VR-Git:
Git Repository Visualization and Immersion in Virtual Reality

Roy Oberhauser
Aalen University
Germany

Based on the conference paper in ICSEA 2022:
"VR-Git: Git Repository Visualization and Immersion in Virtual Reality"

Presenter: Roy Oberhauser

- Worked for 14 years in the software industry in the Silicon Valley and in Germany doing research and development.
- Since 2004 he has been a Professor of Computer Science at Aalen University in Germany, teaching in the areas of software engineering.
- His research interest is to leverage technologies and techniques to innovate, automate, support, and improve the production and quality of software for society.

Contents

- Challenge / Problem
- Solution
- Implementation
- Evaluation
- Conclusion

Challenge / Problem

- The increasing demand for software functionality necessitates an increasing amount of program source code that is retained and managed in version control systems such as Git.
- As the number, size, and complexity of Git repositories increases, so do the number of collaborating developers, maintainers, and other stakeholders over a repository's lifetime.
- In particular, visual limitations of Git tooling hampers repository comprehension, analysis, and collaboration across one or multiple repositories with a larger stakeholder spectrum. Possible scenarios include:
  - someone transferred to the development team (ramp-up),
  - joining an open-source code project,
  - quality assurance activities,
  - forensic or intellectual property analysis,
  - maintenance activities,
  - defect or resolution tracking,
  - repository fork analysis,
  - etc.

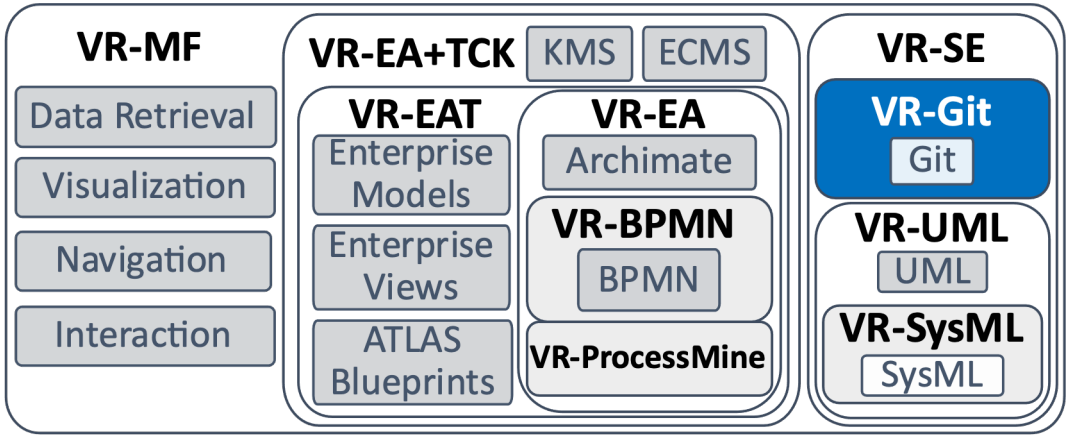
Solution Approach

- Virtual Reality (VR) provides an unlimited immersive space for visualizing and analyzing a growing and complex set of system models and their interrelationships simultaneously in a 3D spatial structure viewable from different perspectives.
- As repository models grow in size and complexity, an immersive digital environment provides additional visualization capabilities to comprehend and analyze code repositories and include and collaborate with a larger spectrum of stakeholders.

VR-Git Solution Concept in Relation to our other Solutions

- VR-Git is based on our generalized VR Modeling Framework (VR-MF) (detailed in [7]). VR-MF provides a VR-based domain-independent hypermodeling framework addressing four aspects requiring special attention when modeling in VR:
  - visualization, navigation, interaction, and data retrieval.

- Our VR-SE area includes VR-Git and the aforementioned VR-UML [5] and VR-SysML [6].
- Since Enterprise Architecture (EA) can encompass SE models and development and be applicable for collaboration in VR, our other VR modeling solutions in the EA area include: VR-EA [7], VR-ProcessMine [8], and VR-BPMN [9].
- VR-EAT [10] integrates the EA tool Atlas to provide dynamically-generated EA diagrams in VR,
- VR-EA+TCK [10] integrates Knowledge Management Systems (KMS) and/or Enterprise Content Management Systems (ECMS).

