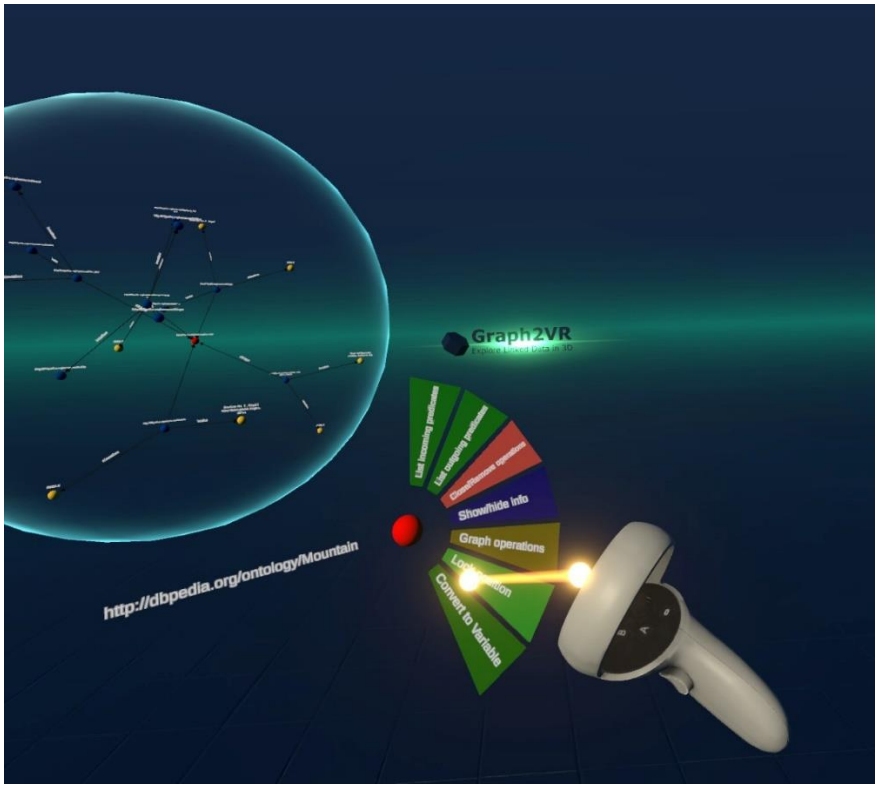
Graph2VR

(version 1.0.0)

User manual

Graph2VR is a PhD project – a prototype to visualize and explore Linked Data in Virtual Reality.
It was developed in Unity (version 2021.2.15f) and uses DotNetRDF [1] to connect to SPARQL Endpoints.



Contents

Starting the application

Standalone (Meta Quest 2)

1) Turn on the Quest 2 headset and connect it to your PC or Mac via cable.
2) Make sure to allow the connection from the PC to the headset.
3) If you have not done so already, enable developer mode on the Quest 2 headset. Go to the Settings menu, click on Headsets Settings and toggle on the Developer Mode.
4) Copy the Graph2VR.apk to the Quest 2 headset. You can use Sidequest to send the apk file to the headset. [2]
5) Graph2VR should now be accessible on the Quest 2 headset. It should appear in the applications after selecting 'unknown sources'.

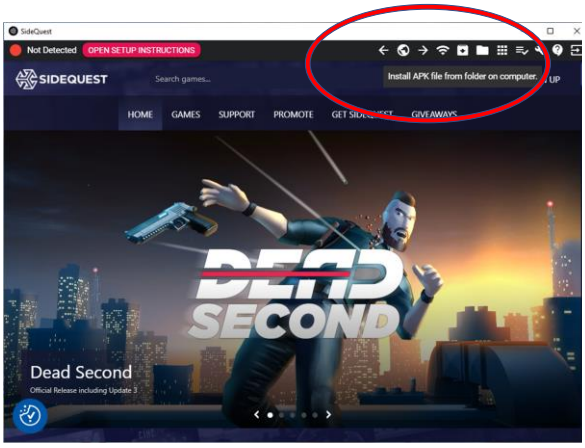


Figure 1. Screenshot: Sidequest can be used to send the Graph2VR.apk file to the Quest 2 headset.

As a Windows application (HTC Vive & Meta Quest 2)

1) Download the Windows built of Graph2VR (http://github.com/molgenis/Graph2VR/Windows_build/).
2) Make sure your VR headset is connected to your computer.
3) Run Graph2VR.exe.

Command Reference

Graph2VR has been developed for the Meta Quest 2 (previously Oculus Quest 2). Graph2VR should also be compatible with the HTC Vive, but the lower resolution of the Vive makes it harder to read the texts in the application.

Meta Quest 2 Controller

Quest 2 controllers are designed for the left and right hand. The right controller has a home button, the left one a settings button.

The A, B, X and Y buttons are not used in Graph2VR because they do not have an equivalent on the HTC Vive.

Figure 2. Meta Quest 2 Controller

HTC Vive Controller

Vive controllers are not explicitly designed for the right or the left hand. Instead of the buttons A, B, X and Y and the thumbstick, they use a touchpad.

In Graph2VR, the Quest's thumbstick controls match the Vive's touchpad controls.

