





GAV-VR: An Extensible Framework for Graph Analysis and Visualisation in Virtual Reality

W. Kerle-Malcharek¹  and S. P. Feyer¹  and F. Schreiber^{1,2}  and K. Klein¹ 

¹University of Konstanz, Germany

²Monash University, Australia

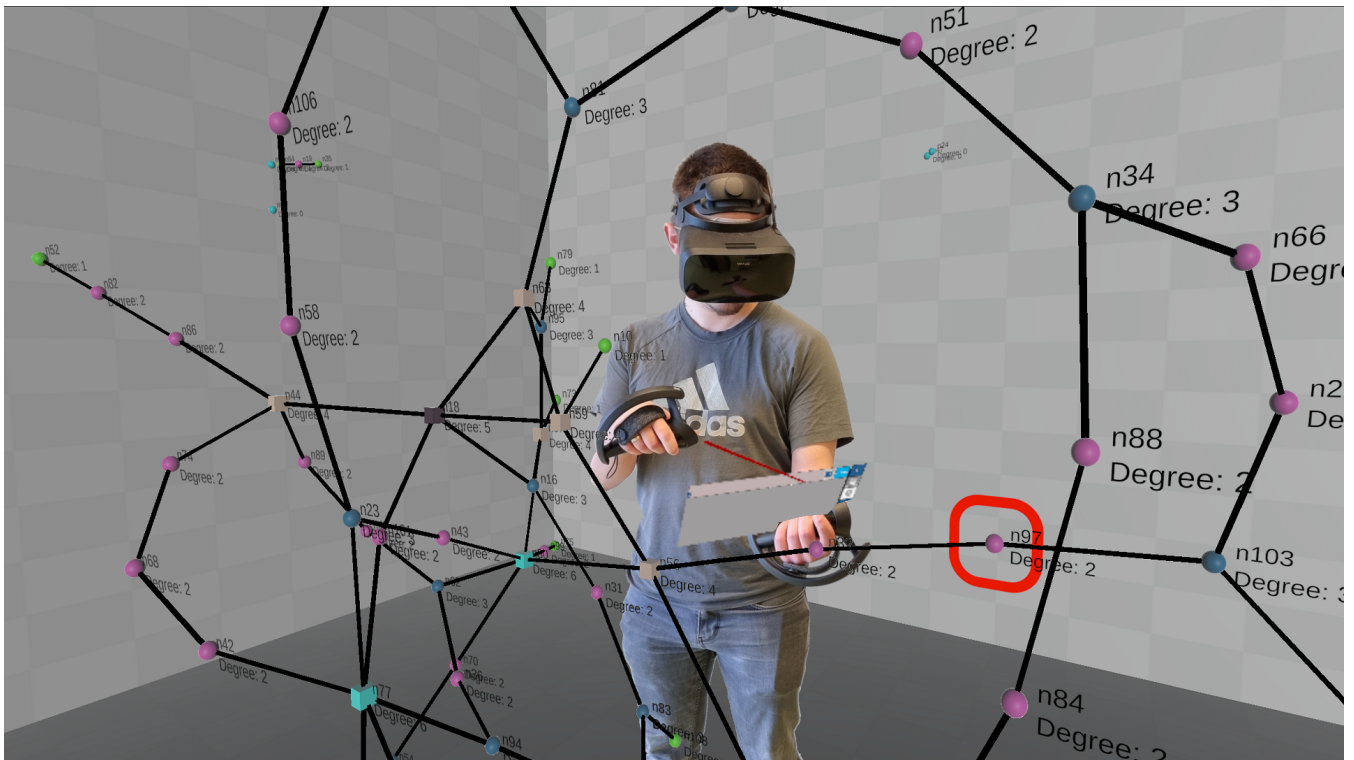


Figure 1: The framework allows the user to specify the mapping of data characteristics to visual variables. In this picture, color coding is used to map the vertex degree and vertices above a certain degree threshold have a cubic shape. The representation of the environment can be altered as well. Here, the walls were chosen to have a chequered pattern.

Abstract

The investigation of interactive graph visualisation in stereoscopic 3D has recently gained increasing attention due to promising initial results and the broad availability of required hardware such as Virtual Reality (VR) headsets. While various software frameworks and libraries for (interactive) graph visualisation in 2D exist, there is a lack of corresponding frameworks supporting VR. This hampers the exploration of design choices and the comparison of approaches, especially for the benefits of virtual environments, slows down the development of novel methods, and requires additional effort from researchers to perform these investigations. We present GAV-VR, a framework for interactive visualisation and analysis of graphs in VR. GAV-VR has a modular architecture to support easy integration of visualisation and analysis method implementations. In this work, we elaborate on the framework's architecture and showcase possible use cases.

CCS Concepts

• **Human-centered computing** → **Visualization toolkits; Virtual reality;**

1. Introduction

Interactive visualisation of graphs in stereoscopic 3D - such as in virtual reality (VR) environments - has shown the potential to support the understanding of a graph's structure and to facilitate analysis tasks in a variety of settings and applications [CDK*16, GPK12, KS14, KFS*22, KEK*21, SS17, WM05]. The large design space for graph representation in 3D and corresponding interaction provides ample opportunities for improvement of the analysis process. The exploration of this potential through the investigation of visualisation and interaction methods as well as of virtual environment designs thus has received a tremendous increase in research activity recently, highlighting benefits and disadvantages alike [DCW*18, HPY23, JJS*22, SKA22, SWKA19, FPK*23b]. The design, implementation, and evaluation of visualisation and interaction approaches for VR and their comparison to established 2D approaches are facilitated by the broad availability of high-quality VR hardware. Software frameworks and libraries for graph visualisation in 2D exist and allow quick prototyping and comparison of methods, such as OGDF [CGJ*13], Graphviz [EGK*02], Tulip [AAB*17], Vanted [JKS06] or Cytoscape [SMO*03]. However, there are little corresponding software frameworks for interactive graph visualisation in VR. Rapid prototyping and the wish for experiment reproducibility and replicability require the availability of flexible and free open-source implementations.

