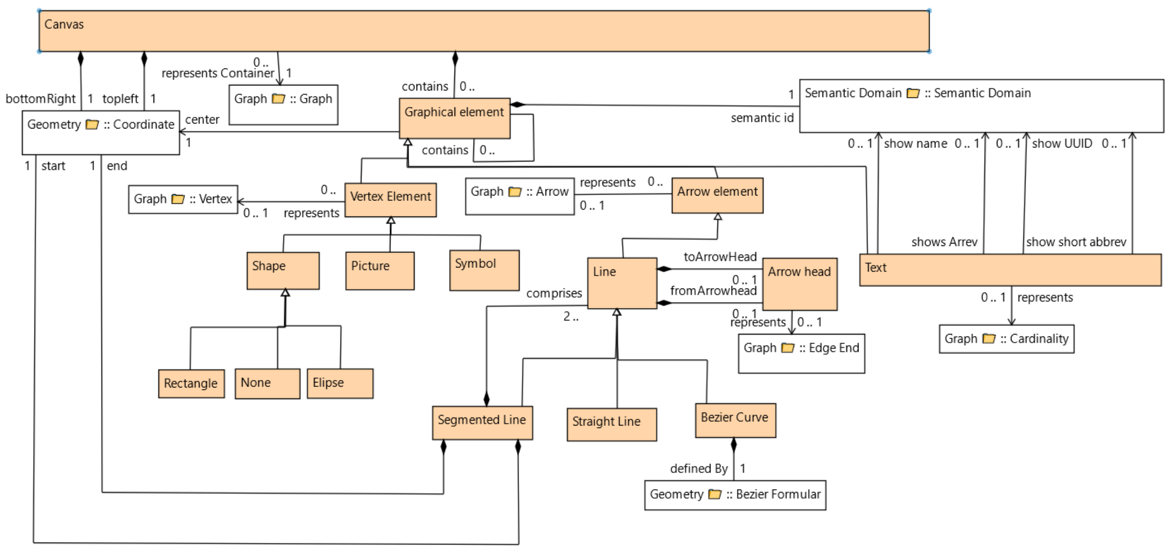


S23M

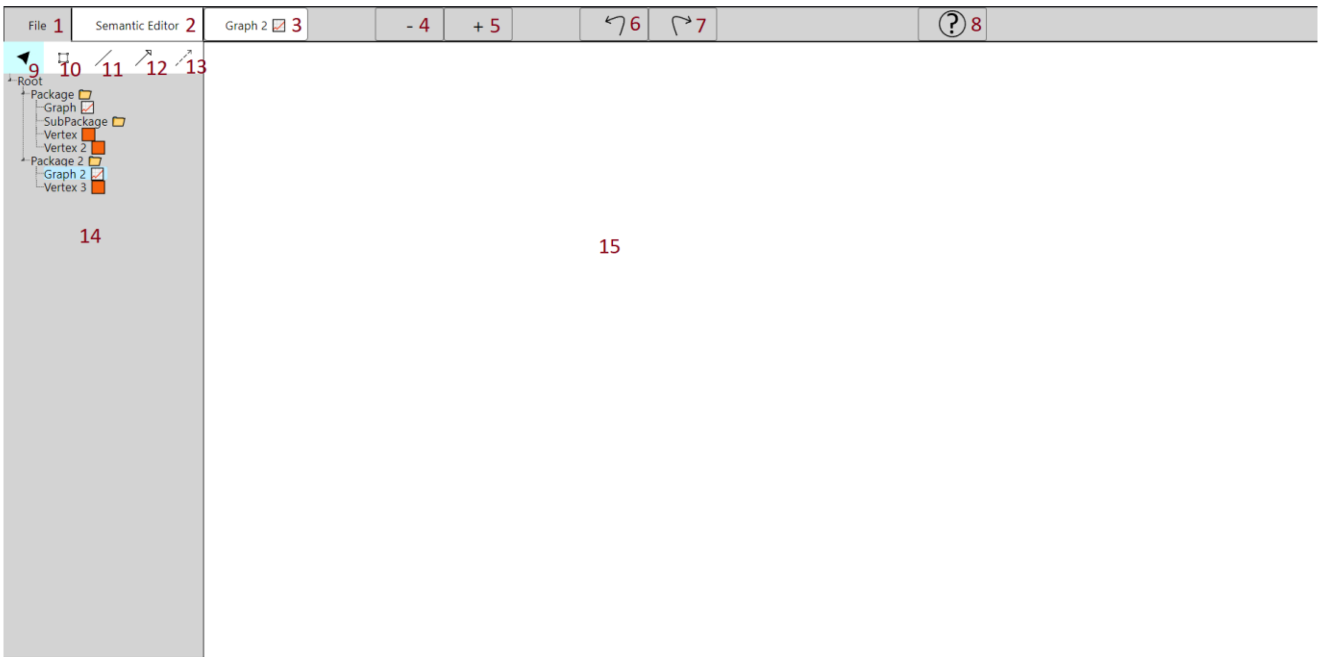
React Model Editor

Last Updated 14/10/2022

Contents

User Interface

Upon Accessing the webpage or loading a new File, you will be Presented with the following interface.

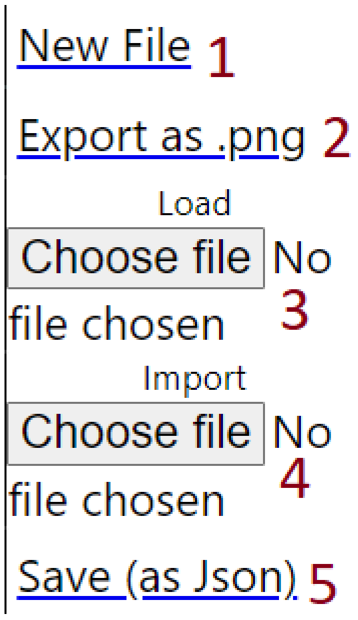


1. File menu - this tab opens up the file menu.
2. Semantic Editor - this tab opens up the semantic editor (See Semantic Editor section for explanation of the semantic editor)
3. Selected Graph - this displays the graph that is currently selected
4. Zoom out - Zooms the canvas out
5. Zoom in - Zooms the canvas in
6. Undo - Undo the previous action
7. Redo - Redo a previously undid action
8. Help button - Brings up the user manual
9. Select Tool - This tool allows users to select and interact with objects on the canvas
10. Vertex Tool - This allows users to draw a vertex on the canvas
11. Edge Tool - this allows users to draw edges between vertices on the canvas

12. Generalisation Tool - this allows users to draw generalisation arrows between vertices on the canvas.
13. Visibility Tool - this allows users to draw visibility arrows between vertices on the canvas.
14. Treeview - this is the area displaying the Treeview. (See Tree View section for explanation of the tree view).
