Exploring Software Cities in Virtual Reality

Florian Fittkau, Alexander Krause, and Wilhelm Hasselbring
Software Engineering Group, Kiel University, Kiel, Germany
Email: {ffi, akr, wha}@informatik.uni-kiel.de

***Abstract*—Software visualizations, such as the software city metaphor, are usually displayed on 2D screens and controlled by means of a mouse and thus often do not take advantage of more natural interaction techniques. Virtual reality (VR) approaches aim to improve the user experience. Emerging new technologies, like the Oculus Rift, dramatically enhance the VR experience at an affordable price. Therefore, new technologies have the potential to provide even higher immersion – and thus benefits – than previous VR approaches.**

We present a VR approach to explore software visualizations following the software city metaphor by using a head-mounted display and gesture-based interaction. Furthermore, we detail our gesture design and how we implemented this approach into our web-based ExplorViz tool. As first evaluation, we conducted structured interviews where participants had to solve three program comprehension tasks and rate the usability of the used gestures and general VR experience for program comprehension.

The participants of our interviews rated the developed gestures for translation, rotation, and selection as highly usable. However, our zooming gesture was less favored. In general, the subjects see potential for virtual reality in program comprehension.

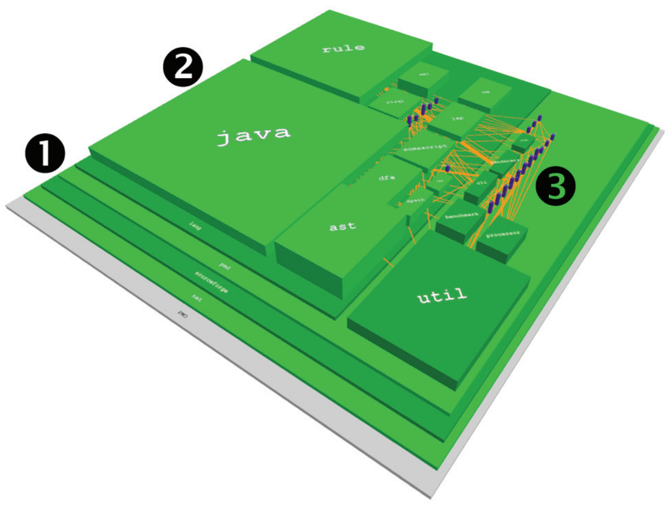


Fig. 1. Execution trace of PMD represented in ExplorViz

I. Introduction

Although 3D software visualizations can deliver more information compared to 2D visualizations, it is often difficult for users to navigate in 3D spaces using a 2D input device [1]. As a consequence, users may get disoriented [2] and thus the advantages of a third dimension may be abolished.

Virtual Reality (VR) can employ the natural perception of spatial locality of users and thus provide advantages for 3D visualization [3], [4]. In addition to stereoscopic [5], [6] display, natural interaction beyond the 2D mouse provides advantages, e.g., creativity can be enhanced by walking around [7].

In this paper, we present our VR approach for exploring software cities with a head-mounted display (HMD) and gesture-based interaction. It is integrated in our web-based ExplorViz[1] [8] tool for live trace visualization. As HMD, we utilize the Oculus Rift DK1 and the Microsoft Kinect v2 for gesture recognition. Our gestures are designed for reusability with other sensors and interaction with 3D models such that other visualization metaphors can take advantages of these.

To evaluate our VR approach, we interviewed eleven participants and asked them to conduct three program comprehension tasks. Afterwards, they rated their sympathy for the employed gestures and the approach in general. To facilitate the verifiability and reproducibility of our results, we provide a package [9] containing all our evaluation data including source code, raw data, and eleven video recordings of the participant sessions.

[1]http://www.explorviz.net

In summary, our main contributions are:

1. A VR approach for exploring software cities with a head-mounted display and gesture-based interaction,
2. a reusable gesture design for 3D model manipulation in program comprehension, and
3. an interview with eleven participants evaluating the usability of our gestures and our approach in general.

The remainder of this paper is organized as follows. Section II introduces our city metaphor used in our ExplorViz visualization. Then, Section III describes our VR approach for exploring software cities. Afterwards, Section IV presents a usability evaluation of our approach. Related work is discussed in Section V. Finally, we draw the conclusions and illustrate future work in Section VI.

