BrainBrowser

API and Architecture

Tarek Sherif
McGill University
Background

- Brain imaging is a fairly young field
  - Became widespread in the 1990s
- Magnetic Resonance Imaging (MRI) has become the most common technique
  - Low invasiveness
  - Lack of radiation exposure
Background

- Areas of Research:
  - Normal brain development
  - Alzheimer’s Disease
  - Multiple Sclerosis
  - Autism
  - Schizophrenia

Alzheimer’s Loss of Cortical Thickness
Multiple Sclerosis Lesions
Normal Brain Development in Children
Background

- Brain imaging research workflow involves:
  - Scanning
    - Get intensity data representing structural or activation patterns (structural vs. functional MRI)
  - Computational Analysis
    - Extract information of interest from the data (e.g. cortical thickness, tissue classification, gyrification)
  - Statistical Analysis
    - Determine significance of results
  - Visualization and Quality Control
BrainBrowser

- A web-based set of tools for visualizing neurological data through modern standards-compliant browsers
  - Volume Viewer
  - Surface Viewer
    - Web Service API available
BrainBrowser

- Technologies used:
  - HTML
  - CSS
  - JavaScript
  - Canvas
  - WebGL (three.js)
  - Web Workers
  - CORS (for the web service)
Volume Viewer

- Navigate 3D MINC volumetric data
- Sagittal, coronal and transverse slices shown from a particular point in 3D space
Volume Viewer

- Entry point is `BrainBrowser.VolumeViewer.start()`
  - viewer object is passed to callback
  - manipulate it to control the app

```javascript
BrainBrowser.VolumeViewer.start("viewer_div", function(viewer) {
  // Manipulate viewer to control the app
});
```
Volume Viewer

- Event model

```javascript
BrainBrowser.VolumeViewer.start("viewer_div", function(viewer) {

    viewer.addEventListener("ready", function() {
        //...
    });

});
```
Volume Viewer

- Load your volumes.

```javascript
BrainBrowser.VolumeViewer.start("viewer_div", function(viewer) {
    viewer.loadVolumes({
        volumes: [{
            type: "minc",
            header_url: "brain1.mnc?headers=true",
            raw_data_url: "brain1.mnc?raw_data=true"
        }]
    });
});
```
Volume Viewer

- Workflow:

  - Parse Raw MINC Data (binary)
  - Get Slice
  - Map Colors
  - Paint to Canvas (Image Data API)
Surface Viewer

- Real-time 3D visualization of surface files
- Apply color data representing different information about the surface (cortical thickness, correlations, etc.)
Surface Viewer

- Entry point is `BrainBrowser.SurfaceViewer.start()`
  - viewer object is passed to callback
  - manipulate it to control the app

```javascript
BrainBrowser.SurfaceViewer.start("viewer_div", function(viewer) {
  // Manipulate viewer to control the app
});
```
Surface Viewer

- Event model

```javascript
BrainBrowser.SurfaceViewer.start("viewer_div", function(viewer) {

  viewer.addEventListener("loadcolormap", function (color_map) {
    //...
  });

  viewer.addEventListener("displaymodel", function(model) {
    //...
  });
});```
Surface Viewer

- Workflow:
Render Scene

BrainBrowser.SurfaceViewer.start("viewer_div", function(viewer) {

    viewer.render();

});
Render Scene

```
var renderer = new THREE.WebGLRenderer(...);
var scene = new THREE.Scene();
var camera = new THREE.PerspectiveCamera(...);

