HyDE: A Hyper-Display Environment in Mixed and Virtual Reality and its Application in a Software Development Case Study

Roy Oberhauser, Alexandre Matic, and Camil Pogolski
Computer Science Dept.
Aalen University
Aalen, Germany
email: {roy.oberhauser, camil.pogolski}@hs-aalen.de, alexandre.matic@studmail.hs-aalen.de

Abstract - **While Virtual Reality (VR) has been applied to various domains to provide new visualization and interaction capabilities, enabling programmers to utilize VR for their software development and maintenance tasks has been insufficiently explored. In this paper, we present the Hyper-Display Environment (HyDE) in the form of a mixed-reality (HyDE-MR) or virtual reality (HyDE-VR) variant respectively, which provides simultaneous multiple operating system window visualization with integrated keyboard/mouse viewing and interaction using MR or in pure VR via a virtual keyboard. This paper applies HyDE in a software development case study as an alternative to typical non-VR Integrated Development Environments (IDEs), supporting software engineering tasks with multiple live screens in VR as an augmented virtuality. The MR solution concept enables programmers to benefit from VR visualization and virtually unlimited information displays while supporting their more natural keyboard interaction for basic code-centric tasks. Thus, developers can leverage VR paradigms and capabilities while directly interacting with their favorite tools to develop and maintain program code. A prototype implementation is described, with a case study demonstrating its feasibility and an initial empirical study showing its potential.**

Keywords - virtual reality; mixed reality; augmented virtuality; integrated development environments; software engineering; computer-aided software engineering; programming.

I. INTRODUCTION

This paper is an extension of [1], which described our Mixed-Reality Fly-Thru-Code (MR-FTC) approach for visualizing software structures in virtual reality (VR) and supported coding via a virtual tablet and integration of a mixed-reality (MR) keyboard.

As digitalization sweeps across society, the amount of program source code created and maintained worldwide is steadily increasing. Google is said to have at least 2bn lines of code (LOC) accessed by over 25K developers [2], while GitHub has over 79m repositories and 28m developers [3]. It has been estimated that well over a trillion LOC exist with 33bn added annually [4]. This is exacerbated by a limited supply of programmers and high employee turnover rates for software companies, e.g., 1.1 years at Google [5]. This has ramifications on the labor expenses involved in software development and maintenance. Approximately 75% of technical software workers are estimated to be doing maintenance [6]. Moreover, program comprehension may consume up to 70% of the software engineering (SE) effort [7]. As an example, the Year 2000 (Y2K) crisis [8] with global costs of $375-750 billion provided an indicator of the scale and importance of program comprehension. Activities involving program comprehension include investigating functionality, internal structures, dependencies, run-time interactions, execution patterns, and program utilization; adding or modifying functionality; assessing the design quality; and domain understanding of the system [9].

Some of the challenges faced by software developers who are now more than ever often facing unfamiliar preexisting codebases are: 1) effectively and efficiently familiarizing and comprehending the structure and intent of collaboratively developed code, 2) programming and testing code changes, and 3) maintaining (debugging, optimizing, or securing) the code. Yet the tools programmers use to work with this code have not significantly changed over the years. The Jolt Productivity Award for 2015 went to an IDE, the Jetbrains IntelliJ IDEA, and other IDEs, Apple's Xcode and Microsoft's Visual Studio, were finalists [10]. RebelLabs Developer Productivity Report asked what tools developers most used and 3 IDEs were reported (IDEA, Eclipse, NetBeans) [11].

Within the scope of developer's program comprehension and informational challenges, one could hypothesize that the simultaneous access to situationally relevant information by developers is at least in part hampered by current physical limitations for viewability on computer displays. This can be observed by the - not infrequent - use of multiple displays (when space and budget allow), high resolutions (when visibility allows), and the multiple windows and tabs open by developers during their tasks, where quick access to relevant information is critical.

As to possible visual interface solutions, a survey of over 21K developers by SlashData in 2017 showed that 25% of professional game developers were targeting VR or MR headsets [12]. That indicates that a growing segment of software developers are becoming familiar with and have access to these headsets during development, whereby the target environment for which the headsets are intended are gamers and not developers. Thus, it may be viable to instead leverage the opportunities afforded by VR [13] and MR interfaces to support and target software developers during their development and provide more comprehensive information. However, a peculiarity and challenge regarding using VR with software developers in contrast to typical VR users is their affinity to keyboard interfaces when interacting with program code. This would thus typically require frequent (un)fastening of the VR headset to view and utilize the real keyboard - an annoying disruption, visually disconcerting from the immersion experience, and inefficient. Alternatively, virtual keyboards would require selecting one key at a time using the typical VR controllers (akin to one finger typing) or some unfamiliar spin-dial approach and would be an inefficient means for code input for various programmers, since they often require special characters. While data gloves might become a future interface alternative for typing, they have not yet become sufficiently popular.

In our prior work we described our Mixed-Reality Fly-Thru-Code (MR-FTC) approach [1], which provided an immersive software structure visualization and fly-thru experience of code structure dependencies and enabled programmers to view code and make text changes on a virtual tablet using MR for keyboard and mouse access. Here we extend [1] by removing the limitations of the virtual tablet and focusing on enhancing the informational display capabilities in the VR environment with a multi-display and heterogeneous tool source capability while hiding the VR software structure visualization.

This paper contributes the Hyper-Display Environment (HyDE) solution concept, and applies it in a case study to enhance available software developer environment capabilities by leveraging VR to integrate multiple information screens in support of software development and maintenance tasks. As visual IDEs often contain multiple sub-windows with information, HyDE displays an unlimited number of operating system (OS) windows that can contain any tool or information desired. Thus, direct integration support of various IDEs, tools, and information sources is enabled as a type of MR projection into the VR environment. The HyDE-MR variant also integrates real keyboard and mouse/trackpad; the HyDE-VR variant provides a virtual keyboard. The solution concept is sufficiently general to be applicable to any domain desiring simultaneous access to multiple informational screens.

The paper is organized as follows: the next section discusses related work; Section III then describes the solution concept. Section IV provides details about our prototype implementation of the solution concept. In Section V, the evaluation, based on a case study, is described, which is followed by a conclusion.

II. Related Work

Work related to viewing desktop applications in virtual or immersive environments or using hyper displays includes VEWL [14], a library for developing applications projecting windows onto polygons within an immersive virtual environment and provide additional information and controls including menus, windows, and buttons. The user's head and a wand are tracked. CAVE2 [15] is a cylindrical system of 72 passive stereo LCD panels that provide a 320-degree panoramic environment for displaying information, either dedicated to one virtual simulation or having a traditional tiled display wall enabling users to work with large numbers of documents at the same time.

Work related to improving IDEs for SE includes IDE++ [16] is an IDE extension framework and interaction monitor and describes four applications (DevTime, Sage, Proctor, and Localizer) with the intent to make IDEs more intelligent. Code Bubbles [17] attempts to improve the IDE user interface with lightweight editable fragments of code using a bubble metaphor.

