

Collecting, Analyzing, and Curating Virtual
Reality Driven Open Source Projects for
Research and Developer Testing Tool

CAI, Yongjie 1155158879

WEI, Youlin 1155157186

April 17, 2024

The Chinese University of Hong Kong

Supervisor: Michael R. Lyu

FYP Number: LYU2304

2023-2024 Term 2



香港中文大學
The Chinese University of Hong Kong



Contents

1	Division of Labor	6
2	Introduction	7
2.1	Background on Virtual Reality Dataset (CAI)	7
2.2	The emergence of the Metaverse concept (CAI)	10
2.3	Importance of Collecting, Analyzing, and Curating Virtual Reality Driven Open Source Projects (CAI)	11
2.4	Purpose of the study	13
2.4.1	Motivation (CAI)	13

3	Related Work	14
3.1	Overview of existing research and development in virtual reality software (CAI)	14
3.2	Discussion of key contributions and advancements in the field (CAI)	18
4	Methodology	20
4.1	Description of the data collection process from GitHub (WEI)	20
4.1.1	The advantages of Using Github for Data Collection (WEI)	22
4.1.2	Identification of Relevant Keywords (WEI)	23
4.1.3	Final scope of repositories (WEI)	24
4.2	Explanation of the criteria used for repository classification (WEI)	26
4.3	Discussion of any limitations or challenges encountered during the data collection and classification (WEI)	29
5	Results and Findings	31

5.1	The key statistics and insights obtained from the repository analysis (CAI)	31
5.2	Overview of database’s categories (CAI)	31
6	Further Expansion of the Dataset	33
6.1	Implementation of a scheduled web crawler for real-time data collection (CAI)	33
6.2	Development of a metadata scraping tool for enhanced dataset information (WEI)	35
6.3	Deploying the dataset to a server (WEI)	37
6.4	Improvements in data visualization and the intelligent features of the website (CAI)	38
7	Validation of the Dataset	45
7.1	Testing three cutting-edge VR open-source software tools: VRGuide, VRGreed, and VRMonkey (CAI)	45
7.2	Reproduction of experiments using our dataset (CAI)	49
7.3	Identification of issues in the tools and enhancement through the some rules (WEI)	54

7.4	How can we improve object coverage and method coverage (WEI)	61
8	Future Work	66
8.1	Exploration of potential directions for future research and development (CAI)	66
8.2	Potential use of databases for debugging in virtual reality projects (CAI)	69
8.3	Proposal of areas where improvements can be made in virtual reality software (WEI)	70
8.4	Consideration of the integration of AI technologies (LLM model) in enhancing VR experiences (CAI)	72
9	Conclusion	73
9.1	Recapitulation of the study’s objectives and key findings (WEI)	73
9.2	Summary of the contributions and implications of the re- search (WEI)	75

Acknowledgements

We would like to express our deepest gratitude to Professor Michael R. Lyu, our supervisor, and Ms. Shuqing Li, our advisor. Their invaluable guidance and profound insights have significantly contributed to the success of our final year project.

1 Division of Labor

In the entire FYP report, our division of work was based on our respective responsibilities. Throughout the entire process of our experiment, we worked step by step. In the initial stages, we collaborated on dataset selection. Subsequently, I primarily handled dataset analysis, while Youlin was responsible for scraping meta data. Later on, I took charge of designing and implementing the website, while Youlin deployed our dataset on the server. In the reproduction of experiments, I focused on replicating VRGuide, while Youlin tackled VRTest. Therefore, we completed our paper collaboratively. The responsibility for each chapter has been indicated in the section titles.

CAI Yong Jie was responsible for: the entire session 2, session 3, session 5, session 6.1, session 6.4, session 7.1, session 7.2, session 8.1, session 8.2, and session 8.4.

WEI You Lin was responsible for: the entire session 4, session 6.2, session 6.3, session 7.3, session 7.4, session 8.3, session 9.1, and session 9.2.

2 Introduction

2.1 Background on Virtual Reality Dataset (CAI)

Virtual reality (VR) datasets are collections of data specifically curated for training and evaluating virtual reality systems. These datasets typically contain various types of information, such as 3D models, textures, motion capture data, depth maps, and audio recordings, among others.

The current situation is that there is a relative scarcity of virtual reality (VR) datasets available in the market, which poses challenges for developers in testing and keeping up with VR development trends. This becomes particularly crucial with the unprecedented growth in the VR market following the release of Apple’s Vision Pro [1]. Having a good VR dataset is now more important than ever.

A high-quality VR dataset is essential for the development and testing of VR applications [2]. Datasets provide valuable resources for creating realistic virtual environments and can be used for training and testing various VR applications such as games, simulations, training, and virtual tourism.

However, there is indeed a lack of widely applicable and high-quality

VR datasets on the market. This may be because creating a complete and realistic VR dataset is a complex and time-consuming task. It requires gathering a large amount of information, including 3D models, textures, motion capture data, depth maps, and other types of data. These data must be carefully processed and integrated to build a realistic and immersive virtual environment.

With the rapid growth of the VR market and the increasing demand for datasets by developers, I believe there will be more efforts invested in creating and releasing VR datasets in the future. We may see more datasets specifically tailored for VR applications to meet the testing and research needs of developers.

At this stage, developers are experimenting with existing small-scale VR datasets or creating and collecting data suitable for their specific applications through their own efforts and creativity. Additionally, they can participate in academic research projects or collaborate with other developer communities to share and access more VR dataset resources. However, these acquisition methods are often limited and inconvenient.

To address the growing data gap, the idea of collecting open-source VR projects from GitHub and building a VR dataset is highly feasible

and meaningful. Through this initiative, we aim to provide developers with a convenient and accessible resource to facilitate innovation and development in VR applications.

Starting with collecting open-source VR projects is an excellent starting point. GitHub is a platform that hosts a vast number of open-source projects, and we can leverage its resources to build the VR dataset. By categorizing and analyzing these projects, we can provide developers with an organized and searchable dataset, enabling them to quickly find the resources they need.

We will also establish a website and server to store the data. This way, developers can browse and download the VR dataset by accessing the website. Ensuring that our website has a user-friendly interface and search functionality will allow users to easily find the projects and data they need. Additionally, ensuring server stability and data security will protect user privacy and data integrity.

During the process of dataset classification and analysis, we will consider common categorization criteria such as application domains (gaming, education, healthcare, etc.), technical requirements (hardware compatibility, development platforms, etc.), or project types (simulators, in-

teractive applications, etc.). This will help developers find projects that meet their specific needs more accurately.

Our goal is to have more dataset resources available in the future to drive innovation and development in virtual reality applications.

2.2 The emergence of the Metaverse concept (CAI)

In recent years, another buzzword that has gained significant attention is the Metaverse. The Metaverse [3] is a virtual world that encompasses all virtual realities, augmented reality, and the internet. It serves as a shared space where users can interact with each other and the virtual environment in real-time. The concept of the Metaverse has garnered interest due to its potential for socialization, entertainment, and business applications, offering a new way to connect and share experiences [4].

In this context, we envision that VR datasets can play a crucial role in accelerating people's entry into the Metaverse. By providing rich resources and tools, VR datasets can assist developers in rapidly constructing realistic and interactive Metaverse environments, thereby expediting people's immersion and engagement in the Metaverse. This, in turn, will offer users a broader and more diverse range of virtual experiences while

driving innovation and advancement within the Metaverse.

2.3 Importance of Collecting, Analyzing, and Curating Virtual Reality Driven Open Source Projects (CAI)

Collecting, analyzing, and curating open source projects driven by virtual reality is incredibly important for a variety of reasons. Firstly, it allows for the sharing of knowledge and collaboration within the VR community. By bringing together these projects, developers and enthusiasts can exchange valuable insights, ideas, and expertise, fostering a sense of collaboration and driving advancements in VR technology.

Secondly, the process of collecting and analyzing open source projects can significantly speed up the development process. By studying the trends, patterns, and successful implementations found within these projects, developers can identify best practices, common challenges, and emerging technologies. This knowledge enables them to streamline the creation of new VR applications, frameworks, and tools, saving time and effort in the development cycle.

Another crucial aspect is the role open source projects play in quality assurance and bug fixing. By collecting and curating these projects, it becomes easier to identify and address common issues, ensuring the stability and reliability of VR applications. Analyzing bug reports and resolutions within these projects provides valuable insights that can contribute to improving the overall VR development process.

Moreover, open source projects are hubs of innovation and novelty in the VR industry. By curating a diverse range of projects, developers gain access to a wealth of creative ideas and novel approaches. This inspires new possibilities and pushes the boundaries of VR technology, driving continuous innovation and advancement in the field.

Lastly, collecting and curating open source projects in VR provides an excellent opportunity for learning and education. Developers, students, and researchers can study different implementation techniques, gain hands-on experience, and deepen their understanding of the underlying principles of VR development. This fosters a vibrant learning environment and helps nurture the growth of the next generation of VR professionals.

2.4 Purpose of the study

2.4.1 Motivation (CAI)

Even though Virtual Reality (VR) technology has been rapidly evolving and growing, the industry has recently faced some major challenges. In 2021, Facebook, a leading tech company heavily invested in VR, rebranded itself as "Meta" and shifted its focus to developing the "metaverse." However, according to Meta's fiscal report for the second quarter of 2023 [5], their Reality Labs division, responsible for VR and Augmented Reality (AR) hardware and software, experienced a decline in revenue and income compared to the previous year [6]. And it's not just Meta; other companies in the industry have also encountered similar obstacles.

A report released by the International Data Corporation on October 6, 2023 [7] revealed a consecutive four-year decline in the global shipment of AR and VR headsets. One of the main reasons for this decline is the underperformance of VR software. A study analyzing complaints related to popular VR games found that while issues like cybersickness are less common, software bugs remain a persistent problem, ranking fourth

among all complaint categories. Even Meta’s flagship metaverse application, an online VR game, has received criticism from Meta employees who reported numerous software bugs affecting its usability [8].

Despite the clear evidence of the industry’s decline, there is a noticeable lack of available VR datasets for developers to use when implementing and testing their VR projects. This scarcity of data could worsen the current state of the industry.

To improve the user experience of VR technology, our aim is to compile a comprehensive dataset. This collection of information will be a valuable resource for developers looking to refine their VR projects and will support the development of tools to further advance the field.

3 Related Work

3.1 Overview of existing research and development in virtual reality software (CAI)

Virtual Reality (VR) has become a popular topic in recent research, as discovered through Google Scholar. Researchers are primarily exploring its application in data visualization and analysis. They are using

VR to process and display complex datasets, which leads to a better understanding and analysis of the data. In particular, studies are exploring how the interactivity and immersive nature of VR environments can enhance data visualization and provide more comprehensive ways of presenting data [9].

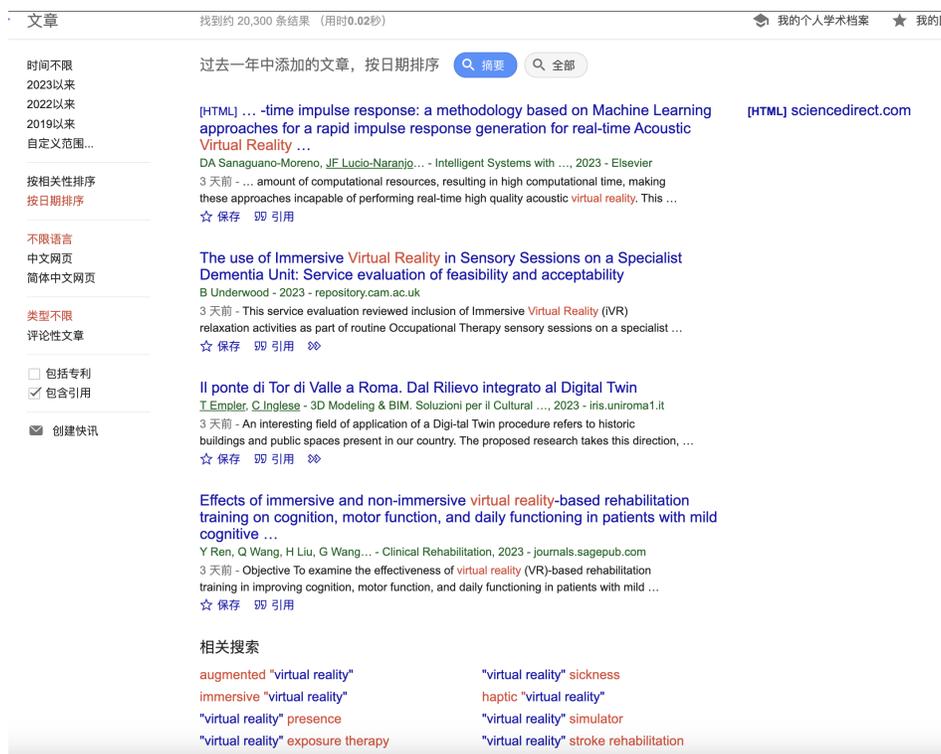


Figure 1: Current Research

Some researchers are examining the visualization and analysis of Earth science datasets using VR technology. They are transforming this data into three-dimensional models within virtual environments, which helps

them gain better insights into the Earth’s surface topography [10], climate patterns, and other geophysical phenomena. This immersive approach to data exploration helps scientists identify patterns and trends that may be hidden within the data, thus driving advancements in Earth science research.

