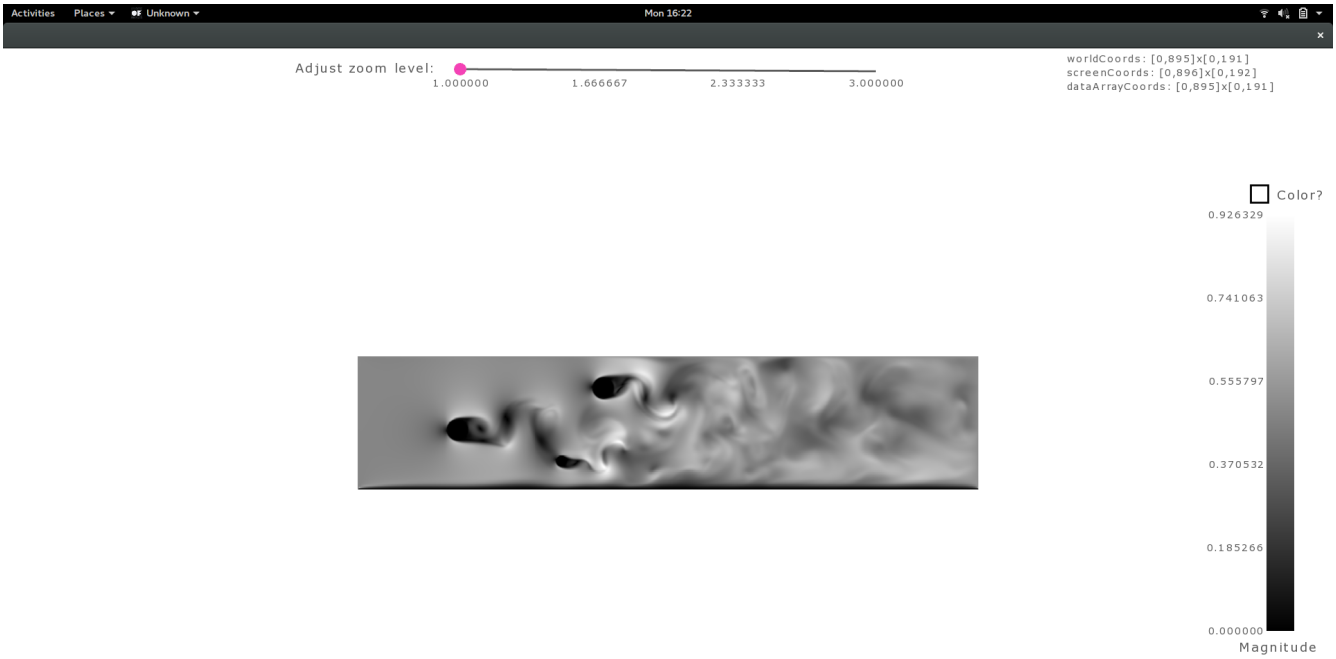
Assignment 07 – Vectors

Direct Visualization

I used bilinear interpolation for computing the vector at a given position for each position in the screen coordinate system. Once I got the vector, I used its magnitude to map to either gray scale or a diverging color map to render a 2D image of the given vector space.

Prior to this, I established the world coordinates, and got vectors for each point in the world coordinate system. These vectors are stored in a separate linear data structure. I also mapped the world coordinates to screen coordinates and screen values are filled by interpolating a vector at each pixel in the screen coordinate system. Here I took care of adding color to each pixel depending on the magnitude of the vector.

Basic direct visualization.



Once the image was drawn on screen, I came up with two possible interactions on it.

1. **Zoom:** I used a slider to enable the user to zoom into (or out of) the image. This slider helps zoom on both x-axis and y-axis proportionally at the same time. It is very similar to a camera lens in working. The picture has maximum bounds to draw the image. When zoomed beyond this bound, the picture gets cropped on the edges, showing a small area of the original picture in zoomed mode.

2. **Pan:** Once you zoom beyond the bounds of the picture, once cannot view certain edges of the original image any more. To help the user to view these hidden areas, I implemented panning. The user can pan by clicking & then dragging the image using mouse. The interface is similar to a touch screen interface where you touch the screen & drag to pan the zoomed image. I felt this was a natural and intuitive way to pan as against using sliders.