VR-related work in support of SE tasks includes software visualization such as Imsovision [18], which visualizes object-oriented software in VR using electromagnetic sensors attached to shutter glasses and a wand for interaction. ExplorViz [19] is a JavaScript-based web application that uses WebVR to support VR exploration of 3D software cities using Oculus Rift together with Microsoft Kinect for gesture recognition. These approaches lack the integration of a keyboard and are thus limited in their ability to support programming.

With regard to MR and augmented reality (AR) support for programming tasks, Tangible Windows [20] provides one open window per tablet and allows the user to switch their application by switching between tablets. Lee et al. [21] describe an approach for authoring tangible augmented reality applications with regard to scenes and object behaviors within the AR application being built, so that the development and testing of the application can be done concurrently and intuitively throughout the development process. However, integration of AR support for non-AR software development is not shown. Billinghurst and Kato [22] show possible concepts for collaboration in VR, but do not depict a keyboard or show how programming task support would work. Neumann et al. [23] do not appear to use VR goggles in augmented reality (AR) for projecting multiple PC screens. In 3d live [24], users view a two-dimensional fiducial marker using a video-see-through augmented reality (AR) interface. Kato and Billinghurst [25] use optical see-through MR, whereby an AR conferencing system was developed that allowed virtual images of remote collaborators to be overlaid on multiple users' real environments. Gupta et al. [26] use a tracking framework, wherein the 3D position of planar pages is monitored as they are turned back and forth by a user, and data is correctly warped and projected onto each page at interactive rates. In each frame, feature points are independently extracted from the camera and projector images and matched in order to recover the geometry of the pages in motion. The book can be loaded with multimedia content, including images, videos, and volumetric datasets.

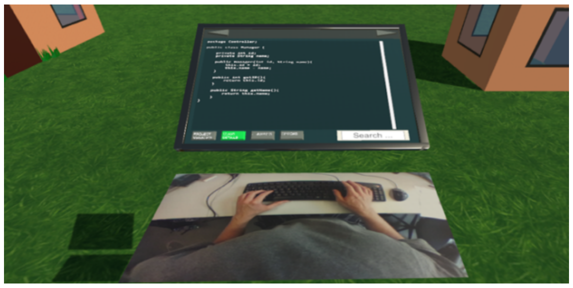


Figure 1. Coding with MR view of keyboard and mouse blended in and scroll bar shown on the virtual tablet.

In our prior MR-FTC approach [1], we visualized software structures using various metaphors and utilized MR with a virtual tablet (called the oracle) to view and edit code for programming tasks, as shown in Figure 1. However, it was limited in functionality and it was not possible to integrate and access other heterogeneous tools.

In contrast to the above work, the HyDE approach leverages VR to enhance simultaneous informational display viewing and interaction, while also supporting basic programming and command-line interface tasks by integrating via MR keyboard and mouse viewing and display interaction in the VR environment.

III. Solution

Our solution concept is domain independent and can be applied to various domains. The information screens displayed are the projections of actual windows from the real operating system and can thus be seen as a form of MR that mixes into the VR environment actual window screens, or more precisely augmented virtuality since the model is mostly virtual and only a relatively small portion is reality. We apply our solution here to the SE or software development area because of its challenges to highlight the solution's potential and capabilities. For instance, to support software developers within the VR environment, our solution concept has an MR variant for integrating keyboard and mouse access which can provide enhanced Graphical User Interface (GUI) and textual interaction for program coding task support.

A. Conceptual Architecture

The GUI from a PC environment can be incorporated in the VR environment projected into the VR landscape of VR goggles or a VR screen (e.g., Google Cardboard) of a VR user, providing a form of MR. As shown in , the operating system (OS) environment (e.g., Windows, Linux, or MacOS) contains various windows or the entire screen of running processes or applications that are viewable ((A-F)).

The VR Environment can incorporate any number of virtual displays of any size placed anywhere in VR space (e.g., a single gigantic virtual display showing all screens (A-F) as separate windows on it) (n-to-1 relationship); correlating screens (1-to-1 relationship); or additional screens that perhaps show historical content or duplicated content (n-to-m relationship). This permits a user to have access to many more screens than physically feasible. Furthermore, various computers limit the number of physical displays that can be attached.

The mechanism to integrate the content of some subset of these windows (or full screen) utilizes available screen capture mechanisms of the operating systems accessed from within the VR Game Engine ((3)). Screenshots are then represented in the VR environment as a texture that can be placed and updated/refreshed (6) on any game object surface, such as one that looks like a virtual display (A-F in the VR Environment Display). These screenshot sequences represent a stream (capture stream) and can be thought to be equivalent to a video stream. A historical view of older screen point in time can be displayed by storing the screenshots and retrieving them (5), permitting time lapse or pausing of screen content. A separate optional Screen Capture Server can be used to set the active window to the foreground and capture the screen (1), and then return it to the background and pass the captured screen image in a place accessible to the VR Game Engine (2).

Interaction with the screens in the VR Environment Display can be supported to effect changes in the PC Environment by eliciting events (7) such as keyboard or mouse events via Input Device mechanisms (e.g., a VR controller, (virtual) keyboard, (virtual) mouse) that can then be transformed and passed on to the OS window (8) as OS events. By utilizing remote desktop applications, one can also view or interact with content across various remote operating system GUIs or windows from within the VR environment.

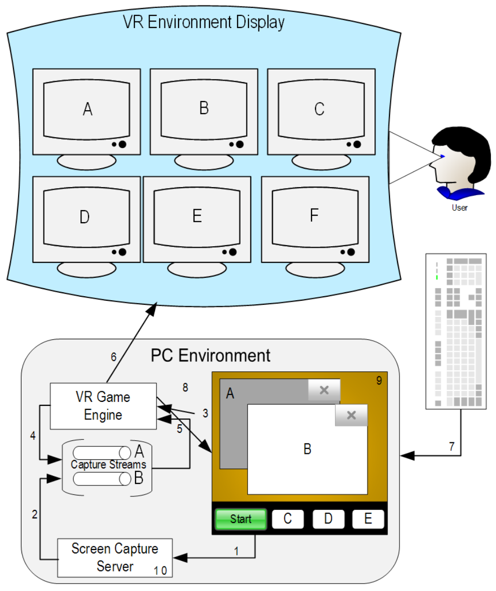


Figure 2. HyDE conceptual architecture.

For the MR variant, a live camera view of the keyboard and mouse can be integrated into the VR landscape. This allows the user to determine where their hands and fingers are relative to the actual hardware. Assuming the subject is seated, for instance, tilting their head down is interpreted as a gesture to activate a live webcam on the VR headset, activating MR mode, similar to the natural head movement made on a desktop PC to look at the keyboard.

B. Process

As shown in Figure 3, the process used by the solution approach can involve the following steps.

For the Window Capture Server:

1) Continually iterate over all processes with GUI windows.

2) Place the window in the foreground.

3) Capture an image of the window using the OS. This can alternatively be an image capture of the screen and then cropped to the foreground window. For the special case of the desktop, the entire screen is captured.

4) Place the image in the image capture stream queue. This can also be persisted if historical records are desired.

5) Return the foreground window to the background.