<p>Visualization of large complex datasets using virtual reality S Sarathy, K Shujaee, K Cannon - ... International Conference on ..., 2000 - ieeexplore.ieee.org ... Presented here is a method which uses Virtual Reality (VR) coupled with multi-media ... the much larger datasets typical in the HPC applications. Virtual Reality has been around for ... ☆ 保存 引用 被引用次数: 18 相关文章 所有 3 个版本</p>	<p>[PDF] ieee.org</p>
<p>Slice WIM: a multi-surface, multi-touch interface for overview+ detail exploration of volume datasets in virtual reality D Coffey, N Malbraaten, T Le, I Borazjani... - ... on Interactive 3D ..., 2011 - dl.acm.org ... datasets that explores the potential of new interfaces made possible by a virtual reality (VR) ... Slice WIM displays a miniature version of the 3D dataset within a head-tracked stereoscopic ... ☆ 保存 引用 被引用次数: 61 相关文章 所有 6 个版本</p>	<p>[PDF] acm.org</p>
<p>[HTML] A dataset for emotion recognition using virtual reality and EEG (DER-VREEG): emotional state classification using low-cost wearable VR-EEG headsets NS Suhaimi, J Mountstephens, J Teo - Big Data and Cognitive Computing, 2022 - mdpi.com ... This study aims to use a virtual reality (VR) headset to ... VR database that is accessible to the public and that can potentially assist with emotion recognition studies using virtual reality ... ☆ 保存 引用 被引用次数: 27 相关文章 所有 3 个版本</p>	<p>[HTML] mdpi.com</p>
<p>Harnessing the power of immersive virtual reality-visualization and analysis of 3D earth science data sets J Zhao, JO Wallgrün, PC LaFemina... - Geo-spatial ..., 2019 - Taylor & Francis ... Historically, these data sets have been analyzed and quarried on 2D ... virtual reality (iVR) allow for the integration, visualization, analysis, and exploration of these 3D geospatial data sets... ☆ 保存 引用 被引用次数: 30 相关文章 所有 8 个版本</p>	<p>[PDF] tandfonline.com</p>
<p>Supporting iterative virtual reality analytics design and evaluation by systematic generation of surrogate clustered datasets SK Tadeja, P Langdon... - ... and Augmented Reality ..., 2021 - ieeexplore.ieee.org ... for synthesizing surrogate clustered datasets for use in virtual reality analytics design and evaluation... how new insights in VR analytics can be gained using surrogate clustered datasets. ... ☆ 保存 引用 被引用次数: 2 相关文章 所有 7 个版本</p>	<p>[PDF] ieee.org</p>
<p>Eye movement biometrics using a new dataset collected in virtual reality DJ Lohr, S Aziz, O Komogortsev - ACM Symposium on Eye Tracking ..., 2020 - dl.acm.org ... dataset collected in virtual reality (VR) that contains both 2D and 3D eye movement data from over 400 subjects. We establish that this dataset ... an existing, similarly constructed dataset. ...</p>	<p>[PDF] acm.org</p>

Figure 2: Current Research 2

Other studies are focusing on using VR technology for emotion recognition and eye-tracking research. Researchers are combining VR environ-

ments with biometric measurement techniques to observe participants' emotional states and eye movements within virtual scenarios. [11] These studies contribute to a deeper understanding of human emotions and cognitive processes and offer novel research methods and tools for applications such as emotion recognition and psychological health assessment.

Furthermore, VR is being applied in various fields, including education [12], training, and visual analytics. By creating virtual environments and interactive virtual objects, researchers and professionals can conduct simulated experiments, training sessions, and visual analysis of complex datasets. This immersive and interactive approach to learning and analysis provides more specific and practical experiences, facilitating better knowledge and skill development.

Based on the Google Scholar search, it appears that existing research lacks content specifically related to virtual reality datasets. Therefore, our research would be highly meaningful in filling this gap. By exploring the creation, processing, and analysis of virtual reality datasets, you can make significant contributions to the application of VR in data visualization and analysis. This will provide richer resources and technologies for other researchers and professionals, thereby driving further advance-

ments in the field of virtual reality.

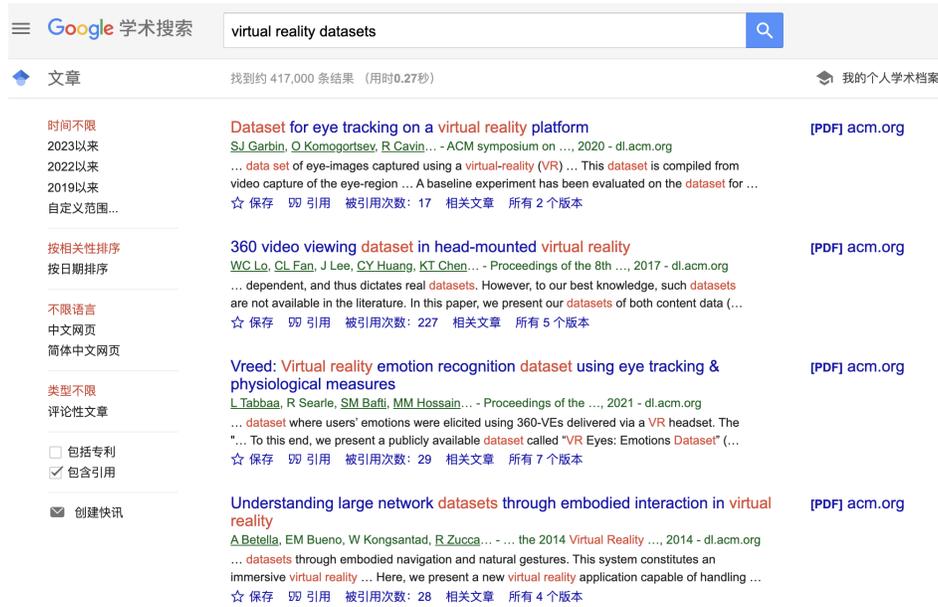


Figure 3: Current Research 2

3.2 Discussion of key contributions and advancements in the field (CAI)

Firstly, our VR dataset provides an intuitive and immersive data experience. It allows developers and researchers to interact with data in a more intuitive and immersive way, helping them better explain and analyze complex information.

Secondly, the VR dataset drives scientific research and experimentation. Our dataset can be used to simulate experimental conditions,

observe, test, and analyze phenomena, and explore new hypotheses and theories. By visualizing and interacting with data in a virtual environment, researchers can delve deeper into the relationships and patterns within the data, thereby advancing scientific research.

Similarly, our VR dataset holds great potential value in the field of education and training. It can be used to create virtual laboratories, simulate training scenarios, and interactive learning environments. By utilizing the VR dataset, students and trainers can engage in experiential learning, enhancing learning outcomes and skill development.

Lastly, the VR dataset supports decision-making and problem-solving. By simulating and visualizing data within a virtual environment, decision-makers and problem solvers can better grasp the complexity of issues and explore different solutions and possibilities. This contributes to improved accuracy and effectiveness in decision-making, as well as driving innovation and improvement. Our VR dataset has several valuable applications. Firstly, it provides an immersive and intuitive experience, helping developers and researchers to better analyze and explain complex information.

4 Methodology

4.1 Description of the data collection process from GitHub (WEI)

In our effort to collect relevant data on Virtual Reality (VR) projects, we chose to focus on open-source projects accessible via GitHub, a leading platform for such resources. This decision was influenced by the methodology described by Geiger et al. [13], where they used a detailed keyword strategy to identify pertinent publications by their titles. This method facilitated the organization of Android application collections, a platform widely used globally. Despite the vast availability of data on Android, resources specific to VR development are relatively scarce [14], notably lacking in robust developer communities for data sourcing. Therefore, GitHub, being one of the largest open-source code repositories, was selected as our primary data source.

One of the primary benefits of using GitHub is its open-source nature, providing free access to all public repositories. As a worldwide platform, it hosts a diverse array of projects, enriching the breadth and depth of data available to us, thus expanding our research capabilities.

Additionally, GitHub offers an Application Programming Interface (API) that enhances our data gathering strategy. The API allows user authentication via an installation access token and supports high-frequency data requests, handling up to 5,000 requests per hour. This capability enables us to gather substantial data volumes efficiently over a short period.

The data sourced from GitHub is typically up-to-date and accurate, with developers continuously updating their projects with new code, bug fixes, and features. This ensures the data's relevance and timeliness, keeping our project aligned with the latest VR trends and developments.

GitHub also provides supplementary project information, such as the number of stars, forks, and watchers, which indicates a project's popularity and influence within the GitHub community. Access to each project's commit history offers a chronological view of its development, providing a valuable resource for further analysis and inspiration in subsequent phases of our project. These metadata elements enhance our analysis, offering additional context and insights beyond just the codebase.

4.1.1 The advantages of Using Github for Data Collection (WEI)

Our project aims to classify and examine repositories within the Virtual Reality (VR) field. We are focused on identifying as many VR-related repositories as possible. To ensure a comprehensive scope, our study also covers related technologies like Mixed Reality (MR) and Extended Reality (XR), although we have decided to exclude Augmented Reality (AR) from our analysis. This exclusion is based on the operational differences between AR and VR; AR typically does not require specialized VR hardware and often uses common devices like smartphones and tablets to superimpose digital content onto the real world. Including AR could bring in factors unrelated to VR hardware usage, potentially skewing our analysis.

To capture a broad spectrum of relevant data without missing out due to terminology variations, we've included abbreviations such as "VR", "MR", "XR", and even "AR" in our keyword search. This approach helps ensure a thorough and precise representation of the current landscape of repositories related to Virtual Reality technologies.

4.1.2 Identification of Relevant Keywords (WEI)

Our project aims to classify and examine repositories within the Virtual Reality (VR) field. We are focused on identifying as many VR-related repositories as possible. To ensure a comprehensive scope, our study also covers related technologies like Mixed Reality (MR) and Extended Reality (XR), although we have decided to exclude Augmented Reality (AR) from our analysis. This exclusion is based on the operational differences between AR and VR; AR typically does not require specialized VR hardware and often uses common devices like smartphones and tablets to superimpose digital content onto the real world. Including AR could bring in factors unrelated to VR hardware usage, potentially skewing our analysis.

To capture a broad spectrum of relevant data without missing out due to terminology variations, we've included abbreviations such as "VR", "MR", "XR", and even "AR" in our keyword search. This approach helps ensure a thorough and precise representation of the current landscape of repositories related to Virtual Reality technologies.

4.1.3 Final scope of repositories (WEI)

Virtual Reality (VR) has become a significant area of interest, leading to an increase in VR-related repositories on GitHub. To ensure the integrity of our data, it is crucial to employ a filtration system to remove repositories that lack substantial or useful content.

Using the GitHub API, we crafted queries incorporating specific VR-related keywords to fetch repositories. The API returns a JSON file with detailed attributes of each repository, including the repository title, author, URL, number of forks, star count, and the date of the last update.

Here's an example of how we structured our API query:

```
base url = "https://api.github.com/search/
repositories?q={}+in:name,description,readme,topics+stars:{}+pushed:{}.
format(keyword, n, since, until)
```

To keep our dataset current and relevant, we excluded repositories that hadn't been updated since before January 1, 2020, considering the rapid advancements in VR technology. We also set a minimum star count of 10 to ensure the repositories included were vetted and recognized by the community for their reliability and popularity.

```

"html_url": "https://github.com/rokups/virtual-reality",
"description": "Stealthy backdoor for Windows operating systems",
"fork": false,
"url": "https://api.github.com/repos/rokups/virtual-reality",
"forks_url": "https://api.github.com/repos/rokups/virtual-reality/forks",
"keys_url": "https://api.github.com/repos/rokups/virtual-reality/keys/{key_id}",
"collaborators_url": "https://api.github.com/repos/rokups/virtual-reality/collaborators/{collaborator}",
"teams_url": "https://api.github.com/repos/rokups/virtual-reality/teams",
"hooks_url": "https://api.github.com/repos/rokups/virtual-reality/hooks",
"issue_events_url": "https://api.github.com/repos/rokups/virtual-reality/issues/events/{number}",
"events_url": "https://api.github.com/repos/rokups/virtual-reality/events",
"assignees_url": "https://api.github.com/repos/rokups/virtual-reality/assignees/{user}",
"branches_url": "https://api.github.com/repos/rokups/virtual-reality/branches/{branch}",
"tags_url": "https://api.github.com/repos/rokups/virtual-reality/tags",
"blobs_url": "https://api.github.com/repos/rokups/virtual-reality/git/blobs/{sha}",
"git_tags_url": "https://api.github.com/repos/rokups/virtual-reality/git/tags/{sha}",
"git_refs_url": "https://api.github.com/repos/rokups/virtual-reality/git/refs/{sha}",
"trees_url": "https://api.github.com/repos/rokups/virtual-reality/git/trees/{sha}",
"statuses_url": "https://api.github.com/repos/rokups/virtual-reality/statuses/{sha}",
"languages_url": "https://api.github.com/repos/rokups/virtual-reality/languages",
"stargazers_url": "https://api.github.com/repos/rokups/virtual-reality/stargazers",
"contributors_url": "https://api.github.com/repos/rokups/virtual-reality/contributors",
"subscribers_url": "https://api.github.com/repos/rokups/virtual-reality/subscribers",
"subscription_url": "https://api.github.com/repos/rokups/virtual-reality/subscription",
"commits_url": "https://api.github.com/repos/rokups/virtual-reality/commits/{sha}",
"git_commits_url": "https://api.github.com/repos/rokups/virtual-reality/git/commits/{sha}",
"comments_url": "https://api.github.com/repos/rokups/virtual-reality/comments/{number}",
"issue_comment_url": "https://api.github.com/repos/rokups/virtual-reality/issues/comments/{number}",
"contents_url": "https://api.github.com/repos/rokups/virtual-reality/contents/{+path}",
"compare_url": "https://api.github.com/repos/rokups/virtual-reality/compare/{base}...{head}",
"merges_url": "https://api.github.com/repos/rokups/virtual-reality/merges",
"archive_url": "https://api.github.com/repos/rokups/virtual-reality/{archive_format}/{ref}",
"downloads_url": "https://api.github.com/repos/rokups/virtual-reality/downloads",
"issues_url": "https://api.github.com/repos/rokups/virtual-reality/issues/{number}",
"pulls_url": "https://api.github.com/repos/rokups/virtual-reality/pulls/{number}",
"milestones_url": "https://api.github.com/repos/rokups/virtual-reality/milestones/{number}",
"notifications_url": "https://api.github.com/repos/rokups/virtual-reality/notifications?since=all,participating",
"labels_url": "https://api.github.com/repos/rokups/virtual-reality/labels/{name}",
"releases_url": "https://api.github.com/repos/rokups/virtual-reality/releases/{id}",
"deployments_url": "https://api.github.com/repos/rokups/virtual-reality/deployments",
"created_at": "2019-02-06T11:29:22Z",
"updated_at": "2023-11-24T09:12:01Z",
"pushed_at": "2020-02-13T14:11:36Z",
"git_url": "git://github.com/rokups/virtual-reality.git",
"ssh_url": "git@github.com:rokups/virtual-reality.git",
"clone_url": "https://github.com/rokups/virtual-reality.git",
"svn_url": "https://github.com/rokups/virtual-reality",
"homepage": "",
"size": 503,
"stargazers_count": 272,
"watchers_count": 272,
"language": "C",
"has_issues": true,
..

