Enhancing CodeCity: Code Evolution in VR using Web Technologies

David Moreno-Lumbreras[1,*,1], Jesus M. Gonzalez-Barahona[1,1] and Gregorio Robles[1,1]

[1]EIF @ Universidad Rey Juan Carlos, Spain

Abstract

CodeCity is a metaphor to visualize software metrics using a city layout. It was first implemented by CodeCity in 2007, and several other implementations have extended it.

The goal is to enhance CodeCity with dynamic visualization of code evolution, decoupling of data extraction and visualization mechanisms, and immersion in Virtual Reality (VR) environment using web technologies. For managing the city's layout, we rely on the spiral algorithm as it efficiently accommodates the introduction and removal of new elements while maintaining the visualization's overall layout.

VR immersion is enabled using WebVR and WebXR standards, and evolution metrics are computed through a toolchain that mines data from git repositories. Data decoupling is achieved by producing JSON documents that are consumed by *BabiaXR* components. Visualization of evolution is supported by a navigation bar for switching over code releases.

The implementation of CodeCity allows dynamic visualization of code evolution and immersive interaction with the scene. It works in both 2D-simulated 3D and immersive 3D VR scenes.

The implementation of CodeCity with dynamic visualization of code evolution and immersion in VR environment using web technologies has potential to improve software visualization. However, future work should evaluate its effectiveness compared to other tools such as GitHub web UI. Further opportunities to extend the CodeCity metaphor should be explored, such as incorporating more software metrics or enabling collaborative exploration of the codebase.

Keywords

code city metaphor, software evolution, software visualization, CodeCity, data visualization, virtual reality, web, 3D

1. Introduction

The city metaphor is probably the most successful visualization for source code metrics in a 3D environment. It was proposed for the first time by Munro *et al.* [1], and had its greatest exponent with the approach presented by Wettel *et al.*, which described CodeCity, its first implementation. The original CodeCity creates cities that look real, due to the combination of layouts, topologies, and metric mappings applied at an appropriate level of granularity. It shows object-oriented software systems as cities that can be intuitively explored [2], by mapping metrics to features (height, size, color) of the buildings, and placing those buildings in locations related to their position in the object hierarchy. The city metaphor offers a clear notion of locality, supporting orientation, and does not hide the underlying structural complexity that cannot be oversimplified.

SATToSE'23

*Corresponding author.

david.morenolu@urjc.es (D. Moreno-Lumbreras);
jesus.gonzalez.barahona@urjc.es (J. M. Gonzalez-Barahona);
gregorio.robles@urjc.es (G. Robles)
https://dlumbrer.github.io/ (D. Moreno-Lumbreras);
http://gsyc.es/~jgb (J. M. Gonzalez-Barahona);
http://gsyc.urjc.es/~grex (G. Robles)
0000-0002-5454-7808 (D. Moreno-Lumbreras);
0000-0001-9682-460X (J. M. Gonzalez-Barahona);
0000-0002-1442-6761 (G. Robles)

The city metaphor has been widely used in software visualization to represent the different components of a software system as buildings and roads. One important aspect of this metaphor is the location algorithm used to place the buildings on the map. Treemaps have emerged as a popular choice for this purpose, providing a hierarchical view of the software components that can be easily navigated by the user. In fact, the majority of city metaphor tools in software visualization use treemaps for this purpose. The use of treemaps has proven to be effective in helping users understand complex software structures, as well as in identifying potential bottlenecks and areas for improvement in the code. As such, treemaps have become an important tool in the software visualization community, helping to make software systems more accessible and understandable to a wider audience.