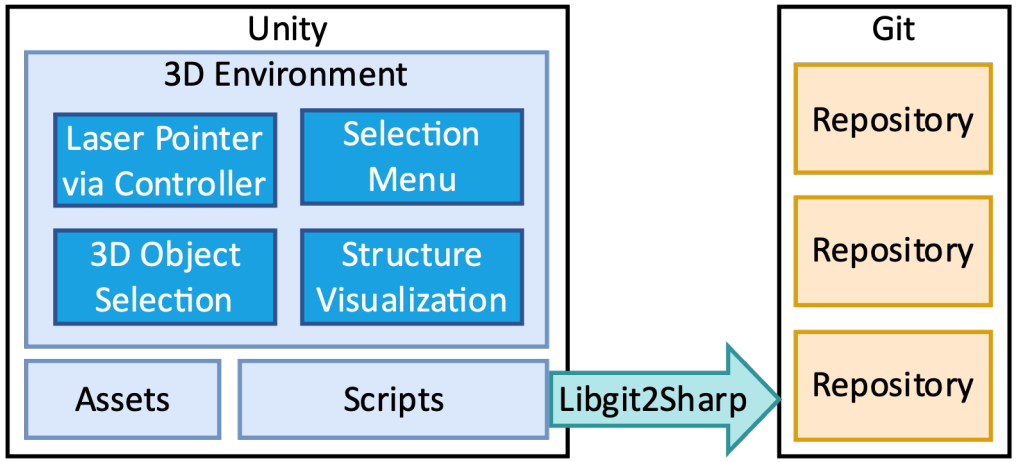


VR-Git Solution Concept

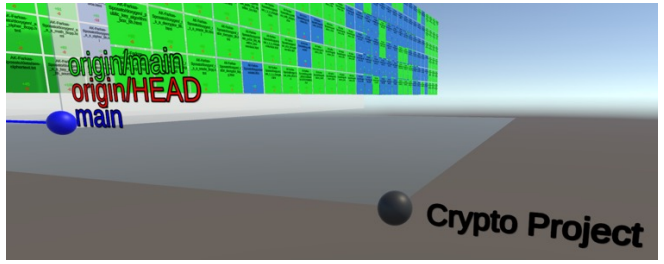
- Visualization in VR
  - A hyperplane is used to intuitively represent and group the commits related to a repository.
  - Each commit is then represented by a vertical *commit plane*.
  - These commit planes are then sequenced chronologically on the hyperplane as a set of planes.
  - Since VR space is unlimited, we can thus convey the sequence of all commits in the repository.
  - Each 2D plane then represents each file involved in that commit as a tile.
    - These are then colored to be able to quickly determine what occurred. Green indicates a file was added, red a file removed, and blue that a file was modified.
  - On the left side of the hyperplane a transparent *branch plane* (branch perspective) perpendicular to the hyperplane and the commit planes depicts branches as an acyclic colored graph to indicate which branch is involved with a commit.
  - In accordance, the commit planes are also slightly offset in height since they dock to a branch, thus “deeper” or “higher” commits indicate how close or far they were relatively from the main branch.
  - Via the anchor, commit planes can be manually collapsed (hidden), expanded, or moved to, for example, compare one commit with another side-by-side.
  - In order to view the contents of a file, when a file tile is selected, a *content plane* (i.e., code view) extends above the commit plane to display the file contents.
- Navigation in VR
  - Besides flythrough, also teleporting (e.g., to a specific commit) offered to reduce likelihood of potential VR sickness symptoms
- Interaction in VR
  - Since interaction with VR elements has not yet become standardized or intuitive, in our VR concept, user-element interaction is handled primarily via the VR controllers and a virtual tablet.
  - VR-Tablet provides detailed context-specific element information, and can provide a virtual keyboard for text entry fields (via laser pointer key selection) when needed.
  - Anchor affordance as ball on corner of hyperplane and plates can be used for moving or collapsing/expanding it.

VR-Git Realization

- Unity 2020.3 and the OpenVR XR Plugin 1.1.4
- Libgit2Sharp [19] is to access the Git commit history of one or more repositories from within Unity.