   Zoomed in.

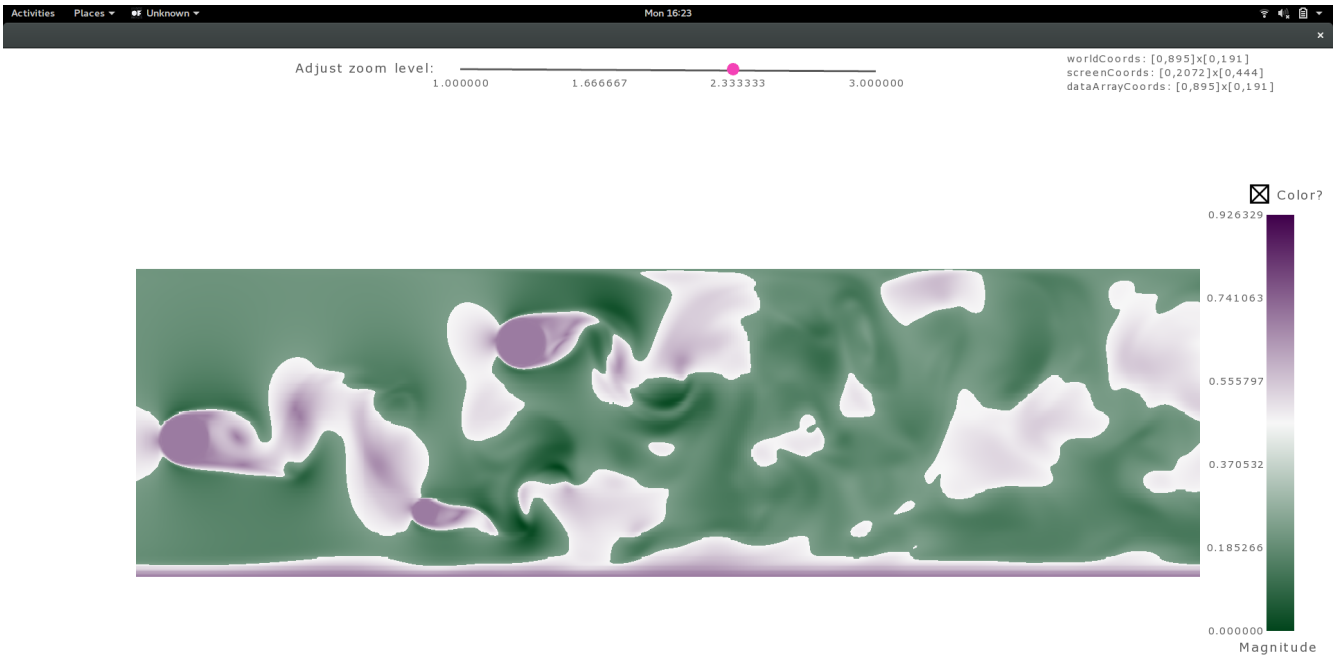
Colormaps: I tried two color maps for the direct visualization. Both of them were useful in their own terms.

1. **Grayscale:** Grayscale was useful in visualizing vector flows because it was very easy to pick up on the waves and have an approximate idea of the direction of the vector and how it bends around objects. Whereas, it was very difficult to distinguish magnitude of the vectors at different regions.
2. **Diverging color map:** Diverging color map was useful where grayscale wasn't. I could easily pick differences in intensities at different regions with this. Although, it was difficult to understand the (waves/direction) flow of vector with color map.

I show the magnitude scale as a color legend on the right side and above it is a check box that is used to switch between diverging and grayscale.

I also displayed the various coordinate system and their bounds on the top right corner of the screen that helps user understand the dataset better in terms of both world & screen space. It also shows current dt value and number of iterations for rk-4 integration.

Diverging color map:



Geometry based visualization

I used Runge-Kutta 4 integration to construct streamlines. I construct the streamlines in world coordinates and convert each point in the line to screen coordinate when rendering. By doing so, I don't have to recompute the streamlines when user zooms/pans/changes color map of the image. By default, I load a default dt as a function of world coordinate system's bounds and keep the number of iterations at 5000. I also enabled user to change these values using two sliders at the bottom. I also added a clear streamlines button at the bottom right corner.

