UNIVERSITÄT
BERN

Pervasive Visualization in Immersive Augmented Reality for Software Performance Monitoring

Bachelor Thesis

Mario Hess

from

Biel BE, Switzerland

Faculty of Science
University of Bern

08 February 2019

Prof. Dr. Oscar Nierstrasz

Dr. Leonel Merino

Software Composition Group
Institute of Computer Science
University of Bern, Switzerland

Abstract

Developers are usually unaware of the impact of code changes to the performance of software systems. Although developers can analyze the performance of a system by executing, for instance, a performance test to compare the performance of two consecutive versions of the system, changing from a programming task to a testing task would disrupt the development flow. Most performance visualization tools provide the user with a detailed view, which can be overwhelming and cause a high cognitive load.

In this thesis, we propose the use of a city visualization that dynamically provides developers a pervasive view of the continuous performance of a system and lessens the cognitive load required to monitor it. We use an immersive augmented reality device (Microsoft HoloLens) to display our visualization and extend the integrated development environment on a computer screen to use the physical space. We report on technical details of the design and implementation of our visualization tool. Our effort explores a new visual metaphor to support the exploration and analysis of possibly very large and multidimensional performance data. Our initial result indicates that the city metaphor can be effectually employed to analyze dynamic performance data on a large and non-trivial software system.

Additionally, we conducted an initial user study with ten participants, comparing performance and user experience in immersive augmented reality to that with a standard computer screen, and we report on the results. We asked participants to complete a set of ten tasks inspired by questions that arise from performance issues. To measure performance, we collected correctness, completion time and recollection of users. We measured user experience by collecting emotions that participants experienced during experiments, and we also measured perceived cognitive load. We observe that participants achieve comparable performance in immersive augmented reality and with a computer screen. We found that developers felt: (i) *interested*, (ii) *open* but also (iii) *confused* and (iv) *angry* when working in immersive augmented reality while achieving comparable performance to working with a standard computer screen.

Contents

1 Introduction

Among the many questions that arise during software development, programmers often formulate questions about the performance of software systems [2, 12]. They ask, for instance, "what is the software doing when performance issues arise?" [24], and "where is most of the time being spent?" [25]. Since developers introduce multiple changes to the source code during the implementation of a software system, being aware of the impact of a code change task to the performance of a software system is an important concern for developers [12].

Methods such as profilers and performance tests allow developers to analyze the performance of only a single version of a system at a time, and force developers to change to a different task (interrupting their flow). Our intuition is that developers would benefit from analyzing the live performance of an evolving system as its source code changes. We expect that such an analysis would help them identify changes that severely decrease the performance of a system.

We conjecture that a pervasive tool, (*i.e.*, a tool that is omnipresent during the implementation of a software system) can make developers aware of important changes in the performance of a system without disrupting the implementation flow. Although using this visualization approach, developers could obtain such a pervasive view of the live performance of a system, we observe that they can be reluctant to sacrifice space in their integrated development environment (IDE). Developers willing to adopt a visualization tool to monitor software performance, either need to use a stand-alone tool that forces them to leave the IDE, or they need to include an add-on to their IDE, sacrificing valuable screen space. We consider that a tool

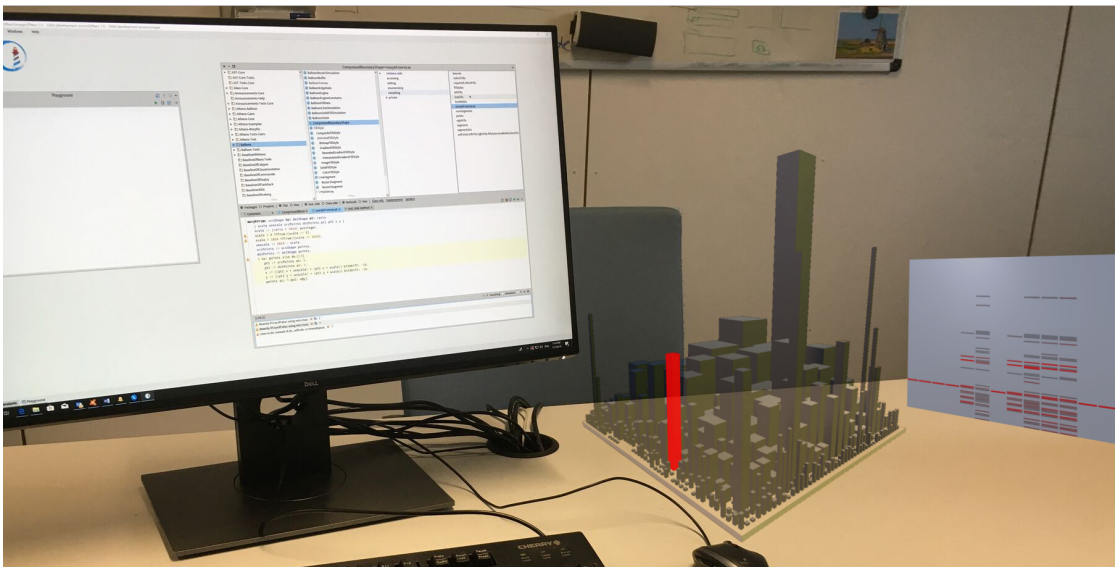

Figure 1.1: A pervasive visualization tool displayed in immersive augmented reality that uses a city metaphor and complementary scatter plot techniques.

displayed in immersive augmented reality (see Figure 1.1) can alleviate these issues, promote the use of performance visualization, and increase developer's awareness of the impact of code changes to software performance. Consequently, we formulate the following research question:

> *RQ: How can a visualization tool support developers in the analysis of the impact of source code changes to the performance of a system?*

To increase developer awareness of the impact of code changes in the performance of software systems, we present *PerfVis*, a pervasive visualization tool displaying in immersive augmented reality. PerfVis provides an omnipresent overview of the system's performance, taking advantage of visualization techniques to increase developer's awareness to the impact of code changes. We build on previous work [18], and focus on designing a visualization tool to support software performance analysis tasks. To unveil the benefits of our tool, we present the results of an initial evaluation that focuses on user performance and user experience. In the evaluation, we not only measure traditional variables of user performance such as completion time and correctness, but also include recollection. Furthermore, we also measure variables of user experience such as cognitive load and emotions.

1.1 Contributions

The main contributions of this thesis are: (i) PerfVis, a tool to visualize software performance in immersive augmented reality, (ii) a discussion on design choices, and (iii) a user study. We also contribute to the

reproducibility of our research by making the implementation of PerfVis publicly available[1].

1.2 Outline

The remainder of this thesis is structured as follows: In Chapter 2 we elaborate on the state-of-the-art of research in 3D software visualization and the use of immersive augmented and virtual reality. Chapter 3 introduces PerfVis, and describes its design and implementation details. Chapter 4 presents an empirical user study. We conclude the thesis in Chapter 5 and outline future work.

[1] http://scg.unibe.ch/research/perfvis

2

State of the Art

We identify three main categories of related work (i) 3D visualizations, (ii) visualizations displayed in immersive augmented and virtual reality, and (iii) visualization of software performance. In the following we elaborate on the main aspects of the related work, and discuss how they differ from our proposed approach.

2.1 3D Visualization

A number of 3D visualization tools have been proposed to support software development concerns. *SeeIt 3D* [26] uses a 3D scatter plot visualization to represent software metrics collected from Java source code to support developers in software comprehension tasks. A comparative evaluation between the proposed visualization tool with a popular IDE (*i.e.*, Eclipse) that included the analysis of data collected from eye tracking showed that participants who used SeeIT 3D achieved a better performance in overview tasks, but took longer time for bug-fix tasks. *SynchroVis* [31] presents a visualization based on the 3D city metaphor that displays the structure and metrics of software systems, as well as program traces and synchronization aspects (*e.g.*, semaphores/wait) to support the analysis of the behavior of concurrent software systems. Tymchuk *et al.* [29] propose a tool to identify quality anomalies in dynamically typed software systems by providing a 3D visualization that allows developers to analyze the co-evolution of source code and

quality metrics in a software system. The visualization uses the space-time cube technique to represent (i) software components, (ii) software versions, and (iii) broken quality rules. We observe that these tools display the 3D visualization on a standard computer screen and focus on performance data collected using static analysis. In contrast, we propose visualizations displayed in immersive augmented reality that focus on live performance data collected using dynamic analysis.

2.2 Visualization in Virtual and Augmented Reality

Some studies have focused on the perceptual and cognitive properties of visualizations displayed in immersive virtual and augmented reality. Elliott *et al.* [9] elaborate on movement and cognitive mechanisms for which virtual reality is of advantage. The study also discusses that spatial memory (boosted by virtual reality) benefits navigation. Also, the manipulation of objects that resemble real reality can improve perception and retention. A number of studies [5, 23, 30] have shown various potential benefits of using visualization in immersive augmented and virtual reality proposed in multiple domains other than software engineering. In contrast, we focus on proposing a visualization tool specifically to support software developers in the analysis of performance.

A few studies have focused on evaluating visualizations displayed in immersive augmented and virtual reality to support software comprehension tasks. Merino *et al.* [15] conjectures that usability issues (e.g navigation, occlusion, selection and text readability) of 3D visualizations are an effect of displaying them in a 2D medium (*e.g.*, standard computer screen). The study proposes that the use of immersive augmented reality can help to overcome these issues. The study elaborates on a user study that analyzes the prevalence of these usability issues by comparing a similar visualization displayed on a standard computer screen to one displayed in immersive augmented reality. An evaluation showed that immersive augmented reality facilitates navigation and reduces occlusion, however, selection and text readability remain open issues. Another study [17] hypothesizes that the medium used to display visualizations has an impact on its effectiveness. The study discusses the results of a user study that compares visualizations that use the city metaphor displayed using (i) a standard computer screen, (ii) a immersive virtual reality device, and (iii) a 3D printed model. The study found that participants using city visualizations displayed in immersive virtual reality achieved the highest recollection. As opposed to these studies, our research focuses on live visualization of a stream of software performance data.

Some studies have proposed visualization tools displayed in virtual reality to support software comprehension tasks. *Code Park* [11] proposes an immersive visualization of software metrics and source code. An evaluation showed that the proposed visualization displayed in virtual reality excels at usability and significantly helps in code understanding. *ExplorViz* [10] presents visualization of software landscapes to support software comprehension tasks. *CityVR* [18] proposes to gamify software implementation using a 3D city visualization displayed in virtual reality to boost developer engagement. The tool visualizes the static structure of software using the city metaphor. Early results of developers using the visualization show

benefits in engagement such as promoting curiosity, immersion, and excitement, and making developers willing to spend considerable time using the visualization. By contrast, we propose the use of immersive augmented reality to improve the awareness of the impact of changes to the source code to the performance of software systems. Moreover, these tools focus on various static software metrics, while our tool includes visualization of dynamic live performance data.

2.3 Visualization of Performance

The visualization of live performance is challenging due to the short time span that users have to analyze the data. In consequence, several performance visualizations have focused on the post-hoc analysis of performance data. *JOVE* [24] is a visualization tool for monitoring performance of Java programs. To obtain an omnipresent view of the performance of the system, the tool adds little overhead to the programming environment. Using the visualization, developers can obtain details on demand to identify routines and threads in which the application spends a long time. Another tool [4] includes visualization of the structure of parallel software systems. The visualization uses an execution graph to simplify the analysis of the complex run-time behavior. Moreta and Telea [21] include a visualization of the behavior of the memory allocator in C programs to optimize functionality, decrease fragmentation, and improve response time. A scatter-plot is used to show how the memory usage changes over time. Another tool [22] facilitates a visualization of software performance in real-time, using the city metaphor to represent the structure and performance of a program. The buildings in the city represent the classes in the system, and the height of the buildings represent the number of times the methods of a class are called during execution. All these tools support the analysis of various aspects of software performance through visualization displayed on a standard computer screen. In contrast, we rely on immersive augmented reality to display our visualization.

Some studies have proposed visualization to support the understanding of performance of large software landscapes. can be hard and crucial, since impeded performance might not be obvious. One study [8] introduces a pattern-based visualization approach to help understand the behavior of large and complex groups of web service applications. Another study [6] proposes a visualization tool, *Streamsight*, for parallel processing systems, to develop, understand and debug these application, which is hard due to their dynamic nature. Streamsight visualizes software systems as graphs, and includes views at different levels of detail. The visualization updates in real time, adding and removing new nodes and updating the flows through each node. Both tools focus on the performance of large software systems. While they might offer visualizations on the application level, the focus of PerfVis is only on the application level.