While treemaps are widely used as a layout algorithm for the city metaphor visualization in software visualization, one of the main drawbacks of using treemaps for representing the evolution of software systems is the difficulty of adding or removing items in the treemap [3]. When new components or modules are added to the system or when some of them are removed, the treemap layout has to be recalculated, which can be a computationally expensive process and the layout changes dramatically. This is because the treemap algorithm has to optimize the layout to ensure that the areas of the rectangles representing the components are proportional to their sizes. As a result, adding or removing items can

cause significant changes in the layout, which may make it difficult for users to understand the new version of the software system. In addition, the rearrangement of components in the treemap can affect the visualizations of the relationships between components, making it more difficult to identify the structure and organization of the system. Therefore, while treemaps have proven to be an effective layout algorithm for representing software systems in the form of a city metaphor, their limitations in dealing with changes in the system may need to be considered when using them as the main visualization tool for software evolution.

On the other hand, the spiral algorithm has been proposed as a viable alternative for representing the evolution of software systems. Unlike treemaps, spiral layouts are able to represent changes in the number of elements without requiring major changes in the visualization. In this way, the spiral algorithm can be used as a more scalable solution for representing the evolution of software systems, particularly when dealing with dynamic changes in the number of software elements.

Recent advances in technology, such as virtual reality (VR) in web technologies, have created new opportunities for software visualization researchers to explore the city metaphor in new ways. These technologies can provide a more immersive and interactive experience for users, allowing them to explore complex software systems in a more intuitive and engaging manner. VR technology can offer users a more realistic and immersive experience, allowing them to move around and interact with the software system as if they were physically present within it. Web technologies can also offer new possibilities for software visualization, as they allow researchers to develop web-based tools that can be easily accessed and used by a wider audience. This can help to democratize access to software visualization tools, making them more widely available and accessible to researchers and practitioners alike.

The structure of the paper is as follows. In Section 2, we present the state of the art and some related work. Then, in Section 3, we show the details of our implementation in *BabiaXR*. After that, in Section 4 we discuss about the approach and other aspects and the future validation for this work in progress. Finally we detail some conclusions (Section 5), and details about the replication package and acknowledgments.

2. Related Works

2.1. *CodeCity*

The city metaphor was initially introduced through the implementation of *Software World* [1], which represented software systems as buildings in a city. Subsequently, various approaches were explored to assist developers in comprehending software systems. Panas *et al.* [4, 5, 6] proposed a software city that exhibited information about static and dynamic data, and Marcus *et al.* [7] proposed a city-like software visualization. Another visualization tool, *Verso* [8], was based on landscapes, but was influenced by the city metaphor.

In 2007 *CodeCity* was presented [2], raising the approach to a new level implementation-wise. Figure 1 shows the original *CodeCity* application, including the interface for interacting with the city and the metrics details.

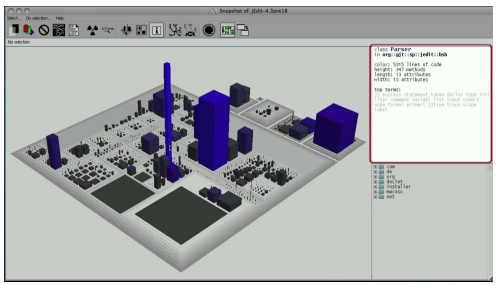


Figure 1: The Original *CodeCity* Tool

CodeCity demonstrated its effectiveness not only in program comprehension [2], but also in software evolution analysis [9] and design problem analysis [10], which led to the emergence of numerous tools and approaches based on the same metaphor. These tools differ slightly in their visualizations, highlighting the flexibility and versatility of the metaphor.

Scarsbrook *et al.* [11] developed a tool that visualizes and debugs large-scale JavaScript program structure using treemaps. Brito *et al.* [12] presented a similar approach, but for the Go programming language. Steinbr"uckner *et al.* [13] proposed an alternative city layout based on streets and sub-streets for the tree structure, which allows for observing the time evolution of the software system. Gamification has also been employed in combination with the city metaphor for software comprehension tasks in *CodeMetropolis* [14], which uses the Minecraft game engine. *M3tricity* [15], a recent re-implementation of *CodeCity* by the original research group, is a web application that visualizes software systems as evolving cities, treating evolution as a first-class concept.

