Effectiveness of 3D Code Visualization Techniques over Time

Alice Truong
University of Zurich
Zurich, Switzerland
alice.truong@uzh.ch

Sarah Zurmühle
University of Zurich
Zurich, Switzerland
sarah.zurmuehle2@uzh.ch

ABSTRACT

KEYWORDS

3D Code Visualization, Code Park, FlyThruCode, CodeCity, Evo-Streets, Time Interval, Apache Software Foundation

ACM Reference Format:
Alice Truong and Sarah Zurmühle. 2018. Effectiveness of 3D Code Visualization Techniques over Time. In *Proceedings of Data Science for Software Engineering (DSforSE'18),* Alice Truong and Sarah Zurmühle (Eds.). ACM, New York, NY, USA, Article 4, 2 pages. https://doi.org/10.475/123_4

1 INTRODUCTION

The usage of 3D code visualization has become a frequent discussed topic. There are multiple studies that research the usage and effectiveness of such methods (e.g. [2], [5], [6]). These studies mainly study the different techniques with an experiment at one time and draw their conclusion from these results.

However, in this study we try to analyze different 3D code visualization techniques over time intervals. This allows us to measure how the efficiency of users develop over time while using a specific 3D code visualization technique. At the end, the convergence of the different techniques can be analyzed and compared.

We plan to analyze two to three different 3D code visualization techniques and one case that does not use any 3D code visualization techniques. The last case is needed to compare the 3D methods to a base case. So, we plan to conduct a field study. We use separate groups of participants for each visualization technique and the base line. Additionally, we choose five different open-source projects that the participants of our study should analyze and use to solve code comprehension tasks. The participants get one project at a time, which is the same for all participants. However, they solve the tasks using the assigned visualization technique. After they solved the tasks in the first time interval, they get another project to solve the same tasks and this continues until all open-source projects were used.

The overall goal of this study is to find out if 3D code visualization techniques can increase developers efficiency (completion time and correctness of comprehension tasks) in a long-term usage. We hypothesize that 3D visualization techniques are more effective compared to the base case over multiple iterations. So, the positive effect of such techniques do not show itself immediately, but in a long-term spectrum.

2 IMPORTANCE

3D Software Visualization provides a way to represent software structure and components. They are especially important due to its potential in helping developers to better understand software systems and algorithms.

With the results of the study we can draw conclusions about the effectiveness of the examined 3D code visualization techniques over time. If our hypothesis that the effectiveness of using these techniques increases over time is correct, this study may encourage developers to actively use the 3D visualizations. We believe that, if there is a long-term increase of effectiveness, the results can motivate developers to get to know and use certain 3D software visualizations types, even if the time they take to get familiar with the visualizations doesn't pay off right from the start.

By comparing different visualization techniques, we can additionally investigate whether their learning curves differ. We believe, that some techniques may take more time to learn, but may have a better long-term effect than other techniques. So, by examining the learning curves of the different techniques and finding different learning curves, we can provide visualization type recommendations for short-term and long-term projects.

3 3D CODE VISUALIZATION TYPES

There are different possible 3D code visualization types that could be used for our study. In this section we will present different possibilities and discuss them.

3.1 Code Park

The 3D code visualization tool *Code Park* represents code in an environment which resembles a 3D game. It tries to be easily understandable and appealing by inexperienced users, e.g. students. The code is mapped to a 3D room-like environment. On the wall of the room are all constituent parts of the class presented, like variable or member functions. The user can use code overview to interact with the code. Also, the vision of the user is in first-person view mode [1].

3.2 FlyThruCode

The 3D code visualization tool *FlyThruCode* visualizes code structures and architectures with a 3D flythrough approach. It uses 3D objects which can be modified by the user in 3D space. The user can use flythrough motions to navigate through the structures. This technique is especially useful to explore large software archidectures [3].

3.3 CodeCity

The 3D code visualization tool *CodeCity* represents software based on a city metaphor. Buildings in the city are used to visualize classes and districts the packages. The width and height of the buildings are used to encode software metrics. CodeCity maps the number of attributes for the classes on the width and length and the number of methods on the height [6].

3.4 Evo-Streets

Similar as CodeCity the visualization technique *Evo-Streets* uses a city metaphor for the visualization of software systems. Classes are also represented by buildings. However, instead of displaying packages as districts, Evo-Streets uses a hierarchical street system to represent the structure. The hierarchy of the structure is modeled in a tree structure with each street representing a certain subsystem. These subsystems can contain further subsystems which are displayed in branching streets [4].

4 OPEN-SOURCE SOFTWARE USED FOR THE FIELD STUDY

For our field study we only use open-source projects from *the Apache Software Foundation*[1]. We picked projects which have a similar purpose, namely Big Data analysis, to ensure that the study does not get biased by the type of the software.

(1) Apache Tez [2]
(2) Apache Bigtop [3]
(3) Apache REEF [4]
(4) Apache Storm [5]
(5) Apache Zeppelin [6]

REFERENCES

[1] Pooya Khaloo, Mehran Maghoumi, Eugene Taranta, David Bettner, and Joseph Laviola. 2017. Code Park: A New 3D Code Visualization Tool. In *Software Visualization (VISSOFT), 2017 IEEE Working Conference on.* IEEE, 43–53.

[2] Leonel Merino, Alexandre Bergel, and Oscar Nierstrasz. 2018. Overcoming issues of 3D software visualization through immersive augmented reality. *Proc. of VISSOFT, page in review. IEEE* (2018).

[3] Carsten Lecon Roy Oberhauser, Christian Silfang. 2016. Code structure visualization using 3D-flythrough. In *2016 11th International Conference on Computer Science Education (ICCSE).* IEEE, 365–370. https://doi.org/10.1109/ICCSE.2016.7581608

[4] Frank Steinbrückner and Claus Lewerentz. 2010. Representing Development History in Software Cities. In *Proceedings of the 5th International Symposium on Software Visualization (SOFTVIS '10).* ACM, New York, NY, USA, 193–202. https://doi.org/10.1145/1879211.1879239

[5] Richard Wettel and Michele Lanza. 2007. Program comprehension through software habitability. In *Program Comprehension, 2007. ICPC'07. 15th IEEE International Conference on.* IEEE, 231–240.

[6] Richard Wettel, Michele Lanza, and Romain Robbes. 2011. Software systems as cities: A controlled experiment. In *Software Engineering (ICSE), 2011 33rd International Conference on.* IEEE, 551–560.

[1]https://www.apache.org
[2]https://tez.apache.org
[3]http://bigtop.apache.org
[4]http://reef.apache.org
[5]http://reef.apache.org
[6]https://zeppelin.apache.org