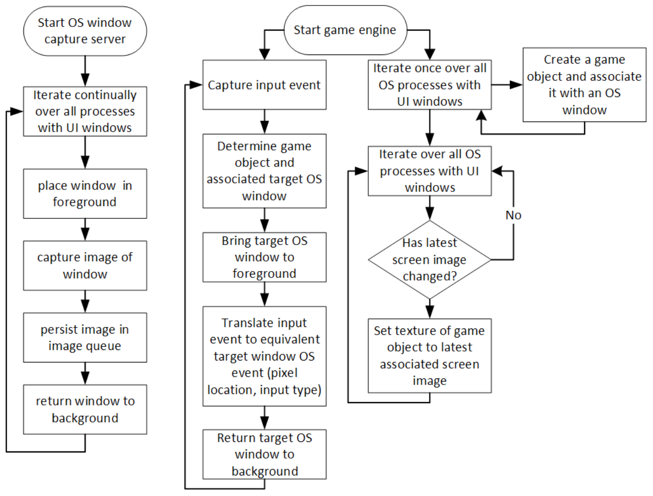

Start OS window capture server
Iterate continually over all processes with UI windows
place window in foreground
capture image of window
persist image in image queue
return window to background
Start game engine
Capture input event
Determine game object and associated target OS window
Bring target OS window to foreground
Translate input event to equivalent target window OS event (pixel location, input type)
Return target OS window to background
Iterate once over all OS processes with UI windows
Create a game object and associate it with an OS window
Iterate over all OS processes with UI windows
Has latest screen image changed?
No
Set texture of game object to latest associated screen image

Figure 3. HyDE process steps.

The steps involved in the game engine process for displaying the screen information are as follows:

1) To create a display in VR, a game object is created and associated to a selected OS window process.

2) In a continual iteration, the texture of any display (a game object) is replaced by the image from that window's capture stream (equivalent to a refresh). As a potential optimization, if no change to the image is detected, no update is required. If the stream is paused, no refresh is invoked. On continuation, either time lapse display (historical playback) or current (discarding all images except the latest) are possible. From a persisted capture stream, any timestamp available can be displayed. This image is overlaid with the mouse pointer and cursor position.

The steps involved in the game engine process to support screen interaction are as follows:

1) For the active display, the associated target OS window is determined

2) The window is brought to the foreground.

3) The input event (mouse or keyboard) is applied to this window

4) The window is returned to the background.

The ongoing capturing process takes care of updating the screen for any resulting changes.

IV. IMPLEMENTATION

A prototype was implemented to determine the feasibility of the solution approach. The Unity game engine 2017.3.0b9 was utilized for the visualization due to its multi-platform support, VR integration, and popularity. Blender 2.79 was used to develop all models. For VR hardware we used HTC Vive, a room scale VR set with a head-mounted display with an integrated camera and two wireless handheld controllers tracked using two 'Lighthouse' base stations.

A. Mixed Reality Variant

For MR, we integrated a live camera view into the VR landscape via a virtual plane object. For a better picture, a Logitech C920 webcam with a 1080p resolution was used instead of the Vive Front camera and a backlit keyboard Corsair K70 RGB Lux. Figure 4 shows the MR setup.

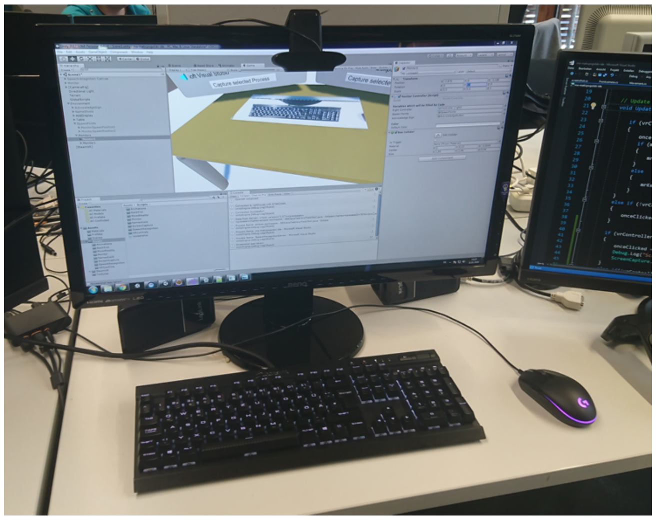


Figure 4. MR setup.

As shown in Figure 5, this allows the user to determine where their hands and fingers are relative to the actual keyboard.

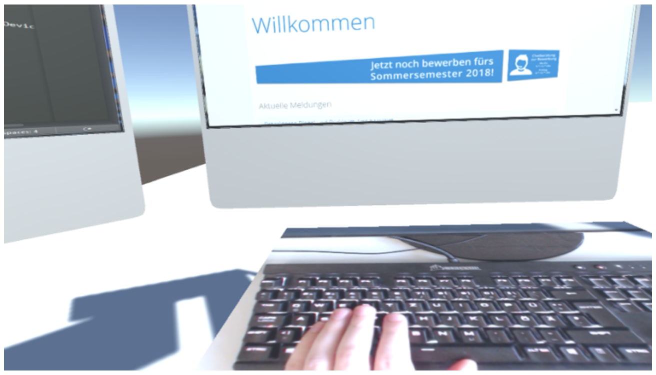


Figure 5. MR, showing real keyboard in cutout of VR desk and displays.

Figure 6 shows a close-up view as to the readability of the keys on the keyboard, with a part of the monitor shown at the top (the webcam view could of course be adjusted to not show the monitor). One advantage of the MR view versus the VR keyboard variant is that the user can utilize their favorite keyboard and mouse that they are already accustomed to.

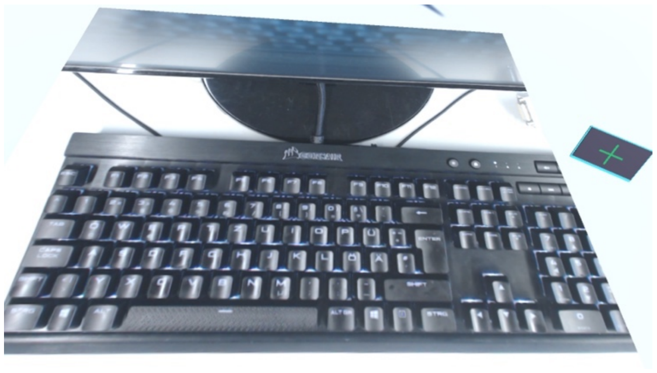


Figure 6. HyDE-MR variant closeup view of the keyboard.

To avoid distraction, it can be configured to hide the MR view as follows: when the VR user tilts their head down, it is interpreted as a gesture to activate the live webcam on the VR headset and blend this into the virtual plane object. When the head is tilted starkly up, MR is deactivated.

B. Virtual Reality Variant

As shown in Figure 7, the HyDE-VR variant avoids MR, requiring the selection of all key and mouse control inputs on any screen be done via the VR controllers.

