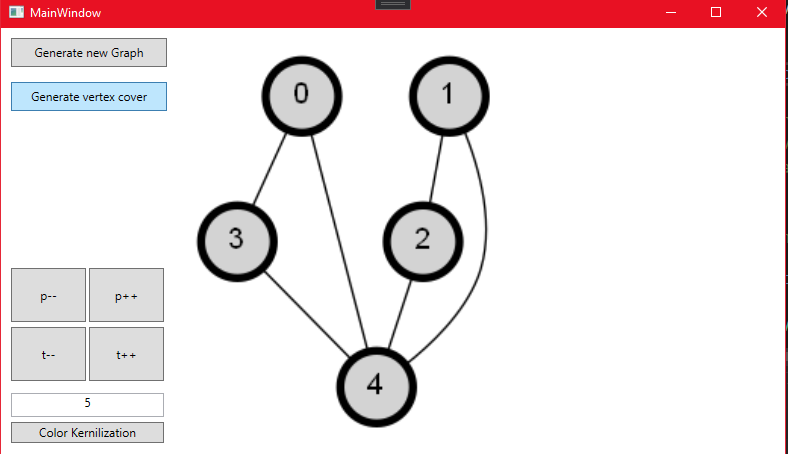
Week 3

# UI with pendent and top buttons

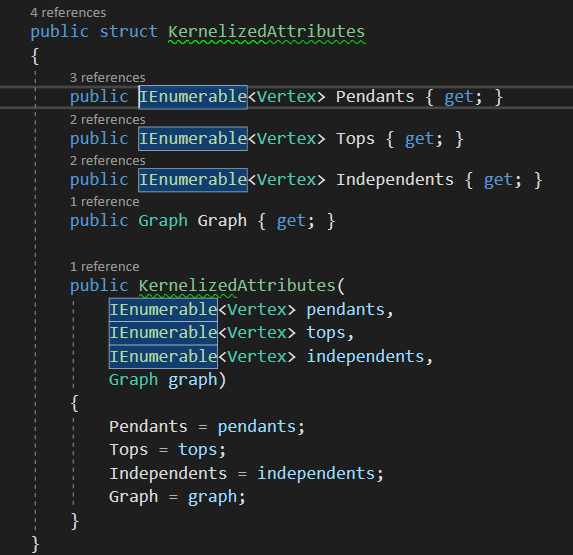
The first thing that we added was the new buttons that where required for the



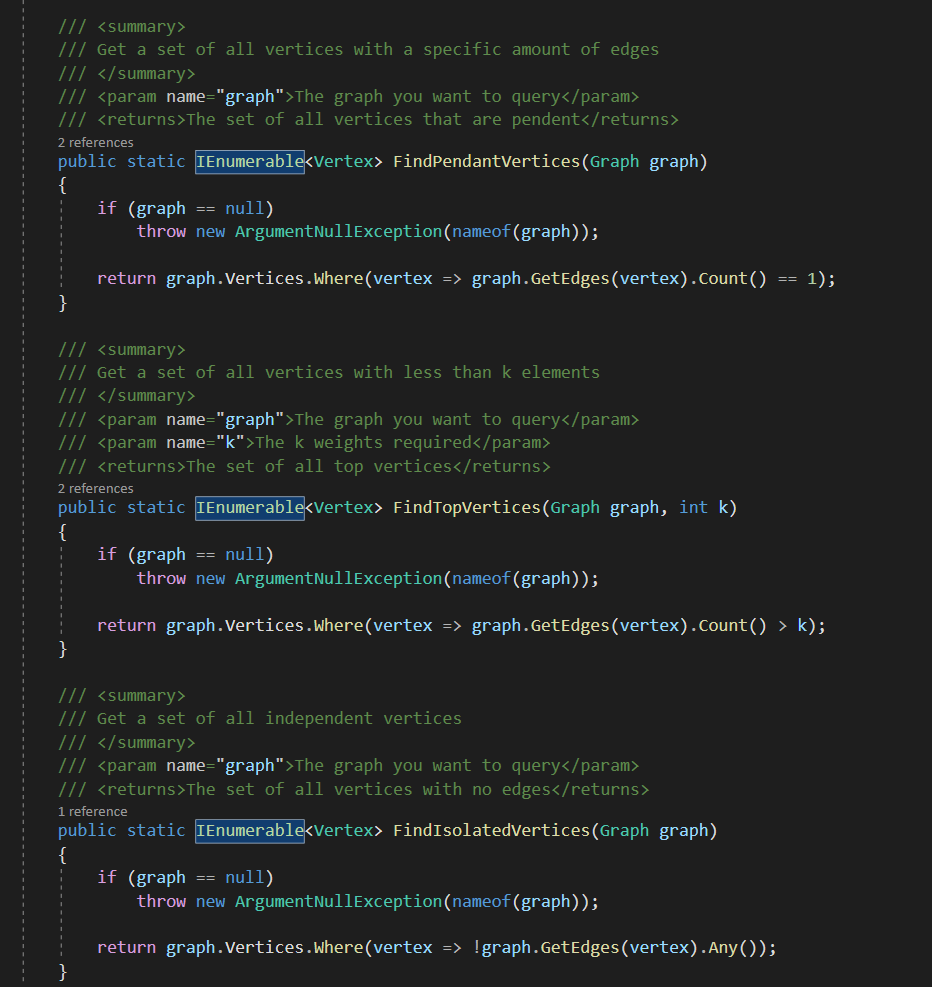
We also added a separate button so that we can color the graph with the kernelization we perform.