II. ExplorViz in a Nutshell

This section briefly introduces our city metaphor used in ExplorViz and thus also in our VR approach. Fig. 1 displays our city metaphor used in ExplorViz. It visualizes an execution trace of PMD which is also used in our evaluation. In ExplorViz, we follow an interactive, top-down approach to show details on demand following the Shneiderman mantra [10] of "Overview first, zoom and filter, then details on demand." Therefore, we represent packages by two entities: open and closed packages. Open packages are visualized by flat green boxes (❶) which show their internal details, i.e., sub packages, classes, and communication. Green boxes on the top layer (❷)

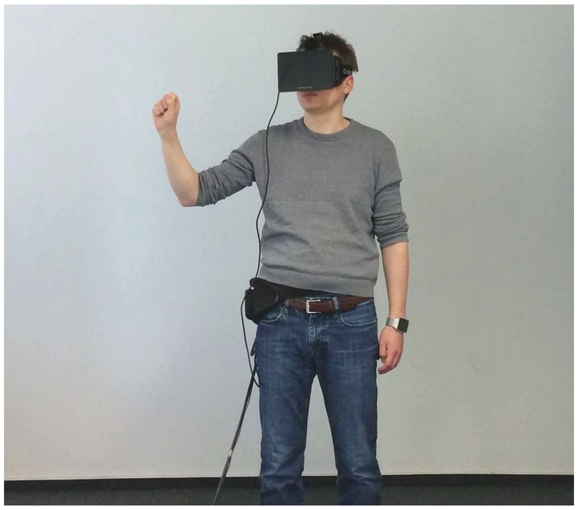


Fig. 2. Setup for our VR approach (doing a translation gesture)

represent closed packages hiding their internal details. Packages can be opened or closed interactively. Classes are visualized as purple boxes and the communication is displayed as orange lines (❸). The width of the line corresponds to the call frequency of the represented methods. The height of classes maps to the active instance count.

III. Virtual Reality Approach

In this section, we describe our VR approach. The basic setup of our VR approach is visualized in Fig. 2. We utilize an Oculus Rift for displaying the software city and use gesture recognition through a Microsoft Kinect v2. For implementation details, we refer to [11]. We first introduce the necessary components for an alternative display. Afterwards, we discuss the gesture design as well as their detection mechanisms.

A. Display

The *Oculus Rift*[2] is a HMD. Its motion sensor and large field of view provide an immersive usage. For our approach, we utilize the Development Kit 1 (DK1) version with an overall display resolution of 1280x800.

Since ExplorViz is a web application, we use the experimental JavaScript API *WebVR*[3] to access the functionality of the Oculus Rift in web browsers. This API is already included in experimental builds of Mozilla Firefox and Google Chrome. It eases the development by providing access to the sensors through JavaScript and provides the necessary rendering effects [12], e.g., distortion, out-of-the-box. Due to the WebVR abstraction, later versions of the Oculus Rift and other HMDs will also function with minor adaptations.

Since the Rift uses one image for each eye, the model created by ExplorViz needs to be rendered twice with different 3D transformations, i.e., a slight translation between the eyes. Fig. 3 shows a screenshot of the wearer's view. Head rotation

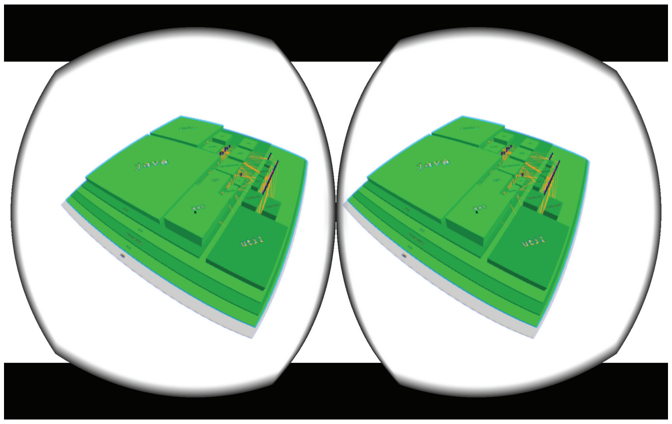


Fig. 3. View on the software city of PMD through the Oculus Rift DK1

leads to viewpoint rotation in the virtual space. Hence, users only need to rotate their head to view near model elements instead of moving them to the center of a firm viewpoint.

B. Gestures

For gesture recognition, we use the Microsoft *Kinect v2 for Windows*.[4] It contains a depth camera with a resolution of 512x424 pixels and can be used as body tracker through an SDK. For gesture recognition, a C# application needs to be deployed on every client separately.