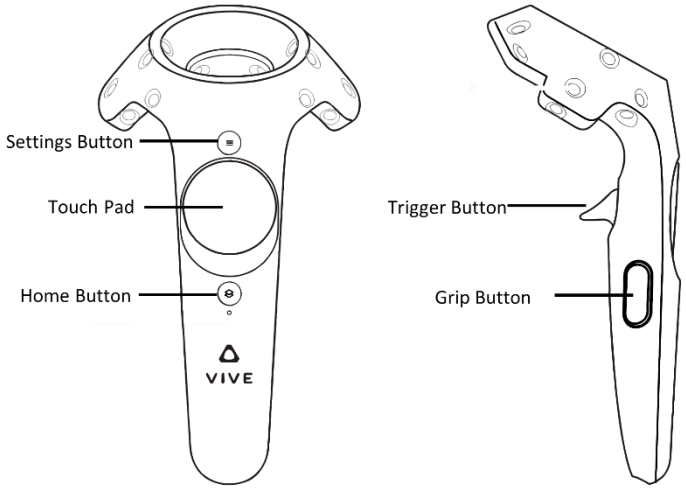


Figure 3. Vive Controllers
(images modified, originals are from [3])

Menu Controls

Open main menu	Settings Button
Select menu item / Set Sliders / Close menu	Trigger Button
Home menu (SteamVR)	Home Button
Grab slider knob (if available)	Grip Button
Open node menu	Aim at a node with the laser and press the Trigger button
Open edge menu	Aim at an edge with the laser and press the Trigger button

Complete Controls

Navigation

Teleportation	Touchpad right controller – hold down to select target position (green circle) then release to teleport
Rotation	Touchpad right controller – Left / Right
Flying	Touchpad left controller (looking direction)
Walking / Turning	Headset position / Looking direction

Node interaction

Drag and Drop nodes	Grip Button (when node is touching the small light ball in front of the controller, touching with the laser doesn't work)
Selecting a node and opening the node menu	Trigger Button
Creating a new variable node	Holding the Trigger button for 2–3 seconds
Highlighting a node and increasing the text	Hover over the node with the laser

Edge interaction

Selecting Edge & Opening Edge Menu	Trigger Button
Creating new edge	Hold the trigger while the controller is touching a node. You can then draw an edge to another node and release the trigger when touching it.

Graph interaction

Grabbing the closest graph	Grip buttons of both controllers
Rotating the whole graph	Hold the grip buttons and rotate the graph with the controllers
Making the graph bigger (or smaller)	Hold the grip buttons and move the controllers away from (or towards) each other
Opening the graph operations menu	Select any node or edge within the graph and find the graph operations in the node / edge menu

Entering text via the keyboard

Pressing a key on the virtual keyboard	Use the drumstick to drum on the key

Navigation in the visualization

The goal for the navigation in Graph2VR was to give the user as much freedom as possible.

In addition to **moving and turning** in the room, the user can either **teleport or fly** in the virtual environment.

To **fly**, use the thumbstick (Quest2) / trackpad (Vive) of the left controller.

The thumbstick/trackpad of the right controller is used to **teleport** and to rotate by a small angle to the side. To teleport, pull the right thumbstick towards you. When holding the thumbstick, a green circle appears on the laser's tip. Once you release the thumbstick, you are teleported to the position of the green circle (See Figure 4).

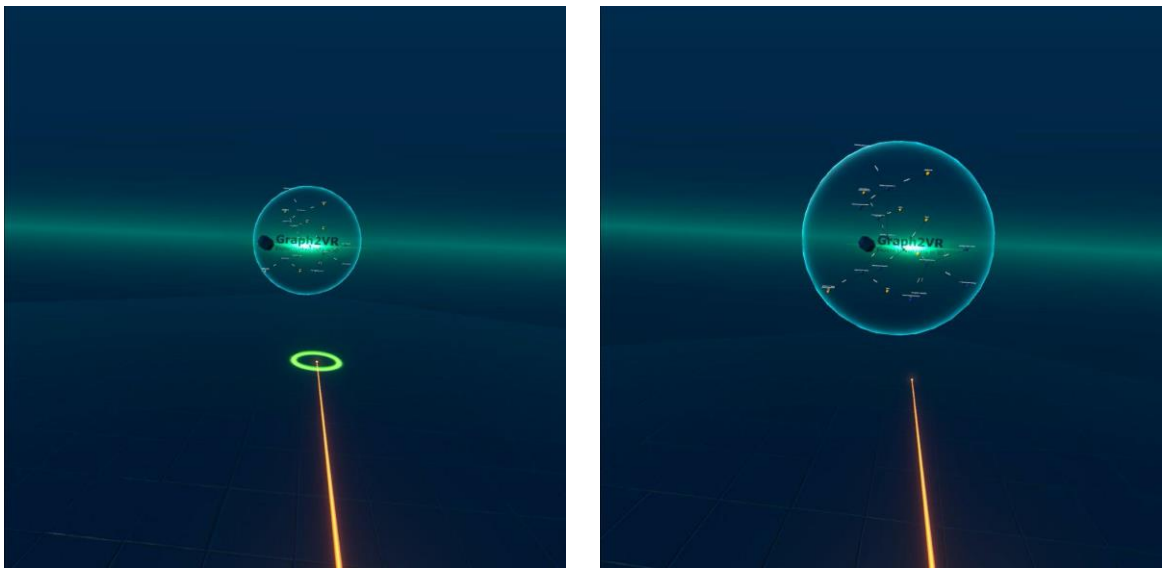


Figure 4. A green circle indicates the target location for teleportation. Screenshots are taken before and after the teleportation.

Gesture controls (grabbing, zooming, rotation)

The gesture controls are used to interact with single nodes or the whole graph.