Observations from datasets

In stuart_vortex, I observed the 3 vortexes in the shape of an eye and other wavy streamlines around them. Looks like the motion is inward like a whirlpool for some stream lines. Picture attached below.

In italy, the streamlines aren't as smooth or as organized as before. The dataset looks like a smoke that comes after you extinguish a flame. The streamlines still played a vital role in understanding the flow of the dataset. Picture attached below.

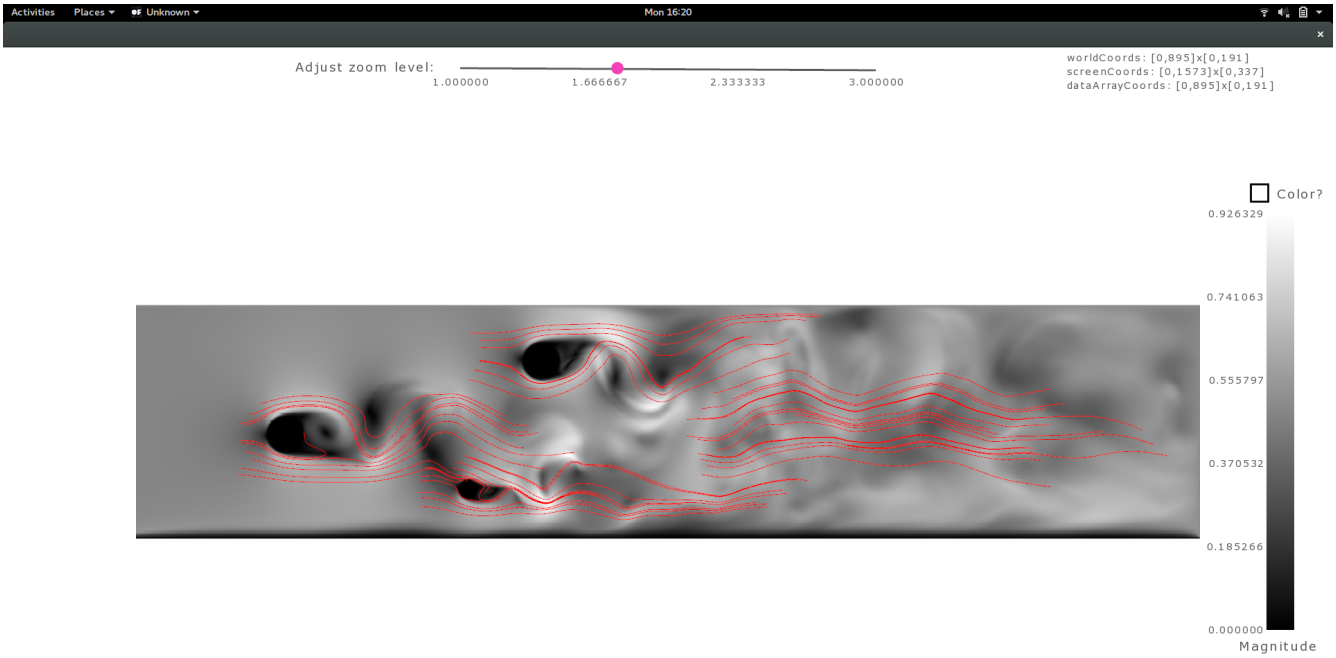
In 3cylflow, even without streamlines, we can clearly see the 3 heads (the dataset looks like sperms in motion). With streamline, the flow and direction is very evident. Picture attached below.