```

Figure 4: The JSON File Returned by GitHub API that containing attributes

While the star count was a primary filter, we also noted the fork count, which helps gauge a repository's impact and acceptance within the

developer community. This data will be referenced in further analysis.

```
"html_url": "https://github.com/rokups/virtual-reality",
"description": "Stealthy backdoor for Windows operating systems",
"fork": false,
"url": "https://api.github.com/repos/rokups/virtual-reality",
"forks_url": "https://api.github.com/repos/rokups/virtual-reality/forks",
"keys_url": "https://api.github.com/repos/rokups/virtual-reality/keys{/key_id}",
"collaborators_url": "https://api.github.com/repos/rokups/virtual-reality/collaborators{/collaborator}",
"teams_url": "https://api.github.com/repos/rokups/virtual-reality/teams",
"hooks_url": "https://api.github.com/repos/rokups/virtual-reality/hooks",
"issue_events_url": "https://api.github.com/repos/rokups/virtual-reality/issues/events{/number}",
"events_url": "https://api.github.com/repos/rokups/virtual-reality/events",
"assignees_url": "https://api.github.com/repos/rokups/virtual-reality/assignees{/user}",
"branches_url": "https://api.github.com/repos/rokups/virtual-reality/branches{/branch}",
"tags_url": "https://api.github.com/repos/rokups/virtual-reality/tags",
"blobs_url": "https://api.github.com/repos/rokups/virtual-reality/git/blobs{/sha}",
"git_tags_url": "https://api.github.com/repos/rokups/virtual-reality/git/tags{/sha}",
"git_refs_url": "https://api.github.com/repos/rokups/virtual-reality/git/refs{/sha}",
"trees_url": "https://api.github.com/repos/rokups/virtual-reality/git/trees{/sha}",
"statuses_url": "https://api.github.com/repos/rokups/virtual-reality/statuses{/sha}",
"languages_url": "https://api.github.com/repos/rokups/virtual-reality/languages",
"stargazers_url": "https://api.github.com/repos/rokups/virtual-reality/stargazers",
"contributors_url": "https://api.github.com/repos/rokups/virtual-reality/contributors",
"subscribers_url": "https://api.github.com/repos/rokups/virtual-reality/subscribers",
"subscription_url": "https://api.github.com/repos/rokups/virtual-reality/subscription",
"commits_url": "https://api.github.com/repos/rokups/virtual-reality/commits{/sha}",
"git_commits_url": "https://api.github.com/repos/rokups/virtual-reality/git/commits{/sha}",
"comments_url": "https://api.github.com/repos/rokups/virtual-reality/comments{/number}",
"issue_comment_url": "https://api.github.com/repos/rokups/virtual-reality/issues/comments{/number}",
"contents_url": "https://api.github.com/repos/rokups/virtual-reality/contents/{+path}",
"compare_url": "https://api.github.com/repos/rokups/virtual-reality/compare/{base}...{head}",
"merges_url": "https://api.github.com/repos/rokups/virtual-reality/merges",
"archive_url": "https://api.github.com/repos/rokups/virtual-reality/{archive_format}{/ref}",
"downloads_url": "https://api.github.com/repos/rokups/virtual-reality/downloads",
"issues_url": "https://api.github.com/repos/rokups/virtual-reality/issues{/number}",
"pulls_url": "https://api.github.com/repos/rokups/virtual-reality/pulls{/number}",
"milestones_url": "https://api.github.com/repos/rokups/virtual-reality/milestones{/number}",
"notifications_url": "https://api.github.com/repos/rokups/virtual-reality/notifications?{since,all,participating}",
"labels_url": "https://api.github.com/repos/rokups/virtual-reality/labels{/name}",
"releases_url": "https://api.github.com/repos/rokups/virtual-reality/releases{/id}",
"deployments_url": "https://api.github.com/repos/rokups/virtual-reality/deployments",
"created_at": "2019-02-06T11:29:22Z",
"updated_at": "2023-11-24T09:12:01Z",
"pushed_at": "2020-02-13T14:11:36Z",
"git_url": "git://github.com/rokups/virtual-reality.git",
"ssh_url": "git@github.com:rokups/virtual-reality.git",
"clone_url": "https://github.com/rokups/virtual-reality.git",
"svn_url": "https://github.com/rokups/virtual-reality",
"homepage": "",
"size": 503,
"stargazers_count": 272,
"watchers_count": 272,
"language": "C",
"has_issues": true,
```

Figure 5: Attributes of Every Repository

The selection criteria and attributes detailed above were pivotal in shaping the methodology of our research into VR technologies on GitHub.

4.2 Explanation of the criteria used for repository classification (WEI)

We successfully curated over 9,000 repositories by applying our predefined criteria. Our initial data-gathering used a tailored crawler script

that stored repository names and URLs in an Excel file, aiming to minimize data omissions. Despite this, abbreviations like "VR" (which could mean "Virtual Reality" or "Vehicle Routing") and "XR" (potentially part of Unix Commands) introduced ambiguity. This necessitated manual review of each repository to ensure relevance and accuracy.

During our review, we also undertook a comprehensive attribute marking process. Repositories were categorized based on their content from a software engineering perspective. For example, stand-alone programs operable in a VR environment were classified as "Application," while those with instructions for using such applications were labeled "Tutorial."

Our labeling approach was initially exploratory and became more refined over time. Adding labels is crucial as it helps developers quickly find projects that meet their needs, improving resource discovery efficiency.

Part of our analysis included checking each repository for the presence of code or asset files, crucial for assessing their completeness. Special attention was given to repositories marked as archived or deprecated, due to the dynamic nature of dependencies in VR projects, especially those using WebXR technology. These projects often become non-functional without regular updates due to changes in hardware specs, browser com-

patibility, and API versions, as pointed out by Zubair and Anyameluhor (2021).

Furthermore, we conducted a detailed review of repositories classified as "Application," categorizing them based on scope and complexity into demo, medium, or large-scale projects. This helps clarify the diversity of VR projects.

To enhance the accuracy of our findings, we implemented a cross-verification method, double-checking each repository's content. This step significantly boosted our confidence in the data's reliability.

Ultimately, we excluded all repositories identified as "irrelevant to VR" from our dataset. This thorough data cleansing and categorization process greatly enhanced the accuracy and reliability of our dataset. Moreover, it offered deep insights into the structure and characteristics of VR-related repositories on GitHub. This meticulous approach has ensured that our dataset truly reflects the current landscape of VR technology development within the GitHub community.

4.3 Discussion of any limitations or challenges encountered during the data collection and classification (WEI)

During the initial labeling phase, we encountered an unexpected challenge. Some repositories, although containing our targeted keywords in their README files, were predominantly focused on Augmented Reality (AR) rather than Virtual Reality (VR). This misalignment required us to refine our dataset to exclude these irrelevant terms, ensuring a more precise focus on VR-related content.

To address this, we implemented a strategy to identify repositories with mentions of "Augmented Reality" or "AR" in their README files. Once these were flagged, we conducted a thorough verification to determine their actual relevance to VR.

For this task, we developed a specialized script distinct from our initial data collection tools. This script was specifically crafted to filter and analyze repositories already in our dataset, eliminating the need for further API calls to GitHub, thus streamlining the process.

We opted to use Selenium, a robust web driver known for its browser

automation capabilities, to optimize this process. Selenium enabled us to automate the scanning of README files, greatly reducing the manual labor and time involved. This refinement step was crucial in our data processing pipeline, ensuring that our dataset remained focused and relevant to our research objectives by effectively filtering out content not directly related to VR.



Figure 6: Selenium: A tool that enables browser automation manual effort and time required.

5 Results and Findings

5.1 The key statistics and insights obtained from the repository analysis (CAI)

We went through four rounds of screening and classification. From a pool of 4,423 Github projects, we carefully handpicked 1,447 projects that are relevant to Virtual Reality (VR). These projects were then categorized into 122 file types based on our standardized classification criteria. Some examples of these file types include Library, Framework, SDK, and Engine. To provide a more visual representation of our classification, we created a tree diagram for each category. We also included the code we used to generate the trees and the tools we employed.

To present our dataset classification results in a more visually appealing manner, we have prepared a Term 1 paper.

5.2 Overview of database's categories (CAI)

Our analysis of the VR repository projects reveals that there are a total of 1,428 projects categorized into 5 different categories. The majority of projects (343) fall under the "Application" category, indicating a focus

on practical VR applications such as games and interactive experiences. Conversely, the "Library" and "Asset" categories have fewer projects (110 and 104 respectively), suggesting a preference for utilizing existing resources rather than developing new ones. The "Plugin" and "Tutorial" categories have 93 and 72 projects respectively, indicating a higher interest in creating plugins and providing tutorial resources. Overall, these statistics demonstrate the emphasis on practical applications, resource sharing, and expanding functionalities and knowledge in the VR domain.

In addition, we have classified the repositories based on their size and identified three levels: Demo, Medium, and Large. These levels cater to developers with varying skill levels and provide different levels of assistance and guidance. Furthermore, we have identified 80 projects as "Archived," indicating that they are no longer actively maintained. However, these archived projects still hold value for learning, reference, and historical purposes.

It is worth noting that archived projects may have fewer updates and support, but they can still offer valuable solutions and implementation approaches.

6 Further Expansion of the Dataset

6.1 Implementation of a scheduled web crawler for real-time data collection (CAI)

We have implemented a solution to ensure that our dataset is always up-to-date with the latest VR open-source projects. To achieve this, we modified the "main.py" web crawler code by adding a new function called "diff1.csv" and defining a file named "tmp1.csv." The script searches GitHub using keywords such as "VR" and "XR," filters the results based on star ratings, and saves the new dataset as "lesstar.csv."

The script also creates a copy of "lesstar.csv" called "tmp1.csv." The next time the script runs, it compares the new "lesstar.csv" with "tmp1.csv" to identify newly added projects within the last 15 days, which are stored in "diff.csv." To automate this process, we use the crontab feature of a Linux server, which allows us to schedule the periodic execution of the "main.py" script.

The crontab command [15] lets us specify fixed intervals, such as minutes, hours, days, months, weeks, or any combination thereof, for executing system commands or shell scripts. This command is partic-

```

        repo_names.append(repo_name)
        since = until
    if (15 <= (date.today() - until).days):
        until = until + timedelta(days=15)
    else:
        until = date.today()

repo_dict_unfiltered = {
    "Author Name": author_names_unfiltered,
    "Repo Name": repo_names_unfiltered,
    "URL": html_urls_unfiltered,
    "Created at": created_ats_unfiltered,
    "Pushed at": pushed_ats_unfiltered,
    "Updated at": updated_ats_unfiltered,
    "Star": stars_counts_unfiltered,
    "Fork": forks_counts_unfiltered
}
df = pd.DataFrame(repo_dict_unfiltered)
df.to_csv("less_stars.csv", mode = 'w')
compare_csv('less_stars.csv', 'tmp2.csv', 'diff2.csv')
df.to_csv("tmp2.csv", mode = 'w')

repo_dict = {
    "Author Name": author_names,
    "Repo Name": repo_names,
    "URL": html_urls,
    "Created at": created_ats,
    "Pushed at": pushed_ats,
    "Updated at": updated_ats,
    "Star": stars_counts,
    "Fork": forks_counts
}
df = pd.DataFrame(repo_dict)
df.to_csv("less_star.csv", mode = 'w')
compare_csv('less_star.csv', 'tmp1.csv', 'diff1.csv')
df.to_csv("tmp1.csv", mode = 'w')

```

Figure 7: Code of Crawler

ularly useful for tasks like periodic log analysis or data backup. Once the entire deployment is completed, we can obtain all recently updated GitHub projects with star ratings exceeding ten every fifteen days.

Our objective is to create a reusable dataset that offers unique advantages over conventional datasets, and this approach helps us achieve that by keeping our dataset up-to-date with the latest VR open-source projects.

