2018 IEEE Working Conference on Software Visualization

A Controlled Experiment on Spatial Orientation in VR-based Software Cities

Marc-Oliver Rüdel
University of Bremen, Germany
mor@uni-bremen.de

Johannes Ganser
University of Bremen, Germany
ganser@uni-bremen.de

Rainer Koschke
University of Bremen, Germany
koschke@uni-bremen.de

***Abstract*—Multiple authors have proposed a city metaphor for visualizing software. While early approaches have used three-dimensional rendering on standard two-dimensional displays, recently researchers have started to use head-mounted displays to visualize software cities in immersive virtual reality systems (IVRS). For IVRS of a higher order it is claimed that they offer a higher degree of engagement and immersion as well as more intuitive interaction.**

On the other hand, spatial orientation may be a challenge in IVRS as already reported by studies on the use of IVRS in domains outside of software engineering such as gaming, education, training, and mechanical engineering or maintenance tasks. This might be even more true for the city metaphor for visualizing software. Software is immaterial and, hence, has no natural appearance. Only a limited number of abstract aspects of software are mapped onto visual representations so that software cities generally lack the details of the real world, such as the rich variety of objects or fine textures, which are often used as clues for orientation in the real world.

In this paper, we report on an experiment in which we compare navigation in a particular kind of software city (EvoStreets) in two variants of IVRS. One with head-mounted display and hand controllers versus a 3D desktop visualization on a standard display with keyboard and mouse interaction involving 20 participants.

***Index Terms*—Visualization, Virtual reality, Software engineering, Navigation**

I. Introduction

The city metaphor in software visualization has emerged from two-dimensional tree maps [1]. Tree maps in software visualization show the hierarchical structure of software and particular characteristics quantified as metrics by leveraging a given two-dimensional space very space-economically. The metrics determine the area of the software hierarchy's leaves (e.g., classes in a package hierarchy or files in source-code directories). The leaves' metrics are accumulated to their containing elements in the hierarchy. The third dimension in the city metaphor is used to show yet another metric (e.g., coupling of classes). Through this third dimension, the areas of two-dimensional tree maps become three-dimensional blocks resembling tower blocks in modern city centers. As a consequence, humans perceive this visualization as a city, which has become eponymous for this type of visualization.

Several researchers have explored the city metaphor for visualizing software using the three-dimensional extension of tree maps known as *code cities* [2], [3]. Code cities are very compact in a given space. On the other hand, deeply nested hierarchies are difficult to distinguish because they offer very few distinct patterns for human orientation. Moreover, when not only a single version of a software is visualized but its evolution over a longer period of time where elements are removed, added, or changed, it may become difficult to maintain the human observer's mental map when the layout changes drastically.

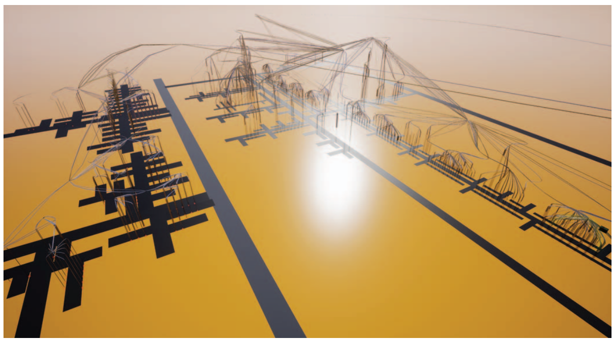


Fig. 1. Beautified bird's eye view of our automatically generated evaluation system

Steinbrückner has introduced an alternative kind of 3D visualization based on the city metaphor, named *EvoStreets* [4]. The nesting is represented by branching streets (see Figure 2 for examples). The top-most street represents the root of the hierarchy and is visualized as the thickest line. Each lower level of the hierarchy is drawn as a right-angled branching street with decreasing thickness. The leaves of the hierarchy are drawn as blocks as in code cities. This layout is less space economic, but offers the advantage that the hierarchy is more easily recognizable, distinct patterns of city districts are more often to emerge, and evolutionary changes can more easily be integrated maintaining the spatial relations of previous versions.

The early approaches of code cities and EvoStreets have used three-dimensional rendering on standard two-dimensional displays. Immersive virtual reality systems (IVRS) that use head-mounted displays (HMD) for consumers have become wide-spread [5], which has lead to the availability of a variety of affordable HMDs and the necessary software infrastructure to program virtual realities. For these reasons, researchers in software visualization have also started to use HMDs to visualize software cities in IVRS [6], [7]. For IVRS of a higher immersion equivalence class as introduced by Slater in [8] and refined in [5], [9], it is claimed that they offer a higher degree of engagement and immersion as well as more intuitive interaction. Research in virtual reality found that the use of a HMD may increase the orientation in three-dimensional virtual environments due to the natural rotation and movement that this medium allows [10]. Thanks to these two components, the question arises whether a developer would benefit from a HMD when exploring a software city, too. Spatial orientation, however, may also be a challenge in IVRS as already reported by studies on the use of IVRS in domains outside of software engineering such as gaming, education, training, and mechanical engineering or maintenance tasks [11]. This might be even more true for the city metaphor for visualizing software. Software is immaterial and, hence, has no natural appearance. Only a limited number of abstract aspects of software are mapped onto visual representations so that software cities generally lack the details of the real world, such as the rich variety of objects or fine textures, which are often used as clues for orientation in the real world.

To further study the potential advantages and disadvantages of HMDs in 3D visualization based on the city metaphor, we ran a controlled experiment—involving 20 participants—in which we compared navigation tasks in EvoStreets (including relation visualization based on hierarchically bundled edges [12]) within the two classes of equivalence of IVRS: one with a HMD and hand-held controllers and one with a 3D desktop visualization on a standard display with keyboard and mouse.

As stated above it is claimed that IVRS of the HMD class offer a more intuitive orientation. But as Ruddle et al. have pointed out, it cannot be taken for granted that participants perform necessarily better with HMDs [13]. This has lead us to formulate our first and primary hypothesis for the experiment:

Hypothesis 1: (HMD over Desktop): Users navigate more effectively and efficiently in EvoStreets when they use a 3D HMD instead of a pseudo 3D desktop system as a display device.

We measure the *efficiency of navigation* as the completion time of homing tasks (finding back to the initial starting point) after some time was spent inside the visualized software project. A homing task is considered *effective* when the participant found back to the initial position.

This kind of performance measurement may be influenced by many factors. Research on human-computer interaction points out that the interaction method might have a greater influence on the efficiency than the used medium [6], [14]. As a consequence, we should take a look at the familiarity of participants with our interaction method (hand-held controller versus keyboard and mouse). Other studies highlight that frequent playing of computer games might have an impact on the reaction time [15]–[17], vision [18], and learning speed [19], [20] of the players [21], [22], taking us to the following hypothesis:

Hypothesis 2: (influence of gaming experience): Users who are familiar with navigating using a keyboard in computer games achieve higher task completion efficiency.