2.2. From 3D to Virtual Reality

One of the early explorations of using VR for visualizing software was done by Young and Munro [16], at the time quite a technical feat. Another early VR-based approach is *Imsovision* [17], which focuses on C++, defining

some metrics that nowadays are still used in the literature. More recently, thanks to technological advances, software visualizations based in VR become an active field of research.

Fittkau *et al.* proposed a VR implementation of *ExplorViz* [18], based on the first versions of *WebVR*, focusing on the runtime and static characteristics of object-oriented programming software systems. Vincur *et al.* [19, 20] proposed a VR city for analyzing object-oriented software. Steinbrückner and Lewerentz [21] proposed stable city layouts for evolving software systems, using layouts other than treemaps. *Getaviz* [22] also uses the city metaphor to generate structural, behavioral, and evolutionary views of software systems for empirical evaluation.

CityVR [23], developed with *Unity3D*, provides the same metrics as the original *CodeCity*, adding interactions using the gaze of the user in the VR headset, and its controllers. The technique we used for moving in the scene technique has similarities with their approach.

Capece *et al.* [24] visualized Java systems with the *Unreal Engine 4* using the city metaphor in VR. Most of the current approaches are based on non-web technologies (*i.e.,* Unreal Engine or Unity), which differentiates them from our approach, since *BabiaXR* is web-based, allowing the visualization of the city in any modern browser.

Other metaphors have been implemented in VR. Misiak, Schreiber *et al.* [25] proposed the island metaphor for visualizing OSGi-based software systems, introducing and emphasizing the visualization of dependencies. Schreiber *et al.* [26] presented an interactive tool that also visualizes OSGi-based systems with their components, packages, services, and dependencies in 3D, using a different metaphor including boxes.

We also conducted, under *BabiaXR* [27], experiments validating the use of the city metaphor in VR [28, 29].

3. Method

In this section we present *BabiaXR*, our version of *CodeCity* using a positioning algorithm for buildings and quarters and our version, still in progress, of the temporal evolution of the city.

3.1. *BabiaXR* in a nutshell

BabiaXR-CodeCity is a part of a larger open-source toolset called *BabiaXR*. The *BabiaXR* toolset is designed for 3D data visualization in the browser and is available from the npm repository. *BabiaXR* is based on a framework called *A-Frame*, which allows for the creation of 3D, augmented reality, and virtual reality experiences in the browser. One of the key advantages of using *A-Frame* is that it extends HTML with new entities, allowing for the description of 3D scenes as a part of an HTML document. These scenes can then be manipulated using techniques common to any front-end web developer.

In turn, *A-Frame* is built on top of *Three.js*, which uses the *WebGL* API provided by modern browsers. *BabiaXR* extends *A-Frame* by providing components to create visualizations, retrieve data, and manage it with filtering capabilities. Visualizations created with *BabiaXR* can be displayed on-screen or on VR devices using the facilities provided by browsers. The versatility of *BabiaXR*, its ability to run in web browsers, and its integration with emerging technologies such as virtual reality and web technologies make it an exciting tool for software visualization. A sample scene created with *BabiaXR* is shown in Figure 2.

Figure 2: Example of a *BabiaXR* Scene

3.2. *BabiaXR* CodeCity

Our implementation of *CodeCity* utilizes a similar approach to the original version in that it maps software metrics to the features of buildings and groups them according to the hierarchical structure of the codebase. However, there are notable differences between our implementation and the original. While the original *CodeCity* was a desktop-based application written in *Smalltalk*, our implementation is built as a browser-based application using *JavaScript*. Additionally, the original version utilized a treemap algorithm for layout, whereas we use a *spiraling algorithm*. With this algorithm, the first building is placed at the center of a spiral and the subsequent buildings are positioned in a spiraling manner around it. This same layout approach is also used for groups of buildings at the same hierarchical level. To better illustrate this approach, Figure 3 provides an example of the layout produced by our implementation of *CodeCity*.