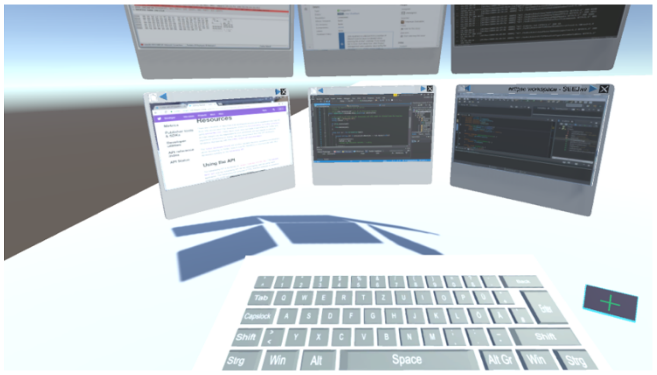


Figure 7. HyDE-VR variant showing a VR (instead of the MR) keyboard.

While data gloves might provide a better option for typing in VR, they are not yet in widespread use within the VR community. We thus opted at this time for practical variants, the MR utilizing any already available keyboard and mouse or the pure VR variant relying on VR controllers only.

C. Display Placement and Interaction

The left controller touchpad allows one to move forwards and backwards (like zoom in or out relative to the displays) and left and right. The right controller touchpad allows one to move up or down. The left controller pointer and trigger can be used to select a display and move it to another position (swap). By pointing and triggering with the right controller on a display, one can freeze (unfreeze) it if one wishes to pause (continue) the capturing (see Figure 8).

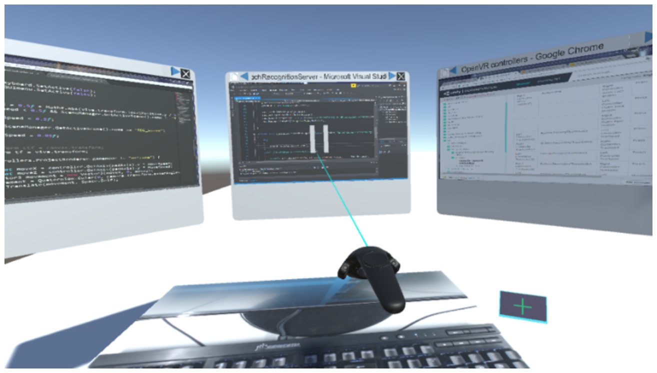


Figure 8. HyDE-MR variant showing pause of capture stream.

New displays are created by pressing the '+' button near the keyboard (see Figure 6 or Figure 7) using the VR controller laser pointer (seen in Figure 9). Placement of the displays is as follows: the first display is placed in a fixed position in front of the keyboard. Further displays are placed relative to the first, the second and third are placed to the left and right of the first at a 45° vertical angle. The fourth through sixth are placed on a second higher row and tilted forward for a better viewing angle without having to move the VR camera position. A bottom third row is placed for the seventh through ninth in a tilted backward manner.

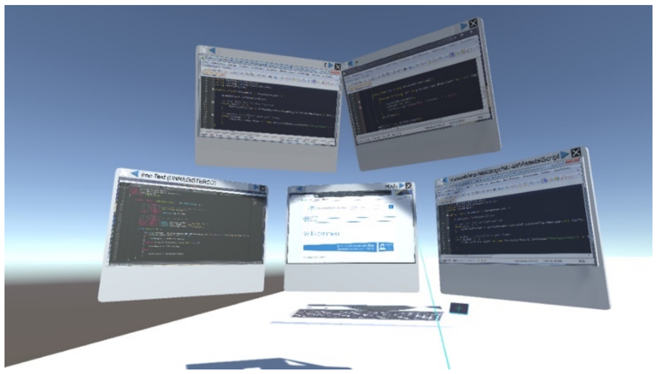


Figure 9. Developer MR desktop setup, with Visual Studio shown on left virtual display, and Eclipse IDE on right virtual display.

Once the tenth display is reached, a stacking configuration of rows of three displays is applied with all rows stacked vertically (no tilting) and the left and right columns angled at 45° without tilting (see Figure 10). By moving relative to this stack the view is adjusted to bring the display of interest up or down and in front to the best viewing angle.

Arrows along the top edge allow one to scroll through the circular list of processes to pick the one to show on that display (see Figure 11). The one to pick is done by pointing the right controller at the process name text and selecting it with the trigger (the process name scrolls when the string is too long for the available space).

As shown in Figure 12, the displays can be closed via an 'x' button on their upper right corner.

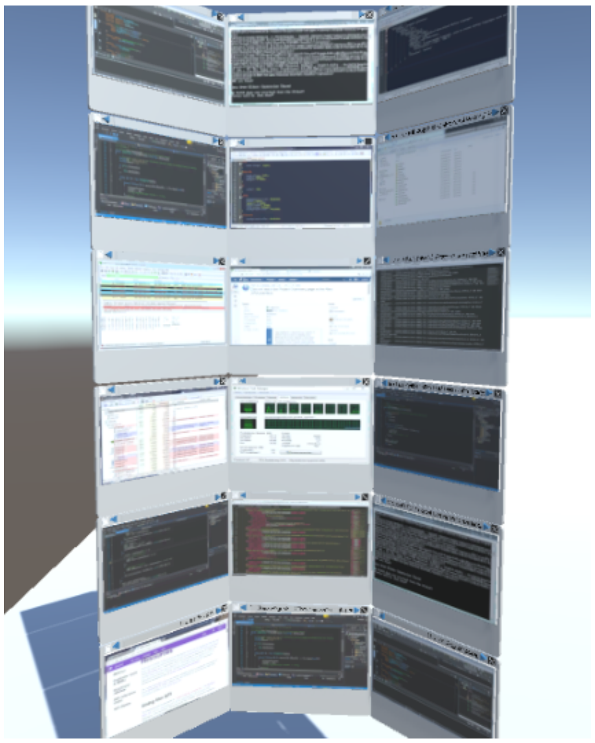


Figure 10. HyDE display stacking.

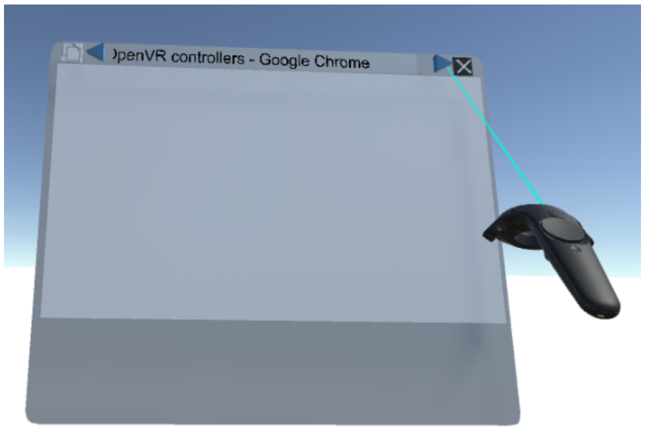


Figure 11. Selecting a process name.

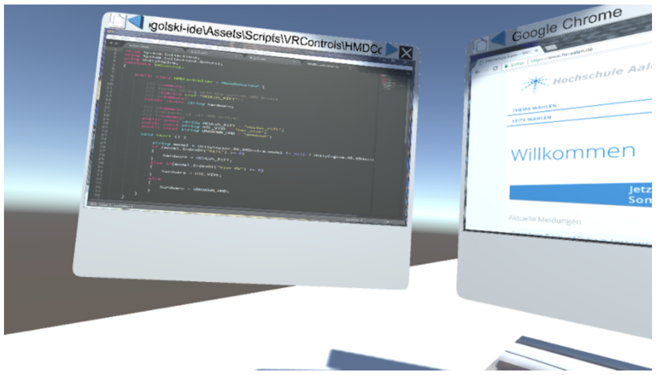


Figure 12. Closeup of a live IDE on virtual display.

