MASARYK
UNIVERSITY

Faculty of Science

Facilitating and Evaluating User Behavior In Virtual 3D Environments

Doctoral disertation thesis

Pavel Ugwitz

Supervisor: Mgr. Bc. Zdeněk Stachoň, Ph.D.

Department of Geography

Brno 2022

Bibliographic entry

Author:	Pavel Ugwitz Department of Geography Faculty of Science, Masaryk University
Title of thesis:	Facilitating and Evaluating User Behavior In Virtual 3D Environments
Degree:	Cartography, Geoinformatics and Remote Sensing
Supervisor:	Mgr. Bc. Zdeněk Stachoň, Ph.D.
Academic year:	2022
No. of Pages:	134
Keywords:	virtual environments, virtual reality, 3D modeling, user interaction, user movement, movement analysis, eye-tracking, eye-tracking analysis

BIBLIOGRAFICKÝ ZÁZNAM

Autor:	Pavel Ugwitz Geografický ústav Přírodovědecká fakulta, Masarykova Univerzita

Název práce:	Zprostředkování a vyhodnocování uživatelského chování ve virtuálních 3D prostředích
Studium:	Kartografie, geoinformatika a dálkový průzkum Země
Školitel:	Mgr. Bc. Zdeněk Stachoň, Ph.D.
Akademický rok:	2022
Počet stran:	134
Klíčová slova:	virtuální prostředí, virtuální realita, 3D modelování, uživatelská interakce, užvatelský pohyb, analýza pohybu, eye-tracking, analýza eye-trackingu

Abstract

Conducting experimental research while considering the nuances of the studied phenomena & measurement tools has provided the case for the rigor of scientific methodology ever since there was a notion of science.

Research methods can be categorized as basic and applied ones, qualitative and quantitative, explorative and verificative. Another way of separation lies in considering the research domain – be it physical space (the surrounding space, the micro or the macro), the intangible realm of the human mind and its constructs, or perhaps the realm of arts and imagination.

One of such research domains is the virtual space, virtual environments. Why study them? Why now? They have *sort of* been around since the time of early 20th century optical illusions, the establishment of computer graphics as a science in the 1970s, or the failed, arcade-ey virtual reality revolution of the 1990s. These periods have spawned plenty of research papers. However, virtual reality did not account for much during these times, generating mostly basic research and academic studies of little utility. Then, from 2013 on, a new era of virtual reality interfaces and computational devices dawned. This coincided with the advancement in other fields; from a broader perspective, this follows suit of the fourth industrial revolution. It is now relevant and needed to be able to conduct research in virtual environments; as in many subjects, the pros of its use simply outweigh the cons.

This dissertation explores the structures of virtual 3D environments to operationalize and analyze them. The research provides tools to measure human behavior, along with discussing & demonstrating their use. Such is the purpose of this thesis – to further the methodology in conducting behavioral research in virtual environments, along with going on a limb of striving to understand the features and behavioral implications of the environments themselves, as per picking up the hints exhibited by users behaving in said environments and searching for generalized conclusions.

Abstrakt

Jakákoliv výzkumná/experimentální činnost musí pro zachování své vědeckosti, verifikovatelnosti a replikovatelnosti zohlednit specifika studovaných fenoménů a zvolených měřících nástrojů.

Výzkumné metody lze tradičně členit na základní a aplikované, kvalitativní a kvantitativní, explorativní či verifikativní. Další způsob kategorizace spočívá v zohlednění domény, v níž má výzkumná činnost probíhat – ať již jde o fyzický prostor (žitý prostor bezprostředního okolí, či mikroprostor a makroprostor), imaginární prostor lidské mysli a jejích konstruktů, či třeba prostor produktů lidské činnosti a kreativity.