renderer.render(scene, camera);
```
Load Model

BrainBrowser.SurfaceViewer.start("viewer_div", function(viewer) {

    viewer.loadModelFromURL("brain.obj", {
        format: "MNIObj",
        complete: function() {
            // Do something when done
        }
    });
});
Load Model

- Model Load Workflow:

  - Get Model Data
  - MNI OBJ Worker
  - Wavefront OBJ Worker
  - Freesurfer ASC Worker
  - De-index Worker
  - Create Geometry
Load Model

- Get Model Data:
  - AJAX request for URL
  - FileReader API to load from a local file
Load Model

- Workers to parse model data:
  - A Web Worker is defined for each supported file type
    - Parse the file and return vertices, colors, indices and normals
  - Plugin architecture
    - Define a new worker and you can support a new file type
  - Currently supported file types:
    - MNI OBJ
    - Wavefront OBJ
    - Freesurfer ASC
**Load Model**

- **De-index Models**
  - Most data formats encode indexed models
  - WebGL indices can only be 8-bit or 16-bit unsigned integers
    - Limit of 65536 vertices for indexed models
  - BrainBrowser uses models that are much larger than this
  - Models are de-indexed by a Web Worker before being used

<table>
<thead>
<tr>
<th>Model</th>
<th>Number of Vertices</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brain</td>
<td>81924</td>
</tr>
<tr>
<td>DTI</td>
<td>478750</td>
</tr>
<tr>
<td>Aeroplane</td>
<td>187358</td>
</tr>
</tbody>
</table>
Load Model

- Create geometry:
  - Was using THREE.Geometry
    - Convenience classes that manage everything: THREE.Vertex, THREE.Color, THREE.Face
  - TOO SLOW:
    - Object creation becomes a bottleneck
    - Updates have to traverse a tree-like structure
Load Model

- Create geometry:
  - Now using THREE.BufferGeometry
    - Still pretty convenient
    - Directly define the vertex, color and normal buffers that will be sent to WebGL
    - MUCH FASTER
Color Model

```
BrainBrowser.SurfaceViewer.start("viewer_div", function(viewer) {

    viewer.loadColorMapFromURL("spectral.txt");

    viewer.loadIntensityDataFromURL("cortical_thickness.txt");

});
```
Color Model

Get Intensity Data → Intensity Data Worker → Map Colors → Update Color Buffer
Color Model

- Get Intensity Data
  - Intensity data is the raw information we're visualizing
    - Original scan or computation run on the original data
  - Can be loaded from a URL or a local file
Color Model

- Parse Intensity Data
  - Intensity data comes in as text
  - Web Worker parses it into array
Color Model

- Map Colors
  - Map intensities to colors based on the current color map
- Color can be tweaked based on several parameters:
  - Minimum and maximum intensity threshold
  - Clamping
  - Flipping the intensity to color map relationship
Color Model

- Update the Color Buffer
  - Write colors to the color buffer
  - Signal that the colors must be updated in WebGL

```javascript
geometry.attributes.color.needsUpdate = true;
```
BrainBrowser.SurfaceViewer.start("viewer_div", function(viewer) {

    viewer.setWireframe(true); // Wireframe mode

    viewer.setTransparency("left_hemisphere", 0.5); // Transparency

    viewer.setIntensityRange(0.5, 1.5); // Intensity range

    viewer.autorotate.x = true; // Autorotate around the x axis

    // AND MORE!!
});
Surface Viewer Web Service

- Simple GET HTTP request for widget HTML to load into page

```html
<div id="display"></div>
<script>
$("#display").load(
    "https://brainbrowser.cbrain.mcgill.ca/surface-viewer-widget?" +
    "version=1.4.1&" +
    "model=brain.obj&" +
    "intensity_data=cortical_thickness.txt&" +
    "color_map=spectral.txt&" +
    "width=100&" +
    "height=100"
);
</script>
```
Surface Viewer Web Service

- Request a specific version of BrainBrowser
  - Prevent widget from breaking if BrainBrowser is updated
  - Appropriate version of three.js is automatically loaded
    - (unless client requests otherwise)
Surface Viewer Web Service

- Control viewer programmatically by defining a viewer callback

```html
<div id="display"></div><input id="wireframe" type="checkbox"/>
<script>
function init(viewer) { // Define callback function
    $('#wireframe').change(function() {
        viewer.setWireframe($(this).is(":checked")); // Use viewer object
    });
}

$('#display').load(
    "https://brainbrowser.cbrain.mcgill.ca/surface-viewer-widget?" +
    "version=1.4.1&model=brain.obj" +
    "viewer_callback=init" // Pass callback name as parameter to web service
); //
</script>
```
Thanks!
BrainBrowser

**BrainBrowser:** https://brainbrowser.cbrain.mcgill.ca/
**CBRAIN:** http://cbrain.mcgill.ca/

**Questions?** tsherif@gmail.com

**Credits:**
- **Lead Developer:** Tarek Sherif
- **Original Author:** Nicolas Kassis
- **Contributing Developer:** Mia Petkova
- **Consultant:** Samir Das
- **CBRAIN System Architect:** Marc Rousseau
- **CBRAIN Manager:** Reza Adalat
- **Principal Investigator:** Alan Evans

---

The CBRAIN Project was funded by:

[McGill University](https://www.mcgill.ca)
[Canarie](https://canarie.ca)