To fill this gap, we present *Graph Analysis and Visualisation in VR* (GAV-VR), a framework for interactive visualisation and analysis of graphs in VR. GAV-VR has a modular architecture regarding VR-HMDs, data formats, analysis and visualisation methods and supports the easy integration of new functionalities. It provides the basis for prototyping and method comparison and it also facilitates the interactive analysis process.

Immersive environments – in particular VR – support stereoscopic 3D visualisation of graphs, user tracking, (spatial) immersion of the analyst, and direct interaction like grabbing, rotating and moving objects around the scene. A major research challenge is to investigate how these features can be utilised for designs of graph analysis environments in order to provide a better understanding of the data, faster and higher quality analysis results, and better memorability.

With GAV-VR, we aim to support these investigations by providing a framework that spares the researcher from (re)implementing basic functionality from scratch. We assume that with GAV-VR researchers and analysts will be able to focus on their core tasks such as the implementation of new methods, comparison of approaches, and data analysis.

Contribution. We provide GAV-VR to the community, an extensible framework and interactive application for graph visualisation and analysis in VR. GAV-VR allows easy prototyping of graph algorithms, visualisations, layout methods and environments by a modular architecture. Pre-implemented modules of GAV-VR already contain common algorithms and methods for graph analysis, graph layouts, navigation and environments to provide a low usage threshold and enable developers and analysts to focus on their core tasks. GAV-VR is fully embedded in the Unity C# environment, reducing the amount of additional software required,

and supports all VR-HMDs which are supported by SteamVR. Currently, GAV-VR and its documentation are available on the RDM system Zenodo [KMFKS23] and will be moved to a dedicated website after publication. In this repository, GAV-VR is available as a Unity project and as a standalone build as an executable, which is configured to operate with Valve Index controllers.

To our knowledge, the approach of GAV-VR providing a framework for researchers and developers to quickly prototype use-case specific environments while maintaining a low availability threshold in terms of software and hardware is unique and leads to significant time-saving. We believe that this framework will be an asset to the community.

2. Related Work

There are many interactive systems for graph visualisation, which support various use-cases and applications in classical 2D desktop setups, e. g. [AAB*17, BHJ09, EGK*02, JKS06, SMO*03], partially also with 3D projections [FHP*22]. Furthermore, there are several libraries and frameworks that provide graph algorithm and method implementations [CGJ*13, EGK*02]. More recently, access to and support for stereoscopic 3D visualisation (S3D), including VR, has increased considerably [LRL*21, PMI*21, RM20]. After a long period of hesitation in investigations of the effectiveness of 3D graph visualisation, the wide availability of high-quality hardware with S3D capabilities sparked a series of research on its benefits and disadvantages. There are publications that provide initial evidence for increased task performance in virtual reality for a variety of tasks like navigation, visualisation or graph-related tasks e. g. [BVD19, CDK*16, GPK12, HPY23, JJS*22, KFS*22, KEK*21, SKA22, SAK*21, VSVDZS*10, WF94, WM05, WML94]. In this research it is a broad consensus that the proper utilisation of S3D will provide improvements in various aspects of graph analysis regarding task performance, understanding of structure, and perception. Consequently, frameworks supporting S3D also encourage further research on those topics.

There are little graph-related frameworks supporting S3D. CellexalVR [LRL*21] is a VR-based tool for graph visualisation and analysis of single-cell gene expression data. The focus of CellexalVR is on specialised graphs for cell selection and visibility, but not on general graphs. VRNetzer [PMI*21] utilises VR to view and analyse 3D graphs with a focus on the exploration and analysis of large graphs. VRNetzer provides an underlying database for the graphs, a Python interface for the analysis, a VR environment driven by Unreal Engine [Epi19], and a web interface to administer the analysis methods. Therefore, VRNetzer could also be seen as a hybrid between a desktop and a VR environment.

In contrast to VRNetzer, GAV-VR is a framework to develop tools similar to VRNetzer with the help of modules of various types (see Section 3.2) to fit needs related to specific research questions. Therefore, our framework is not limited to the purpose of graph analysis. Extending GAV-VR with modules, and thus functionality, is simpler compared to VRNetzer, as it only requires the implementation of an abstract class of the corresponding module. Instead of using different platforms for analysis and visualisation, GAV-VR is completely embedded in a Unity C# environment, making it easy to set up, use and extend.

The novelty of GAV-VR lies in its goal to be an easily extensible framework for graph exploration and analysis in virtual reality that facilitates the development of research question-specific functionality for an extensive range of graph-related domains.

3. The GAV-VR System

GAV-VR's concept considers two types of users: *analysts* and *contributors*. As this framework is considered to be an aid in developing environments for graph analysis and visualisation, oftentimes a contributor will have to switch to an analyst role and back to verify the validity of their newly added content. For our system design, we considered the two roles as distinct, to address the different requirements they have, respectively.

Analysts use the system as-is. They want to use it to investigate, explore, or analyse graphs with a given set of features and methods. They might also want to investigate the impact of available methods or features in the context of graphs and virtual reality. *Analysts* should be able to use our framework without writing code. Therefore, we provide a ready-to-use build. This build includes a set of pre-implemented features (see Section 3.2.1 and 3.2.2) ready to be applied on graphs, and a small set of example graphs for an easy start.

Contributors enrich GAV-VR with features with the help of the modular system provided by the framework. They develop and design with the help of our systems modules to provide new features, methods, or visualisations to *analysts*. *Contributors* benefit from the fact that GAV-VR is completely embedded in a Unity C# environment. It utilises a combination of abstract classes, pre-defined routines and data structures to ease the integration of new features without compromising the general functionality of the whole environment. Therefore, no complete understanding of the framework is required to extend the framework with new features. The knowledge about how the interfaces of GAV-VR work and basic programming skills are sufficient to implement basic new features.

3.1. Architecture

The software architecture of GAV-VR consists of two major components: the *core* and the *modules*, compare Fig. 2. The *core* of GAV-VR provides the core functionality and manages the framework's modular features. The *core* provides the user interface for *analysts* to interact with the virtual environment and the objects therein. Such objects in our context typically are graphs, edges, vertices, or the UI itself and can be extended by developers with e. g. position markers. The *core* also contains the controller classes which manage the module interfaces, input controls, movement, user interface, visualisations, file input and output, or the scene in general.

