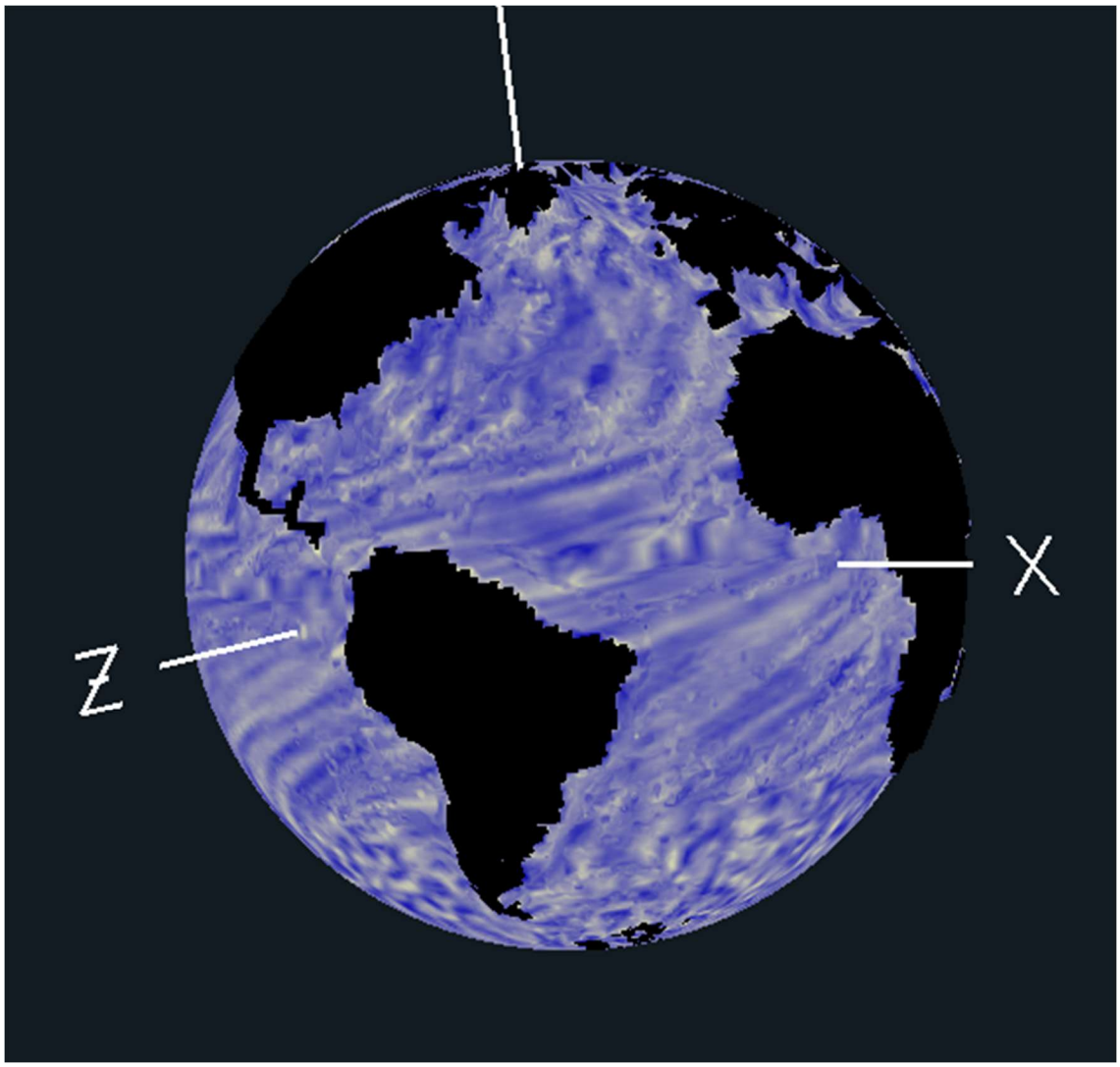
Name: Tran Tran

Email Address: trantr3@oregonstate.edu

CS 557 Computer Graphic Shader

Link: Flow Visualization of Ocean Current - OSU MediaSpace

Final Project – Ocean Current Visualization

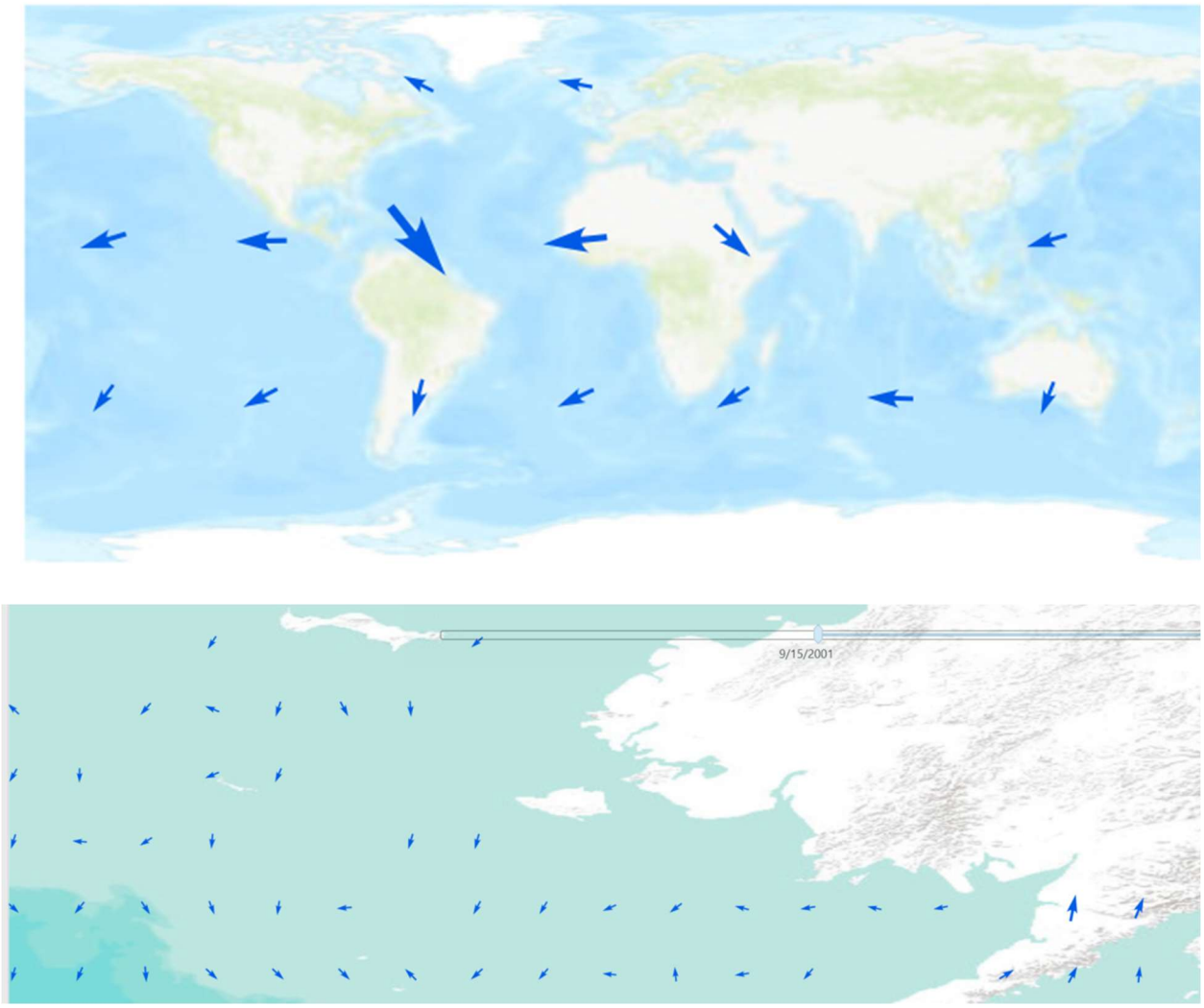


Flow Visualization

The 2D Line Integral Convolution is a technique that uses encoded 2D flow field texture to smear the pixel of an image. The fragment shader takes two textures: encoded flow field texture and the base image (in this case, I use the white noise).

The encoded flow field texture have encoded information of the vector fields that we want to use. Here, I use the red and green component to capture x and y axis value of each vector based on the direction.