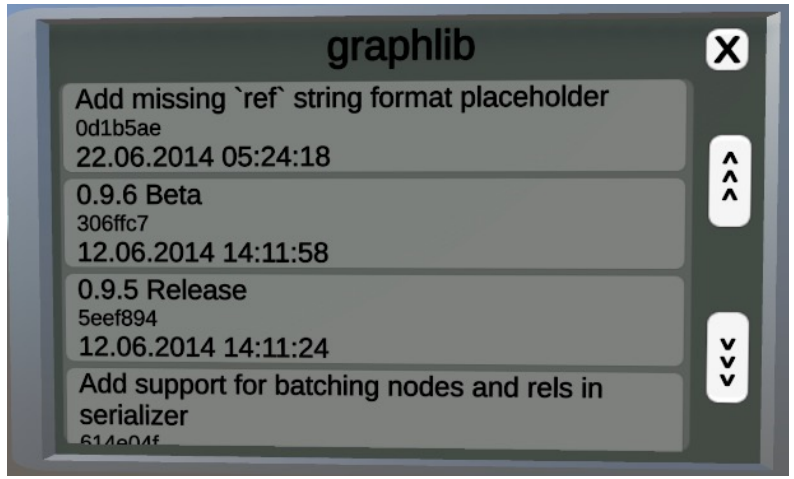


Anchor affordance as ball on corner of hyperplane and plates can be used for moving or collapsing/expanding it.



Project list in VR-Tablet

Git commit messages in VR-Tablet



Evaluation

- The evaluation of our solution concept is based on the design science method and principles [20], in particular, a viable artifact, problem relevance, and design evaluation (utility, quality, efficacy).
- We use a case study based on common Git repository comprehension and analysis scenarios:
  - branch analysis
  - commit analysis
  - multi-repository analysis
- Various git repositories were used to evaluate the prototype.

Evaluation: Branch Analysis Scenario

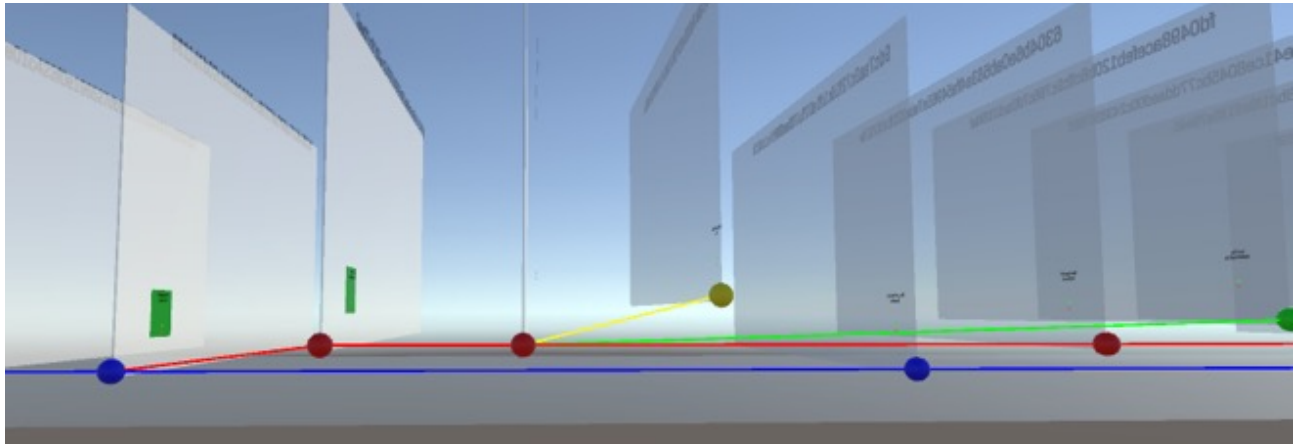
- To support branch analysis, at the front of the hyperplane oriented to the left side, an invisible branch graph plane is rendered perpendicular to the hyperplane and a color-coded list of all the branches can be seen next to the first commit plane.
- These colored labels can be used for orientation.
- By selecting a branch label, one can be teleported to the first commit of that branch.
- The branches could also be referenced in the VR-Tablet in case one forgets, and can be used for teleporting as well.
- We chose to repeat the branch labels throughout the graph to reduce the textual visual clutter