There are two basic concepts for designing gesture-based movement actions. The first concept is commonly used by control sticks in game controllers. A user performs a gesture and holds the position at a boundary. While he holds this position, the movement is conducted continuously into the implied direction. The second concept is a direct mapping between the hand movement and the movement action in the model, similar to how a computer mouse works.

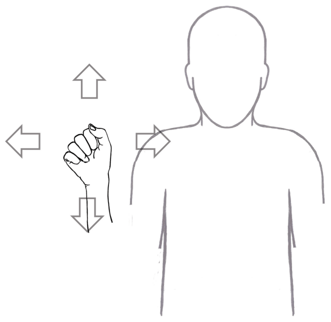
In our prior tests conducted with three users, they familiarized with a direct mapping faster than with the first concept. Furthermore, participants working with the continuous movement sometimes tried to manipulate the model as if they would use a direct mapping approach. Thus, we discarded the first concept and designed our gestures with a direct mapping of hand to manipulation.

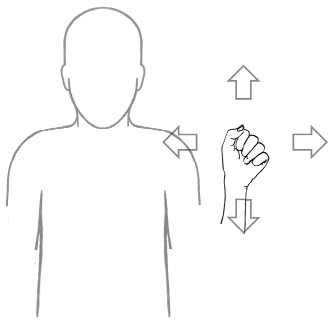
In the following, we describe the gestures of our VR approach. Although we distinguish between left and right hand, whether a person is left or right-handed should make no difference. We discuss different design alternatives and present our final gestures. Examples of the execution of these gestures can be found in the video recordings [9] of our interviews.

1) Translation: Fig. 4a shows the gesture for moving the model. The user lifts the right hand, clenches into a fist, and then moves the object. The gesture is derived from a translation of an object in reality by graping an object and then moving it. Furthermore, it bears a similarity with dragging and swiping on touch-based displays. Since this gesture quickly turned out

[2]http://www.oculus.com
[3]http://webvr.info

[4]http://www.microsoft.com/en-us/kinectforwindows


(a) Gesture for moving the model


(b) Gesture for rotating the model

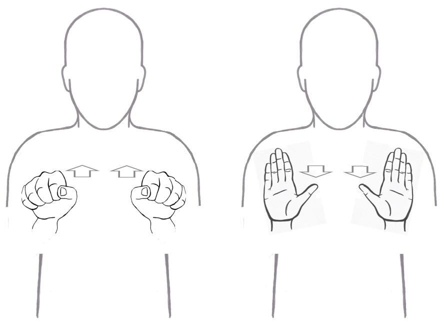

(c) Gesture for zooming in and out

Fig. 4. Gesture concepts for interacting with the software city model

as intuitive and understandable in our first tests, we did not design an alternative.

2) Rotation: Our first design for rotating the model was derived from holding and spinning a ball with two hands. A virtual line between the hands formed an axis used for the rotation. Unfortunately, it was not possible to detect all real-world interactions, e.g., rotation of the hands, due to some restrictions of the Kinect sensor. For example, when the hands overlap in the depth dimension, the background hand joints are not detected correctly. Fig. 4b shows the current design of this gesture. It is very similar to the translation gesture and only differs in using the left hand.

3) Zoom: The first design used moving the hands apart and together for zooming similar to pinching on a mobile device. However, we wanted to use the depth of the room. The gesture only used two dimensions and was not comparable to manipulating real objects. The second design used the body tilt as calculation base. Leaning forward with the upper body resulted in zooming in. Leaning backward with the upper body led to zooming out. Our tests revealed that users tend to lower or raise their head while performing this gesture. Hence, they also change the viewpoint due to the Oculus Rift rotation sensor leading to confusion during performing this gesture. Our next concept involved walking forward and backward to zoom in and out. Due to the possible lack of space and users rotating their body while performing this gesture, this approach was also inappropriate. Fig. 4c shows the current implementation. The gesture is derived from real life interaction similar to rowing. To zoom in, the user raises his hands, clenches both into a fist, and pulls them towards his chest. To zoom out, the user raises his hands and pushes them away from the body. Thus, the gesture maps to pulling and pushing the model towards or away from the viewer.

4) Selection: To select an entity in the software city model, the user raises his right hand, and quickly closes and opens it. To open or close a package, users have to do this gesture twice. It is important to fully open the hand because the Kinect might recognize a half-open hand as closed.