You can grab a node with your controller when it is close to the node. The reference point is the light dot in front of the controller. Then, while pressing the grip button of this controller, you can drag and drop the node.

When you press the grip buttons of both controllers at the same time, you grab the closest graph (based on the middle point of the graph). You can drag and drop the graph while holding both grip buttons.
It is also possible to rotate and zoom the whole graph with the controllers (See Figure 5).

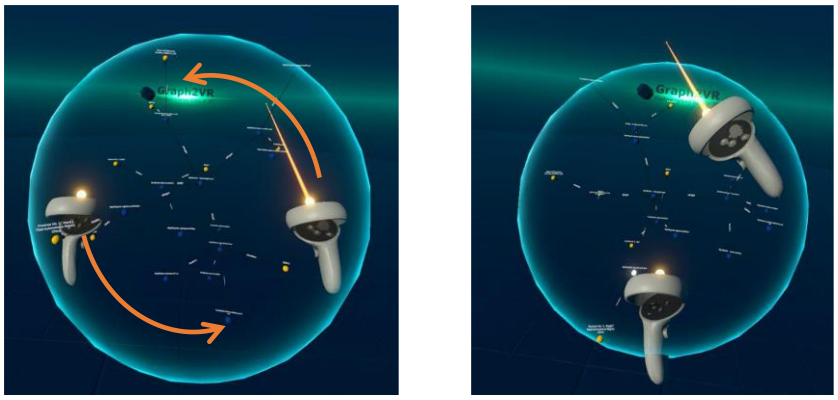


Figure 5. Two screenshots showing a graph being rotated to the left.

To make a graph smaller, move the controllers towards each other while holding the graph. Vice versa, you can expand a graph by moving the controllers apart (See Figure 6).

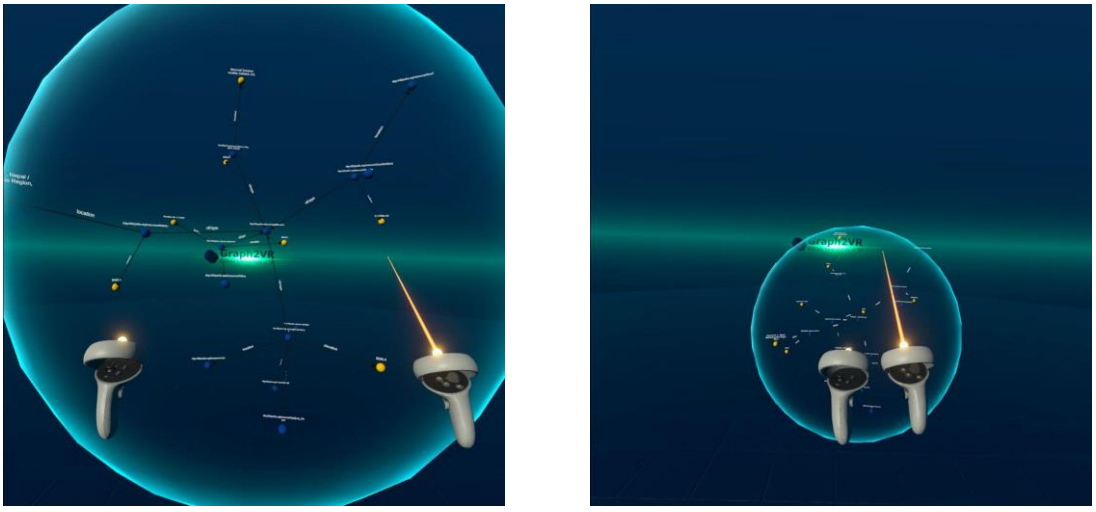


Figure 6. A graph can be zoomed in / out by moving the controllers towards / away from each other while holding the graph with the grip buttons of both controllers.

Graph2VR Menus

Main Menu

You can open the main menu by pressing the Settings button. It includes general functionalities to:

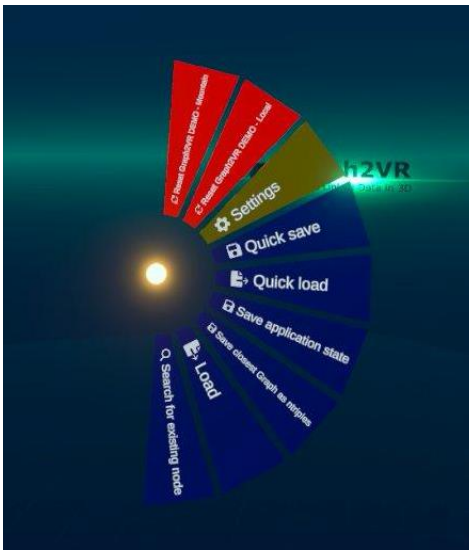


Figure 7. A screenshot of the Graph2VR Main Menu.

- **Reset the application**
  "Reset Graph2VR DEMO – Mountains" opens a graph with information about mountains, their location and elevation from DBpedia.

  "Reset Graph2VR DEMO – local" accesses a local SPARQL Endpoint with a dataset we used for demo purposes.
- **Settings**
  Leads to the Settings menu.
- **Quick save** / **Quick load** function
  Saves the state of the application. There is only one Quick Save state, which gets overwritten when saving again.
- **Save application state**
  Works exactly like Quick Save but allows you to specify a filename for the saved file. This can be used to create multiple save states. Saving the application state also saves (a reduced version of) the images loaded into the application. The application states can be used offline when the program has no connection to the internet.
- **Save closest graph as n-triples**
  N-triples (.nt) is a standard format to save and load triples. An n-triples file can be used in other applications, e.g. to upload the data into a Virtuoso server.
- **Load**
  Load can be used to load a save state or import an n-triples file. Note that loading a save state will overwrite the current session, while data from an n-triples file is added as a new graph.
- **Search for an existing node**
  This search function can be used to find a specific node within the graph database. It is placed in the main menu so that the user can create a new graph if there are no graphs left.