15. Canvas - this is the area that displays the canvas for the currently selected graph.(See The Canvas section for explanation of the canvas)

File Menu

The file menu is a small menu that gets displayed in the top left of the screen upon selecting the “File” tab.

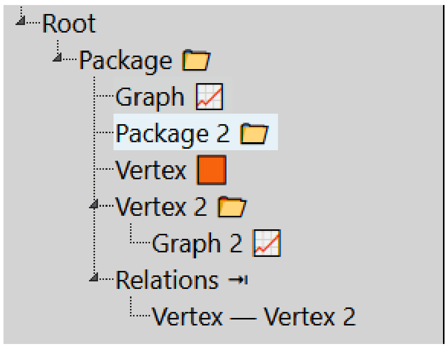


The file menu has 5 options;

1. New file - This option opens a fresh new file for the user.
2. Export as .png - This allows the user to export their current canvas a .png file
3. Load - this allows the user to load a .json file and it will have all the packages, graphs, vertices and edges as the loaded .json file had. Note that this will get rid of the current file that the user had and display the loaded file.
4. Import - this allows a user to import a .json file which will add any packages, graphs, vertices and edges to the current file that the user is working on.
5. Save - this allows the user to save the current file as a .json file, which can be loaded or imported.

Tree View

Elements



The treeview Contains 4 types of objects; Packages, Graphs, Vertex's and Edge's.

A Package is a container object that can be used to store other objects in the treeview, including other packages.

