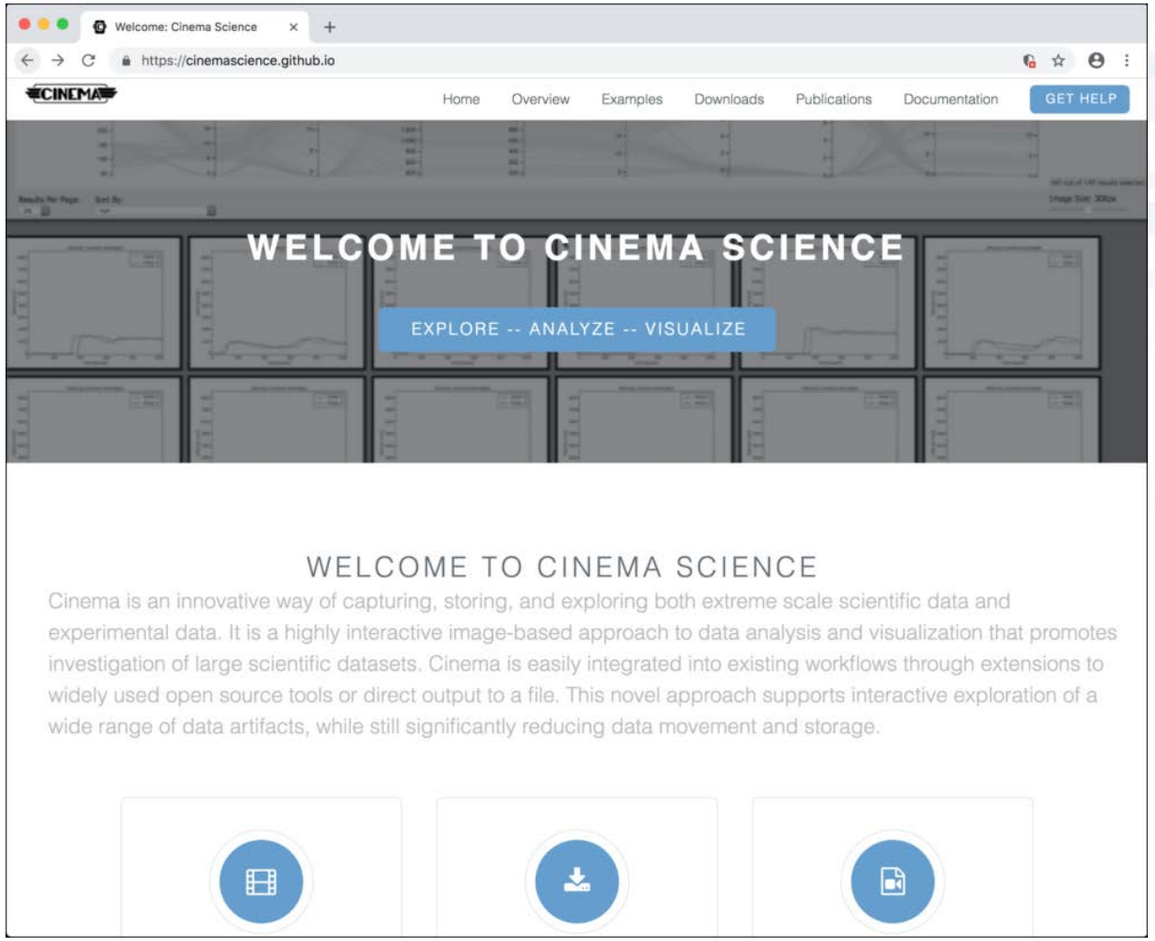
```
[
    {
        "name" : "sphere",
        "directory" : "data/sphere.cdb"
    },
    {
        "name" : "Sphere Multi-Image",
        "directory" : "data/sphere_multi-image.cdb"
    },
    {
        "name" : "Multiple Artifacts",
        "directory" : "data/multiple_artifact.cdb"
    }
]
```

