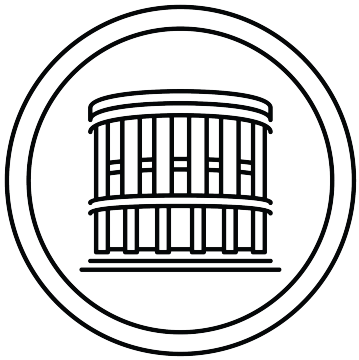
COMENIUS UNIVERSITY IN BRATISLAVA
FACULTY OF MATHEMATICS PHYSICS AND INFORMATICS

SOFTWARE DEVELOPMENT AND EXTENDED REALITY

Master thesis

2022 Bc. Matej Budoš

COMENIUS UNIVERSITY IN BRATISLAVA
FACULTY OF MATHEMATICS PHYSICS AND INFORMATICS



SOFTWARE DEVELOPMENT AND EXTENDED REALITY

Master thesis

Study program:	Applied informatics
Branch of study:	Applied informatics
Department:	Department of Applied Informatics
Supervisor:	doc. Ing. Ivan Polášek, PhD.
Consultant:	Ing. Juraj Vincur, PhD.

Bratislava, 2025 Bc. Matej Budoš

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Bc. Matej Budoš
Študijný program: aplikovaná informatika (Jednoodborové štúdium, magisterský II. st., denná forma)
Študijný odbor: informatika
Typ záverečnej práce: diplomová
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: Vývoj softvéru a rozšírená realita
Software development and extended reality

Anotácia: Programátori pri vývoji softvéru najčastejšie narábajú so zdrojovým kódom v rámci vývojárskeho prostredia pomocou klávesnice a myši, a tento kód im je zobrazovaný prostredníctvom malého počtu 2D monitorov. Takýto tradičný prístup ignoruje možnosti, ktoré nám poskytujú novovzniknuté technológie pre zobrazovanie virtuálnej reality ako napr. Oculus Quest alebo obohatenej reality ako napr. XREAL Light. Veríme, že vhodná migrácia jednotlivých funkcionalít vývojárskeho prostredia do sveta virtuálnej alebo obohatenej reality zvýši produktivitu programátorov, zlepší ich používateľský zážitok a skráti čas potrebný na oboznámenie programátora s neznámym softvérovým systémom.
Analyzujte existujúce podporné nástroje a prostredia určené na vývoj softvéru. Zamerajte sa najmä na 3D riešenia.

Cieľ: Analyzujte už identifikované
problémy, nedostatky a návrhy vylepšení existujúcich prístupov. Navrhnite a implementujte vlastný prístup, v rámci ktorého aplikujete technológie virtuálnej alebo obohatenej reality v kontexte vývoja softvéru. Vyhodnoťte dopad vášho riešenia v porovnaní s tradičnými prístupmi.

Literatúra: Noptanit Chotisarn, Leonel Merino, Xu Zheng, Supaporn Lonapalawong, Tianye Zhang, Mingliang Xu and Wei Chen. 2020. A systematic literature review of modern software visualization. J. Vis., 23(4), 539–558.

Vedúci: doc. Ing. Ivan Polášek, PhD.
Konzultant: Ing. Juraj Vincúr, PhD.
Katedra: FMFI.KAI - Katedra aplikovanej informatiky
Vedúci katedry: doc. RNDr. Tatiana Jajcayová, PhD.

Dátum zadania: 09.12.2024

Dátum schválenia: 10.12.2024 prof. RNDr. Roman Ďurikovič, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

I hereby declare that I have written this thesis by myself, only with help of referenced literature, under the careful supervision of my thesis advisor.

.....

Bratislava, 2025 Bc. Matej Budoš

Acknowledgement

Abstract

Keywords:

Abstrakt

Kľúčové slová:

Contents

List of Figures

List of Tables

Terminology

Motivation

Chapter 1

Úvod

Virtuálna a augmentovaná realita patria medzi prelomové technológie, ktoré zásadne ovplyvňujú spôsob, akým človek interaguje s digitálnym obsahom. Ich využitie presahuje hranice herného priemyslu a čoraz viac sa uplatňuje v oblastiach ako vzdelávanie, simulácie či softvérové inžinierstvo. Obe technológie majú spoločný cieľ – rozšíriť ľudské vnímanie prostredia prostredníctvom priestorovej a interaktívnej vizualizácie dát.

V modernej terminológii sa používa všeobecný pojem XR, ktorý zahŕňa VR, AR a zmiešané formy reality[21]. XR umožňuje vytvárať priestorové pracovné prostredia v ktorých sú komplexné informácie usporiadané, čím sa zvyšuje situačné povedomie a znižuje kognitívna záťaž vyplývajúca z rozdelených 2D rozhraní. V kontexte vývoja softvéru, vývojári strávia významnú časť pracovného času ladením a porozumením správania systému. Proces debuggingu typicky prebieha iteratívne a zahŕňa kroký ako familiarizácia, reprodukcia chyby, jej lokalizovanie, porozumenie príčine a jej následné opravenie.

V tradičných IDE sú tieto aktivity rozptýlené do viacerých oddelenýchh pohľadov a externých nástrojov. Prepínanie medzi oknami (kód, debugger, profiler, dokumentácia), zariadeniami (monitor, laptop, vr zariadenie) a opätovné spúšťanie procesu ladenia generuje výrazné kognitívne a časové náklady.

Cieľom práce je preskúmať, do akej miery môže takáto priestorová reprezentácia priebehu ladenia v XR prostredí pomôcť pri orientácií v kóde a lokalizácií chyby a znížiť kognitívnu záťaž spôsobenú prepínaním kontextu.

RQ1: Dokáže XR aplikácia na podporu debuggovania zredukovať čas na identifikovanie a následné opravenie chyby v kóde?

RQ2: Má používanie XR/AR debuggeru v porovnaní s klasickým IDE za následok zníženie kognitívnej záťaže súvisiacej s prepínaním pozornosti medzi rôznymi zdrojmi informácií (zásobník volaní, súbory, breakpoints)?

-nieco okolo xr -vývojári strávia veľa času v debuggingu -proces debuggingu: orientacia v projekte reprodukcia chyby lokalizovanie chyby porozumenie príčine oprava

testovanie chyby -cieľom je vymyslieť nejakú aplikáciu kde by sa vývoj/debug softvéru presunul do xr sveta a poskytol vývojárom podporu pri debuggovaní a vizualizácii architektúry bez toho aby sme museli prechádzať medzi zariadeniami/ aplikáciami/ oknami - teda toto prepinanie - context switch spôsobuje kognitivny overload

Chapter 2

Súčasný stav

Táto kapitola sa zaoberá teoretickými a technologickými východiskami práce. Cieľom je zadefinovať základné pojmy, aktuálne technológie ich význam pre oblasť vizualizácie softvéru, ako aj nástroje použité pri návrhu a implementácii prototypu.

2.1 Virtuálna realita, augmentovaná realita a iné

2.1.1 Virtuálna realita (VR)

Virtuálna realita je počítačom generované trojrozmerné prostredie, ktoré používateľovi prezentované prostredníctvom zobrazovacieho zariadenia – typicky náhlavnej súpravy alebo iného vizuálneho výstupu. Cieľom VR je dosiahnuť pocit prehĺbenia (immersion) a prítomnosti (presence), teda subjektívny dojem, že používateľ skutočne je v danom digitálnom prostredí kde zmyslové vnemy korešpondujú s virtuálnym svetom.

2.1.2 Augmentovaná realita (AR)

Na rozdiel od VR, v prípade augmentovanej reality nejde o úplné nahradenie reálneho sveta, ale skôr o rozšírenie vnímaného reálneho prostredia o digitálny obsah. Digitálne objekty alebo informácie sú premietané alebo zobrazované tak, aby boli kontextovo a priestorovo previazané s fyzickým svetom - napríklad v kontexte kamery smartfónu, AR zariadenia so see-through alebo passthrough technológiou. AR tak umožňuje kombináciu reálneho prostredia a virtuálnych prvkov, bez úplného vylúčenia reálneho sveta z vnímania používateľa.

2.1.3 Ostatné pojmy

Koncept reality–virtuality kontinuum bol predstavený autormi Paul Milgram a Fumio Kishino[18], ktorí zaradili AR a VR ako dva opačné póly kontinua, medzi ktorými sa nachádzajú rôzne stupne prepojenia reálneho a virtuálneho sveta.

