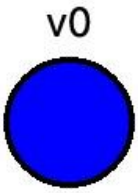
Sketchpad App User Manual

Nodes:

Creating Nodes: Nodes can be created in the *Controls* tab of the menu (1), with the *Add Node* button. By default, they will be blue, with the name “vx” where x is the lowest non-negative number available (not used by another node). They will also have no edges with other nodes on creation.

Editing Nodes: When selecting a node, you are able to make the following changes: *Change Name* (2.1), *Choose Color* (2.2), Adjust size(2.3), and *Delete* (2.4). Type in the textbox any label for the node, press the *Change Name* button to save the change. To choose a color, press the *Choose Color* button and a dialog for choosing a color will appear. Deleting a node will also delete all edges connected to this node.

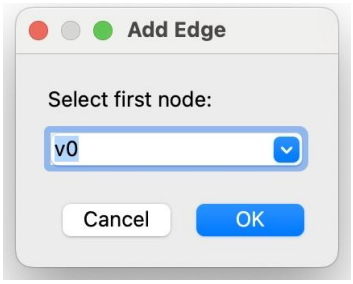


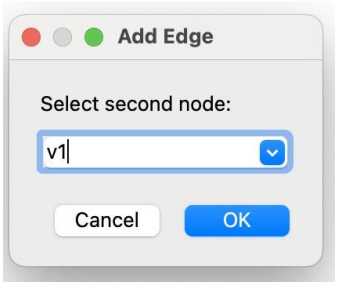
Node Selection: Nodes can be selected by double clicking them, or by selecting them in the drop down in the menu (3). You can unselect them by double clicking a selected node again, selecting a different node or edge, or by selecting “None” in the node drop down. On selection, they will have a red outline to signify they are the selected node.

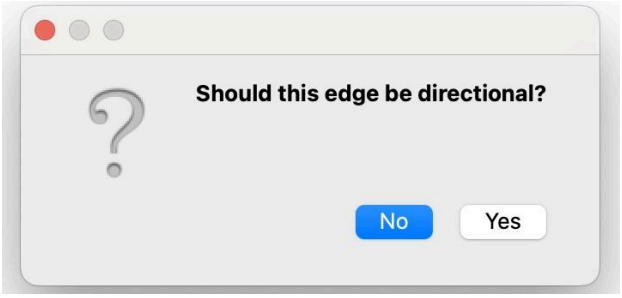
Moving Nodes: Nodes can be moved by clicking and dragging, and connected edges will update as you move the node.

Edges:

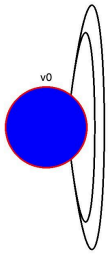
Creating Edges: Edges can be created in the *Controls* tab of the menu (1) using the *Add Edge* button. Upon clicking, you will be prompted to select two nodes to connect. Edges are by default black and named "edge x," where x is the lowest available non-negative number not used by another edge. You can specify if the edge is directional during creation.

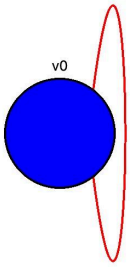


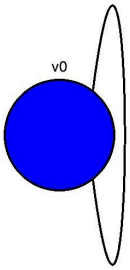


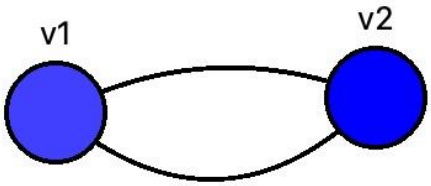


Editing Edges: When an edge is selected, you can perform the following actions: *Change Name* (2.1), *Delete* (2.4) and Toggle Directionality. These functions work similarly to described in the nodes section. Directionality is exclusive to edges and can be turned on or off. A directional edge has an arrow to show its direction while a non–directional edge does not have this. Edges between a single node are loops, they are shown on the right.