The *modules* represent the second major component of GAV-VR's architecture and are the set of instances that utilise the different module interfaces. They serve as the entry point for contributors. *Modules* provide the abstract classes and interfaces required to declare new instances of a majority of our modules. The others are file-based modules, see Section 3.2.2. Furthermore, *modules* are

the part of GAV-VR *contributors* are implementing to add new features. Those *modules* are interchangeable and can be made accessible to *analysts*. Instances of those *modules* can be self-implemented or just included as existing libraries. Consequently, modular content can be shared between contributors and used in individual configurations of the framework. The instances of *modules* are automatically integrated into the UI by the *core*, from where they can be used by a ray-casting interaction approach. The framework offers a set of prepared modules, to ensure basic functionality, and also serve as example implementation.

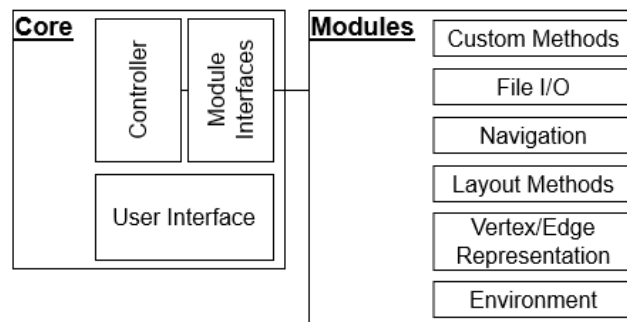


Figure 2: Basic architecture of GAV-VR: core and modules. The core handles the automatism such as loading modules or the UI, and the modules are the custom content provided by contributors.

3.2. Types of Modules

Modular content describes a set of features that use the framework's pre-defined classes and routines. Those *modules* can be analysis methods, layout methods, visualisations, environments, interactions, file input and output or representations. There are two types of *modules*: script-based *modules* (Section 3.2.1) and file-based *modules* (Section 3.2.2).

3.2.1. Script-Based Modules

Script-based *modules* make use of the provided C# abstract classes and interfaces and typically represent methods that either alter graphs in some way, produce analysis results, initiate interaction, or read/write files to and from the system. The framework provides a variety of script-based *modules* which give contributors plenty of ways to enrich the framework. The *Pipe* abstract class is used to translate outputs of parsers to a format that the framework can understand. With *pipes*, *contributors* can include various parsers to the framework to be able to load graphs from any required file format. A *pipe* has only a small amount of functions to ease its usage for *contributors*. If multiple pipes for the same file format are available, the UI will ask the analyst to choose from the different options when trying to parse a graph of the corresponding format. GAV-VR natively contains a reader for graphml files and csv files as edge lists. *Pipes* are extensions of the file input interface, the *IReaderModule*. If different reader classes are required, this interface can be used to realise them. The file output is realised using the *IWriterModule* and is used for storing information in the local file system.

An implementation of the *IWriterModule* is the *GraphWriter* class which can be used to implement new file formats to store graphs. We provide a simple writer to store graphs as edge lists in CSV format, and also for our own GAV-VR file format, see Section 3.5. The *IWriterModule* is also used to store results of analysis methods to the local file system, if required.

GAV-VR offers different movement modes. This is also a module that can be extended by contributors through the respective interface class which lets them define which controller button will trigger which action. This mapping eases the introduction of new movement schemes such that developers do not need to have a deep understanding of how the used SteamVR plugin works. GAV-VR comes with two instances for this module, which is continuous motion and one-handed flying, based on the work of Drogemuller et al. [DCW*18]. The movement mode can be chosen in the options tab of the UI. If another brand of controllers has to be supported, the *contributor* can adjust the setup through the SteamVR input plugin.

Custom Methods is a script-based module, which is used for multi-purpose methods that are intended to be applied to graphs. They range from visualisation manipulation of the graphs to analysis methods. We implemented a set of example functions for this module: a version of the Ortmann-Brandes counting triangles framework [OB14] (see Fig. 6), calculating the degree distribution, finding the shortest path between two selected vertices using Dijkstra's algorithm [Dij59] and animating the process, and an animated version of the Fruchterman Reingold algorithm [FR91].

Layout Methods can be considered a special case of the *custom methods*. It is a *module* which is applied to graphs at the last step of the parsing pipeline to modulate their embedding in their local space according to the implementation. GAV-VR comes with an example of a *Layout Method* which maps the vertices of a graph to a sphere.

3.2.2. File-Based Modules

File-based *modules* alter the representation of objects in the scene and usually require no code writing. Unity per default realises these representations in the form of *prefab* files. Those files store the values to visualise objects in a Unity scene and are used by GAV-VR, too. Integral items to represent in the context of node-link graphs are vertices and edges. GAV-VR allows *contributors* to include depictions of vertices and edges as required, by including the respective prefab files in the intended directories. The framework will recognise these files and include them in its pipeline, making them selectable in the UI. For vertices, we added spheres and cubes with different shaders. For edges, we added a cylindrical representation.

GAV-VR offers the possibility to view details about selected objects or graphs. This detailed information is shown in a dedicated UI segment in the form of a scrollable list. Contributors can directly influence what is shown and how it is shown in that UI segment with the corresponding file-based module *DetailWindow*. These can either be configured to just list details as a table, or be completely customised with pictures, logos, UI elements like buttons etc.. For this module, we hand out a simple table representation of information about selected vertices and graphs. The framework

also provides labels which can hover in proximity to vertices. These are generally considered to be used for showing vertices' details, or their names and are customisable, as well. We added two different alternatives for that, either just showing the name or showing the vertices' name and degree.

The virtual environment the analyst is located in, is a file-based *module*, too, thus can be altered by a *contributor*. The environment in this context means the virtual environment representation in which an analyst is situated and the corresponding class of that virtual environment is called *Room*. Our framework comes with three different rooms. It has a small grey plane with a black background, a small room with walls that have a tiled texture (see Fig. 1), and a big white room with a blue circle in its centre (see Fig 3).

