Overview of Software Visualization in Virtual Reality

David Valverde Garro
School of Computer Engineering
Costa Rica Institute of Technology

***Abstract*—Software visualization is a concept related to viewing graphical representations of many different aspects that constitute a code project. In recent years, several authors have conducted research that aims to study the potential applications of virtual reality to this particular type of visualization. This paper aims to generate an overview of this sub-field, identifying common methods, data and the benefits of VR visualizations. We identified several benefits and opportunities of visualization in VR related to the concept of affordances. Among our results we found the use of metaphors tends to be common practice for generating an understanding of structural aspects. We also note the categorization of data along three main classes depending on when they are observable.**

***Index Terms*—Software visualization, Virtual reality, Metaphors**

I. Introduction

SOFTWARE or program visualization is an area of information visualization concerned with visualizing actual program code or data structures in either static or dynamic form [1]. Such tools are necessary given that the raw format of code, the one that is human-readable at least, comes in the form of large text documents. These documents require a significant investment of time in order to gain an understanding of how the program is organized and structured.

As mentioned by Dimara and Perin [2], the traditional keyboard and mouse desktop setup is the de facto environment for visualization software. However, new immersive technologies are potential candidates for integration with the current paradigm. Especially when it comes to 3D visualizations as shown by Bach et al.'s study [3]. One such immersive technology is virtual reality (VR) which seeks to have the user observe and possibly interact with a virtual world which completely overlaps the physical one. Currently a common means of achieving this is via head-mounted displays (HMDs) which use two lenses and a stereoscopic image to achieve an illusion of depth.

The following paper provides an overview of the research done in recent years in the field of software visualization using VR. We seek to answer a series of research questions that may help in forming an introductory view of the area. Our research questions are as follows:

- Why is software visualization important and what are some common goals?
- Which advantages are gained by visualizing in VR?
- Which types of data are interesting to visualize? How are they organized?
- What are some common VR software visualization methods found in the literature?

II. Related work

In our research we found plenty of review or survey works on different aspects of software visualization as a whole. Teyseyre and Campo [4] synthesized an overview of the research on three-dimensional software visualizations. They identified several areas of improvement, among which are an interest in new display technologies and increased availability of 3D displays.

Shahin et al. [5] performed a systematic review with the objective of identifying and classifying techniques in this area. Their results indicate the presence of four main classes, in which one can find the graph-based and metaphor-based techniques, with the former being more popular than the latter. Further ahead, another systematic review by Mattila et al. [6] found that the understanding of structure, behavior, and evolution of software are by far the most common topics of research in this area. Merino et al.'s [7] review focused on evaluation methods of software visualization systems. They concluded that there is a lack of solid evaluation methods that avoid pitfalls such as unclear goals, unclear definitions, and inconsistent tasks, among others. This same conclusion is also found in Bedu et al.'s [8] review of secondary reseach on this field.

As we mentioned previously, there seems to be a reasonable amount of survey and review papers that study the general area of software visualization. However, to the best of our knowledge, there hasn't been an effort to generate a review in the context of virtual reality so far.

III. Methodology

We followed a simple procedure to gather the literature used in this review. Since we aim to only give a brief view of the field, we limited ourselves to a small number of publications. We also decided to only include documents related to active research such as journal articles and conference proceedings. We employed the search terms found in table I, where the documents returned by the query must contain at least one of the terms from each column. This way we made sure to only take into account papers that are directly related to visualization

TABLE I
SEARCH TERMS USED WHILE GATHERING DOCUMENTS FOR THIS REVIEW.

	Software	Visualization	Virtual Reality
Search Terms	Software	Visualization	Virtual reality
	Software architecture	Exploration	VR
	Program	Visual analysis	HMD
	Source code		
	Dependency graph		
	Object-oriented		

of software in virtual reality. The searched databases include IEEE Xplore [1], ScienceDirect [2], and Springer Link [3].