The data obtained from profiling some software can be overwhelming. Several other approaches have proposed visualization to support software performance analysis. Lin *et al.* [13] propose visualization to navigate and analyze profiling call trees. The study confirms that filtering and aggregation are key techniques to cope with the complexity of large call trees. They propose an approach to set the appropriate

abstraction level of the visualized data. Another study [7] introduces a novel visualization tool, *Zinsight*, to help understand and analyze operation system traces. Adamoli *et al.* [1] introduce *Trevis*, a tool that visualizes calling context trees produced by profilers. Trevis produces calling context ring charts to better scale to large data. Additionally, ring charts (*i.e.*, calling context trees) can be compared with intersection clustering and other methods. Another study [33] introduces *lviz*, a visualization tool to analyze operation system traces to cope with complex and large data sets. All these approaches focus on reducing the complexity of analyzing calling graphs produced by profilers through visualization that use various techniques displayed on a standard computer screen. In contrast, we opted to use well-know visualization techniques that displayed in immersive augmented reality can complement an integrated development environment of choice unobtrusively.

A few software visualizations adopt augmented reality as a display medium. One study [15] evaluated a city visualization displayed in immersive augmented reality to support software comprehension tasks. Another study [28] that used the city metaphor in augmented reality to support the analysis of software evolution. In contrast, we propose a city based visualization using immersive augmented reality to support software performance tasks, which is an innovative and not yet explored effort.

3

PerfVis Overview

PerfVis is a tool to visualize software performance through immersive augmented reality [20]. Using PerfVis developers can be aware of the workload of a software system without requiring their constant attention. We summarize the characteristics of the context of PerfVis using a taxonomy proposed by Maletic *et al.* [14] and Merino *et al.* [19].

PerfVis provides visualization in *immersive augmented reality* to support the *programmer* audience's efforts in system performance tasks. The visualization shows *static data* of the structure of the system and *dynamic data* of the live performance of the system.

PerfVis is an extension of CityVR by Merino*et al.* [18]. They implement the city metaphor to visualize a software system in immersive virtual reality to support developers in system comprehension tasks. We extend this tool with: (i) visualization of live dynamic performance data, (ii) the complementary scatter plot, (iii) and implementing interactions.

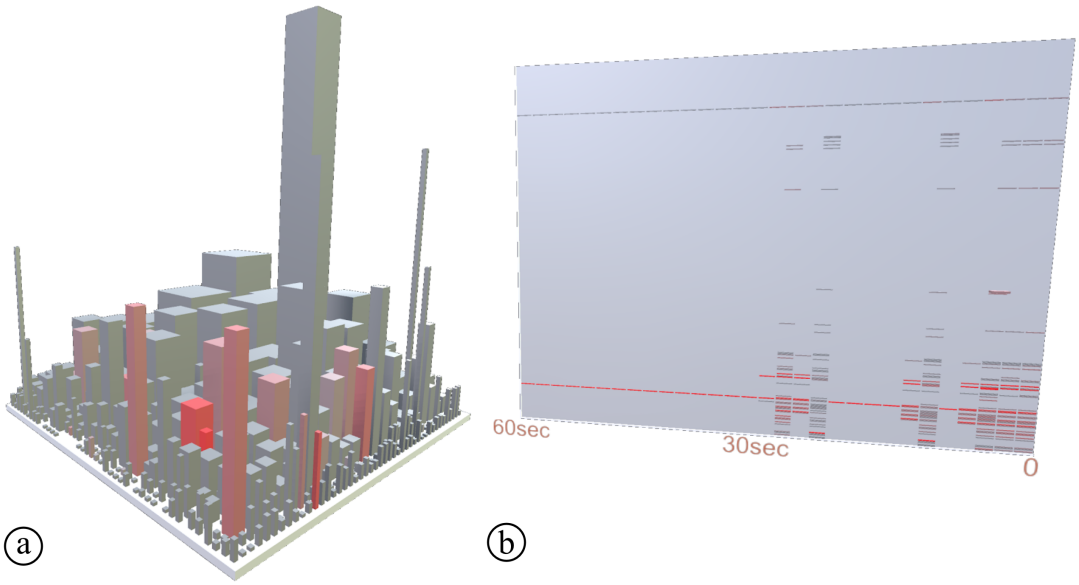

Figure 3.1: ⓐ A detailed view of the dynamic city visualization, and ⓑ the complementary scatter plot that keeps the history of the overall performance of the system for a configurable interval of time.

3.1 Design

3.1.1 Medium: Immersive Augmented Reality

Most visualizations that support performance analysis tasks are displayed on a computer screen [19]. They require developers to either use an application outside their development environment, or add a plug-in to their environment that sacrifices some valuable screen space. Amongst the few visualizations that implement a different medium most are displayed in immersive virtual reality [17]. These visualizations require that developers completely change their development environment, and isolate themselves wearing a headset. As opposed to these approaches, we argue that a medium that disrupts the development flow as little as possible can significantly increase the usability of visualizations, since research [16] suggests that the medium used to display a visualization can impact its effectiveness and efficiency . Immersive augmented reality can reduce the effort required to find information, ease navigation, and improve engagement of the users [5, 23, 30]. Consequently, we chose immersive augmented reality to display PerfVis, because (i) developers can use the visualization as a complement to their usual development environment (*i.e.*, developers are required to wear a headset, but not to modify the rest of their environment), and (ii) visualizations in immersive augmented reality are inherently displayed in 3D, which provides an extra dimension to encode performance data of a software system.

We chose the Microsoft HoloLens to display the visualizations in immersive augmented reality and support interaction with the user.

3.1.2 Technique: City visualization and Scatter plot.

In a previous study [19], we observed that most visualizations to support developers in monitoring performance used pixel-based techniques. These visualizations usually attempt to provide an overview of a significant amount of data to promote the detection of visual patterns that may signal anomalies in the performance of software systems. Instead, we adopt lightweight visualization techniques that involve less cognitive effort to make developers aware of the performance of a system. Our visualization includes two views: (i) a city visualization that provides a pervasive overview of the structure and performance of a software system, and (ii) a scatter plot that displays the history of the performance of the system in the past. We chose the city metaphor not only because it has shown to be effective in system comprehension tasks [32] but also because of our familiarity with navigating a city. Since the visualization is intended as an omnipresent companion to an IDE, we expect that the tool can offer opportune information without overwhelming the user with the data's complexity.

3.1.2.1 City visualization.

We opted to represent a metric of the performance of the software system using the color of buildings. The visualization is always on and indicates the system performance collected during a short time frame. In consequence, a glimpse at the city can make developers aware of the current overall status of the performance, and a detail of the classes that are involved. Figure 3.1 (a) shows a visualized software city. Each building represents a class in the software system. Buildings are grouped into districts according to their packages. Each building can be configured to represent software metrics using three properties: (i) height, (ii) footprint (*e.g.*, squared base), and (iii) color. We consequently configured the city in the following manner: each building's (i) height encodes the number of methods of the represented class, (ii) footprint (*i.e.*, squared base) encodes the number of attributes of the represented class, and (iii) color encodes the number of times that methods of the class are called during execution. A building turning red represents the number of times methods of the respective class have been called in the immediate past. The more intense the red, the higher is the number of method calls to that class.

3.1.2.2 Scatter plot.

The scatter plot helps developers analyze changes in the dynamic city visualization that might be overlooked by providing an overview of the immediate past performance. With the scatter plot a developer can: (i) compare performance at different times, (ii) search for outliers or (iii) inspect the behavior over time. Figure 3.1 (b) shows the scatter plot to the right of the city visualization. The X-axis encodes time, and the Y-axis encodes the classes in the system. Each mark represents a class that was involved in the execution of the system at a point in time. The color of a mark encodes the number of times methods of a class were called. The more intense the red, the more times methods of a class were called. Each row in the scatter

plot represents the evolution of the number of times the methods of a class are called along time. Each column is a snapshot of the performance of the whole system at a point in time.

3.1.3 Interaction: Selection and Navigation

Developers can interact with the visualizations through:

1. **Selection:** Through head movements users can point at and select elements in the visualization. When the pointer hovers over a building in the city visualization or a mark in the scatter plot, the name of the corresponding class is displayed. Users select buildings by performing an *airtap* gesture[1]. Once a building is selected that building and the marks that represent the same class are simultaneously highlighted with a yellow background. Similarly, when selecting a mark in the scatter plot, the building and marks that represent the same class are highlighted. Using this feature developers relate the information they get from one visualization to the elements in the complementary visualization. Highlighting elements is also useful to re-identify buildings after focus was disturbed.
2. **Navigation:** To gain an overview over the whole system, the user can move around and inspect the city from different angles. This way the user can compare metrics between different buildings.

3.2 Implementation

We used Unity in conjunction with the Microsoft Mixed Reality Toolkit[2] to develop PerfVis. Designing a tool for immersive augmented reality can pose a great challenge due to the novelty of the medium. The aforementioned combination of tools allowed us, with limited computer graphics experience, to start developing in immersive augmented reality. We used Unity to design holograms and program their behaviours, while the Mixed Reality Toolkit provides interfaces for gesture recognition and renders the virtual objects into immersive augmented reality. In Unity, entities are realized with *GameObjects*, containing transform-, material-, behaviour-components and also other GameObjects. This allows child objects to inherit properties from their parents, especially the position. In this section we elaborate on some components and particularities we encountered during development. See Figure 3.2 for an graphical overview.

Data Flow We animate PerfVis with the dynamic performance data of the monitored system to visualize the current number of method calls to all classes in the system. To attain this data, PerfVis periodically queries a web server that exposes a file generated by a profiler. The file (.csv) is a list of pairs of a class name and a value between 0 and 1 (*e.g.*, `MapBuilder`: 0.5). The file is then iterated through and the

[1] `https://docs.microsoft.com/en-us/hololens/`
[2] `https://github.com/Microsoft/MixedRealityToolkit-Unity`

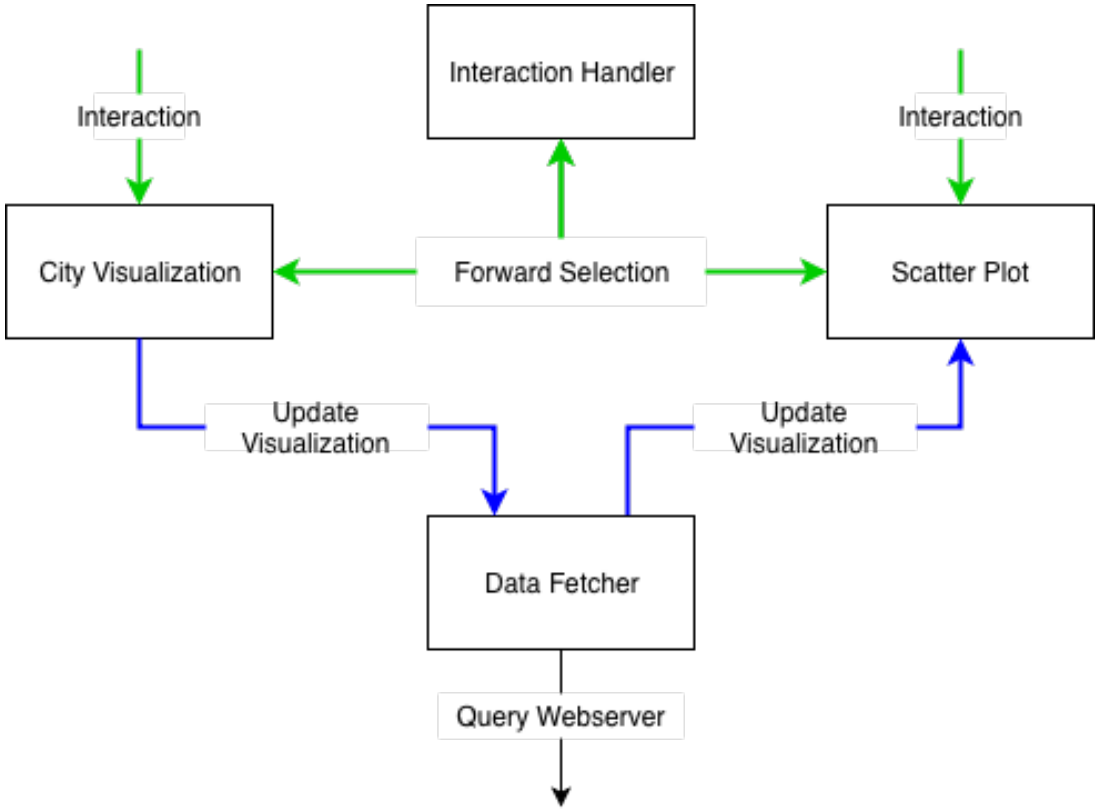

Figure 3.2: An overview over a few components of the implementation and their communication.

corresponding buildings are colored accordingly. Additionally, all received data is aggregated and saved for later display in the scatter plot.