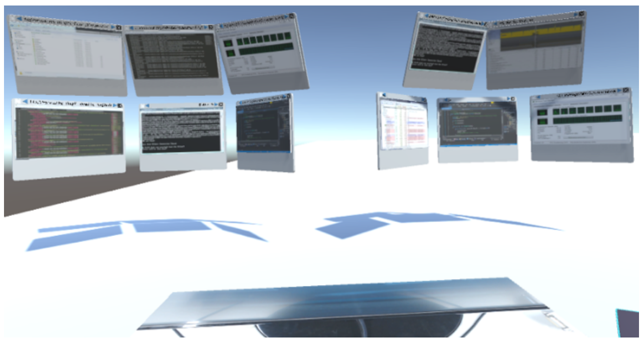


Figure 13. HyDE situationally-related multi-area concept.

A new Area (a group of situationally-relevant displays) can be created by moving sufficiently away from the current Area and pressing the create display button. Figure 13 shows two areas. For example, one area could be focused on fixing a bug from a previous release, while another is focused on optimizing a performance issue, or another is focused on developing some new feature

In case the desired process is not yet running, a special 'null' process can be selected, which will display the desktop so the user can start the desired application, at which point the user can select the process name.

D. *Implementation Details*

The Server is programmed in C# using .NET 4.7 and runs on Microsoft Windows. The Server holds all pertinent information about each running process with an open window.

Unity acts as a client and binds via TCP-IP with the Server to retrieve all process information in XML. Information provided are: process name, process ID, coordinates of the top left position of the open window, window width and height (in pixels), and the window handle. If a process name is selected, the window is captured. If a window is moved, the handle can be used to determine its new position.

Capturing of a process window is done by making the process window the active one (in the foreground) and then capturing the entire screen and then this image is then cropped to the known coordinates of the desired window. This is done to support displaying the cursor. Using a variable of type CURSERINFO, information about the cursor position can be retrieved and its position drawn as an Icon and overlaid on the cropped window picture when a cursor is in that window. Each captured window is handled by a separate thread. For each window the mouse cursor is tracked so it displays the appropriate position for that screen.

V. Evaluation

The evaluation of the HyDE solution concept consists of a case study and empirical study in software development.

A. *Case Study*

In this case study, we take a typical SE maintenance situation where a software developer is attempting to address a bug report that is related to a distributed application. Note that we exemplify this fictional case with analogous screens that do not directly relate to each other nor show actual bug information, they are however from actual live screens.

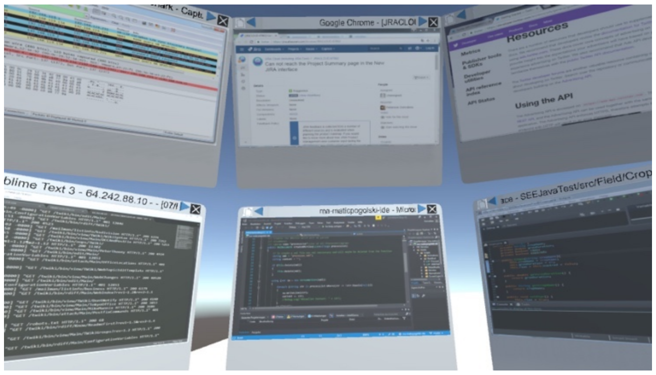


Figure 14. Closeup of a live IDE on virtual display.

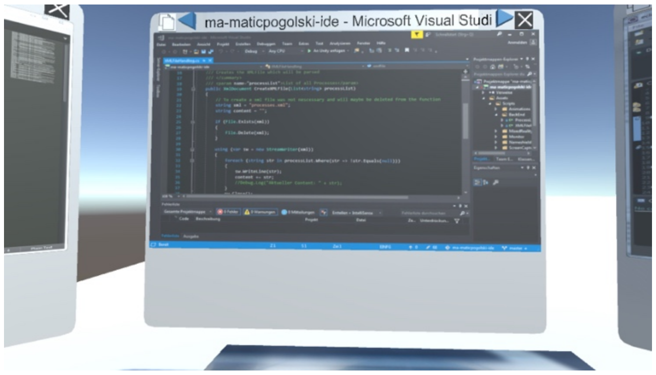


Figure 15. Closeup of a live Visual Studio IDE on virtual display.

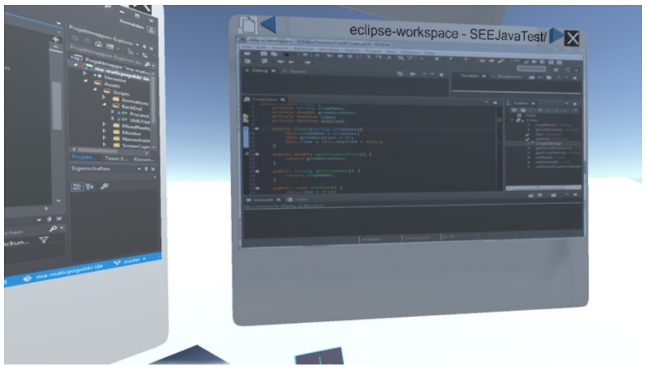


Figure 16. Closeup of a live Eclipse IDE on virtual display.

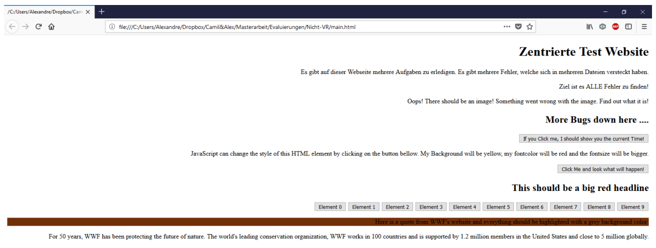
Figure 14 shows the bug report information in the Google Chrome web browser in the center display of the upper row. Perhaps it is unclear if the .NET client is not making a network-based Representational State Transfer (REST) call to the backend Java-based service, or if the service did not return. For this, related information might be to analyze certain log files to see if there are any warnings or errors. Figure 14 shows the bottom left screen with a log file opened in the Sublime Text editor. The client may have been written in a Common Language Infrastructure (CLI) language and for which Visual Studio might be an applicable IDE, shown in the bottom center display. The documentation for the backend service web application programming interface (API) may need to be consulted to familiarize oneself with the call and parameters specifications, shown in the upper right display. Debugging of the backend service, which is written in Java, is done using another IDE, in this case Eclipse shown on the bottom right display. To actually monitor the network packets, the Wireshark tool might be used, shown in the upper left display.

Figure 15 shows a close-up of the Visual Studio IDE, while Figure 16 shows a close-up of the Eclipse IDE.