The height, width, and breadth of blocks in an EvoStreet is—unlike in real cities—influenced by the metrics mapped onto these visual attributes. EvoStreets are designed for visualizing the evolution of software where metric values may change (see Section III-B). Moreover, different metric mappings are applied frequently to show different aspects (e.g., #defects vs. #developers, size vs. complexity, etc.). Although this mapping does not change the structure, the visual appearance can still be effected drastically when these metrics are radically different. This influence is even more of concern for different versions of a software. Therefore it is of interest whether participants can transfer their spatial knowledge between visualizations when different metrics are used, pertaining to the next hypothesis:

Hypothesis 3: (influence of different metric mappings): Users who are already familiar with the EvoStreet of a software for one particular metric mapping can navigate equally well if only the metric mapping changes (same structure, same starting point).

The remainder of this paper is organized as follows. Section II discusses related work in IVRS in general and software visualization based on the city metaphor in particular. Section III presents our controlled experiment and Section IV discusses its results. Section V finishes with our conclusions.

II. Related Work

This section describes related research. We will first summarize related work in the area of navigation in virtual environments in general and interaction design and then works on using the city metaphor in software visualization.

Navigation in Virtual Reality. The topic of navigation in virtual environments has been researched for a while [23]–[25] and will be a topic of interest in the future due to the fast progress in hardware and rendering techniques that allow the creation of much bigger and more complex environments.

Sousa Santos et al. give an overview on the virtual environments, discussing several papers that compared HMDs to desktops [26], and describe an experiment on navigation in a virtual maze they conducted. They found that, although users were generally satisfied with the VR system and found the HMD interaction intuitive and natural, most performed better with the desktop setup.

Ruddle et al. investigated the possible benefit of HMDs over traditional desktop systems with respect to navigation in virtual environments [13], [27]. They researched the ability to navigate within large virtual buildings, consisting of one floor with nine rooms and one corridor [13]. The experiment was a repeated round tour through several rooms, first with instructions, later without. In another experiment, they focused on proprioceptive feedback and its influence on navigation within a maze as virtual environment [27]. In their first experiment in a virtual building [13], they found that the HMD had an advantage to the speed of the participants, in contrast to their later experiment [27] and to Sousa Santos et al.'s experiment [26], which both took place in a maze.

Besides the direct comparison between the two classes of IVRS, other experiments where designed to research the spatial cognitive capabilities within virtual environments. Homing tasks [28]—also known as triangle completion tasks [29]—were used to observe the influence of the amount of visual-only spatial information on the spatial orientation of participants. These experiments were conducted in very constrained virtual environments with the task of moving a cylinder. They found a strong association of triangle layout on homing performance, but no effect of geometrical fields of view: variations in the amount of simultaneous visible spatial information did not influence the acquisition of spatial knowledge in the environments used. Whether and how these results obtained for very specific geometrical problems can be transferred to navigation in virtual software cities is a separate question. The concept of *homing* is what we have applied in our experiment: finding a path back to an intial starting point.

As for the interaction, especially the locomotion in virtual environments has an impact on the spatial orientation of participants as Bowman et al. [23] found out in their maze based experiment. Locomotion describes ways to move through the virtual worlds such as teleportation, flying, etc. Ways of locomotion differ also in terms of who is in control. A system can transport a user from one point to another, or a user can decide on his or her own where to move. Bowman et al. found that the latter better supports spatial orientation. We chose user-controlled flying with a constant speed, that is, a user can point freely in the direction she or he wants to go and click a button on the controller to start moving. This way of locomotion has shown to be simple and intuitively understandable and decreases the chances of sickness.

Compared to classical software visualizations, 3D representations of software have the added value of the spatial dimension. However, in our scenario, this spatial dimension can only be used by explorers of the software if they have orientation. In order to acquire orientation, it is necessary to move in the environment [24], [30]. We did not specify any paths that the participant had to use on his/her way home, but—as usual in the homing task—left the path finding to the subject.

The type of movement seems to play an essential role: a continuous movement is advantageous for the formation of a spatial model [23], [24], [30], [31]. According to [23], the flight metaphor seems to have been one of the best studied motion metaphors. A flight metaphor offers—even in reality—the highest flexibility of movement (the restrictions on given 'paths', independent path finding), continuous updating of the spatial perception, and a bird's eye view.

Furthermore, many types of movement used in modern computer games seem to be of limited suitability for the acquisition of spatial orientation, since, according to [23], they tend to lead to disorientation. We decided against both teleportation and guided movements.

Many of these experiments have been conducted in small and/or maze like virtual environments. So for the purpose of software comprehension through exploration of city visualizations findings might differ, because of the difference in these kinds of virtual environments. This is especially true for visualizations of modern software systems that consist of several thousands of entities shown as buildings.

Interaction Design. The interaction with a system has an impact on the performance as already discussed. In an experiment, interaction must be easy to learn, so that neither the interaction nor learning effects influence the results too much. To achieve such an interface, several approaches are feasible. First, we can use an interface that is already well known, second (which is a special case of the first) we can use a natural user interface, meaning an interface that is just as it would be in the real world and therefore already well known. In the latter case, Mine provides an overview on different approaches [32] as well as Guan and Zheng, and Nielsen et al. in regards to gestures [33], [34].

In our experiment we decided to use the simple pointing gesture for the HMD, to move and also to mark elements in our EvoStreets, activated with a button on the controller and visualized as a solid beam in the VR for feedback to the user. Desktop systems with mouse and keyboard are today well established. Common interfaces for movement in virtual environments can be found in games, game engines and application for graphic modeling. Other research that compared the common WASD input method were described by McMahan et al. and Nacke et al. [35], [36].

Software City Metaphor. We have already introduced code cities and EvoStreets in Section I. We have chosen EvoStreets for our experiment because—in contrast to code cities in the line of Wettel's seminal work [2]—it has a strong continuity over changes between versions and provides a structure that stretches itself out, leaving more space between objects of interest [4]. For virtual environments that are meant to be explored and not only to be seen from afar it is of advantage that EvoStreets take advantage of the infinite virtual space.

Each visualization has a specific target group and information that it elaborates, which is also true for software cities. Knight and Munro give an overview on virtual reality for software visualization [37]. Panas et al. approached the idea of a multi-stakeholder visualization in 3D [38], [39]. Recent researchers such as Khaloo et al. [40] and Merino et al. [6] approach developers at their software comprehension tasks. Ogami et al. also target developers with their code city, but focus on runtime performance information for the localization of bottlenecks or leaks [41]. On the tool side there are the contributions by Schreiber and Brüggemann with a visualization technique for OSGi component specification based systems [42] and Baum et al. with GETAVIZ, a tool for evaluation of software visualizations [43].

III. Controlled Experiment

To verify the hypotheses stated in Section I, we conducted a controlled experiment in which participants had to solve a navigation task in two virtual environments of the described different equivalence classes of IVRS [5], [8], [9]: one with a HMD and hand-held controllers and one with a 3D desktop visualization on a standard display with keyboard and mouse. In the following we summarize the design of our experiment.

We used a HTC Vive as HMD and a 30” monitor for the desktop. Our evaluation system ran on a simple gaming-ready PC with an NVIDIA GeForce GTX 1070 graphics card.

A. Design of the Experiment

The main objective of our experiment was to validate the influence of the IVRS equivalence class (independent variable) on the navigation efficiency and effectiveness of the participants in a typical EvoStreet. For this purpose, we defined *homing tasks*, which each participant should complete after a training phase on two different software projects (SW 1 and SW 2) with two different metric mappings (see Table I).