Branch overview

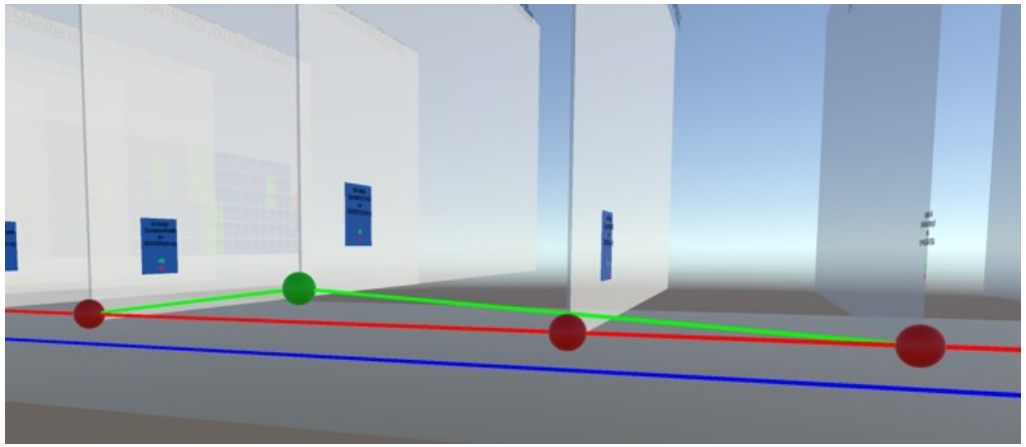
Evaluation: Branch Analysis Scenario

- The branch perspective of the hyperplane shows a contiguous color-coded graph of the branches, with commit plane heights offset based on the branch to which they are associated.
- This can provide a quick visual cue as to how relatively close or far the commit is from the main branch.
- A merge of two branches is shown on the right