# Finding Pendant and Top vertices

We created a utility that finds the pendant, top and independent vertices. For ease of use we also decided to create a basic data structure. The data structure holds all pendent, top and independent vertices:

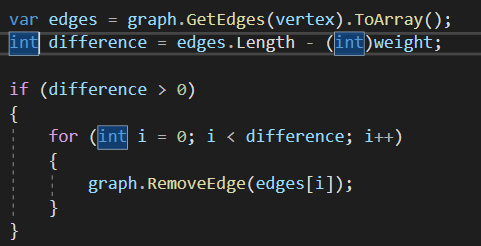


To find the values to put into this datatype we used these methods:



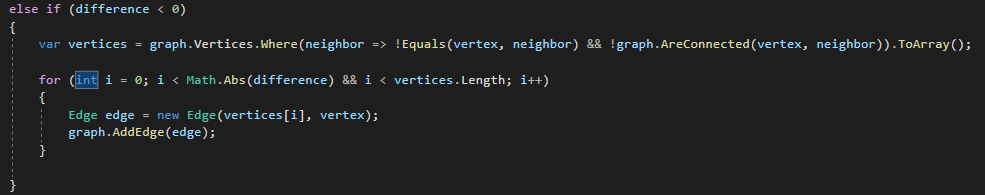
# Updating the amount of edges

We created a method that removes or add edges so that a specific vertex will be of that degree. The first thing we do in this method is determine the distance from the requested degree this vertex has. We then have two paths. The first path is the for positive values. This means that we need to delete edges to get to the request vertex degree. That looks like this:



For the part where we have to add vertices that is the negative difference. We had to create a more complex query. This query gets all vertices that are not adjacent to the vertex we chance the edges for. We also did not want vertices that connect to themselves so we also excluded that vertex from the queried data.

We then add all the edges that we can so that we get the request vertex degree in the graph.

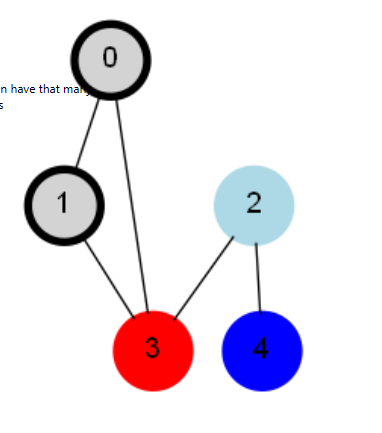


Of course when the difference is 0 we do nothing.

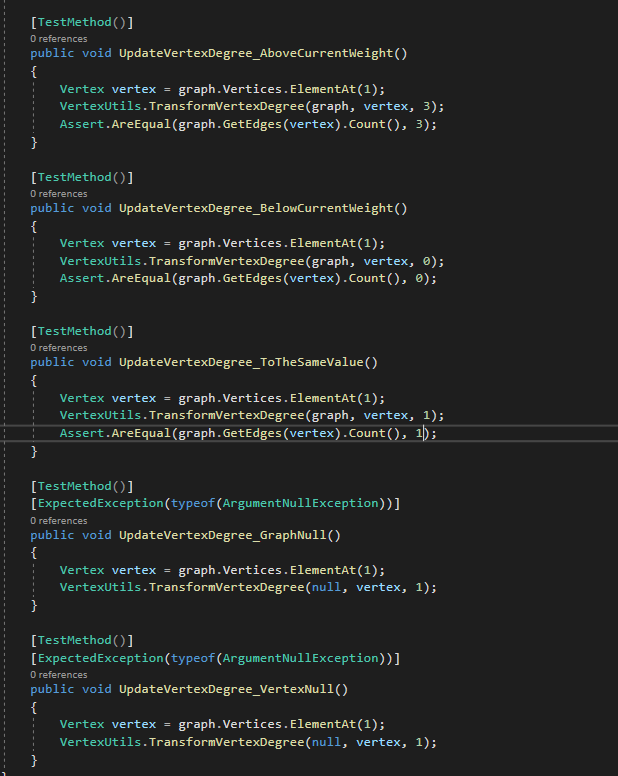
# Coloring the Kernilization

For the coloring of the kernelization we decided on a few basic colors to represent the kernelization. We decided to color the top vertices red, the pendent vertices blue, the neigbouring vertices of the pendent vertices light blue, The independent vertex are colored green.

This is the end result after coloring in vertixes:



# Testing



# 