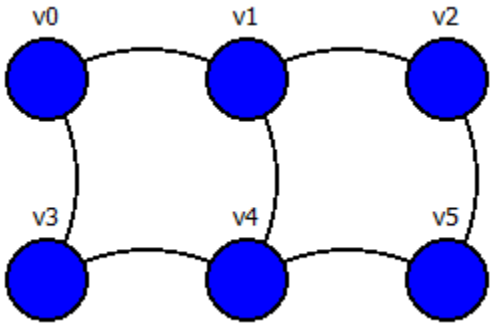


Edge Selection: Edges can be selected by double-clicking them on the graph or by choosing them from the edge drop-down menu in the controls (3). To unselect an edge, double-click it again, select another node or edge, or choose "None" from the edge drop-down. Selected edges are highlighted to indicate selection.

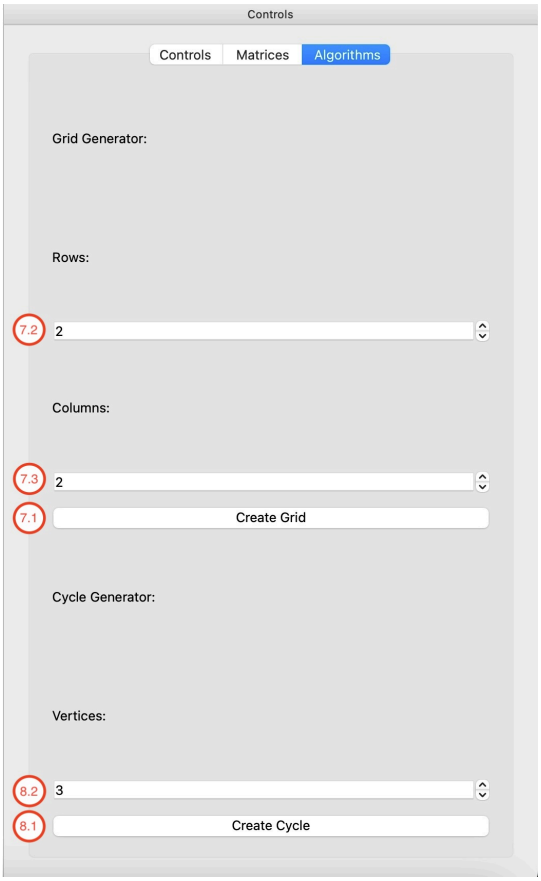
Algorithms:

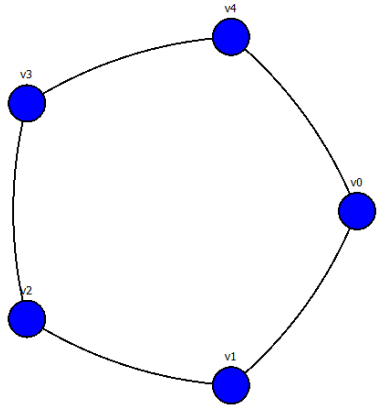
The algorithms tab includes functionality for algorithmic creation of “pretty” drawings of graphs, specifically cycles that actually look like cycles, and grids that look like grids.

Creating Grids: A grid can be automatically created using the Grid Generator section by entering the desired number of rows in the *Rows* textbox (7.2), and the desired number of columns in the *Columns* textbox (7.3), then clicking the *Create Grid* button (7.1). The grid will then be automatically drawn.



Creating Cycles: A cycle can be automatically created using the Cycle Generator section by entering the desired number of vertices to be included in the cycle in the *Vertices* textbox (8.2), then clicking the *Create Cycle* button (8.1). The cycle will then be automatically drawn.

