Geometría Computacional
Proyecto Final

Profesor:
Adriana Ramírez Vigueras

Ayudante:
José Emiliano Cabrera Blancas

12 de mayo de 2014

1. Objetivos:

- Que el alumno implemente el algoritmo incremental para construir el diagrama de *Voronoi*.
- Que el alumno dibuje en *Ruby-Processing* los pasos que realizo el algoritmo incremental para construir un conjunto aleatorio de puntos dados.
- Que el alumno utilize la *DCEL* que representa el diagrama de *Voronoi* para dibujar la triangulación de *Delauney*.

2. Descripción

Aunque el algoritmo incremental para contruir el diagrama de *voronoi* es sencillo en su descripción, la dificultad de este radica en mantener bien la estructura que se utilize para representarla.
Como ya les habia mencionado, reutilizaremos la *DCEL* para representar el diagrama de *voronoi*. La estructura que representa le diagrama de *voronoi* no solo contiene un apuntador hacia la *DCEL* que lo representa, si no también otros datos que nos ayudan a procesar el algoritmo.
Por otro parte, tambíen se les proporcionan las funciones básicas para calcular la bisectriz perpendicular de dos puntos y posibles intersecciones con aristas.
La descripción de cada función y estructura que se agregaron para que ustedes puedan implementar el algoritmo vienen en los siguientes archivos del código fuente:

Archivos fuente

```
src/algorithms/algorithms.h
src/algorithms/algorithms.c
```

```
src/dcel/dcel.h
src/dcel/dcel.c

src/equations/equations.h
src/equations/equations.c

src/face/face.h
src/face/face.c

src/gui/voronoi.rb

src/main.c

src/points/2d_points.h
src/points/2d_points.c

src/shared_libraries/algorithms.rb
src/shared_libraries/voronoi.rb

src/tests/voronoi_tests.c
src/tests/voronoi_tests.h
src/tests/voronoi_tests.rb

src/voronoi/voronoi.h
src/voronoi/voronoi.c
```

3. *Tests*

Su código debe de pasar los siguientes *tests*:

Tests

```
algorithms.h: Voronoi de solo 2 puntos                          [FAIL]
algorithms.h: Voronoi de 3 puntos del caso degenerado           [FAIL]
algorithms.h: Voronoi de 7 puntos aleatorios                    [FAIL]
```

El primer *test* verifica que la función *voronoi_incremental()* genere un diagrama de *voronoi* correcto para dos puntos.
El segundo *test* verifica que la función *voronoi_incremental()* genere un diagrama de *voronoi* correcto para el caso degenerado de solo 3 puntos. El último *test* verifica que la respuesta que genera la función *voronoi_incremental()* sea correcta para cualquier conjunto de siente puntos generados de forma aleatoria.

4. ¿Qué debes de programar y dónde comentar tus cambios?

Básicamente debes de completar las funciones que se describen en el archivo *src/algorithms/algorithms.h* y el archivo *gui/voronoi.rb* que sirven para dibujar la *DCEL* que genera el diagrama de *voronoi* y la triangulación de *delauney*.
Para dibujar el digrama de *voronoi* debes de leer el archivo que genera la función *process_voronoi()*, dicho archivo es *src/voronoi.txt* y para dibujar la triangulación debes de leer el archivo que genera la función *process_delauney*,
El formato que tiene el archivo *voronoi.txt* es el siguiente:

voronoi.txt

```
Semillas: 0
Aristas: 8
400.000000 0.000000 0.000000 0.000000
0.000000 0.000000 400.000000 0.000000
0.000000 0.000000 0.000000 400.000000
0.000000 400.000000 0.000000 0.000000
0.000000 400.000000 400.000000 400.000000
400.000000 400.000000 0.000000 400.000000
400.000000 400.000000 400.000000 0.000000
400.000000 0.000000 400.000000 400.000000
Semillas: 1
65.000000 5.000000 Normal
Aristas: 8
400.000000 0.000000 0.000000 0.000000
0.000000 0.000000 400.000000 0.000000
0.000000 0.000000 0.000000 400.000000
0.000000 400.000000 0.000000 0.000000
0.000000 400.000000 400.000000 400.000000
400.000000 400.000000 0.000000 400.000000
400.000000 400.000000 400.000000 0.000000
400.000000 0.000000 400.000000 400.000000
```

Donde *Semillas: N* nos indica cuantos vertices semilla se deben de dibujar y *Aristas: M* indica que aristas debemos dibujar hasta ese momento.
Tú puedes elegir que formato utilizar para guardar el archivo *delauney.txt*.

Por último, como ya se les habia dicho en clase, el archivo *README.txt* debe contener tres pseudocodigo, ya sea descritos solo con palabras o alguna notación fácil de leer. El primer pseudocodigo debe describir los pasos que hicieron para completar la función *merge_faces()*,el segundo pseudocodigo debe describir los pasos que realiza *voronoi_incremental()* y el tercer pseudocodigo debe contener los pasos de la función *upgrade_voronoi_diagram()* para actualizar la *DCEL*.

Estos pseudocodigos son requisitos indispensables para calificar su proyecto final.

Entrega 6 de junio de 2014