3.2.3. Integrating Script-Based Modules

To integrate a script-based *module*, a contributor creates a new C# file in the project's directory, preferably in the dedicated project folder. The C# file has to implement a class that corresponds to the type of script-based *module* it is intended to extend, for example the *pipe* class. The implemented abstract class requires the contributor to define a small set of variables as well as the actual functionality. The variables of the abstract class mostly consist of booleans, which tell the system whether the method produces a report, or if they use the Unity internal *IEnumerator* class. The variables can also be strings that for example define which file extension a *pipe* is supposed to recognise. The *module* is then available through an automatically generated UI element (e.g. a button). The functionality of the module instance is executed when a *user* interacts with the respective UI component (see Fig. 3).

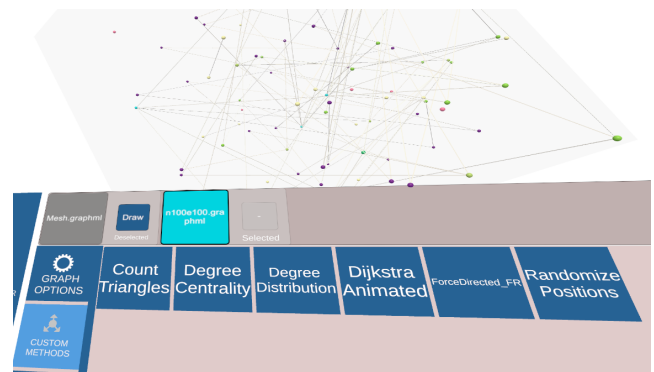


Figure 3: The UI which contains the available custom methods. These can be applied to a graph which has been selected. Selected graphs appear highlighted while the others are greyed out. The vertices have been coloured for better visibility.

3.2.4. Integrating File-Based Modules

The framework integrates custom objects that are stored in the defined project directories. These objects are detected by the framework upon start and then integrated into the UI (e.g. as drop-down list). This design ensures a convenient way to add additional appearances of supported elements, such as vertices or edges (see Fig. 4).

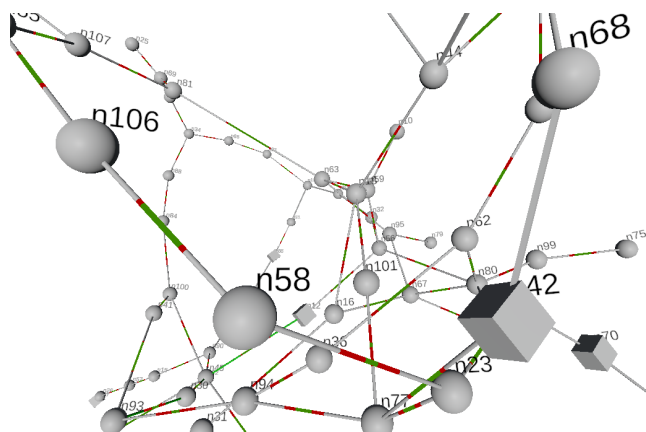


Figure 4: Different vertex and edge appearances in the same graph. The vertices in this graphic have either the shape of a sphere or the shape of a cube. The edges have a green-red colouring in their centre, which has a higher portion coloured green if their weight is bigger.

3.3. Visualisation

GAV-VR visualises graphs in node-link representation, consisting of vertices and edges. These vertices are static spheres per default but can be changed to look and behave differently. Edges are cylindrical and are adjusting their positions on demand, but can be changed to look and behave differently, too. The embedding of a graph can be obtained from the source files and then transformed, or calculated in a completely different way, with or without considering the pre-implemented layout.

The elements of graphs, e.g. vertices, can be coloured independently with dedicated colour attributes to produce mappings of different characteristics, such as weights for edges or vertex degrees. The UI is realised as a floating screen with an adjustable set of colours. The colours of it are divided into groups and can be influenced by contributors. The UI has a separate log window attached, which can optionally be used to view log output in situ to ease development for contributors. Environments, are located at the centre of each scene and typically consist of at least a floor to ease orientation for users. The default environment has a light grey square on the floor in its centre and a black background. The main tool for interaction, the raycast, changes its colour depending on what it points at: objects, UI, and everything else.

3.4. Interaction

The main interaction method for GAV-VR is the user interface. Users have the option to either attach the UI as a smaller version to the left controller or to let it hover as a big version in front of them. If the "big UI" mode is selected, the UI can also be displaced to be located in a more comfortable position. It is still attached to and following the user. The UI is interactable by using the raycast interaction paradigm which is a point-and-click type of interaction (see Fig.6) that mimics the way we typically interact with 2D interfaces.

The UI has multiple tabs like file browser, load graphs, reports,

or options which contain UI elements that can trigger specific actions e. g. buttons. The "File Browser" and "Load Graphs" tabs display graphs that have supported file formats. Further modules imported by contributors will automatically appear in dedicated tabs, like custom methods or vertex appearances which are located in a tab for graph customisation. Further interactions include the direct manipulation of graphs. Users can grab them with their controllers to rotate or translate them, as well as re-scaling them. The same holds for vertices, for example. When reaching into a graph, it will light up to indicate that it is being interacted with. Since it can cause problems while working with spacious graphs, GAV-VR also offers the possibility to create a cube agent for selected graphs (see Fig.5). Cube agents represent the graph that is currently selected and can be grabbed instead of the actual graph to rotate, translate, or rescale the graph itself with the same set of interactions.

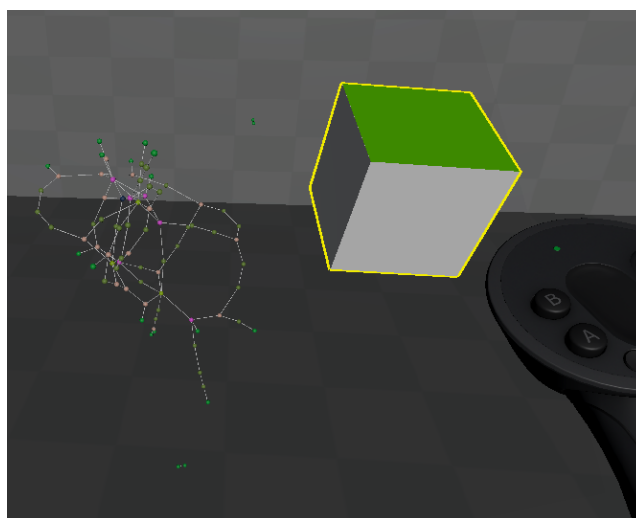


Figure 5: A cuboid which is the agent for the currently selected graph. Through hovering over it with the controller, the user can interact with the graph through the execution of the configured grab action. The cuboid lightens up in yellow when interacted with.