City Visualization Each time the visualization is started, the city visualization is built from the .csv file created by CodeCity. Each line in the file contains information to create one building with: (i) the dimensions, (ii) position, (iii) and the scale. In the same step, each building is assigned the name of the corresponding class and the behaviour to change color. After all buildings are created, a dictionary is set up to find buildings in the city by their name. As the visualization is running, and buildings are colored red, each building has its own behaviour to turn back to its base color after a set amount of time has elapsed.

Scatter Plot The scatter plot contains an overview over the past performance of the whole system, a time vs class chart. The implementation posed a number of challenges during development, first of all, visualization of time. With the passing of time, marks in the scatter plot have to move backwards. Due to concerns about computational power of our immersive augmented reality device, we opted for discrete instead of continuous time steps (*i.e.*, marks periodically move backwards a set distance and are

still otherwise). Another question we encountered was about how to insert marks into the scatter plot. Marks have to: (i) be able to be inserted at the correct height, according to a class name, (ii) be moved backwards in the scatter plot, (iii) highlighted when the corresponding building is selected, Consequently we implemented the scatter plot in the following manner: Marks in the scatter plot are aggregated into an object by their position on the x-axis. With this approach, when elements in the scatter plot have to move backwards, we only have to move each x-object backwards one step, and thus moving all marks (through inheritance). When a row has to be highlighted, the tool simply traverses all x-groups and passes on which mark to highlight. To ascertain at which height a mark has to be inserted, in order to represent the correct class, we used a dictionary that maps class names to a float value representing the classes height in the scatter plot.

Consequently when the visualization starts, the background, the axes are created, an array is set up to contain the objects representing the discrete steps on the x-axis and the dictionary mapping class names to a height is created. During operation, the graph moves through these steps:

1. Move all objects in the x-axis backwards.
2. Delete the object that leaves the scatter plot, create a new empty object and place it at the beginning of the scatter plot.
3. Fetch the aggregated performance data, calculate the average for each class and then insert the corresponding marks.
4. Repeat from step 1.

Interaction Handlers All elements (*i.e.*, buildings in the city and marks in the scatter plot) can be interacted with either by hovering or selection. When the user hovers over an object, the cursor is placed on the hovered object and rotated to be aligned with the surface. When a building in the city visualization is hovered over, the corresponding class name is displayed in a fixed relation to the camera (top left of the users vision). When an element is selected (in the scatter plot or the city visualization), the element is highlighted with a orange glow, see figure 3.3, and the selection is forwarded to the respective other view.

3.3 Workflow

PerfVis uses the Moose analysis environment[3] to extract data of the structure and metrics of a software system. The static landscape of the city visualization is built using the CodeCity[4] implementation. The dynamic data of the performance of the system is obtained using the Spy2[5] profiler tool. The visualizations

[3] `http://moosetechnology.org`
[4] `http://smalltalkhub.com/#!/~RichardWettel/CodeCity`
[5] `https://github.com/ObjectProfile/Spy2`

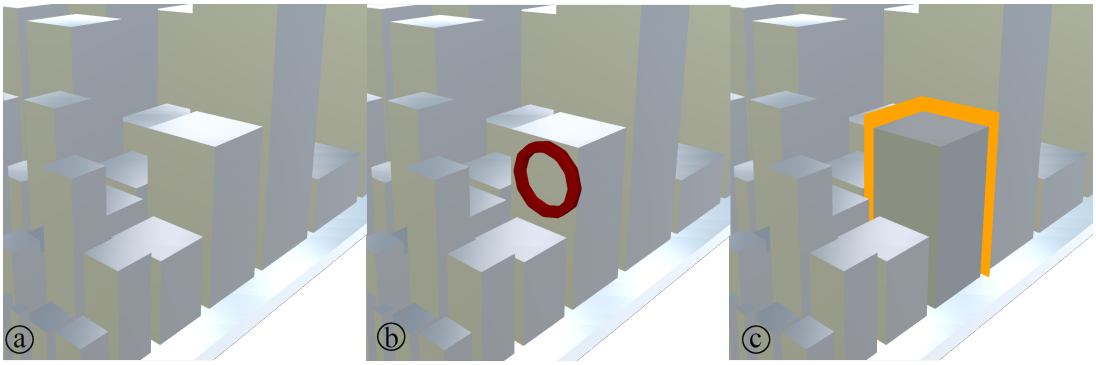

Figure 3.3: A building in the city visualization ⓐ unselected, ⓑ hovered and ⓒ selected and highlighted

are built using Unity[6]. Figure 3.4 shows 5-step workflow that developers have to follow when working with PerfVis, the steps are:

(1) **Create Model:** Using Moose, developers create a model of the software system they want to monitor. The model contains data of the structure and static metrics of the system.

(2) **Build the City:** Using CodeCity, developers create a city visualization of the modeled system. The properties of the visualization (*e.g.*, position and size of buildings) are exported to a file.

(3) **Profiling:** Using Spy2, our tool obtains a stream of metric performance data of the running system. The stream is made available online using a web server.

(4) **Setup Unity:** Using the "Holographic Remoting Player" for Unity, users connect Unity to their HoloLens through a wireless network [7]. Unity then has to be configured to use the model created in Step (2) as input for the visualization, and the IP address of the web server that exposes the performance data. Additional parameters can then be set, such as the size of the buffer of the scatter plot, and the rate at which the dynamic performance data is retrieved.

(5) **Immersive City Visualization:** Using a MS HoloLens device, the developer can use the live visualization to monitor the performance of the running software system.

After following the steps, the setup will look like in Figure 3.5. Three machines are involved: (i) Computer 1, running the software system to be monitored, the web server, and the profiler tool, (ii) Computer 2 running the visualization tool, and streaming the visualization to the HoloLens device, and a (iii) HoloLens,

[6] https://unity3d.com/de
[7] https://docs.microsoft.com/en-us/windows/mixed-reality/holographic-remoting-player

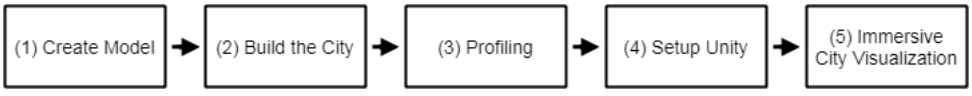

Figure 3.4: Workflow when working with PerfVis

rendering the visualization in immersive augmented reality as a companion tool for the IDE. A wireless network is used for communication. We argue that a visualization that disturbs the existing workflow as little as possible is more effective. This setup allows for minimal disturbance of the existing workflow, as only a profiler and web server have to be installed, while the remaining parts (*i.e.*, Computer 2 and the HoloLens) only have to be set up once. Connectivity between Computer 1 and Computer 2 is established by configuring Unity with the IP address of the web server, and between Computer 2 and the HoloLens through the "Holographic Remoting Player". Although all the components could co-exist in a single computer, we think this distributed architecture could be useful to extend the uses of our visualizations for remote monitoring.

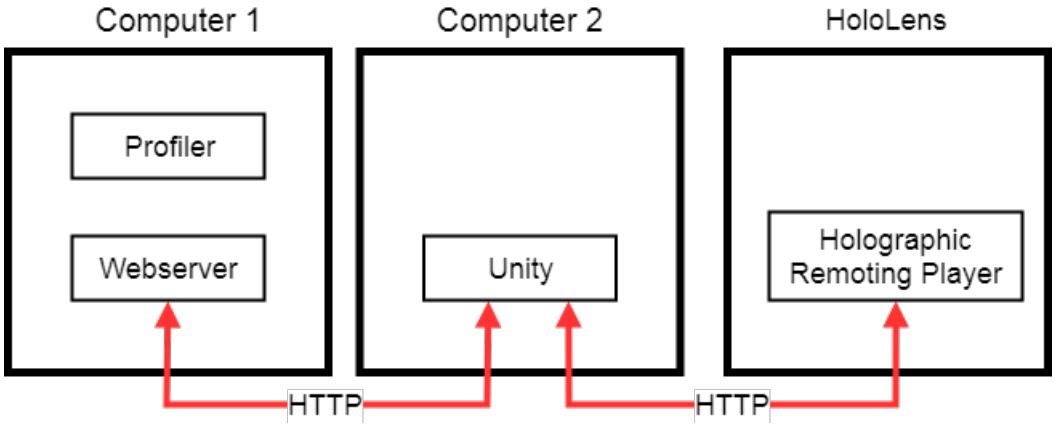

Figure 3.5: Setup required to work with PerfVis. Three machines are involved: *(i)* Computer 1, *(ii)* Computer 2 and the *(iii)* HoloLens.

More details are available in the PerfVis web page[8].

[8] http://scg.unibe.ch/research/perfvis

4 Evaluation

In this Chapter we present a user study we carried out to evaluate PerfVis. Firstly, we elaborate on the design of the study. Secondly, we present an analysis and discussion of the results. Finally, we discuss lessons learned during the development of the tool.

4.1 User Study

We conducted a user study to evaluate the effectiveness of PerfVis to support software performance tasks. We characterize the evaluation of PerfVis using a taxonomy proposed by Maletic *et al.* [14] and Merino *et al.* [19]. Firstly, we discuss our design choices, and explain our thought process. We then present the procedure that we followed and the data we collected.

The objective of the study is to assess the impact of using PerfVis on the user performance and user experience of developers in software performance tasks. Since PerfVis is a prototype tool that uses visualization in immersive augmented reality, comparing PerfVis to a state-of-the-art software performance tool (*e.g.*, AppDynamics[1], New Relic APM[2]) could produce misleading conclusions (since these tools are mostly based on text and they are displayed on the computer screen). We evaluate our tool via a user

[1] `https://www.appdynamics.com/`
[2] `https://newrelic.com/products/application-monitoring`

study in which the only independent variable is the medium. To this end, we compare PerfVis to a similar visualization tool displayed on the computer screen. Using the capabilities of the development environment used to implement our tool (*i.e.*, Unity), we deployed PerfVis on a standard computer screen as well as in immersive augmented reality.

4.1.1 Study Design

The results of previous studies [15, 17, 18] indicate that immersive augmented (and virtual) reality can increase the effectiveness of 3D visualizations. To evaluate the effectiveness of our tool, we include in the evaluation dependent variables of user performance (*i.e.*, completion time, correctness, and recollection) and user experience (*i.e.*, emotions, cognitive load) of participants who used our visualization tool in a laboratory setting.

We choose to use a within-subjects design. That is, all participants use our visualization tool both displayed in immersive augmented reality, and also displayed on a computer screen. This design allows participants to reflect and compare the two setups, and by using the think-aloud protocol allows us to obtain valuable insights. Based on the results of pilots to test various configurations, we adapted the design of the study. We especially took care that the difficulty of tasks and the complexity of scenarios allowed participants to complete the session in approximately one hour.

We split the session into two parts: one in which participants used our visualizations displayed in immersive augmented reality, and one in which the visualizations are displayed on a standard computer screen. In each part of the study participants followed the same protocol. First, participants had a training phase to understand the visualization techniques and to get comfortable with our tool deployed in a particular medium. Then we asked participants to complete a set of tasks using our proposed visualization tool. Finally, participants filled in a questionnaire to collect their impressions.

To analyze recollection we showed participants five pictures as shown in Figure 4.1. Each picture contains

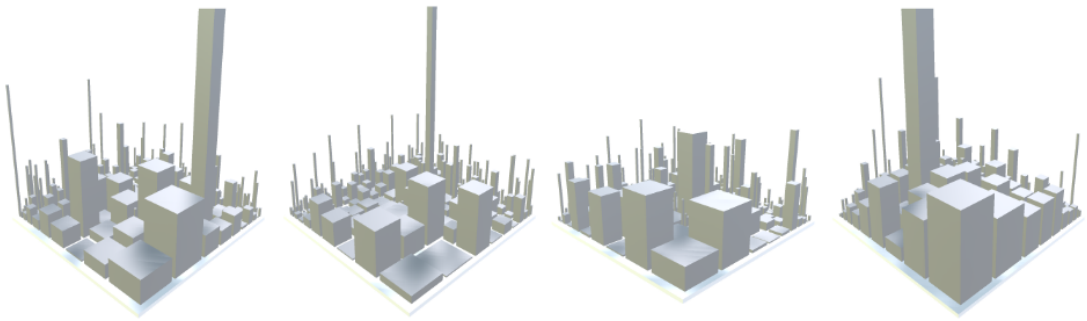

Figure 4.1: An exemplary figure we used to measure user recollection

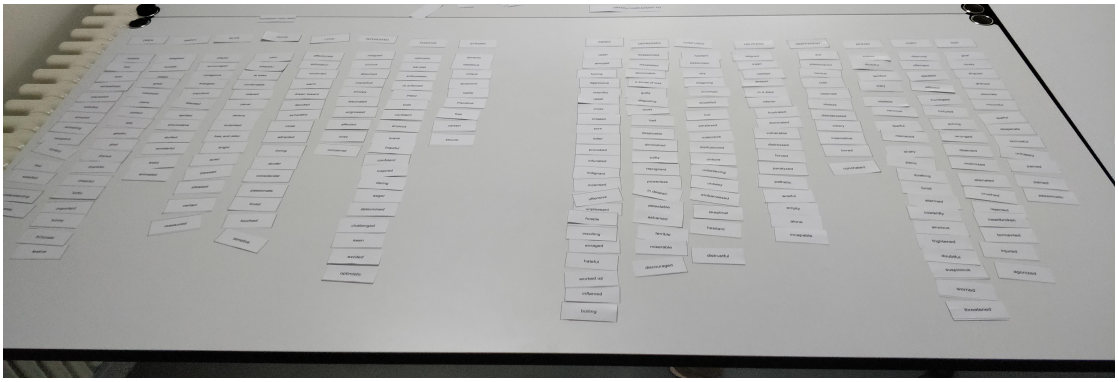

Figure 4.2: Example of the data set we used to collect experienced emotions.

a view of the city visualization that participants actually used during the session as well as three other views of similar city visualizations. On each picture participants were asked to identify the city they used to solve the tasks.