Settings menu

Within the settings menu, you can adjust settings that affect the whole application (see Figure 8).

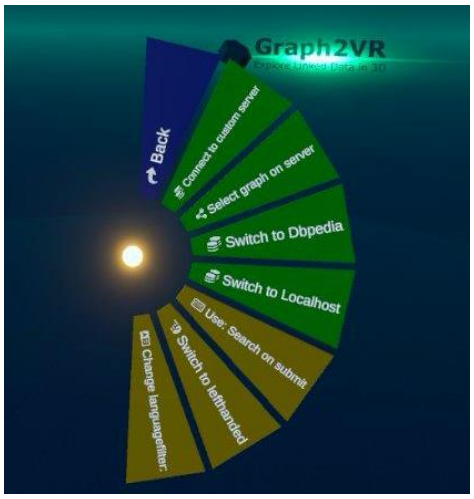


Figure 8. Screenshot of the Graph2VR Settings Menu.

One of the most important settings is the graph to which database queries need to be sent:

- **Connect to custom server**
  To connect to a SPARQL endpoint that has not been predefined via the Settings singleton, it is possible to manually type the address of a server.
  User tip – If you need a specific server often, add it to the settings singleton.
- **Selecting a graph on the server**
  Just as a database server can host multiple databases, it is also possible for a graph database to host multiple graphs, e.g. for the internal settings and data itself.

  All graphs within the selected SPARQL Endpoint are requested and listed in a submenu (see Figure 9).
- **Switch to DBpedia**
  DBpedia has one of the most famous SPARQL endpoints, hosting Linked Data created based on Wikipedia. It is shown because it is one of the graphs added to the settings singleton. The entry includes the SPARQL server and the graph.
- **Switch to Localhost**
  It is common to host a graph database on a local PC and to access the data from there. For example, we used a local database to host our demo database, and the graph saved in the settings singleton points to the graph we used to store it.
  User tip – You probably need to adjust the graph to access your dataset.

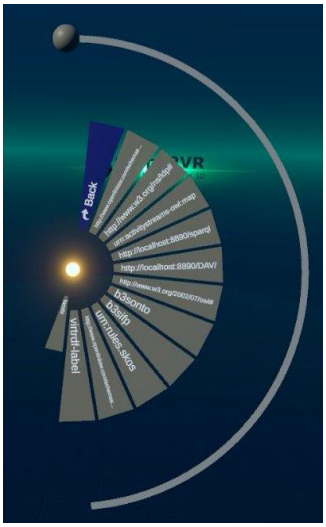


Figure 9. Screenshot of the Graphs on the DBpedia server.

- **Use: Search on submit / Search on keypress**
  This option determines whether a query should be sent after every keystroke or only after pressing the return key on the virtual keyboard. When the database has longer response times, 'search on submit' might be the better option. 'search on keypress' only searches for inputs with at least four letters.
- **Switch to left-handed/right-handed mode**
  This option toggles between the right-handed/left-handed mode of the application. In the left-handed mode, the laser pointer is on the left hand and the menu is on the right hand. The direction of the menu, as well as teleportation/flying, does not switch. This might be improved in future versions.

- **Change language filter**

  For literals (strings), the language is often encoded within the data. This makes it easy to represent and translate the same information (relationships) in other languages by just translating the name. This is especially important for setting the labels in Graph2VR. A language filter can be used to avoid receiving a random language, which could be a language that cannot be displayed correctly. Some languages are preset. Other language tags can be added manually.

  Important to know: The language filter only allows results that either have the desired language tag or no language tag at all. (This might reduce the number of query results if results are only available in other languages). Use the "Don't filter on language code" button to remove this filter. Do not expect the current graph to update automatically – the language filter is only applied to new queries.

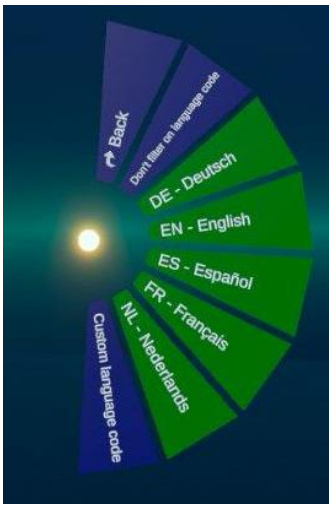


Figure 10. Screenshot of the language filter menu in Graph2VR.

Node Menu

Selecting a node with the laser and pressing the trigger button opens the node menu for that node in the circle menu. The small red node in the middle of the menu indicates the selected node (Figure 11). The selected node in the graph is red as well. The node's name (or URI) is displayed next to the circle menu (e.g. “http://dbpedia.org/resource/Kangchenjunga” in Figure 11).