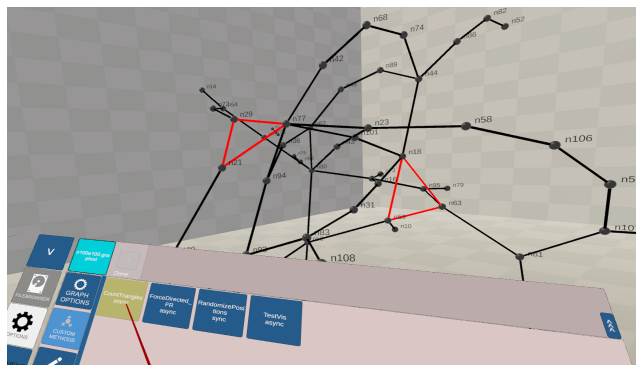


Figure 6: GAV-VR user interface (bottom). The raycast points at a button to apply the custom method count triangles to the graph. The detected triangles are highlighted in red.

3.5. Data Handling and Analysis

The majority of data traverses the *core* at some point. Because of this, it has a high responsibility in terms of data validation to ensure that no side effects are caused through module integration. For example, it validates the supported file extensions or the uniqueness of graphs. The *core* also controls the application of changed options. When the framework is loaded, the saved options are applied where the respective optional content has to be set. GAV-VR offers various options to customise the *analysts'* experience, such as the dropdown for available appearances of the environment the *user* is placed in or if a window with further information about selected graphs will be permanently displayed. Contributors can add new options by adding optional fields in the UIs options window to allow further customisation of the experience in the virtual environment. All these options can also be saved.

Additionally, saving and loading of graphs is done with the custom file format of GAV-VR: ".gavvr". The ".gavvr" file format has a human-readable text format. It contains only basic information with a focus on static attributes. This includes the position, colour, size of vertices, the width and weight of edges, and more. The purpose is to have a quickly iterable, simple-to-manipulate file format which contains all relevant information on the basic appearance of graphs and their content. Apart from saving and loading, graphs can also be copied. This functionality can be found in the *Graph Actions* tab and is one of the native methods which do not belong to the *modules*, but are part of the *core*. Besides copying, a graph can be deleted. Another useful functionality is the generation of a random graph for testing purposes with the build-in graph generator. To calculate graph measures (e.g. centralities) and other characteristics, contributors can resort to typical data structures and attributes of graphs and graph elements which are automatically maintained. These include adjacency lists or the in- and out-degree of vertices. Also, we added a Dijkstra SSSP algorithm for two selected vertices as well as a method to count connected components.

3.6. Performance

The framework has been developed and tested with a basic VR-ready system with an NVIDIA GeForce GTX1660 Ti graphics card, an AMD Ryzen 5 3400G processor and 16 GB RAM on a Windows 10 system together with an Oculus Rift S. Additionally, it has been tested successfully with the Meta Quest 2, the Valve Index, Varjo Aero, Varjo XR-2, and the HTC VIVE Pro 2. The visualisation of graphs as node-link diagrams in the framework's current state supports two different modes - the interactive and the non-interactive mode. In the interactive mode, all vertices can be grabbed, dragged around, and rotated interactively (i.e. by hand). Therefore, the graphs can actively be manipulated. This interaction mode requires objects to execute small routines at all times which has a toll on the performance, capping the maximum amount of objects in the scene (e.g. vertices or edges) for this hardware-minimal setup to roughly 2,000 objects to keep the number of frames per second (FPS) for a stereoscopic rendering above 60. The non-interactive mode combines the meshes of all objects of a graph if possible. Direct interaction with vertices is not possible in this mode, but the amount of objects that can be rendered with 60+ FPS

in this setup is increased to approximately 20,000. Better hardware will provide corresponding improvements in performance.

3.7. Licensing

The framework is licensed under GNU Affero General Public License V3, which allows free usage of the whole software as long as it is non-commercial. This holds for the framework's "as-is" state and does not include the different modular contents that can be added in the future.

4. Use Cases and Benefits

GAV-VR's use cases naturally have different manifestations, depending on which user role they refer to. *Analysts* face use cases where they do not add content to the framework. This includes but is not limited to, the exploration of a graph or answering questions in the form of a study where the questions refer to the graph(s) or the form of navigation of a user. A *contributor's* use cases focus on the addition of new modular content, the adaptation of existing content, or configuring the frameworks' environment for a particular use. Such cases could be the deployment of new ways how to navigate through the scene and new methods that analyse or alter graphs or representations of objects in the scene.

4.1. Graph Exploration

A basic use case of GAV-VR is that an *analyst* explores a graph or investigates how an algorithm impacts the layout of a graph. To explore a graph, the *analyst* uses the UI to open the file browser and selects the graph in question. After selecting the graph, the *analyst* parses the graph with the corresponding button. Once the graph is parsed, the graph is rendered into the scene. The *analyst* is interested to see the vertices with the highest degrees in the graph. First, the *analyst* activates the vertex labels, to see the names of the vertices. Then, the *analyst* applies a *custom method* which layouts the graph so that the higher degree vertices are more in the centre of the graph, while the lower degree vertices are pushed outside. Since the *analyst* is facing a high-density graph the *analyst* now initiates a degree-based colour mapping with random colours. While this is in process, the *analyst* wants to compare the result with an actual heat map which is based on colours. Therefore, the *analyst* creates a copy of the graph and places it next to its original. Then, the *analyst* applies a heat map colouring to the copy (see Fig.7). The *analyst* is now able to compare both graphs visually but decides that due to their high density, that the graphs have to be re-scaled. The *analyst* puts a controller into the graph and pushes a button on the controller to resize the graphs. Finally, the *analyst* has two graphs with an identical layout and up-scaled, with different colour mappings, both showing the desired outcome.