6.2 Development of a metadata scraping tool for enhanced dataset information (WEI)

We have established a database that is updated regularly to handle the large volumes of data we are working with. Initially, the data was presented in Excel format, which was not very intuitive or user-friendly. To improve accessibility and usability for VR developers, it is essential that we provide a comprehensive introduction for each GitHub repository. These introductions are designed to help developers easily navigate and locate the specific projects they need for their development work. Fortunately, the GitHub REST API offers features that greatly facilitate this process.

By using the command `curl https://api.github.com/repos/owner/repo`, the REST API returns a JSON file containing detailed metadata about the repository. This metadata includes essential information such as the repository's URL, its name, the owner's information, a detailed description of the repository, and whether the repository includes a wiki. This information is crucial as it allows VR application developers to quickly get an overview and evaluate whether the repository meets their needs. Additionally, to further aid developers in understanding the repository's

```
if response.status_code == 200:
    repo_info = json.loads(response.text)
    with open(os.path.join(folder_path, 'metadata.json'), 'w') as f:
        json.dump(repo_info, f, indent=4)
    print(f"Metadata for {name} saved in {folder_path} folder.")
```

Figure 8: The crawler codes for metadata

structure and contents at a glance, I utilized the API's 'contents' functionality. This feature allows us to download a list of all files within each repository, providing a clear view of the repository's structure without needing to download or clone the repository locally.

Lastly, to ensure that developers have all the necessary information to get started with the repository, we also downloaded the README file from each repository. The README file typically includes detailed

instructions on how to set up and use the repository, any dependencies that might be required, and often a basic example or tutorial. This is an invaluable resource for developers who are evaluating or beginning to work with a new repository.

Additionally, these scripts have been integrated into a crawler that is set up to automatically update at regular intervals. This ensures that both the metadata and the repository information are kept synchronized with the latest changes.

Through these measures, we aim to make our vast database not only more accessible but also more intuitive for VR developers, thereby enhancing their productivity and easing the integration process with our data resources.

6.3 Deploying the dataset to a server (WEI)

After writing the scripts, the challenge we faced was how to store the massive amount of data. Ultimately, we decided to organize the metadata JSON, README, and file list information for each GitHub repository into a directory named after the full name of the repository. These directories are then placed under a root directory, which is collectively

uploaded to a remote server.

6.4 Improvements in data visualization and the intelligent features of the website (CAI)

We have made significant updates to our website [16], adding more features and making it dynamically update in sync with our server. The website's content is updated based on the information provided on May 1st. All new projects are now available on the website.

There is a new search bar at the top of the website where users can enter keywords to search for their desired VR projects. Additionally, users can use the "Share" button to share project links on social media platforms such as WeChat, Twitter, and Instagram.

The introduction module on the left side of the website provides information about our dataset collection's goals and vision. In the middle, there is a project display module that shows various open-source VR projects from GitHub [17]. The dots in different colors represent different project categories. For instance, the green color scheme represents the "Hardware" category, which includes "Hardware introduction," "Hardware architecture," and "Hardware description," with different shades

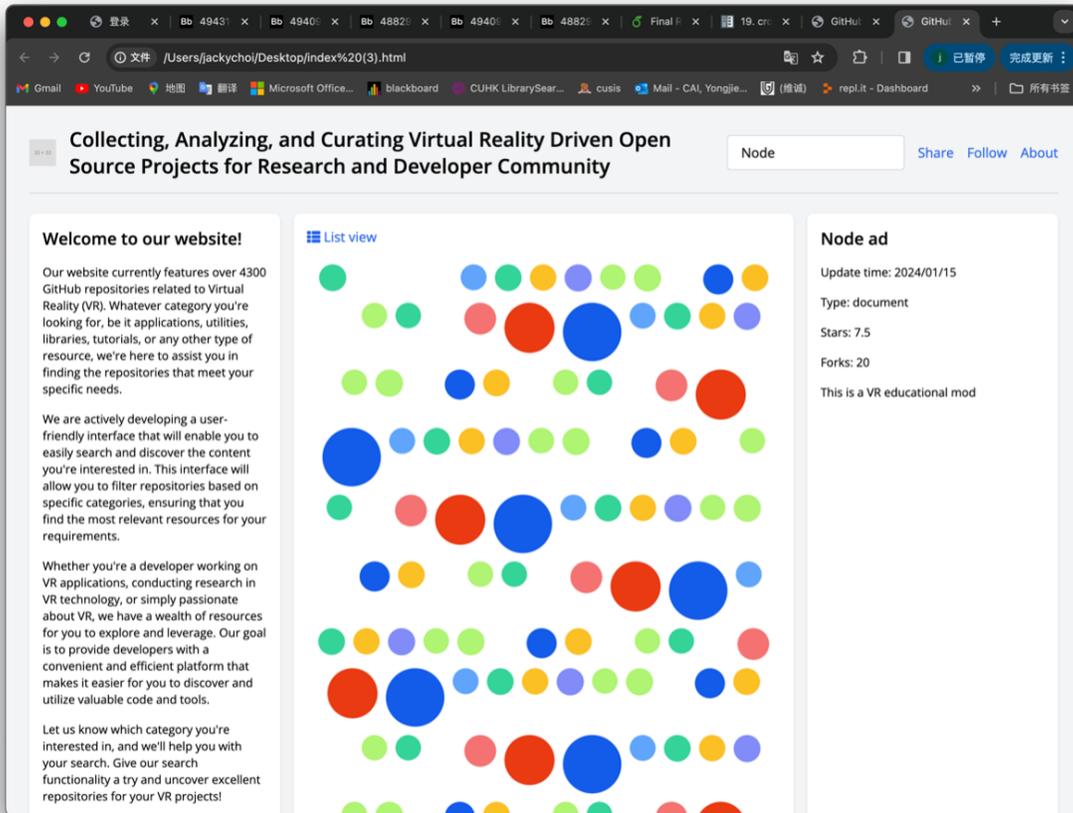
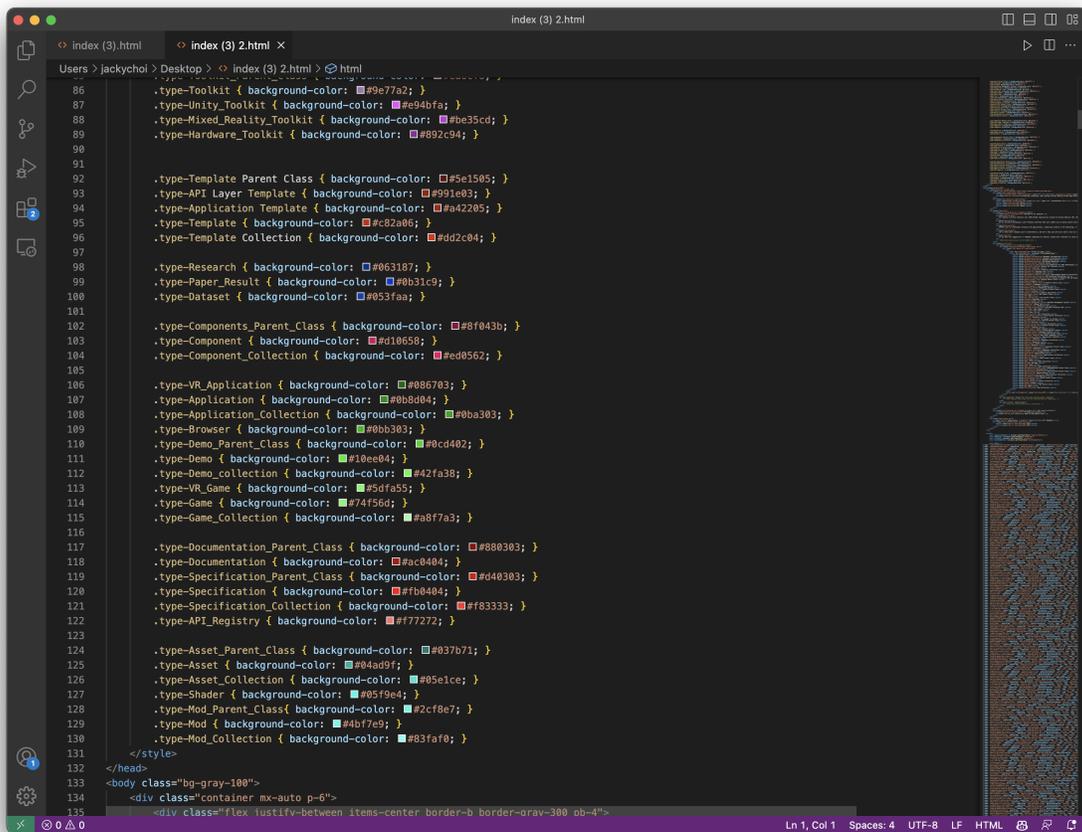


Figure 9: New website

distinguishing between subcategories. In contrast, the "Documentation" category is represented in red, with the parent category "Documentation" in a deep red color (880303) and its subcategory "Specification" in a slightly lighter shade of red (d40303). The size of each dot indicates the number of stars the project has received, with larger dots representing projects with more stars.



```
86 .type-Toolkit { background-color: #9e77a2; }
87 .type-Unity_Toolkit { background-color: #e94bf9; }
88 .type-Mixed_Reality_Toolkit { background-color: #be35cd; }
89 .type-Hardware_Toolkit { background-color: #892c94; }
90
91
92 .type-Template Parent Class { background-color: #5e1585; }
93 .type-API Layer Template { background-color: #991e03; }
94 .type-Application Template { background-color: #a42205; }
95 .type-Template { background-color: #c82a86; }
96 .type-Template Collection { background-color: #dd2c04; }
97
98 .type-Research { background-color: #063187; }
99 .type-Paper_Result { background-color: #0b31c9; }
100 .type-Dataset { background-color: #053faa; }
101
102 .type-Components_Parent_Class { background-color: #8f843b; }
103 .type-Component { background-color: #d10658; }
104 .type-Component_Collection { background-color: #ed0562; }
105
106 .type-VR_Application { background-color: #086703; }
107 .type-Application { background-color: #0b8d04; }
108 .type-Application_Collection { background-color: #0ba303; }
109 .type-Browser { background-color: #0bb303; }
110 .type-Demo_Parent_Class { background-color: #0cd402; }
111 .type-Demo { background-color: #10ee04; }
112 .type-Demo_collection { background-color: #42fa38; }
113 .type-VR_Game { background-color: #58fa55; }
114 .type-Game { background-color: #74f56d; }
115 .type-Game_Collection { background-color: #a8f7a3; }
116
117 .type-Documentation_Parent_Class { background-color: #880303; }
118 .type-Documentation { background-color: #ac0404; }
119 .type-Specification_Parent_Class { background-color: #d40303; }
120 .type-Specification { background-color: #fb0404; }
121 .type-Specification_Collection { background-color: #f83333; }
122 .type-API_Registry { background-color: #f77272; }
123
124 .type-Asset_Parent_Class { background-color: #037b71; }
125 .type-Asset { background-color: #04ad9f; }
126 .type-Asset_Collection { background-color: #05e1ce; }
127 .type-Shader { background-color: #0519e9; }
128 .type-Mod_Parent_Class { background-color: #2cf8e7; }
129 .type-Mod { background-color: #4bf7e9; }
130 .type-Mod_Collection { background-color: #83faf0; }
131
132 </style>
133 </head>
134 <body class="bg-gray-100">
135 <div class="container mx-auto p-6">
136 <div class="flex justify-between items-center border-b border-gray-300 pb-4">
```

Figure 10: Code of Color 1

The image shows a code editor window titled "index (3) 2.html". The editor displays a list of CSS classes, each with a background-color property. The classes are listed in a vertical column, with line numbers on the left. The background colors are represented by small colored squares followed by their hex codes. A sidebar on the right shows a file explorer view. At the bottom, there is a status bar with "Ln 1, Col 1" and "Spaces: 4 UTF-8 LF HTML". A notification at the bottom right says "Sign in to use GitHub Copilot." with a "Sign in to GitHub" button.

```
37 .type-Hardware_Introduction { background-color: #03432b; }
38 .type-Hardware_Architecture { background-color: #039a65; }
39 .type-Hardware_Description { background-color: #15bc06; }
40
41 .type-VR_Related_Resources { background-color: #091fec; }
42 .type-Tutorial_Parent { background-color: #135ae9; }
43 .type-Checklist_for_game_developing { background-color: #3f5bfa; }
44 .type-Device_DIY_Tutorial { background-color: #7c9477; }
45 .type-Tutorial { background-color: #7c9119; }
46 .type-Tutorial_Collection { background-color: #6466c4; }
47
48 .type-Related_List { background-color: #0975f8; }
49 .type-Awesome_List { background-color: #0f69b3; }
50 .type-Development_Resource_Collection { background-color: #2766af; }
51 .type-VR_Resources_Collection { background-color: #448cc8; }
52
53
54 .type-Library_Framework_SDK_and_Engine { background-color: #e9f586; }
55 .type-Engine_Parent_Class { background-color: #cbd518; }
56 .type-Engine { background-color: #9ea42b; }
57 .type-Framework_Parent_Class { background-color: #cfff65; }
58 .type-Framework { background-color: #fff400; }
59 .type-Test_Framework { background-color: #f8d861; }
60 .type-Rendering_System { background-color: #9eb027; }
61 .type-Library_Parent_Class { background-color: #c3cc20; }
62 .type-Library { background-color: #94ac0c; }
63 .type-Library_Collection { background-color: #e9f236; }
64 .type-SDK_Parent_Class { background-color: #eff75a; }
65 .type-SDK { background-color: #e9f23e; }
66
67
68 .type-Tool_Parent_Class { background-color: #de04f7; }
69 .type-Bindings { background-color: #e025f5; }
70 .type-Client { background-color: #e55ff4; }
71 .type-Database_Management_System { background-color: #ec8ef7; }
72 .type-Debug_Tool { background-color: #ef85f5; }
73 .type-Inspector_Tool { background-color: #cd75d7; }
74 .type-Low-code_scripting_Tool { background-color: #9957a1; }
75 .type-Mod_Loader { background-color: #6f2177; }
76 .type-Test_Tool { background-color: #9b23a9; }
77 .type-Tool { background-color: #7d1689; }
78 .type-Tool_Collection { background-color: #d655e4; }
79 .type-Unity_Editor_Extension { background-color: #ab75b1; }
80 .type-Polyfill { background-color: #6b4378; }
81 .type-R_wrapper_to_iframe { background-color: #dfafe4; }
82 .type-Utility_Parent_Class { background-color: #f1bbf7; }
83 .type-Utility { background-color: #f091f8; }
84 .type-Utility_Collection { background-color: #ea91f4; }
85 .type-Toolkit_Parent_Class { background-color: #ea6cf8; }
86 .type-Toolkit { background-color: #e07752; }
```

Figure 11: Code of Color 2

On the right side of the interface, you'll notice the Node Details module, which provides users with comprehensive information about individual nodes. When a node is selected, users can conveniently view details such as its name, update time, type, number of stars, number of forks, and the content of its Readme file.

