GaTech CS-6491 Fall 2019 Project 3
Vector Field

Rukai Zhao

Objective:

In this project, I have implemented a field-aligned triangle-mesh T, traces and samples and based on the newly generated traces, I have created a new triangle mesh using constrained Delaunay Triangulation. I have also created an example in the form of a bridge.

Vector-Field:

In this part, I have created a new member variable called boolean[] constrained inside the Mesh class. This array keeps track of each vertex whether it is constrained or not. For simplicity, I denoted four constrained vertex.

When solving for the unconstrained vectors, I used tuck-untuck span to do the computation. The computation was done inside functions: tuck_untuck_span(),vec find_vec(pt vertex, vec vec_v) inside the mesh2D file. For each triangle, compute the average of the vectors of its vertices and for each vertex, compute the average of the vectors of its incident triangles:

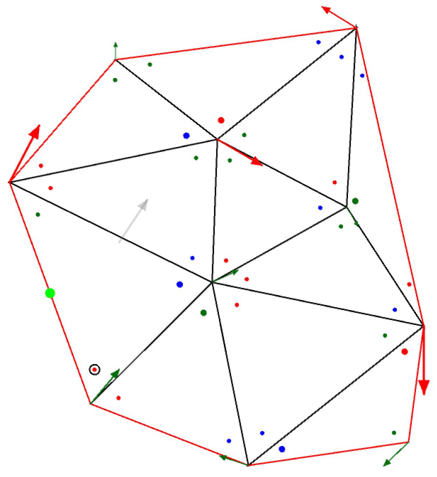
Initialize unconstrained vectors to 0

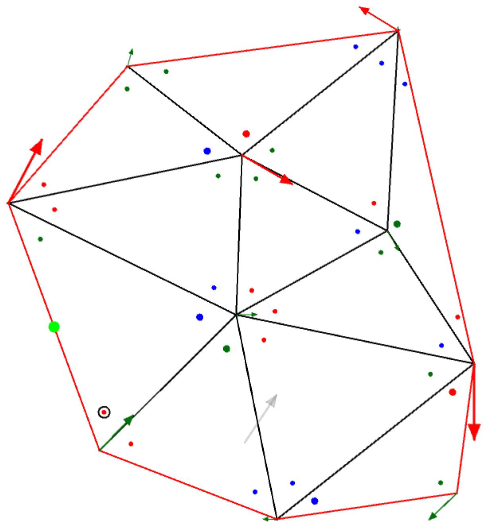
For several interations:

turk :LERP towards the average of neighbours

Snap: preserve constraint vectors

I did a LERP version using LerpAverageOfArrows and a log-spiral averaging version using SteadyAverageOfArrows:

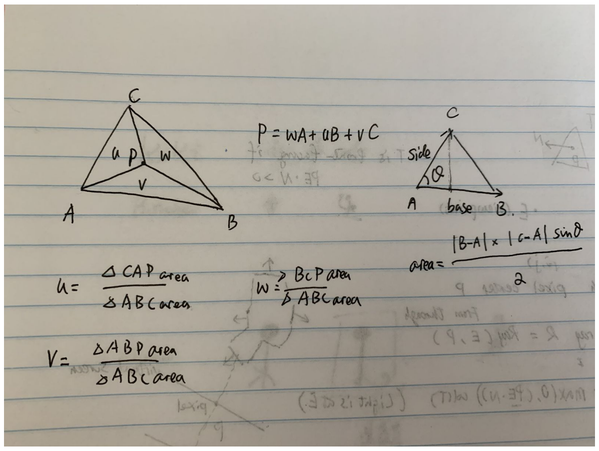

Log spiral


Lerp

Field-aligned triangle-mesh:

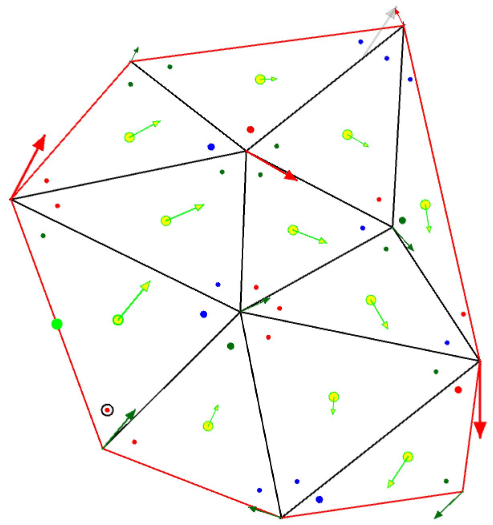
In this part, since I want the average vectors of the three vertices in a triangle, I used barycentric coordinates to compute the vector inside a triangle. Log-spiral won’t work here since the field inside the triangle is not a spiral vector field. Every point can be denoted as P=wA+uB+vC given three vertices A, B, C.