V modernej literatúre sa čoraz častejšie používa pojem Rozšírená Realita (XR) ako nadkategória zahŕňajúca VR, AR a ďalšie formy zmiešaných alebo hybridných realít - teda Mixed Reality (MR), prípadne aj ďalšie varianty. Podľa autorov, ktorí navrhujú aktuálnejší rámec pomenovania, XR definuje premenná „X“ ako zástupný symbol pre akýkoľvek typ reality (virtuálna, rozšírená, zmiešaná atď.)[21].

Zhrnutie pojmov:

- **AR** – reálny svet + digitálne/pridané prvky
- **VR** – plne virtuálny svet, realita nahradená
- **MR** – všeobecný pojem pre implementácie, ktoré kombinujú reálne a digitálne prvky tak, že sú súčasne prezentované používateľovi
- **XR** – najširší pojem, nadkategória zahŕňajúca AR, VR, MR a potenciálne ďalšie formy realít.

2.2 Typy zariadení

Moderné zariadenia pre virtuálnu realitu je možné klasifikovať primárne podľa spôsobu, akým spracovávajú a zobrazujú virtuálne prostredie. Podľa Angelov et al.[2] je možné dnešné VR headsety rozdeliť do troch hlavných kategórií: samostatné (standalone), pripojené (tethered) a mobilné VR zariadenia, pričom v posledných rokoch vzniká aj hybridná kategória headsetov kombinujúcich vlastnosti oboch hlavných typov.

2.2.1 Samostatné zariadenia (Standalone)

Samostatné VR headsety obsahujú všetky potrebné výpočtové komponenty zabudované priamo v zariadení. Disponujú integrovaným grafickým procesorom, zobrazovacím systémom a senzormi na určovanie polohy a orientácie používateľa. Týmto spôsobom umožňujú plnohodnotný VR zážitok bez potreby externého hardvéru.

Výhody:

- vysoká mobilita a ergonómia, keďže nevyžadujú káblové prepojenie,
- rýchle nasadenie a jednoduché používanie,
- vhodné na prototypovanie, prezentačné účely a nenáročné aplikácie.

Nevýhody:

- obmedzený výpočtový výkon viazaný na mobilné procesory,
- nemožnosť spúšťať technicky náročné VR aplikácie,
- výdrž batérie limitujúca dĺžku nepretržitého používania.

2.2.2 Pripojené zariadenia (Tethered)

Pripojené VR headsety vyžadujú externý výkonný počítač, ktorý zabezpečuje generovanie grafiky, spracovanie senzorových dát a rendering VR prostredia. Headset typicky obsahuje len zobrazovaciu jednotku a priestorové senzory.

Výhody:

- výrazne vyšší výpočtový výkon limitovaný iba pripojeným PC,
- podpora náročných simulácií, vizualizácií a profesionálnych aplikácií,
- vysoká kvalita obrazu a presné sledovanie polohy.

Nevýhody:

- káblové prepojenie znižuje mobilitu a voľnosť pohybu,
- závislosť od externého hardvéru a zložitejšia inštalácia.

2.2.3 Mobilné VR zariadenia

Mobilné VR headsety využívajú smartfón ako hlavný výpočtový komponent aj displej. Headset zväčša obsahuje iba optické šošovky a držiak na telefón, pričom spracovanie VR obsahu prebieha priamo v mobilnom zariadení využívajúcom zabudované senzory.

Výhody:

- najnižšia cena a jednoduchá dostupnosť,
- vysoká mobilita,
- vhodné na nenáročné 360° video a základné VR aplikácie.

Nevýhody:

- veľmi obmedzený výkon, silne závislý od konkrétneho smartfónu,
- chýbajúce ovládače a obmedzené možnosti interakcie,
- nízka presnosť sledovania, nevhodné pre seriózne VR aplikácie.

2.3 Passthrough vs Seethrough

2.3.1 Passthrough

Technológia passthrough využíva kamery umiestnené na prednej časti headsetu, ktoré snímajú reálny svet a prenášajú ho do displeja v reálnom čase. Ide o formu **video-based AR**, ktorá umožňuje stabilné priestorové ukotvenie virtuálnych objektov a plynulé miešanie s fyzickým priestorom.

Výhody:

- úplná kontrola nad vizuálnou scénou – zariadenie môže pixelovo presne kombinovať reálne a virtuálne prvky,
- možnosť úplného prekrytia reálneho prostredia (napr. pri tmavých alebo kontrastných vizualizáciách),
- podpora hlbokovej percepcie vďaka farebnému obrazu a depth-senzorom,
- kompatibilita s existujúcimi VR interakčnými technikami.

Nevýhody:

- latencia – aj moderné kamery majú malé, ale citeľné oneskorenie, čo môže ovplyvniť komfort a presnosť interakcie,
- nižšia vizuálna kvalita reálneho sveta (kompresia, šum, skreslenie optiky),
- závislosť od kamier – pri slabom osvetlení alebo vysokom kontraste klesá kvalita obrazu,
- potenciálna nevoľnosť spôsobená rozdielmi medzi pohybom hlavy a aktualizáciou obrazu,
- vyššia výpočtová náročnosť, keďže zariadenie musí spracovávať a renderovať video aj virtuálne objekty.

Meta Quest 3 Meta Quest 3 prináša farebný passthrough s vysokým rozlíšením, integrovaný depth-senzor a efektívne mapovanie priestoru. V kontexte vizualizácie softvéru umožňuje kombinovať reálne pracovné prostredie s virtuálnymi artefaktmi, čo redukuje kognitívne prepínanie medzi kontextmi.

Ďalšie významné passthrough headsety:

- **Varjo XR-4**
- **Apple Vision Pro**

2.3.2 Seethrough

Seethrough technológia, využíva priesvitné optické prvky, cez ktoré používateľ vidí priamo reálny svet. Virtuálny obraz je premietaný na transparentné displeje, ktoré dopĺňajú realitu o digitálne vrstvy.

Výhody:

- nulová latencia vnímania reálneho sveta – používateľ ho vidí bez sprostredkovania kamerou,
- prirodzené vnímanie hĺbky a farieb,
- vhodné pre úlohy, kde je kritická vizuálna integrita reality.

Nevýhody:

- obmedzený jas a kontrast virtuálnych objektov – pri jasnom dennom svetle môžu byť sotva viditeľné,
- obmedzené prekrytie reality – virtuálne objekty nemôžu úplne nahradiť alebo „zakryť“ reálne,
- užšie zorné pole (FOV) v porovnaní s VR/passthrough riešeniami,
- vyššia cena a technická komplexnosť vzhľadom na pokročilú optiku.

Projekt Northstar Projekt *Northstar*, open-source AR headset od Leap Motion/Ultraleap, poskytuje výskumné prostredie pre experimentovanie s see-through vizualizáciami. Kombinuje priesvitné displeje s vysokou obnovovacou frekvenciou, presné sledovanie rúk (Leap Motion) a otvorený hardvérový dizajn vhodný pre výskum interakčných techník.

Ďalšie see-through zariadenia:

- **Microsoft HoloLens 2**
- **Magic Leap 2**

2.4 Vývojové nástroje

V dnešnej dobe dominujú vývoju VR/AR aplikácií platformy Unity a Unreal Engine. Zatiaľ čo je Unity často preferovanou voľbou pre svoje užívateľské rozhranie, flexibilitu, podporu multiplatformového nasadenia a všeobecne rýchle prototypovanie, Unreal Engine predstavuje štandard pri projektoch, kde je požadovaná vysoká vizuálna kvalita a realistické renderovanie v reálnom čase.

2.4.1 XR vývojové frameworky

OpenXR je low-level štandard vytvorený organizáciou Khronos Group. Poskytuje jednotné API pre rendering v XR ako tracking (hlava, ruky, kontroléry) a priestorové ukotvenie.

Toolkity predstavujú high-level frameworky, ktoré fungujú ako nadstavba nad štandardizovaným rozhraním OpenXR. Poskytujú interakčné vzory, prefaby a UI komponenty ktoré výrazne zrýchľujú a zjednodušujú vývoj VR/AR aplikácií. Medzi ne patria Mixed Reality Toolkit (MRTK) od Microsoft, XR Interaction Toolkit od Unity, XRTK toolkit čo je oddelená vetva z MRTK spravovaná komunitou.