Branch tree graph

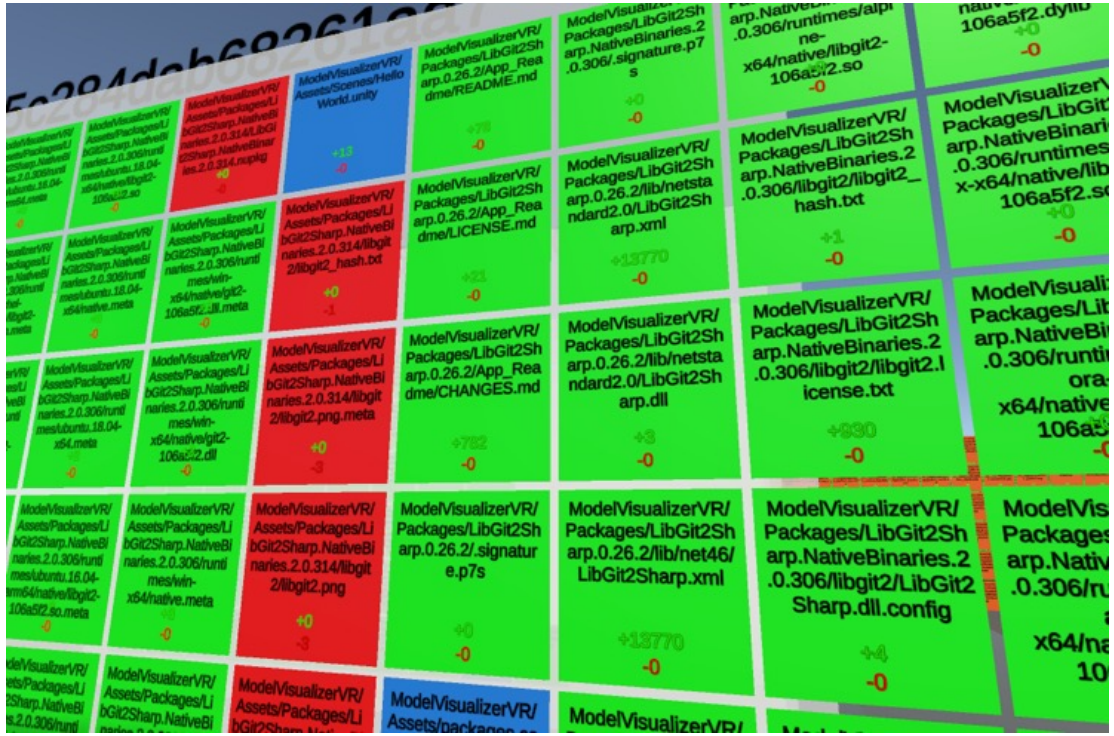


Branch merge



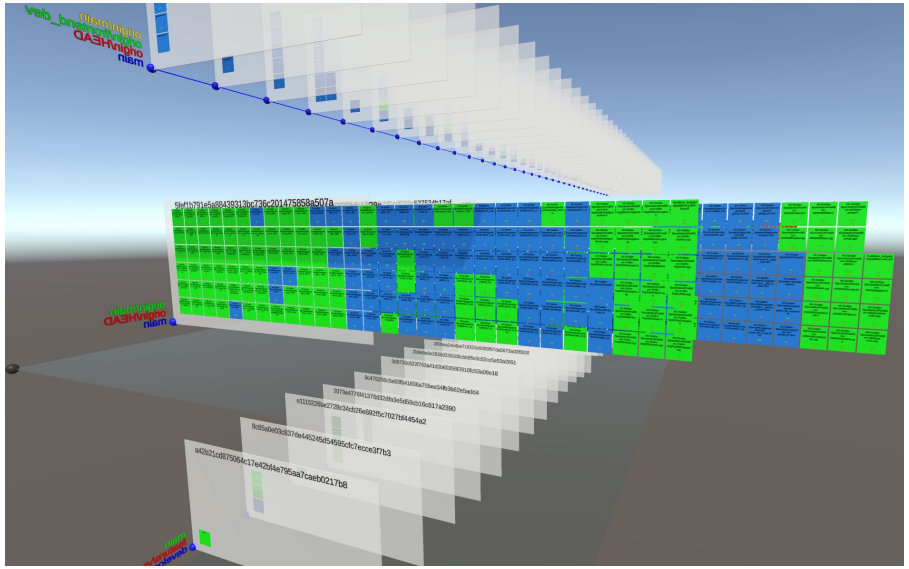
Evaluation: Commit Analysis Scenario

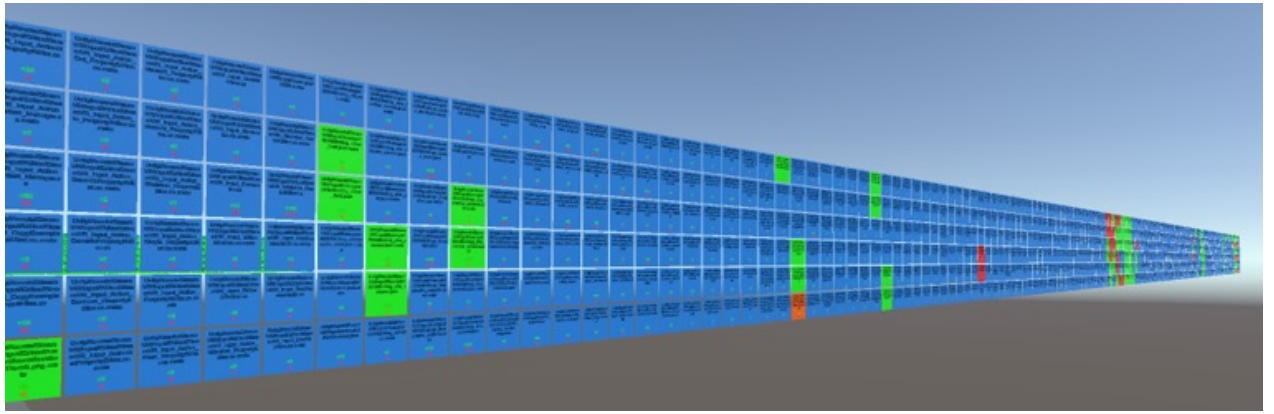
- Git commits are a snapshot of a repository. In a typical commit analysis, a stakeholder is interested in what changed with a commit, i.e., what files were added, deleted, or modified.
- To readily indicate this, tiles labeled with the file pathname are placed on the commit plane to represent changed files, with colors of green representing files added, blue changed, and red for deleted.
- In addition, the number of lines of text are shown at the bottom or a tile, with positive numbers in green indicating the number of lines added, and negative red values below it for the lines removed.



Evaluation: Commit Analysis Scenario

- In the top figure we see how commits that affect a large number of files can be readily determined. This can be helpful in analysis to quickly hone in on commit with the greatest impacts.
- The bottom figure shows VR's visual scaling for a very large commit file set.