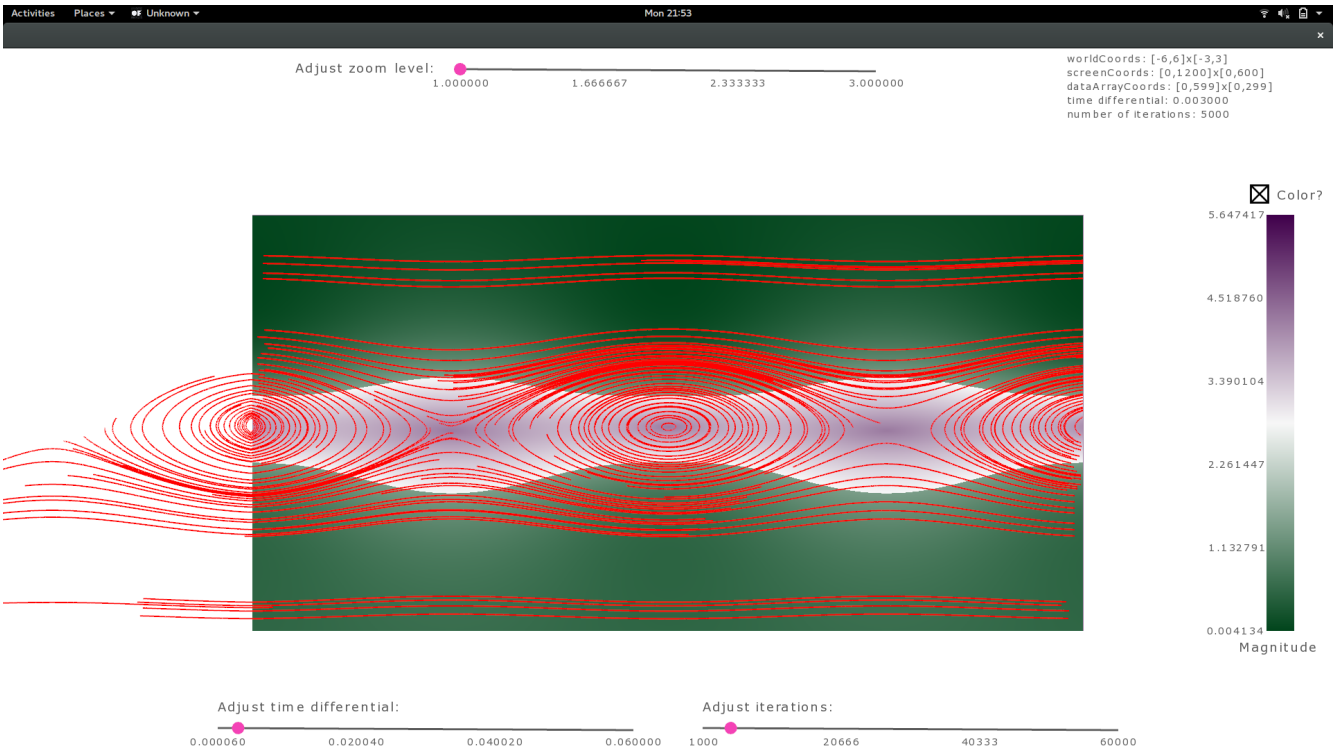
Focus dataset was more interesting than others. The flow was not clearly visible in Direction visualization. But when I started drawing the streamlines, it was very clear that the it was again like a vortex where it is converging to the center. Picture attached below.