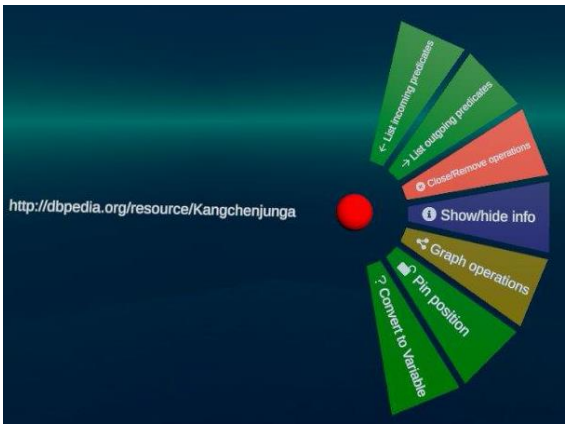


Figure 11. Screenshot of the node menu in Graph2VR.

The node menu contains the following options:

- **List incoming predicates**

  All (up to 100) distinct predicates of triples where the selected node is the object are listed in the menu.

- **List outgoing predicates**

  All (up to 100) distinct predicates of triples where the selected node is the subject are listed in the menu.

  Literals (indicated as yellow nodes) do not have outgoing predicates, so the node menu does not show this option either (see Figure 12).

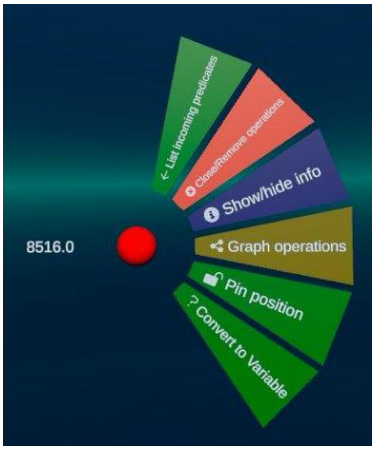


Figure 12. Literals do not have any outgoing predicates, so the node menu does not show outgoing predicates either.

The number next to the name of the predicate indicates how many triples of this kind are in the database. These can be used to extend the graph by adding new triples.

If there is a label for the predicate, it is displayed – these buttons have a darker grey button colour.

Predicates that do not have a label are ordered alphabetically by their URI. Their names are shortened for readability purposes, but the complete URI is displayed when hovering over the button.

A scrollbar appears if there are too many entries in the menu, as shown in Figure 13. You can grab the small knob with your grip button or point on the scrollbar to position it. If you try to move the knob with the laser, you may accidentally close the menu.

The amount of new spawning connections depends on the number of entries in the database and the query limit. The query limit can be set on the Limit slider below the Circle menu.

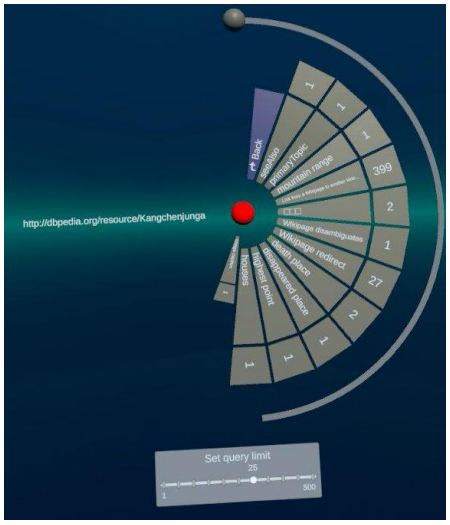


Figure 13. Screenshot of the incoming connections with the selected node as object.

The Node menu continues with the

- **Close/Remove operations**
  This leads to another submenu with several options to remove Triples and nodes around the selected node.

The Close/Remove operations submenu offers the following options:

- **Collapse Incoming**
  Removes "leaf nodes" that are part of a triple with the selected node as the object. Leaf nodes are nodes that are only connected to the selected node and have no further connections.
- **Collapse Outgoing**
  Removes "leaf nodes" that are part of a triple with the selected node as the subject.
- **Collapse All**
  Removes nodes that are either incoming or outgoing leaf nodes connected to the selected node.
- **Close node**
  Removes the selected node from the visualization (including its leaf nodes).

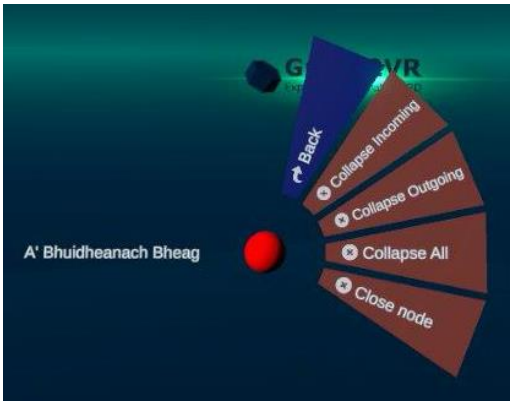


Figure 14. Screenshot of the Close/Remove operations submenu of a node.

Collapsing incoming or outgoing connections removes "leaf nodes" that are not further connected in the graph. "Collapse All" combines the two operations and removes leaf nodes of incoming and outgoing connections. Close node removes the selected node from the graph. In contrast to the previous operations, the Close option can also remove nodes with multiple connections. These operations will, of course, only affect the visualization and will not remove those triples from the database.

The node menu continues as follows:

- **Show/hide info**
  The info panel contains the URI of the node, its label(s) and type(s). Selecting the URI with the laser via the trigger button opens that URI in a browser.
- **Graph operations**
  Graph operations are operations that affect the whole graph. This is described in the Graph operations.
- **Pin position/Unpin position**
  This operation pins the selected node to its current position. It is still possible to move it by hand or with the whole graph, but it should be affected by the layout algorithms while pinned. The pinned position is visually indicated by a nail pinning the node to that position.
- **Convert to Variable**
  This converts the node into a variable node. A variable node is green, and its name changes to a variable name, e.g. "?Variable1". The question mark in the name also indicates that this is a variable for SPARQL. You use variable nodes (and edges) to find datasets that match the query pattern. If variables have the same name, they will be treated as the same variable in the query.