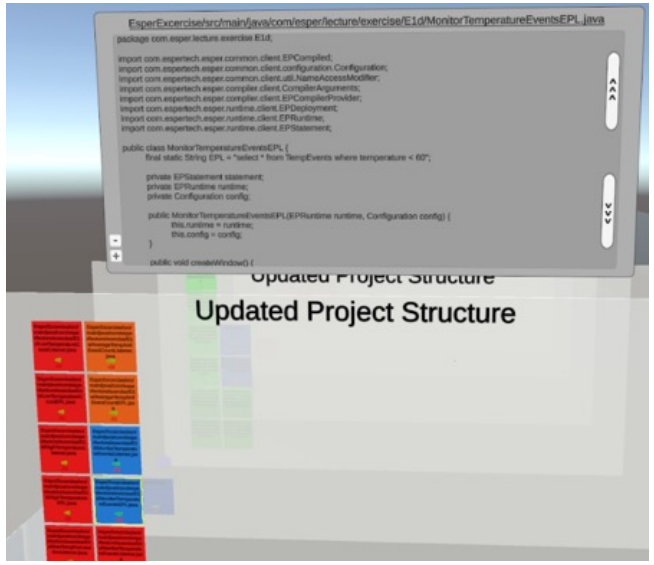


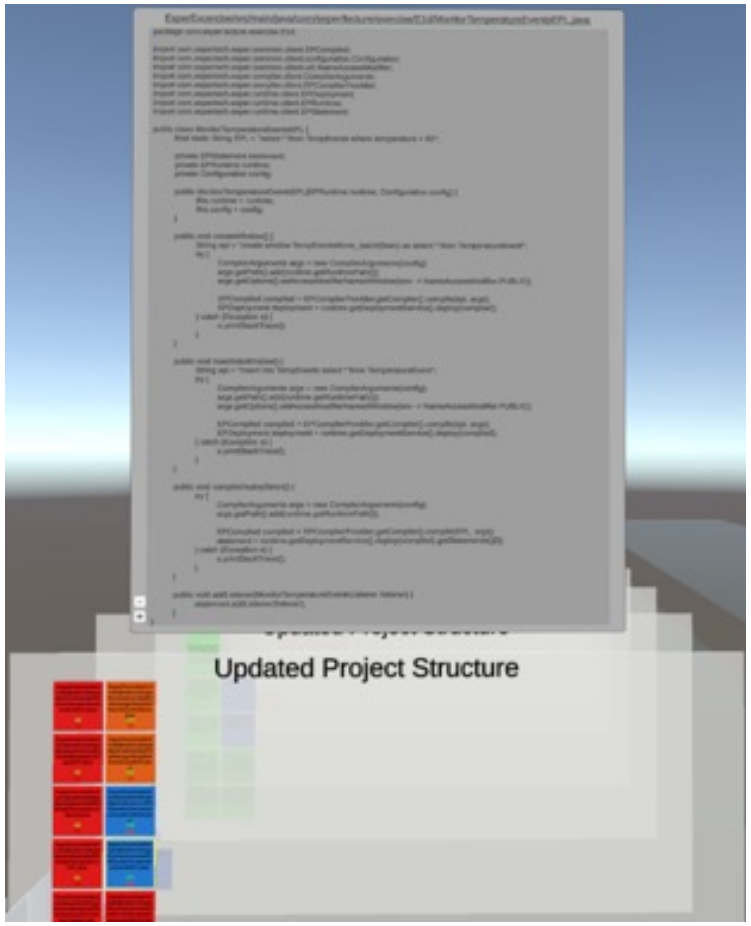


Evaluation: Commit Analysis Scenario

- Selecting specific tile (file) causes a file contents plane (i.e., code view) to pop up above displaying the contents of that file for that commit.
- Since file contents can be too lengthy and wide for practical depiction in our VR-Tablet, we chose to display the plane above the commit plane, providing a clear association.

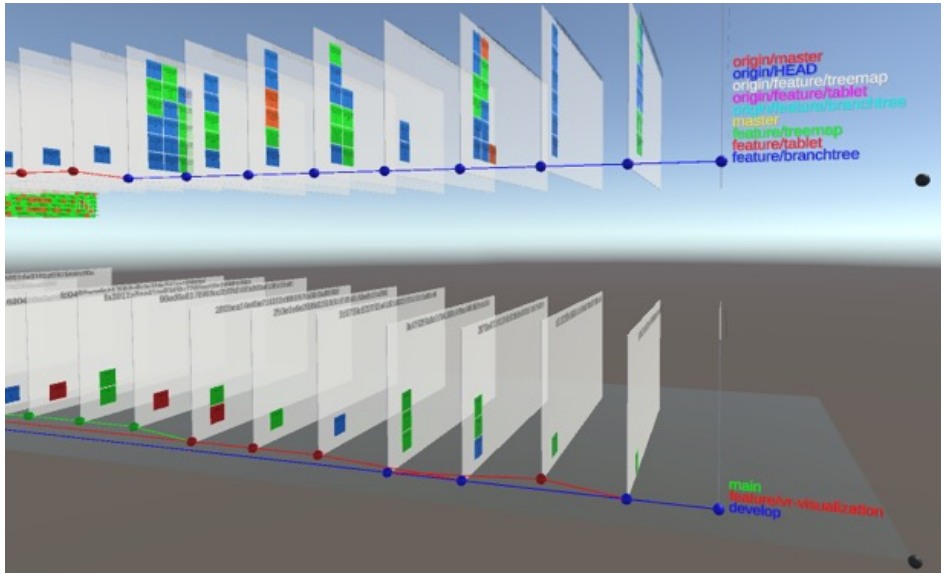
- The contents are initially scrollable, and can be expanded with the plus icon to show the entire file contents if desired.
- Since VR is not limited, one can navigate by moving the VR camera to any part of the code plane to see the code there.

