Program Comprehension in Virtual Reality

James Dominic*, Brock Tubre*, Jada Houser, Charles Ritter, Deborah Kunkel, Paige Rodeghero
{domini4,btubre,jahouse,crritte,dekunke,prodegh}@clemson.edu
Clemson University
Clemson, USA

ABSTRACT

Virtual reality is an emerging technology for various domains such as medicine, psychotherapy, architecture, and gaming. Recently, software engineering researchers have started to explore virtual reality as a tool for programmers. However, few studies examine source code comprehension in a virtual reality (VR) environment. In this paper, we explore the human experience of comprehending source code in VR. We conducted a study with 26 graduate students. We found that the programmers experienced more challenges when reading and comprehending code in VR. We found no statistically significant difference in the programmers' perceived productivity between our VR and traditional comprehension studies.

CCS CONCEPTS

• **Software and its engineering → Programming teams**.

KEYWORDS

collaboration, pair programming, virtual reality

ACM Reference Format:
James Dominic*, Brock Tubre*, Jada Houser, Charles Ritter, Deborah Kunkel, Paige Rodeghero. 2020. Program Comprehension in Virtual Reality. In *28th International Conference on Program Comprehension (ICPC '20), October 5–6, 2020, Seoul, Republic of Korea.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3387904.3389287

1 INTRODUCTION

The use of Virtual Reality (VR) technology is growing in various domains such as education, medicine, and architecture [23, 29, 34, 38, 39, 48]. Software engineering researchers have recently started to study the benefits of VR for developers and researchers [1, 21, 44, 55]. Shepherd *et al.* found that VR reduces the distractions of software engineers [42]. Researchers have also started to explore the ways that programmers can communicate and work together in a VR environment [3, 8, 50].

For programmers, one of the most challenging and time-consuming tasks is comprehending source code [2, 6, 26, 46]. Current solutions to address these problems include using visualizations to display code, enhanced syntax highlighting, and gamification of comprehension [33, 36, 52]. Programmers also find that collaboration leads to better code comprehension [5].

Many programmers work remotely and find it difficult to collaborate [37]. The lack of social presence is one of the factors impacting remote collaboration [4]. A possible solution to this problem is to have remote programmers collaborate within VR. VR research has shown to increase the social presence during collaboration [35].

In this paper, we study the human experience (i.e. frustration, effort) during code comprehension. We study graduate student programmers comprehending source code in both a traditional computer setup and in a VR environment. We recruited 26 programmers to participate in this study. The program comprehension tasks included asking programmers to read source code, provide the code output, and write a plain English text summary of the code. In the study, 13 programmers worked in VR, and the other 13 programmers used the traditional computer setup. We asked programmers in both studies to complete as many comprehension tasks as they could from 8 different code snippets in 40 minutes. We found that VR comprehension leads to lower performance among programmers. Programmers in VR reported lower levels of concentration and higher levels of frustration, effort, and mental demand. However, we found that the VR programmers' perceived productivity was not significantly different than the traditional desktop programmers' perceived productivity.

2 THE PROBLEM

We address the following gap in the current program comprehension literature: there are no studies on how programmers read and understand source code in VR. As a community, we are starting to explore various uses of VR [13, 33, 42]. This paper is as close as we, as a community, have reached researching comprehension in VR. We have not taken the risks that other fields have for years. Many fields have found benefits to using VR for training and education [22, 32, 43, 51]. Other fields have found benefits from using VR for collaborations [20, 25]. However, until now, we have not had programmers enter into a VR environment and study program comprehension. Unfortunately, even though we recognize program comprehension as one of the most significant factors for success, we have not spent the time to explore the potential of using VR.

Remote work has become more prominent in the software engineering field. This means some programmers never enter their company's physical office location. Remote programmers use Skype and various other tools for collaboration [53]. Unfortunately, this solution is problematic. Remote programmers still find it challenging to collaborate with these tools [27]. One solution is to use VR to bring programmers into the same environment for collaboration. If code comprehension and the human experience are similar or better in VR, then it is worth further exploring for programmers.