The spiral pattern expands in a clockwise direction, starting from the green quarter located in the center-left of the Figure, and progressing to the next green quarter on its right, followed by the one below it, and continuing with some buildings towards the left, and so on.

The spiral algorithm proves to be more effective in representing the evolution of a project when new elements,

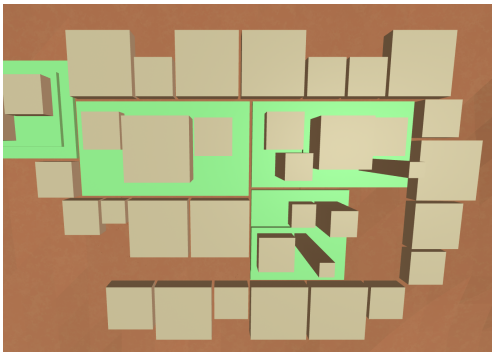


Figure 3: Example of a *BabiaXR-CodeCity* Scene

such as buildings, emerge and disappear in any place in the hierarchy. In contrast, rectangle packing may cause existing buildings to shift to accommodate new ones, thereby disrupting the perceivable relationship between the buildings' prior and new locations.

BabiaXR-CodeCity utilizes software metrics to map features of the buildings. Each building is associated with a file, and each district corresponds to a directory that can contain files and subdirectories. *BabiaXR-CodeCity* facilitates the mapping of any metric available to either the base's area, height, or color of each building.

BabiaXR-CodeCity scenes are interactive, enabling users to explore the city using cursor keys in the on-screen mode, and by walking in the physical world in VR. Users can also view data about buildings of interest. In the on-screen mode, hovering the cursor over a building opens a tooltip displaying the file name and metric values such as the number of functions, lines of code, and cyclomatic complexity [30]. In VR, the same feature is facilitated by the raycaster provided by the VR controller, which is the Oculus Quest 2 pointing mechanism.

Figure 4 summarizes the complete workflow to produce a scene with *BabiaXR-CodeCity* starting from a source code snapshot in a *Git* repository.

3.3. Time evolution

The virtual reality scene of the city features a highly interactive navigation bar, which can be easily accessed and utilized with the virtual reality device's controllers. The navigation bar is prominently displayed on the user interface, as depicted in Figure 5. It consists of a horizontal line, displaying the active time instant, along with a series of interaction buttons placed below to enable users to move forward or backward in time. By default, the navigation bar automatically advances the scene every five seconds, from the past to the future, though this default setting is fully customizable. Users can easily configure the direction and speed of movement between instants

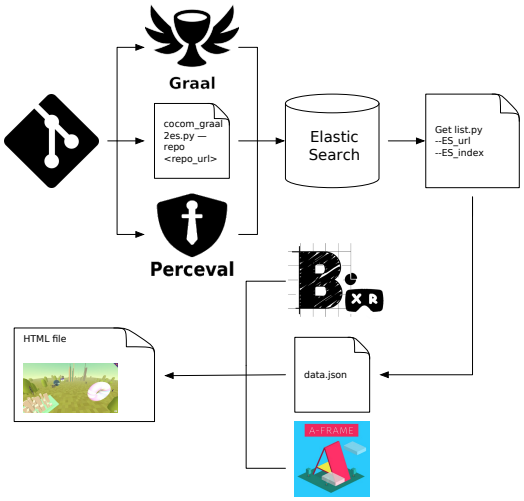


Figure 4: *BabiaXR* Workflow: From Source Code to a Scene

using the two vertical bars placed on the right-hand side of the navigation bar. Additionally, users can also make sudden and non-consecutive jumps to any point in time using the buttons below or by simply interacting with the timeline on the navigation bar itself.