2.5 Vizualizácia softvéru vo VR/AR

2.5.1 Statická vizualizácia

Statická analýza sa zaoberá skúmaním zdrojového kódu bez jeho spustenia a bez potreby vykonania samotného programu. Súčasťou každého integrovaného vývojového prostredia (IDE) je statická analýza vo forme syntaktického zvýraznenia, kontroly nepoužitých premenných či odhalenia potenciálnych chýb. Avšak tradičné textovo založené editory alebo IDE často neponúkajú „veľký obraz“ architektúry systému, štruktúr a vzťahov medzi jednotlivými komponentmi. Táto hraničná medzera medzi lokálnym pohľadom na kód a jeho globálnou architektúrou predstavuje jednu z hlavných motivácií pre využitie vizualizácií v VR/AR.

Zvyčajne ide o vizualizáciu kde objekt odpovedá triede a jeho tvar, farba, a pozícia medzi ostatnými objektami odpovedajú rôznym atribútom ako je počet metód, počet riadkov kódu, väzbu s inými triedami či pozícia v projekte. Existuje mnoho metafor ktoré vizualizujú softvér ako mesta [16][27][24][11][17], ostrovy[22], lesy[5], mapy[8] alebo len ako objekty v priestore [15][10].

2.5.2 Dynamická vizualizácia

Dynamická analýza sa na druhej strane zameriava na pozorovanie správania kódu počas jeho behu, pričom údaje sú získavané v reálnom čase alebo v kontrolovanom experimentálnom prostredí. Na rozdiel od statickej analýzy, ktorá pracuje výhradne so zdrojovým kódom, dynamická analýza odhaľuje aspekty, ktoré sa prejavujú až počas skutočného vykonávania programu, ako sú výkonnostné charakteristiky, alokácie pamäte, interakcie medzi modulmi a hodnoty premenných.

Zvyčajne sa vizualizujú spúšťané procesy, volania funkcií, komunikáciu medzi komponentmi či tok dát ako priestorovo usporiadané objekty, ktorých vlastnosti sa menia v závislosti od správania systému [15]. Existujú aj riešnia ktoré kombinujú dynamickú

analýzu spolu so statickou kde sa na vizualizácii projektu generuje aj jeho správanie [16][27][17].

2.5.3 Evolúcia systému

Evolúcia softvérového systému predstavuje dlhodobý proces zmien systému ako pridávanie a odoberanie funkcionality, refaktorovanie, odstraňovanie chýb či celkové zmeny architektúry. Tradičné nástroje na analýzu evolúcie, ako sú textové difs často nedokážu efektívne podporiť orientáciu v komplexných a rozsiahlych projektoch, v ktorých sa zmeny prejavujú na viacerých úrovniach - od súborov, tried a modulov, až po architektonické vrstvy. Využitím verzionovacieho systému Git je možné v trojrozmernom priestore vizualizovať príspevky autorov a zmeny statických aspektov systému v definovanom časovom období[27][24].

2.6 Vývoj softvéru vo VR/AR

2.6.1 Programovanie

Vo VR/AR sa programovanie realizuje buď ako moderné IDE s panelmi kódu, konzolou a nástrojmi rozmiestnenými v 3D priestore, často s podporou priameho písania cez klávesnicu a živého spúšťania kódu [19][12]. Alternatívou sú vizuálne a modelovo riadené prístupy, v ktorých sú fragmenty kódu naviazané na prvky UML[13] alebo na abstraktnom syntaktickom strome kódu [6]. Na výučbu programovania sú rozšírené blokové jazyky ktoré umožňujú skladať programy z 3D blokov na princípe Scratch a okamžite pozorovať správanie [7][26][25].

2.6.2 Debuggovanie

Priestorové debuggovanie stavia na 3D reprezentáciách toku riadenia a dát ako premenné [23] alebo zásobník volaní a časová os[14] tvorená breakpointami krokovania ktoré sú vizualizované ako navigovateľné objekty. Iné riešenie[20] vizualizuje delta-debugging metódu kde sa minimalizuje vstup ktorý spôsobuje chybu v systéme.

2.6.3 Kolaborácia

AR prirodzene rozširuje kolaboráciu pri vývoji kde analýza softvéru, spoločné opravy chýb a diskusie prebiehajú v zdieľanej miestnosti. Prehľady a vizualizácie správania systému možno skúmať kolaboratívne, čo podporuje spoločné pochopenie komplexných systémov na príklad pri priebehu údržby[11][15]. Daľšia možnosť kolaborácie je vzdi-

alené párové programovanie so zdieľanou obrazovkou a avatarovou prítomnosťou, čo zvyšuje sociálnu prítomnosť a rýchlosť riešenia úloh[4].

2.7 Analýza existujúcich riešení

2.7.1 VrCity

Riešenie VR City[27] predstavuje nástroj na vizualizáciu softvéru vo VR, ktorý modifikuje mestskú metaforu a umožňuje súčasne zobrazovať statické, dynamické aj evolučné aspekty na jemnej granularite metód.

Kľúčom je nový layoutovací algoritmus, ktorý zohľadňuje väzby medzi triedami. Postup vychádza zo spektrálneho usporiadania grafu závislostí (Fiedlerov vektor) a pri rozložení blokov využíva Hilbertovu krivku. Triedy sú reprezentované budovami a metódy poschodiami. Metriky (napr. počet riadkov) definujú počet blokov na poschodí a výšku podlažia. Riešenie obsahuje viacvrstvovú štruktúru: vrstvu spojení (vzťahy), vrstvu autorov (sféry nad mestom s trajektóriami podľa nedávnych zmien),vrstvu mesta, priestor pre kód a UI vrstvu (voľby režimov). Dynamické aspekty podporuje „trace mode“ so zvýrazňovaním volaných metód farebnosťou a transparentnosťou. Evolúciu zase animácia príspevkov autorov a zmien metrík medzi revíziami. Na navigáciu a interakcie využívajú HTC Vive (chôdza v miestnosti, teleport, výber ukazovadlom, uchopenie gripmi, spúšť). Súčasťou je aj režim remodularizačnej analýzy s vyfarbením podľa balíkov alebo semantického zhlukovania.

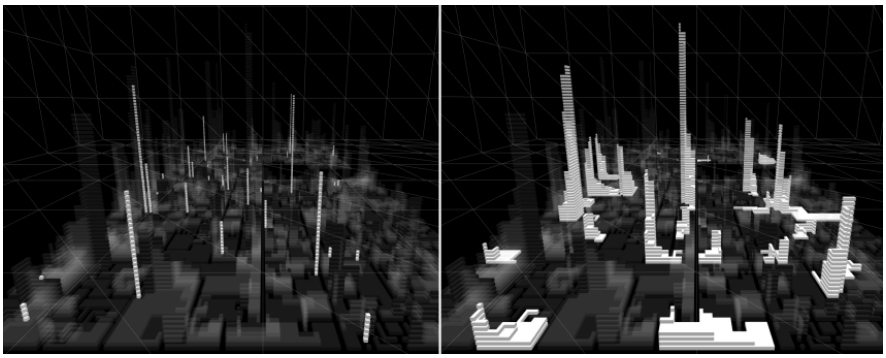


Figure 2.1: Pohľad na mesto so zvýraznenými interfejsmi (na ľavo) a abstraktnými triedami (v pravo)[27].

Evaluácia

Autori overili prístup demonštračnými prípadovými štúdiami na dvoch open-source Java systémoch (JHotDraw, JUnit). Oba projekty vizualizovali vo VrCity a sledovali:

- statické aspekty cez mapovanie metrík na budovy a poschodia,

- dynamické aspekty cez prehrávanie zaznamenaných trás programového behu (získaných nástrojom inTrace) s farebným zvýrazňovaním volaných metód a ich prekrytia,
- evolučné aspekty cez vzorkovanie commitov z Git a animovanú vizualizáciu zmien a príspevkov autorov.

Vyhodnotenie bolo kvalitatívne: demonštrovali, že vizualizácia podporuje vizuálne porovnanie štruktúry, identifikáciu indícií zápachov kódu na úrovni metód, sledovanie scenárov a mapovanie príspevkov vývojárov.

