**cs 529 @ UIC:  Visual Data Science**

**Fall 2022**

**Homework #3: Flow Visualization with Three.js**

**Out:  Tuesday, Sep 27 2022**

**Due: Thursday, Oct 13th 11:59pm**

**1. Overview**

By the end of this individual project, you:

* will have learned/practiced three.js skills, and learned how to display a 3D model on the web
* will have learned how to load and display a 3D point cloud on the web
* will have embedded the 3D point cloud within a nice 2D visualization
* will have practiced the concepts of linked views, brushing and linking, view manipulation and filtering

Collaboration is not allowed on this project. However, you are encouraged to post questions and answer any questions related to web programming using the Piazza account for the course. Remember to tag the post as “Hw 3”. If you get stuck at any point either in a tutorial or in your solution, by all means post---you have nothing to lose. Correct and helpful answers to questions posted by classmates will get the poster extra credit. Remember that you can always post anonymously, if you so wish.

Please note the homework is calibrated to take less than a week to complete, while balancing your midterm and other classes at the same time. You can get started on Hw03 as soon as it’s posted, to get a sense of whether you’d like working with spatial data for your final project, but it won’t be due for another week after your VDS midterm.

**Tutorial:**Three.js is extremely well documented, and there are many examples available under the threejs.org website: [http://threejs.org/docs/](https://www.google.com/url?q=http://threejs.org/docs/&sa=D&source=editors&ust=1665353367741703&usg=AOvVaw1TCtKw3COO3e3cCfe-KBlq). Start by completing the [Creating a Scene](https://www.google.com/url?q=https://threejs.org/docs/%23manual/en/introduction/Creating-a-scene&sa=D&source=editors&ust=1665353367741869&usg=AOvVaw3OImgQlZ36hxN8RfONZ6Aq) tutorial on the same threejs.org website. The three.js terminology abuses slightly the term “mesh” (all geometries are meshes to them).

Next, read through the [Matrix Transformations](https://www.google.com/url?q=https://threejs.org/docs/%23manual/en/introduction/Matrix-transformations&sa=D&source=editors&ust=1665353367742169&usg=AOvVaw3nPjnpcSEaIxcKqDvbFfZu) manual page (underneath the first tutorial), to get a sense of how to use geometric transformations; we won’t need quaternions for this assignment. Experiment with scaling and translating the cube you created in the first tutorial example (for your own sake, you might want to use first Line to draw the coordinate system axes; a line segment from 0,0,0 to 1,0,0 for the X axis etc). Also experiment with swapping the order in which you specify transformations. What happens if you first translate, then rotate the cube? What if you rotate first, then translate?

Now that you have a mastery of geometric transformations, check out the manual entries under Geometries (in particular, the BoxGeometry and the CylinderGeometry). Add a cylinder to your scene.

For fun, you might want to check out the “How to create VR content” manual entry as well. The webpage tells you what you need to do to enable VR, and it’s best if you go ahead and check out one of the examples they give. An easier alternative for VR than straight three.js is [http://aframe.io](https://www.google.com/url?q=http://aframe.io/&sa=D&source=editors&ust=1665353367742629&usg=AOvVaw2WXZ7Swhtox674D9OyEWJh), which is a framework on top of threejs. All the VR aspects are handled by aframe, while the rendering and graphics parts are handled by three.js. Furthermore, aframe exposes the three.js API, so you'd still be working in three.js to build your scene. Additionally, aframe uses an html-like syntax that is easy to pick up.

Now, on to the actual assignment.

**2. Instructions**

For this assignment, we will be working with a computational fluid flow simulation dataset from the San Diego Supercomputing Center. You can learn more about the science behind the simulation here: [http://www.uni-kl.de/sciviscontest/](https://www.google.com/url?q=http://www.uni-kl.de/sciviscontest/&sa=D&source=editors&ust=1665353367743233&usg=AOvVaw3Aa7adx4I8sZWF5ygpfwB_)

We are providing you with a data set, and example Three.js code for loading up the dataset, and for displaying a cylinder geometry (courtesy of EVL students Andrew Burks and Juan Trelles):

[https://github.com/uic-evl/cs529-vds](https://www.google.com/url?q=https://github.com/uic-evl/cs529-vds&sa=D&source=editors&ust=1665353367743567&usg=AOvVaw21y9X-4zHYIeWKSnbEOigv)

Alternatively, you could access the point cloud example and Starter code (run 14) directly from here :

[https://drive.google.com/file/d/1vhn1m3OqxygGddzPdi4F5-PHBE1eoUxF/view?usp=sharing](https://www.google.com/url?q=https://drive.google.com/file/d/1vhn1m3OqxygGddzPdi4F5-PHBE1eoUxF/view?usp%3Dsharing&sa=D&source=editors&ust=1665353367743899&usg=AOvVaw1cd3zSdPJEqgJXOY3Fh9rp)

and the actual data to read in from here (you’ll only need one timestep):

[https://drive.google.com/open?id=1tzopK6xqQkefopSAw1ik5rXMVCJWS978](https://www.google.com/url?q=https://drive.google.com/open?id%3D1tzopK6xqQkefopSAw1ik5rXMVCJWS978&sa=D&source=editors&ust=1665353367744150&usg=AOvVaw0VmFACNrZxPpHU5S_-OaVj)

But the code will have fewer comments and may be less clean.

Here is what you need to do:

1. Show the data (see the dummy createParticleSystem function in the support code) as a point cloud; get rid of the cylinder outline once you have the data (20p)

The points2 value in the data corresponds to the height of the cylinder. Reading in the data you may be inclined to use 0 1 2 -> x y z, but then your z would correspond to the height of the cylinder. Simply mapping points2 and velocity2 to y and points1 and velocity1 to z should restore the dimensions to the standard.

1. Colormap the point cloud by concentration (10p)
2. Create a second view using D3 and show a vertical 2D slice of the data in it (keep the z value fixed, and that should give you an XY slice) (20p)
3. Allow the user to rotate the cylinder containing the flow (10p)
4. Add a 2D cut-plane filter to the 3D flow (draw a vertical XY rectangle in the 3D view, and let the user move the rectangle through the flow along the Z axis) (10p)
5. Link the 3D view and the 2D view through the rectangle filter (i.e., show in the 2D view only the 3D points inside the rectangle); hint: use the rectangle’s Z current position to select only those cloud points with matching Z (10p)
6. Implement a naive brushing and linking operation (gray out all the 3D points outside the rectangle cut-plane) (10p)
7. Use D3 to add more controls or text to the interface. (10p)

Please submit a snapshot of and a link to your solution on the Piazza forum for the class.

Good luck, and have fun!

**Additional notes:**

The startup code uses the latest version of Three.js and d3.js.

Three.js has gone through some syntax changes in recent years, so if you’ll watch/read any tutorials that are more than a year older, you might find deprecated examples. Always make sure to double check any tutorial materials with the official Three.js website materials.