To measure user experience we reused a set of emotions introduced by Merino *et al.* [17]. In it, emotions are divided into pleasant and unpleasant and then further grouped into categories (*e.g.*, interested, alive, confused, afraid). Each category contains around fifteen emotions. After the session, participants were asked to select ten emotions they experienced from the set. Figure 4.2 shows how we arranged the emotion cards on a table, for participants to select from. This allowed us to then aggregate the experienced emotions by category.

We elaborate on the main characteristics of our study according to the extended benchmark properties proposed by [17] (*i.e.*, medium, technique, tasks, data set, and interactions).

Medium We design the study to use the following media: 1. Immersive Augmented Reality 2. Standard Computer Screen. In particular, we choose the following apparatus to conduct the study:

1. **Microsoft HoloLens**: We choose Microsoft HoloLens to render our visualization in immersive augmented reality. The HoloLens is an untethered head-mounted stereoscopic display, featuring two stereoscopic displays with 1268×720 pixel resolution, 60 Hz content refresh rate, and a 30° H and 17.5° V field of view. Interaction is possible through head tracking, gesture input, and voice support.
2. **Alienware Laptop**: We choose an Alienware laptop to render our visualizations on a standard computer screen. The laptop features a four core 3.8 GHz CPU, a Nvidia GTX 1060 GPU, a 13.3 inch 2560×1440 display, and 16GB of memory.

Technique The PerfVis tool includes two visualization techniques:

1. **City Metaphor**: In a software city buildings represent the classes of a software system. Buildings

are grouped into districts that represent the packages of the system. Each building can encode three metrics of the classes of the software system. The city metaphor can visualize data in three dimensions: (i) the height, (ii) the base, and (iii) the color of a building, Consequently, we configured our city visualization so buildings (i) height represents the number of methods in classes, (ii) square base size represents the number of attributes in classes, and (iii) color represents the number of method calls to a class in a interval of time.

2. **Scatter Plot**: The X-axis represents time and the Y-axis represents the classes in the system. A mark in the scatter plot represents that methods of a class have been called at some point in the execution of the program. The color of the mark represents the number of method calls to the class in a interval of time. Following a row in the scatter plot reveals the evolution of the number of method calls to a class over time. Each column is a snapshot of the performance of the whole system at a point in time.

Tasks To formulate appropriate tasks we draw inspiration from questions that involve performance concerns that arise during software development. Table 4.1 presents the tasks we included in the study as well as our rationale to include them. Firstly, we included five tasks (*i.e.*, *T1, T2, T3, T4* and *T5*) to observe participants as they obtain an overview over the city visualization. Tasks *T6* and *T7* require participants to identify outliers in the city visualization and the scatter plot. Tasks *T8* and *T9* require participants to analyze multiple metrics of static and dynamic aspects of the system using the city visualization (*i.e.*, participants have to consider height and color of buildings at the same time). Finally, we wanted to observe participants as they navigated the visualizations to get the most useful view of the city visualization to concentrate on multiple buildings at the same time with task *T10*.

Data Set Since multiple changes in the source code can impact the performance of system, to reproduce a data set that captures realistic anomalies in the performance of a software system is a challenging task. To conduct a user study that allows us to evaluate the benefits of our visualization approach, we define a criterion to which the data set has to comply. In consequence, two data sets are used to conduct our study: (i) a file of static metric data of a software system that are visualized as a city, and (ii) a stream of dynamic performance data that are visualized in the colors of the city and the scatter plot. To this end, we selected the *Roassal*[3] system for Pharo Smalltalk as the subject system to be visualized with PerfVis. To mitigate a learning effect due the use of a within-subjects approach, we require two different city visualizations to be used in each part of the sessions. To achieve this, we build a synthetic data set by preserving the layout of buildings and arbitrarily exchanging the heights of pairs of buildings in the city. As a result we obtain two similar city visualizations with the same layout, a similar distribution of heights, but different landscapes.

When conducting pilot observational studies we observed that simply stressing the application to collect performance data, and thus animating the city, presents several issues. Reproducibility is not guaranteed (a software system might not work in a deterministic way). Furthermore, it is difficult to stress an

[3] `http://agilevisualization.com/`

Table 4.1: Tasks included in our user study

ID	Task	Rationale
T1	Identify three classes that were involved in the execution of the system	To gain an initial understanding of changes in the performance of a software system, we assess whether our visualizations boost the abilities of developers to obtain an *overview*.
T2	Identify the 3 classes that were active for the most amount of time during the execution	
T3	Identify the 3 classes that were active most intensively during the execution (*hint*: most intense red color)	
T4	Identify 3 classes that were executed sequentially. (*hint*: buildings that turn red sequentially)	
T5	Identify 3 classes that were active in parallel	
T6	Identify the class with the highest number of executed methods	Once developers obtain an overview, they usually need to identify *outliers* that can be symptoms of abnormal behavior of performance.
T7	Point to the time during the execution at which the highest amount of classes were active	
T8	Estimate the percentage of the system that was executed at least once in terms of volume of buildings	To identify an abnormal behavior, we evaluated whether the visualizations helped developers to obtain quick estimations of performance metrics.
T9	Estimate the percentage of the system that was executed at least once in terms of number of buildings	
T10	Rate the activity level of classes X,Y,Z during the execution (0 never used - 5 always active) *Note:* X,Y and Z were the classes LabelGenerator, MapBuilder and AxisAdaptedBuilderExample in immersive augmented reality, and BrowserClosing, BrowserClosed and Examples in the standard computer screen deployment.	To gain insights into the usability of the visualizations as a complement to developer's environment, we evaluated whether our visualizations provide suitable navigation and interactions.

application in a consistent way manually. Consequently, we had find a way to feed performance data into the visualization that delivers consistent results and had to come up with a way to generate performance data.

We developed a small tool to generate a synthetic stream of performance data. We stored performance data as files (using a comma separated values format). Each line in the file contains a name of a class followed by a float number between 0 and 1 (*e.g.*, LabelGenerator: 0.7). When the visualization reads the data of each line then the corresponding building is colored according to the value (*i.e.*, 0, the building stays gray, 1 the building turns red). Sequentially reading the files periodically allows the tool to animate the city visualization in a way that resembles authentic performance data.

The method we use to generate performance data to be used in the study has to meet the following criteria:

1. **Time**: We have to be able to set the duration so that the dynamic performance data animates the city for six to ten seconds.
2. **Complexity**: The approach has to generate dynamic performance data with normally distributed values (*i.e.*, different classes have to be animated in a variety of shades of red).
3. **Flexibility**: The approach has to be flexible enough to allow us to tailor data sets for a particular task. For example for *T3* three classes have to be used in a sequential order.

We first collected performance data from the *Spy2* [3] profiler tool. However, data generated this way did not stress the capabilities of our visualization tool (animated only a few buildings for most of the execution). In consequence, we decided to generate the performance data randomly. We ensured that our approach allowed us to: (i) set the length of a data set stream (meeting criterion one), (ii) distribute animated classes and shades of red normally, and (iii) build a tailored stream of performance data that fulfill the needs of specific tasks, *e.g.*, task *T3* requires three buildings to be active sequentially.

Interactions We aimed at providing interactions in the visualizations displayed in both media (*i.e.*, immersive augmented reality, computer screen) that were as similar as possible. To produce similar interactions in both deployments, we implemented:

1. **Selection**: In PerfVis users hover over elements using head movements, and select elements with airtap gestures. In the visualizations displayed on the computer screen, selection is carried out using the mouse to hover over elements, and click to select them.
2. **Navigation**: In PerfVis users freely navigate the visualization using their body (*e.g.*, approaching, standing up, walking). In the visualizations displayed on the computer screen users can rotate the city visualization by clicking and dragging using the mouse. The position of the city visualization and the scatter plot are otherwise fixed (in both media).

4.1.2 Quality Focus

The quality focus of our study is twofold: user performance and user experience. We observe that the standard computer screen is the display medium used in most software visualizations. Also developers are used to interacting with applications using a mouse and keyboard. Due to these facts we expect that user performance could be higher for participants who use visualizations displayed on the computer screen. Previous studies found that immersive augmented and virtual reality can boost user experience. Therefore, we conjecture that user experience could be greater when using visualizations displayed in immersive augmented reality.

4.1.3 Participants

A total of 10 developers participated in the study, of whom 4 were PhD students and 6 were Bachelor students in Computer Science. Participants were not paid, but they were invited and opted to participate freely. The average age was 25.9 years with a standard deviation of 3.28. The average self reported experience as software developer was 3.9 years (participants had up to 10 years of experience). We also collected data of the previous experience of participants using immersive augmented reality devices and using 3D visualizations in general. To this end, we used a 5-step Likert scale. We observe that most participants have limited experience using augmented reality (rated 1.5 average), and using 3D visualizations (rated 1.9 average). We mitigated this lack of experience by including a training phase, in which participants could develop familiarity with such technologies. Due to incomplete information we had to remove one participant from recollection analysis, and three participants from the cognitive load analysis.

4.1.4 Procedure

We conducted the study in locations in two cities of Switzerland: Biel and Bern. The rooms in both locations had a similar size and lighting. When participants arrived, they were asked to fill out a form of consent to allow us to collect data during the session (*e.g.*, video recordings). The session consisted of a sequence of tasks that participants had to complete with the visualizations displayed in both media. When working in immersive augmented reality, participants wore an immersive augmented reality device (*i.e.*, Microsoft HoloLens). When working with the standard computer screen, participants were seated at a table, and interacted using a mouse and keyboard. A printout of the legend of the encoding of the visualizations was available on the desk during the whole sessions.

Firstly, we read an introduction to the context of the study to the participants. In the introduction, we explained to the participants the structure of the study, the visualizations to be used during study (*e.g.*, city metaphor), the available interactions in the visualizations, and the tasks that they had to complete. Secondly, participants had a training phase to learn the encoding and the interactions available in the

visualizations, and to understand the type of tasks they had to complete using the visualizations. When participants felt comfortable using the visualizations and did not have more questions, they continued with the following phase. Third, participants completed the phase to collect performance data, in which we asked them to complete 10 tasks. For each task, we read out loud a task, then loaded a data set tailored to the task, and then made annotations of the answers of participants. Fourth, participants were presented with a table in which we placed 250 emotion cards. To facilitate search, we sorted the cards into pleasant and unpleasant emotions (and placed them in different sides of the table), and grouped the cards into eight categories of positive emotions (*i.e.*, *open*, *happy*, *alive*, *good*, *love*, *interested*, *positive*, and *strong*, and eight of negative emotions (*i.e.*, *angry*, *depressed*, *confused*, *helpless*, *indifferent*, *afraid*, and *hurt*, and *sad*) (see Figure 4.2). The set of emotions used was previously introduced by Merino *et al.* [17]. Participants were asked to select 10 cards that identify the emotions they experienced when using the visualizations displayed in each medium. Fifth, participants were asked to fill out the NASA TLX questionnaire [4] to measure cognitive load. Finally, we measured recollection. We prepared five sets of four different city visualizations (see Figure 4.1). Participants were shown each set in a sequence, and for each asked to identify the city visualization that they used previously to complete the set of tasks.

4.1.5 Data Collection

To collect relevant data during the study that allowed us to explore our conjectures, we

(i) video recorded participants as they interacted with the visualizations and answered questions,

(ii) video recorded the view of visualizations as seen by participants,

(iii) kept records of the list of emotion cards that participants introspectively selected to identify their emotions when using visualizations, and

(iv) kept records of the cognitive load index from the NASA TLX questionnaire.

4.2 Results

We collected user performance (*i.e.*, correctness, completion time and recollection) and user experience (*i.e.*, emotions and cognitive load) data. In the following we present an analysis and discussion of the results.