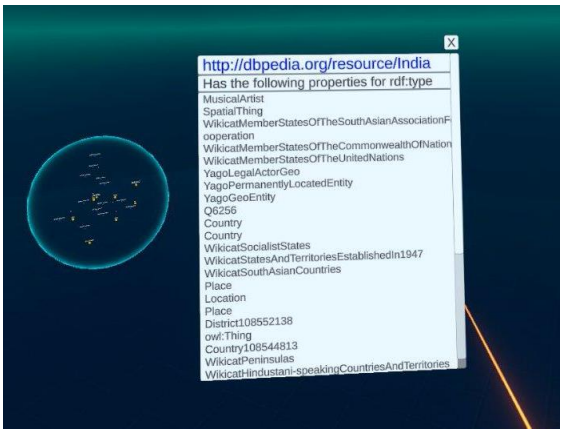


Figure 15. Screenshot of the info panel about a selected node.

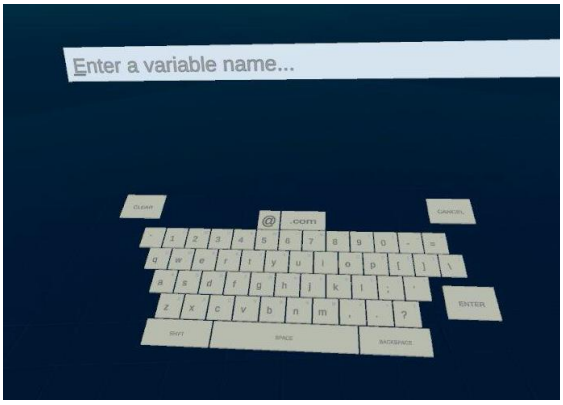


Figure 16. Texts for search terms and variable names can be entered via a virtual keyboard.

Once a node has been converted to a variable, the menu has some additional options:

- **Undo Variable Conversion**
  This converts the variable back to the previous node.
- **Rename variable**
  This renames the node's variable name. You may want to use an existing variable name for your SPARQL query (see Figure 16).
- **Context specific search**
  This searches for entries that fill this variable's place while taking the selected triples into account. (See Edge Menu).

You can also create a new variable node without using the menu:

Hold the trigger button for 2–3 seconds to spawn a new variable node. New nodes will spawn at the position of your left controller. Some of the layout algorithms will move them immediately.

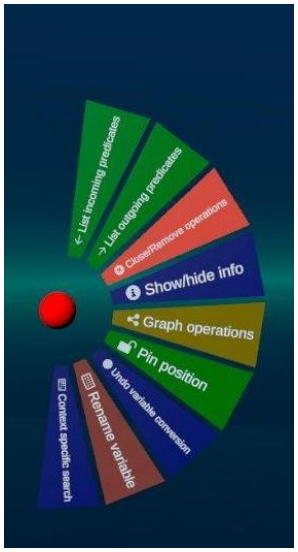


Figure 17. Screenshot of the node menu for a variable node.

Edge Menu

The edge menu opens when you select an edge with the laser and click the trigger button. A red cylinder indicates that an edge is selected. In addition, the edge's name is displayed on the left of the circle menu.

The basic edge menu contains the following options:

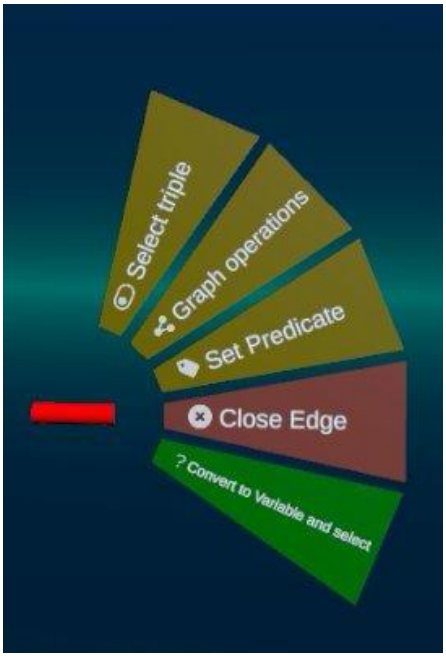


Figure 18. Screenshot of the edge menu in Graph2VR.

- **Select triple**
  Selected triples contain a subject (node), predicate (edge) and object (node). They are highlighted in yellow. Selected triples can be used to create a SPARQL query visually. All selected triples are part of the query pattern. Once a triple has been selected, its edge menu offers options to modify and send the query.
- **Graph operations**
  Graph operations are operations that affect the whole graph. This is described in the Graph operations.
- **Set Predicate**
  This replaces the predicate of the current edge with another (predefined) predicate.
- **Close Edge**
  This removes the selected edge. If a leaf node is connected via this edge, it is also removed.
- **Convert to variable and select**
  This converts the selected edge to a variable and makes it part of the current selection.

Selected Triples (node, edge, node) are part of the query.

If an edge is a variable edge but is not selected, it is shown in green. In this case, it is not part of the query.