As the user interacts with the navigation bar to change the time instant, the city undergoes a seamless transformation, with buildings and quarters gracefully shifting positions to match the new timeframe. To ensure that the user's perception remains uninterrupted, any new structures that appear momentarily become transparent, while disappearing buildings and quarters are promptly removed from view. This fluid motion enables the user to keep a clear and coherent understanding of the evolving city.

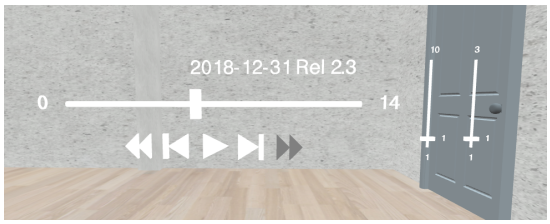


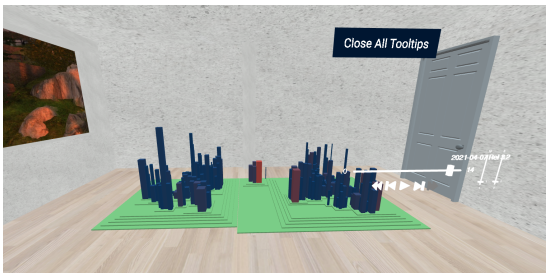
Figure 5: Navigation bar for time evolution

The design of this navigation bar has been carefully selected based on common types of navigation bars used for multimedia players, such as video or sound players. The aim of the design is to provide users with a user-friendly and intuitive interface, that is both easy to use and quick to learn. By providing users with these powerful tools to control the evolution of the city, *BabiaXR-CodeCity*

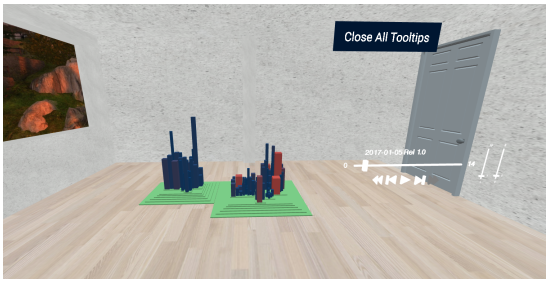
offers a highly interactive and customizable experience that encourages exploration and experimentation with the data.

3.4. Use case

In this first work in porgress use case, the scene allows the user to interact with a *CodeCity* visualization of the source code of *JetUML*,[1].



(a) Release 3.2 of *JetUML*



(b) Release 1.0 of *JetUML*

Figure 6: The *JetUML* in two different releases

The *JetUML* city, in all its releases from 1.0 to 3.2, requires the user to utilize the navigation bar and controls of the Quest 2 in order to navigate through its various scenes. By default, the city evolves from past to future automatically, with the layout, buildings, and quarters adjusting accordingly as the displayed release is changed. However, users also have the ability to advance from future to past, as well as configure the speed at which the changes between releases occur. Furthermore, users can make sudden changes between non-consecutive releases, enabling them to compare and contrast between two distant releases of the city. This level of flexibility and interactivity adds to the overall immersive experience and utility of the *JetUML* city.

Mapping of metrics. In both of our experiments we use a mapping of metrics to features of the buildings which are similar to those in the original *CodeCity* implementation. Each building represents a file. The area of its base corresponds to the number of functions (`num_funs`), its height to the lines of code per function (`loc_per_function`), and its color the Cyclomatic Complexity Number (CCN, in a blue to red scale. Therefore, the volume of the building represents the number of lines of code (LOC) of the file.

There is a replication package available with all the needed steps to deploy and see this scene [2]

4. Discussion

The city metaphor has been widely used in software visualization due to its ability to represent complex software systems in a more intuitive and understandable way. However, the traditional 2D representations of the city metaphor may have limitations in fully conveying the complexity of software systems. This is where Virtual Reality (VR) can make a significant difference [28]. By immersing the user in a 3D environment where they can interact with the city metaphor, they can gain a better understanding of the software system's structure, behavior, and evolution. In addition, VR can provide a more engaging and interactive experience that enhances the user's learning and exploration process. Therefore, we believe that the city metaphor could be even better in VR, and we encourage further research in this area to explore the potential benefits of VR for software visualization.

