Delaunay Triangulation Project

GitHub Repository

Francesca Cella, 60/73/65172

The approach choosen for this implementation takes adavantage of indices and direct access: all the lists of elements used in this project are implemented using vectors and, when possible, arrays; the consequence is that most of the members for implemented classes are integer indices. In some cases references are used, for example in drawables and in function inputs: the project is structured in order to exploit the utility of references and to avoid errors when a data structure, like a vector (that can be reallocated), change its memory position. There isn't any dynamic object and there isn't any usage of pointers. The edge legalization uses a recursive approach while the search into the DAG follows an iterative approach.

Each Node of the DAG is implemented using indices of the vector of nodes and index of the considered triangle; the same approach is used for storing the adjacencies: each cell of the vector represents a triangle with the same index, each element of the array (that is the cell of the vector) is the index of the adjacent triangle in the triangulation. The incremental step and the edge legalization are implemented separately from data structures, they take in input the triangulation and the dag and use their methods for computing the triangulation.

The incremental step of the triangulation manages only the insertion of a point inside a triangle, by creating 3 triangles following a particular pattern for adjacencies and vertices. After each triangle is created, for each of them there is the edge legalization: in the edge legalization all the possibile cases are managed, according to the adjacencies of the considered triangle and its adjacent.

The drawable triangulation is always active in the scene during the execution, so the methods to draw and to erase the triangulation are not implemented. Since the drawable object takes the reference of the triangulation, when a point is inserted the canvas and the drawn triangulation is updated.

Classes that represent data structures are stored under the *data structure* folder, while classes that represent drawable objects are stored under the *drawables* folder. Functions that implement the algorithm are stored under the *algorithms* folder.