Once the initial list of papers was compiled, we proceeded to filter them out according to the following exclusion criteria:

- The paper was published prior to 2010.
- The paper had less than 10 citations.
- The paper had less than 15 references.

The 12 year threshold was chosen due to the recent popularization of consumer VR devices such as the Oculus Rift [4], which employs HMD technology. Meanwhile, the citation and reference requirements were chosen to ensure a minimum level of relevance and quality of the selected research. Table II shows the list of selected papers after applying the exclusion criteria.

IV. RESULTS

A. Importance and uses of software visualization

The abstract, modularized, and text-based nature of source code, while necessary, makes it harder to understand upon first inspection [11]. As such, certain tasks will be inherently harder for someone who is not already familiar with the codebase. Elliott et al. [14], mention the example of code reviews, where superficial issues such as convention violations tend to be reported due to a lack of understanding. The ever-growing size of codebases the average programmer works with also serves to worsen this problem [11], [12].

Software visualization aims to serve as a tool to aid developers in day-to-day tasks. One of the most commonly stated tasks among the reviewed literature is software or program comprehension [9], [10], [12], [13], [15]. Other frequent tasks which are related to or require software understanding are analysis [10], [11], maintenance [13], [15], exploration and familiarization [11]. With this in mind, the need for this specific type of visualizations becomes clear.

B. Advantages of visualization in VR

Since virtual reality is an inherently 3D medium, it is often used as an environment for 3D visualizations. This alone brings some important benefits as mentioned by Vincur et al. [10]. Three-dimensional visualization spaces allow for better identification, recall, and understanding capabilities. Vincur et al. [10] also list the leverage of spatial memory as an advantage, which is one of the affordances of VR studied by Elliott et al. [14].

Gaver [17] describes affordances as properties of the world that are compatible for interaction. Meanwhile, Elliott et al. [14] contextualize this concept in virtual reality as a device that leverages human cognitive abilities. Focusing on said context, [14] synthesizes and describes three main affordances provided by this medium: spatial cognition, manipulation and movement, and feedback. Spatial cognition refers to the activation of neurons related to physical navigation when moving in a VR space as well as the improved depth perception from stereoscopic images. This suggests that current VR systems should better engage spatial memory since they incorporate such features. The manipulation and movement affordance relates to the improvement of perception, understanding, recall, retention, and location when manipulating physical objects. Since typical VR input methods simulate this quite closely, this affordance is relevant. Lastly, the feedback affordance corresponds to how quickly the results of an action are perceived. While this is not something exclusive to VR, the immersive nature of visualization in this environment makes for an improved perception of changes.

To further exemplify the benefits of the spatial cognition affordance, we take can look at Vincur et al. [10] where they mention several advantages of VR visualization. These include: reduced disorientation due to the match between a 3D space and a 3D input method, better perception of the relative size of objects, better usage of spatial memory when navigating, an elevated sense of presence, bigger structures can be visualized at a time, a better understanding of structures due to manipulation, and the possibility for virtual collaborative spaces. This view on collaborative spaces is also shared by Elliott et al. [14] and Merino et al. [15].

C. Data of interest

As one might expect the main source of data to create software visualizations is the source code. More specifically, one wishes to visualize certain aspects of the program and its structure. Vincur et al. [10], and Schreiber and Misiak [12] conveniently categorize said aspects into static, dynamic, and evolutional. Static aspects correspond to those that can be observed by just looking at the source code, while dynamic aspects appear during execution. Meanwhile, evolutional aspects refer to those that manifest as the software is being built, maintained, and modified through time. Table III summarizes the different software aspects found in our review according to the previously mentioned categories.