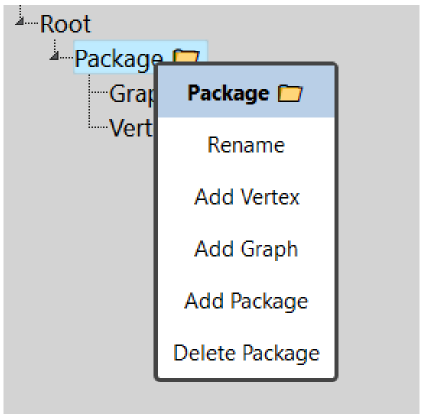
A Graph is a displayable object that stores canvas objects (see Canvas)

A (Tree) Vertex is a container object that can be drawn onto a graph, and is the Origin of any drawn vertices. The Vertex icon will change between an orange square or an open folder icon depending on if the vertex contains elements or not respectively.

An Edge is a relation between two Vertices that has been drawn within a graph. The icon next to the Edge name represents the relation type of the source and destination.

Creating/Deleting Elements

Tree view elements can be created through the context menu by right clicking on the desired parent container object and selecting "Add Package", "Add Graph" or "Add Vertex".



You will then be prompted to name this new object, and upon hitting enter the new object will be created under the chosen container.

To delete an Element right click the element you wish to delete and select “Delete [Element]” from the context menu. This will prompt you to double check if you wish to delete the chosen item, and upon selecting yes will delete said element and all of its dependents From the tree view and any graphs.

Other functionality

To rename an element, right click on the element and Select rename. Type the new name and hit enter. In the case of a vertex, this new name will be displayed on all instances of the vertex inside the canvas.

Root

Root is a special package in that it isn’t a real package and is only used as the top level to store packages. Therefore you can only create Packages in root, and cannot rename or delete Root.

The Canvas

The canvas is used to display and allow the editing of a Graph.

The four elements that can be drawn with the canvas are Vertex’s, Edge’s, Generalisation arrows and Visibility arrows.

Creating Vertex’s

A Vertex Can be Drawn on the Canvas in two ways, from an already existing vertex in a Package, or Created fresh on the canvas.

To Place an existing Vertex onto the canvas, simply drag and drop it from the Treeview onto the canvas. This will create a copy of the vertex stored in the tree. These vertex’s will be semantically Linked. Ie. Dragging and dropping the same vertex onto two graphs will both have the same Semantic Identity.

When a Vertex is created on the graph from a different package, The vertex will appear white with its parent package in the vertex name.

To create a new Vertex directly on the canvas, Select the vertex tool and click on the canvas where you would like to create the Vertex. This new Vertex will initially be named “Drawn Vertex” and its origin Vertex will be created in the package of the currently selected graph.

Creating Edge’s

To create an Edge, Select The Edge tool. Click on the vertex to be the source to begin creating the edge. While drawing an edge the user can click again anywhere on the canvas to create anchor points for the edge. (Anchor points currently cant be added/removed after Edge creation, so a new edge will have to be drawn to change). To finalize the edge select the destination vertex and then the edge will have been created.

Creating a generalisation arrow and visibility arrow is the same process as creating an edge, with the only difference being that the user must select the generalisation arrow or visibility arrow tool.

Moving Elements

To move elements on the canvas the user must first have the select tool selected. The user can then select the element they wish to move/alter.

Vertex

In the case of a vertex, the user can click and hold a vertex to drag it anywhere on the canvas. If there is an edge connected to the vertex the edge will follow the vertex while it is being dragged. Any vertex's that are connected to the vertex being moved this way will not be moved.

To resize a vertex the vertex must first be clicked and it will show 4 anchor points on the corners of the vertex. The user can then select one of those anchor points and drag it in any direction to resize the vertex in the direction that it is being dragged. Any edges connected to the vertex while it is being resized will still follow the vertex, but other vertex's will not be affected.

Vertex's can be ctrl clicked to select multiple at a time. if there are multiple selected they can all be dragged at once, and any vertex's that are not selected will not be affected.

If a large vertex is shift clicked and then dragged, any vertex smaller than the one selected will be dragged with it, even if they are not selected.