Once an edge has been converted to a (selected) variable, the menu has additional options (see Figure 19):

- **Order By**
  Select the variables that should be used for ordering the results. Those results can be ordered ascending (ASC) or descending (DESC). It is possible to choose multiple variables and their priority order.
- **Rename Variable**
  It is possible to rename variable nodes and edges to make them easier to find in the Order By submenu. Variable names always need to start with a question mark. If the question mark is not typed while renaming, it is added automatically.
- **Set predicate**
  This is meant to create a new connection between nodes. Instead of a generic URI, it can be useful to set the URI of a predicate to a specific URI. We added templates for URIs to describe

the similarity of different datasets to each other. Therefore, the predicates suggested in the Scientific Lenses paper have been selected [3].

- **Limit Slider**
  The limit slider below the circle menu limits the maximum number of results you will receive. The limit is also used when expanding the graph via the incoming or outgoing predicates. It has some predefined fixed values between 1 and 400.

Once you have selected all the triples necessary for your query, you can click an edge from one of the selected triples. There should be two different buttons to request results for the query in the circle menu:

- **Query similar patterns**
  The results will be a series of 2D result graphs ("semantic planes"). These will appear in the direction you are looking. To prevent the result graphs from colliding with the ground, do not look at the floor of the platform.
- **Query similar patterns (single layer)**
  This will request and merge the results into a single graph without duplicate nodes.

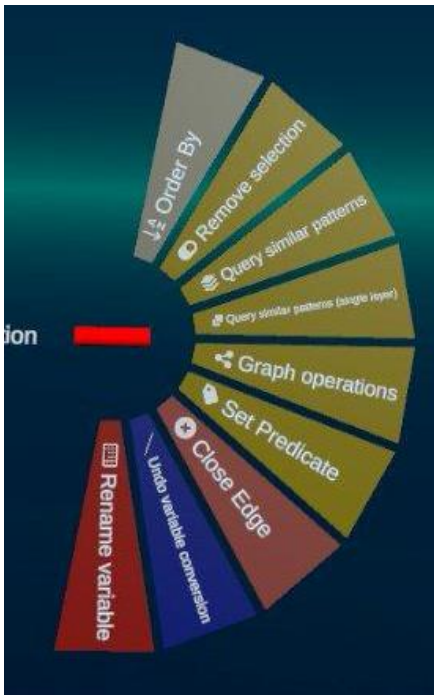


Figure 19. Screenshot of the menu of a selected variable edge in Graph2VR.

You can also create a new variable edge without using the menu:

Put your controller at the position of a node and keep the trigger button pressed. You can now create a new (outgoing) edge to a second node and then release the trigger button. If there is more than one edge between the same nodes, they will be curved to increase the readability of their labels.

Graph operations

The Graph Operations submenu can be opened from any node or edge within a graph. It contains options that affect a whole graph.

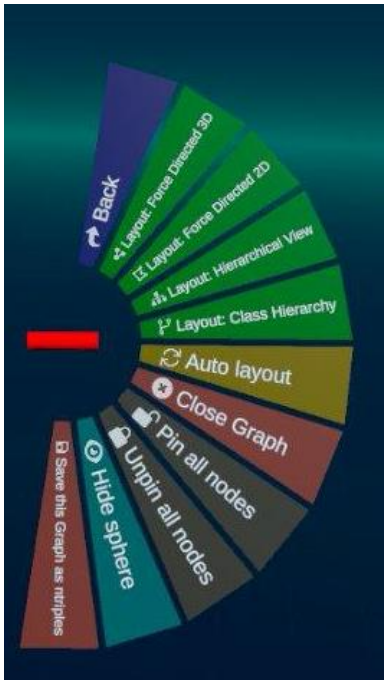


Figure 20. Screenshot of the graph operations menu in Graph2VR.

The options are:

- **Layout: Force-directed 3D**
  The Force-directed 3D layout is a 3D layout wich is based on the Fruchterman-Rheingold Algorithms. It reorders the graph by applying attracting and repulsive forces between the nodes. Those forces get smaller over time (cooling).
- **Layout: Force-directed 2D**
  This layout is a (quickly) improvised algorithm to reorder the nodes in a 2D pattern.
- **Layout: Hierarchical View**
  Since the 3D force-directed algorithm orders the nodes in the form of a sphere, it can be hard to spot the node you are looking for. They are easier to spot when the nodes are in linear and alphabetical order. This can be done with the hierarchical layout.

  When new connections are opened, the predicates should all be stacked in the same direction. The nodes spawn, alternating as horizontal or vertical stacks, to avoid too many overlapping nodes.
- **Layout: Class Hierarchy**
  This layout is based on the Class hierarchy tree structure known from Protégé. It uses the Subclass hierarchy to form a (horizontal) tree structure. Individuals of those classes are opened in the vertical axes of (one of) their classes. The nodes are sorted alphabetically. Nodes that do not fit in the class hierarchy are stacked horizontally.

  Further connections from those individuals either point to the side or directly to nodes that already have their place within the class hierarchy. If a node is not in the subclass hierarchy, it will be stacked next to the hierarchy.
- **Auto layout**
  This button could also be called "Refresh layout". It triggers the current layout algorithm to refresh the order of the nodes. While this is relevant for the force-directed algorithms, the "Hierarchical View" and the "Class Hierarchy" layout immediately update the graph.
- **Close Graph**
  Removes the whole graph from the visualization.
- **Pin all nodes**
  Pinned nodes are not affected by the layout algorithms. This option pins all nodes within the current graph.
- **Unpin all nodes**

This option unpins all nodes within the current graph.