*Both authors contributed equally to this research.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

ICPC '20, October 5–6, 2020, Seoul, Republic of Korea
© 2020 Association for Computing Machinery.
ACM ISBN 978-1-4503-7958-8/20/05…$15.00
https://doi.org/10.1145/3387904.3389287

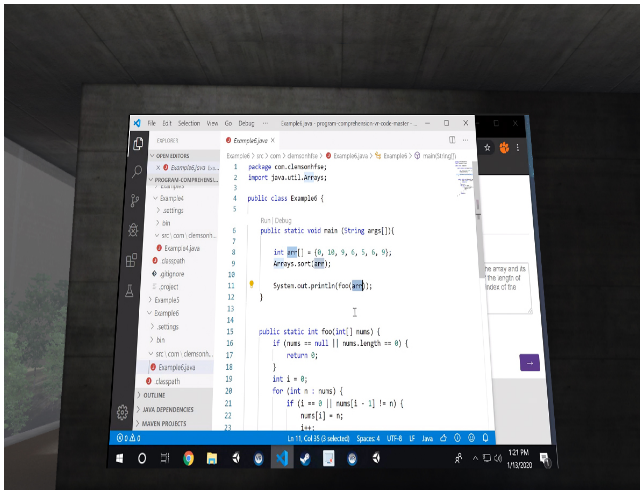

Figure 1: The virtual reality environment used by the programmer to complete source code comprehension tasks. This figure shows the desktop mirror as the programmer comprehended source code.

3 BACKGROUND AND RELATED WORK

In this section, we discuss the background and related work on virtual reality in software engineering and the human experience during source code comprehension tasks.

3.1 Virtual Reality in Software Engineering

Virtual Reality (VR) is a realistic interactive environment that is created by computer graphics. VR systems have been implemented in a wide variety of research areas, including gaming, training, education, therapy, and more [10, 30, 31]. As a community, software engineering researchers have started to explore the uses of VR for programmers. Elliott *et al.* created an environment where users could type code into a text editor and manipulate their virtual environment [14]. A study by Takala showed how VR could be used as a tool to provide help with learning how to program software systems. This environment included "3D User Interface Building Blocks," which make it easier for developers to interact with various peripherals that would be usable for VR [47]. Fittkau *et al.* researched "software cities" [15], which are 3D representations, in virtual reality, of the packages and classes in a software system.

3.2 Human Experience During Comprehension

The human experience during source code comprehension is complex [2, 26, 49]. Research has found that programmers do have similar human experiences while comprehending source code, such as frustration [12, 17]. Researchers have used fMRI machines to understand better how the brain comprehends source code [11, 16, 45]. Unfortunately, due to space limitations, we are unable to discuss the human experience literature in detail in this paper.

4 VIRTUAL REALITY STUDY DESIGN

In this section, we describe our research objective, research questions, and our methodology.

4.1 Research Questions

Our research objective is to understand the human experience and how programmers comprehend source code when working in a virtual reality environment. Therefore, we ask the following Research Questions (RQs):

RQ_1 To what degree does the virtual reality comprehension and traditional comprehension differ in terms of concentration?

RQ_2 To what degree does the virtual reality comprehension and traditional comprehension differ in terms of perceived productivity?

The rationale behind RQ_1 is that concentration is a weak predictor in reading comprehension, according to Wolfgramm *et al*, but it may be a stronger predictor in code comprehension [54]. The rationale behind RQ_2 is that comprehension influences productivity [40]. By exploring the perceived productivity in different environments, we can explore environmental influences on comprehension in order to optimize working conditions for programmers.

4.2 Methodology

We designed a research methodology based on related work in program comprehension [19, 41]. We studied individual programmers reading and comprehending source code. We had two groups of participants. One group of participants wore a virtual reality headset for the entire study. The other group of participants sat at a desk and used the traditional computer monitor to read the code and complete the comprehension tasks.