4.2.1 User Performance

An effective use of visualizations implies achieving the desired goal. An efficient use implies achieving the goal (*i.e.*, accurately support the analysis of software performance data) with the least resources (*i.e.*,

[4] `https://en.wikipedia.org/wiki/NASA-TLX`

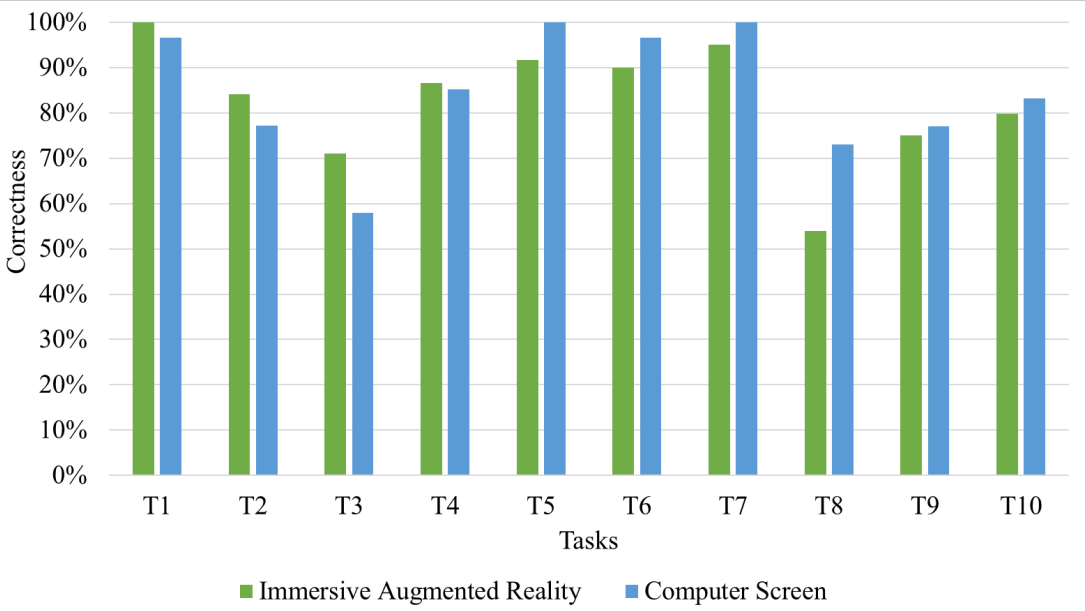

Figure 4.3: Average correctness of all participants in each task.

completion time). We now present and discuss the results related to user performance.

4.2.1.1 Correctness

To measure correctness, we rated the answers of participants using a percentage scale (*i.e.*, 100% being completely correct, and 0% being completely incorrect). See Table 4.1 for a list of all tasks. To rate tasks in which we asked participants to identify one or three classes (*i.e.*, *T1*, *T2*, *T3*, *T4*, *T5*, and *T6*) we calculated the percentages in the following way:

1. In most tasks (*i.e.*, *T1*, *T2*, *T3*, *T4*, and *T5*) we calculated the percentage by interpolating the values using the most incorrect and the most correct answer.
2. In task *T6* we observe that defining the most incorrect answer could be misleading. Comparing to the shortest building would not represent a realistic answer, and it would hide the differences between the answers of participants. To this end, we chose to interpolate a value by comparing the top ten closest answers to the correct one.
3. When participants had to answer numerical or percentage values (*i.e.*, *T7, T8, T9* and *T10*) we also calculated the percentage by interpolating between the most correct and the most incorrect answer.

Figure 4.3 shows the average correctness of all participants for the tasks. We observe that correctness is fairly similar in all tasks, with the exception of tasks *T8*, *T9*, and *T10* that present lower levels of correctness. Participants who used immersive augmented reality reached a slightly higher correctness in

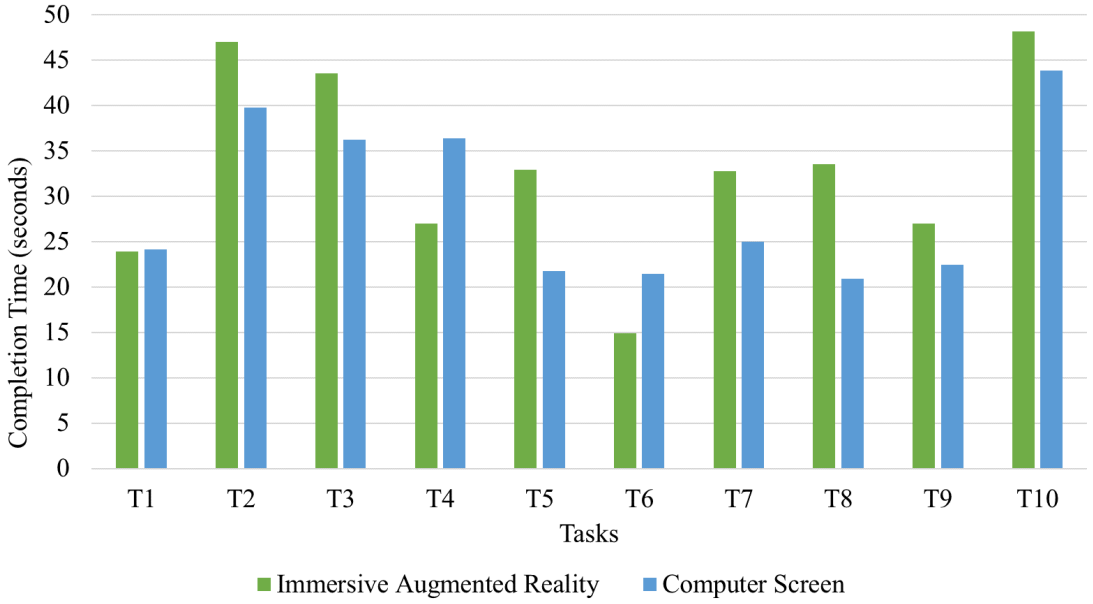

Figure 4.4: Average completion time of all participants in each task

tasks *T1*, *T2*, *T3*, and *T4*. These tasks were included to observe participants as they get an overview. A reason for the higher correctness in these tasks could be that participants in immersive augmented reality were able to navigate the visualization using natural body movements, which facilitated their navigation to obtain an overview. Additionally, immersive augmented reality allows participants to get a better overview by providing a much larger display area.

Correctness in tasks *T5* through *T10* turned out as initially expected, as participants using the standard computer screen achieved higher correctness. The difference in correctness is on average 7% (tasks *T8* presents a difference of 19%). We asked participants to estimate the percentage of the classes of the system that were called at least once during the execution weighted by the number of buildings in the city visualization (*T9*), and weighted by the volume of buildings (*T8*). We observe that participants achieved higher correctness in task *T9* than in *T8*, which could be due to the added complexity of weighing buildings by their volume.

4.2.1.2 Completion Time

We define completion time as the interval beginning when the dynamic visualization starts and ending with the definitive answer of participants. We extracted completion time from videos recorded during the study. Since all participants have significantly more experience using a standard computer screen than an immersive augmented reality device, we expected participants to achieve faster completion time using a computer screen.

Figure 4.4 shows average completion time for all tasks. Participants completed most tasks in less time when using visualizations displayed on a computer screen than when visualizations were displayed in immersive augmented reality. Interestingly, tasks *T4* and *T6* were completed much faster in immersive augmented reality. Task *T6* asked participants to identify the class with the highest number of classes (*i.e.*, the highest building) that was used. In immersive augmented reality, participants can position themselves to view the visualization from a side (*i.e.*, a skyline view). Possibly this made it easier to identify the highest building in the city visualization.

Participants in immersive augmented reality completed tasks on average 4 seconds slower. Completion time in task *T8* shows the highest difference (on average participants required 13 seconds more to complete the task when using the visualizations in immersive augmented reality). We reflect this could be due the previous experience of participants using a computer screen (and interacting with a keyboard and a mouse).

4.2.1.3 Recollection

We chose to measure recollection as another aspect of user performance. To measure recollection we used five pictures that contain four similar city visualizations (as shown in Figure 4.1). Participants were shown the pictures in a sequence and asked to identify in each of the pictures the city visualization they used to complete the tasks.

We observe that most participants report little experience using augmented reality devices (1.5 out of 5 in average). Therefore, we conjecture that they could be willing to spend some time using the visualizations, and so to increase their recollection. Consequently, we expect recollection in immersive augmented reality to be comparable to or higher than with a computer screen.

Figure 4.5 shows recollection scores of participants using immersive augmented reality and Figure 4.6 the computer screen. Participants achieved a fairly similar recollection in both deployments. After using visualizations in immersive augmented reality only six recollection questions were answered incorrectly (one more incorrect answer than participants who visualized using a computer screen).

4.2.2 User Experience

The rise of unpleasant emotions when using a visualization tool can make developers reluctant to adopt it for daily usage. Therefore, we opted to include user experience (in the dependent variables) to evaluate the effectiveness of PerfVis.

A previous study [15] suggests that displaying visualizations in immersive augmented reality can help to alleviate usability issues (*e.g.*, navigation, occlusion). Another study [18] suggests that immersive augmented reality can boost user experience. Hence, we expect user experience could be higher when using visualizations in immersive augmented reality than when they are displayed on a computer screen.

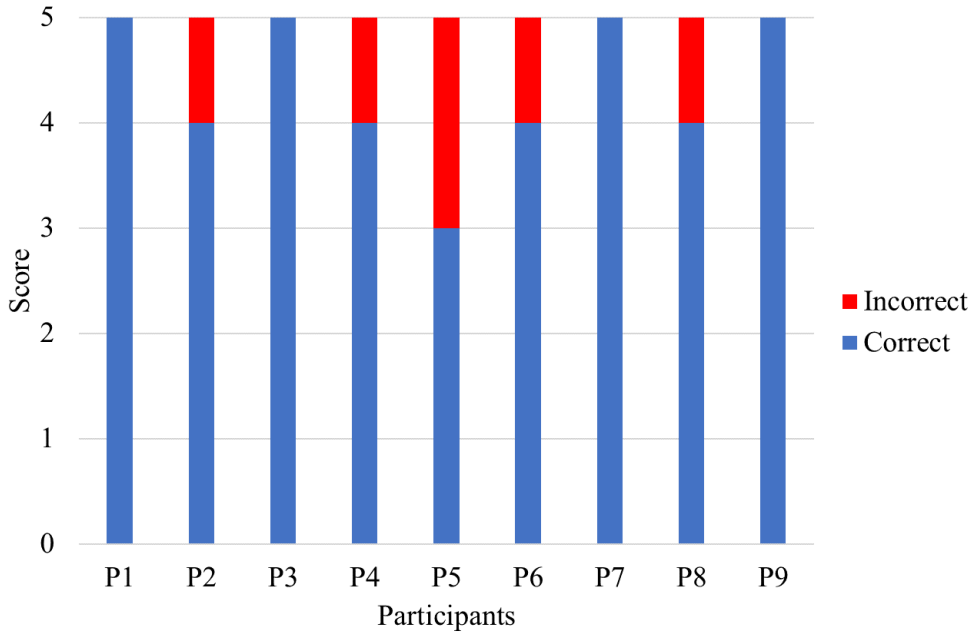

Figure 4.5: Scores for each participant in Recollection for the HoloLens

4.2.2.1 Emotions

The emotions felt while working with a software visualization tool are important to measure the overall user experience. However, for developers to identify the emotions they felt when using visualizations can be challenging.

To collect emotions we utilize a set of emotions introduced by Merino *et al.* [17]. In it, emotions are divided into pleasant and unpleasant emotions, and then further grouped into categories (*e.g.*, interested, alive, confused or afraid). Each category contains fifteen emotions on average. After each session, participants were asked to select ten emotions they experienced from the emotion cards. This facilitated developers to identify their emotions, and also allowed us to then aggregate the experienced emotions by category.

Since immersive augmented reality can reduce usability issues of 3D visualizations, we expect participants of the study to experience more pleasant emotions when using the visualizations displayed in immersive augmented reality. Additionally, the novelty effect of using an augmented reality device, can make a positive impact on the experienced emotions when using visualizations in immersive augmented reality.