Online Resources

- This Tutorial
  - http://cinemascience.org/tutorials/2019-11_SC
  - https://github.com/cinemascience/cinema_tutorial_2019-11_SC
- Cinema github
  - https://github.com/cinemascience
- Cinema websites
  - http://www.cinemascience.org
  - http://www.cinemaviewer.org
- Cinema Examples
  - http://portal.nersc.gov/project/alpine/2018_ECPReview_Cinema/review.new.html

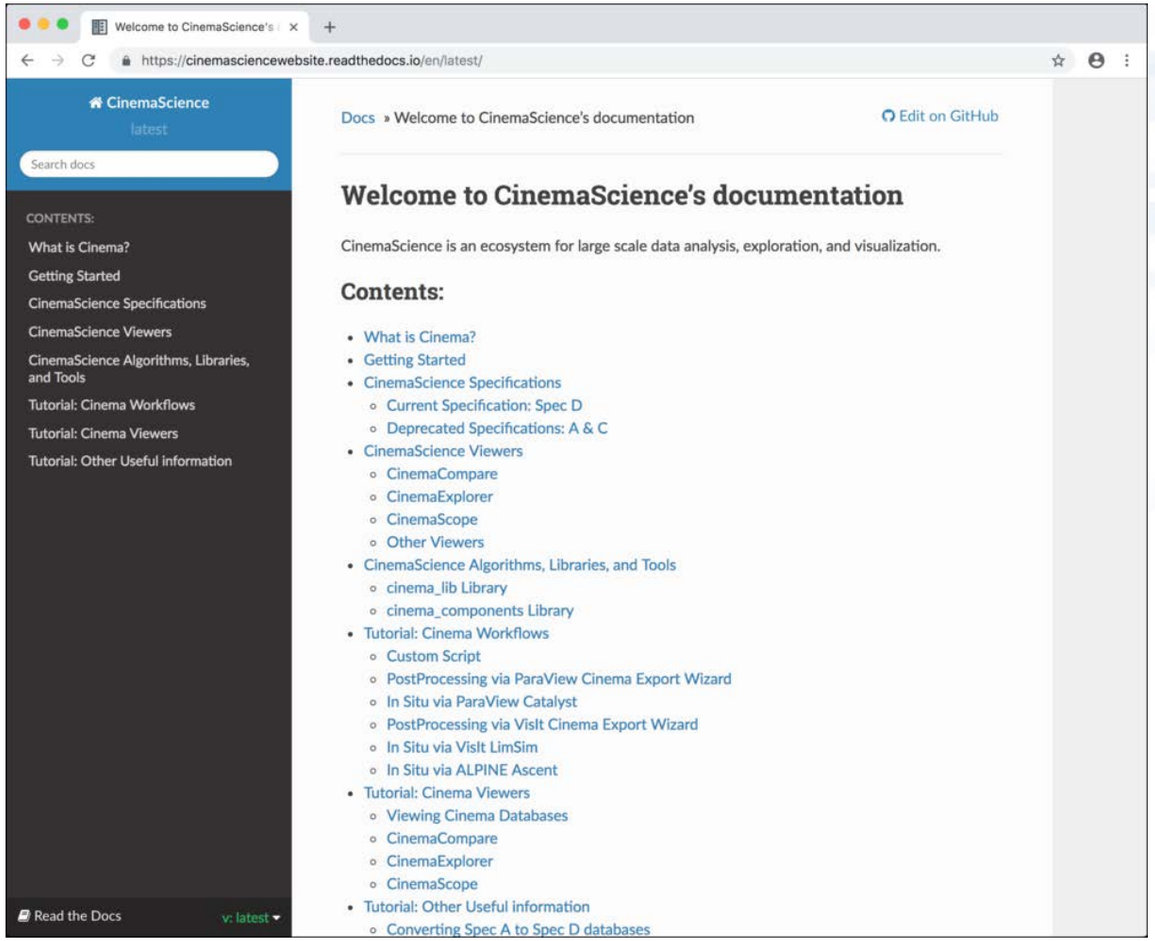
Cinema Website

- Examples (videos)
- Links to code repositories
- Links to expanded documentation