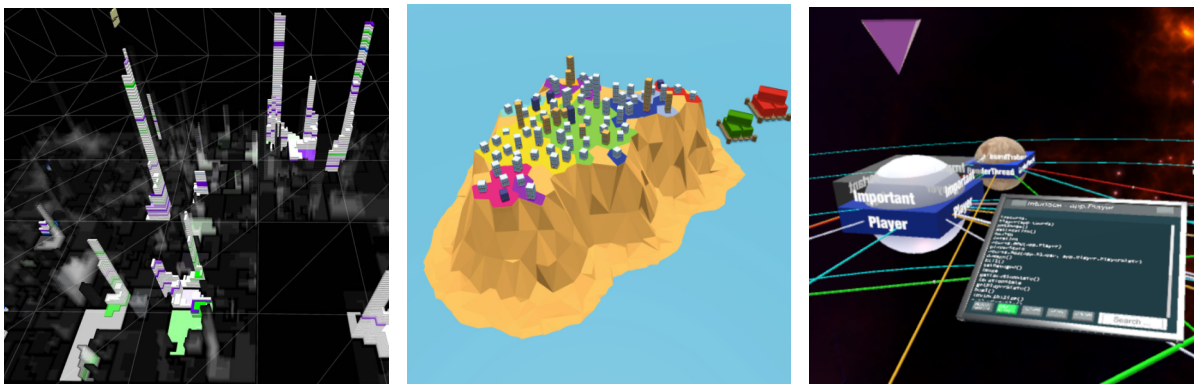
D. Methods found in literature

The following section summarizes the methods used for software visualization used employed in the literature under review. We describe said methods along two axes: metaphors and interaction or navigation.

[1]https://ieeexplore.ieee.org
[2]https://www.sciencedirect.com
[3]https://link.springer.com
[4]https://www.kickstarter.com/projects/1523379957/oculus-rift-step-into-the-game

TABLE II
PAPERS INCLUDED IN OUR LITERARY REVIEW AFTER FILTERING.

	Title	Authors	Year	No. Citations
[9]	Exploring software cities in virtual reality	Fittkau et al.	2015	119
[10]	VR City: Software Analysis in Virtual Reality Environment	Vincur et al.	2017	61
[11]	Gamified Virtual Reality for Program Code Structure Comprehension	Oberhauser and Lecon	2017	20
[12]	Visualizing Software Architectures in Virtual Reality with an Island Metaphor	Schreiber and Misiak	2018	14
[13]	On the use of virtual reality in software visualization: The case of the city metaphor	Romano et al.	2019	26
[14]	Virtual Reality in Software Engineering: Affordances, Applications, and Challenges	Elliott et al.	2015	58
[15]	CityVR: Gameful Software Visualization	Merino et al.	2017	71
[16]	Metaphors for software visualization systems based on virtual reality	Averbukh et al.	2019	10



(a) City (b) Island (c) Universe

Fig. 1. Different metaphors used to represent program structures in VR visualizations. **(a)** shows the city metaphor by Vincur et al. [10], **(b)** shows the island metaphor by Schreiber and Misiak [12], and **(c)** shows the universe metaphor implemented by Oberhauser and Lecon [11].

TABLE III
CATEGORIZATION OF COMMON SOFTWARE ASPECTS FOUND IN THE REVIEWED LITERATURE.

	Category		
	Static	**Dynamic**	**Evolutional**
Aspects	Packages	Object instances	Changes to code
	Classes	Communication	Authors
	Code fragments	Execution trace	Commits
	Code smells	Metrics	
	Dependencies	Usage frequency	
	Hierarchies		
	Lines of Code		
	Methods		
	Metrics		
	Modules		
	Sub-packages		

1) Metaphors: By far the most recurrent theme in the included papers is that of using metaphors to characterize software and its code. Averbukh et al. [16] describe a metaphor in this context as a mapping of objects and concepts to a system of similarities and analogies. Furthermore, Vincur et al. [10] mention that metaphors serve to reduce disorientation as they rely on real-world knowledge, while Romano et al. [13] also suggest they provide insight about the system being visualized. In the reviewed literature we found a small set of metaphors that are used in VR software visualization proposals:

City metaphor:

The city metaphor (figure 1a) was overwhelmingly the most common one among the selected works [9]–[11], [13], [15], an observation with which authors such as Romano et al. [13] and Merino [15] agreed with. Each implementation of the city metaphor is different, however, they are always based around the idea of creating buildings of differing heights that represent a certain aspect of the software. Some examples of what the buildings can represent are: packages [9] and classes [10], [11], [13], [15]. Further in, buildings are composed of floors which can be mapped to sub-packages [9], methods [10], or metrics [11], [13], [15]. Dependencies between packages or classes can be represented explicitly like in [9]–[11] or implicitly via the proximity between buildings.