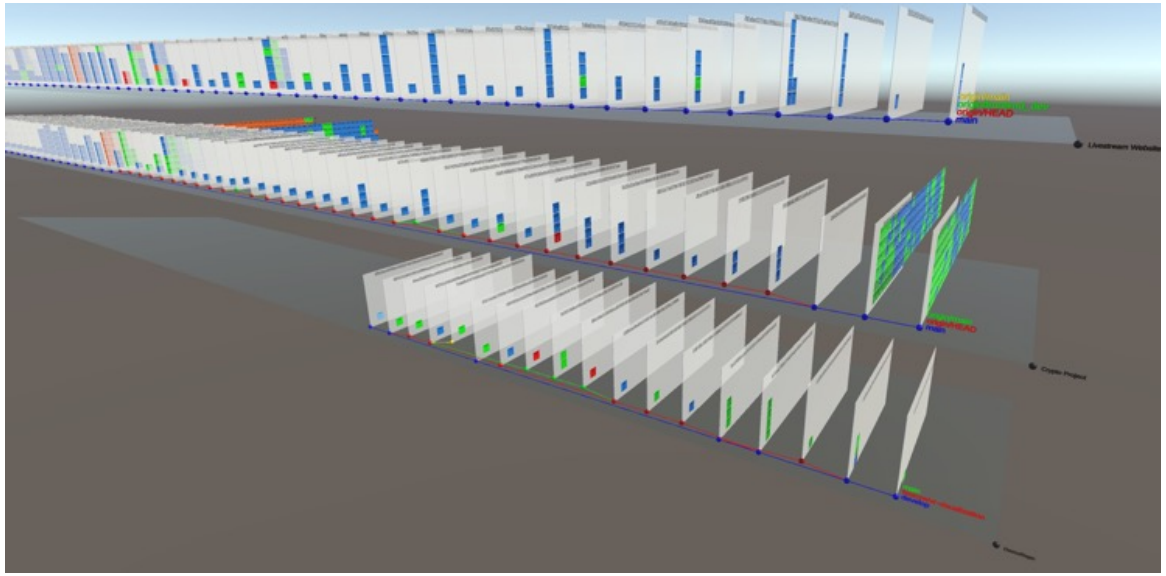




Evaluation: Multi-Repository Analysis Scenario

- To support multiple repository analysis, hyperplanes are used to represent each separate repository.
- Via the anchors, these can be placed where appropriate for the user.
- The left figure shows how branch and commit comparisons can be made from the branch perspective, with visual cues being offered by the tiles. Here one can see how the branches developed with their commits.
- The right figure shows how a larger visual depiction of three repositories and readily indicates which ones involved more commits, and via the extended commit planes where large commits were involved.





VR-Git Conclusion

- Contributes an immersive software repository experience for visually depicting and navigating repositories in VR.
- Provides a convenient way for stakeholder collaboration
  - Grass-roots, those who may not be developers yet have a legitimate interest in the code development to collaborate.
  - This can further onboarding of maintenance or quality assurance personnel.
- The unlimited space in VR facilitates the depiction and visual navigation of large repositories
  - Relations within and between artifacts, groups, and versions can be analyzed.
  - Additional related repositories or models can be visualized and analyzed simultaneously and benefit more complex collaboration and comprehension.
- The sensory immersion of VR can support task focus during comprehension and increase enjoyment, while limiting the visual distractions that typical 2D display surroundings incur.
- The solution concept was evaluated with our protype using a case study based on typical Git comprehension and analysis scenarios:
  - Branch analysis, commit analysis, and multi-repository analysis.
- The results indicate that VR-Git can support these analysis scenarios and thus provide an immersive collaborative environment to involve and include a larger stakeholder spectrum in understanding Git repository development