Computation of w,u,v:

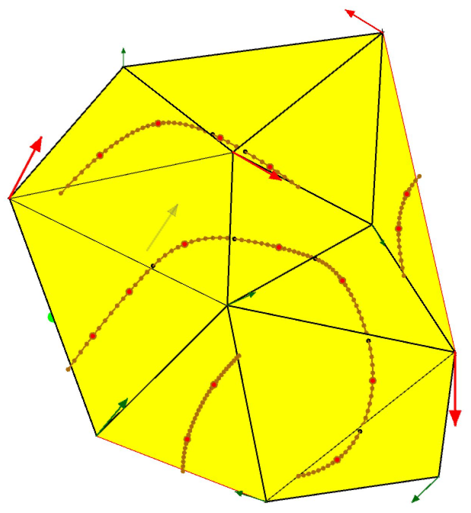

C
u P w
v
A
B
P = wA+ uB + vC
C
Side
θ
A
base
B.
area = |B-A| × |C-A| sinθ / 2
u = △CAP area / △ABC area
w = △BCP area / △ABC area
v = △ABP area / △ABC area

Since we get u,w,v, we can easily compute vector V as V= wA_v + uB_v +vC_v where A_v, B_v and C_v to be the vectors of vertices A,B,C. This part was done in vec_in_tri function inside mesh2D file.

Visualize the field by showing the vector at the centroid of each triangle.

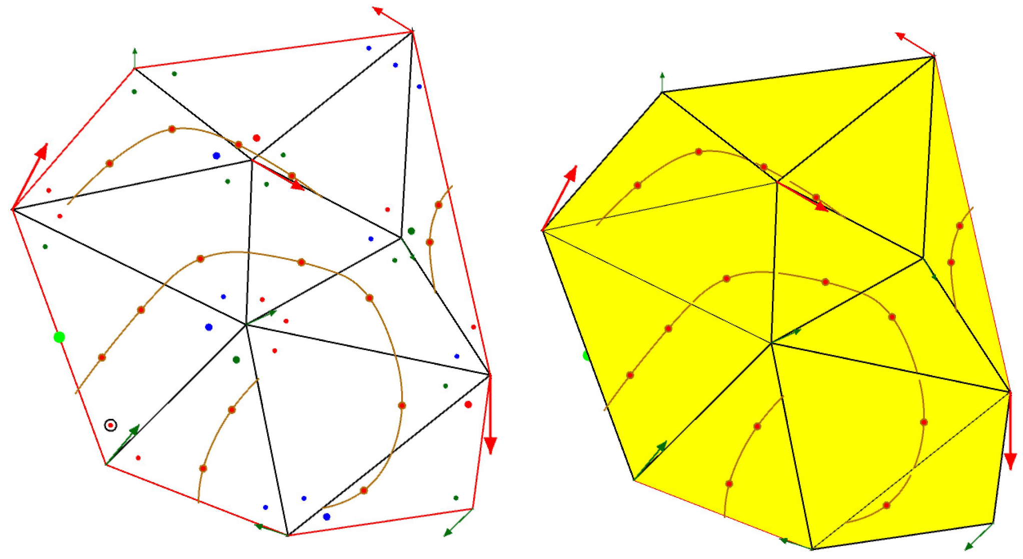


For generating the trace, I used P = P+sV(P) inside the naive_trace function in mesh2D. I also used P= P+sV(P).inverse, since I also want to trace in the opposite direction. I denoted each corner as visited or not visited since I want to know whether the triangle is visited or not. For each triangle, if the triangle is not visited, I'll trace from the centroid of the triangle until the boundary is met or the iteration is stoped.