To counter learning effects between the two types of IVRS, we separated the participants randomly into two different groups. Group 1 started with a desktop computer and then moved on to the HMD, while group 2 followed the opposite order. Each software project was presented in two different variants which differed only in the metric mapping, but not the structure. That is, the only difference between these two variants was the width, breadth, and height of the buildings.

The first task for each type of IVRS was a simple training task that was ignored in the evaluation. The order of the software projects and chosen metric mappings was the same for all participants. As a consequence, the order of the viewed EvoStreets is consistent between the two different groups and allows us to make interpersonal comparisons between both groups where only the IVRS is varied. Every participant participated in all six tasks (including the training tasks), which allows us to compare the results intrapersonally from one type of IVRS and software project with varying metric mappings (Task 1-1 vs. Task 1-2 and Task 2-1 vs. Task 2-2) as well as between different IVRS (Task 1-1 and Task 1-2 vs. Task 2-1 and Task 2-2).

The homing task is based on a typical task in the area of software exploration: “Follow the call graph until you find the function you are looking for and then go back to where you came from.” Our scenario reflects a situation in which a developer must understand a given method A that calls other methods transitively. In order to understand A, she/he must understand a method B directly called by A, which in turn calls another method C, which calls yet another method D. The developer follows the call path A, B, C, D and is satisfied at D. She/he must now return to the original context A. The same happens in an IDE when you follow calls. Homing is an essential part here. Our experiment investigates whether developers can return to home without any additional means analogously to the real world, which VR code cities claim to simulate. To avoid overloading the participants, we limited the homing task to just three steps, before they had to find their way back to the starting point. So, participants with lesser memory capabilities should not have had a major disadvantage.

The participants were told in advance, that is, at their initial starting point, that they would be asked to return to this point before they started. They were reminded of that each time they entered a new EvoStreet to start a new homing task.

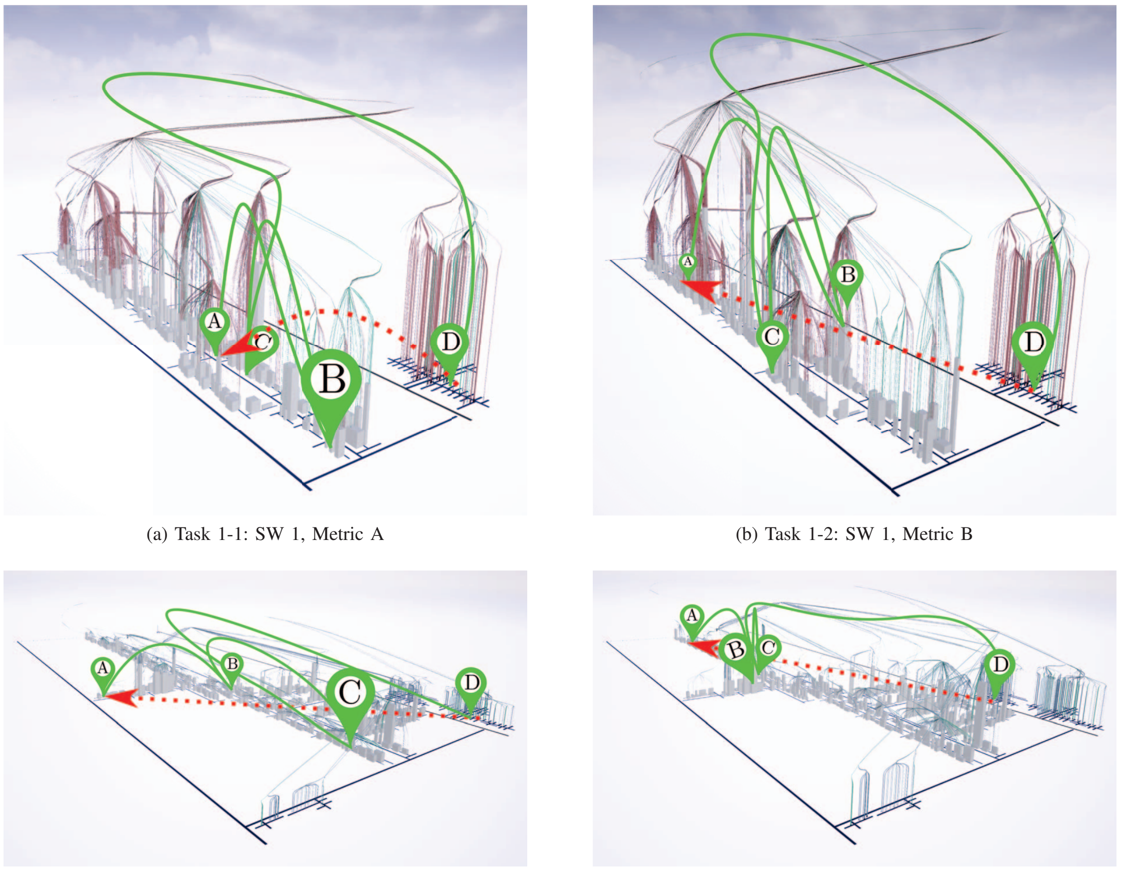
From each point in the call graph, there are several relations to other points. Only one of the relation leads to the correct next point. Both the points and the links are labeled with an alphanumeric code so that the participants merely had to follow the link with that code to get to the next node. Thus, the participants did not have to deal with the internal structure of the presented software to find the right way. They would only need to memorize the decision points to find their way back. They were not allowed to write down the labels.

The starting and ending points and both intermediate points of each path have been determined by us before the start of the test. We chose the points in such a way that they were located in different regions of the EvoStreets so that the participants could make themselves familiar with different city districts and also to enforce the need for a new orientation from one navigation task to another one. If the points had been too close together, the participants might have just followed unconsciously the memorized directions along the path of the previous task. Furthermore, based on our real-world experience, we assumed that navigation would be much easier when the very same path had to be taken a second time, even if the EvoStreet looks slightly different due to a changed metric mapping. To counter this learning effect, we chose a different path for each task. However, when choosing the points of the second tasks for the same IVRS, we took care that the paths between the points ran through areas of the first task in the same software project, so that the participants could possibly take advantage of the knowledge of the first task to allow the creation of a mental map.

To mitigate the influence of different experimenters, the same experimenter guided all participants in all tasks. For each participant, the experimenter briefly explained the task (homing as fast as possible). Before using a new virtual environment, participants were introduced to the control options in each of the virtual environments. Subsequently, the participants were allowed to perform the experiment on a smaller software project as a training task. The experimenter emphasized the task of the experiment (the homing task). Once the participants stated that they felt safe enough to handle the controls, the actual experimental treatment and measurement began.

During the experiment, the experimenter stated the number of the next edge to take and next node to arrive via this edge. The participants were always allowed to ask for these navigation information again. After reaching the final point in the path, the experimenter asked the participants to now return to the starting point as fast as possible. Since we did not want to measure the memory performance but the intuitive orientation ability of the participants, the experimenter was allowed to name them the alphanumeric codes, but not the direction or whether they have already found back to the starting point.