References

1. GitHub repositories [Online]. https://web.archive.org/web/20220509204719/https://github.com/search 2022.06.10
2. GitHub users [Online]. https://web.archive.org/web/20220529205506/https://github.com/search 2022.06.10
3. C. Metz, “Google Is 2 Billion Lines of Code—And It’s All in One Place,” 2015. [Online]. http://www.wired.com/2015/09/ google-2-billion-lines-codeand-one-place/
4. Evans Data Corporation. [Online]. https://evansdata.com/press/viewRelease.php?pressID=293 2022.06.10
5. R. Oberhauser, “VR-UML: The unified modeling language in virtual reality – an immersive modeling experience,” International Symposium on Business Modeling and Software Design, Springer, Cham, 2021, pp. 40-58.
6. R. Oberhauser, “VR-SysML: SysML Model Visualization and Immersion in Virtual Reality,” International Conference of Modern Systems Engineering Solutions (MODERN SYSTEMS 2022), IARIA, 2022. To be published.
7. R. Oberhauser and C. Pogolski, "VR-EA: Virtual Reality Visualization of Enterprise Architecture Models with ArchiMate and BPMN," In: Shishkov, B. (ed.) BMSD 2019. LNBIP, vol. 356, Springer, Cham, 2019, pp. 170–187.
8. R. Oberhauser, “VR-ProcessMine: Immersive Process Mining Visualization and Analysis in Virtual Reality,” The Fourteenth International Conference on Information, Process, and Knowledge Management (eKNOW 2022), IARIA, 2022. To be published.
9. R. Oberhauser, C. Pogolski, and A. Matic, "VR-BPMN: Visualizing BPMN models in Virtual Reality," In: Shishkov, B. (ed.) BMSD 2018. LNBIP, vol. 319, Springer, Cham, 2018, pp. 83–97. https://doi.org/10.1007/978-3-319-94214-8_6
10. R. Oberhauser, P. Sousa, and F. Michel, "VR-EAT: Visualization of Enterprise Architecture Tool Diagrams in Virtual Reality," In: Shishkov B. (eds) Business Modeling and Software Design. BMSD 2020. LNBIP, vol 391, Springer, Cham, 2020, pp. 221-239. https://doi.org/10.1007/978-3-030-52306-0_14
11. R. Oberhauser, M. Baehre, and P. Sousa, “VR-EA+TCK: Visualizing Enterprise Architecture, Content, and Knowledge in Virtual Reality,” In: Shishkov, B. (ed.) BMSD 2022, LNBIP, Springer, Cham, 2022. To be published.
12. H. Bjørklund, “Visualisation of Git in Virtual Reality,” Master's thesis, NTNU, 2017.
13. GitHub Skyline [Online]. https://skyline.github.com 2022.06.10
14. J. Feiner and K. Andrews, “Repovis: Visual overviews and full-text search in software repositories,” In: 2018 IEEE Working Conference on Software Visualization (VISSOFT), IEEE, 2018, pp. 1-11.
15. Y. Kim et al., “Githru: Visual analytics for understanding software development history through git metadata analysis,” IEEE Transactions on Visualization and Computer Graphics, 27(2), IEEE, 2020, pp.656-666.
16. S. Elsen, “VisGi: Visualizing git branches,” In 2013 First IEEE Working Conference on Software Visualization, IEEE, 2013, pp. 1-4.
17. A. Ciani, R. Minelli, A. Mocci, and M. Lanza, “UrbanIt: Visualizing repositories everywhere,” In 2015 IEEE International Conference on Software Maintenance and Evolution (ICSME), IEEE, 2015, pp. 324-326.
18. R. Müller, P. Kovacs, J. Schilbach, and D. Zeckzer, "How to master challenges in experimental evaluation of 2D versus 3D software visualizations," In: 2014 IEEE VIS International Workshop on 3Dvis (3Dvis), IEEE, 2014, pp. 33-36
19. Libgit2Sharp. [Online]. https://github.com/libgit2/libgit2sharp 2022.06.10
20. A.R. Hevner, S.T. March, J. Park, and S. Ram, “Design science in information systems research,” MIS Quarterly, 28(1), 2004, pp. 75-105