5) Reset: Users can reset their viewpoint to the origin again by performing a jump gesture. The first implementation required the lifting and subsequent closing and opening of both hands above the head. Our tests revealed that this design is inappropriate, since users sometimes adjust the wearing of the Oculus Rift with their hands and thus trigger the unintended action. In contrast, the jump gesture is completely different to the other gestures and is therefore unlikely to be triggered unintentionally.

IV. Usability Evaluation

In this section, we report on an evaluation of the usability of our VR approach. We conducted eleven structured interviews where the subjects should solve three program comprehension tasks and rate the usability of the designed gestures. As object system, we chose PMD since we already gathered some experience with its visualization in ExplorViz [13].

We start by describing the design of the evaluation. Then, its operation is briefly presented. Afterwards, results are discussed and the qualitative feedback is shown.

A. Design

Since it is a first evaluation of our approach, we aim for a qualitative analysis of the gestures and the VR approach in general. Therefore, we let the participant solve program comprehension tasks and afterwards, each user should anonymously rate the approach and gestures.

1) Subjects: Eleven computer science students took part in our interview. Four participants were pursuing the Bachelor degree. Six subjects were in the Master's degree program and one PhD student took part. The average self-rated programming experience of the subjects was between *Intermediate* and *Advanced*. The self-rated experience with software architectures was between *Beginner* and *Intermediate*.

TABLE I
DESCRIPTION OF THE PROGRAM COMPREHENSION TASKS FOR THE INTERVIEW

ID	Description
	Context: Metric-Based Analysis
T1	Name the largest package inside the `lang` package.
	Context: Structural Understanding
T2	Name all classes communicating with the class `Java15Handler` inside of the `java` package.
	Context: Concept Location
T3	What is the purpose of the `Language` class inside the `lang` package?

TABLE II
DEBRIEFING QUESTIONNAIRE RESULTS FOR OUR INTERVIEW (HIGHER IS BETTER)

	Score	
	mean	stdev.
VR for program comprehension (0 to 2)	1.27	0.64
Alternative to classic monitors (0 to 2)	1.00	0.77
Favor of Gestures (0 - 4)		
Translation	2.73	1.10
Rotation	2.64	0.80
Zoom	1.45	1.03
Selection	2.64	0.92

2) Tasks: Table I shows the three tasks for our interviews. We chose to have only three tasks since new users of the Oculus Rift DK1 are advised to wear it only a short duration for the first time to avoid Videogame-Related Motion Sickness [14], [15]. From our own experience, this duration increases after multiple usage and will be minimized with future versions of HMDs. The tasks are forming a plot thread and start form easy metric-based detection to harder concept location. Since we are only interested in the qualitative results, the correlation of the questions is not harmful for our analysis.

B. Operation

1) Generating the Input: We generated the input for ExplorViz directly from the execution of PMD version 5.1.2. ExplorViz persists its data model into a file which acts as a replay source during the structured interviews.

2) Procedure: Our interviews took place at the Kiel University. Each participant had a single session of up to one hour. All subjects used the same computer and equipment. Before the interviews took place, we benchmarked the computer to ensure that both the gesture recognition application and the ExplorViz tool run smoothly at the same time.

After telling the participants that they can ask questions at all times, each subject received a tutorial for the gestures and ExplorViz. Then, each subject could freely test the gestures to get familiar with the model manipulation and viewpoint rotation by the Oculus Rift.

Afterwards, all program comprehension tasks were read to the participants and answered by acclamation. Poorly executed gestures were also adjusted by the instructor. The session ended with the debriefing questionnaire which was filled out anonymously to mitigate social influences.

3) Tutorial: The tutorial was conducted by the same instructor for each session. It started with a brief introduction to PMD and its scanning procedure. Next, gestures and ExplorViz's semantics were explained one after another. All participants were told that they need to face the Kinect sensor at all times to minimize unintended gesture recognition.

C. Results and Discussion

The solutions to the tasks and time spent are irrelevant for our evaluation since we aim for qualitative feedback and have no comparison values. After conducting the interviews, we analyzed the video recordings. Due to the low resolution of the Oculus Rift DK1, the labels in the model were hard to read such that the subjects had to zoom in very closely to the labels to read them. Furthermore, the recognition of the gestures was sometimes hindered due to inaccurate execution of it or due to not facing the Kinect directly. Two of eleven subjects got motion sickness and had to skip the third task.