The participant had to tell the experimenter the building she or he thought to be the initial starting point. If this answer



(a) Task 1-1: SW 1, Metric A

(b) Task 1-2: SW 1, Metric B

(c) Task 2-1: SW 2, Metric A

(d) Task 2-2: SW 2, Metric B

Fig. 2. Homing tasks per software project, metric mapping, and IVRS: Subjects were instructed to follow the path along call edges from the starting point A over B and then C to the returning point D (shown in green) and then to return to A by themselves. The shortest path back from the returning point D to A is shown in red. Details of the software projects' size can be found in Table II.

was correct, the homing task ended and was considered solved. If not, the participant could continue. There was no limit on the attempts to identify the starting point. A participant could give up anytime if she or he felt that the homing task was past hope. The experimenter told the participant in this case that a canceled attempt is not bad, so as to not demotivate the participant for the next task.

The experimental setup was tested in a pilot study with three participants, where no problems occurred.

We measured the *effectiveness* and *efficiency* of task completion. The effectiveness of task completion was defined by two factors: Firstly, (a) whether the participant found the way back to the starting point on his own and secondly (b) how many points he had erroneously called the final target (the initial starting point). Efficiency was measured as the time required for task completion.

B. *EvoStreets*

We decided to show representations of real software as EvoStreets in our experiments. The global dependency graph for the selected software projects were extracted by the Java analyzer of the *Axivion Bauhaus Suite* [44]. As a graphics engine, we used the Unreal Engine 4.15, for which we created plug-ins and an application that could read the dependency graph from a file and transform it according to selectable parameters such as display depth, color, texture, scaling, and visibility in an EvoStreet. Hierarchically bundled edges of different types can be included in the EvoStreet, if needed.

We selected two open-source projects of the Apache Commons project [45], see Table II. The two projects are similar in size with respect to source lines of code (SLOC). However, software project SW 1 includes about twice as many Java files, a quarter more classes, and about twice as many methods.

Based on our basic tasks (navigation in a call hierarchy), we used the *method view* as the EvoStreets for the experiment. That means *buildings* represent methods, *relations* constitute calls between two methods, and *streets* represent syntactic nesting of packages and classes (cf. Figure 1).

Other researchers have proposed to use colors, textures, or

TABLE I
DESIGN OF THE EXPERIMENT

Phase	**Group 1**	**Group 2**
Demography	1. Data Privacy Statement 2. Security Statement 3. Waiver 4. Demographic Questionnaire	
Introduction	Explanation of the main purpose of the experiment	
Try-Out	Desktop	HMD
Task 1-1 (SW 1, Metric A)	Desktop	HMD
Task 1-2 (SW 1, Metric B)	Desktop	HMD
Try-Out	HMD	Desktop
Task 2-1 (SW 2, Metric A)	HMD	Desktop
Task 2-2 (SW 2, Metric B)	HMD	Desktop
Qualitative Statistics	1. meCue Questionnaire 2. SUS Questionnaire 3. additional questions	

other kinds of decorations to show additional information [2], [4]. These may also provide additional clues to memorize locations, but they may also overwhelm a user through information overloading while navigating. We have abstained from these visual attributes to avoid yet another dimension of variation for the experiment. We consider the investigation of the influence of colors, textures, and decorations on navigation as future research.

TABLE II
METRIC TOP-LEVEL DATA OF THE SOURCE CODE USED

	SW 1 commons-cli-1.4-src	SW 2 commons-chain-1.2-src
SLOC	6.681	8.010
# Java files	50	97
# classes	73	94
# methods	398	799
# relations	3.122	5.488

The height of a building represents the lines of code of the represented method, the width and depth of a building (from the direction of the associated street) represent the halstead complexity and the maximal nesting depth, respectively, of a method. Our EvoStreet layouter is based on the algorithm of Steinbrückner [4], supplemented by an adjustable distance between the buildings to better separate them visually from each other.

Because the call graph is our basic navigation target, we decided to always show all realtions. Steinbrückner [4] suggests the use of relations and their bundling in EvoStreets. To increase the clarity in graphs with many realtions, it is advisable to bundle the relations hierarchically [12]. The structure of an EvoStreet results from the hierarchy which we used also for the bundling of the connections.

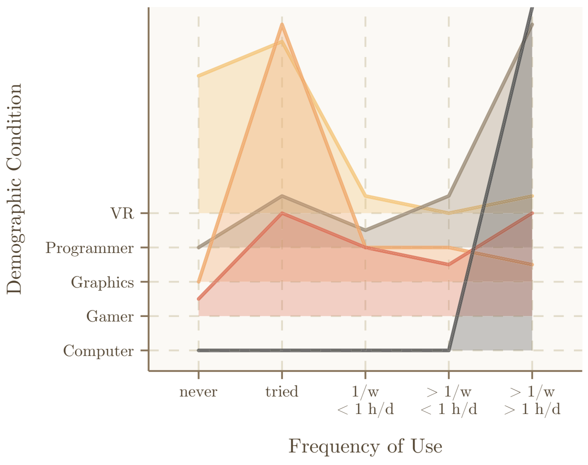


Fig. 3. Histogram of Demographic Informations

A participant can select individual elements (buildings and realtions) of the EvoStreet, thereby the elements are marked clearly visible and additional information is displayed (unique alphanumeric ID, metric values, source-location information).

C. Participants

In the absence of deeper insights into relevant key characteristics of developers and their distribution in the target population of developers in general and also because of practical reasons, we used convenience sampling to find participants for our experiment. We asked friends, fellow students, and colleagues for participation. All of them had a strong background in computer science and software development. Participation was voluntary and no incentive was given. Initially, 23 people agreed to participate but during the experiment three dropped out because of dizziness.

The age of the participants was in the range of 20–50 with only one exceptional participant at the age of 60. 15 of them were in between 20 and 35. Three were females. Eleven participants were students and nine researchers. We asked them for their experience in the use of computers, programming, computer gaming, graphic tools, and virtual reality in terms of frequency per week and hours per day (cf. Figure 3). We also asked the participants for a participative self-assessment of their perceived orientation skills On a scale of "bad/rather bad/rather good/good" more than 50 % of the participants rated themselves as "rather good" and 25 % as "rather bad".

IV. DISCUSSION AND RESULTS

During the experiment, we recorded the times each participant needed from one point to the next, and in particular the times for the way back (excluding the time required for explanations) and the exact way points taken. Furthermore, for the qualitative analysis, we used a screen recorder with audio and thus logged all statements of the participants as well as the meCue and SUS questionnaire.

We used the collected data to perform a quantitative analysis of the participants' effectiveness and efficiency based on the factors described above. Though giving up in reality is usually not an option, we allowed the participants to cancel their search if, during navigation, they felt that they had no more ideas where the target point might be.

A. *Effectiveness*

In this section, we report results relevant to hypotheses with respect to effectiveness. Specifically, for effectiveness, we analyzed how many of the participants found back and if they found it, how many wrong buildings they stated to be the initial starting point. We calculated the percentage of successful returns (a binary decision as to whether they reached the starting point or not) and the number of erroneous supposed starting points.