B. Empirical Evaluation

For an empirical evaluation, for our experiment design we chose to compare subject performance for program debugging tasks using HyDE-MR. The interface type (VR-based HyDE vs. a non-VR notebook) is an independent variable and effectiveness and efficiency are dependent variables. A convenience sample of seven Computer Science (CS) students and one Information Systems (IS) student, who were either in their senior year or master students, was selected with only one unfamiliar with VR. Only one indicated not personally using a multi-monitor setup.

The experiment was supervised and a brief training to show how to utilize HyDE functionality was initially given. Due to only having access to one (heavily shared) VR station and subject constraints, we chose to create an SE task set that required a maximum of 60 minutes per subject (30 minutes maximum in VR) and was basic enough that students across various semesters could do it. We intentionally injected 8 errors into a webpage consisting of HTML, Cascading Style Sheets (CSS), and JavaScript errors (e.g., text that indicates it should be centered but is not). The intended functionality was documented directly in the webpage and it was up to the subject to analyze the text and determine if the webpage was functioning properly for that requirement, and if not, to indicate that a defect was found and then correct it. One set of errors were made for normal monitor (non-VR) usage scenario (see Figure 17), and another set for the VR usage scenario (see Figure 18). After correction, the non-VR webpage should like Figure 19 while the VR webpage should like Figure 20. In the non-VR scenario, the subjects had access to one display but could use multiple windows. In the VR scenario, we observed them creating and using 4-5 displays.



/C:/Users/Alexandre/Dropbox/Cam

file:///C:/Users/Alexandre/Dropbox/Camil&Alex/Masterarbeit/Evaluierungen/Nicht-VR/main.html

Zentrierte Test Website

Es gibt auf dieser Webseite mehrere Aufgaben zu erledigen. Es gibt mehrere Fehler, welche sich in mehreren Dateien versteckt haben.

Ziel ist es ALLE Fehler zu finden!

Oops! There should be an image! Something went wrong with the image. Find out what it is!

More Bugs down here ....

If you Click me, I should show you the current Time!

JavaScript can change the style of this HTML element by clicking on the button bellow. My Background will be yellow, my fontcolor will be red and the fontsize will be bigger.

Click Me and look what will happen!

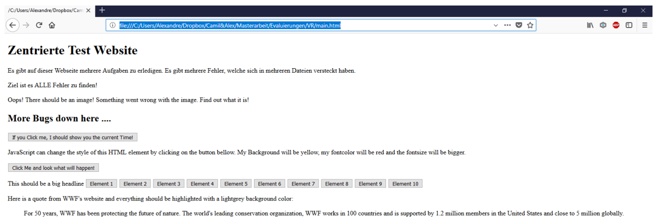
This should be a big red headline

Element 0 Element 1 Element 2 Element 3 Element 4 Element 5 Element 6 Element 7 Element 8 Element 9

Here is a quote from WWF's website and everything should be highlighted with a grey background color:

For 50 years, WWF has been protecting the future of nature. The world's leading conservation organization, WWF works in 100 countries and is supported by 1.2 million members in the United States and close to 5 million globally.

Figure 17. For non-VR: webpage with injected defects.


/C:/Users/Alexandre/Dropbox/Cam
file:///C:/Users/Alexandre/Dropbox/Camil&Alex/Masterarbeit/Evaluierungen/VR/main.html
If you Click me, I should show you the current Time!
Click Me and look what will happen!
Element 1
Element 2
Element 3
Element 4
Element 5
Element 6
Element 7
Element 8
Element 9
Element 10

Zentrierte Test Website

Es gibt auf dieser Webseite mehrere Aufgaben zu erledigen. Es gibt mehrere Fehler, welche sich in mehreren Dateien versteckt haben.

Ziel ist es ALLE Fehler zu finden!

Oops! There should be an image! Something went wrong with the image. Find out what it is!

More Bugs down here ....

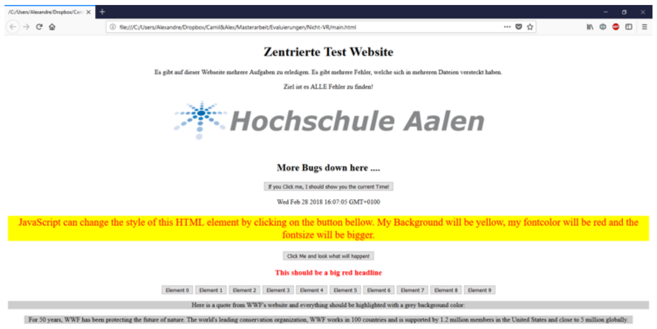
JavaScript can change the style of this HTML element by clicking on the button bellow. My Background will be yellow, my fontcolor will be red and the fontsize will be bigger.

This should be a big headline

Here is a quote from WWF's website and everything should be highlighted with a lightgrey background color:

For 50 years, WWF has been protecting the future of nature. The world's leading conservation organization, WWF works in 100 countries and is supported by 1.2 million members in the United States and close to 5 million globally.

Figure 18. For VR: webpage with injected defects.


/C:/Users/Alexandre/Dropbox/Cam
file:///C:/Users/Alexandre/Dropbox/Camil&Alex/Masterarbeit/Evaluierungen/Nicht-VR/main.html
Hochschule Aalen
If you Click me, I should show you the current Time!
Click Me and look what will happen!
Element 0
Element 1
Element 2
Element 3
Element 4
Element 5
Element 6
Element 7
Element 8
Element 9

Zentrierte Test Website

Es gibt auf dieser Webseite mehrere Aufgaben zu erledigen. Es gibt mehrere Fehler, welche sich in mehreren Dateien versteckt haben.

Ziel ist es ALLE Fehler zu finden!

More Bugs down here ....

Wed Feb 28 2018 16:07:05 GMT+0100

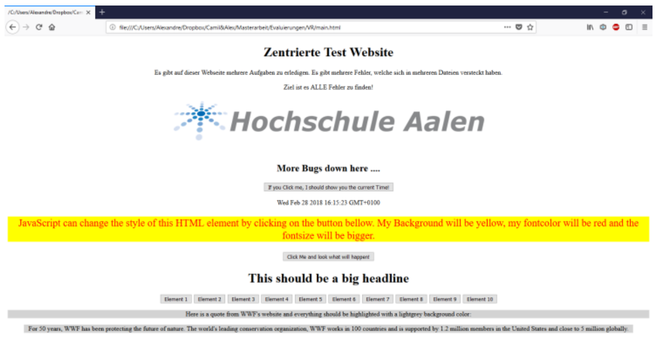
JavaScript can change the style of this HTML element by clicking on the button bellow. My Background will be yellow, my fontcolor will be red and the fontsize will be bigger.

This should be a big red headline

Here is a quote from WWF's website and everything should be highlighted with a grey background color:

For 50 years, WWF has been protecting the future of nature. The world's leading conservation organization, WWF works in 100 countries and is supported by 1.2 million members in the United States and close to 5 million globally.

Figure 19. For non-VR: corrected webpage.


/C:/Users/Alexandre/Dropbox/Cam
file:///C:/Users/Alexandre/Dropbox/Camil&Alex/Masterarbeit/Evaluierungen/VR/main.html
Hochschule Aalen
If you Click me, I should show you the current Time!
Click Me and look what will happen!
Element 1
Element 2
Element 3
Element 4
Element 5
Element 6
Element 7
Element 8
Element 9
Element 10

Zentrierte Test Website

Es gibt auf dieser Webseite mehrere Aufgaben zu erledigen. Es gibt mehrere Fehler, welche sich in mehreren Dateien versteckt haben.