The implementation of the time evolution of the city metaphor in *BabiaXR-CodeCity* could show results in terms of its ability to represent software evolution over time. One important aspect of this implementation is the use of the spiral algorithm for the layout of the city. We believe that the use of the spiral algorithm could be particularly effective in representing the evolution of software projects, where new elements are constantly being added and removed. The spiral layout allows for new buildings to be added anywhere in the hierarchy without forcing all existing buildings to move, as is the case with rectangle packing. This results in a more perceptible relationship between the location of a building and its place in the hierarchy, which is important for understanding the evolution of a software project over time. However, further research and validation is needed to determine the effectiveness of the spiral algorithm for software visualization, particularly in comparison to other layout algorithms.

Moreover, the use of web technologies in the field of software visualization and virtual reality is becoming increasingly common. Web-based tools have the advantage of being easily accessible from any device with a modern web browser, without the need for additional software or hardware. This accessibility opens up opportunities for wider dissemination and collaboration

[1] *JetUML*: https://github.com/prmr/JetMUL

[2] https://doi.org/10.5281/zenodo.7828691

in the field. Furthermore, the use of web technologies can facilitate the integration of different visualization techniques and tools, which can enhance the capabilities of software visualization systems. In the case of our approach, the use of web technologies has allowed us to develop a web-based VR tool that can be used with the Quest 2, a widely accessible VR device. Additionally, the use of web technologies has enabled us to easily share and disseminate the tool with the community, which can help to foster collaboration and facilitate the validation and improvement of our approach.

5. Conclusions and Future Work

In this study, we have presented a novel approach for the temporal evolution of the visualization of software systems using *BabiaXR-CodeCity* as a city metaphor. While this approach is still in its early stages of development and represents the first version of our methodology, we have observed promising results and believe that further effort and validation are necessary to refine and improve our approach.

To demonstrate the potential of our approach, we have provided a detailed use case example of a real software city and its evolution over time with successive releases. Our approach is unique in that it allows for the mapping of software metrics to building features, enabling the user to interactively explore the city and visualize data about specific buildings of interest.

As a next step in our research, we plan to validate our approach by comparing it to existing tools that showcase similar metrics and evolution. Additionally, we intend to conduct user studies with practitioners in the software development field to obtain feedback on the usefulness and usability of our approach. Ultimately, we aim to refine and optimize our methodology to provide an effective and efficient tool for software visualization and analysis.

In adition, it is important to further investigate the effectiveness of the spiral algorithm utilized in this approach. While the spiral algorithm has demonstrated advantages in representing the evolution of a project with new elements, it is necessary to evaluate its performance in comparison to other layout algorithms in different scenarios. This evaluation can be carried out by measuring various metrics such as visual perception, scalability, and usability. Furthermore, it would be valuable to conduct user studies to assess how the spiral algorithm impacts users' cognitive load and understanding of the temporal evolution of the software system being visualized. Such an investigation would ultimately contribute to the refinement and improvement of the *BabiaXR-CodeCity* approach, making it a more effective and practical tool for software visualization.

Acknowledgments

We acknowledge the financial support of the Community of Madrid for the project IND2018/TIC-9669, and the Spanish Government for the project RTI-2018-101963-B-100, and of the Madrid Regional Government (e-Madrid-CM - P2018/TCS-4307), co-financed by EU Structural Funds (FSE and FEDER).

References

[1] C. Knight, M. Munro, Comprehension with[in] virtual environment visualisations, in: Proceedings Seventh International Workshop on Program Comprehension, 1999, pp. 4–11. doi:10.1109/WPC.1999.777733.

[2] R. Wettel, M. Lanza, Program comprehension through software habitability, in: 15th IEEE International Conference on Program Comprehension (ICPC '07), 2007, pp. 231–240. doi:10.1109/ICPC.2007.30.