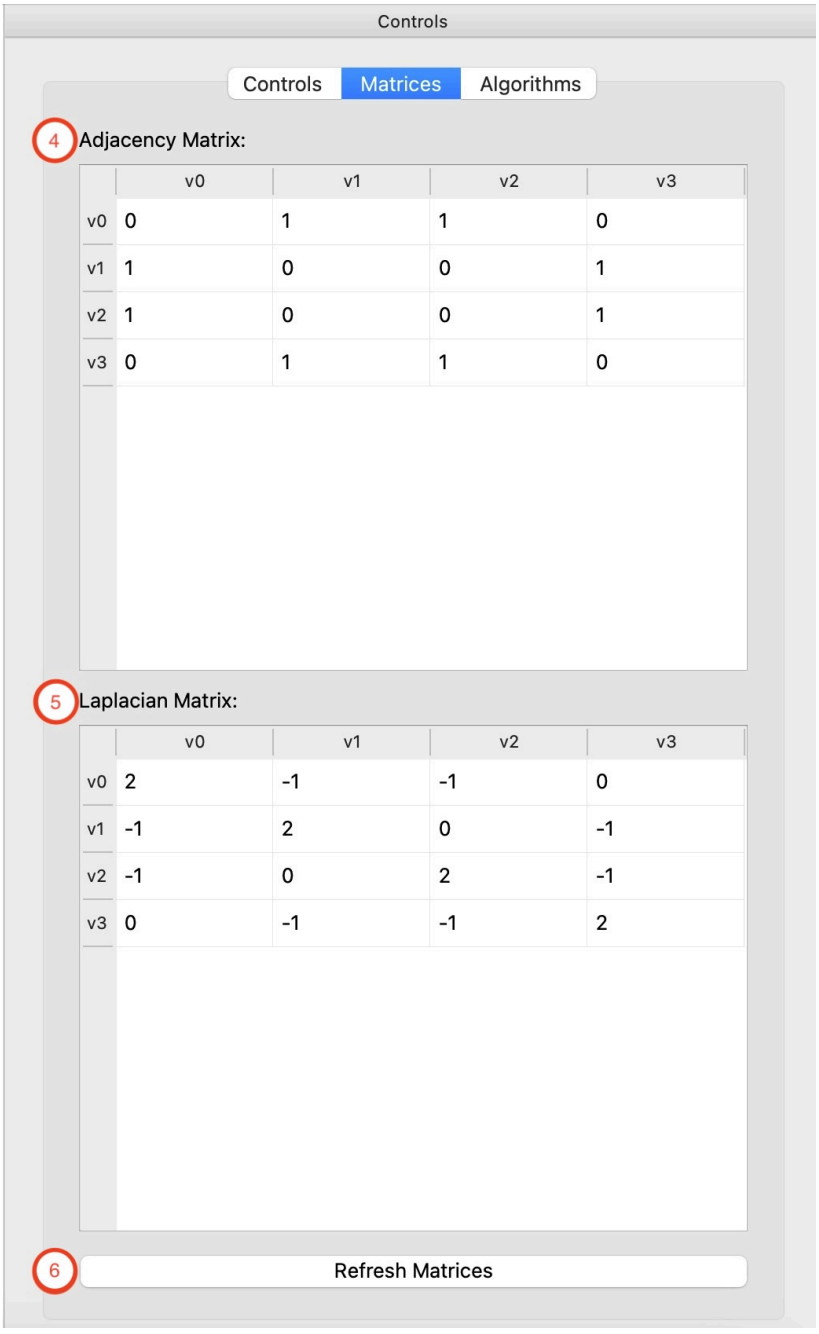




Matrices:

Viewing Matrices: The *Matrices* tab provides matrix representations of the entire graph.
Adjacency Matrix: The *Adjacency Matrix* (4) can be used to more easily and simply understand the connections between nodes and components, and to analyze paths in more complex graphs.
Laplacian Matrix: The *Laplacian Matrix* (5) can be used to gain a more comprehensive overview of the graph's structure, such as viewing the degree of all vertices in the graph.
Refreshing Matrices: While the matrices display will usually automatically refresh to accurately depict the contents of the graph, in the rare case that the graph is changed and the automatic refresh doesn't occur, the *Refresh Matrices* (6) button can be used to manually refresh.

The example matrices to the right reflect a simple 2x2 grid graph.



Implementation:

- We implemented this app in python, using the PyQt5 library's QtWidgets and QtGui modules for User interface components, and the QtCore module for many backend components.
- We followed an object-oriented approach when designing the app, and used Model-View separation to protect the logic of the backend from potential errors due to user input on the frontend.
- The graph object GraphWidget represents the main view where the graph is drawn and transformed, which contains node and edge objects, each of which has its own internal data for backend logic, and a viewmodel for its visual representation.
- The controls of the app are QWidgets contained within a QDockWidget, which are used to switch between the various tabs used to manipulate and transform the graph.