1) Lost or Found: Not all participants found back to the starting point. We observed a dropout rate of around 20 % regardless of other factors. Individual large differences were found. Some participants found back in each experiment, others reached the starting point in only 25 % of the tasks. If we consider the factors *IVRS* and *participant group*, we find in the first case nearly identical success rates for both IVRS (desktop: 0.83, HMD: 0.84), in the second case distinctly different success rates (group 1: 0.90, group 2: 0.75). The averaged values of all eight tasks across all participants are shown in Figure 4. The mean for all these task is 0.83 with a standard deviation of 0.02.

The very high dropout rate of 20 % compared to our pilot run surprised us. During the experiments, we did not notice any decreasing seriousness in the task processing of the participants. Thus, we attribute this high number, on the one hand, to a suppressed, physical burden on the participants (some participants had to deal with dizziness independently of the kind of IVRS), and on the other hand, to the particular characteristics of the participants in the pilot run.

Group 1 participants (who started with the desktop IVRS) returned to the starting point more frequently than group 2 participants (0.90 vs. 0.75) This may be due to the fact that group 1 members started with an IVRS already known and were able to carry over their experience from the desktop to the HMD IVRS, whereas group 2 started with an IVRS not yet known. As one can see in Figure 4, the successful return rates between the different tasks diverge (0.83 ± 0.02).

In summary, differences could be observed for the order in which the two different types of IVRS were applied, but not for the types of IVRS as such.

2) Number of Errors: On their way back, we recorded how often the participants called buildings home that were actually not the starting point they were looking for. 38 % of the erroneous tasks were done with 0–5 wrong selections. Some participants needed many attempts until they either found the target element or gave up. As shown in Figure 5 where the

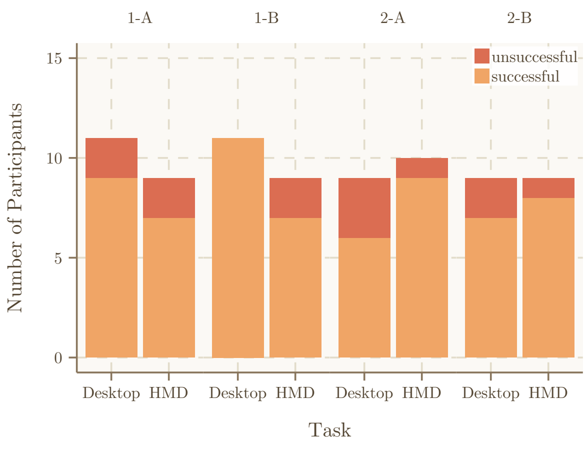


Fig. 4. Effectiveness—general success by subtask

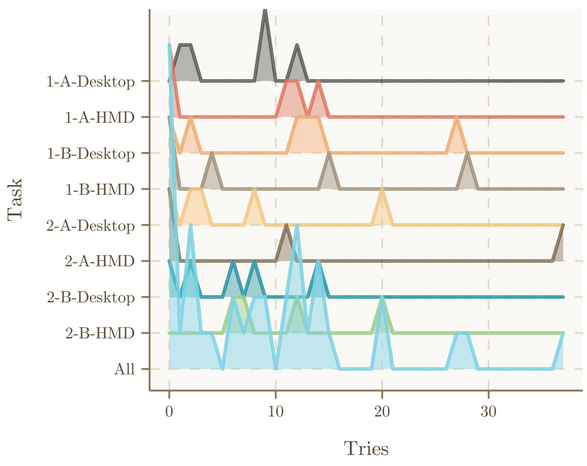


Fig. 5. Effectiveness—number of errors by subtask

full details of all tasks are presented, there does not seem to be a systematic difference between the different tasks.

The participants followed different strategies for the tasks. While the analysis of these strategies was not in the focus of our investigation, we were nevertheless able to identify patterns in task accomplishment. Partly the participants also mentioned during the task what they wanted to do differently/better in the next task (they thought aloud without being instructed to do so).

Some participants always flew with an "inner peace" very purposefully without fixation on a certain wrong building and chose the right home building well. Other participants asked a lot, tried many buildings, tried to explore the internal structure of the buildings to get an indication of where the sought-after place might be. These pursued a kind of brute-force try-and-error strategy. Whether this strategy was successful or not often depended on whether their unconscious or conscious orientation skills had guided them to the right region. Again other participants, despite knowing that they should search for the fastest route, shuffled all the way back to the starting point via the intermediate points C and B. Although this strategy was the most time-consuming, it had the highest probability of success at 100 %. Each participant appeared to make a time-accuracy trade off, which is likely to be driven by their prior experiences and personality.

3) Gamers/Graphic Artists: Gamers and graphic artists may have an advantage over other types of users given their experience in computer interactions similar to the ones we offered in our desktop IRVS as formulated in hypothesis 2. In the following, we will label gamers and graphic artists together as *gamers* and all others as *non-gamers*.

Gamers have a higher successful-return rate (88.10 % versus 77.14 %) but also a higher mean error rate (9.02 versus 6.8). Yet, both differences are not statistically significant with a p-value over 0.2.

4) Metric Mappings: Hypothesis 3 expects no difference between two different metric mappings with respect to effectiveness. Our measurements show that there is a higher successful-return rate (79.49 % versus 86.84 %) but a lower mean error rate (7.95 versus 9.50) for metric mapping A versus metric mapping B. Yet again, both differences are not statistically significant with a p-value of 0.4.

B. *Efficiency*

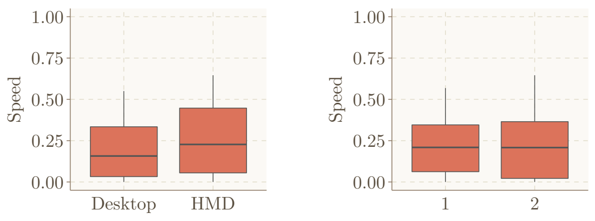
For efficiency we measured the time from the returning point to the starting point. A comparison of the required time in the homing task to the times in the previous steps from the starting to the return point would be suitable for generating a personal speed profile for the virtual environments or their control mechanisms. It would allow us to put the speed for their homing task into relation to their normal speed of travel. This was not the focus of our study.

Prior to the experiments, we determined for each homing task the shortest duration if a participant would go straight to the starting point without hesitation. We normalized the actually achieved times of the participants in relation to the best time, then converted these times into speeds. The resulting speed values are given in percent of a possible maximum speed.

Some of the tasks were canceled by the participants before reaching the target. The speeds for these tasks (about 20 %, see Section IV-A1) were set to 0 and included in the further calculation.

We used the Shapiro-Wilk test to check whether speed for desktop and HMD is normally distributed. The result suggests it is (W=0.92, p=0.00009), whereas the Kolmogorov-Smirnov test indicates that we need to reject the null hypothesis that speed is normally distributed (D=0.15, p=0.07), although the p-value is very close to our significance level of 0.05. If we apply the two tests to the subgroups of the 'gamers' and the 'non-gamers', we get again somewhat inconclusive results: while Shapiro-Wilk's test suggests a normal distribution in the group of 'non-gamers' (W=0.86, p=0.0003), but not for 'gamers' (W=0.95, p=0.055; again slightly above the significance level of 0.05), the Kolmogorov-Smirnov test indicates both groups to be normally distributed ('gamers': D=0.5, p=2e-09; 'non-gamers': D=0.5, p=5e-08). To be safe, we assume no normal distribution in the following.