Ziel ist es ALLE Fehler zu finden!

More Bugs down here ....

Wed Feb 28 2018 16:15:23 GMT+0100

JavaScript can change the style of this HTML element by clicking on the button bellow. My Background will be yellow, my fontcolor will be red and the fontsize will be bigger.

This should be a big headline

Here is a quote from WWF's website and everything should be highlighted with a lightgrey background color:

For 50 years, WWF has been protecting the future of nature. The world's leading conservation organization, WWF works in 100 countries and is supported by 1.2 million members in the United States and close to 5 million globally.

Figure 20. For VR: corrected webpage.

We asked them for a self-assessment of their HTML, CSS, and Java Script competency, the results of which are shown relative to their performance in Figure 21. To determine if the task order of VR or non-VR affected the results distinctly, each subject was randomly assigned to either begin with VR or with non-VR, and subjects 1, 5, and 8 initially began without VR. The total time needed until self-indicated completion was measured in seconds, and then this total duration was divided by the number of errors found (and corrected) to get an average duration per error for VR and non-VR separately. This is also shown in Figure 21. When comparing these average durations with the competency self-assessment, we see some correlation, with subject 1 who indicated little to no experience having some of the longest times, with subject 3 and 5 that have some experience have the second cluster of longer times, and subjects 2, 7, and 8 indicating the most competency and having some of the shortest times. Note that subject 1 was unfamiliar with VR, and unfamiliarity with this new environment may also have negatively affected this longest VR time; some prior training sessions may have reduced the duration. However, since the same subject is being compared in the two modes, the subject's programming and debugging experience and competency self-assessment should not significantly impact their own performance relative to themselves. Rather, the independent variable will likely have the greater effect.

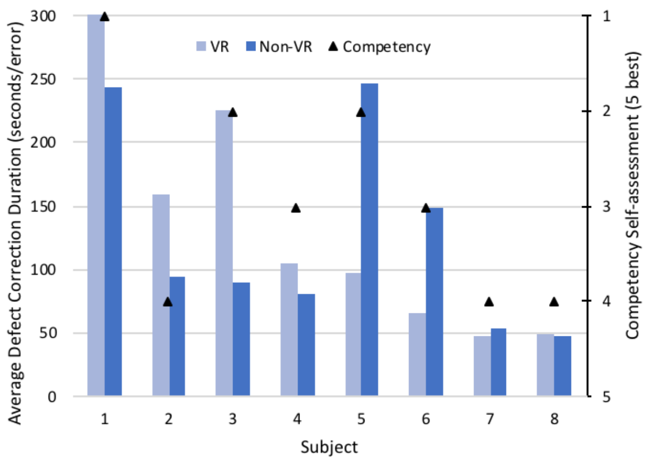


Figure 21. Competency self-assessment and average defect correction duration results for each subject in both the VR and non-VR setting.

As to task training or familiarity effects, as seen in Figure 21 the three subjects that began with non-VR tasks (1, 5, and 8) were slower on VR, faster on VR, and equivalently fast respectively. The other subjects also showed no bias trend with one or the other always faster. Thus, we conclude that there is little to no side effect as to task training in their ability to complete the task faster in the second environment.

TABLE I. HYDE-MR EFFICIENCY AND EFFECTIVENESS IMPROVEMENT VERSUS NON-VR

Subject	Experience[a]	HyDE-MR Improvement	
		Efficiency	*Effectiveness*
1	1	-23%	-13%
2	4	-69%	0%
3	2	-154%	0%
4	3	-30%	13%
5	2	60%	0%
6	3	56%	13%
7	4	11%	0%
8	4	-2%	0%
Average		-9%	13%

a. Scale of 1-5 (5 best)

Table I shows the performance difference of HyDE-MR relative to non-VR for the dependent variables efficiency and effectiveness. On average VR was overall 9% slower (97 seconds) and effectiveness was improved by 13%. However, these are only slight variations Conceivably this might could be attributed to the subjects having less total experience in

VR or to the margin of error given the relatively small population sample size. We also assume that because of the relatively short sessions of less than 30 minutes, that no significant performance impacts are attributed to task fatigue. Subjects 7 and 8 show almost no performance difference with HyDE, whereas for subjects 5 and 6 performance improved, and for 1-4 performance was mostly worse.

In the non-VR scenario, the subjects had access to one display but could use multiple windows. In the VR scenario, we observed them creating and using 4-5. After completing the tasks, subjects were debriefed as to how intuitive, suitable, and enjoyable HyDE was based on a Likert scale of 1 to 5, with 5 being very high and 1 being very low. The results are shown in Figure 22, where all the results were either positive or neutral (except one regarding intuitiveness). This indicates that the HyDE solution approach had a positive or neutral effect on subjective factors. None of the subjects reported VR sickness symptoms, a type of visually-induced motion sickness exhibiting disorientation [27], despite the inclusion of MR keyboard and mouse in VR and ongoing multiple display refresh. We also asked how many VR displays they would consider utilizing in they were given the opportunity, and their preference is shown in Figure 23, where we see that they could see themselves using up to 7 or 9 displays, whereas for non-VR the majority would use 3 real monitors if they could. This indicates that the subjects understood the potential of HyDE. When asked what environment they preferred, 75% preferred the VR over the classic non-VR environment (see Figure 24). Even those who preferred the classic environment liked the HyDE solution concept and prototype and think it has potential.

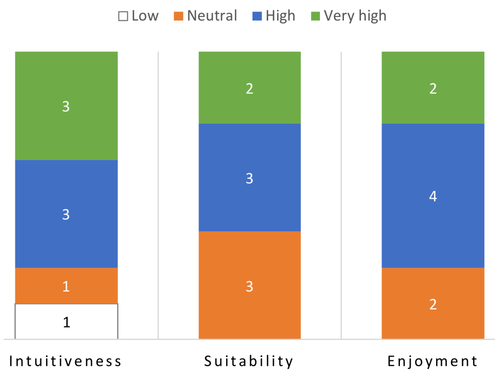


Figure 22. Rating given by the number of subjects for intuitiveness, suitability, and enjoyment of the HyDE prototype.

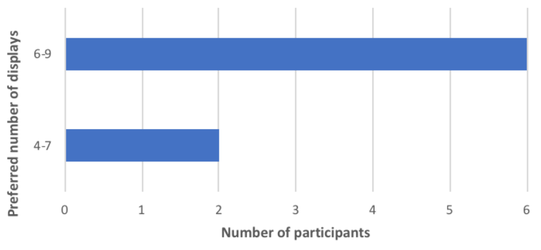


Figure 23. Preferred number of VR displays.

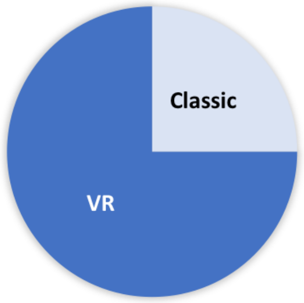


Figure 24. Subjects' preferred environment after the experiment.