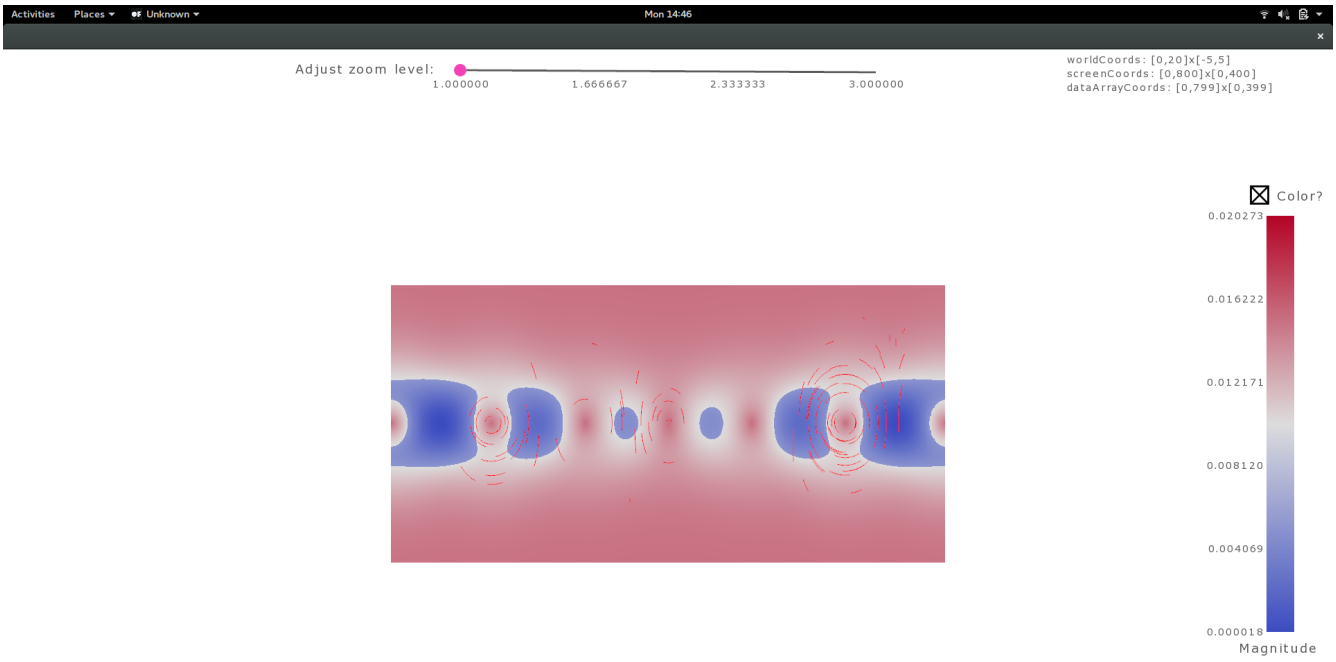
Changing datasets

Change the dataset by modifying first line in the setup function of the ofApp.cpp.

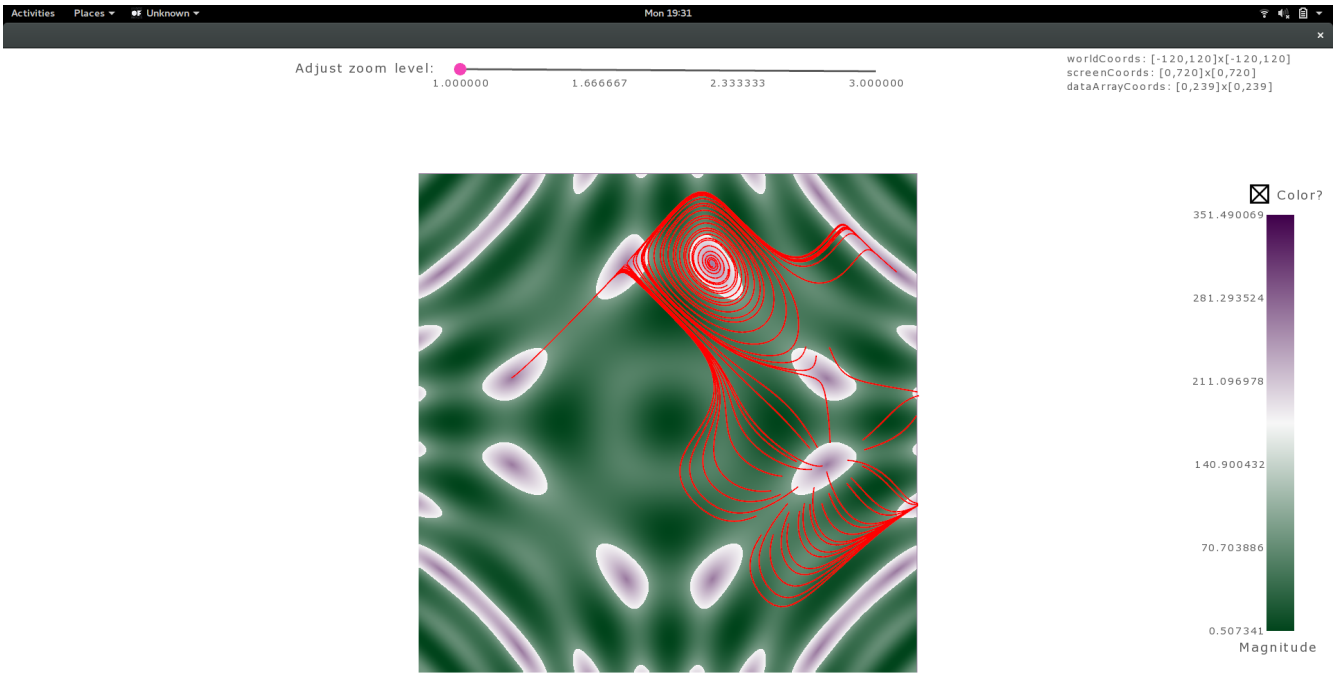
Here are some of the screenshots.