Silné stránky

- Jednotný pohľad na statiku, dynamiku a evolúciu na úrovni metód - zobrazenie metrík, väzieb, volaní a zmien v čase.
- Layout reflektuje coupling - variabilný tvar budov uľahčuje vizuálnu identifikáciu rozhraní, abstraktných tried a indícií zápachov kódu (dlhé/god metódy, komplexné podmienky).
- Trace mode s farebným prekrytím a animáciou umožňuje rýchlo určiť triedy/metódy zapojené do scenárov a ich prekrytie.
- Priestor kódu v VR (rozšíriteľné „code labels“) podporuje prehliadanie zdrojov bez opustenia prostredia a využíva priestorovú pamäť.
- Priestorové interakcie a navigácia (chôdza, uchopenie, škálovanie, teleport) znižujú kognitívnu záťaž typickú pre 3D pri 2D vstupe.
- Režim remodularizácie umožňuje porovnávať balíky vs. semantické klastre v kontexte väzieb.

Slabé stránky

- Evolučná animácia nemení pozície budov a zobrazuje iba artefakty prítomné v aktuálnej revízii - vytvorené a odstránené triedy sa v animácii nikdy neobjavia.
- Autori sami spochybňujú praktický prínos vizualizácie evolúcie vo všeobecnom vývoji (vidia skôr využitie pri code review).
- Stabilita rozloženia v čase nie je riešená - ako budúcu prácu uvádzajú kombináciu s vnoreným alebo grafovým layoutom pre väčšiu stabilitu.
- Detekcia zápachov kódu nie je automatizovaná (navrhnuté ako budúce rozšírenie) - aktuálne ide o vizuálne indície.

- Obmedzenia VR priestoru (cca 3×3 m) vyžadujú teleport ako kompenzáciu pri veľkých mestách.

2.7.2 IDEvelopAR

IDEvelopAR [12] je rozhranie v augmentovanej realite zamerané na podporu pochopenia kódu a navigácie počas údržby. Rieši limity klasických integrovaných vývojových prostredí (strata vizuálneho kontextu, preťaženie prepínaním) prenesením pracovného priestoru do augmentovaného priestoru s prepojenými panelmi s kódom. Architektúra pozostáva z aplikácie pre HoloLens 2 a zásuvného modulu pre integrované vývojové prostredie z rodiny JetBrains (napr. IntelliJ), ktorý po sieti poskytuje zdrojové súbory, analýzy aj návrhy dopĺňania. Všetka logika beží v integrovanom vývojovom prostredí.

V augmentovanom priestore sú triedy, metódy a konštruktory reprezentované panelmi, ktoré možno voľne umiestňovať, škálovať a otáčať gestami (stlačenie ukazováka s palcom a ťahanie). Nové panely sa pri navigácii ukladajú pomocou automatického rozloženia a vizuálnych prepojení: prechod na metódu alebo konštruktor je zobrazený zelenou väzbou, prechod na triedu modrou. Väzby súčasne reprezentujú históriu navigácie aj vzťahy. Pri väčšej vzdialenosti sa panely prepínajú do prehľadového zobrazenia s menším množstvom informácií (semantické priblíženie).

Navigácia zahŕňa menu na dlani (prepínanie režimu programovania, spustenie a kompiláciu v stolovom integrovanom vývojovom prostredí, hierarchický prehľad projektu) a otváranie panelov priamo z kódu. Posúvanie využíva snímanie pohľadu zariadenia, ktoré sa prispôsobuje rýchlosti čítania. V režime programovania je možné kód priamo upravovať pomocou klávesnice s bezdrôtovým pripojením a s podporou dopĺňania kódu. Zmeny sa okamžite synchronizujú s integrovaným vývojovým prostredím.

Evaluácia

Overenie nástroja prebehlo dvojfázovo: najskôr kognitívnym prechodom podľa rámca kognitívnych dimenzií a následne formujúcou používateľskou štúdiou.

Kognitívny prechod podľa rámca kognitívnych dimenzií:

- interná tímová analýza prototypu podľa dimenzií (napr. vyjadriteľnosť, viditeľnosť, odolnosť voči zmenám),
- identifikácia a prioritizácia problémov použiteľnosti,
- zapracovanie kľúčových úprav pred štúdiou s používateľmi (napr. jednorazové zatvorenie celej vetvy panelov, farebné odlíšenie prepojení).

Formujúca používateľská štúdia:

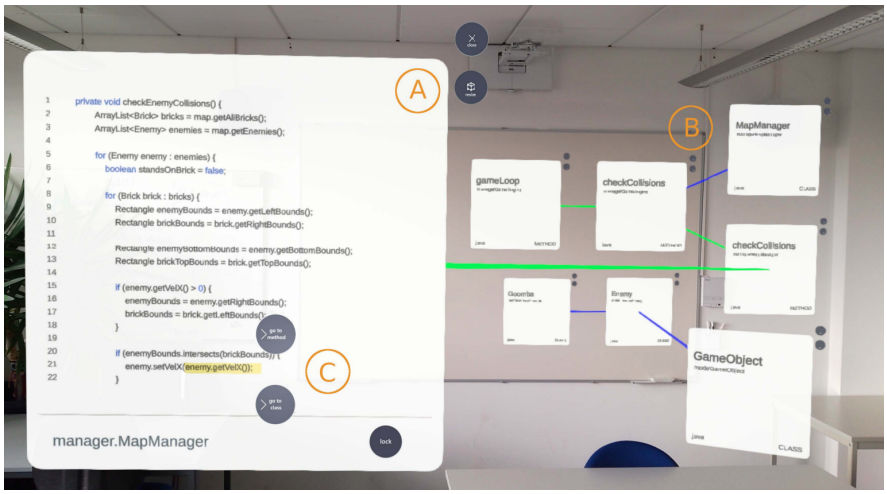


Figure 2.2: Ukážka IDE v praxi. (A) Panel s kódom, (B) Dva nezávislé, vizuálne prepojené stromy panelov kódu zobrazené na rôznych úrovniach detailu (semantické priblíženie), (C) Navigácia v kóde po kliknutí na sledovateľný prvok kódu.[12]

- krátke zaškolenie v ovládaní zariadenia a následne v používaní nástroja,
- párové programovanie pri vyhľadávaní a opravovaní chýb v ukážkovom projekte,
- účastníci sa striedali v rolách,
- záznam zobrazenia v okuliaroch, zvukový záznam komunikácie a protokolovanie interakcií,
- vyplnenie dotazníkov (Systémová škála použiteľnosti (SUS) a Dotazník používateľskej skúsenosti) a krátky rozhovor,
- po prvom behu bola odstránená zistená prekážka v otváraní panelov cez menu na dlani a ďalšie behy prebehli s upravenou verziou nástroja.

Silné stránky

- Účastníci uvádzali lepší prehľad a porozumenie kódu oproti klasickému IDE. Strom prepojených panelov a vizuálne linky uľahčovali sledovanie histórie navigácie a častí volacích hierarchií.
- Kvantitatívne výsledky naznačujú akceptovateľnú použiteľnosť: SUS 69–72 (OK–good) - väčšina funkcií bola hodnotená ako užitočná (blízko 5–6 bodom).
- V UEQ bola perspicuita nad priemerom - používateľské komentáre potvrdzovali väčší prehľad, a orientáciu v projekte.
- Semantické zoomovanie bolo hodnotené pozitívne („minimized view was really good“) a manipulácia s panelmi (presuny/umiestnenie) fungovala dobre.

- Prejavil sa učebný efekt a tvorba vlastných priestorových stratégií (vrátane 360° rozloženia), čo naznačuje potenciál zlepšovania efektivity s časom.