Jednou takovou nehmotnou výzkumnou doménou je virtuální prostor, virtuální prostředí. Jaký přínos spočívá ve výzkumu virtuálních prostředí, a proč tyto zkoumat právě nyní? Lze namítnout, že virtuální prostory byly ve hledáčku vědy již od dob optických iluzí, popsaných na počátku dvacátého století, etablování počítačové grafiky jako vědního oboru v sedmdesátých letech, či od doby selhané, arkádovité "revoluce virtuální reality" let devadesátých. Za tuto dobu bylo jistě publikováno mnoho výzkumů; lze ale namítnout, že šlo převážně o základní výzkum či počiny s omezeným aplikačním potenciálem – vzhledem k omezeným množnostem tehdejšího počítačového hardwaru. Zlomovým se stal až rok 2013, kdy došlo k vydání prvních novodobých zařízení pro virtuální realitu – tento milník koincidoval s inkrementálním vývojem zobrazovacích a ovládacích zařízení, či obecně, s nástupem čtvrté industriální revoluce. Tato novodobá aplikabilita otevírá možnosti (a nutnosti!) výzkumu ve virtuálních prostředích.

Tato práce exploruje zaužívané struktury virtuálních 3D prostředí, vykreslovaných v reálném čase – tak, aby tyto bylo možno operacionalizovat, analyzovat, či využít. Provedené experimenty poskytují implementace nástrojů pro měření lidského chování, vč. zohlednění možností užití a verifikaci funkcionality na předkládaných experimentech samotných. Cílem práce je posun v metodologii behaviorálních výzkumů ve virtuálních prostředích – a to jak na úrovni reportování a observace chování manifestovaného uživateli, tak s náznakem k parametrizaci a porozumění samotných prostředí, která by mohla zpětnovazebně determinovat chování uživatelů.

Acknowledgments

This is the place to thank my advisor for creating a working environment suitable for doing decent science and publishing papers – with such support, things like writing up this dissertation followed through easily.

Another set of thanks goes to my co-authors and team members. As long as there was a value exchange where we the involved people got *something* out of the interaction, that was, likely, worth some.

I would also like to acknowledge the involved research grants or projects that aided along the way or followed similar research interests. Namely:

- The *Geoexcelence* stipend (Faculty of Science, Masaryk University), which financed my research.
- The grant from the Czech Science Foundation agreement No. GC19-09265J called *"The influence of socio-cultural factors and writing systems on perception and cognition of complex visual stimuli"*, which funded the second and the third publication (see Annex 2 & 3).
- The grant from the Technology Agency of the Czech Republic No. TL02000103 called *"Cognitive psychology and space syntax in virtual environments for agent modeling"*, which utilized the technical solutions presented in the third and fourth publications and funded the fourth publication (see Annex 3 & 4).

Original publications and authorship contribution

The body of the thesis is composed of four primary papers listed below:

Paper I. (Annex 1 to this doctoral thesis)
Ugwitz, P., Herman, L., Stachoň, Z. (2021). 3D visualization of historical buildings: Methods, their utility in the tourism industry and beyond. *Regionální rozvoj mezi teorií a praxí*, 2021(1)

Conceptual workflows mentioned in this theoretical paper were realized and put into writing by Pavel Ugwitz – authoring 90% of the text (excluding chapters 1.3 and 1.4).

Paper II. (Annex 2 to this doctoral thesis)
Ugwitz, P., Juřík, V., Herman, L., Stachoň, Z., Kubíček, P., Šašinka, Č. (2019). Spatial analysis of navigation in virtual geographic environments. *Applied Sciences*, 9(9). https://doi.org/10.3390/app9091873

The software solution provided in this study was created by Pavel Ugwitz. The statistical analysis and 50% of manuscript writing is also his work (excluding the introduction and parts of methodology & discussion).

Paper III. (Annex 3 to this doctoral thesis)
Ugwitz, P., Šašinková, A., Šašinka, Č., Stachoň, Z., Juřík, V. (2021). Toggle Toolkit: A Tool for Conducting Experiments in Unity Virtual Environments. *Behavior Research Methods*. https://doi.org/10.3758/s13428-020-01510-4

The software solution provided and described throughout the study was produced by Pavel Ugwitz. So was manuscript writing (minus the revisions).

