CS34515 Fall 2014: Project 4

Project 4: XOR of Triangles
Students:

Bo Chen

Shen Yang

Project description
Drawing two triangles with different color, if any one of triangles is counter-clockwise, fill it in light gray. At each frame, fine the intersections of two triangles, show them as black dots and store their location into an array of point. Also at each frame compute a GEL representation of polyloops that form the boundary of the connected components of the interior of XOR, [ABC] XOR [CDE]. At the end, compute the area of each connected component and fill the component with a matching color.
Motivation:
In the computer graphic, XOR-based drawing methods are often used to manage such items as bounding boxes and cursors on systems without alpha channels or overlay planes. Such as when we create the image or the animation, we can see the shadow or the shape of the under layer object. XOR is sometimes used as a simple mixing function in cryptography, for example, with one-time pad or Feistel network systems.

GEL Description:
In this project, we start to study about the GEL representation algorithm. I will explain how to implement the GEL in our algorithm. As we known:
G: representing vertices as an array G[] of points, we created a array list in the class to implement G.

```
G = new ArrayList<pt>();
```

E: represents the oriented edges as two arrays V[] and N[].V[e] is denoted e.v and represents the integer index to the starting vertex of edge. N[e], denoted e.n represents the next edge along the polyloop.

```
ArrayList<Integer> V = new ArrayList<Integer>();
ArrayList<Integer> N = new ArrayList<Integer>();
```

The V[] contains the intersections and vertex of two triangles, also as the beginning of edge. And the parallel index of N[] contains the index of second vertex of edges.
So for example, we have V[]N[] as:

	0	1	2	3	4	5	6	7	8	9
V[]:	0	1	2	3	4	5	6	6	7	7
N[]:	1	2	8	7	9	3	0	4	6	5

Assume we want to find a edge index 2 of VN, then the V[2] is the first vertex of edge, N[2] point to the index of second in V[], N[2] = 8, so it should be v[8] = 7. The answer is edge(2, 7)
L: represents the loops for each component by storing, for each component: the number of loops and, for each loop: the index of one of its edges. Usually we decided to store the first edge of this loop.

CS34515 Fall 2014: Project 4

Finding intersections:

1. Create two E, one contains all the default edges to create two for loops as E1, another is real time E, keep storing edges of each frame as E.
2. Using two for loops, go through all the edge to check if any two edge cross, if they cross then find the intersection of these two edges:
   a. Define two edge cross by using det(edge1, edge2);

```
return (det(AB, AD) > 0 != det(AB, AE) > 0) && (det(DE, DA) >0 != det(DE, DB) > 0);
```

   b. Finding the intersection by using the formula

Point of collision expressed by: $P(t) = C + t\overrightarrow{CD}$

We know at the point of collision: $\overrightarrow{AP} \cdot \underline{\overrightarrow{AB}^{\circ}} = 0$

Substitute:

$(C + t\overrightarrow{CD} - A) \cdot \underline{\overrightarrow{AB}^{\circ}} = 0$

Distribute (dot-products can do that because they're linear =D), solve for t:

$$t = \frac{-\overrightarrow{AC} \cdot \overrightarrow{AB}^{\circ}}{\overrightarrow{CD} \cdot \overrightarrow{AB}^{\circ}}$$

3. Each time we get a intersection, store it into the G
4. Using the insert method Prof. Jarek had provided in the class to update the E every time we have new intersection.

```
insert(v, e){
  V[ne] = v;
  N[ne] = e.n;
  N[e] = ne;
  ne++;
}
```

Computing Components

After we got the complete E, we can find the components.

1. Start from the first element in the E(V[0]), keep finding the next edge vertex(V[N[e]]), if there are multiples next edge, comparing them by using the det(edge1, edge2) function, choose the closest right edge.
2. Delete the edge we have went through, until the loop comes to the start point. Then add the edge to the L, keep repeating this process until E is empty.
3. **L** will contains the complete loop information, by using L, we can find the complete component in the graph

Coloring the components

By using data of L, we easy to find out the component, so following the edge from the loop, draw out each edge we can get the shape of the component. By the order of edge number, we change the color of area.

Project Status

As we submitted the project, we have implement the most of algorithms, we can get a complete and correct data for the GE, the function to find the loop is not complete yet. But we found the way to find the loop, just didn't have time to implement the function, we think once done the loop function, it is easy coloring the completed components. In this report, there is not too much space to show the code in our project, we don't push too much coding work here.