Participants sat in front of a traditional computer setup with a monitor, or they were asked to sit down and to put a virtual reality headset on. We had every other participant comprehend source code in virtual reality. For the VR participants, we first introduced them to the virtual reality system. The system consisted of a virtual environment with a Windows desktop mirrored in VR (See Figure 1). The desktop mirror allowed participants to have full interaction with a computer. For more information on the software used, see Section 4.10. The programmers who worked with a traditional desktop setup (no VR), worked on the same desktop system that was being mirrored in VR for the VR participants. The methodology is the same for both groups of participants throughout the rest of this methodology section.

We provided the participants with eight different Java projects. We randomized the order of the projects presented for each participant. The participant read one piece of code at a time. He or she then provided the output and an English text summary of the code. The participants had 40 minutes to complete the study. During the study, the programmer had internet access and could use any internet resource (i.e. Stack Overflow).

After the study, the programmers were given a post-experiment survey capturing information on their experiences during the code comprehension study. This survey captured data about how mentally and physically demanding the task was, along with more critical contributions to the workload for the task. We also surveyed programmers about their experiences using virtual reality.

4.3 Surveys

We conducted three surveys. We collected a demographics survey that asked questions about programming experience, including years of programming experience and experience in Java. At the end of the study, we asked all participants to fill out a NASA TLX survey. The NASA TLX survey is a multi-dimensional questionnaire that is designed to determine the workload while participants perform a particular task. All survey questions and anonymized data can be found at our online appendix (see Section 4.10).

4.4 Data Collection

We collected screen recordings of each study. We also collected both the function outputs and the code summaries. Finally, we collected the participants' demographics, self-reported programming experience, their experience in the virtual space, and the mental and physical workload experienced during the experiment.

4.5 Subject Application

The subject applications consist of 8 Java projects. These projects were selected because they are common methods and functions used in programming paradigms. These subject application consisted of methods that found the maximum and minimum values, compute the median, and determine if a string is a palindrome. The subject applications were collected from various coding platforms such as Leetcode and W3Schools. These projects are unique in their size, domain, and architecture, and they range in size from 22 lines to 57 lines of code. The subject applications can be found at our online appendix (see Section 4.10).

4.6 Participants

The programmers in the program comprehension study were all graduate students recruited through email. There were a total of 26 programmers that participated. The majority of the programmers self-reported their Java experience level as intermediate.

4.7 Statistical Tests

The data obtained in this study are ordinal in nature: the NASA-TLX scores are measured on a twenty-point ordinal scale, and the perceived performance survey provides ratings measured on a five-point ordinal scale. The responses for the perceived performance survey were first coded as numeric values (1 = no productivity; 5 = extremely productive), and the Mann-Whitney U test was used to assess these data for differences among the VR and desktop groups. This nonparametric test [24] compares ranks among groups and is appropriate for non-numeric data. A continuity correction was applied to the U test statistic to enable computation of an approximate p-value in the presence of tied ranks.

4.8 Equipment

The virtual reality system we used requires specific equipment for programmers to be able to type in virtual reality. We used one HTC Vive Pro headsets. The headset had a Leap Motion attached to the front of the headset for hand tracking. We also used two HTC Vive trackers. One tracker was attached to the keyboard. Another tracker was attached to the mouse. The HTC Vive trackers were used to track the keys on both the mouse and keyboard. When the participant clicked the mouse or typed, the virtual keyboard and mouse replicated the physical keyboard and mouse feedback. For more information on the equipment and setup, see Section 4.10.

4.9 Threats to Validity

Similar to any evaluation, our study carries threats to validity. First, there is a threat to the students who were used during our study. Factors like fatigue, stress, and error are all part of being human. The differences in programming experience, virtual reality experience, and personal biases can all affect our study. We cannot rule out the possibility that our results would differ if these factors were eliminated. However, we minimize this threat by recruiting 26 graduate-level programmers rather than relying on a single graduate student or recruiting undergraduate students. Another key source of threat is in the code snippets that we selected and written for our study. If we used a different set of code snippets, programming paradigms, a different programming language could all be threats to the validity. To mitigate this threat, we used code snippets that covered many different programming paradigms, object-oriented best practices, and common functions seen in programming.