We have introduced two additional design features to enhance the middle node module and provide more specialized functionality to our users. Firstly, we have included a filtering feature that allows users to search for projects based on their type, making it easier to find the specific projects they need. Secondly, we have introduced a "list view" option, which presents all nodes in a table format. This view includes columns for Name, Update Time, and Type, and an added filter in the table view enables users to further narrow down projects based on specific time criteria.

By implementing these upgrades, we aim to deliver a more efficient and user-friendly experience. Our goal is to provide advanced functionality while ensuring ease of use. We hope that these features will aid users in their work and enable them to accomplish their objectives more effectively.

Node rudrajikadra/Virtual-Reality-Tour-Unity-3D-World-Tour

Update time: 2020-05-30T15:56:35Z

Type: Application

Stars: 11

Forks: 7

Testing

Figure 12: Node Details

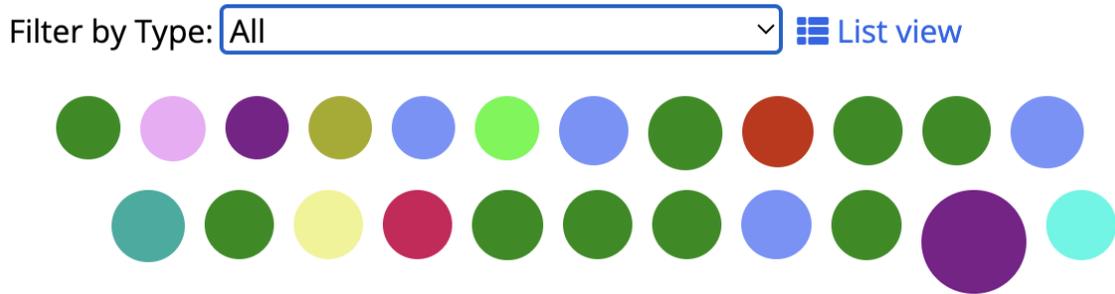


Figure 13: Filter and List View

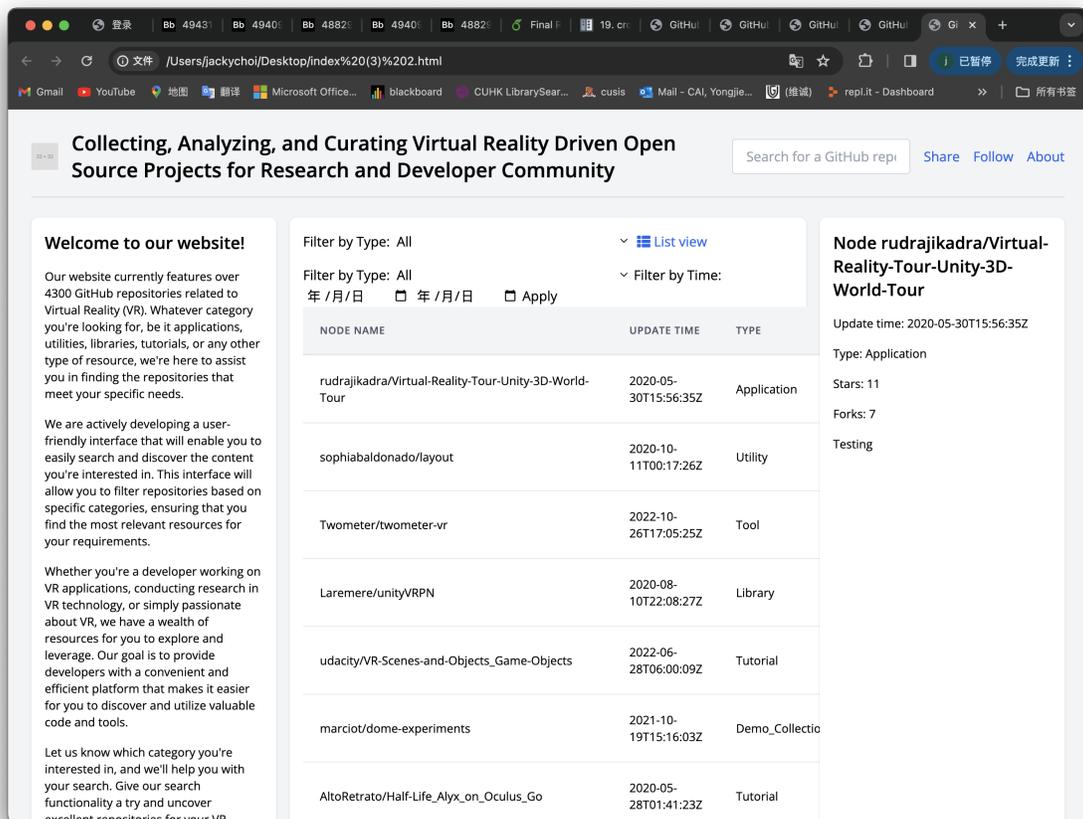


Figure 14: List View Table

7 Validation of the Dataset

7.1 Testing three cutting-edge VR open-source software tools: VRGuide, VRGreed, and VRMonkey (CAI)

To demonstrate the practicality of our dataset, we decided to conduct some experimental tests using it. We found two VR testing-related papers online: "VRTest: An Extensible Framework for Automatic Testing of Virtual Reality Scenes [18]" and "VRGuide: Efficient Testing of Virtual Reality Scenes via Dynamic Cut Coverage. [19]" Before conducting our own experiments, let's first understand what VRTest and VRGuide are.

VRTest is an extensible framework designed for automatically testing virtual reality (VR) scenes. It automates the testing process by extracting information from VR scenes, controlling the user's camera to explore the scenes, and interacting with virtual objects based on specific testing strategies. VRTest addresses the lack of well-studied quality assurance techniques for VR software and reduces the reliance on manual testing.

The framework supports two built-in testing strategies: VRMonkey and VRGreed. VRMonkey uses a pure random exploration strategy, while VRGreed employs a greedy algorithm to explore and interact with all interactable objects in VR scenes. These strategies offer different approaches to testing the functionality and behavior of VR software.

VRTest consists of several components, including the VR Scene Monitor, Virtual Object Instrumenter, Exploration Controller, Test Configuration Interface, and Testing Technique Interface. These components work together to fetch information about virtual objects, control the exploration process, and provide feedback to the testing technique.

The framework has been evaluated on top VR software projects and has demonstrated flexibility in incorporating different testing strategies. The evaluation results show that VRTest is capable of enhancing test coverage and improving the quality of VR software.

VRGuide, It is specifically designed for testing VR scenes and aims to address the unique challenges posed by testing VR applications. Existing testing frameworks like VRTest do not effectively handle issues such as object occlusion and movement in VR scenes.

VRGuide utilizes a computer geometry technique called Cut Exten-

sion to optimize the exploration of VR scenes. It calculates interaction values for different locations in the VR scene based on the positions of all objects. The camera is guided towards locations with the highest interaction values, allowing for more efficient coverage of interactable objects.

Through evaluations using various VR software projects, VRGuide demonstrates higher test coverage and improved testing efficiency compared to VRTest in most cases. It also successfully detects previously unknown bugs in real-world projects.

Now that we have a general understanding of these two testing tools, we will replicate their experiments using our dataset. We will use Unity version 2021.3 LTS with Visual Studio 2017 for compiling C# source code and conduct the experiments on a computer with an Intel Core i76500U CPU, 8GB of memory, and an Intel HD 520 Graphics card. We will set a timeout of 300 seconds, which is the default timeout value of VRTest. Testing of most subjects reaches saturation before 300 seconds.

In order to record method coverage, which was not mentioned in the paper regarding the recording method, we utilized the Unity Code Coverage tool [20] in our replication experiment. According to the official Unity website, the Code Coverage package is used in conjunction

with the Test Runner to gather and present test coverage information. Enabling code coverage during test runs allows us to precisely identify which lines of code were executed and whether the tests passed or failed. After completing a test run, the Code Coverage package generates an HTML coverage report that highlights the lines of code covered by the tests. Currently, Code Coverage supports both PlayMode and EditMode tests, and it also enables tracking of code coverage changes over time. Additionally, the Code Coverage package provides a Coverage Recording feature, allowing us to capture coverage data on demand, especially when there are no tests in the project or when conducting manual testing.

With the Code Coverage package, we are able to effectively record the method coverage for each project under the influence of three scripts: VRGreed, VRGuide, and VRMonkey.

As for the other important metric, Object coverage [21], we made certain modifications to the code. The modified test code automatically calculates the coverage by dividing the total number of objects by the number of discovered objects and reports it every 30 seconds. We conducted three experiments for each project and took the average to ensure the fairness of the testing process.

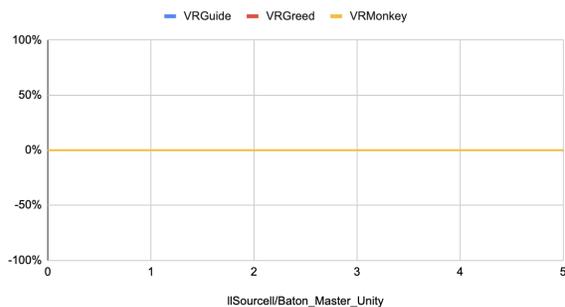
7.2 Reproduction of experiments using our dataset (CAI)

We conducted tests based on our VR dataset and extracted metadata, which successfully reproduced some of the conclusions. In all ten cases we examined, VRGuide consistently showed higher object and method coverage compared to VRGreed and VRMonkey. For example, in the Randomball example, VRGuide achieved 100% object coverage within two and a half minutes, while VRGreed required three minutes, and VRMonkey still did not reach 100% even after five minutes.

A	B	C	D	E	F	G	H	I	J	K	L
Randomball	0	0.5	1	1.5	2	2.5	3	3.5	4	4.5	5
VRGuide	0%	20%	30%	40%	70%	100%	100%	100%	100%	100%	100%
VRGreed	0%	10%	25%	30%	50%	70%	100%	100%	100%	100%	100%
VRMonkey	0%	10%	15%	20%	25%	30%	50%	60%	70%	70%	70%
UnityVR-master	0	0.5	1	1.5	2	2.5	3	3.5	4	4.5	5
VRGuide	0%	33%	66%	100%	100%	100%	100%	100%	100%	100%	100%
VRGreed	0%	33%	66%	100%	100%	100%	100%	100%	100%	100%	100%
VRMonkey	0%	0%	0%	33%	66%	66%	66%	66%	66%	66%	66%
XR Interaction D	0	0.5	1	1.5	2	2.5	3	3.5	4	4.5	5
VRGuide	0%	50%	100%	100%	100%	100%	100%	100%	100%	100%	100%
VRGreed	0%	50%	100%	100%	100%	100%	100%	100%	100%	100%	100%
VRMonkey	0%	0%	50%	100%	100%	100%	100%	100%	100%	100%	100%
VR_Escapae_Roi	0	0.5	1	1.5	2	2.5	3	3.5	4	4.5	5
VRGuide	0%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
VRGreed	0%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
VRMonkey	0%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
DataViz	0	0.5	1	1.5	2	2.5	3	3.5	4	4.5	5
VRGuide	0%	39%	47%	80%	100%	100%	100%	100%	100%	100%	100%
VRGreed	0%	23%	58%	64%	77%	98%	100%	100%	100%	100%	100%
VRMonkey	0%	10%	30%	30%	50%	50%	50%	50%	50%	50%	50%
UnityVR-maze-1	0	0.5	1	1.5	2	2.5	3	3.5	4	4.5	5
VRGuide	0%	26%	47%	70%	97%	100%	100%	100%	100%	100%	100%
VRGreed	0%	38%	58%	91%	97%	97%	97%	97%	97%	97%	97%
VRMonkey	0%	20%	30%	30%	30%	30%	30%	30%	30%	30%	30%