1) Speed by Virtual Environment: This section discusses the results for hypothesis 1 with respect to efficiency. Figure IV-B1 shows the averaged speeds and their standard deviation. There is hardly any difference between the two virtual environments as confirmed by the Wilcoxon-Mann-Whitney test (W=600, p=0.2). Thus, we must reject our hypothesis 1: There does not seem to be a significant difference between the two IVRS 'desktop' and 'HMD' with respect to efficiency.



(a) Speeds by IVRS

IVRS	Mean	SD	Median
Desktop	0.19	0.16	0.16
HMD	0.26	0.21	0.23

(b) Speeds by group

Group	Mean	SD	Median
1	0.23	0.17	0.21
2	0.21	0.19	0.21

Fig. 6. Efficiency

2) Learning Effects: In Figure 8 and Figure 7 we give an overview of the achieved speeds in the eight tasks. In this representation, it can be seen that the speeds in each group of the first task are lower than in any other later task. Even the switch between the two different IVRS from Task 1-2 to Task 2-1 shows no drop of speed back to the level of the first task in both groups. We conjecture that either the training for the EvoStreet was not yet completed despite statements to the contrary by the participants or that the participants were overwhelmed by the increased size of the EvoStreets for the actual experiment.

3) Speed by Group: By comparing the participants' speeds in the two groups 1 and 2, we examined whether there were any real speed differences that could be attributed solely to the fact that the participants started with a particular kind of IVRS. The Wilcoxon-Mann-Whitney test shows no statistically significant difference between both groups (W=600, p=0.2). Thus, we can rule out that participants generally had a speed advantage only by starting with a particular kind of IVRS.

4) Speed of Gamers/Graphic Artist: This section looks into hypothesis 2 regarding the influence of gaming experience. A frequently described challenge when comparing different virtual environments are previous experiences of the participants in the means of interaction offered by an IVRS. There could also be such advantages in our experiment, especially as we have used an interaction scheme for the desktop system that

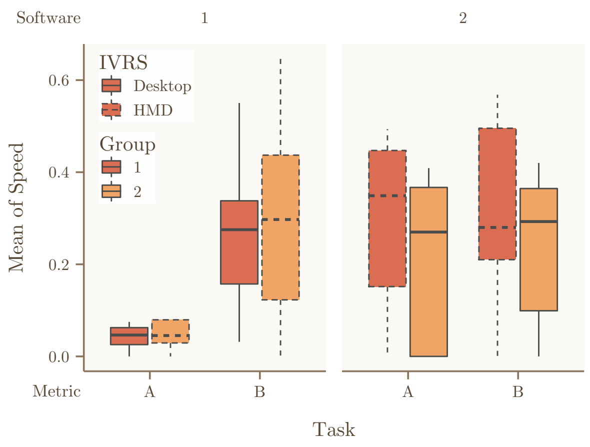


Fig. 7. Efficiency—Mean of speed by group

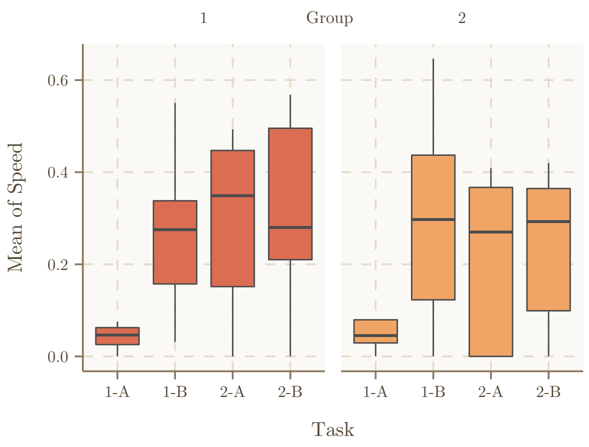


Fig. 9. Efficiency—Mean speeds of all participants in all tasks by group with error indicators

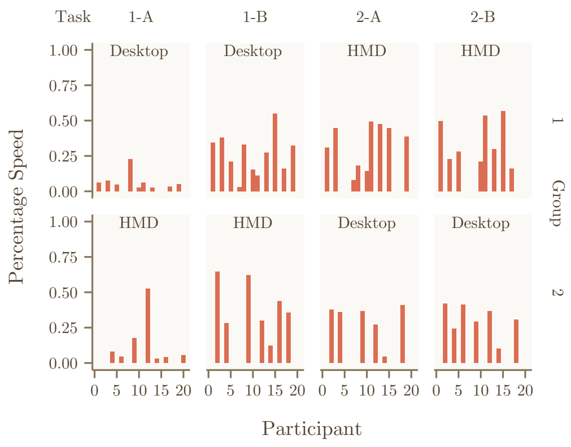


Fig. 8. Efficiency—Speeds of all participants in all tasks by group; upper part shows group 1, lower part shows group 2

is familiar to certain user groups: gamers and graphic artists. In order to be able to better assess possible advantages of computer gamers, we examined whether there are differences for participants who—according to their self-assessment—spend more than a day a week playing desktop computer games.

As expected, we noticed that the speeds in the steps *before* the homing task actually started (i.e., on their path from the starting point to the point where they were expected to return) was a lot higher for computer gamers. Likewise, during the homing task the speeds of the 'gamers' were also statistically significantly higher than the 'non-gamers' (cf. Figure 10a) according to the Wilcoxon-Mann-Whitney test (W=1000, p=0.005). Thus, gamers travel faster.

When assigning participants randomly to the groups, we made sure that they were distributed homogeneously over the two test groups 'desktop' and 'HMD', so that the effects of this previous experience was balanced out.

5) Speed by Metric Mapping: In this section, we turn to hypothesis 3 on the influence of changing metric mappings. The same software project was used twice for the task, where only the metric mapping was changed. Thus, the paired tasks on the same software project differed primarily in the applied metric mapping. If our hypothesis 3 were correct, we should see no difference when comparing Task 1-1 to Task 1-2 and Task 2-1 to Task 2-2 (cf. Table I).

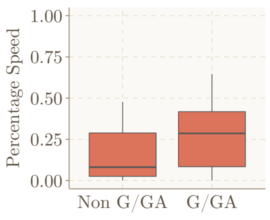
Figure 10b shows the results of this comparison. The participants were faster in the second metric mapping. As already noted above, Task 1-1 is somewhat exceptional as we observed the slowest speeds for it (cf. Figure 8). Yet, even if we compare the speeds for two different metric mappings only in project SW 2, the Wilcoxon-Mann-Whitney test finds no significant differences. From this we conclude that changing only the metric mapping in an EvoStreets graph has no positive effect (neither a negative one) on the spatial orientation or participants in an EvoStreets visualization they have already visited.

C. *User Experience*

After the experiment, we asked the participants to fill out two common user-experience questionnaires. Neither *meCue* (desktop 3.68, HMD 3.35 for the overall evaluation) nor *SUS* (desktop 80.63, HMD 81.25 for total scores) shows any significant difference. The participant showed no preference for one particular kind of IVRS—neither in any specific aspect of the questionnaires nor in the overall summary result.

