Week 2 Report (6/16/16)

Pseudocode

Mesh generation is started by creating a quad-grid with user-defined input of columns and rows. From there we save all the points in a list and begin triangulation. Triangles should be consistent throughout the graph. There are two types of triangles: clockwise and counter-clockwise. The clockwise triangle is generated by vertices {*i, i+1, i+(Nx+1)}* for any *i*: *0 ≤ i ≥ Ny-1.* The clockwise triangle is generated by vertices {*i, i+(NX+1), i+(Nx)}.* After triangulation we can start storing all the triangles in a list which gives each triangle an index to retrieve from. Given the declaration of triangles and their id’s, we can then allow for users to add and edit singularities onto the mesh.

//Goal: To create a vector field from user-defined input regarding mesh range and singularities.

Mesh graph using paintComponent from java.swing

class Mesh{

List<Point> pointlist;

List<Triangle> trianglelist;

int NX, NY; //user input. User-defined set of points (Nx\*Ny sample points)

x\_range, y\_range = getMax() – getMin();

x\_int = x\_range/(Nx-1) //the distance between two points.

y\_int = y\_range/(Ny-1)

//Generate points with loop

for(int j = 0; j ≤ getMax() ; j+= Ny)

for(int i = 0; i ≤ getMax() ; i+= Nx){

pointlist.add(new Point(i, j));

}

//Generate Triangles

for(int n =0; n ≤ pointlist.length()-1; n++){

Point temp = pointlist[n];

If(temp.x != Nx && temp.y != Ny){ //corner points do not form triangles

//clockwise

Point B = pointlist[n+1];

Point C = pointlist[n+(Nx+1)];

new Triangle(0, temp, B, C );

trianglelist.add();

//counter-clockwise

Point B = pointlist[n+Nx];

Point C = pointlist[n+(Nx+1)];

new Triangle(1, temp, B, C );

trianglelist.add();

}}}

}

class Point{

float x,y;

float vx, vy;

Point (int x, int y){} //constructor

}

Class Triangle{

Point a, b, c;

Int type; //0-clockwise, 1-counter-clockwise

Triangle(int type, Point a, Point b, Point c){} //constructor

}

class SingularElement{

int ID; // numbering each element

int Triangle\_ID; //the triangle that contains the singular element

double centerx, centery; //center under global frame

int type; //singular type (0 –Null, 1—source, 2—sink, 3—saddle, 4—cwcenter,

//Matrix can be represented with 2d/3d arrays as long there are math operations.

Matrix transform\_matrix; //transformation matrix for user editing

Matrix Jacobian; //Jacobian matrix is full rank. Used to calculate the vector divergence and curl (attract,repel,saddle).

double s; //size, scale, geometric representation

bool deleted;

}

Conclusion

Storing the points and triangles of the mesh in a list is advantageous over a 2d array because of memory accessing since the connecting vertices are already determined and calculated with the data types. I will begin creating the model implementation using java swing interface to represent the mesh generation from user input. The model will show each point with a representative ray. I will make sure to keep track of the different complexities while working on this implementation and for the future.