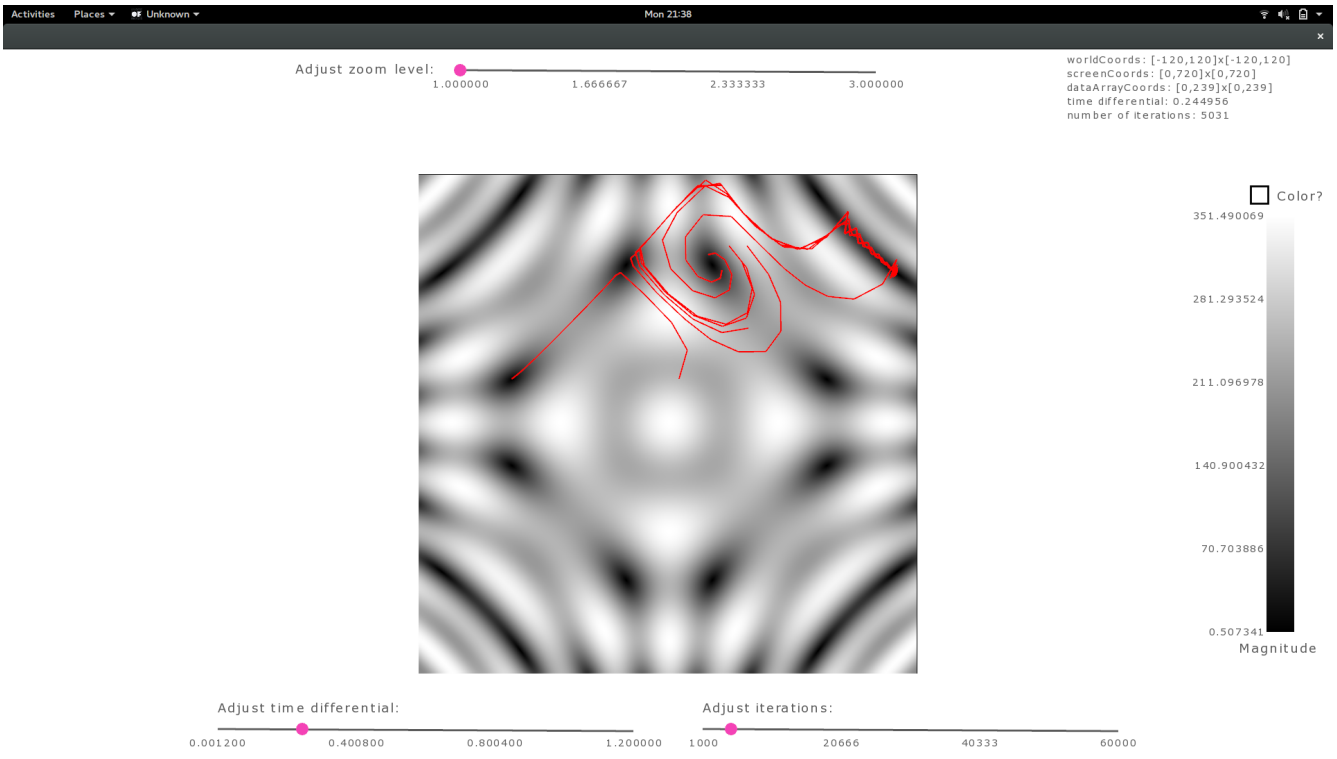

Activities
Places
Unknown
Mon 21:53
Adjust zoom level:
1.000000
1.666667
2.333333
3.000000
worldCoords: [-6,6]x[-3,3]
screenCoords: [0,1200]x[0,600]
dataArrayCoords: [0,599]x[0,299]
time differential: 0.003000
number of iterations: 5000
Color?
5.647417
4.518760
3.390104
2.261447
1.132791
0.004134
Magnitude
Adjust time differential:
0.000060
0.020040
0.040020
0.060000
Adjust iterations:
1000
20666
40333
60000