4.10 Reproducibility & Online Appendix

For reproducibility, we have made all data and software available via an online appendix: http://d27hun5khpxjwi.cloudfront.net

5 HUMAN EXPERIENCE RESULTS

In this section, we present our answers to each research question, as well as our rationale, and an interpretation of the answers.

5.1 RQ_1: Concentration

We found that the programmers performing traditional comprehension attempted a total of 93 code snippets and comprehended 70 (75%) of them correctly. The programmer doing virtual reality comprehension attempted a total of 84 code snippets and comprehended 55 (65%) of them correctly. Figure 2 shows the results of the NASA TLX survey filled by the programmers. We observed significantly more overall task load ($p = 0.003$) on programmers in virtual reality comprehension than traditional code comprehension. Virtual reality comprehension is also more mentally demanding ($p = 0.034$) and physically demanding ($p = 0.017$) on programmers.

H_n The difference in concentration between virtual reality comprehension and traditional comprehension is statistically-significant.

The performance of software programmers can be adversely affected by increased task load [18, 28]. We found that virtual reality comprehension leads to lowered performance among the programmers. We argue that this is due to the inability of a programmer to concentrate on source code comprehension due to the added task load exerted on them during virtual comprehension.

5.2 RQ_2: Perceived Productivity

We found no statistically significant evidence that suggests any difference in perceived productivity ($p = 0.118$) between virtual comprehension and traditional comprehension. However, programmers

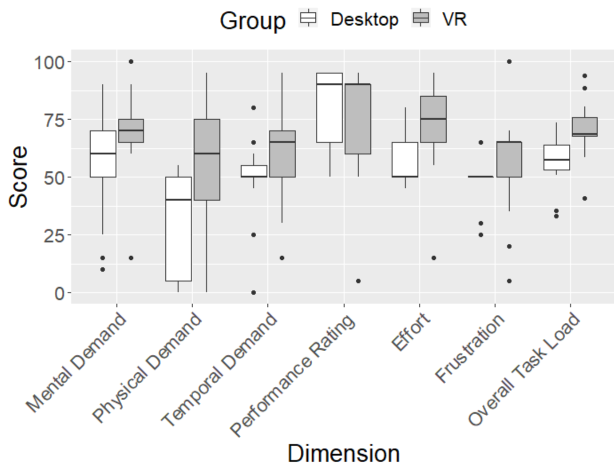

Figure 2: NASA TLX scores reported by programmers in virtual comprehension and traditional comprehension.

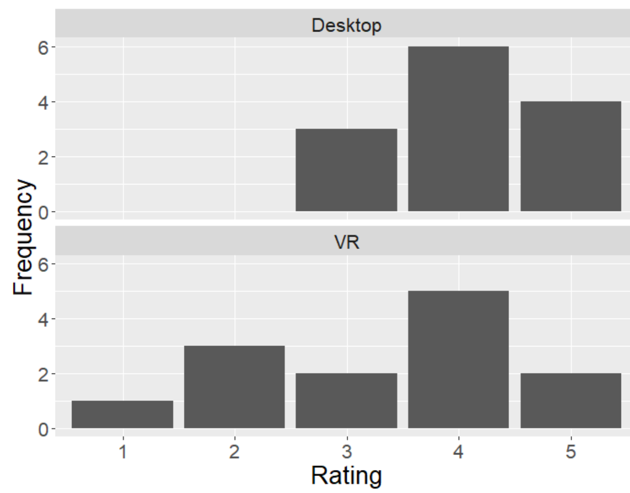

Figure 3: Distribution of perceived productivity ratings by group (1 = no productivity, 5 = extremely productive).

reported higher levels of perceived productivity doing traditional comprehension over virtual comprehension. Figure 3 shows the self-reported productivity score from the programmers.