Edges / Generalisation arrows / Visibility arrows

In the case of edges, the user must first select the edge they wish to move before they can begin altering it.

Once an edge is selected it will display anchor points at the source, destination and anywhere that the user created anchor points during the edge's creation. These anchor points can then be clicked and dragged to move that point of the edge on the canvas.

Element Properties

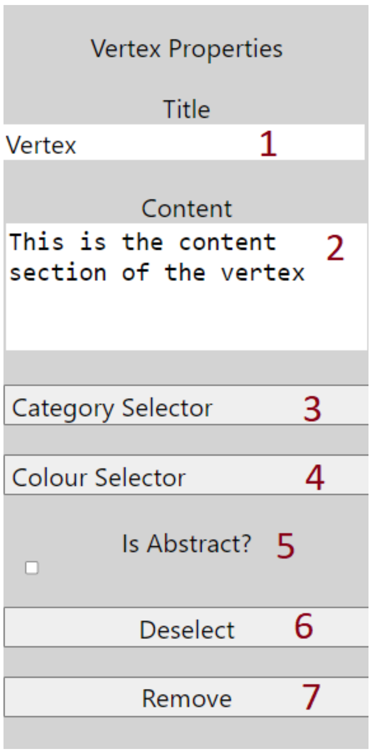
When Selecting an element on the Canvas with the select tool, the TreeView will be replaced with a properties panel for the specific object.

When changing a property, The change will happen dynamically, and in the case of a vertex the change will affect all semantically linked Vertex's as these are in essence the same object.

For example, If Vertex A exists on Graph 1 and Graph 2, making a change to Vertex A on either graph or the Treeview Vertex, will result in all 3 of these instances of Vertex A being updated.

Vertex Properties

The vertex properties menu can be accessed by selecting a vertex. The changes made in this menu will only affect the selected vertex and the other instances of that vertex.

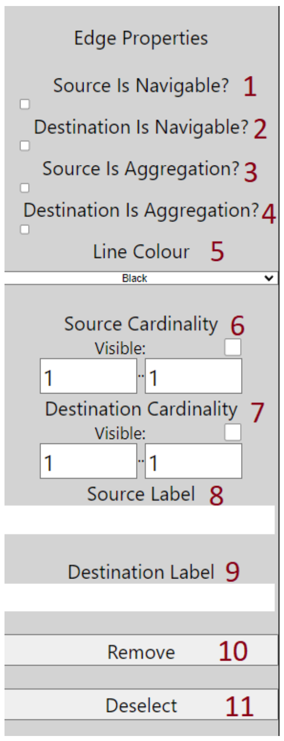


1. Title - This section is the title of the vertex.
2. Content - this section allows the user to write the content of a vertex.
3. Category selector - This button opens up a list of ModaMode categories that can be selected. the user can select if they want to show the name and/or icon of a category on the vertex. Multiple categories can be selected.
4. Colour selector - this button opens up a colour picker which allows the user to change the colour of the vertex.
5. Is Abstract? - this is a toggleable option which if toggled on will display the vertex name in italics to signal that the vertex is abstract
6. Deselect - This button deselects the vertex and brings back the tree view. Note that the vertex can also be de-selected by clicking anywhere in the canvas.
7. Remove - This button removes the selected vertex from the canvas (does not remove the vertex from the tree view).

*Vertex size will automatically adjust if needed to fit property values

Edge Properties

The edge properties menu can be open by selecting any edge on the canvas. Changes made will only affect the selected edge.