- **Hide sphere / Show sphere**
  This toggle button switches the sphere around a graph on and off.
- **Save the graph as n-triples**
  This saves all triples in the graph as an n-triples file. However, single nodes in the graph and variables are not saved.

After a query has been performed, additional buttons are added to the menu.

For the "parent graph", there is the option:

- **Close Child graphs**
  This removes all graphs created by a query from this graph.

Child graphs have the additional option to remove graphs that have been created with the same query:

- **Close unmodified sibling graphs**
  This closes all the other graphs that were created while executing the same query from the same parent graph. The current child graph persists. Once one of the "sibling graphs" has been modified (e.g. by expanding or collapsing triples), they should not be affected by this operation anymore.

The Settings file

The Settings singleton is a file that contains most of the settings and predefined predicates, as well as the known SPARQL Endpoint. It can be modified with a text editor or Unity (see Figure 21) in the uncompiled version of Graph2VR.

Within the settings singleton, there are predefined SPARQL Endpoints.

For each **SPARQL endpoint**, the base URI defines to which server a SPARQL request must be sent. This could, for example, be the DBpedia SPARQL Endpoint (https://dbpedia.org/sparql).

Within each endpoint, there can be different graphs that host different information.

In the case of DBpedia, the **base graph** (http://dbpedia.org) contains the actual DBpedia data. Other graphs might host different information, like some settings for the host server.

"**Database Supports bif:contains**" – This is the checkbox to use the index if the SPARQL server supports the bif:contains command. This is not supported on all servers. It allows a faster search based on an indexed SQL table on the host server.

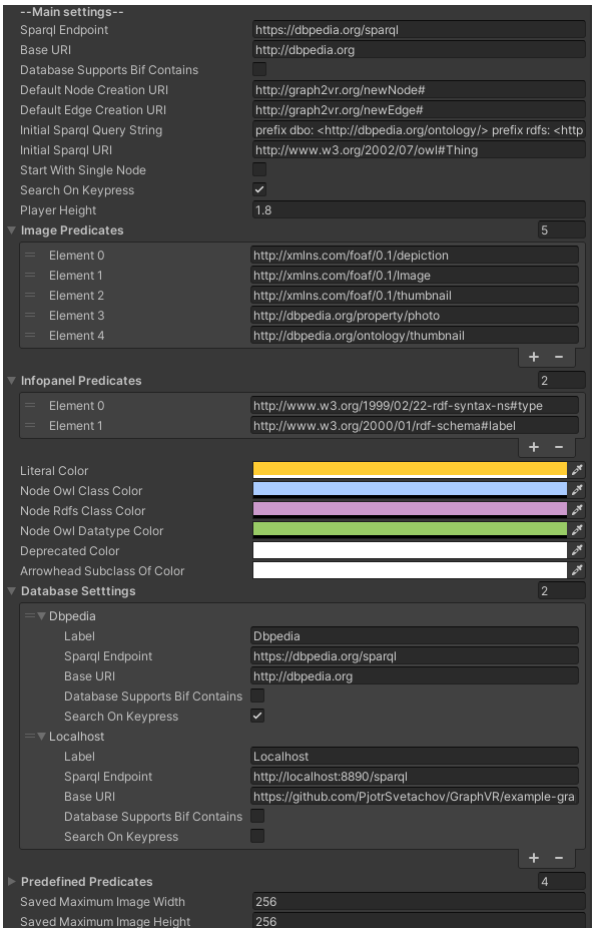


Figure 21. Presets are defined in the Settings Singleton. This is the settings singleton in the Unity inspector.

Not every SPARQL endpoint has such a pre-indexed table. So, even if a server like Virtuoso supports the command, it is not guaranteed that such an index has been created.

If a SPARQL endpoint like DBpedia supports the "bif:contains"-command, the textual search is much faster than using a FILTER command with a **regular expression** to search for a string. If you know or want to check whether your SPARQL endpoint supports this, you can set this variable to 'True'. There might be more effective ways to search for strings in a graph database than using those commands. There is even some full-text search support from DotNetRDF. However, we decided to stick to the SPARQL commands.

Graph2VR can load images when they are linked to a node. Therefore, it needs to know which kind of predicate encodes images. The predicates used for this are also defined in the settings singleton and might be adapted.

Limitations

Graph2VR (version 1.0) does not yet offer all SPARQL keywords. Although DotNetRDF would support them, common keywords like "Count", "Optional", "Filter", "Group By", "Bind", "Union", "Minus", or "Service" are not yet accessible via the Graph2VR menu.
Please note that Graph2VR (version 1.0) is a prototype and still has some (known) bugs. Please refer to our GitHub website (https://github.com/molgenis/Graph2VR) for further details and more recent information.

References

[1] GitHub, *dotnetrdf/dotnetrdf*, https://github.com/dotnetrdf/dotnetrdf/wiki/UserGuide-Querying-With-SPARQL **2020** (12.08.2020).

[2] *Sidequest*, https://sidequestvr.com/.

[3] C. Batchelor, C. Y. A. Brenninkmeijer, C. Chichester, M. Davies, D. Digles, I. Dunlop, C. T. Evelo, A. Gaulton, C. Goble, A. J. G. Gray, P. Groth, L. Harland, K. Karapetyan, A. Loizou, J. P. Overington, S. Pettifer, J. Steele, R. Stevens, V. Tkachenko, A. Waagmeester, A. Williams, E. L. Willighagen, in *The Semantic Web - ISWC 2014* (Eds: P. Mika et al.), Springer International Publishing. Cham **2014**.