H_n The difference in perceived productivity between virtual reality comprehension and traditional comprehension is not statistically-significant.

It is well understood that the introduction of new technology could, at times, distract the users and have unintended consequences [7, 9]. Even though the use of VR code comprehension is a novel approach, programmers did not report significant differences in perceived productivity even though they reported lower levels of concentration and higher levels of mental demand, frustration, and physical demand while performing virtual comprehension.

5.3 Summary of Human Experience Results

We derive a few key observations from the study results. First, programmers experience more challenges comprehending source code in virtual reality over traditional comprehension. We base this observation on the statistical-significance we found in the higher levels of mental demand, physical demand, and overall task load reported by the programmers on the NASA TLX survey while doing virtual program comprehension. Programmers completed fewer number of code comprehension and even fewer comprehended correctly in virtual reality.

Second, even though virtual program comprehension placed higher task load on the programmer, they found virtual reality as useful as traditional methods for program comprehension. We found no statistical significance for the difference between the perceived productivity of programmers while doing virtual program comprehension or traditional program comprehension.

6 CONCLUSION

We have presented a study exploring the use of virtual reality in program comprehension for software engineers. We compare the use of VR for program comprehension with traditional program comprehension. We have explored two research questions aimed at understanding how virtual reality affects programmers' ability to comprehend source code. We showed that programmers working in virtual reality face more difficulties comprehending code versus traditional program comprehension. However, we found no statistically significant evidence that suggests a difference in perceived productivity between comprehension in virtual reality and comprehension with a traditional desktop setup.

REFERENCES

[1] Akhan Akbulut, Cagatay Catal, and Burak Yıldız. 2018. On the effectiveness of virtual reality in the education of software engineering. *Computer Applications in Engineering Education* 26, 4 (2018), 918–927.

[2] Nedhal A Al-Saiyd. 2017. Source code comprehension analysis in software maintenance. In *2017 2nd International Conference on Computer and Communication Systems (ICCCS)*. IEEE, 1–5.

[3] Jacopo Aleotti, Stefano Caselli, and Monica Reggiani. 2004. Leveraging on a virtual environment for robot programming by demonstration. *Robotics and Autonomous Systems* 47, 2-3 (2004), 153–161.

[4] Ernesto Arias, Hal Eden, Gerhard Fischer, Andrew Gorman, and Eric Scharff. 2000. Transcending the individual human mind-creating shared understanding through collaborative design. *ACM Transactions on Computer-Human Interaction (TOCHI)* 7, 1 (2000), 84–113.

[5] Erik Arisholm, Hans Gallis, Tore Dyba, and Dag IK Sjoberg. 2007. Evaluating pair programming with respect to system complexity and programmer expertise. *IEEE Transactions on Software Engineering* 33, 2 (2007), 65–86.

[6] Ivan Bacher, Brian Mac Namee, and John D Kelleher. 2017. Scoped: Visualising the Scope Chain Within Source Code.. In *EuroVis (Short Papers)*. 115–119.

[7] Louis-Philippe Beland and Richard Murphy. 2016. Ill communication: technology, distraction & student performance. *Labour Economics* 41 (2016), 61–76.

[8] Allen Bierbaum, Christopher Just, Patrick Hartling, Kevin Meinert, Albert Baker, and Carolina Cruz-Neira. 2001. VR Juggler: A virtual platform for virtual reality application development. In *Proceedings IEEE Virtual Reality 2001*. IEEE, 89–96.

[9] M Blok, EM De Korte, L Groenesteijn, M Formanoy, and P Vink. 2009. The effects of a task facilitating working environment on office space use, communication, concentration, collaboration, privacy and distraction. In *Proceedings of the 17th World Congress on Ergonomics (IEA 2009), 9-14 August 2009, Beijing, China*. International Ergonomics Association.