[3] W. Scheibel, D. Limberger, J. Döllner, Survey of treemap layout algorithms, in: Proceedings of the 13th International Symposium on Visual Information Communication and Interaction, VINCI '20, Association for Computing Machinery, New York, NY, USA, 2020. URL: https://doi.org/10.1145/3430036.3430041. doi:10.1145/3430036.3430041.

[4] T. Panas, R. Berrigan, J. Grundy, A 3D metaphor for software production visualization, in: Proceedings on Seventh International Conference on Information Visualization, 2003. IV 2003., 2003, pp. 314–319. doi:10.1109/IV.2003.1217996.

[5] T. Panas, R. Lincke, W. Löwe, Online-configuration of software visualizations with vizz3d, in: Proceedings of the 2005 ACM Symposium on Software Visualization, SoftVis '05, Association for Computing Machinery, New York, NY, USA, 2005, p. 173–182. URL: https://doi.org/10.1145/1056018.1056043. doi:10.1145/1056018.1056043.

[6] T. Panas, T. Epperly, D. Quinlan, A. Saebjornsen, R. Vuduc, Communicating software architecture using a unified single-view visualization, in: 12th IEEE International Conference on Engineering Complex Computer Systems (ICECCS 2007), 2007, pp. 217–228. doi:10.1109/ICECCS.2007.20.

[7] A. Marcus, L. Feng, J. I. Maletic, 3D representations for software visualization, in: Proceedings of the 2003 ACM Symposium on Software Visualization, SoftVis '03, Association for Computing Machinery, New York, NY, USA, 2003, p. 27–ff. URL: https://doi.org/10.1145/774833.774837. doi:10.1145/774833.774837.

[8] G. Langelier, H. Sahraoui, P. Poulin, Visualization-

based analysis of quality for large-scale software systems, in: Proceedings of the 20th IEEE/ACM International Conference on Automated Software Engineering, ASE '05, Association for Computing Machinery, New York, NY, USA, 2005, p. 214–223. URL: https://doi.org/10.1145/1101908.1101941. doi:10.1145/1101908.1101941.

[9] R. Wettel, M. Lanza, Visual exploration of large-scale system evolution, in: 2008 15th Working Conference on Reverse Engineering, 2008, pp. 219–228.

[10] R. Wettel, M. Lanza, Visually localizing design problems with disharmony maps, in: Proceedings of the 4th ACM Symposium on Software Visualization, SoftVis '08, Association for Computing Machinery, New York, NY, USA, 2008, p. 155–164. doi:10.1145/1409720.1409745.

[11] J. D. Scarsbrook, R. K. L. Ko, B. Rogers, D. Bainbridge, MetropolJS: Visualizing and debugging large-scale JavaScript program structure with treemaps, in: 2018 IEEE/ACM 26th International Conference on Program Comprehension (ICPC), 2018, pp. 389–3893.

[12] R. Brito, A. Brito, G. Brito, M. T. Valente, GoCity: Code city for Go, in: 2019 IEEE 26th International Conference on Software Analysis, Evolution and Reengineering (SANER), 2019, pp. 649–653.

[13] F. Steinbrückner, C. Lewerentz, Representing development history in software cities, in: Proceedings of the 5th International Symposium on Software Visualization, SOFTVIS '10, Association for Computing Machinery, New York, NY, USA, 2010, p. 193–202. doi:10.1145/1879211.1879239.

[14] G. Balogh, T. Gergely, A. Beszedes, T. Gyimothy, Using the city metaphor for visualizing test-related metrics, in: 2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER), volume 2, 2016, pp. 17–20. doi:10.1109/SANER.2016.48.

[15] F. Pfahler, R. Minelli, C. Nagy, M. Lanza, Visualizing evolving software cities, in: 2020 Working Conference on Software Visualization (VISSOFT), 2020, pp. 22–26. doi:10.1109/VISSOFT51673.2020.00007.