Island metaphor:

Schreiber and Misiak [12] proposed a different type of metaphor based on islands in an ocean (figure 1b). Their approach is intended to work with OSGi projects which are organized into modules and services that connect them. The metaphor represents each module as an island, and

each island is divided into regions for each package. The region allocation algorithm reserves spaces based on the number of classes in a particular package. This is done with the intention of using each cell to draw a building that corresponds to a particular class and whose height codifies the lines of code that constitute it. Finally, dependencies between modules are drawn as incoming and departing arrows that connect dedicated port objects located next to each island.

Universe metaphor:

Oberhauser and Lecon's [11] proposal allows the user to select among two different metaphors for visualization, as well as customize elements to their preferences. One of the preset metaphors is a universe (figure 1c) populated with solar systems, where each system corresponds to a Java package, the planets within them represent classes, and dependencies are drawn as beams of light that connect them.

2) Interaction and navigation: While the methods of representation seem to be consolidated around the city metaphor, methods of interaction and navigation can vary greatly between authors. Fittkau et al. [9] employ a gesture tracking system with a Microsoft Kinect which maps hand motions to actions such as panning, rotating, zooming, and selecting objects. Controllers are another frequent method of input, these include motion-tracked [10]–[12] and gamepads [13]. With this modality, users can use buttons, analog sticks, and tactile pads to interact with objects and move in the virtual space. Some authors [11]–[13], [15] even implement virtual representations of real-world devices such as tablets, PDAs, and keyboards that allow the input and output of even more information.

V. Conclusions and future work

We set out to study a subset of frequently cited research relating to the field of software visualization in virtual reality. The results of our review suggest that there is an abundance of different methods concerned with visualizing static, dynamic, and evolutional aspects of software. Although the variety of techniques is quite large and hasn't reached a consensus yet, the community seems to agree on the usage of metaphors as a convenient means for easily understanding abstract structures and concepts. We also found that the use of virtual reality brings with it an advantage in the form of affordances, which allow humans more extensively utilize their cognitive abilities.

It is still possible to ask more research questions in this context. As part of future work, we would like to explore the common challenges faced when creating a VR software visualization, as well as identify active research points and gaps in a more explicit fashion.

References

[1] J. T. Stasko, M. H. Brown, J. B. Domingue, and B. A. Price, *Software Visualization*. MIT Press, 1998, ISBN: 9780262193955.

[2] E. Dimara and C. Perin, "What is Interaction for Data Visualization?" *IEEE Transactions on Visualization and Computer Graphics*, vol. 26, no. 1, pp. 119–129, 2020. DOI: 10.1109/TVCG.2019.2934283. [Online]. Available: https://hal.archives-ouvertes.fr/hal-02197062.

[3] B. Bach, R. Sicat, J. Beyer, M. Cordeil, and H. Pfister, "The hologram in my hand: How effective is interactive exploration of 3d visualizations in immersive tangible augmented reality?" *IEEE Transactions on Visualization and Computer Graphics*, vol. 24, no. 1, pp. 457–467, 2018. DOI: 10.1109/TVCG.2017.2745941.

[4] A. R. Teyseyre and M. R. Campo, "An overview of 3d software visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 15, no. 1, pp. 87–105, 2009. DOI: 10.1109/TVCG.2008.86.

[5] M. Shahin, P. Liang, and M. A. Babar, "A systematic review of software architecture visualization techniques," *Journal of Systems and Software*, vol. 94, pp. 161–185, 2014, ISSN: 0164-1212. DOI: https://doi.org/10.1016/j.jss.2014.03.071. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0164121214000831.