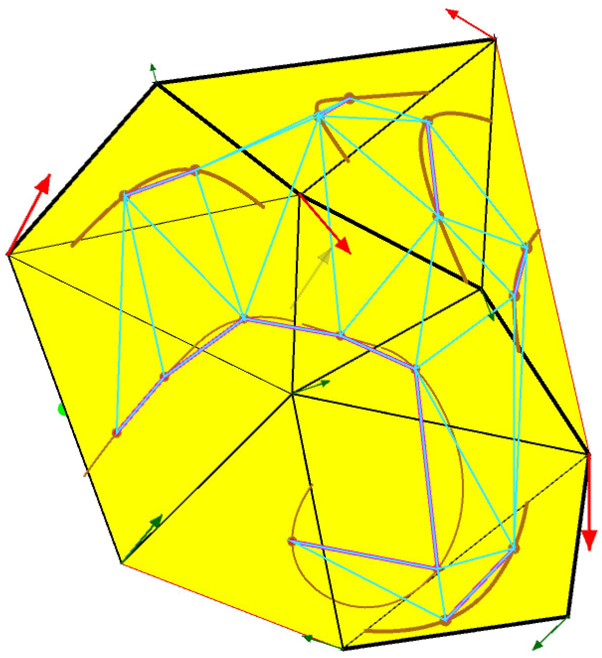
Constrained Delaunay Triangulate:

For selecting the samples, I select the points half way between the cross points (where the trace hit the triangle edges) and between the starting point(centroid) and the cross point.

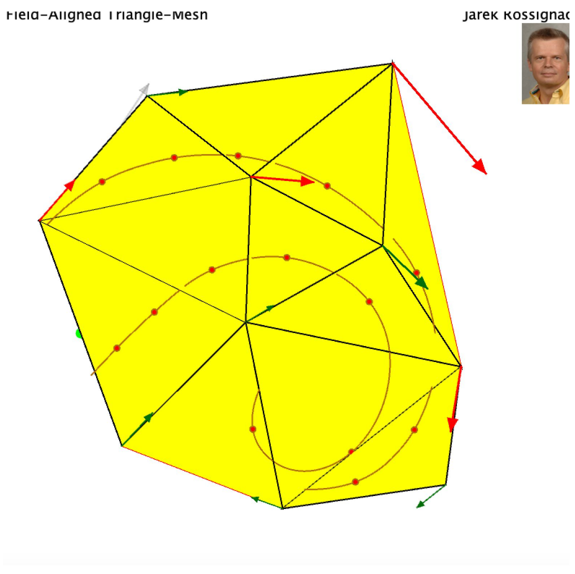


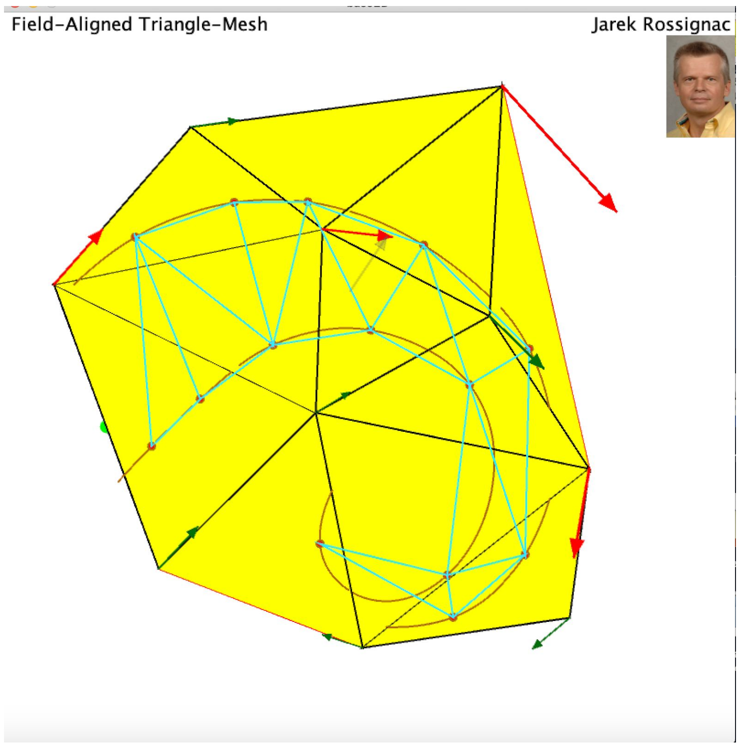
The point is colored as red points on the traces. After finding the samples, I use delaunay triangulation to generate the new mesh. Inside the triangulation, I made a condition as no three vertices of a triangle should on the same trace. I also made a condition that delete the triangle which has an angle between 175-180 degree.

Edge on the traces:(colored as thich magenta)



I also made a form of bridge by adjusting the constrianed vector field:


Field-Aligned Triangle-Mesh
Jarek Rossignac


Field-Aligned Triangle-Mesh
Jarek Rossignac