[16] P. Young, M. Munro, Visualising software in virtual reality, in: Proceedings. 6th International Workshop on Program Comprehension. IWPC'98 (Cat. No.98TB100242), 1998, pp. 19–26. doi:10.1109/WPC.1998.693276.

[17] J. Maletic, J. Leigh, A. Marcus, G. Dunlap, Visualizing object-oriented software in virtual reality, in: Proceedings 9th International Workshop on Program Comprehension. IWPC 2001, 2001, pp. 26–35. doi:10.1109/WPC.2001.921711.

[18] F. Fittkau, A. Krause, 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.

[19] J. Vincur, I. Polasek, P. Navrat, Searching and exploring software repositories in virtual reality, in: Proceedings of the 23rd ACM Symposium on Virtual Reality Software and Technology, VRST '17, Association for Computing Machinery, New York, NY, USA, 2017. doi:10.1145/3139131.3141209.

[20] J. Vincur, P. Navrat, 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.

[21] F. Steinbrückner, C. Lewerentz, Representing development history in software cities, in: Proceedings of the 5th International Symposium on Software Visualization, SOFTVIS '10, Association for Computing Machinery, New York, NY, USA, 2010, p. 193–202. doi:10.1145/1879211.1879239.

[22] D. Baum, J. Schilbach, P. Kovacs, U. Eisenecker, R. Müller, GETAVIZ: Generating structural, behavioral, and evolutionary views of software systems for empirical evaluation, in: 2017 IEEE Working Conference on Software Visualization (VISSOFT), 2017, pp. 114–118. doi:10.1109/VISSOFT.2017.12.

[23] L. Merino, M. Ghafari, C. Anslow, 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.

[24] N. Capece, U. Erra, S. Romano, G. Scanniello, Visualising a software system as a city through virtual reality, in: L. T. De Paolis, P. Bourdot, A. Mongelli (Eds.), Augmented Reality, Virtual Reality, and Computer Graphics, Springer International Publishing, Cham, 2017, pp. 319–327.

[25] M. Misiak, A. Schreiber, A. Fuhrmann, S. Zur, D. Seider, L. Nafeie, IslandViz: A tool for visualizing modular software systems in virtual reality, in: 2018 IEEE Working Conference on Software Visualization (VISSOFT), 2018, pp. 112–116. doi:10.1109/VISSOFT.2018.00020.

[26] A. Schreiber, M. Brüggemann, Interactive visualization of software components with virtual reality headsets, in: 2017 IEEE Working Conference on Software Visualization (VISSOFT), 2017, pp. 119–123. doi:10.1109/VISSOFT.2017.20.

[27] D. Moreno-Lumbreras, J. M. Gonzalez-Barahona, A. Villaverde, BabiaXR: Virtual reality software data visualizations for the web, in: 2022 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW), IEEE, 2022, pp. 71–74.

[28] D. Moreno-Lumbreras, R. Minelli, A. Villaverde,

J. M. Gonzalez-Barahona, M. Lanza, Codecity: A comparison of on-screen and virtual reality, Information and Software Technology 153 (2023) 107064. URL: https://www.sciencedirect.com/science/article/pii/S0950584922001732. doi:https://doi.org/10.1016/j.infsof.2022.107064.

[29] D. Moreno-Lumbreras, R. Minelli, A. Villaverde, J. M. Gonzalez-Barahona, M. Lanza, CodeCity: On-screen or in virtual reality?, in: Working Conference on Software Visualization, VISSOFT 2021, Luxembourg, September 27-28, 2021, IEEE, 2021, pp. 12–22. URL: https://doi.org/10.1109/VISSOFT52517.2021.00011. doi:10.1109/VISSOFT52517.2021.00011.

[30] Harrison, Magel, Kluczny, DeKock, Applying software complexity metrics to program maintenance, Computer 15 (1982) 65–79. doi:10.1109/MC.1982.1654138.