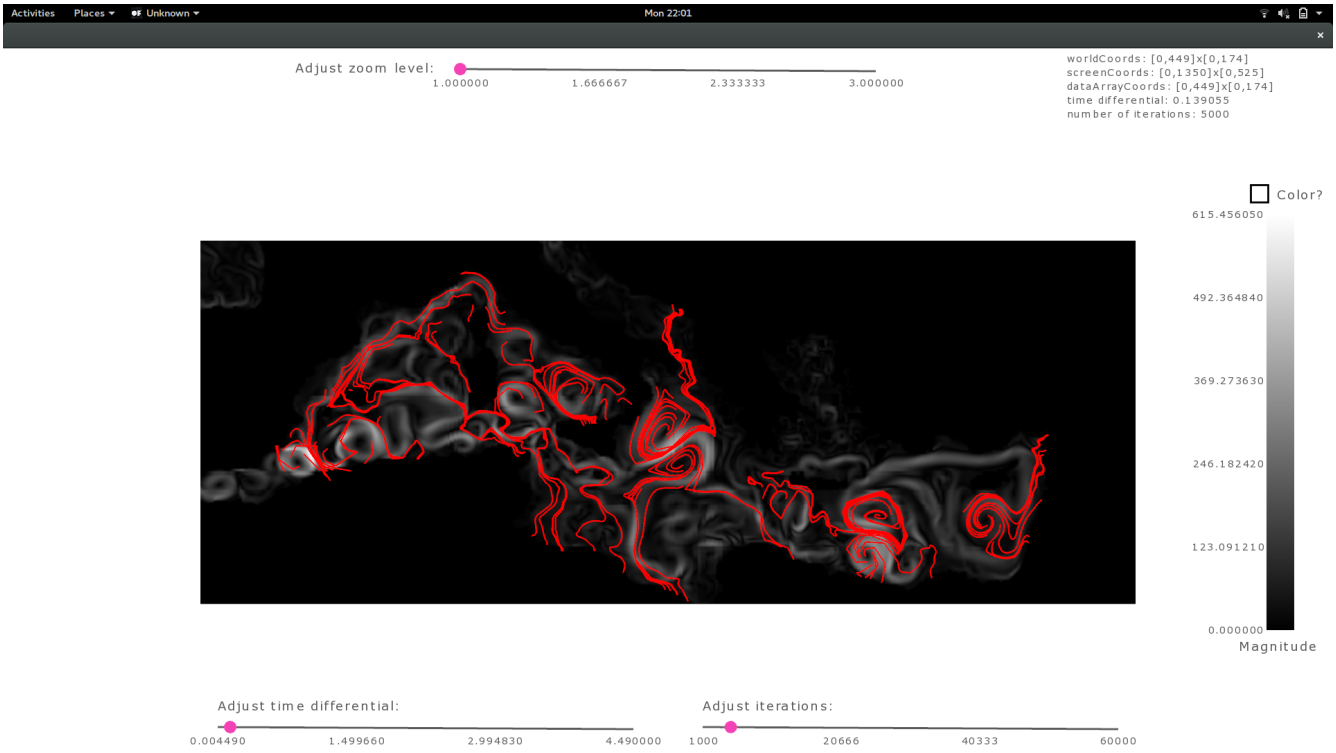

Activities
Places
Unknown
Mon 14:46
Adjust zoom level:
1.000000
1.666667
2.333333
3.000000
worldCoords: [0,20]x[-5,5]
screenCoords: [0,800]x[0,400]
dataArrayCoords: [0,799]x[0,399]
Color?
0.020273
0.016222
0.012171
0.008120
0.004069
0.000018
Magnitude