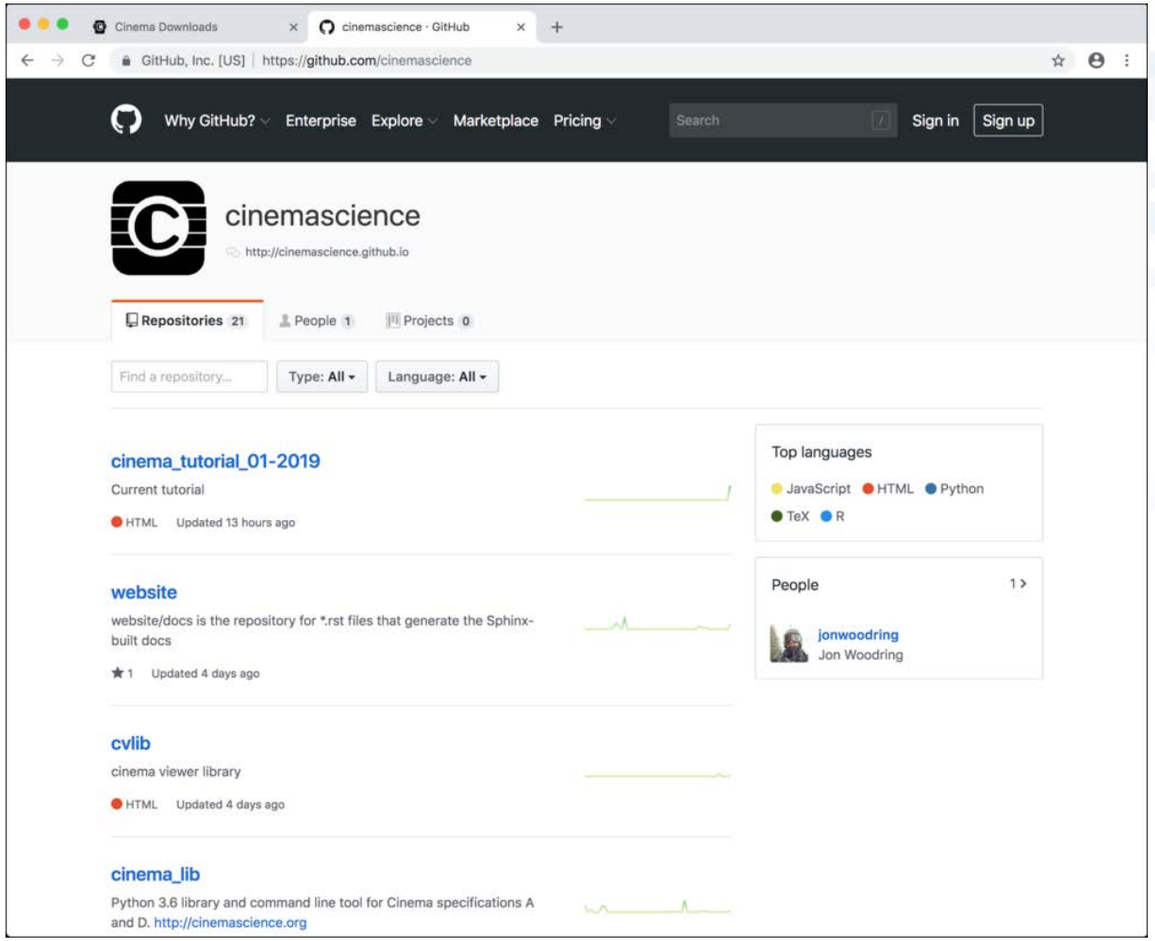
Cinema Website Documentation

- Summarized code examples
- Third party partner instructions
  - ParaView
  - VisIt
- Example datasets