Slabé stránky

- Efektivita bola vnímaná nižšia než v bežnom IDE (UEQ - subjektívne komentáre „niektoré veci zaberajú príliš veľa času“).
- Problémy s interakciami založenými na očiach: únavnosť/miešané názory na rolovanie, ťažkopádne stlačenie „eye-scroll lock“ pre neúmyselné skrolovanie pri zacieľovaní tlačidla.
- Editácia kódu: umiestnenie kurzora (Ctrl + pohľad) bolo pre viacerých neintuitívne a nespoľahlivé.
- Technické/tracking limity HoloLens 2 a gest viedli k neúspešným interakciám na prvý pokus - prvý beh odhalil vážny problém s otváraním panelov cez hand menu (následne opravené).
- Niektoré funkcie mali nižšiu vnímanú užitočnosť (napr. zmena veľkosti panelov, semantické zoomovanie) a vyskytla sa rozporuplná použiteľnosť vybraných prvkov (otváranie panelov, rolovanie).
- Chýbajú známe IDE pomôcky (zvýrazňovanie chýb, indície nepoužitých entít), čo znižovalo komfort práce.
- Kognitívna prechádzka identifikovala ďalšie potreby: bohatšie linky, odlíšenie typov panelov, zobrazenie ľubovoľných súborov, indikácia duplicitne otvorených panelov, delenie stromu, voľba implementácie pri abstraktných triedach, perzistencia relácie, pripínanie komentárov a možnosti pripnutia/zarovnania panelov do zorného poľa.

2.7.3 A Novel Approach for Software 3D-Debugging in Virtual Reality

Riešenie[23] predstavuje koncept 3D ladenia vo virtuálnej realite, ktorý má podporiť budovanie mentálnych modelov programátora počas explorácie a ladenia kódu. Autori cielia na zníženie dezorientácie a vizualizáciu cesty k aktuálnemu problému, pričom vychádzajú zo stratégií porozumenia kódu zdola-nahor a kognitívnych dizajnových prvkov. Implementácia je zameraná na JavaScript: serverová časť využíva ladiaci nástroj prehliadača Google Chrome (rozhranie WebSocket), ktoré je sprístupnené cez sprostredkujúci server v NodeJS (mapovanie na rozhranie HTTP), používateľská časť je

pripravená v nástroji Unity a beží na zariadení Oculus Quest 2. V priestore sa zobrazuje viacero okien s kódom s aktuálnym riadkom (šípka, zvýraznenie syntaxe, číslovanie riadkov). Zásobník volaní je reprezentovaný trojrozmernými kockami s názvom funkcie a premennými (stav v červených boxoch). Časová os tvorí snímky zásobníka volaní v čase a umožňuje prehliadanie minulých krokov. Navigácia kombinuje fyzickú chôdzu a teleportáciu (platformy každých päť krokov). Aktuálne zameranie sa indikuje polohou používateľa na časovej osi a vizuálnymi väzbami (farebná línia medzi zásobníkom volaní a oknom kódu).

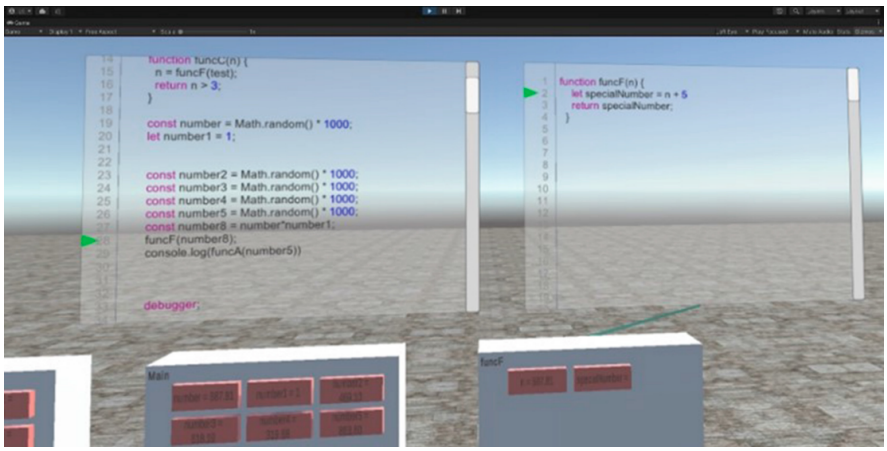


Figure 2.3: Ladiaci nástroj zobrazujúci okná s kódom a zásobník volaní[14].

Evaluácia

Overenie prebehlo formou prieskumnej používateľskej štúdie zameranej na uskutočniteľnosť a prvotnú prijateľnosť.

- zostavenie funkčného prototypu a chybného programu v jazyku JavaScript na ladenie,
- malá homogénna vzorka 6 osôb s predchádzajúcou skúsenosťou s programovaním a ladením (bez požiadavky na skúsenosť s virtuálnou realitou),
- krátke oboznámenie s ovládaním zariadenia a princípmi práce v prototypovej aplikácii,
- nájsť a vysvetliť príčinu chyby v pripravenom programe s využitím dostupných funkcií (prechádzanie krokov, zobrazenie zásobníka volaní, práca s časovou osou),
- otázky pred a po riešení úlohy (očakávania, vnímané prínosy, skúsenosť s navigáciou a zrozumiteľnosťou zobrazení) a sprievodná diskusia po dokončení,
- opisná kvalitatívna syntéza pozorovaní a odpovedí s formulovaním silných a slabých stránok a návrhov zlepšení.

Silné stránky

- Účastníci úspešne dokončili úlohu a celkové vnímanie 3D ladenia bolo pozitívne a videli potenciál pre budúce pracovné využitie.
- Navigácia bola hodnotená priaznivo: chôdza po časovej osi podporovala porozumenie a teleportácia uľahčovala presuny na väčšie vzdialenosti pri obmedzenom priestore.
- Indikácia aktuálneho kroku (šípka v kóde) a vizuálne prepojenie zásobníka volaní s príslušným súborom zlepšovali orientáciu v kontexte.
- Koncept časovej osi (prehliadanie predchádzajúcich stavov volaní a premenných) zlepšoval orientovanie pri ladení.
- Účastníci so skúsenosťou s virtuálnou realitou sa adaptovali rýchlejšie, čo naznačuje učebný efekt a potenciál zvyšovania efektivity s praxou.

Slabé stránky

- Zmiešané ohlasy na vnímanie intuitívnosti. Niektoré prvky boli pre časť používateľov nejasné v závislosti od ich mentálneho modelu, skúseností s virtuálnou realitou a programátorskej expertízy.
- Chýbajúce funkcie znižovali komfort: potreba zobraziť názvy súborov, označiť zmenené premenné medzi krokmi, zobraziť hierarchiu projektu, umožniť preskupovanie blokov či farebné kódovanie.
- Prechod medzi klasickým integrovaným vývojovým prostredím a ladením vo virtuálnej realite môže vyžadovať prepínanie mentálnych modelov a zvyšovať kognitívnu záťaž.

2.7.4 Virtual Reality Debugging Support

Riešenie H.I.D.E.[23] predstavuje prototyp virtuálneho prostredia pre ladenie kódu zameraný na začiatočníkov. Cieľom je podporiť aktivity pozorovania, preskúmavania a tvorby hypotéz počas ladenia prostredníctvom 3D vizualizácií toku riadenia a dát. Systém je implementovaný v Unity a nasadený na HTC Vive, pracuje s programami v jazyku Lua. Každý lexikálny token je zobrazený ako 3D objekt, pričom pri pohľade alebo zacielení ovládačom sú kontextovo zobrazené hodnoty premenných v danom mieste vykonávania. Napríklad pri podmienke `if` systém indikuje, či je jej telo vstúpené, a pri porovnaní (napr. `n == 0`) zobrazuje aktuálne hodnoty operandov a výsledok. Navigácia využíva dotykové plochy ovládačov, výber a zvýraznenie prebieha cez lúče

vychádzajúce z náhlavnej súpravy a ovládačov kde sa spúšť používala na udržanie zamerania. Účelom je spriehľadniť logické vzťahy a znížiť kognitívnu záťaž mapovaní medzi kódom, tokom riadenia a dátami.

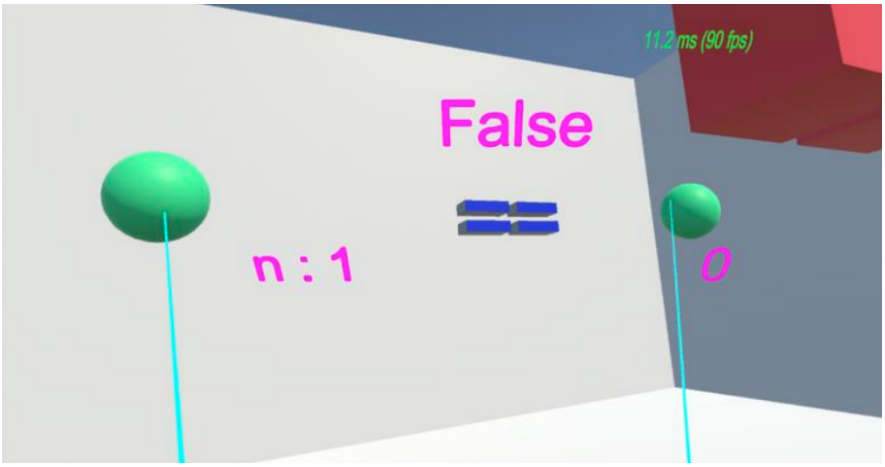


Figure 2.4: H.I.D.E – ukážka s vyhodnotenou podmienkou[23].