As to interpreting the results, a convenience sample can obviously contain a number of biases, including under- or overrepresentation. The motivation of each individual at any point in time to find and fix a defect is another unknown factor, and we provided no reward system which gamification could induce. The differences observed in efficiency and effectiveness between HyDE on non-VR are in our opinion negligible, and we weight their subjective responses and the 75% preference of VR-based HyDE as an initial indicator of support for our solution approach or at a minimum an openness to utilizing VR for software development tasks. Further investigation is still needed.

VI. Conclusion

As VR devices become ubiquitous, it is only a matter of time before programmers wish to utilize VR capabilities for their development tasks as well. Our HyDE prototype demonstrated that the HyDE-VR and HyDE-MR hyper-display solution approach is feasible and can be a viable alternative to desktop displays. The HyDE-MR variant enables touch typing and the use of the mouse for screen interaction where appropriate, enabling programmers to interact more naturally for their code-centric programming tasks in the VR environment while remaining immersed. They thus can avoid interrupting their VR experience to take of the goggles and do programming changes and then put the VR gear on again. The HyDE concept is also generalized such that it can be applied to various domains beyond software development requiring simultaneous viewing and/or interaction of informational screens. The domain software development was selected to show HyDE's potential in intensive informational screen settings where software and tool preferences are non-uniform and support of and access to a large spectrum of software is imperative.

The evaluation based on a case-study and an empirical evaluation showed that the effectiveness in finding bugs was on par (1 bug more for one person), and although the sample size was small, the average task efficiency in VR was only slightly worse (-9% or 97 seconds per defect), which can be considered to be within the margin of error given the subjects' first use of this environment, their competency level for fixing these types of defects, and the sample size.

Future work includes a comprehensive empirical study including industrial usage, the inclusion of additional features, and performance optimizations.

References

[1] R. Oberhauser, “Immersive Coding: A Virtual and Mixed Reality Environment for Programmers,” In: Proceedings of The Twelfth International Conference on Software Engineering Advances (ICSEA 2017), IARIA XPS Press, 2017, pp. 250-255.

[2] C. Metz, *Google Is 2 Billion Lines of Code—And It’s All in One Place*. http://www.wired.com/2015/09/google-2-billion-lines-codeand-one-place/ [retrieved 2018.02.28]

[3] GitHub. https://github.com [retrieved 2018.02.28]

[4] G. Booch, "The complexity of programming models," Keynote talk at AOSD 2005, Chicago, IL, Mar. 14-18, 2005.

[5] PayScale, Full List of Most and Least Loyal Employees. http://www.payscale.com/data-packages/employee-loyalty/full-list [retrieved 2018.02.28]

[6] C. Jones, "The economics of software maintenance in the twenty first century," Version 3, 2006. [Online]. Available from: http://www.compaid.com/caiinternet/ezine/capersjones-maintenance.pdf 2017.02.23

[7] R. Minelli, A. Mocci, and M. Lanza, "I know what you did last summer: an investigation of how developers spend their time," Proc. IEEE 23rd International Conference on Program Comprehension, IEEE Press, 2015, pp. 25-35.

[8] L. Kappelman, "Some strategic Y2K blessings," Software, IEEE, 17(2), 2000, pp. 42-46.

[9] M.J. Pacione, M. Roper, and M. Wood, “A novel software visualisation model to support software comprehension,” Proc. 11th IEEE Working Conference on Reverse Engineering. IEEE, 2004, pp. 70-79.

[10] Dr. Dobb’s. Jolt Awards 2015: Coding Tools. http://www.drdobbs.com/joltawards/jolt-awards-2015-coding-tools/240169420 [retrieved 2018.02.28]

[11] RebelLabs Developer Productivity Report 2017

[12] S. Schuermans, M. Wilcox, L. Hecht, and C. Voskoglou. Developer Economics: State of the Developer Nation Q3 2017. SlashData, 2017.

[13] J. Steuer, “Defining virtual reality: Dimensions determining telepresence,” Journal of communication, 42(4), 1992, pp.73-93.

[14] D. Larimer and and Bowman, “VEWL: A Framework for Building a Windowing Interface in a Virtual Environment,” Proc. of Int. Conf. on Human-Computer Interaction Interact '2003. IOS Press, 2003, pp. 809-812.

[15] A. Febretti et al., “CAVE2: a hybrid reality environment for immersive simulation and information analysis,” The Engineering Reality of Virtual Reality, Vol. 8649 [864903]. International Society for Optics and Photonics, 2013.

[16] Z. Gu, D. Schleck, E.T. Barr, and Z. Su, “Capturing and exploiting IDE interactions,” Proceedings of the 2014 ACM International Symposium on New Ideas, New Paradigms, and Reflections on Programming & Software. ACM, 2014, pp. 83-94.

[17] A. Bragdon et al., “Code bubbles: rethinking the user interface paradigm of integrated development environments,” Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 1. ACM, 2014, pp. 455-464.

[18] J. I. Maletic, J. Leigh, and A. Marcus, “Visualizing software in an immersive virtual reality environment,” 23rd Intl. Conf. on Softw. Eng. (ICSE 2001) Vol. 1, IEEE, 2001, pp. 12-13.

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

[20] Diehl, J. and Borchers, J., Tangible Windows. Technical report, RWTH Aachen University, 2013. URL http://hci. rwth-aachen. de/diehl.

[21] G.A. Lee, D. Nelles, M. Billinghurst, and G.J.Kim, "Immersive authoring of tangible augmented reality applications," Proc. of the 3rd IEEE/ACM international Symposium on Mixed and Augmented Reality, IEEE Computer Society, 2004, pp. 172-181.

[22] M. Billinghurst and H. Kato, "Collaborative mixed reality," Proc. First International Symposium on Mixed Reality (ISMR ’99), Springer Verlag, 1999, pp. 261-284.

[23] U. Neumann, S. You, J. Hu, B. Jiang, and J.W. Lee, "Augmented virtual environments (ave): Dynamic fusion of imagery and 3d models," IEEE Proc. Virtual Reality. IEEE, 2003, pp. 61-67.

[24] S. Prince, A.D. Cheok, F. Farbiz, T. Williamson, N. Johnson, M. Billinghurst, and H. Kato. "3d live: Real time captured content for mixed reality," Proc. International Symposium on Mixed and Augmented Reality (ISMAR 2002). IEEE, 2002, pp. 7-317.

[25] H. Kato and M. Billinghurst, "Marker tracking and hmd calibration for a video-based augmented reality conferencing system," Proc. 2nd IEEE and ACM International Workshop on Augmented Reality (IWAR'99). IEEE, 1999, pp. 85-94.

[26] Gupta, Shilpi, and Christopher Jaynes. "The universal media book: tracking and augmenting moving surfaces with projected information." In Proceedings of the 5th IEEE and ACM International Symposium on Mixed and Augmented Reality, pp. 177-180. IEEE Computer Society, 2006.

[27] R.S. Kennedy, K.M. Stanney, and W.P. Dunlap, “Duration and exposure to virtual environments: sickness curves during and across sessions,” Presence: Teleoperators & Virtual Environments, 9(5), 2000, pp. 463-472.