[6] A.-L. Mattila, P. Ihantola, T. Kilamo, A. Luoto, M. Nurminen, and H. Väätäjä, "Software visualization today: Systematic literature review," in *Proceedings of the 20th International Academic Mindtrek Conference*, ser. AcademicMindtrek '16, Tampere, Finland: Association for Computing Machinery, 2016, pp. 262–271, ISBN: 9781450343671. DOI: 10.1145/2994310.2994327. [Online]. Available: https://doi-org.ezproxy.itcr.ac.cr/10.1145/2994310.2994327.

[7] L. Merino, M. Ghafari, C. Anslow, and O. Nierstrasz, "A systematic literature review of software visualization evaluation," *Journal of Systems and Software*, vol. 144, pp. 165–180, 2018, ISSN: 0164-1212. DOI: https://doi.org/10.1016/j.jss.2018.06.027. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0164121218301237.

[8] L. Bedu, O. Tinh, and F. Petrillo, "A tertiary systematic literature review on software visualization," in *2019 Working Conference on Software Visualization (VISSOFT)*, 2019, pp. 33–44. DOI: 10.1109/VISSOFT.2019.00013.

[9] F. Fittkau, A. Krause, and W. Hasselbring, "Exploring software cities in virtual reality," in *2015 IEEE 3rd Working Conference on Software Visualization (VISSOFT)*, 2015, pp. 130–134. DOI: 10.1109/VISSOFT.2015.7332423.

[10] J. Vincur, P. Navrat, and I. Polasek, "Vr city: Software analysis in virtual reality environment," in *2017 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, 2017, pp. 509–516. DOI: 10.1109/QRS-C.2017.88.

[11] R. Oberhauser and C. Lecon, "Gamified virtual reality for program code structure comprehension," *The International Journal of Virtual Reality*, vol. 17, pp. 77–86, 02 Jan. 2017. DOI: 10.20870/IJVR.2017.17.2.2894. [Online]. Available: https://ijvr.eu/article/view/2894.

[12] A. Schreiber and M. Misiak, "Visualizing software architectures in virtual reality with an island metaphor," in *Virtual, Augmented and Mixed Reality: Interaction, Navigation, Visualization, Embodiment, and Simulation*, J. Y. Chen and G. Fragomeni, Eds., Cham: Springer International Publishing, 2018, pp. 168–182, ISBN: 978-3-319-91581-4.

[13] S. Romano, N. Capece, U. Erra, G. Scanniello, and M. Lanza, "On the use of virtual reality in software visualization: The case of the city metaphor," *Information and Software Technology*, vol. 114, pp. 92–106, 2019, ISSN: 0950-5849. DOI: https://doi.org/10.1016/j.infsof.2019.06.007. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950584919301405.

[14] A. Elliott, B. Peiris, and C. Parnin, "Virtual reality in software engineering: Affordances, applications, and challenges," in *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, vol. 2, 2015, pp. 547–550. DOI: 10.1109/ICSE.2015.191.

[15] L. Merino, M. Ghafari, C. Anslow, and O. Nierstrasz, "Cityvr: Gameful software visualization," in *2017 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, 2017, pp. 633–637. DOI: 10.1109/ICSME.2017.70.

[16] V. Averbukh, N. Averbukh, P. Vasev, *et al.*, "Metaphors for software visualization systems based on virtual reality," in *Augmented Reality, Virtual Reality, and Computer Graphics*, L. T. De Paolis and P. Bourdot, Eds., Cham: Springer International Publishing, 2019, pp. 60–70, ISBN: 978-3-030-25965-5.

[17] W. W. Gaver, "Technology affordances," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '91, New Orleans, Louisiana, USA: Association for Computing Machinery, 1991, pp. 79–84, ISBN: 0897913833. DOI: 10.1145/108844.108856. [Online]. Available: https://doi.org/10.1145/108844.108856.