[10] Monica Bordegoni and Francesco Ferrise. 2013. Designing interaction with consumer products in a multisensory virtual reality environment. *Virtual and Physical Prototyping* 8, 1 (2013), 51–64. https://doi.org/10.1080/17452759.2012.762612 arXiv:https://doi.org/10.1080/17452759.2012.762612

[11] Joao Castelhano, Isabel C Duarte, Carlos Ferreira, Joao Duraes, Henrique Madeira, and Miguel Castelo-Branco. 2019. The role of the insula in intuitive expert bug detection in computer code: an fMRI study. *Brain imaging and behavior* 13, 3 (2019), 623–637.

[12] Robert DeLine, Mary Czerwinski, and George Robertson. 2005. Easing program comprehension by sharing navigation data. In *2005 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC'05)*. IEEE, 241–248.

[13] A. Elliott, B. Peiris, and C. Parnin. 2015. Virtual Reality in Software Engineering: Affordances, Applications, and Challenges. In *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, Vol. 2. 547–550. https://doi.org/10.1109/ICSE.2015.191

[14] Anthony Elliott, Brian Peiris, and Chris Parnin. 2015. Virtual reality in software engineering: Affordances, applications, and challenges. In *Proceedings of the 37th International Conference on Software Engineering-Volume 2*. IEEE Press, 547–550.

[15] Florian Fittkau, Alexander Krause, and Wilhelm Hasselbring. 2015. Exploring software cities in virtual reality. In *2015 ieee 3rd working conference on software visualization (vissoft)*. IEEE, 130–134.

[16] Benjamin Floyd, Tyler Santander, and Westley Weimer. 2017. Decoding the representation of code in the brain: An fMRI study of code review and expertise. In *2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE)*. IEEE, 175–186.

[17] Thomas Fritz, Andrew Begel, Sebastian C Müller, Serap Yigit-Elliott, and Manuela Züger. 2014. Using psycho-physiological measures to assess task difficulty in software development. In *Proceedings of the 36th international conference on software engineering*. 402–413.

[18] Tsuneo Furuyama, Yoshio Arai, and Kazuhiko Iio. 1996. Analysis of fault generation caused by stress during software development. In *Achieving Quality in Software*. Springer, 14–28.

[19] Judith Good and Paul Brna. 2004. Program comprehension and authentic measurement:: a scheme for analysing descriptions of programs. *International Journal of Human-Computer Studies* 61, 2 (2004), 169–185.

[20] Kelleher Guerin and Gregory D Hager. 2017. Robot control, training and collaboration in an immersive virtual reality environment. US Patent 9,643,314.

[21] Ulas Gulec, Murat Yilmaz, Veysi Isler, Rory V O'Connor, and Paul Clarke. 2018. Adopting virtual reality as a medium for software development process education. In *Proceedings of the 2018 International Conference on Software and System Process*. 71–75.

[22] Aleshia Hayes and Karen Johnson. 2019. Cultural Embodiment in Virtual Reality Education and Training: A Reflection on Representation of Diversity. In *Foundations and Trends in Smart Learning*. Springer, 93–96.

[23] Brian Hayes, Yusun Chang, and George Riley. 2018. Controlled unfair adaptive 360 VR video delivery over an MPTCP/QUIC architecture. In *2018 IEEE International Conference on Communications (ICC)*. IEEE, 1–6.

[24] M. Hollander, D.A. Wolfe, and E. Chicken. 2013. *Nonparametric Statistical Methods*. Wiley.

[25] Adrian H Hoppe, Kai Westerkamp, Sebastian Maier, Florian van de Camp, and Rainer Stiefelhagen. 2018. Multi-user Collaboration on Complex Data in Virtual and Augmented Reality. In *International Conference on Human-Computer Interaction*. Springer, 258–265.

[26] Pakiso J Khomokhoana and Liezel Nel. 2019. Decoding source code comprehension: bottlenecks experienced by senior computer science students. In *Annual Conference of the Southern African Computer Lecturers' Association*. Springer, 17–32.