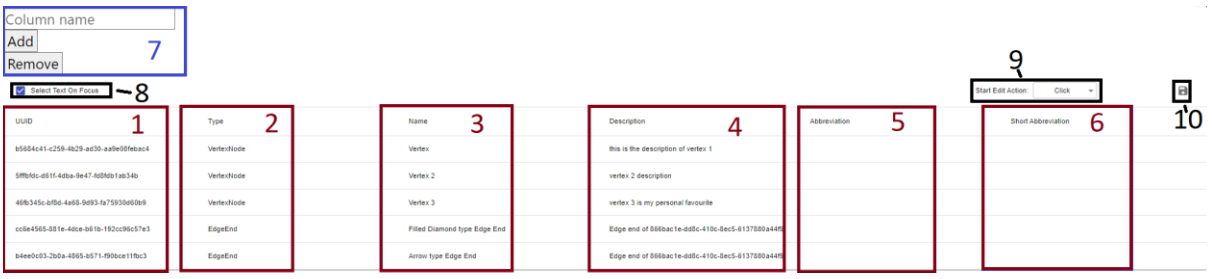
1. Source is Navigable - toggleable option which puts a navigable arrowhead on the source end.
2. Destination is Navigable - toggleable option which puts a navigable arrowhead on the destination end.
3. Source is Aggregation - toggleable option which puts an aggregation arrowhead on the source end.
4. Destination is Aggregation - toggleable option which puts an aggregation arrowhead on the destination end
5. Line Colour - Drop-down box which allows the user to select a line colour between black, red, blue and green.
6. Source Cardinality - allows the user to toggle the visibility of the source cardinality. The cardinality is based on the numbers inside the boxes.
7. Destination Cardinality - allows the user to toggle the visibility of the source cardinality. The cardinality is based on the numbers inside the boxes.
8. Source Label - allows the user to input text which will be displayed on the source end.
9. Destination Label - allows the user to input text which will be displayed on the source end.
10. Remove - Button which removes the edge from the canvas.
11. Deselect - A button which deselects the currently selected edge. Note that the edge can also be deselected by clicking anywhere on the canvas.

Generalisation and visibility arrow properties

The generalisation arrows and visibility arrows only contain two properties, which are the "Deselect" and "Remove" properties which provide the same functionality as the deselect and remove properties that are contained in both vertex's and edge's.

Semantic editor

The Semantic editor is a special table displaying all the information of the different vertices and edge ends that are present within the different packages and graphs. Upon clicking on the semantic editor tab, the canvas and the tree view will be hidden and replaced with the UI for the semantic editor which will look like this;



UUID	Type	Name	Description	Abbreviation	Short Abbreviation
b5684c41-c259-4b29-ad30-aa9e08febac4	VertexNode	Vertex	this is the description of vertex 1		
5fffbfdc-d61f-4dba-9e47-fd8fdb1ab34b	VertexNode	Vertex 2	vertex 2 description		
46fb345c-bf8d-4a68-9d93-fa75930d60b9	VertexNode	Vertex 3	vertex 3 is my personal favourite		
cc6e4565-881e-4dce-b61b-192cc96c57e3	EdgeEnd	Filled Diamond type Edge End	Edge end of 866bac1e-dd8c-410c-8ec5-6137880a44f9		
b4ee0c03-2b0a-4865-b571-f90bce11fbc3	EdgeEnd	Arrow type Edge End	Edge end of 866bac1e-dd8c-410c-8ec5-6137880a44f9		

1. UUID - This is the unique ID of all the vertex and arrow objects that are present within packages and graphs. This section of the semantic editor cannot be edited.
2. Type - This is the object's type. There are currently only two types which are; Vertex nodes and Edge ends. This section of the semantic editor cannot be edited.
3. Name - This displays the name of the vertices and the edge ends. This section of the semantic editor can be edited to change the names of the edge ends and vertices. In the case of a vertex, updating its name in the semantic editor updates its name in the tree view and canvas for all instances of the vertex.
4. Description - This contains a description for the vertices or edge ends. This section of the semantic editor can be edited to write a description for vertices and edge ends.
5. Abbreviation - this contains an abbreviation for the vertices and edge ends. This section of the semantic editor can be edited to create an abbreviation for the vertices and edge ends.
6. Short abbreviation - this contains a short abbreviation for the vertices and edge ends. This section of the semantic editor can be edited to create a short abbreviation for the vertices and edge ends.
7. Add / remove column - this section of the semantic editor allows the user to add and remove their own custom columns within the semantic editor.
8. Select text on focus option - this is a toggleable option which allows the user to choose whether they would like to select the whole text field upon editing text.
9. Start edit action options - this allows the user to choose if they would prefer to double click or just single click in order to select text within the semantic editor
10. Export semantic editor data - this allows the user to export the semantic editor data into an Excel worksheet file.