Prepare Ocean Current Data

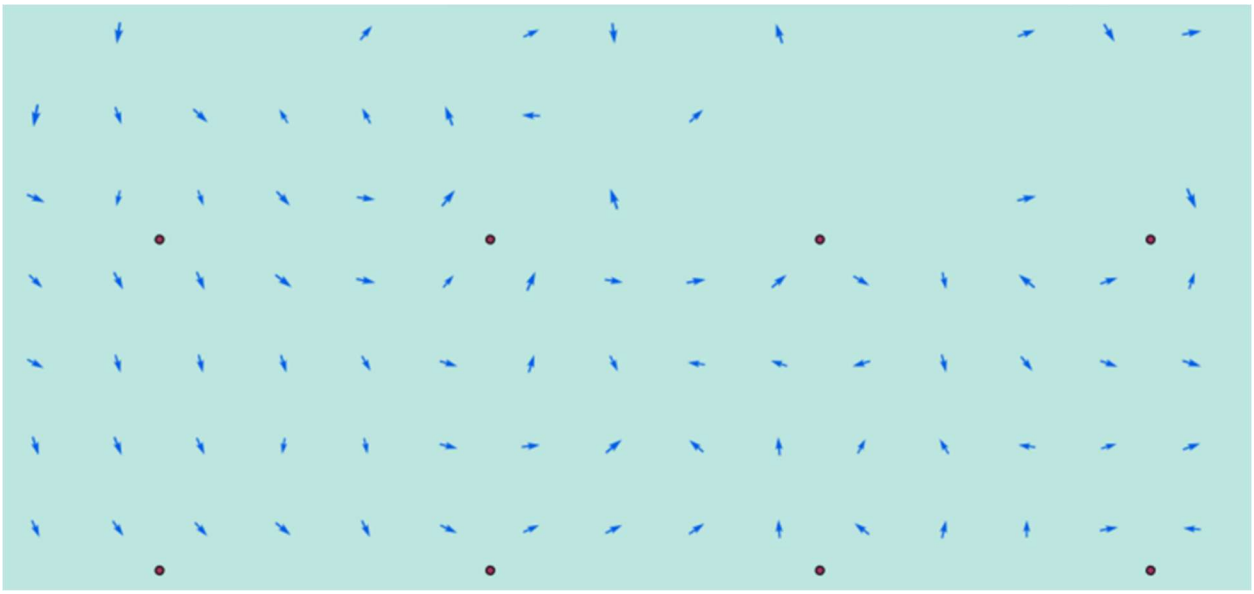


Data Source: Ocean Surface Current Layer

ArcGIS Pro is the common tool to perform geoprocessing on geographical data sets as it is also a host of millions of datasets.

Ocean Surface Current data is hosted as a raster layer. “A raster consists of a matrix of pixels (or cells) organized into rows and columns (or a grid) in which each pixel contains a

value representing information)". In the picture above showing the flow field of the ocean current at each raster cell.



Next, **Aggregate** tool is then used to aggregate the raster layer producing an aggregated raster layer containing the average vector of all the vectors of the small cell. Then we can sample the raster layer to a feature Layer of Points. Each point has the rounded x coordinate (ranging from -180, 180) and y coordinate (ranging from -90, 90), the vector direction.

	OBJECTID *	Shape *	MEAN_L67D3_Vector_Direction	X	Y	X_Unit	Y_Unit	Coord
43	43	Point	52.042632	-1	-48	-0.788469	-0.615075	-1,-48
44	44	Point	75.206382	-1	-49	-0.966852	-0.255338	-1,-49
45	45	Point	2.292414	-1	-5	-0.04	-0.9992	-1,-5
46	46	Point	102.38865	-1	-50	-0.976715	0.214542	-1,-50
47	47	Point	109.32365	-1	-51	-0.943664	0.330904	-1,-51
48	48	Point	104.360507	-1	-52	-0.968754	0.248022	-1,-52
49	49	Point	99.890507	-1	-53	-0.985138	0.171766	-1,-53
50	50	Point	86.567389	-1	-54	-0.998206	-0.059875	-1,-54

Then, with vector direction, we use trigonometric functions to compute the normalized X and Y component