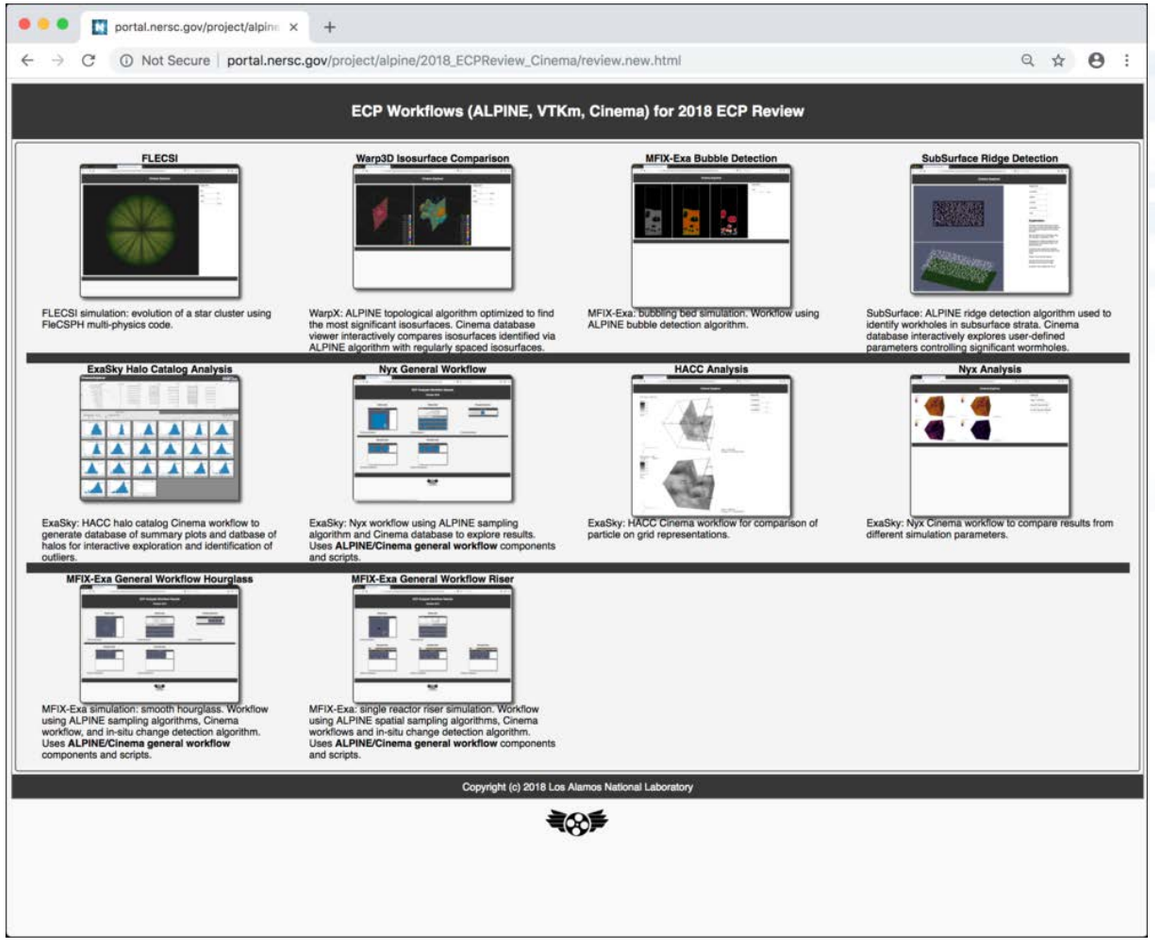
Cinema Github Repositories

- `cinemascience` group
- All public released and in-progress code
  - See individual projects for status



Cinema Examples

- ECP app examples for 2018 review
- View of ongoing workflow development for ECP app analysis and tasks



Questions?

Acknowledgements

- Slides created with `Marp`
- Nyx cosmology simulation code:
  A. S. Almgren, J. B. Bell, M.J. Lijewski, Z. Lukic, E. Van Andel, "Nyx: A Massively Parallel AMR Code for Computational Cosmology" Astrophysical Journal, 765, 39, 2013. https://amrex-astro.github.io/Nyx/