To exit out of the semantic editor, all the user needs to do is select the Semantic Editor tab again and it will bring back the canvas and treeview.

Navigation

Through the context menu there exists the ability to navigate to different occurrences of a Vertex.

To do this, Right click on a Vertex either in the treeview, or on the current canvas and Select navigate. You will be shown a List of Locations this vertex appears. Clicking one of these locations will load the relative graph onto the canvas and select the vertex.

Undo/Redo

the undo/Redo buttons will undo/redo recent actions in the canvas and/or treeview. Due to undo/redo's reliance on memory, undo/redoable actions is limited to the 20* most recent actions.

*This limit can be changed and is currently very reserved as in testing with small models an "action" only takes up a few kilobytes, but with large models they could be a few megabytes, therefore with a high limit such as 100, the browser would require several hundred megabytes of memory.

Saving and Loading

Models Can be Saved to a File by selecting Save To Json in the File menu.

You can Load saved models by clicking Browse under Load in the file menu and selecting the saved file. This will load the model from the file replacing anything currently in the editor.

Importing

Loading a file through import, instead of load, will load the data into the current model.

Build Instructions

in the project directory you can run “npm start”. This runs the app in development mode and can be viewed if http://localhost:3000 is opened in a browser (running npm start should automatically open http://localhost:3000). if any edits are made and saved to the program's code this page will reload upon saving those edits to reflect what has been done.

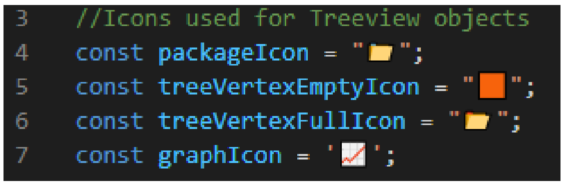
there is also a github pages live build that can be accessed with the following link:
https://s23m.github.io/lclb/

When hosting elsewhere, the homepage in package.json will need to be changed from `"https://s23m.github.io/lclb/"` to the new address.

In order to update the repository github pages build, the user can run the deploy script with “npm run deploy” in the project directory. this will create a build of the current checked out branch and make that the live github pages build.

Config.js

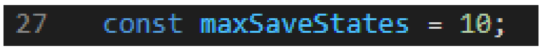
Config.js is a file created to allow for certain aspects of the program to be easily configured by changing the variables inside it. The first set of variables are for the icons that are displayed in the tree view of the program and can be found and configured in lines 4-7.



```
3   //Icons used for Treeview objects
4   const packageIcon = "📂";
5   const treeVertexEmptyIcon = "🟧";
6   const treeVertexFullIcon = "📂";
7   const graphIcon = '📈';
```

- packageIcon - This is the variable that determines what icon is displayed next to packages
- treeVertexEmptyIcon - this variable determines what icon is displayed next to empty vertex’s
- treeVertexFullIcon - this variable determines the icon that is displayed next to a vertex that contains other elements within itself
- graphIcon - this is the icon that is displayed next to graphs.

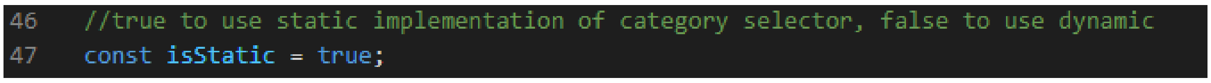
The second configurable variable in this file is the maxSaveStates variable, which can be found and configured on line 27.



```
27   const maxSaveStates = 10;
```

This variable changes how many redo/undo actions the program is limited to. for example, if the maxSaveStates is 10, the program will only allow the user to undo 10 actions. Keep in mind save states are stored in memory which is why they are limited, however they are relatively small and only scale to be a few kilobytes per object. The default value for this is 10.

the third configurable variable is the “isStatic” variable, which can be found and configured on line 47



```
46   //true to use static implementation of category selector, false to use dynamic
47   const isStatic = true;
```

This variable determines whether the category selector uses the static implementation or the dynamic implementation. The difference between the static and dynamic implementations of the category selector is that the static category selector downloads all the icons from the website and stores them locally. dynamic however makes fetch requests to the server for each new icon. The default value to this is true.