[27] Michael Kircher, Prashant Jain, Angelo Corsaro, and David Levine. 2001. Distributed extreme programming. *Extreme Programming and Flexible Processes in Software Engineering, Italy* (2001), 66–71.

[28] Way Kuo, Wei-Ting Kary Chien, and Taeho Kim. 2013. *Reliability, yield, and stress burn-in: a unified approach for microelectronics systems manufacturing & software development*. Springer Science & Business Media.

[29] Masaru KOMORIb Yoshimasa KURUMI and Shigehiro MORIKAWA. 2016. Active and Passive Haptic Training Approaches in VR Laparoscopic Surgery Training. *Medicine Meets Virtual Reality 22: NextMed/MMVR22* 220 (2016), 215.

[30] Lange B George S Deutsch JE Saposnik G Laver, KE and M Crotty. 2017. Virtual reality for stroke rehabilitation. *Cochrane Database of Systematic Reviews* 11 (2017). https://doi.org/10.1002/14651858.CD008349.pub4

[31] Keith R. Lohse, Courtney G. E. Hilderman, Katharine L. Cheung, Sandy Tatla, and H. F. Machiel Van der Loos. 2014. Virtual Reality Therapy for Adults Post-Stroke: A Systematic Review and Meta-Analysis Exploring Virtual Environments and Commercial Games in Therapy. *PLOS ONE* 9, 3 (03 2014), 1–13. https://doi.org/10.1371/journal.pone.0093318

[32] Jorge Martin-Gutierrez, Jorge Martín-Gutiérrez, Carlos Efrén Mora, Beatriz Añorbe-Díaz, and Antonio González-Marrero. 2017. Learning Strategies in Engineering Education Using Virtual and Augmented Reality Technologies. *Eurasia Journal of Mathematics Science and Technology Education* 13, 2 (2017), 297–300.

[33] Leonel Merino, Alexandre Bergel, and Oscar Nierstrasz. 2018. Overcoming issues of 3D software visualization through immersive augmented reality. In *2018 IEEE Working Conference on Software Visualization (VISSOFT)*. IEEE, 54–64.

[34] Merve Denizci Nazligul, Murat Yilmaz, Ulas Gulec, Mert Ali Gozcu, Rory V O'Connor, and Paul M Clarke. 2017. Overcoming public speaking anxiety of software engineers using virtual reality exposure therapy. In *European Conference on Software Process Improvement*. Springer, 191–202.

[35] Kristine L Nowak and Frank Biocca. 2003. The effect of the agency and anthropomorphism on users' sense of telepresence, copresence, and social presence in virtual environments. *Presence: Teleoperators & Virtual Environments* 12, 5 (2003), 481–494.

[36] Roy Oberhauser and Carsten Lecon. 2017. Gamified Virtual Reality for Program Code Structure Comprehension. *International Journal of Virtual Reality* 17, 2 (2017).

[37] Gary M Olson and Judith S Olson. 2000. Distance matters. *Human–computer interaction* 15, 2-3 (2000), 139–178.

[38] Guillaume Pourchera, Daphné Micheletb, Thomas Recanzonec, Sabrina Stitic, Erwan Jolivetc, and Jessy Barréb. 2018. INTEREST OF VIRTUAL REALITY (VR) SIMULATION FOR SURGICAL LEARNING: VR SINGLE PORT SLEEVE GASTRECTOMY. In *OBESITY SURGERY*, Vol. 28. SPRINGER 233 SPRING ST, NEW YORK, NY 10013 USA, 531–531.

[39] Anett Racz and Gergo Zilizi. 2018. VR aided architecture and interior design. In *2018 International Conference on Advances in Computing and Communication Engineering (ICACCE)*. IEEE, 11–16.

[40] Ayushi Rastogi, Suresh Thummalapenta, Thomas Zimmermann, Nachiappan Nagappan, and Jacek Czerwonka. 2017. Ramp-up Journey of New Hires: Do strategic practices of software companies influence productivity?. In *Proceedings of the 10th Innovations in Software Engineering Conference*. 107–111.

