Project goal and requirements: Create a program to draw and color planar triangulated graphs, allowing dynamic vertex addition and validation.
The program shall have a graphical interface where a set of buttons will allow a set of commands:
The program will normally start by drawing a triangle with vertices V1, V2 and V3.
To that primitive graph we will be able to add new vertices. Each time a new vertex is added, it is added at the periphery of the graph touching **m>1** contiguous vertices of that periphery. Those vertices are delimitated by two extreme vertices Vp and Vq. In order to define all the vertices to be touched by the new added one we specify Vp and Vq with the mouse. The new vertex will touch those extremes and all the vertices in between them (clockwise sense). The program will check if the selected vertices are ok. Depending on the number of vertices that this new vertex touches, its location will be one or the other. The edges may be curved to improve the compaction of the graph.
A graph shall be represented internally by a list of vertices and a set of edges that represents the connections between them. Also, we need to represent the periphery that is a subgraph that dynamically changes with the adding of a new vertex. Every vertex will have an associated location (X, Y) in the plane. This location is used to implement some commands.
Every vertex shall be represented by a circle in color 1 and filled with a color 2. Colors can change to represent different aspects of the graph.

The graph must allow at least 10,000 vertices and shall check every command and color to find its validity.

COMMANDS:

S: Start basic graph (triangle V1-V2-V3).

R: Add random vertex to graph. Two random vertices are selected in the actual graph periphery and a new vertex is added to the graph touching them and all the vertices between them in a clockwise way.

A: Add a new vertex to the periphery. The program will ask the user to select Vp and then Vq before adding the new vertex.

Gm: Go to vertex m. The graph hide vertices with index over m and all associated edges.

Z+: Zooms + To be done with the mouse

Z-: Zooms –

C: Center graph in screen and resizes it

Left mouse + move to view a specific part of the graph

Right mouse +

T: Toggles between color or index for every vertex.

Recommendations:

1.- Anytime a vertex is added try to maintain the graph contour convex and edges with a similar distance to the previous periphery.
2.- To add a vertex to the graph periphery, find the vertex in the periphery in the middle of the set of vertices to be connected to it and start connecting from that vertex up to the rest of the other vertices in that set, connecting first the closer vertices and last the ones that are father away. Try to maintain distances more or less constant for every added vertex. If the angle left for a vertex gets close to 60º make a redraw to share the angle between all vertices o allow a new one to be added.
3.- Use variable balls with a certain diameter to represent vertices. When the number to assign to the vertex gets bigger the diameter of the ball will be increased to enclose that bigger number. Then make a redraw to get more space in general for big numbers.
4.- Because the only elements to represent are balls and pipes as edges, if you use nice format to represent them the result will be very good.
5.- If you represent internally the graph as a matrix, try to represent also its peripheries by means of this matrix.
6.- Represent colors with natural numbers 1 to 4. Then with a user setup palette link a color to every number.

A "Redraw" command is recommended to take the distances and positions on the last periphery of the graph and will draw it considering them. Angles can be divided in equal parts incrementing the lengths of the edges and a kind of edge length can be use to homogenize al edges. This redraw command will be applied after every addition of a new vertex to the graph until all vertices have been added. Then a go to vertex 4 will be applied and the graph drawing can be restarted knowing the future of the graph.