4.2. Developing Graph Algorithms

A researcher is about to develop a novel method to layout a graph in 3D space. For a better understanding of the algorithm, the researcher uses GAV-VR to visualise how the layout evolves over time. To do so, the researcher uses the *custom method* module. The researcher (who is now a *contributor*) opens the project, navigates

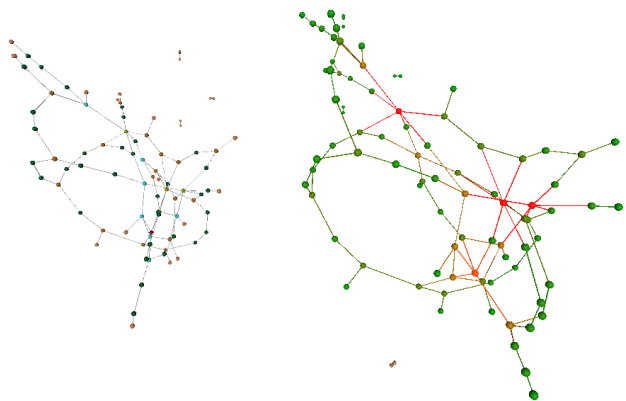


Figure 7: Two graphs are rendered next to each other to compare them. The right graph has a colour mapping ranging from red to green to indicate the degree of centrality.

to the respective folder and creates a new C# file. In this file, the researcher implements the *CustomMethod* abstract class. This requires the researcher to set variables, which specify if the method is shown and usable in the UI, if a report of the outcome should be created, how such a report should look like, and if the method is supposed to make use of the *IEnumerator* class of Unity, to allow an execution over time. The researcher decides, that no report is needed but an execution over time is required to be able to see the visualisation of the algorithm. After the researcher has finished the first version of the method, the researcher starts the system and tries the new method. The researcher (now in the role of an *analyst*) parses and places a graph which seems well-suited for testing purposes. Then, the researcher selects the graph and navigates to the *Custom Methods* panel which automatically has the new method integrated and triggers the execution of the new method (see Fig. 6). The researcher observes how the algorithm alternates the graph, which provides valuable insights for further developing of the algorithm. After several iterations the researcher shares the newly developed algorithm with a colleague by sending the C# file to that colleague, who can just paste it into the project and use it on own graphs.

4.3. Studies on Graphs in VR

Due to its high extensibility, GAV-VR is predestined to be used for studies on graphs in virtual reality. GAV-VR's wide range of *module* types allows high customisation of the framework, facilitating an extensive range of studies to conduct: impact of visual elements like the virtual environment, vertex and edge appearance, or comparison of layout methods, analysis methods or navigation methods.

As example, in 2022 an experiment was conducted with GAV-VR to investigate the effectiveness of 2D, 2.5D, and 3D layer arrangements of multilayer networks (a graph where the vertices are

located on layers) [MRA*21] in stereoscopic virtual reality settings, see Figure 8. This work has been published in IEEE Transactions on Visualization & Computer Graphics in 2023 [FPK*23a].

Prior to the experiment, graphs were externally generated in the graphml file format. Layers were contained as parent vertices for all vertices which are part of that layer. The *IReaderModule* read the graph from the file system and the corresponding *pipe* (see Section 3.2.1) *module* was then written to process these graph files. For the parsing process, the framework has been configured to use a *layout method module* which only re-scales the graph as the layout has already been calculated and integrated into the graphml file. Customised vertices and edges were provided by file-based modules. For instance, the edges between layers were coloured in yellow. Regarding the virtual environment, a custom 5x5 meters dark blue room was added via a file-based module. Routines for enclosing single layers in grey transparent boxes have been added as well as annotations of those layers with letters. Furthermore, a panel was added to a wall of the virtual environment to navigate through the experiment. On the panel, the current task was displayed, the rendering of the graph could be initiated, and the solution to the task was captured. The measurements of the experiment were logged and stored to the file system. To avoid unintentional interactions with the framework, the controller-based movement was disabled and the UI was hidden. Only pointing with the ray-cast, pressing GUI buttons and navigating through body motion were allowed. GAV-VR proved to be a suitable tool for conducting experiments for graphs in virtual reality in this experiment.

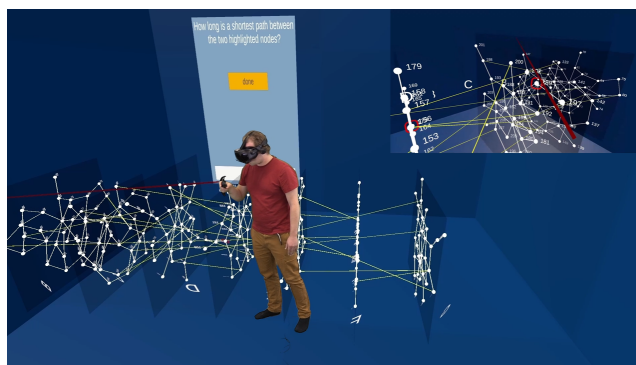


Figure 8: Study on multilayer network visualisations in VR. GAV-VR is extended by a panel at the wall to guide through the study, a custom room, layers as substructures of a graph including layer labels, specialised parser and pipes translated the multilayer network into the virtual environment

4.4. Benefits for Research

As described in Section 4.3, GAV-VR leads to a novel approach towards designing virtual spaces for studies, since researchers are spared the implementation and design of those to a considerable extent. In the field of S3D research, this can help in numerous applications, such as evaluating navigation techniques in VR. Drogemuller et al. [DCW*18] who evaluated different navigation paradigms, could have used GAV-VR as their foundation and expanded on it

by including their navigation paradigms and some of their own visualisations. Works like from Joos et al. [JJS*22] who investigated different information encodings in the form of networks could have benefitted from the possibility of including such encodings, without the need to design a whole graph structure, interaction or the environment. Other, more complex works like VRNetzer or CellexalVR [PMI*21, LRL*21], which support visualisation and analysis tasks on networks, could have benefitted from the possibility to prototype their ideas or use some of the core architecture to structure their networks and modules. Further exemplary fields which could benefit from our framework are the visualisation of interaction networks in the field of collective behaviour, researching user interfaces in 3D, how multi-modal representation impacts data understanding, or whether a problem which traditionally requires spatial information, such as molecular interactions, can be analysed more efficiently in VR.

