**Software Design Description**

**January 14, 2016**

**Graph maker**

**Oleksiy Tsubera**

1. **Data design**
2. **Architecture design**

The application will consist of the next modules:

* 1. **Graph**. This module represents a structure of a graph. It provides an ability to change its structure by adding or deleting elements to the graph. It can convert graph structure to/from JSON format to provide saving/opening the graph. Graph module also provides finding a subgraph that includes all vertices of the graph (aka a spanning tree).

Methods:

* void addVertex(QString name, QColor color, double positionX, double positionY) – this function adds a new vertex to the graph.
* void removeVertex(Vertex &vertex) – this function removes the vertex from the graph by its reference.
* void addLine(int weight, QColor color, Vertex &vertex1, Vertex &vertex2) – this function adds new line that connects vertex1 and vertex2.
* void removeLine(Vertex &vertex1, Vertex &vertex2) – this function removes the line that connects vertex1 and vertex2.
* QString toJson() – this function converts graph structure to JSON format and returns string as result.
* void fromJson(QString jsonGraph) – this function creates a graph from a JSON format.
* Graph getSpanningTree() – this function finds a spanning tree in the graph and returns a nested graph.

Members:

* list<Vertex> vertices – list of vertices that the graph consists of.
* list<Line> lines – list of lines that the graph consists of.
  1. **Vertex**. This module represents a vertex of a graph. According to data provided by object of this type an appropriate vertex will be painted on a canvas.

Methods:

* QString getName() – this function returns a name of a vertex.
* void setName() – this function sets a name of a vertex.
* QColor getColor() – this function returns a color of a vertex.
* void setColor(QColor color) – this function sets new color of a vertex.
* Position getPosition() – this function gets position of a vertex.
* void setPosition(Position newPosition) – this function sets new position of a vertex.

Members:

* QString name – name of a vertex.
* QColor color – color of a vertex.
* Position position – position of a vertex on a canvas.
  1. **Position**. This class represents coordinates of an object on a canvas.

Methods:

* double getX() – returns X-coordinate of a position.
* void setX(double value) – sets new value of X-coordinate.
* double getY() – returns Y-coordinate of a position.
* void setY(double value) – sets new value of Y-coordinate.

Members:

* double x – X-coordinate of a position.
* double y – Y-coordinate of a position.
  1. **Line**. This class represents an edge of a graph. It connects two different vertices of a graph.

Methods:

* int getWeight() – returns a weight of an edge.
* void setWeight(int value) – sets new weight of an edge.
* QColor getColor() – returns color of a line.
* void setColor(QColor color) – sets new color of a line.
* const Vertex& getVertex1() – returns a reference to first vertex that a line connects to.
* void setVertex1(const Vertex &vertex) – sets new first vertex of a line.
* Const Vertex& getVertex2() – returns second vertex of a line.
* void setVertex2(const Vertex &vertex) – sets new second vertex of a line.

Members:

* int weight – weight of an edge of a graph.
* QColor color – color of a line.
* Vertex &vertex1 – reference to a first vertex that a line connects to.
* Vertex &vertex2 – reference to a second vertex that a line connects to.
  1. **Canvas**. This module represents a graph drawer that will draw a graph to the screen. It will provide ability to draw separate graph’s elements such as vertices and lines. Also it allows highlight a part of a graph or entire graph what means changing color of its lines and vertices.

Methods:

* void drawGraph(const Graph &graph) – this function paints the graph to the screen.
* void drawLine(const Line &line) – this function draws the line.
* void drawVertex(const Vertex &vertex) – this function draws the vertex.
* void highlight(const Graph &spanningTree) – this function highlights the graph by changing it color.
* void clear() – clears the canvas.
  1. **MainWindow**. This module represents a main window of the application. It will contain current graph and a canvas to draw. It will also provide ability to create new graph, save and open a graph.

Methods:

* void newGraph(quint16 vertexCount, quint16 lineCount) – this function allows to create a new graph with provided parameters.
* void saveGraphAsPicture(QString fileName) – this function saves current graph picture into external file.
* void saveGraphAsJson(QString fileName) – this function saves current graph structure into external JSON file.
* void openGraph(QString fileName) – this function opens saved graph from external JSON file and stores it to current graph.

Members:

* Canvas canvas – canvas to draw the graph.
* Graph graph – current graph.

1. **Interface design**

The application has graphic user interface that allows user to easy and fast build graphs. There is the main window that contains a canvas to draw a graph, menu where you can find all functionality you need to build graphs, toolbar where you can find most used tool to build graphs. There is the properties window where you can see and change properties of a graph’s elements. More details you can see on application mockups.