Figure 15: Test Result in Excel

VRGuide、VRGuide 和VRGuide



VRGuide、VRGreed 和VRMonkey

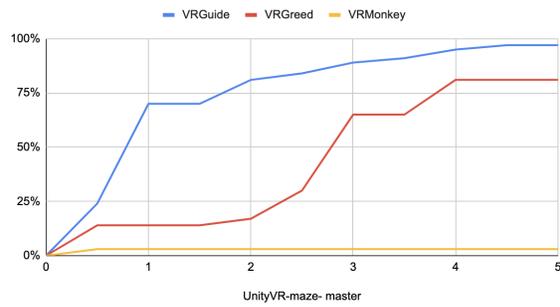
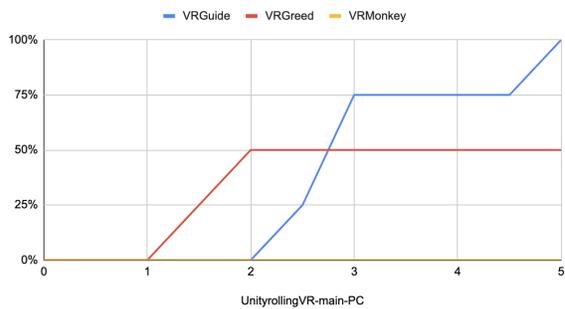


Figure 16: Test Result 1

VRGuide、VRGreed 和VRMonkey



VRGuide、VRGreed 和VRMonkey

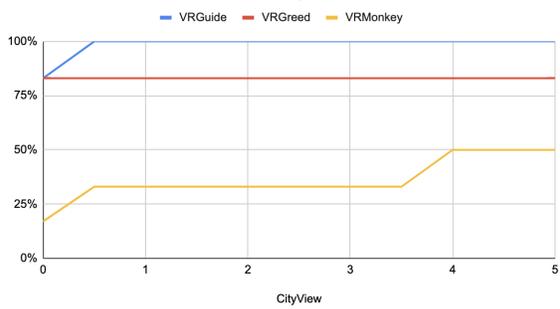
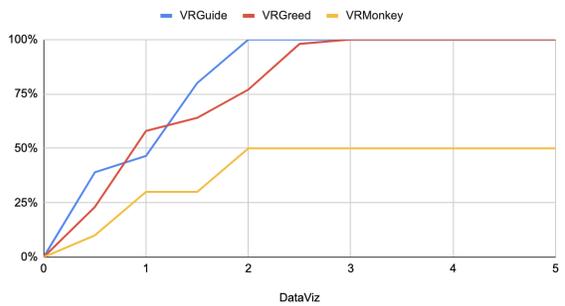


Figure 17: Test Result 2

VRGuide、VRGreed 和VRMonkey



VRGuide、VRGreed 和VRMonkey

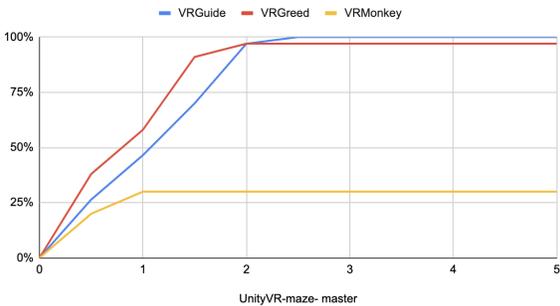
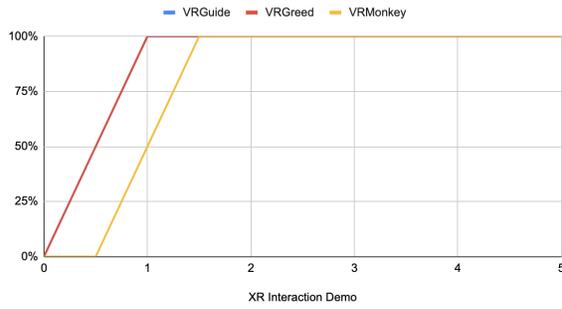


Figure 18: Test Result 3

VRGuide、VRGreed 和VRMonkey



VRGuide、VRGreed 和VRMonkey

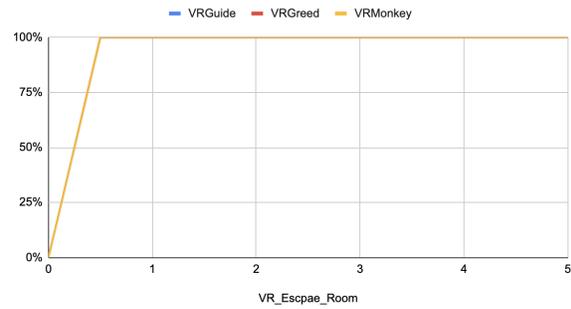
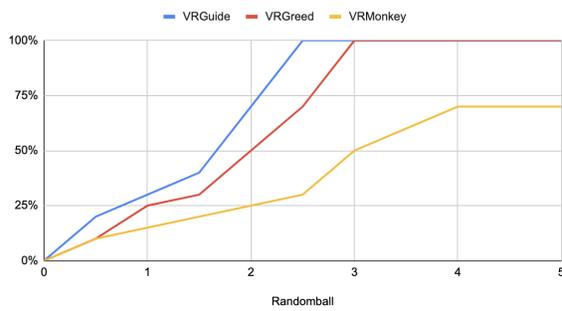


Figure 19: Test Result 4

VRGuide、VRGreed 和VRMonkey



VRGuide、VRGreed 和VRMonkey

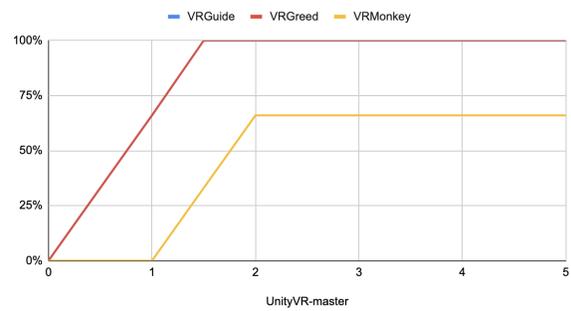


Figure 20: Test Result 5

During the process of reproducing the experiments, we identified several issues that could be areas for improvement in the two papers. Firstly, the testing described in the papers was imprecise. The VRGuide paper presented an example called "RondomBlocks" where the initial positions of objects and the player in the Unity project were randomly generated. As a result, each test run produced different results for coverage evaluation. The papers did not mention the number of tests conducted or whether the coverage results were averaged. To ensure more accurate results, we conducted three tests for each project and calculated the average coverage.

Secondly, during our testing, we discovered a problem that needs improvement in both VRGuide, VRGreed and VRMonkey scripts. For some projects, all three scripts would bring the player to a position where they could no longer move forward. In this state, the player's XYZ coordinates would have two axes fixed while the other would oscillate within a small range (e.g., 0.01). Consequently, the player was unable to interact with other objects, resulting in a significant decrease in object and method coverage. We resolved this issue by improving the code, which significantly increased coverage.

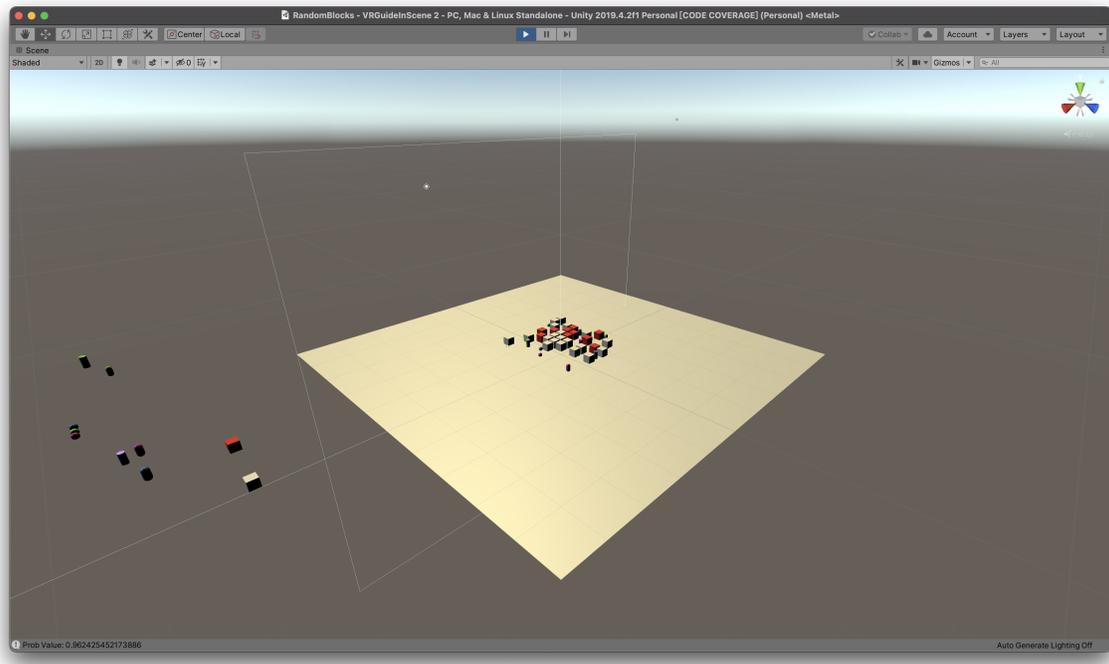


Figure 21: RandomBlocks

These are the key points we have observed during our testing, and we will further discuss these issues to enhance the understanding and improve the results.

7.3 Identification of issues in the tools and enhancement through the some rules (WEI)

In the process of reproducing the results of the paper, we encountered an issue that the original paper did not mention. When the player follows the scripted path, there are instances where the player gets stuck. For example, while testing the UnityVR-master project, whose complete code can be found in the dataset provided by the original paper, the VR project's scene is a closed room with furniture. Starting the game in Unity, within three seconds, the player gets stuck very close to the target location, oscillating within a small range (typically around 0.2). After pausing the game and manually adjusting the coordinates, the player is able to move normally again. This issue also occasionally occurs when testing other scripts. There could be two reasons for this issue:

1. The player is colliding with other objects on the path.



Figure 22: Example for player being stuck

2. Since the stuck locations are very close to the target coordinates, it might be due to the precision of floating-point numbers; even though the player has passed the trigger point, it is still judged as not having reached the destination.

To determine the exact cause of the stalling, we modified the original code. The scripts provided by the original author were all based on modifications derived from `VRTest.cs`, and we tried to concentrate the modifications within the `VRTest.cs` framework to reduce the workload of subsequent modifications.

In the initial `VRTest` script, the author used raycasting to detect obstacles. If the implementation of raycasting was not accurate enough, it might fail to detect obstacles correctly. Therefore, I modified the `Mov-`

able function used by the original author for obstacle detection, adding more raycasting statements. The main modifications include:

1. Adding the `Physics.DefaultRaycastLayers` parameter in the `Physics.Raycast` function to ensure that only the default collision layers are considered during raycasting, avoiding obstruction by objects from other layers.
2. Adding the `QueryTriggerInteraction.Ignore` parameter in the `Physics.Raycast` function to ensure that triggers are ignored during raycasting, preventing them from obstructing the player.

```
public virtual bool Movable(Vector3 position, int flag)
{
    switch (flag)
    {
        case 0:
            return position.x + moveStep < moveUpperBound.x && !Physics.Raycast
                (position, Vector3.right, moveStep, Physics.DefaultRaycastLayers, QueryTriggerInteraction.Ignore);
        case 1:
            return position.x - moveStep > moveLowerBound.x && !Physics.Raycast
                (position, Vector3.left, moveStep, Physics.DefaultRaycastLayers, QueryTriggerInteraction.Ignore);
        case 2:
            return position.y + moveStep < moveUpperBound.y && !Physics.Raycast
                (position, Vector3.up, moveStep, Physics.DefaultRaycastLayers, QueryTriggerInteraction.Ignore);
        case 3:
            return position.y - moveStep > moveLowerBound.y && !Physics.Raycast
                (position, Vector3.down, moveStep, Physics.DefaultRaycastLayers, QueryTriggerInteraction.Ignore);
        case 4:
            return position.z + moveStep < moveUpperBound.z && !Physics.Raycast
                (position, Vector3.forward, moveStep, Physics.DefaultRaycastLayers, QueryTriggerInteraction.Ignore);
        case 5:
            return position.z - moveStep > moveLowerBound.z && !Physics.Raycast
                (position, Vector3.back, moveStep, Physics.DefaultRaycastLayers, QueryTriggerInteraction.Ignore);
        default:
            return false;
    }
}
```

Figure 23: Raycaster Object Detec

However, the Unity console still did not print any information about

obstacles, indicating that collisions were not causing the issue.

We then investigated the second reason, to prevent the player from vibrating at the target location due to the precision of floating points when too close to an interactable target object. We could modify the `FixedUpdate` function, adding a tolerance value to determine whether the target position has been reached and immediately set a new target position and rotation upon reaching. The main modifications are as follows:

1. Defined two tolerance values, `positionTolerance` and `rotationTolerance`. When the player's position and rotation are close to the target position and target rotation within these tolerance ranges, it is considered that the target has been reached.
2. Defined two tolerance values, `positionTolerance` and `rotationTolerance`. When the player's position and rotation are close to the target position and target rotation within these tolerance ranges, it is considered that the target has been reached.
3. After setting the new target position and rotation, immediately call the `Trigger` function to ensure that related objects are correctly triggered upon reaching the new target.