HoloLens Figure 4.7 shows collected emotions (aggregated into categories) from participants using immersive augmented reality. Although there are more pleasant emotions experienced, participants experienced a considerable number of unpleasant emotions (53 pleasant, 40 unpleasant). Most unpleasant emotions (31 out of 40) belong to the categories *confused*, *angry*, and *helpless*. This could be due to

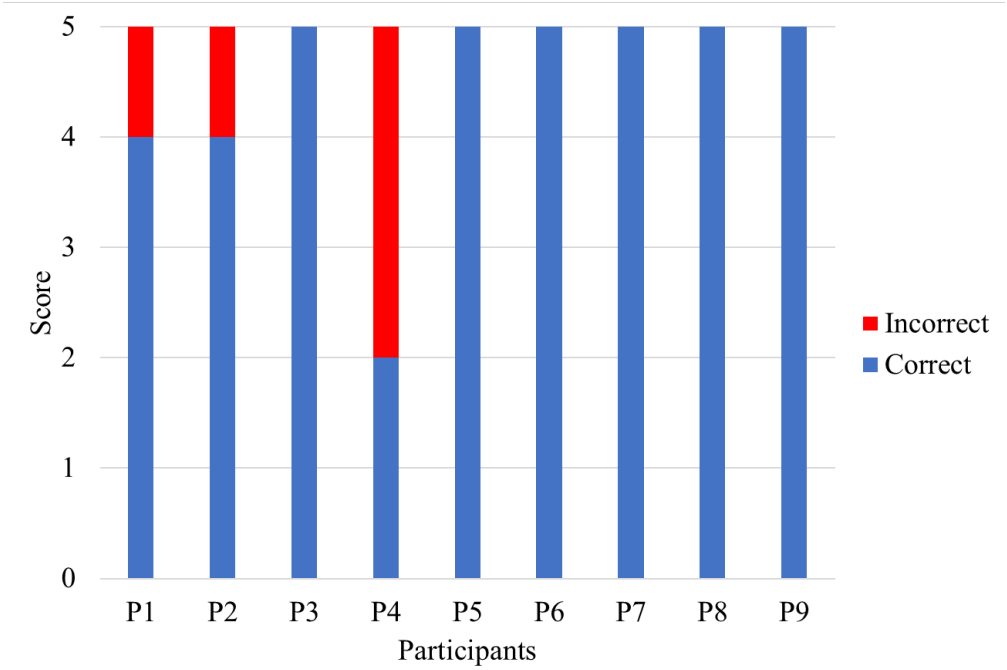

Figure 4.6: Scores for each participant in Recollection for the Screen

difficulties in interacting with the visualizations. We observed that some participants struggled to perform a correct airtap gesture, and to select an element in the scatter plot (due to its small size). We consider this a fault in our design choices when developing PerfVis and plan to address these issues in future work. Four pleasant categories of emotions (*interested*, *open*, *alive* and *positive*) were frequently experienced (40 out of 53 selected emotions), and no outlier can be identified. The most commonly experienced pleasant category *interested*) could be due to the novelty effect of using immersive augmented reality. The second most commonly experienced category (*open*) can be related to the feeling of freedom that immersive augmented reality frequently offers (since visualizations are not limited to the size of a screen). Three participants experienced significantly more pleasant than unpleasant emotions (7 pleasant, 3 unpleasant). On the other hand, only one participant experienced just 3 pleasant and 7 unpleasant emotions. Several positive and negative emotions were common amongst participants: *interested* ($x6$), *playful* ($x5$), *intrigued*, *free* and *curious* ($x3$ each), *frustrated* and *irritated* ($x4$ each), and *annoyed* ($x3$).

Standard computer screen Figure 4.8 shows the collected emotions of participants using a standard computer screen (that we aggregated into categories). When working with the computer screen, participants experienced many more pleasant (49) than unpleasant (32) emotions. Most positive emotions (16 out of 49) belong to the *open* category, which could be due to participant's familiarity with the medium, giving them *confidence* (experienced 5 times) to accomplish the tasks. Although no clear outlier can be identified from the unpleasant categories, most emotions commonly experienced belonged to the categories *confused*

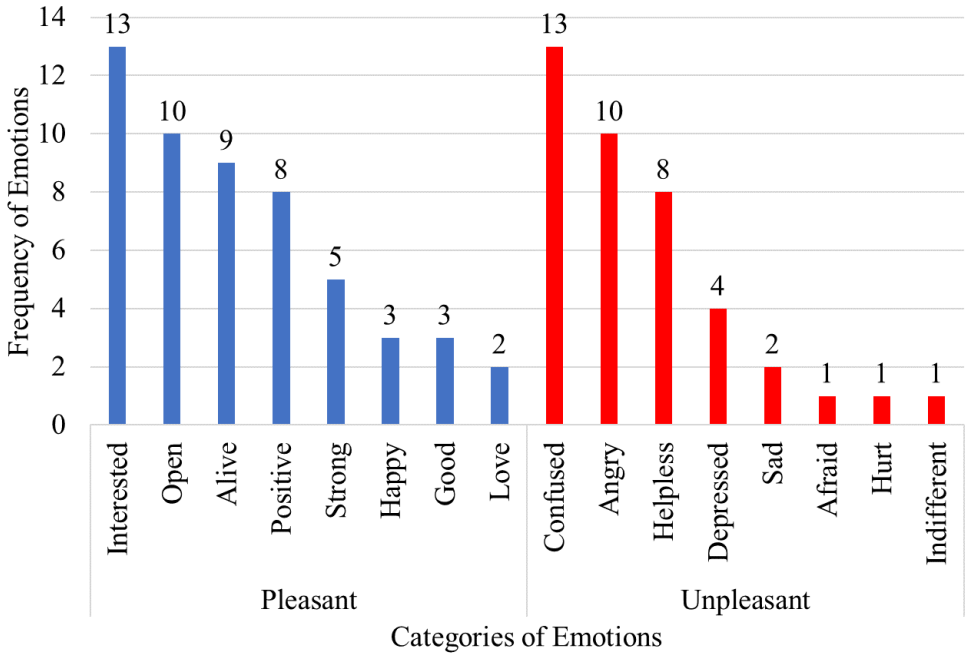

Figure 4.7: Collected emotions per category experienced when working in immersive augmented reality

and *angry*. We think this is due to participants being frustrated with the tool, since they expected higher quality or more features. Common emotions amongst the participants were: *confident* ($\times$5), *interested* ($\times$4), *challenged* ($\times$3), *irritated* and *uncertain* ($\times$3 each).

Comparison We expected participants to have a better experience when working in immersive augmented reality. Comparing emotions gathered in both implementations does not entirely confirm this expectation. Although most pleasant emotions were experienced in immersive augmented reality (53) the difference to those experienced with the computer screen (49) is not significant. Additionally, fewer unpleasant emotions were experienced with the computer screen (32) than in immersive augmented reality (40). In immersive augmented reality, participants felt *interested* and *open*, but also *confused* and *angry*. When using visualizations on a computer screen participants felt *open* and *positive*, but also *confused* and *angry*. In conclusion, we do not identify clear distinctions between the emotions experienced in immersive augmented reality and a computer screen.

4.2.2.2 Cognitive Load

We argue that the cognitive load has an impact on the user experience. Working for a long time with a tool that involves high cognitive load can be tiring, and thus decrease the user experience. We gather data of the cognitive load of participants of the study by using the NASA TLX questionnaire, a widely used

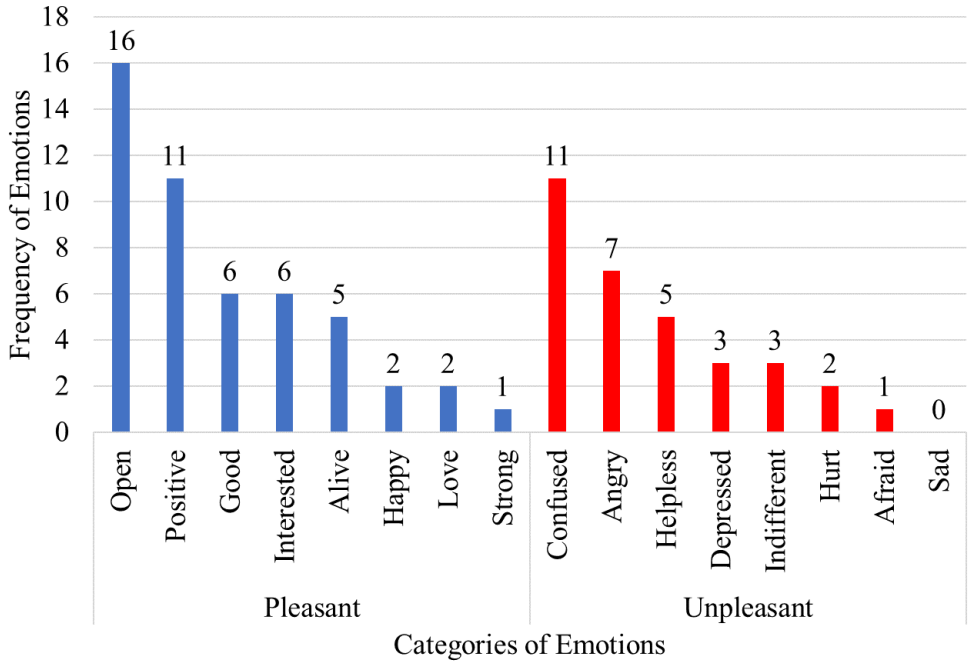

Figure 4.8: Collected emotions per category experienced when working with a computer screen

assessment tool. Since all participants have little experience using and interacting in immersive augmented reality, we expect the cognitive load could be higher in that medium.

Figure 4.9 and Figure 4.10 contain answers of seven participants to the TLX questionnaire for both deployments. From the figures it is apparent that participants experienced various levels of mental demand. Only two participants (*P2* and *P5*) experienced higher physical demand when working in immersive augmented reality, where body movement is used to navigate the visualizations, while usually participants were sitting to work with the computer screen. During the session, most participants in immersive augmented reality mostly sat in a chair and approached the visualizations to navigate them. Participants using the computer screen tended to perceive themselves to be more successful in accomplishing tasks, which could be explained by their familiarity with the computer screen setup. *P3*, *P5*, *P6* and *P7* show a similar trend in questions four and five. Each perceived themselves to be similarly successful in accomplishing tasks (question four) in both deployments, and perceived a similar effort required to achieve that level of performance (question five). *P2*, on the other hand, was more successful when using the computer screen than in immersive augmented reality, even though the participant perceived the same amount of work was necessary to reach the same level of performance. *P4* expressed a similar perception. Frustration was experienced by participants using both deployments in various levels. Interestingly, a trend can be identified, in that participants who perceived high frustration in one deployment, perceived less frustration with the other. In augmented reality *P3* felt average mental demand but only slight physical demand, while feeling very successful in accomplishing tasks and reporting very low frustration. *P3*

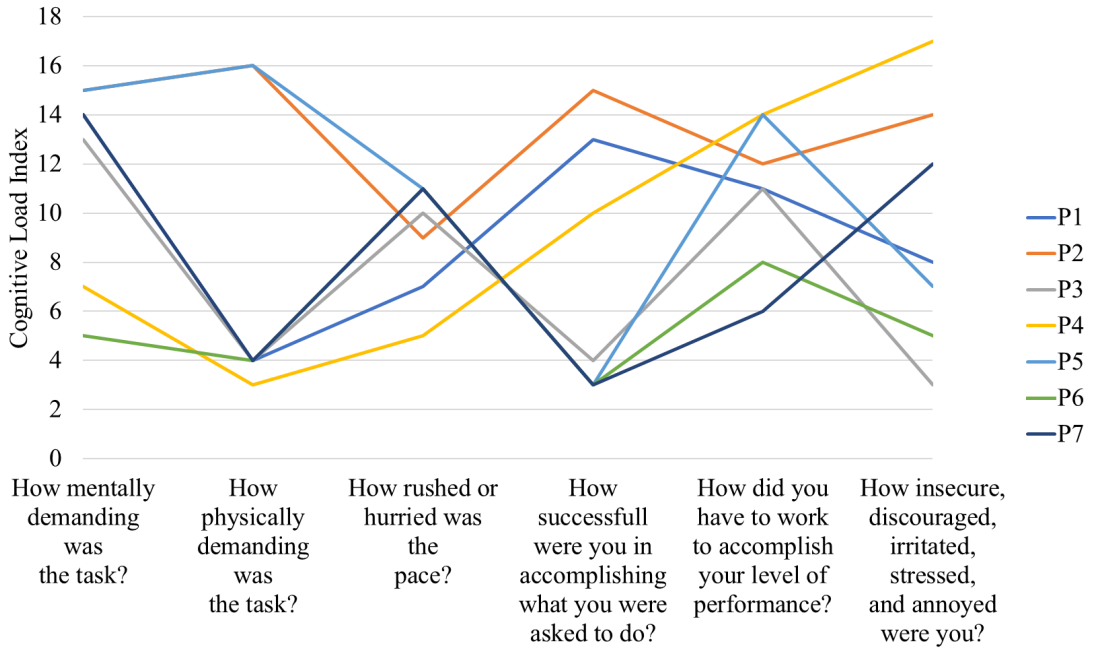

Figure 4.9: Answers to the NASA TLX questionnaire after working in immersive augmented reality