5. Conclusion and Future Work

GAV-VR is designed as a framework that aims to remove the barriers for researchers to implement and perform graph visualisation and run graphs analysis algorithms in a customised VR environment. The design supports using the framework without programming as well as extending it to create an environment which fits specific needs connected to a research question. The architecture has a modular design in a plug-and-play fashion to allow further implementations to extend existing functionality. The modularity is realised with abstract classes and Unity's prefabs. To automatically integrate new methods into the framework as interactable UI components, those methods have to implement the respective abstract classes and to meet a small set of requirements. The prefabs are prefabricated objects which only have to be placed in the respective project folders to be available. Entire digital rooms can be built and different navigation concepts can also be integrated and provided as options to existing modules, which yields the potential to enable contributors to exchange implementations they made by simply sharing one file.

Since we intend to maintain GAV-VR as a long-term development we also plan to integrate contributions by the community into the main version of the framework. As an extension to that, we plan to provide further instances of our modules as well as new module types together with exemplary implementations. Such new modules will be alternative representations for graphs, for example. Furthermore, we plan to improve the performance with respect to the framerate as well as the usability from the perspective of developers and analysts. To improve the usability, input from the community will be collected in the form of a usability study, since such a study is currently missing for GAV-VR.

Additionally, we plan to integrate further features, such as support to conduct studies with GAV-VR on e.g. visualisation methods. The study support will come in the form of customisable and interactive UI elements, which can be used to use file I/O to upload question sheets and save answer sheets where answers can be connected to the UI or to objects in the virtual environment. The seamless integration of the Open Graph Drawing Framework [CGJ*13] as a source for graph and layout algorithms is planned, too.

Lastly, we plan on supporting multiple users in one session to

allow for collaborative graph analysis or multi-user studies either co-located or completely remote.

GAV-VR still has room for improvement, as this is the nature of such a framework. But still, the existing features provide a strong baseline for quickly building a range of different environments to test an idea, investigate a graph or even conduct a study. This makes GAV-VR a multipurpose tool to ease graph analysis and visualisation in virtual reality which we gladly want to share with the VR community.

Supplemental Materials

All supplemental materials are released under the GNU AGPL V3 license and can be accessed at Zenodo <https://doi.org/10.5281/zenodo.8289512> (version 2) for the persistent submission version. We provide the full documentation, an open-source project version and a ready-to-use build version.

Acknowledgments

We acknowledge funding by DFG, under Germany's Excellence Strategy – EXC 2117 – 422037984, and DFG project ID 251654672 – TRR 161

References

- [AAB*17] AUBER D., ARCHAMBAULT D., BOURQUI R., DELEST M., DUBOIS J., LAMBERT A., MARY P., MATHIAUT M., MELANÇON G., PINAUD B., RENOUST B., VALLET J.: TULIP 5. In *Encyclopedia of Social Network Analysis and Mining*, Alhaji R., Rokne J., (Eds.). Springer, 2017, pp. 1–28. URL: <https://hal.science/hal-01654518>. 2
- [BHJ09] BASTIAN M., HEYMANN S., JACOMY M.: Gephi: an open source software for exploring and manipulating networks. In *Proc. Internat. AAAI Conference on Web and Social Media* (2009), vol. 3(1), pp. 361–362. 2
- [BVD19] BÜSCHEL W., VOGT S., DACHSELT R.: Augmented reality graph visualizations. *IEEE Computer Graphics and Applications* 39, 3 (2019), 29–40. 2
- [CDK*16] CORDEIL M., DWYER T., KLEIN K., LAHA B., MARRIOTT K., THOMAS B. H.: Immersive collaborative analysis of network connectivity: CAVE-style or head-mounted display? *IEEE Transactions on Visualization and Computer Graphics* 23, 1 (2016), 441–450. 2
- [CGJ*13] CHIMANI M., GUTWENGER C., JÜNGER M., KLAU G. W., KLEIN K., MUTZEL P.: The Open Graph Drawing Framework (OGDF). In *Handbook on Graph Drawing and Visualization*, Tamassia R., (Ed.). Chapman and Hall/CRC, 2013, pp. 543–569. 2, 8
- [DCW*18] DROGEMULLER A., CUNNINGHAM A., WALSH J., CORDEIL M., ROSS W., THOMAS B.: Evaluating navigation techniques for 3D graph visualizations in virtual reality. In *Proc. Internat. Symp. Big Data Visual and Immersive Analytics (BDVA)* (2018). 2, 4, 7
- [Dij59] DIJKSTRA E. W.: A note on two problems in connexion with graphs. *Numerische mathematik* 1, 1 (1959), 269–271. 4
- [EGK*02] ELLSON J., GANSNER E., KOUTSOFIOS L., NORTH S. C., WOODHULL G.: Graphviz—open source graph drawing tools. In *Proc. Graph Drawing (GD 2001)* (2002), Springer, pp. 483–484. 2
- [Epi19] EPIC GAMES: Unreal Engine, 2019. last accessed 2023-04-25. URL: <https://www.unrealengine.com>. 2