The results of the debriefing questionnaire are shown in Table II. The subjects rated the potential of VR for program comprehension with a 1.27 and as alternative to a classic monitor with a 1.00. The first value maps to slightly better than *With Adaptations* and the second value maps directly to *With Adaptations*. We attribute this mainly to the low display resolution of the used HMD. The gestures for translation, rotation, and selection were rated around 2.6 on a scale between 0 and 4, which maps to between the neutral center and a *Like*. However, the zoom gesture only got a 1.45 which maps to slightly better than a *Dislike*. Therefore, future work should investigate other zooming gestures.

Notably, for higher external validity, the interviewed set of persons should be larger and more diverse, e.g., professionals.

D. Qualitative Feedback

Six subjects want a higher resolution such that labels become more readable. Two subjects noted that gestures are slower than the mouse for them which is potentially caused by a bad gesture recognition in their case. Two subjects stated that the familiarization with the gestures takes time. For example, they had to stand at a defined position and they had to fully open the hand such that the Kinect recognizes the gesture.

V. Related Work

In this section, we describe related work of VR and augmented reality approaches for software visualization and related work of the city metaphor in general.

Imsovision [16] represents object-oriented software in a VR environment. Electromagnetic sensors, which are attached to the shutter glasses, track the user and a *wand* is used for 3D navigation. In contrast to them, we use a hands-free gesture recognition. SykscrapAR [17] aims to represent software evolution by utilizing an augmented reality approach employing the city metaphor. The user can interact with a physical marker platform in an intuitive way. Contrary to our approach, the user only sees the 3D model on a monitor.

Delimarschi et al. [4] introduce a concept of natural user interfaces for IDEs by using voice commands and gesture-based interaction with a Microsoft Kinect. In contrast, they do not utilize HMDs to create an immersive VR experience. Young and Munro [18] developed FileVis which visualizes files and provides an overview of the system. Although they aim for virtual environment, no technological approach is described, e.g., no HMDs or gesture recognition sensor.

Several city metaphor approaches have been presented in the literature, e.g., [19]–[22]. However, to the best of our knowledge, no VR approach which uses HMDs and gesture-based interaction exist.

VI. Conclusions And Outlook

In this paper, we presented a VR approach to explore software visualizations following the 3D city metaphor. We use the Oculus Rift DK1 for displaying the software city and a Microsoft Kinect v2 sensor for gesture recognition. In the section about the gesture design, different possibilities for gestures and our experiences with them were described. To evaluate the usability of our VR approach, we conducted eleven structured interviews. The subjects were asked to solve three program comprehension tasks and to rate the usability of each gesture.

The structured interviews revealed that the gestures for translation, rotation, and selection provide good usability. However, our zooming gesture was less favored. In general, the subjects see potential for VR in program comprehension.

To facilitate the verifiability and reproducibility for replications and further evaluations, we provide a package containing all our evaluation data. Included are the employed version of ExplorViz v1.0-vr (including source code and manual), input files, questionnaire, results, and eleven video recordings of the participant sessions. The package is available online [9] with source code under the Apache 2.0 License and the data under a Creative Commons License (CC BY 3.0).

As future work, we will test other HMDs since higher display resolutions will provide a better user experience, especially for reading labels. Candidates for this test are newer versions of the Oculus Rift and the current competitors, e.g., the *HTC Vive*. Furthermore, our approach should be evaluated in a controlled experiment where the performance of solving program comprehension tasks are compared to a classical monitor and mouse setup. Such experiment would reveal quantitative results about the impact of our VR approach.

Further future work lies in testing other input sensors and methods. Hands-free gestures can also be recognized with the *Leap Motion* system which should be compared to the performance of the Kinect v2 for the presented gestures. Position-tracked controllers, like the *Sixense Stem* or the *Oculus Touch*, might provide higher accuracy for the recognition of the presented gestures. As completely different input method, we see potential, for example, in using brain user interfaces enabled by, e.g., the *Emotiv Insight*.

Our VR approach should also be tested with other types of software visualizations. The display and developed gestures should be easily transferable to the new context. Furthermore, a test of the VR experience instead of focusing on the gestures could provide further insights.

References

[1] A. Teyseyre and M. Campo, "An overview of 3D software visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 15, no. 1, pp. 87–105, Jan. 2009.