Evaluácia

Krátke overenie prebehlo ako pilotná skúška s dvoma študentmi informatiky (2–3 roky praxe, bez predchádzajúcich skúseností s jazykom Lua). Porovnávali bežný textový editor (Notepad++) s H.I.D.E riešením. Postup:

- zaškolenie účastníkov v ovládaní systému a oboznámenie s úlohami,
- riešenie dvoch malých skriptov v dvoch prostrediach so striedaným poradím,
- zber údajov ako časy riešenia, úspešnosť pri pochopení skriptov, slovné komentáre účastníkov a pozorovania z priebehu,
- časový limit 5 minút na každú kombináciu skript–prostredie,
- Zaznamenávaný bol priebeh spolu so slovnými komentármi,
- po skončení každej úlohy stručné vysvetlenie, čo podľa účastníkov program robí.

Silné stránky

- Účastníci opisovali systém ako intuitívny. Priestorové usporiadanie uľahčovalo porozumenie kontextu v krátkom čase.
- Vizualizácia je vhodná na preskúmavanie kódu a ponúka novú perspektívu ako sekundárny pohľad popri tradičnom ladiacom nástroji.
- Riešenie má potenciál pre výučbu a menšie tímy. Komentáre naznačili použitie ako didaktickej pomôcky a na zdieľanie kódu.

Slabé stránky

- V pilotnej skúške nik z účastníkov neodhalil chybu v H.I.D.E., čo naznačuje potrebu dlhšej fázy oboznámenia alebo úprav dizajnu.

- Preťaženie informáciami, úzke zorné pole a nižšie rozlíšenie náhlavnej súpravy spôsobovali zvýšenú kognitívnu záťaž.

- Rozloženie bolo vnímané ako príliš rozťahané. Farby a transparentnosť boli nejasné, farebné kódovanie pôsobilo neintuitívne.

- Interakcia, pri ktorej treba pozerať na objekt, aby sa zobrazili hodnoty, bola pre niektorých frustrujúca.

- Škálovanie na veľké projekty môže zhoršiť čitateľnosť.

2.7.5 Toward a VR-Native Live Programming Environment

Cieľom riešenia[6] je skrátenie slučky spätnej väzby pri vývoji aplikácií pre virtuálnu realitu odstránením potreby prepínania medzi integrovaným vývojovým prostredím a náhlavnou súpravou. Autori navrhujú a realizujú VR-natívne živé programovacie prostredie, v ktorom sa aplikácie tvoria priamo vo virtuálnej realite pomocou priamej manipulácie a štruktúrovaného editora pre Smalltalk. Kód je zobrazený ako 3D bloky odvodené zo syntaxového stromu. Interakcia prebieha pomocou virtuálnej paličky v ruke (precízne mierenie), s dvoma režimami: uchopenie blokov (ťahaj-a-pusť vrátane odnímania a vkladania do zvýraznených miest) a umiestnenie textového kurzora na textové pole. Vkladanie nových jazykových konštruktov prebieha písaním. Akonáhle je vstup syntakticky jednoznačný, editor vytvorí zodpovedajúci blok, s transformáciami v štýle GrammarCells a doplňaním identifikátorov v kontextovom menu. Pre textový vstup boli implementované dve techniky: písanie vo vzduchu a klasická virtuálna klávesnica. Prehľadávanie kódu replikuje Smalltalk prehliadač (kategórie, triedy, metódy) s otváraním metód do 3D priestoru a možnosťou vytvoriť inštancie VR tvarov (morfov) priamo z prehliadača. Spúšťanie je živé, teda zmeny metód sa okamžite prejavia, chyby sú zobrazované na virtuálnej obrazovke a chybné metódy volané v každom snímku sa dočasne pozastavia. Implementačne bola vykonávacia vrstva v Squeak/Smalltalk (živé programovanie, synchronizácia s existujúcim dvojrozmerným blokovým editorom), zobrazovacia vrstva v hernom engine Godot (GDScript), komunikácia cez zásuvný modul a rozhranie pre volanie cudzích funkcií a testovanie na náhlavnej súprave Valve Index.

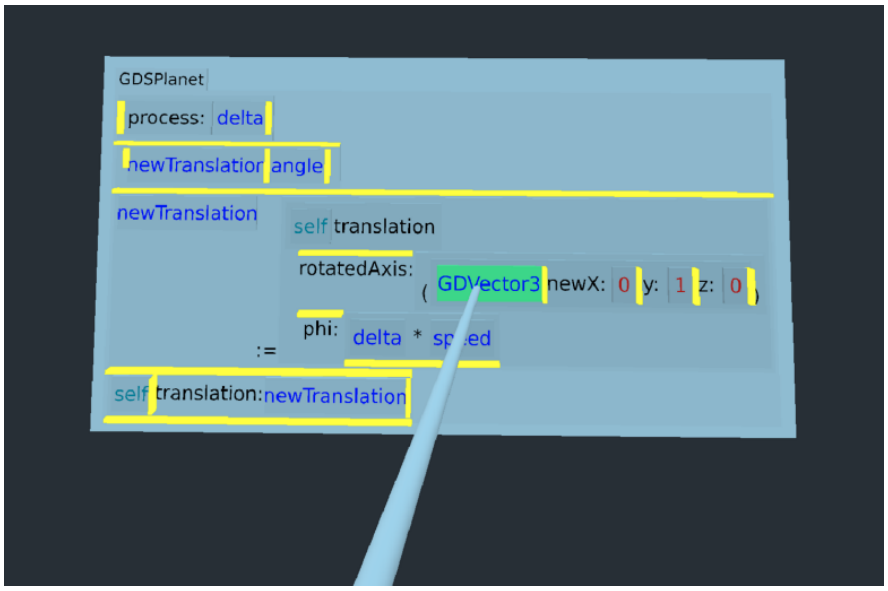


Figure 2.5: Ukážka mechanizmu na precízne mierenie kurzora pomocou paličky[6].

Evaluácia

Overenie prebehlo formou prieskumnej používateľskej štúdie zameranej na uskutočniteľnosť interakčného návrhu a identifikáciu problémov použiteľnosti. Postup bol nasledovný:

- návrh scenára úprav kódu v malej ukážkovej aplikácii (model slnečnej sústavy) a príprava dvoch skupín: bez inštruktáže a s krátkou inštruktážou,
- nábor štyroch pokročilých študentov informatiky a náhodné rozdelenie po dvoch do každej skupiny,
- zaškolenie v prípade skupiny dva s krátkou ukážkou hlavných interakcií,
- riešenie porovnateľných úloh: najprv na stolovom Smalltalk prostredí (napr. roztočenie planét), následne vo virtuálnej realite (napr. zmena rýchlostí planét podľa rýchlosti pravého ovládača),
- priebežné myslenie nahlas pri zjavnom zaseknutí dlhšom než jedna minúta poskytnutie stručnej nápovedy,
- po ukončení úloh pološtruktúrovaný rozhovor o skúsenosti s editáciou, textovým vstupom, prehľadávaním kódu a spúšťaním,
- zhrnutie kvalitatívnych zistení a pomenovanie hlavných obmedzení prototypu.

Silné stránky

- V podmienke s krátkou inštruktážou účastníci úlohu vo prostredí virtuálnej reality úspešne dokončili – koncept sa ukázal ako realizovateľný.