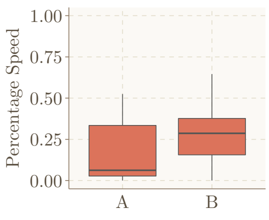
D. *Threats to Validity*

We used convenience sampling to reach out for experimental participants. On the one hand, convenience sampling is speedy,



(a) Speeds by Gamer

Sample	Mean	SD	Median
G	0.28	0.19	0.28
Non-G	0.15	0.15	0.08



(b) Speeds by metric mapping

Metric	Mean	SD	Median
A	0.16	0.18	0.06
B	0.28	0.18	0.29

Fig. 10. Efficiency

easy, readily available, and cost effective. On the other hand, the results cannot necessarily be generalized to the target population consisting of all possible developers due to potential disproportional representation of subgroups in the sample. Another threat may be that the participants might have wanted to do the experimenters a favor. Yet, subjective opinions were collected only in the usability questionnaire. The hypotheses were tested with factual and objectively measured data.

We limited our study to two Java programs and four kinds of metrics. We cannot necessarily generalize our results to other programming languages, programs, and metrics with other characteristics. Our specific rendering of EvoStreets and means for interaction may have had an influence on the results. Other visual mappings and forms of interaction may lead to different results.

V. Conclusion

Software cities help at comprehension tasks, by giving a spatial component as well as a structure, which is supposedly intuitive due to the used metaphor [31]. The comprehension achieved by the use of a software city contains knowledge about the spatial orientation of the city elements. Having in mind that—thanks to technological improvements—it is possible to explore software cities interactively in VR today, we researched whether a change in the IVRS used to explore the software city makes a difference in spatial orientation. We compared two classes of IVRS: HMD and desktop.

To this end, we implemented an Unreal Engine Plugin to visualize graphs that represent software systems rendering them in a configurable 3D scene. We used EvoStreets, which might be easier for orientation tasks than the 3D tree maps, known as code cities [2], thanks to additional space between elements and a more diverse structure. The EvoStreets used in our experiment were explored in a method view, containing several hundreds methods each. Additionally the relations between the methods were shown as hierarchically bundled edges above the roofs.

The orientation tasks our participants were confronted with were homing tasks, where the path through an EvoStreets from a starting point A to a return point D (passing two intermediate points) was the learning phase and the homing back from D to the start A was the measured part.

Our primary hypothesis was that the class of HMDs would make it easier to gain a spatial orientation inside of software cities. We have to reject this hypothesis based on the results of our controlled experiment with 20 participants.

In addition to that, we found that gamers have an advantage in navigation speed (but not in effectiveness), hinting at that interaction skills may have an influence, and overall efficiency may be increased if one learns an interaction well. As a practical implication, we need to give users enough time to familiarize themselves with new types of interaction and visualization.

We found no learning effect when a participant has visited an EvoStreet before, after changing only the metric mapping, which affects the size attributes of buildings, but not the structure of the city. This must be taken into account if developers of 3D code-city tools consider to allow users to adjust the metric mapping and also if the same software project is shown over a period of evolution where such attributes can change. We need to think about means to maintain the mental map even for such non-structural changes.

Acknowledgment

The authors would like to thank all participants of the experiment.

References

[1] B. Johnson and B. Shneiderman, “Tree-maps: A space-filling approach to the visualization of hierarchical information structures,” in *Proceedings of the Conference on Visualization*. IEEE Computer Society Press, 1991, pp. 284–291. [Online]. Available: http://dl.acm.org/citation.cfm?id=949607.949654

[2] R. Wettel and M. Lanza, “Visual exploration of large-scale system evolution,” in *Reverse Engineering, 2008. WCRE'08. 15th Working Conference on*. IEEE, 2008, pp. 219–228.

[3] T. Bladh, D. A. Carr, and M. Kljun, “The effect of animated transitions on user navigation in 3d tree-maps,” in *Information Visualisation, 2005. Proceedings. Ninth International Conference on*. IEEE, 2005, pp. 297–305.

[4] F. Steinbrückner, “Consistent software cities: supporting comprehension of evolving software systems,” Ph.D. dissertation, Brandenburgischen Technischen Universität Cottbus, Cottbus, 06 2013. [Online]. Available: https://opus4.kobv.de/opus4-btu/frontdoor/index/index/docId/1681

[5] M. V. Sanchez-Vives and M. Slater, “From presence to consciousness through virtual reality,” *Nature Reviews Neuroscience*, vol. 6, no. 4, pp. 332–339, 2005.

[6] L. Merino, J. Fuchs, M. Blumenschein, C. Anslow, M. Ghafari, O. Nierstrasz, M. Behrisch, and D. A. Keim, “On the impact of the medium in the effectiveness of 3d software visualizations,” in *Software Visualization (VISSOFT), 2017 IEEE Working Conference on*. IEEE, 2017, pp. 11–21.

[7] F. Fittkau, A. Krause, and W. Hasselbring, “Exploring software cities in virtual reality,” in *Software Visualization (VISSOFT), 2015 IEEE 3rd Working Conference on*. IEEE, 2015, pp. 130–134.

[8] M. Slater, M. Usoh, and A. Steed, “Depth of presence in virtual environments,” *Presence: Teleoper. Virtual Environ.*, vol. 3, no. 2, pp. 130–144, Jan. 1994. [Online]. Available: http://dx.doi.org/10.1162/pres.1994.3.2.130

[9] M. Slater, “Place illusion and plausibility can lead to realistic behaviour in immersive virtual environments,” *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, vol. 364, no. 1535, pp. 3549–3557, 2009. [Online]. Available: http://rstb.royalsocietypublishing.org/content/364/1535/3549

[10] S. S. Chance, F. Gaunet, A. C. Beall, and J. M. Loomis, “Locomotion mode affects the updating of objects encountered during travel: The contribution of vestibular and proprioceptive inputs to path integration,” *Presence*, vol. 7, no. 2, pp. 168–178, 1998.

[11] A. R. Teyseyre and M. R. Campo, “An overview of 3d software visualization,” *IEEE transactions on visualization and computer graphics*, vol. 15, no. 1, pp. 87–105, 2009.

[12] D. Holten, “Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 5, pp. 741–748, 2006.

[13] R. A. Ruddle, S. J. Payne, and D. M. Jones, “Navigating large-scale virtual environments: what differences occur between helmet-mounted and desk-top displays?” *Presence: Teleoperators & Virtual Environments*, vol. 8, no. 2, pp. 157–168, 1999.

[14] R. B. Welch, T. Blackmon, A. Liu, B. Mellers, and L. W. Stark, “The effects of pictorial realism, delay of visual feedback, and observer interactivity on the subjective sense of presence,” *Presence: Teleoperators and Virtual Environments*, vol. 5, pp. 263–273, 06 1996.