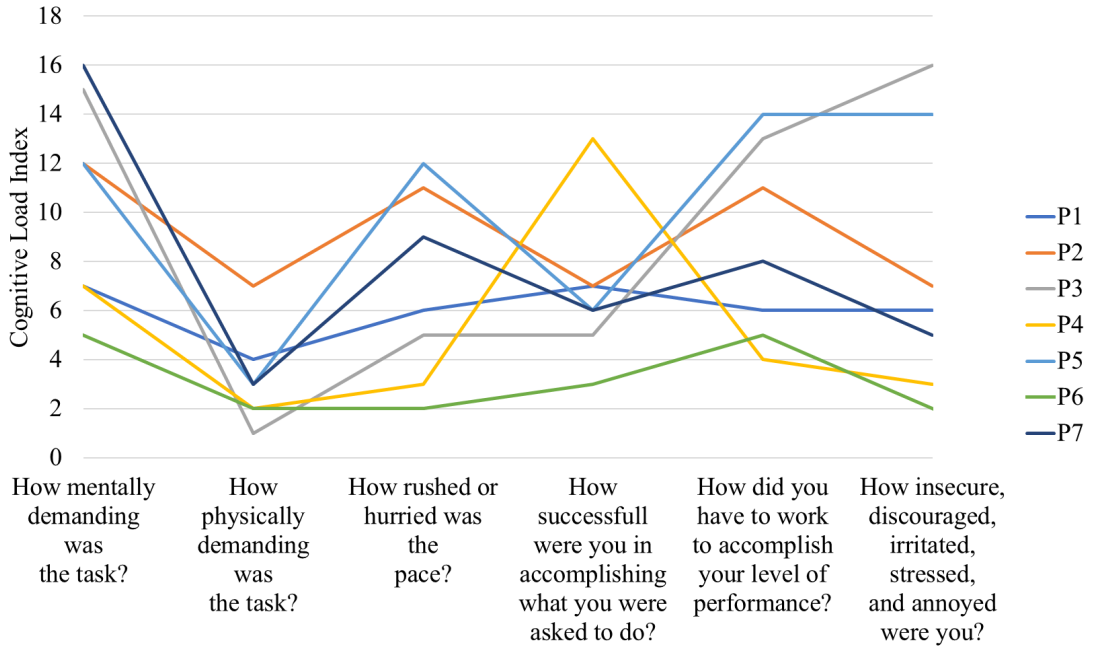

Figure 4.10: Answers to the NASA TLX questionnaire after working with a computer screen

also reported many positive emotions (7) and only few negative emotions (3). With a computer screen, *P3* reported similar mental, physical and temporal demand but felt less successful and more frustrated. Consequently *P3* had a worse experience with the computer screen, experiencing many more negative emotions(7 negative, 3 positive emotions), probably due to frustration with the medium. *P4* on the other hand, answered similarly in mental demand, physical demand, temporal demand and success but very dissimilarly in all other questions. In immersive augmented reality *P4* felt that he had to work very hard to accomplish his level of success and high frustration, thus reported only 4 positive emotions and 6 negative emotions. With a computer screen, *P4* hard to work less and felt little to no frustration, thus having a much better experience (9 positive, 1 negative emotion). *P4* was very frstrated when working in immersive augmented reality, which could come from inexperience with the medium but also lack in quality of the tool. *P2* reported that he had to put in similar amounts of effort in both deployments, but felt significantly more successful working with a computer screen and reported higher frustration working in immersive augmented reality. He felt much more efficient, working more successfully with the same effort, with the computer screen and consequently reported emotions like *reliable* and *confident*.

4.3 Lessons Learned

Scatter plot When surveying software performance in the city visualization developers can miss some detail, in which case an overview of the immediate past performance can help. Our approach to provide such an overview is the scatter plot. We observed that participants used the scatter plot well to compare performance at different times (*T7*) but otherwise struggled to interact with it. We observe some issues with the scatter plot:

1. Selection involves head movement. Reliable selection thus requires stable head movement and posture, which can be hard when the developer is not used to it. Additionally, performing an airtap gesture can further disturb the steady head posture.
2. Since the Y-axis represents classes (*i.e.*, contains all classes in the system), marks in the scatter plot are very small in that dimension and difficult to select.
3. Configuring the buffer of the scatter plot to be too large can lead to performance issues, since too many marks would be contained, making the visualization tool slow.

To ease selection we coped with the former issue by enlarging marks automatically when hovering over them. Since the size of the buffer is configurable, the latter issue can be avoided. We observe that additional work has to be put into the scatter plot to increase its usability.

Scale PerfVis aims to give developers an overview of the live performance of a software system with the least disruption possible to the IDE. We tested PerfVis visualizations scaled at various sizes. We observed

that the visualizations had to be scaled to be large enough to give an overview of the complete system, but also small enough to be used as a complement to a computer screen.

Text We learned that the position, rotation, and scale of text can greatly impact usability when navigating by head movement. In an early prototype we chose to display the name of classes on top of their corresponding building, however, we found that depending on the position of the user and the size of the building, the text might be difficult to read or occluded by neighbouring buildings. We addressed these issues by positioning the text in relation to the position of the camera. Specifically, we positioned the text at the top left corner of the user's field of view. Benefits of this approach are: (i) text is never occluded, (ii) text always faces the user, and (iii) no movement of the head and only little movement of the eyes is required to read the text.

Color We confirmed that selecting colors with high contrast eases distinction, recognition, and readability of the elements of the visualizations. Also that abrupt changes in the color of buildings burden the analysis of the city visualization. For instance, classes that are intensively called at some point in time and then stop being called afterwards can lead to a sequence of colors that can abruptly change from an intense red to a light gray. The result looks closer to a sequence of independent pictures than to an animation of the same city visualization. To address this issue we standardized the minimum time after which a building is colored and added an animation to smoothly change colors.

We used a continuous range of colors to represent the number of method calls to classes (*i.e.*, color of buildings). During the study, we observed that the continuous range may not constitute the best approach, since participants struggled to distinguish the shades of red. In future, a discontinuous range of colors to visualize method calls could be used.

4.3.1 Threats to Validity

Various issues might influence the validity this evaluation: (1) we compared user performance and user experience (dependent variables) of participants using two different media (dependent variable), *i.e.*, immersive augmented reality and a standard computer screen. To mitigate the learning effect we designed the study to use multiple city visualizations and dynamic performance data for each deployment. In an effort to produce similar city visualizations that do not a represent bias, we derived one city visualization from the other one, by keeping the layout and randomly reassigning heights to buildings. All dynamic performance data was generated following the same procedure, which guarantees to produce uniformly distributed values; (2) To mitigate external bias, we included in the study the use of the city visualization of a fairly popular software system (*i.e.*, Roassal for Pharo Smalltalk). Tasks were inspired from questions that arise in software performance issues described in the literature. To mitigate bias introduced by the characteristics of the dynamic performance data, we used data randomly generated. Additionally, developers participating in the study represent a convenience sample, and might have introduced a bias.

Finally, as the quality of the visualizations might have affected the obtained results, we mitigated this fact by using the ”Holographic Remoting Player” feature to stream the visualization through a wireless network, and making sure of a suitable bandwidth and latency. (3) To increase the reliability of the study, we provide the protocol that we followed, furthermore we present the procedure used to score participants answers in the tasks, and the procedures used to calculate completion time in Section 4.2. The set we used to collect participant's emotions and the NASA TLX is publicly available.

5

Conclusion and Future work

The impact code changes have on performance are hard to predict. A visualization can raise the awareness for the impact but visualization tools force the user to change focus to a different task (*e.g.*, from the IDE to the visualization tool). We conjecture that a pervasive tool (an omnipresent companion during the development process) can provide insight into the performance, without development process.

We propose PerfVis, a pervasive visualization approach to support developers in the analysis of software performance. We display PerfVis in augmented reality, to gain an additional visualization dimension and leave the IDE undisturbed. We selected the city metaphor to visualize a software, since research has show its effectiveness in software visualization [15, 17, 18]. We included a complementary scatter plot, to give developers an overview of the performance during the last minutes. We argue that our approach can provide pervasive performance awareness without disrupting the development process.

We report on lessons learned collected from the development of PerfVis. For instance, we observed that scaling the visualization to a suitable size was a challenging task. On the one hand the visualization needs to be as large as possible to allow users obtain a detailed view, but there is a risk of scaling the visualization to a too large size which would present a barrier to the purpose of our tool (*i.e.*, as an unobtrusive companion to the existing development setup). On the other hand, scaling the visualization to a too small size would impede the users ability to accurately interact with the visualizations, in particular, with small elements. Therefore, we explored various sizes during pilot experiments and found a suitable size that balances user performance in interaction and exploration.

We also learned that text position and rotation can be an issue in immersive augmented reality if choices are not well grounded. For instance, we placed the text on top of the buildings and observed that when the name of a small class is displayed the text can be hidden by surrounding buildings. We explored some other ways and realized that since selection is based on head movement the displayed text has to be readable with little head and eye movement, since both kinds of movement might slightly change the positioning of the head thus moving selection away from the desired element.

Finally, we chose to place the text in a fixed position and rotation in respect to the camera, which guaranteed that the text was always facing the developers and never occluded behind the buildings.

To evaluate the effectiveness of our tool we designed and conducted a user study. We carried out a pilot study to identify ways to improve our tool. For instance, we observed that to trigger functionalities of the visualizations through new types of interactions (*e.g.*, airtap, head movements) can be hard to achieve for users who have little or no experience with the involved technology. In the study, we compare our visualization tool displayed in immersive augmented reality to the same visualization displayed on the computer screen. We collect data of the user performance (*i.e.*, completion time, correctness, recollection) and user experience (*i.e.*, emotions, cognitive load)of ten participants. We analyze the data, and discuss the results.

In the evaluation of PerfVis we found that user performance was similar for participants who used visualizations in an immersive augmented reality and on a computer screen. We consider this result positive since participants had much less experience using an immersive augmented reality device than a computer screen. Independent of the medium used to display the visualizations participants reported to obtain a good user experience. After working in immersive augmented reality, participants reported many positive emotions that might be a symptom of a high engagement (*e.g.*, interested, open), but also some negative emotions (*e.g.*, confused, angry) possibly due the lack of functionalities and complex interactions. We observe mixed results in terms of cognitive load. Some participants experienced a high cognitive load, especially high frustration and low success when working in immersive augmented reality, while others felt more successful and less frustrated in immersive augmented reality whom in return felt frustrated when working with a computer screen. We conjecture that our design choices (*e.g.*, implementation of the scatter plot) might have some impact in the negative experience of participants during the experiment. However, we gained valuable insight for future work in order to mitigate some negative feelings of user experience.

All in all we discuss the benefits and challenges of an immersive augmented visualization tool. Our work demonstrates that a pervasive visualization is feasible, and also that it can bring possibilities to improve the quality of the working environment for software developers.

We outline various avenues for future work:

1. **Time Control:** We observe that users would benefit from being able to stop and rewind the dynamic scatter plot visualization. For example, user can use the city visualization to detect an anomaly in the performance of a system, and then use the scatter plot visualization to investigate what occurred in the past to identify potential explanations of such abnormal behavior of performance.

2. **User Specific Layout:** Users could benefit from being able to modify the layout of the city visualization. For instance, they could move particular buildings of interest to the front, and keep other buildings in the back.
3. **Filtering:** Following the visualization mantra [27], users could filter the displayed buildings according to their current needs (for instance, by making them transparent).
4. **Aggregation:** Users could set the granularity at which they want to analyze the performance of a system. For example, they could be interested in analyzing performance at the *package* level. Thus, all classes that belong to the same package would be represented as a single building.
5. **Collaborative Visualization**: We could allow multiple users to explore the same visualization simultaneously and interact with each other.
6. **Scale**: In future work, choosing the scale more deliberately and allowing re-scaling when the visualization is running could improve the usability.

Acknowledgment

I would like to thank Dr. Leonel Merino for supervising this thesis and all participants for taking part in the user study.

Bibliography

[1] A. Adamoli and M. Hauswirth. Trevis: a context tree visualization & analysis framework and its use for classifying performance failure reports. In *Proc. of SOFTVIS*, pages 73–82. ACM, 2010.

[2] J.-P. S. Alcocer, A. Bergel, S. Ducasse, and M. Denker. Performance evolution blueprint: Understanding the impact of software evolution on performance. In *2013 First IEEE Working Conference on Software Visualization (VISSOFT)*, pages 1–9, Sept. 2013.

[3] A. Bergel, F. Bañados, R. Robbes, and D. Röthlisberger. Spy: A flexible code profiling framework. *Journal of Computer Languages, Systems and Structures*, 38(1), Dec. 2011.

[4] W. Blochinger, M. Kaufmann, and M. Siebenhaller. Visualizing structural properties of irregular parallel computations. In *Proc. of SOFTVIS*, pages 125–134. ACM, 2005.

[5] D. A. Bowman and R. P. McMahan. Virtual reality: how much immersion is enough? *Computer*, 40(7), 2007.