- Živé spúšťanie a bezprostredná vizuálna spätná väzba boli hodnotené ako intuitívne a užitočné, najmä pri menších zmenách a interakciách špecifických pre virtuálnu realitu.
- Vnímaný prínos veľkého trojrozmerného pracovného priestoru – možnosť priestorovo usporiadať viac metód a pozorovať aplikáciu vedľa súvisiaceho kódu.
- Integrovanosť (editor aj aplikácia vo virtuálnej realite) redukovala kontextové prepínanie medzi nástrojmi.
- Písanie vo vzduchu sa po krátkom učení javilo účastníkom intuitívne – chybovosť zadávania klesala v čase.
- Základná navigácia kódom cez prehliadač fungovala, znalosť Smalltalk prehliadača pomáhala orientácii.

Slabé stránky

- Nižšia subjektívna produktivita oproti práci v stolovom prostredí – viaceré akcie pôsobili ťažkopádne.
- Nepresné umiestňovanie kurzora a náročné mierenie na tenké bloky.
- Kontextové návrhy neboli vždy v zornom poli – chýbal mechanizmus upozornenia na udalosti mimo pohľadu.
- Textový vstup bol pomalší než písanie desiatimi prstami – pri väčšom množstve textu by používatelia preferovali rýchlejšie modality.
- Obmedzenia prehliadača kódu: chýbalo vyhľadávanie, metódy tried, definície tried a zvýraznenia výberov.
- Nedostatky z pohľadu použiteľnosti prototypu, napríklad prekrývanie otváraných metód na rovnakom mieste a jednotlivé technické zlyhania.

2.8 Súhrn nedostatkov

- Nejasné prepojenie medzi vizualizovanými objektmi a miestami v kóde (aktuálne umiestnenie v projekte, názov aktuálneho súboru alebo hierarchia projektu) viedli k zdržaniu a častejšiemu hľadaniu súvislostí v IDE a celkovému subjektívnemu zníženiu produktivity.
- Pri riešeniach s ladením kódu chýbal vizuálny podnet ktorý by naznačoval zmenu premennej počas krokovania.

- Niektoré riešenie neponúkali dostatočnú funkcionalitu (preskupovanie vizualizovaných prvkov, farebné zvýraznenie kódu či indície nepoužitých entít) čo znižovalo komfort z celkového použitia.
- Riešenia vo VR veľa krát neumožňovali v danom prostredí meniť kód čo spôsobovalo prepínanie prostredí, mentálnych modelov a kognitívnu záťaž.
- Riešenia s integrovaným vývojovým prostredím priamo vo VR/AR boli celkovo menej efektívne ako bežné IDE.

Chapter 3

Návrh riešenia

-o čom je kapitola

3.1 Prístup

unity - jednoduchšie prototypovanie - meta quest 3 -AR - vidíme na klávesnicu a myš, pohodlnejšie pracovanie za počítačom -islo sa smerom passthrough a nie see through kvoli slabemu seethrough - technoogia tam este nie je

3.2 Zrkadlenie Obrazovky

- kvalita obrazu je cez meta quest 3 passthrough nízka - zrkadlena obrazovka s vs code oknom v priestore Riešenie neefektivity písania kódu vo xr - máme k dispozícií plne funkčný code editor ktorý pozná každý vývojár - presnosť a efektivita kvôli použitiu myši -farebné zvýraznenie kódu - netreba meniť prostredia medzi ide(reálny svet) a vizualizáciou(virtuálny svet)** - minimalny delay - 50ms

3.3 VS code rozšírenie

na generovanie ast kódu a jeho následné preposielanie na python server

Zlepšenie mentálneho prepojenia vizualizácie a procesu ladenia získanie cesty k debuggovnému súboru na zlepšenie mentálneho prepojenia vizualizácie s kódom na obrazovke**

3.4 Python server

proxy medzi vs code a debuggerom spravy medzi vs code a debuggerom posuva do unity.

endpoint na http get žiadosť na získanie ast kódu

3.5 Vizualizácia

Unity ovládanie čisto rukami aby sa nemusali prehadzovať ovladače a klávesnice s myšou spracovavanie udalosti z debuggera

vizualizacia interaktývnych breakpointov

Jeden riadok reprezentuje jeden konkrétny breakpoint.

Na začiatku riadku na nachádza označenie v ktorom súbore sa nachádza a na ktorom riadku pre lepšie mentálne prepojenie.

Stĺpce reprezentujú po sebe idúce udalosti ladenia (breakpoint hit, step, continue, step into, step out...).

Farebné označenia breakpointov vyjadrujú typ vykonanej akcie – modrá = step, červená = continue = breakpoint hit, zelená = step into, ružová = step out.

Po kliknutí na ktorýkoľvek z vykreslených breakpointov sa používateľovi zobrazí: Chceme aby bolo čo najviac relevantných informácií/funkcií počas ladenia už vo vizualizácií aby používateľ nemusel prepínať svoju pozornosť. Aktuálny call stack v danom momente vykonávania. Prehľad funkcií, ktoré boli súčasťou stacku – každý sa dá otvoriť, čítať a prehľadávať ďalej. Evaluované premenné sú napísané v otvorenom paneli. Panel s globálnymi premennými Ovládanie debuggeru priamo v menu na ruke.

kominkacia s python serverom na posielanie žiadosti na debugger ako su next,step in, continue a získanie štruktúry kódu.

3.6 Výskum

Cieľom výskumu je empiricky porovnať efektivitu a subjektívnu záťaž pri ladení kódu v dvoch odlišných prostrediach. V štandardnom prostredí VS Code s integrovaným debuggerom a v riešení s vizualizovaným debuggerom v AR. Účastníkmi budú študenti štúdia v oblasti informatiky, aby sa zabezpečila primeraná úroveň technickej zdatnosti, skúseností s ladením a orientáciou v menších softvérových projektoch.

Pred začiatkom experimentu účastníci vyplnia vstupný dotazník zameraný na zistenie ich predchádzajúcich skúseností s jazykom Python, s prostredím VS Code a s technológiami AR/VR. Tieto údaje budú slúžiť na kontrolu homogenity vzorky a na exploratívne porovnania v rámci podskupín podľa skúseností. Každý účastník vyrieši oba scenáre, čím sa zabezpečí párový (within-subjects) dizajn s tým že poradie prostredí a úloh bude náhodne priradené, aby sa eliminoval prípadný tréningový efekt. Na konci prebehne uzatvorenie merania a zber záverečných poznámok účastníkov k priebehu.

3.6.1 Úlohy

Pre každú situáciu sú pripravené dve sady úloh(Systém na online nákup a Systém na donášku jedla), z ktorých bude účastníkovi náhodne priradená jedna. Poradie situácií je takisto náhodné. Sady sú implementované ako menší projekt pozostávajúci z 3–4 súborov; v každom súbore je jedna trieda a triedy medzi sebou komunikujú prostredníctvom metód a parametrov. Cieľom je simulovať realistický, no stále zvládnuteľný kontext ladenia menšieho modulu.

Charakter úloh: Poradie úloh je pre každého účastníka náhodne vybrané, aby sa rozložili prípadné poradie efekty a minimalizovala systematická zaujatosť.

1. úloha
   Nesprávny výpočet finálnej sumy pomocou vzorca. Problém spočíva buď v nesprávnom poradí operácií, alebo v nesprávnej vrátenej hodnote ktorá je súčasťou výpočtu. Úlohou účastníka je identifikovať zdroj odchýlky a opraviť výpočet.

2. úloha
   Premenná prechádza niekoľkými funkciami ako parameter a v niektorej medzikrokovej časti sa prepíše (napr. predvolenou hodnotou). Účastník má vystopovať miesto, kde k nechcenému prepísaniu dochádza, a opraviť chybu.

3. úloha
   Chyba v cykle spôsobuje, že v každom kroku prebieha chybné rátanie. Cieľom je odhaliť v ktorej časti iterácie vzniká chyba a následne ju opraviť.

Breakpointy budú predom určené v problematickej metóde, aby sa znížila variabilita spôsobená rozdielnymi stratégiami nastavovania breakpointov a aby bolo možné porovnať postupy účastníkov nad rovnakými východiskovými bodmi.

3.6.2 Scenár pre účastníka

1. Účastník vyplní dotazník o skúsenostiach, ktorý zisťuje prax s Python, VS Code a používaním technológií AR/VR.

2. Účastník bude oboznámený s AR riešením vrátane jeho funkcionality a základného ovládania, aby mal primerané porozumenie interakčným možnostiam pred začiatkom merania.

3. Následne prebehne priradenie poradia prostredí, ich príslušných sád úloh a poradia úloh v rámci sady, pričom všetky tieto poradia budú generované náhodne.