After addressing the floating-point precision issue, which allowed for normal triggering of interactable targets, we encountered a subsequent

```

void FixedUpdate()
{
    if (clickStay)
    {
        passed = passed + Time.deltaTime;
        if (passed > clickStayLength)
        {
            clickStay = false;
            passed = 0.0f;
        }
    }
    else
    {
        // Define a threshold
        float positionTolerance = 0.01f;
        float rotationTolerance = 0.01f;

        transform.position = Vector3.MoveTowards(transform.position, dest, moveStep * 0.02f);
        Quaternion q = Quaternion.RotateTowards(transform.rotation, destrotate, turnStep * 0.02f);
        q.eulerAngles = new Vector3(q.eulerAngles.x, q.eulerAngles.y, 0f);
        transform.rotation = q;

        // Whether it reaches the destination and rotation
        if (Vector3.Distance(transform.position, dest) < positionTolerance && Quaternion.Angle(transform.rotation, destrotate) <
            rotationTolerance)
        {
            // Set the position and rotation as expected
            transform.position = dest;
            transform.rotation = destrotate;

            // Reset the new destination and rotation
            destrotate = Turn();
            Debug.Log("rotate:" + destrotate);
            dest = Move();
            Debug.Log("dest:" + dest);
            Trigger();
        }
    }
}

```

Figure 24: fixupdate

problem: when two consecutive targets are positioned too closely, the player oscillates between these two points after interacting with the item at the previous target location. To address this, we made modifications that included setting a time interval—specifically, 5 seconds. If the player’s position remains within a 0.1 unit distance from the last target for the duration of this interval, then the player is reset to a randomly selected new position.

These modifications were implemented through the following adjustments in the code:

1. Added `resetTimer` and `resetInterval` variables to track the reset timer and reset interval.

```
private float resetTimer = 0f;  
0 references  
private float resetInterval = 5f; // Reset time interval, adjust as needed  
0 references  
private Vector3 resetPosition;
```

Figure 25: `resetTimer` and `resetInterval` added

2. Added a `resetPosition` variable to record the previous target position to determine whether the player is too close to it.
3. In the `Start` function, initialized `resetPosition` to the initial position.

```
dest = transform.position;  
destrotate = transform.rotation;  
resetPosition = transform.position;  
// initialize reset position to initial position
```

Figure 26: `resetPosition` added

4. In the `FixedUpdate` function, when the player successfully reaches a target position, the `resetTimer` is reset to zero, and the `resetPosition`

is updated to reflect the newly reached target location designated by `dest`.

```
float positionTolerance = 0.01f;
float rotationTolerance = 0.01f;

transform.position = Vector3.MoveTowards(transform.position, dest, moveStep * 0.02f);
Quaternion q = Quaternion.RotateTowards(transform.rotation, destrotate, turnStep * 0.02f);
q.eulerAngles = new Vector3(q.eulerAngles.x, q.eulerAngles.y, 0f);
transform.rotation = q;

if (Vector3.Distance(transform.position, dest)
    < positionTolerance && Quaternion.Angle(transform.rotation, destrotate)
    < rotationTolerance)
{
    transform.position = dest;
    transform.rotation = destrotate;

    destrotate = Turn();
    dest = Move();
    Trigger();

    // Reset timer and reset position
    resetTimer = 0f;
    resetPosition = dest;
}
```

Figure 27: Reset when reaches a target position

5. In the `FixedUpdate` function, if the player is less than 0.1 units away from the previous target position `resetPosition`, start the timer. If the timer exceeds `resetInterval`, call the newly added `ResetPlayerPosition` function to reset the player's position.
6. The `ResetPlayerPosition` function randomly generates a new position

within the movement boundary, sets the player's position, target position dest, and target rotation destrotate to this new position, and resets the resetTimer.

```
private void ResetPlayerPosition()
{
    // Generate a new random position
    Vector3 newPosition = new Vector3(UnityEngine.Random.Range(moveLowerBound.x, moveUpperBound.x),
                                     UnityEngine.Random.Range(moveLowerBound.y, moveUpperBound.y),
                                     UnityEngine.Random.Range(moveLowerBound.z, moveUpperBound.z));

    // Set player to the new position
    transform.position = newPosition;
    dest = newPosition;
    destrotate = transform.rotation;

    // Reset timer
    resetTimer = 0f;
}
```

Figure 28: The ResetPlayerPosition

7.4 How can we improve object coverage and method coverage (WEI)

In this segment of our final year project report, we explore two significant metrics used to assess the efficiency and quality of script testing in Unity: Object Coverage and Method Coverage. These metrics are crucial for evaluating the robustness and comprehensiveness of testing scripts within the virtual environment.

Object Coverage: is derived by calculating the ratio of interactable objects triggered by different scripts within a span of five minutes to the total number of interactable objects available in the project. This metric provides a standard to measure the testing script's efficiency in interacting with the environment, which is vital for ensuring comprehensive test coverage in interactive applications. **Method Coverage**, on the other hand, is a well-established metric within the field of software testing, particularly in the context of Unity. Unity defines Method Coverage as a measure within code coverage tools that quantifies the percentage of methods that are covered by unit tests during their execution relative to the total number of methods. Unity provides an official tool for testing Method Coverage, which can be accessed and analyzed through the Unity Test Runner tool. The calculation for Method Coverage is represented by the formula:

$$\text{Method Coverage} = \left(\frac{\text{Total number of methods}}{\text{Number of methods executed}} \right) \times 100\% \quad (1)$$

Our first measurement of Method Coverage was only 6.9 percent. This low percentage indicated that the scripts we were using in our unit tests

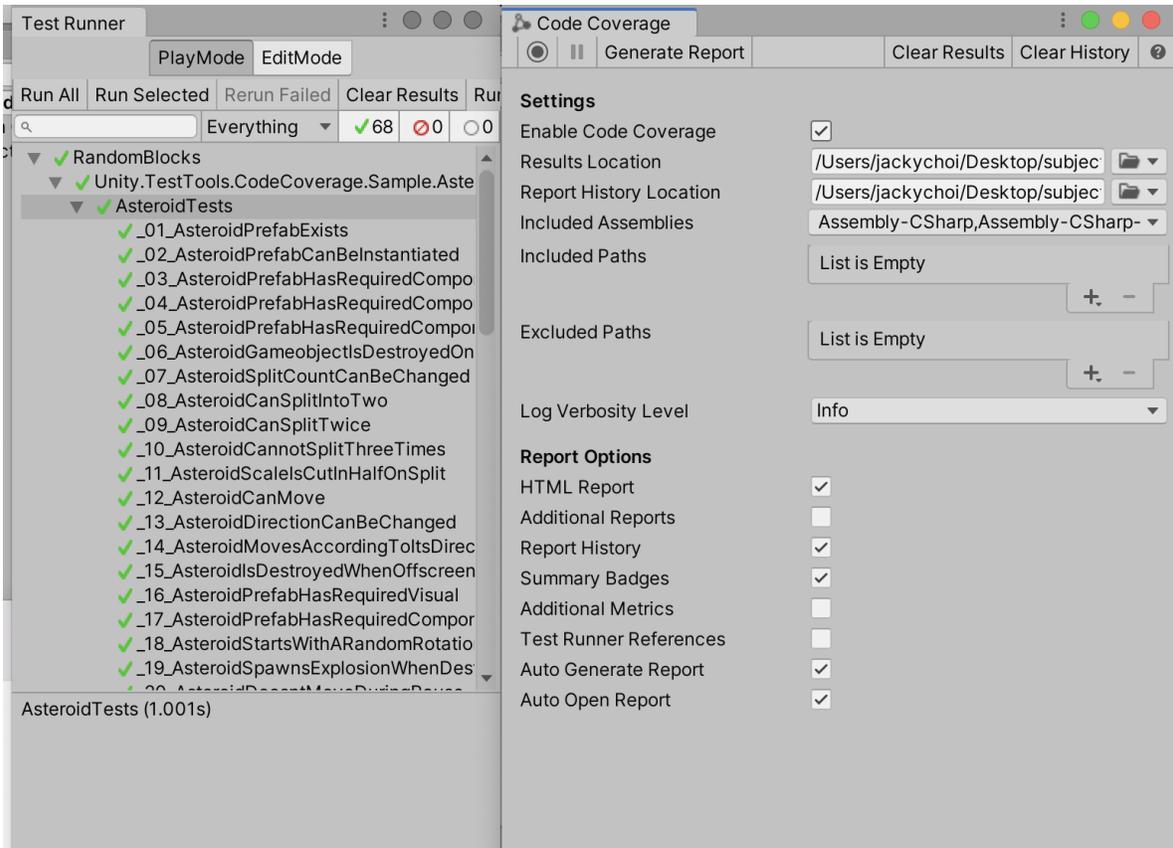


Figure 29: The Official Tool to Measure Method Coverage

were not sufficiently comprehensive, failing to execute a large portion of the available methods in the codebase. To address this issue, we delved deeper into the testing framework and identified a significant problem related to object stasis during tests, where objects were not responding or updating as expected, thereby not triggering the associated methods. After implementing a series of optimizations to resolve this stalling issue, we observed a substantial improvement in our test results.

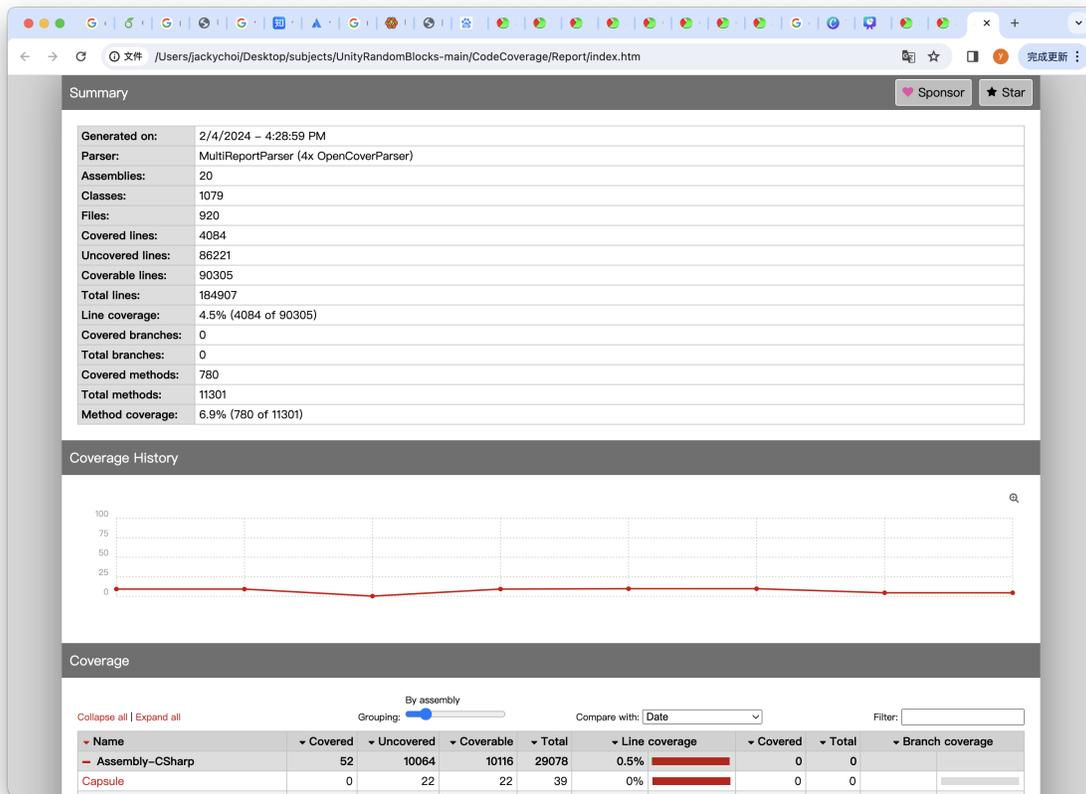


Figure 30: Initial Result Under Expectation

Post-improvement, the Method Coverage metric rose to 45 percent. This enhancement not only indicates a more comprehensive engagement of the scripts with the codebase but also reflects an increased reliability in our testing process. The adjustments made to the test scripts have allowed for a more dynamic interaction during test runs, ensuring that a greater number of methods are being executed. These findings underscore the importance of continuous refinement and optimization in test

Covered methods:	12
Total methods:	16
Method coverage:	75% (12 of 16)

Coverage History



Coverage

Collapse all | Expand all Grouping: By assembly Compare with: Date Filter:

Name	Covered	Uncovered	Coverable	Total	Line coverage	Covered	Total	Branch coverage
Assembly-CSharp	43	52	95	234	45.2%	0	0	
Autowalk	15	26	41	93	36.5%	0	0	
Exit	0	3	3	11	0%	0	0	
PlayTV	5	0	5	15	100%	0	0	
RunPC	1	21	22	44	4.5%	0	0	
SoundPlayer	4	0	4	15	100%	0	0	
SpawnPaper	7	0	7	27	100%	0	0	
Timeout	11	2	13	29	84.6%	0	0	

Generated by ReportGenerator 5.0.10

Figure 31: Method Coverage After Optimization

scripts, especially in environments as complex and interactive as those created in Unity. The improvements in Method Coverage not only enhance the quality of the testing process but also contribute to the overall development cycle by ensuring that potential issues are identified and addressed promptly.

8 Future Work

8.1 Exploration of potential directions for future research and development (CAI)

Although we have made progress in addressing latency issues and improving test coverage, there is still plenty of room for improvement.