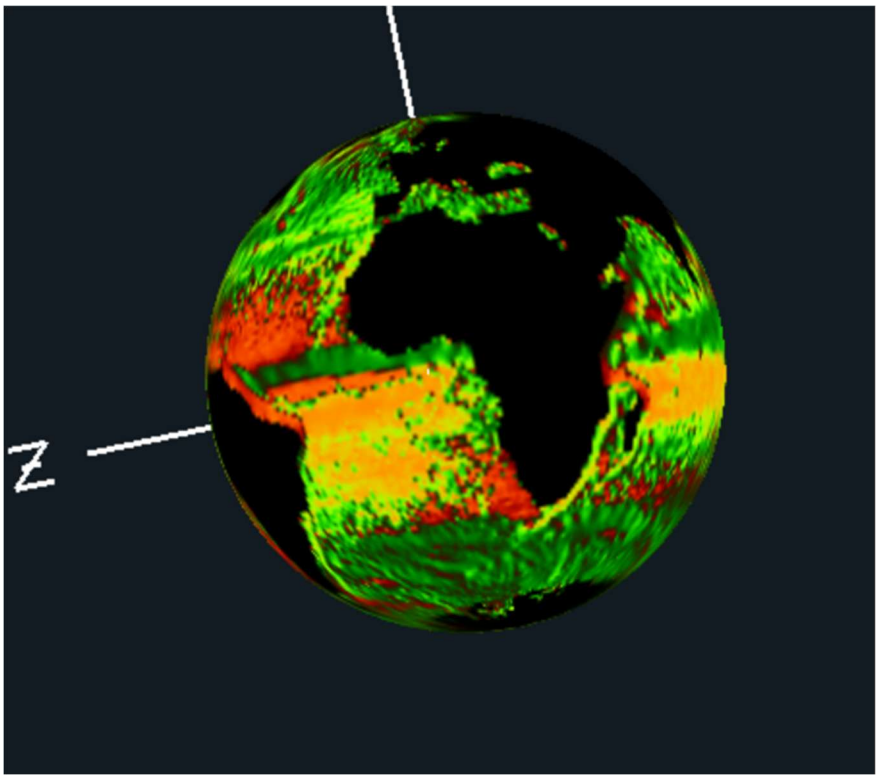
$$\frac{X}{1} = \cos(\theta)$$

$$\frac{Y}{1} = \sin(\theta)$$

The points' coordinates are then clamped to positive s and t texture coordinates

Read table to texture file

Once the table is exported in csv file, I write a function to loop through each entry of the table and create a 2D array of points indexed by s and t value of the points along with its vector value (X_Unit and Y_Unit). A point is also a vector of three integers that store red, green and blue component of the texture. X_Unit is stored as red, Y_unit is stored as green. Then this texture array is written to the texture file to be read in the shader. The picture shows how the encoded texture file looks.

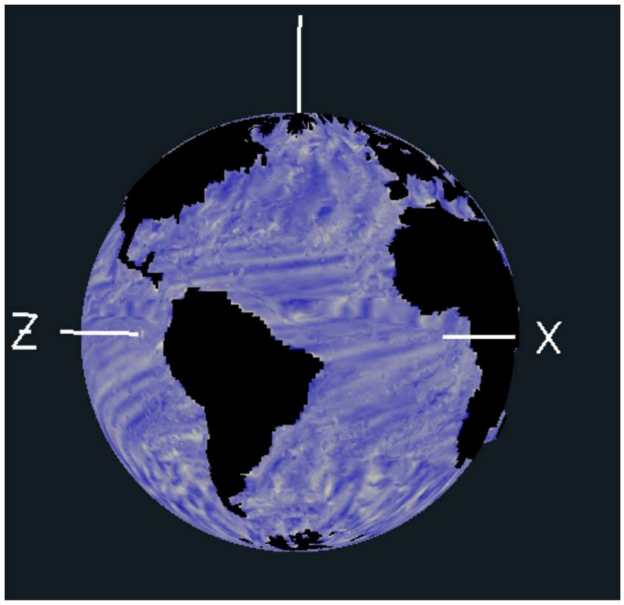


Displacement mapping in Vertex Shader

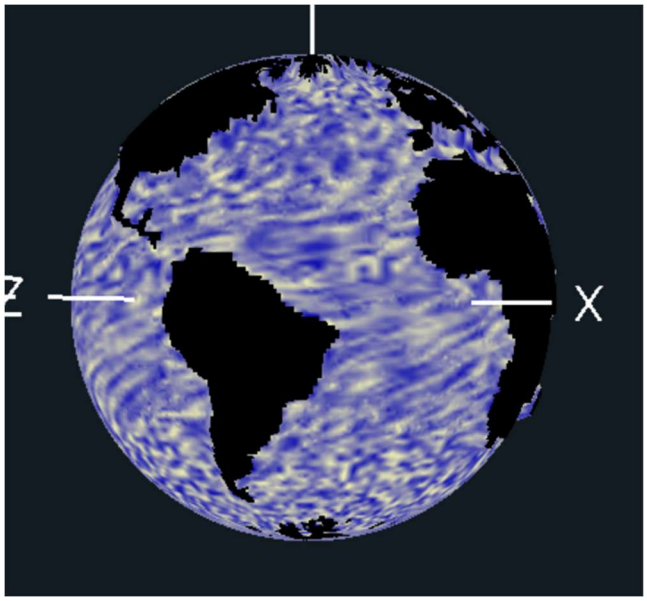
In order to add some elevation to the terrain, every terrain vertex is displaced outward by 0.05. A vertex is on land when the red and green component of the vector is 0

Line Integral Convolution In the fragment shader,

Both the basemap and the flow texture file are read to the fragment shader. Basemap texture is read as a base color. Let's *v* be a 2D vector of the flow at the current fragment and *st* be the texture coordinate. We also use *Length* variable to indicate how long the vector should be. At each fragment, we displace its texture position *st* by the flow vector in both directions: $\vec{v} + \overrightarrow{st}$ and $\vec{v} - \overrightarrow{st}$. Then the fragment color at the displaced texture position is then used to color the current fragment while the land is colored with black.



Visualization of Length = 5



Flow Visualization of Length = 1