[15] C. S. Green, A. Pouget, and D. Bavelier, “Improved probabilistic inference as a general learning mechanism with action video games,” *Current Biology*, vol. 20, no. 17, pp. 1573–1579, Sep 2010. [Online]. Available: http://dx.doi.org/10.1016/j.cub.2010.07.040

[16] Reaction time differences in video game and non-video game players - viewcontent.cgi. [Online]. Available: https://digitalcommons.cwu.edu/cgi/viewcontent.cgi?article=1689&context=source

[17] S. Kühn, T. Gleich, R. C. Lorenz, U. Lindenberger, and J. Gallinat, “Playing super mario induces structural brain plasticity: gray matter changes resulting from training with a commercial video game,” *Molecular Psychiatry*, vol. 19, pp. 265 EP –, Oct 2013, original Article. [Online]. Available: http://dx.doi.org/10.1038/mp.2013.120

[18] L. G. Appelbaum, M. S. Cain, E. F. Darling, and S. R. Mitroff, “Action video game playing is associated with improved visual sensitivity, but not alterations in visual sensory memory,” *Attention, Perception, & Psychophysics*, vol. 75, no. 6, pp. 1161–1167, Aug 2013. [Online]. Available: https://doi.org/10.3758/s13414-013-0472-7

[19] V. R. Bejjanki, R. Zhang, R. Li, A. Pouget, C. S. Green, Z.-L. Lu, and D. Bavelier, “Action video game play facilitates the development of better perceptual templates,” *Proceedings of the National Academy of Sciences*, 2014. [Online]. Available: http://www.pnas.org/content/early/2014/11/05/1417056111

[20] A. V. Berard, M. S. Cain, T. Watanabe, and Y. Sasaki, “Frequent video game players resist perceptual interference,” *PLOS ONE*, vol. 10, no. 3, pp. 1–10, 03 2015. [Online]. Available: https://doi.org/10.1371/journal.pone.0120011

[21] I. Granic, A. Lobel, and R. C. Engels, “The benefits of playing video games.” *American psychologist*, vol. 69, no. 1, p. 66, 2014.

[22] A. Eichenbaum, D. Bavelier, and C. S. Green, “Video games: play that can do serious good,” *American Journal of Play*, vol. 7, no. 1, pp. 50–72, 2014, iD: unige:84313. [Online]. Available: https://archive-ouverte.unige.ch/unige:84313

[23] D. A. Bowman, E. T. Davis, L. F. Hodges, and A. N. Badre, “Maintaining spatial orientation during travel in an immersive virtual environment,” *Presence: Teleoper. Virtual Environ.*, vol. 8, no. 6, pp. 618–631, 1999. [Online]. Available: http://dx.doi.org/10.1162/105474699566521

[24] B. E. Riecke, D. W. Cunningham, and H. H. Bülthoff, “Spatial updating in virtual reality: the sufficiency of visual information,” *Psychological Research*, vol. 71, no. 3, pp. 298–313, May 2007. [Online]. Available: https://doi.org/10.1007/s00426-006-0085-z

[25] J. W. Regian, W. L. Shebilske, and J. M. Monk, “Virtual reality: An instructional medium for visual-spatial tasks,” *Journal of Communication*, vol. 42, no. 4, pp. 136–149, 1992. [Online]. Available: http://dx.doi.org/10.1111/j.1460-2466.1992.tb00815.x

[26] B. Sousa Santos, P. Dias, A. Pimentel, J.-W. Baggerman, C. Ferreira, S. Silva, and J. Madeira, “Head-mounted display versus desktop for 3d navigation in virtual reality: A user study,” *Multimedia Tools and Applications*, vol. 41, no. 1, pp. 161–181, Jan. 2009. [Online]. Available: http://dx.doi.org/10.1007/s11042-008-0223-2

[27] R. A. Ruddle and P. Péruch, “Effects of proprioceptive feedback and environmental characteristics on spatial learning in virtual environments,” *International Journal of Human-Computer Studies*, vol. 60, no. 3, pp. 299 – 326, 2004. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1071581903001733

[28] P. Péruch, M. May, and F. Wartenberg, “Homing in virtual environments: Effects of field of view and path layout,” *Perception*, vol. 26, no. 3, pp. 301–311, 1997.

[29] F. Wartenberg, M. May, and P. Péruch, “Spatial orientation in virtual environments: Background considerations and experiments,” in *Spatial cognition*. Springer, 1998, pp. 469–489.

[30] R. F. Wang, “Between reality and imagination: When is spatial updating automatic?” *Perception & Psychophysics*, vol. 66, no. 1, pp. 68–76, Jan 2004. [Online]. Available: https://doi.org/10.3758/BF03194862

[31] R. Wettel, “Software systems as cities,” Ph.D. Thesis, University of Lugano, Switzerland, 2010.

[32] M. R. Mine, “Virtual environment interaction techniques,” University of North Carolina at Chapel Hill, Chapel Hill, NC, USA, Tech. Rep., 1995.

[33] Y. Guan and M. Zheng, “Real-time 3d pointing gesture recognition for natural hci,” in *2008 7th World Congress on Intelligent Control and Automation*, June 2008, pp. 2433–2436.

[34] M. Nielsen, M. Störring, T. B. Moeslund, and E. Granum, “A procedure for developing intuitive and ergonomic gesture interfaces for hci,” in *International gesture workshop*. Springer, 2003, pp. 409–420.

[35] R. P. McMahan, D. A. Bowman, D. J. Zielinski, and R. B. Brady, “Evaluating display fidelity and interaction fidelity in a virtual reality game,” *IEEE transactions on visualization and computer graphics*, vol. 18, no. 4, pp. 626–633, 2012.

[36] L. E. Nacke, S. Stellmach, D. Sasse, and C. A. Lindley, “Gameplay experience in a gaze interaction game,” *arXiv preprint arXiv:1004.0259*, 2010.

[37] C. Knight and M. Munro, “Virtual but visible software,” in *Information Visualization, 2000. Proceedings. IEEE International Conference on*. IEEE, 2000, pp. 198–205.

[38] T. Panas, R. Berrigan, and J. Grundy, “A 3d metaphor for software production visualization,” in *Information Visualization, 2003. IV 2003. Proceedings. Seventh International Conference on*. IEEE, 2003, pp. 314–319.

[39] T. Panas, T. Epperly, D. Quinlan, A. Saebjornsen, and R. Vuduc, “Communicating software architecture using a unified single-view visualization,” in *Engineering Complex Computer Systems, 2007. 12th IEEE International Conference on*. IEEE, 2007, pp. 217–228.

[40] P. Khaloo, M. Maghoumi, E. Taranta, D. Bettner, and J. Laviola, “Code park: A new 3d code visualization tool,” in *Software Visualization (VISSOFT), 2017 IEEE Working Conference on*. IEEE, 2017, pp. 43–53.

[41] K. Ogami, R. G. Kula, H. Hata, T. Ishio, and K. Matsumoto, “Using high-rising cities to visualize performance in real-time,” in *Software Visualization (VISSOFT), 2017 IEEE Working Conference on*. IEEE, 2017, pp. 33–42.

[42] A. Schreiber and M. Brüggemann, “Interactive visualization of software components with virtual reality headsets,” in *Software Visualization (VISSOFT), 2017 IEEE Working Conference on*. IEEE, 2017, pp. 119–123.

[43] D. Baum, J. Schilbach, P. Kovacs, U. Eisenecker, and R. Müller, “Getaviz: Generating structural, behavioral, and evolutionary views of software systems for empirical evaluation,” in *Software Visualization (VISSOFT), 2017 IEEE Working Conference on*. IEEE, 2017, pp. 114–118.

[44] Axivion GmbH, “Axivion bauhaus suite,” 08.05.2018. [Online]. Available: http://www.axivion.com/

[45] Apache Commons Project, “Apache commons,” 08.05.2018. [Online]. Available: https://commons.apache.org/