[41] Paige Rodeghero, Collin McMillan, Paul W. McBurney, Nigel Bosch, and Sidney D'Mello. 2014. Improving Automated Source Code Summarization via an Eye-Tracking Study of Programmers. In *Proceedings of the 36th International Conference on Software Engineering (ICSE 2014)*. Association for Computing Machinery, New York, NY, USA, 390–401. https://doi.org/10.1145/2568225.2568247

[42] Anastasia Ruvimova, Junhyeok Kim, Thomas Fritz, Mark Hancock, and David C Shepherd. 2020. "Transport Me Away": Fostering Flow in Open Offices through Virtual Reality. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*. ACM.

[43] Mian Usman Sattar, Sellappan Palaniappan, Asiah Lokman, Nauman Shah, Usman Khalid, and Raza Hasan. 2020. Motivating Medical Students Using Virtual Reality Based Education. *International Journal of Emerging Technologies in Learning (iJET)* 15, 02 (2020), 160–174.

[44] Vibhu Saujanya Sharma, Rohit Mehra, Vikrant Kaulgud, and Sanjay Podder. 2018. An immersive future for software engineering: Avenues and approaches. In *Proceedings of the 40th International Conference on Software Engineering: New Ideas and Emerging Results*. 105–108.

[45] Janet Siegmund, Christian Kästner, Sven Apel, Chris Parnin, Anja Bethmann, Thomas Leich, Gunter Saake, and André Brechmann. 2014. Understanding understanding source code with functional magnetic resonance imaging. In *Proceedings of the 36th International Conference on Software Engineering*. 378–389.

[46] Igor Steinmacher, Tayana Uchoa Conte, Christoph Treude, and Marco Aurélio Gerosa. 2016. Overcoming open source project entry barriers with a portal for newcomers. In *Proceedings of the 38th International Conference on Software Engineering*. 273–284.

[47] Tuukka M Takala. 2014. RUIS: A toolkit for developing virtual reality applications with spatial interaction. In *Proceedings of the 2nd ACM symposium on Spatial user interaction*. ACM, 94–103.

[48] Gustav Taxén and Ambjörn Naeve. 2002. A system for exploring open issues in VR-based education. *Computers & Graphics* 26, 4 (2002), 593–598.

[49] Jozef Tvarozek, Martin Konopka, Pavol Navrat, and Maria Bielikova. 2016. Studying various source code comprehension strategies in programming education. *Eye Movements in Programming: Models to Data* 23 (2016), 25–26.

[50] Juraj Vincur, Martin Konopka, Jozef Tvarozek, Martin Hoang, and Pavol Navrat. 2017. Cubely: virtual reality block-based programming environment. In *Proceedings of the 23rd ACM Symposium on Virtual Reality Software and Technology*. 1–2.

[51] Peng Wang, Peng Wu, Jun Wang, Hung-Lin Chi, and Xiangyu Wang. 2018. A critical review of the use of virtual reality in construction engineering education and training. *International journal of environmental research and public health* 15, 6 (2018), 1204.

[52] Eliane Wiese, Anna Rafferty, and Armando Fox. 2019. Linking code readability, structure, and comprehension among novices: it's complicated. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)*. IEEE, 84–94.

[53] Laurie Williams, Robert R Kessler, Ward Cunningham, and Ron Jeffries. 2000. Strengthening the case for pair programming. *IEEE software* 17, 4 (2000), 19–25.

[54] Christine Wolfgramm, Nicole Suter, and Eva Göksel. 2016. Examining the Role of Concentration, Vocabulary and Self-concept in Listening and Reading Comprehension. *International Journal of Listening* 30, 1-2 (2016), 25–46.

[55] Christian Zirkelbach, Alexander Krause, and Wilhelm Hasselbring. 2019. Hands-on: experiencing software architecture in virtual reality. (2019).