For example, one area for improvement is enhancing test accuracy and addressing inconsistencies in initial conditions and random object generation within virtual reality scenarios. [22] Development techniques can be employed to ensure test repeatability and accuracy by controlling and standardizing the initial state of objects and players. This could involve defining specific starting configurations or employing techniques to generate consistent initial conditions for testing, rather than relying on direct testing three times and averaging the results.

Additionally, optimizing test coverage is another area that can be improved in the future. Continued research and development are needed to devise algorithms or strategies that achieve optimal test coverage in virtual reality scenarios. Advanced exploration techniques can be explored, taking into account object occlusion, movement, and complex

scene geometry. Unlike the original paper's focus on comparing a single test subject, it is essential to ensure that algorithms can handle more sophisticated scenarios. This may involve utilizing computer geometry techniques, path planning algorithms, or machine learning methods to guide the exploration process and maximize coverage.

Furthermore, enhancing the stability and robustness of the virtual reality testing framework can be achieved by addressing issues such as player getting stuck or inability to interact with objects. Research and development efforts can be directed towards detecting and handling problematic situations, such as identifying unreachable areas or resolving obstructive collisions that impede progress.

Another important aspect is achieving multi-platform testing. Currently, the testing framework described in the paper is based on the Windows system. However, the VR software available in the market encompasses various platforms, some of which are not mutually compatible. For instance, when attempting to test the project from the paper on a Mac computer, a "DllNotFoundException: user32.dll" [23] error occurs. This error arises because the testing script, `MouseOperations.cs`, references `user32.dll`, which is specific to the Windows operating system. To

address this issue, we need to modify the code in `MouseOperations.cs` to utilize the `CoreGraphics` framework specific to macOS for mouse operations. The `CGSetLocalEventsSuppressionInterval` function is used to set the event suppression interval, and the `CGPostMouseEvent` function is used to simulate mouse events. By doing so, we can carry out our tests successfully. Therefore, expanding the virtual reality testing framework to support multiple virtual reality platforms and devices is crucial. Considering the unique characteristics and functionalities of different virtual reality systems, such as HTC Vive, Oculus Rift, or PlayStation VR, is essential for ensuring compatibility and comprehensive testing across various platforms.

Automation and tool integration also require further improvement in the future. Although we have achieved some level of automation in our implementation, the original paper's testing process is still cumbersome. Configuring three scripts, namely "VRGuide," "VRGreed," and "VRMonkey," manually for object coverage or method coverage testing for each project, along with manual result recording, is demanding. In our reproduction, we have added features for automatic counting and script configuration, which has partially automated the testing process

compared to the previous version. However, there is room for further improvement.

These are some areas where we can make future attempts and strive for improvement.

8.2 Potential use of databases for debugging in virtual reality projects (CAI)

I understand that the project's overall goal is to improve VRTest and VRGuide by identifying and addressing any issues through the use of our dataset. So far, we have seen a 25% increase in both method coverage and object coverage, which shows the effectiveness of our VR dataset in conducting testing experiments.

Our dataset is not limited to VR testing as it can be used as an experimental playground for developers of testing tools. Our user-friendly website provides easy access to the testing subjects, enabling developers to identify their own issues and make improvements. This is the outcome we strive to achieve.

In addition to VR testing, our dataset can also be beneficial for VR game developers who are looking for suitable testing tools. We have

included numerous VR testing tools, such as the improved Unity Science tool, which is available on our website for game developers to use. We plan to release more testing tools in the future to expand our dataset even further.

Overall, our VR dataset is a versatile library and tool repository capable of fulfilling the requirements of VR project developers.

8.3 Proposal of areas where improvements can be made in virtual reality software (WEI)

Although developers are increasingly interested in creating Virtual Reality (VR) projects that offer immersive experiences to users, there has been a consistent lack of interest in the development of VR testing tools. According to a paper by Qing Liu, Gustavo Alves, and others [24], the difficulty in finding and reproducing bugs is ranked as the second most significant challenge in VR software development. The intricate 3D modeling environments often leave developers struggling to pinpoint small errors.

To address these challenges, we have developed modified versions of VR testing tools: VRGuide, VRGreed, and VRMonkey. These tools are

designed to enhance the testing phase of VR development by detecting all interactive objects within a VR scene. They can trigger these objects and subsequently output the results in the console. This functionality not only aids developers in ensuring that all elements are responsive and interact as expected but also simplifies the process of identifying and fixing bugs in complex VR environments.

These tools represent a step forward in improving the efficiency and effectiveness of VR software testing. By providing developers with the ability to automatically test and verify the interactive components of their VR environments, we can significantly reduce the time and effort involved in the debugging process. This is crucial for the advancement of VR technology, as it ensures that developers can focus more on creativity and user experience, rather than being bogged down by technical setbacks.

8.4 Consideration of the integration of AI technologies (LLM model) in enhancing VR experiences (CAI)

When we use ChatGPT [25] on our VR dataset and integrate its capabilities into our website testing tool, our VR dataset becomes much more practical.

With ChatGPT, users can have natural conversations with characters in the virtual environment. They can quickly find the desired content from the dataset, with precise and ample samples. Additionally, AI responses are faster, making it more convenient to use.

Furthermore, ChatGPT can provide personalized answers and guidance based on users' preferences and needs. Whether it's recommending relevant VR open-source projects based on user interests or offering tailored suggestions for testing or software development based on the user's context.

Moreover, users can ask ChatGPT specific questions about VR scenes, objects, or actions in a particular VR project, and receive detailed explanations and guidance. ChatGPT can help explain the meaning, function-

ality, and usage of elements in the virtual environment, enabling users to gain a deeper understanding and explore the VR experience more thoroughly.

By using ChatGPT as a user support and training tool on our website, users can receive real-time assistance, get answers to their questions, or learn techniques and skills for using the VR environment. ChatGPT can provide instant tutorials, advice, or solutions, helping users overcome difficulties and improve their effectiveness and skills.

To sum up, by using ChatGPT on our VR dataset or integrating AI capabilities into our website testing tool, users will benefit from in-depth scene explanations and real-time user support and training. This will greatly aid in the development of better VR software for users.

9 Conclusion

9.1 Recapitulation of the study's objectives and key findings (WEI)

The final year project embarked on a detailed exploration into the domain of Virtual Reality (VR) and Extended Reality (XR) through the lens

of GitHub repository analysis from the period of 2020 to June 2023. The project was meticulously planned with multiple objectives aimed at enhancing the understanding and development within these technological fields. Here, we provide an overview of the objectives and summarize the significant findings:

1. **Data Collection with Web Scraping:** The project successfully deployed a web crawler specifically designed to extract data from GitHub repositories containing keywords related to VR and XR. This objective was pivotal in gathering a rich dataset for further analysis.
2. **Script Improvements for Regular Updates:** The project enhanced the scraping scripts, enabling them to perform regular updates. This ensures the continuous relevance of the data by capturing the latest developments and changes within the VR/XR repositories.
3. **Efficient Repository Access:** By downloading READMEs, metadata, and file listings, the project significantly eased the process for developers and researchers to quickly locate and review relevant repositories, thereby facilitating smoother research and development activities.
4. **Data Storage and Accessibility:** Uploading the scraped data to a remote server was a crucial step that enhanced data security, reliability, and accessibility across

different geographic locations and platforms.

5. **Data Visualization:** The project did not merely focus on data collection and storage but extended into the realm of data visualization, providing critical insights through graphical representations of the analyzed data, which helped in understanding trends and patterns effectively.
6. **Replication and Enhancement of VR Testing Scripts:** The project undertook a challenging task of replicating and improving upon the scripts from a VR testing study, namely VRGreed, VRMonkey, and VRGuide. The enhancements led to increased method coverage and resolved issues related to object stasis during testing, thus contributing to more robust VR testing methodologies.

9.2 Summary of the contributions and implications of the research (WEI)

The contributions of this project extend beyond the academic fulfillment of a final year project; they have practical and theoretical implications in the fields of VR and XR:

1. **Advancement in Data Collection Techniques:** The development of a specialized web crawler for VR/XR repositories represents a significant advancement in data collection techniques within these fields, enabling stakeholders to keep pace with

rapid technological advancements.

2. **Improvements in Software Development Practices:** The enhancements made to the scripts for regular updates promote best practices in software development, particularly in terms of maintainability and scalability.
3. **Enhanced Research Efficiency:** The availability of well-organized and easily accessible data through the remote server empowers researchers and developers to conduct more efficient and effective studies, thus accelerating innovation in VR/XR technologies.
4. **Insights through Visualization:** The visualizations created as part of this project not only aid in the easier comprehension of complex datasets but also serve as a vital tool for presenting data-driven insights in a user-friendly manner, which is crucial for both academic and industrial stakeholders.
5. **Contribution to VR Testing Methodologies:** By replicating and enhancing existing VR testing scripts, the project has contributed to the body of knowledge in VR testing, offering improved tools and methodologies that enhance the reliability and effectiveness of VR applications.

References

- [1] Apple. (2024) Apple vision pro. [Online]. Available: <https://www.apple.com/apple-vision-pro/>
- [2] D. E. Rzig, N. Iqbal, I. Attisano, X. Qin, and F. Hassan, “Virtual reality (vr) automated testing in the wild: A case study on unity-based vr applications,” in *Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis*, 2023, pp. 1269–1281.
- [3] S. Mystakidis, “Metaverse,” *Encyclopedia*, vol. 2, no. 1, pp. 486–497, 2022.
- [4] Meta. (unknown) The metaverse is the future of digital connection — meta. [Online]. Available: <https://about.meta.com/metaverse/>
- [5] —, “Meta reports second quarter 2023 results,” Meta, Second Quarter Results, 2023.
- [6] S. Kraus, D. K. Kanbach, P. M. Krysta *et al.*, “Facebook and the creation of the metaverse: radical business model innovation or incre-

- mental transformation?” *International Journal of Entrepreneurial Behavior & Research*, vol. 28, no. 9, pp. 52–77, 2022.
- [7] R. R. Jitesh Ubrani, Ramon Llamas, “Ar/vr another slow year expected for ar/vr headsets before 2024 rebound, according to idc,” 10 2023.
- [8] A. Heath, “Meta’s flagship metaverse app is barely used by the employees building it,” 2022, [Online]. Available: <https://www.theverge.com/2022/10/6/23391895/meta-facebook-horizon-worlds-vr-social-network-too-buggy-leaked-memo>.
- [9] C. Donalek, S. G. Djorgovski, A. Cioc *et al.*, “Immersive and collaborative data visualization using virtual reality platforms,” in *2014 IEEE International Conference on Big Data (Big Data)*. IEEE, 2014, pp. 609–614.
- [10] J. Zhao, J. O. Wallgrün, P. C. LaFemina *et al.*, “Harnessing the power of immersive virtual reality-visualization and analysis of 3d earth science data sets,” *Geo-spatial Information Science*, vol. 22, no. 4, pp. 237–250, 2019.

- [11] M. Akdere, Y. Jiang, and F. D. Lobo, “Evaluation and assessment of virtual reality-based simulated training: exploring the human–technology frontier,” *European Journal of Training and Development*, vol. 46, no. 5/6, pp. 434–449, 2022.
- [12] Q. Zhang, K. Wang, and S. Zhou, “Application and practice of vr virtual education platform in improving the quality and ability of college students,” *IEEE Access*, vol. 8, pp. 162 830–162 837, 2020.
- [13] I. M. Franz-Xaver Geiger, “Datasets of android applications: a literature review,” 9 2018.
- [14] C.-P. B. Rain Epp, Dayi Lin, “An empirical study of trends of popular virtual reality games and their complaints,” *IEEE Transactions on Games*, 9 2021.
- [15] “19. crontab — linux tools quick tutorial,” <https://linuxtools-rst.readthedocs.io/zh-cn/latest/tool/crontab.html>, accessed on 15th April 2024.
- [16] M. Tanaka. (2014) C3.js — d3-based reusable chart library. [Online]. Available: <https://c3js.org/>

- [17] “Chart.js documentation,” <https://www.chartjs.org/docs/latest/>.
- [18] X. Wang, “Vrtest: An extensible framework for automatic testing of virtual reality scenes,” in *Proceedings of the ACM/IEEE 44th International Conference on Software Engineering: Companion Proceedings*, 2022, pp. 232–236.
- [19] M. Peter, R. Horst, and R. Dörner, “Vr-guide: A specific user role for asymmetric virtual reality setups in distributed virtual reality applications,” in *Proceedings of the 10th Workshop Virtual and Augmented Reality of the GI Group VR/AR*, 2018, pp. 83–94.
- [20] Unity Technologies. (2023) Code coverage — code coverage — 1.1.1. [Online]. Available: <https://docs.unity3d.com/Packages/com.unity.testtools.codecoverage@1.1/manual/index.html#code-coverage-package>
- [21] ——. (2024) Unity - manual: Vr development in unity. [Online]. Available: <https://docs.unity3d.com/Manual/VROverview.html>
- [22] M. T. Schultheis, J. Himmelstein, and A. A. Rizzo, “Virtual reality and neuropsychology: upgrading the current tools,” *The Journal of head trauma rehabilitation*, vol. 17, no. 5, pp. 378–394, 2002.

- [23] Unity Technologies. (2023) Dllnotfoundexception: user32.dll. [Online]. Available: <https://forum.unity.com/threads/dllnotfoundexception-user32-dll.475641/>
- [24] J. Z. Qing Liu, Gustavo Alves, “Challenges and opportunities for software testing in virtual reality application development,” *2019 21st Symposium on Virtual and Augmented Reality (SVR)*, 12 2019.
- [25] OpenAI. (2021) Openai chatgpt. [Online]. Available: <https://chat.openai.com/>