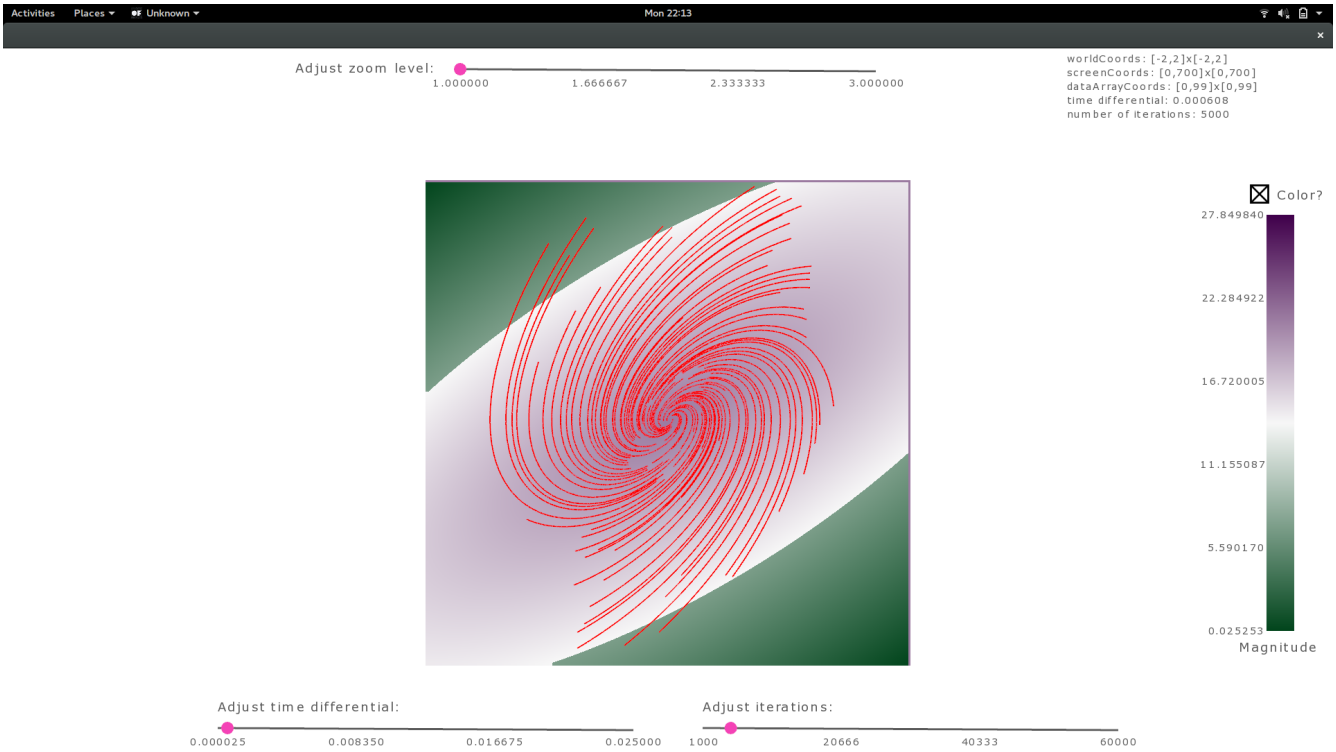
With smaller time differential, where streamlines are smooth.



With larger time differential, where streamlines are not as smooth as before




Activities
Places
Unknown
Mon 22:01
Adjust zoom level:
1.000000
1.666667
2.333333
3.000000
worldCoords: [0,449]x[0,174]
screenCoords: [0,1350]x[0,525]
dataArrayCoords: [0,449]x[0,174]
time differential: 0.139055
number of iterations: 5000
Color?
615.456050
492.364840
369.273630
246.182420
123.091210
0.000000
Magnitude
Adjust time differential:
0.004490
1.499660
2.994830
4.490000
Adjust iterations:
1000
20666
40333
60000


Activities
Places
Unknown
Mon 22:13
Adjust zoom level:
1.000000
1.666667
2.333333
3.000000
worldCoords: [-2,2]x[-2,2]
screenCoords: [0,700]x[0,700]
dataArrayCoords: [0,99]x[0,99]
time differential: 0.000608
number of iterations: 5000
Color?
27.849840
22.284922
16.720005
11.155087
5.590170
0.025253
Magnitude
Adjust time differential:
0.000025
0.008350
0.016675
0.025000
Adjust iterations:
1000
20666
40333
60000