[2] K. P. Herndon, A. van Dam, and M. Gleicher, "The challenges of 3D interaction: A CHI '94 Workshop," *SIGCHI Bull.*, vol. 26, no. 4, pp. 36–43, Oct. 1994.

[3] A. Elliott, B. Peiris, and C. Parnin, "Virtual reality in software engineering: Affordances, applications, and challenges," in *Proc. of 37th Int Conf. on Software Engineering (ICSE 2015)*. IEEE, May 2015.

[4] D. Delimarschi, G. Swartzendruber, and H. Kagdi, "Enabling integrated development environments with natural user interface interactions," in *Proceedings of the 22nd International Conference on Program Comprehension (ICPC 2014)*. ACM, 2014, pp. 126–129.

[5] C. Ware, K. Arthur, and K. S. Booth, "Fish tank virtual reality," in *Proceedings of the INTERACT 1993 and Conference on Human Factors in Computing Systems (CHI 1993)*. ACM, 1993, pp. 37–42.

[6] C. Ware and P. Mitchell, "Reevaluating stereo and motion cues for visualizing graphs in three dimensions," in *Proc. of 2nd Symp. on Applied Perception in Graphics and Vis. (APGV 2005)*. ACM, 2005.

[7] M. Oppezzo and D. L. Schwartz, "Give your ideas some legs: The positive effect of walking on creative thinking." *Journal of experimental psychology. Learning, memory, and cognition*, 2014.

[8] F. Fittkau, J. Waller, C. Wulf, and W. Hasselbring, "Live trace visualization for comprehending large software landscapes: The ExplorViz approach," in *Proceedings of the 1st International Working Conference on Software Visualization (VISSOFT 2013)*, Sep. 2013.

[9] F. Fittkau, A. Krause, and W. Hasselbring, "Experimental data for: Exploring software cities in virtual reality," DOI: 10.5281/zenodo.23168.

[10] B. Shneiderman, "The eyes have it: A task by data type taxonomy for information visualizations," in *Proceedings of the IEEE Symposium on Visual Languages*. IEEE, 1996, pp. 336–343.

[11] A. Krause, "Erkundung von Softwarestädten mithilfe der virtuellen Realität," Sep. 2015, Bachelor's Thesis, Kiel University, (to appear, in German).

[12] Oculus, "Oculus - best practices guide," 2015, http://developer.oculusvr.com/best-practices.

[13] F. Fittkau, S. Finke, W. Hasselbring, and J. Waller, "Comparing trace visualizations for program comprehension through controlled experiments," in *Proceedings of the 23rd IEEE International Conference on Program Comprehension (ICPC 2015)*. IEEE, May 2015.

[14] E. M. Kolasinski, "Simulator sickness in virtual environments," DTIC Document, Tech. Rep. 1027, 1995.

[15] K. Meyer, H. L. Applewhite, and F. A. Biocca, "A survey of position trackers," in *Presence: Teleoperators and Virtual Env.*, vol. 1, 1992.

[16] J. I. Maletic, J. Leigh, and A. Marcus, "Visualizing software in an immersive virtual reality environment," in *Proc. of 23rd Int Conf. on Software Engineering (ICSE 2001)*. IEEE, 2001.

[17] R. Souza, B. Silva, T. Mendes, and M. Mendonca, "SkyscrapAR: An augmented reality visualization for software evolution," in *Proc. of 2nd Brazilian Workshop on Software Visualization (WBVS 2012)*, 2012.

[18] P. Young and M. Munro, "Visualising software in virtual reality," in *Proceedings of the 6th International Workshop on Program Comprehension (IWPC 1998)*, 1998, pp. 19–26.

[19] C. Knight and M. Munro, "Virtual but visible software," in *Proceedings of the IEEE International Conference on Information Visualization (IV 2000)*. IEEE, 2000, pp. 198–205.

[20] ——, "Comprehension with[in] virtual environment visualisations," in *Proceedings of the 7th International Workshop on Program Comprehension (IWPC 1999)*, 1999, pp. 4–11.

[21] T. Panas, R. Berrigan, and J. Grundy, "A 3D metaphor for software production visualization," in *Proc. of 7th Int. Conf. on Information Visualization (IV 2003)*. IEEE Computer Society, 2003, pp. 314–320.

[22] R. Wettel and M. Lanza, "Visualizing software systems as cities," in *Proc. of 4th IEEE Int. Workshop on Visualizing Software for Understanding and Analysis (VISSOFT 2007)*. IEEE, 2007, pp. 92–99.