[6] W. De Pauw, H. Andrade, and L. Amini. Streamsight: a visualization tool for large-scale streaming applications. In *Proc. of SOFTVIS*, pages 125–134. ACM, 2008.

[7] W. De Pauw and S. Heisig. Zinsight: a visual and analytic environment for exploring large event traces. In *Proceedings of the 5th international symposium on Software visualization*, SOFTVIS '10, pages 143–152, New York, NY, USA, 2010. ACM.

[8] W. De Pauw, S. Krasikov, and J. Morar. Execution patterns for visualizing web services. In *Proceedings ACM International Conference on Software Visualization (SoftVis'06)*, New York NY, Sept. 2006. ACM Press.

[9] A. Elliott, B. Peiris, and C. Parnin. Virtual reality in software engineering: Affordances, applications, and challenges. In *Proc. of ICSE*, pages 547–550. IEEE Press, 2015.

[10] F. Fittkau, A. Krause, and W. Hasselbring. Software landscape and application visualization for system comprehension with ExplorViz. *Information and Software Technology*, 87:259–277, 2017.

[11] P. Khaloo, M. Maghoumi, E. Taranta, D. Bettner, and J. Laviola. Code Park: A new 3D code visualization tool. In *Proc. of VISSOFT*, pages 43–53. IEEE, 2017.

[12] T. D. LaToza and B. A. Myers. Hard-to-answer questions about code. In *Proc of. PLATEAU*, pages 8:1–8:6, New York, NY, USA, 2010. ACM.

[13] S. Lin, F. Taïani, T. C. Ormerod, and L. J. Ball. Towards anomaly comprehension: using structural compression to navigate profiling call-trees. In *Proc. of SOFTVIS*, pages 103–112. ACM, 2010.

[14] J. I. Maletic, A. Marcus, and M. Collard. A task oriented view of software visualization. In *Proceedings of the 1st Workshop on Visualizing Software for Understanding and Analysis (VISSOFT 2002)*, pages 32–40. IEEE, June 2002.

[15] L. Merino, A. Bergel, and O. Nierstrasz. Overcoming issues of 3D software visualization through immersive augmented reality. In *VISSOFT'18: Proceedings of the 6th IEEE Working Conference on Software Visualization*, pages 54–64. IEEE, 2018.

[16] L. Merino, A. Bergel, and O. Nierstrasz. Overcoming issues of 3D software visualization through immersive augmented reality. In *Proc. of VISSOFT*. IEEE, 2018.

[17] L. Merino, J. Fuchs, M. Blumenschein, C. Anslow, M. Ghafari, O. Nierstrasz, M. Behrisch, and D. Keim. On the impact of the medium in the effectiveness of 3D software visualization. In *Proc. of VISSOFT*. IEEE, 2017.

[18] L. Merino, M. Ghafari, C. Anslow, and O. Nierstrasz. CityVR: Gameful software visualization. In *Proc. of VISSOFT*. IEEE, 2017.

[19] L. Merino, M. Ghafari, and O. Nierstrasz. Towards actionable visualization for software developers. *Journal of Software: Evolution and Process*, 30(2):e1923–n/a, 2017.

[20] L. Merino, M. Hess, A. Bergel, O. Nierstrasz, and D. Weiskopf. PerfVis: Pervasive visualization in immersive augmented reality for performance awareness. In *Proc. of ICPE*, page in review. IEEE, 2019.

[21] S. Moreta and A. Telea. Visualizing dynamic memory allocations. In *Proc. of VISSOFT*, pages 31–38. IEEE, 2007.

[22] K. Ogami, R. G. Kula, H. Hata, T. Ishio, and K. Matsumoto. Using high-rising cities to visualize performance in real-time. *arXiv preprint arXiv:1709.05768*, 2017.

[23] D. Raja, D. Bowman, J. Lucas, and C. North. Exploring the benefits of immersion in abstract information visualization. In *Proc. Immersive Projection Technology Workshop*, pages 61–69, 2004.

[24] S. P. Reiss. The paradox of software visualization. *VISSOFT 2005. 3rd IEEE International Workshop on Visualizing Software for Understanding and Analysis*, page 19, 2005.

[25] M. Shahin, P. Liang, and M. A. Babar. A systematic review of software architecture visualization techniques. *Journal of Systems and Software*, 94:161–185, 2014.

[26] B. Sharif, G. Jetty, J. Aponte, and E. Parra. An empirical study assessing the effect of SeeIT 3D on comprehension. In *Proc. of VISSOFT*, pages 1–10. IEEE, 2013.

[27] B. Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. In *IEEE Visual Languages*, pages 336–343, College Park, Maryland 20742, U.S.A., 1996.

[28] R. Souza, B. Silva, T. Mendes, and M. Mendonça. SkyscrapAR: An augmented reality visualization for software evolution. In *Proc. of WBVS*, 2012.

[29] Y. Tymchuk, L. Merino, M. Ghafari, and O. Nierstrasz. Walls, pillars and beams: A 3d decomposition of quality anomalies. In *VISSOFT'16: Proceedings of the 4th IEEE Working Conference on Software Visualization*, pages 126–135. IEEE, 2016.

[30] J. A. Wagner Filho, M. F. Rey, C. M. Freitas, and L. Nedel. Immersive visualization of abstract information: An evaluation on dimensionally-reduced data scatterplots. In *Proceedings of the 25th IEEE Conference on Virtual Reality and 3D User Interfaces (March 2018)*, page 4, 2018.

[31] J. Waller, C. Wulf, F. Fittkau, P. Dohring, and W. Hasselbring. Synchrovis: 3D visualization of monitoring traces in the city metaphor for analyzing concurrency. In *Proc. of VISSOFT*, pages 1–4. IEEE, 2013.

[32] R. Wettel, M. Lanza, and R. Robbes. Software systems as cities: a controlled experiment. In *Proceedings of the 33rd International Conference on Software Engineering*, ICSE '11, pages 551–560, New York, NY, USA, 2011. ACM.

[33] Y. Wu, R. H. Yap, and F. Halim. Visualizing windows system traces. In *Proc. of SOFTVIS*, pages 123–132. ACM, 2010.

A

Anleitung zum wissenschaftlichen Arbeiten

A.1 Anleitung zu wissenschaftlichen Arbeiten

The following Appendix provides a guide to set up the various parts of the install and run PerfVis. Essentially, four parts are needed to run the visualizations: 1) the Pharo environment, 2) a web server, 3) the PerfVis Unity Project, and 3) a HoloLens enabled device. The setup can be run in two different ways:

1. The Pharo environment, the web server, and the Unity Project running on the same machine, like in Figure A.1 ⓐ.
2. The web server and Pharo environment on one machine, and the Unity project on another, like in Figure A.1 ⓑ.

No matter which setup is chosen, the following instructions apply. These instructions result in a visualization of the Roassal2 visualization engine in Pharo. We assume that the reader has basic knowledge in the navigation of C#, Pharo, and Console scripting.

A.1.1 Required Tools

For the visualization to work you require:

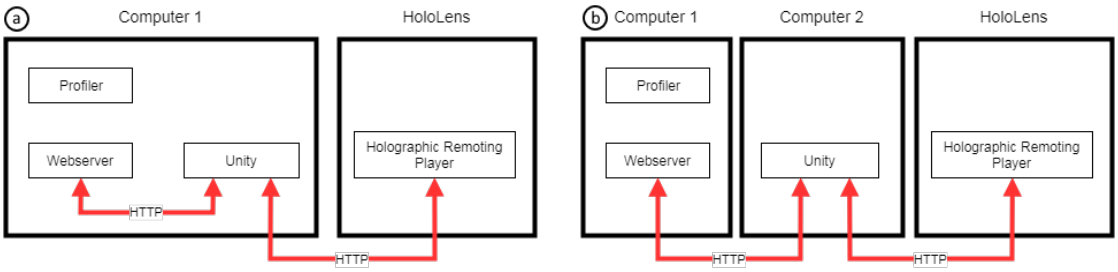

Figure A.1: The two ways to setup the visualizations

- a HoloLens device,
- a computer running Windows 10 (build 14318 or later),
- a wireless network, and
- (Optionally) a second computer machine.

All machines have to be connected through a network.

A.1.2 Install Tools and Environments

A.1.2.1 Pharo Environment

- Download a new Pharo image and VM if necessary
- Install *Spy2* from the *Catalog* browser
- Open the *Monticello* browser, click *+Repository* and select the *http://smalltalkhub.com* button
- Enter owner: `merino`, project: `SpyToHolo`, and select the *OK* button
- Select `Spy2-Examples-LeonelMerino.49.mcz`, and select the *Load* button
- Open the *System* browser, and select `Spy-Examples` → `Counting` → `S2C` → `visualization` → `recordData:`
- Set the *file* reference (see Figure A.2) to your preferred location to save the file, *e.g.*, `'Users/hess/Desktop/city-colors.csv'`
- Open a new *Playground*, and write:

```
1  profiler := S2C new.
2  profiler startOnPackageNamed: 'Roassal2'.
3  profiler animatedVisualization.
```

- Select the three lines, and select *Inspect It*

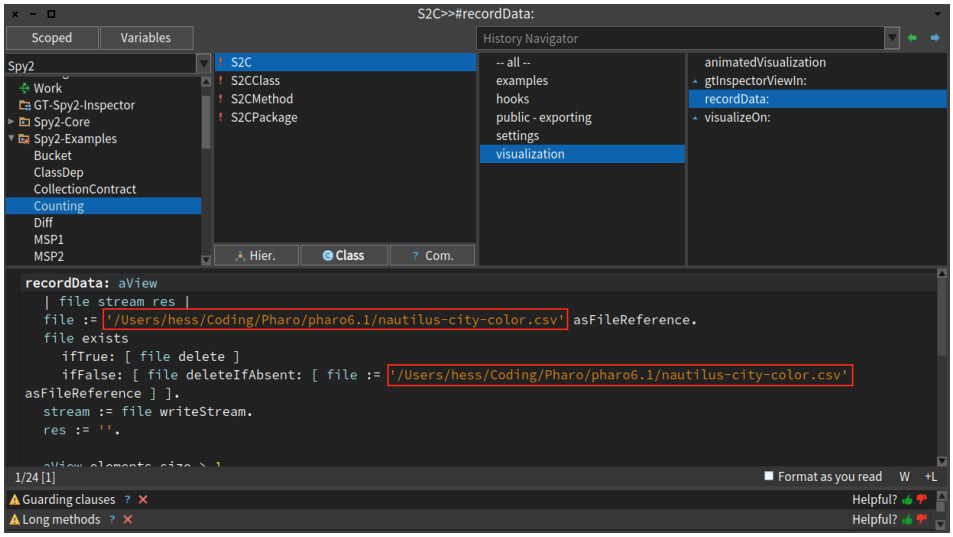

Figure A.2: Code reference for the save location of the pharo file

- After a moment the *Inspector* tool will open, *Spy2* will be monitoring
- To stop the monitoring tool, execute `S2Profiler remove`

A.1.2.2 Web server

For the web server we recommend to use Python, since it offers an easy to set up web server to serve from the filesystem (but any other web server works as well). The web server needs to be running on the same machine as the Pharo Environment.

- Acquire a distribution of Python and install it
- In a Console, navigate to the location you set up earlier, *e.g.*, *'Users/hess/Desktop'*
- Run `py -m http.server 8000` in the Console
- Remember the IP address and port of the web server, *e.g.*, *192.168.2.3:8000*

A.1.2.3 Unity Project & HoloLens

- Download Unity 2017.3.0b8.
- Clone the Unity project from its git repository `git://scg.unibe.ch/project-hess-holo-2018` and open it.
- In Unity, select the *Cube* GameObject, and set the *URL* variable of the `UpdateData.cs` component to {web server IP address}/{file name}, *e.g.*, `192.168.2.3:8000/city-colors.csv`.

- On the HoloLens, go to the Microsoft Store, and install the *Holographic Remoting Player*.
- On the HoloLens, start the *Holographic Remoting Player*.
- In Unity, go to the *Window* menu and select *Holographic Emulation*.
- Set *Emulation Mode* to *Remote to Device*, and enter your HoloLens' IP address for *Remote Machine*.
- select the *Connect* button (the Connection Status should change to *Connected*, and the screen in your HoloLens should go blank).

A.1.3 Running the Visualization

The multiple parts of PerfVis should now be connected and configured properly. Before entering *play mode* make sure that the *Inspector* window in *Pharo* is not minimized, and it is visible. Otherwise the performance metrics will not be updated in the web server.

Finally, enter *play mode* in Unity. After a couple seconds the visualizations should be seen. Notice that there are always two classes active in the visualizations. We suspect this is due to the *Inspector* window, which needs to be opened and that displays metric values.

To stress *Roassal2* go to the *World* menu → Roassal → Roassal Examples, select some category, and execute an example.