- [FHP*22] FREEMAN T. C., HORSEWELL S., PATIR A., HARLING-LEE J., REGAN T., SHIH B. B., PRENDERGAST J., HUME D. A., ANGUS T.: Graphia: A platform for the graph-based visualisation and analysis of high dimensional data. *PLoS Computational Biology* 18, 7 (2022), e1010310. 2
- [FPK*23a] FEYER S. P., PINAUD B., KOBOUROV S., BRICH N., KRONE M., KERREN A., BEHRISCH M., SCHREIBER F., KLEIN K.: 2d, 2.5d, or 3d? an exploratory study on multilayer network visualisations in virtual reality. *IEEE Transactions on Visualization and Computer Graphics* (2023), 1–11. doi:10.1109/TVCG.2023.3327402. 7
- [FPK*23b] FEYER S. P., PINAUD B., KOBOUROV S. G., BRICH N., KRONE M., KERREN A., BEHRISCH M., SCHREIBER F., KLEIN K.: 2D, 2.5D, or 3D? an exploratory study on multilayer network visualisations in virtual reality. *IEEE Transactions on Visualization & Computer Graphics*, 01 (2023), 1–11. doi:10.1109/TVCG.2023.3327402. 2
- [FR91] FRUCHTERMAN T. M., REINGOLD E. M.: Graph drawing by force-directed placement. *Software: Practice and Experience* 21, 11 (1991), 1129–1164. 4
- [GPK12] GREFFARD N., PICAROUGNE F., KUNTZ P.: Visual community detection: An evaluation of 2D, 3D perspective and 3D stereoscopic displays. In *Proc. Graph Drawing (GD 2011)* (2012), pp. 215–225. 2
- [HPY23] HUANG H. H., PFISTER H., YANG Y.: Is embodied interaction beneficial? a study on navigating network visualizations. *Proc. Information Visualization* (2023), 14738716231157082. first published online. 2
- [JJS*22] JOOS L., JAEGER-HONZ S., SCHREIBER F., KEIM D. A., KLEIN K.: Visual comparison of networks in VR. *IEEE Transaction on Visualization and Computer Graphics* 28, 11 (2022), 3651–3661. 2, 8
- [JKS06] JUNKER B. H., KLUKAS C., SCHREIBER F.: Vanted: a system for advanced data analysis and visualization in the context of biological networks. *BMC Bioinformatics* 7, 1 (2006), 1–13. 2
- [KEK*21] KUZNETSOV M., ELOR A., KURNIAWAN S., BOSWORTH C., ROSEN Y., HEYER N., TEODORESCU M., PATEN B., HAUSSLER D.: The immersive graph genome explorer: navigating genomics in immersive virtual reality. In *Proc. IEEE Intl. Conf. on Serious Games and Applications for Health (SeGAH)* (2021), pp. 1–8. 2
- [KFS*22] KRAUS M., FUCHS J., SOMMER B., KLEIN K., ENGELKE U., KEIM D. A., SCHREIBER F.: Immersive analytics with abstract 3D visualizations: A survey. *Computer Graphics Forum* 41, 1 (2022), 201–229. 2
- [KMFKS23] KERLE-MALCHAREK W., FEYER S. P., KLEIN K., SCHREIBER F.: GAV-VR: an extensible framework for graph analysis and visualisation, 2023. doi:10.5281/zenodo.8289512. 2
- [KS14] KERREN A., SCHREIBER F.: Why Integrate InfoVis and SciVis? An Example from Systems Biology. *IEEE Computer Graphics and Applications* 34, 6 (2014), 69–73. 2
- [LRL*21] LEGETH O., RODHE J., LANG S., DHAPOLA P., WALLERGÅRD M., SONEJI S.: CellexVR: A virtual reality platform to visualize and analyze single-cell omics data. *iScience* 24, 11 (2021), 103251. 2, 8
- [MRA*21] MCGEE F., RENOUST B., ARCHAMBAULT D., GHONIEM M., KERREN A., PINAUD B., POHL M., OTJACQUES B., MELANÇON G., VON LANDESBERGER T.: *Visual Analysis of Multilayer Networks*. Morgan & Claypool Publishers, 2021. doi:10.2200/S01094ED1V01Y202104VIS012. 7
- [OB14] ORTMANN M., BRANDES U.: Triangle listing algorithms: Back from the diversion. In *2014 Proc. Sixteenth Workshop on Algorithm Engineering and Experiments (ALENEX)* (2014), pp. 1–8. 4
- [PMI*21] PIRCH S., MÜLLER F., IOFINOVA E., PAZMANDI J., HÜTTER C. V. R., CHIETTINI M., SIN C., BOZTUG K., PODKOSOVA I., KAUFMANN H., MENCHE J.: The VRNetzer platform enables interactive network analysis in Virtual Reality. *Nature Communications* 12, 1 (2021). 2, 8
- [RM20] ROSSI L., MAXIM L.: Multilayer network exploration tool in virtual reality, 2020. URL: <http://leonardmaxim.com/mnetvr/index.html>. 2
- [SAK*21] SORGER J., ARLEO A., KÁN P., KNECHT W., WALDNER M.: Egocentric network exploration for immersive analytics. *Computer Graphics Forum* 40, 7 (2021), 241–252. 2
- [SKA22] SCHRÖDER K., KOHL S., AJDADILISH B.: Netimmerse - evaluating user experience in immersive network exploration. In *Digital Human Modeling and Applications in Health, Safety, Ergonomics and Risk Management. Health, Operations Management, and Design* (2022), Duffy V. G., (Ed.), Springer, pp. 391–403. 2
- [SMO*03] SHANNON P., MARKIEL A., OZIER O., BALIGA N. S., WANG J. T., RAMAGE D., AMIN N., SCHWIKOWSKI B., IDEKER T.: Cytoscape: a software environment for integrated models of biomolecular interaction networks. *Genome Research* 13, 11 (2003), 2498–2504. 2
- [SS17] SOMMER B., SCHREIBER F.: Integration and virtual reality exploration of biomedical data with CmPI and VANTED. *it – Information Technology* 59, 4 (2017), 181–190. 2
- [SWKA19] SORGER J., WALDNER M., KNECHT W., ARLEO A.: Immersive analytics of large dynamic networks via overview and detail navigation. In *Proc. IEEE Internat. Conf. on Artificial Intelligence and Virtual Reality (AIVR)* (2019), pp. 144–1447. 2
- [VSVDZS*10] VAN SCHOOTEN B. W., VAN DIJK E. M., ZUDILOVA-SEINSTR A., SUINESIAPUTRA A., REIBER J. H.: The effect of stereoscopy and motion cues on 3D interpretation task performance. In *Proc. Internat. Conf. on Advanced Visual Interfaces* (2010), pp. 167–170. 2
- [WF94] WARE C., FRANCK G.: Viewing a graph in a virtual reality display is three times as good as a 2D diagram. In *Proc. IEEE Symposium on Visual Languages* (1994), IEEE, pp. 182–183. 2
- [WM05] WARE C., MITCHELL P.: Reevaluating stereo and motion cues for visualizing graphs in three dimensions. In *Proc. Symposium on Applied Perception in Graphics and Visualization* (2005), pp. 51–58. 2
- [WML94] WICKENS C. D., MERWIN D. H., LIN E. L.: Implications of graphics enhancements for the visualization of scientific data: Dimensional integrality, stereopsis, motion, and mesh. *Human Factors* 36, 1 (1994), 44–61. 2