4. Riešenie sady X – pre prostredie A

- Účastník má 1 minútu na rýchle oboznámenie sa so štruktúrou projektu, prípadne s class diagramom, aby získal základný prehľad o komponentoch a ich vzťahoch.
- Nasleduje riešenie úloh 1–3 podľa náhodne určeného poradia, s využitím dostupných nástrojov daného prostredia.
- Po dokončení sady účastník vyplní dotazník NASA-TLX na zhodnotenie subjektívnej záťaže pri danej podmienke.

5. Riešenie sady Y – pre prostredie B prebieha obdobne s rovnakým postupom, časovaním a následným vyplnením NASA-TLX.

3.6.3 Merania

Vstupný dotazník bude slúžiť na zber demografických a skúsenostných údajov, ktoré umožnia popísať vzorku študentov. Tieto dáta budú použité aj pri analýzach vplyvu skúseností na výkon a záťaž.
Odpoveď na RQ1 bude založená na nasledujúcich metrikách výkonu:

- počet opätovných spustení debuggera (reštartov relácie), ktorý indikuje zmenu stratégie alebo potrebu obnovy stavu,
- úspešnosť, definovaná ako binárny ukazovateľ vyriešenia úlohy (správna oprava chyby) v rámci dostupného času,
- čas vyriešenia úlohy meraný od začiatku riešenia danej úlohy po identifikáciu a implementáciu korekcie.

Odpoveď na RQ2 bude vychádzať z dotazníka NASA-TLX s priamym porovnaním odpovedí v dvoch scenároch. Zváži sa spriemerovanie škál alebo použitie mediánu pre robustnejšie zhrnutie pri potenciálnych odľahlých hodnotách. Porovnanie subjektívnej záťaže medzi prostrediami umožní posúdiť, či vizualizovaný debugger v AR redukuje vnímanú náročnosť oproti štandardnému prostrediu.

Chapter 4

Implementácia

Chapter 5

Záver

5.1 Výsledky výskumu

5.2 Diskusia

Bibliography

[1] Diogo Amaral, Gil Domingues, João Dias, Hugo Ferreira, Ademar Aguiar, Rui Nóbrega, and Filipe Correia. *Live Software Development Environment Using Virtual Reality: A Prototype and Experiment*, pages 83–107. 02 2020.

[2] Vladislav Angelov, Emiliyan Petkov, Teodor Kalushkov, and Georgi Shipkovenski. Modern virtual reality headsets. In *2020 22nd International Conference on Computer Systems and Technologies (CompSysTech)*, pages 106–113, 2020.

[3] Andrew Bragdon, Steven P. Reiss, Robert Zeleznik, Suman Karumuri, William Cheung, Joshua Kaplan, Christopher Coleman, Ferdi Adeputra, and Joseph J. LaViola. Code bubbles: rethinking the user interface paradigm of integrated development environments. In *2010 ACM/IEEE 32nd International Conference on Software Engineering*, volume 1, pages 455–464, 2010.

[4] James Dominic, Brock Tubre, Charles Ritter, Jada Houser, Colton Smith, and Paige Rodeghero. Remote pair programming in virtual reality. In *2020 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pages 406–417, 2020.

[5] Ugo Erra, Giuseppe Scanniello, and Nicola Capece. Visualizing the evolution of software systems using the forest metaphor. pages 87 –92, 07 2012.

[6] Leonard Geier, Clemens Tiedt, Tom Beckmann, Marcel Taeumel, and Robert Hirschfeld. Toward a vr-native live programming environment. In *Proceedings of the 1st ACM SIGPLAN International Workshop on Programming Abstractions and Interactive Notations, Tools, and Environments*, PAINT 2022, page 26–34, New York, NY, USA, 2022. Association for Computing Machinery.

[7] Lorenzo Gerini, Giorgio Delzanno, Giovanna Guerrini, Fabio Solari, and Manuela Chessa. Gamified virtual reality for computational thinking. pages 13–21, 12 2023.

[8] Nathan Hawes, Stuart Marshall, and Craig Anslow. Codesurveyor: Mapping large-scale software to aid in code comprehension. In *2015 IEEE 3rd Working Conference on Software Visualization (VISSOFT)*, pages 96–105, 2015.

[9] Rodi Jolak, Khan-Duy Le, Kaan Burak Sener, and Michel R.V. Chaudron. Octobubbles: A multi-view interactive environment for concurrent visualization and synchronization of uml models and code. In *2018 IEEE 25th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, pages 482–486, 2018.

[10] Pooya Khaloo, Mehran Maghoumi, Eugene Taranta, David Bettner, and Joseph Laviola. Code park: A new 3d code visualization tool. In *2017 IEEE Working Conference on Software Visualization (VISSOFT)*, pages 43–53, 2017.

[11] Alexander Krause-Glau, Malte Hansen, and Wilhelm Hasselbring. Collaborative program comprehension via software visualization in extended reality. *Information and Software Technology*, 151:107007, 2022.

[12] Lucas Kreber, Stephan Diehl, and Patrick Weil. Idevelopar: A programming interface to enhance code understanding in augmented reality. In *2022 Working Conference on Software Visualization (VISSOFT)*, pages 87–95, 2022.

[13] Jakub Kucecka, Juraj Vincur, Peter Kapec, and Pavel Cicak. Uml-based live programming environment in virtual reality. pages 177–181, 10 2022.

[14] Sven-Tizian Mauer, Laura Ködel, Lukas Ertl, David Flaig, Martin Hedlund, and Gerrit Meixner. A novel approach for software 3d-debugging in virtual reality. In Jessie Y. C. Chen and Gino Fragomeni, editors, *Virtual, Augmented and Mixed Reality*, pages 235–251, Cham, 2024. Springer Nature Switzerland.

[15] Rohit Mehra, Vibhu Saujanya Sharma, Vikrant Kaulgud, and Sanjay Podder. Xrase: Towards virtually tangible software using augmented reality. In *2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 1194–1197, 2019.

[16] Leonel Merino, Mohammad Ghafari, Craig Anslow, and Oscar Nierstrasz. Cityvr: Gameful software visualization. In *2017 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pages 633–637, 2017.

[17] Leonel Merino, Mario Hess, Alexandre Bergel, Oscar Nierstrasz, and Daniel Weiskopf. Perfvis: Pervasive visualization in immersive augmented reality for performance awareness. pages 13–16, 03 2019.

[18] Paul Milgram and Fumio Kishino. A taxonomy of mixed reality visual displays. *IEICE Trans. Information Systems*, vol. E77-D, no. 12:1321–1329, 12 1994.

[19] R. Oberhauser. “immersive coding: A virtual and mixed reality environment for programmers”, twelfth international conference on software engineering advances (icsea 2017). pages 250–255, 2017.

[20] R. Oberhauser. “vr-deltadebugging”, in lavazza, l. and oberhauser, r. (eds.), proceedings of the twentieth international conference on software engineering advances (icsea 2025). page 72–77, 2025.

[21] Philipp A. Rauschnabel, Reto Felix, Chris Hinsch, Hamza Shahab, and Florian Alt. What is xr? towards a framework for augmented and virtual reality. *Computers in Human Behavior*, 133:107289, 2022.

[22] Andreas Schreiber, Lisa Nafeie, Artur Baranowski, Peter Seipel, and Martin Misiak. Visualization of software architectures in virtual reality and augmented reality. In *2019 IEEE Aerospace Conference*, pages 1–12, 2019.

[23] Nicolas Slack and Kate Howland. Virtual reality debugging support. 07 2017.

[24] Rodrigo Souza, Bruno da Silva, Thiago Mendes, and Manoel Mendonça. Skyscrapar: An augmented reality visualization for software evolution. 01 2012.

[25] Nanlin Sun, Annette Feng, Ryan Guillard-Patton, Yotam Gingold, and Wallace Lages. Programmable virtual reality environments. pages 619–620, 03 2021.

[26] Juraj Vincur, Martin Konopka, Jozef Tvarozek, Martin Hoang, and Pavol Navrat. Cubely: virtual reality block-based programming environment. 11 2017.

[27] Juraj Vincur, Pavol Navrat, and Ivan Polasek. Vr city: Software analysis in virtual reality environment. In *2017 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, pages 509–516, 2017.