Paper IV. (Annex 4 to this doctoral thesis)
Ugwitz, P., Kvarda, O., Juříková, Z., Šašinka, Č., Tamm, S. (2022). Eye-Tracking in Interactive Virtual Environments: Implementation and Evaluation. *Applied Sciences*, 12(3). https://doi.org/10.3390/app12031027

The provided software (minus the 3D modeling) was carried out by Pavel Ugwitz, as was 90% of manuscript writing (excluding chapter 3.4) and 60% of the revisions.

Other works related to the primary papers are mentioned too (see Annex 5).

Contents

I Thesis definition 1

1 Introduction 2
- 1.1 Thesis conceptualization . . . 3
- 1.2 Structure of the dissertation . . . 5

2 The current state of the research field 6
- 2.1 Visual variables and structures in 2D and 3D, graphics engines 6
- 2.2 The utility and validity of virtual environments . . . 9
- 2.3 Experimental data collection & virtual interaction . . . 13
- 2.4 Spatial strategies and algorithms . . . 17

3 Employed methods and reached goals 21
- 3.1 Research methods . . . 21
- 3.2 Overview of the published papers . . . 24
- 3.3 Supported research . . . 25

4 Discussion 27
- 4.1 Results . . . 27
- 4.2 Future work . . . 30

Literature 34

II Appendices 43

Annex 1: Paper I 45

Annex 2: Paper II 66

Annex 3: Paper III 89

Annex 4: Paper IV 101

Annex 5: Related publications and sources 132

Annex 6: Curriculum Vitae 134

Part I

Thesis definition

Chapter 1

Introduction

In retrospect, the 2010s were a decent decade – at least from the perspective of a person interested in computer-generated imagery (CGI) and human-computer interaction (HCI). We experienced wide adoption of commercial-grade virtual reality head-mounted displays (VR HMDs) of reasonable visual fidelity from 2013 on, along with specialized VR controllers, from 2015 on (Gackenbach & Bown, 2017). In addition to this, computer graphics performance increased over tenfold, e.g., from R9 7970 to GTX3090, utilizing programmable graphics shaders (Boreskov & Shikin, 2017). The increase in graphics rendering power also facilitated the introduction of *physically based rendering (PBR), ray-tracing, photogrammetry, smart geometry/materials, and other visual fidelity technologies* (Figure 1.1). 3D graphics engines have opened up to third-party programmability and customizability.

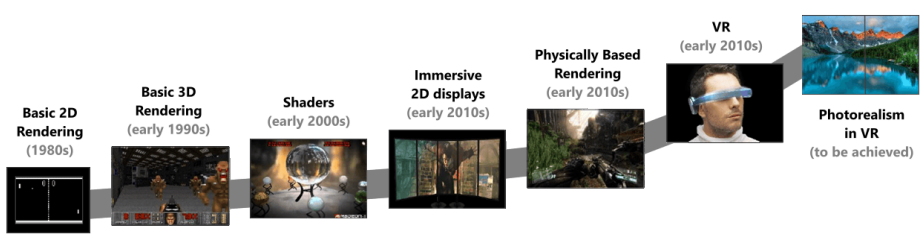


Figure 1.1: The evolution of visual experiences through time – image borrowed and adjusted from *AMD Liquid VR* presentation (Nelson, 2015).

To be clear: lousy CGI programming & modeling implementations are still lousy implementations – something to be best avoided. However, the advances over the decade have expanded the window of possibility to new heights & prevented some of the lousy by providing fool-proof industry-grade tools of high visual fidelity, technical art utility, and algorithmic optimizations (Figure 1.2). And as it goes in scientific (r)evolutions, the new and advancing technology also opens up new frontiers of research possibilities – *either by refining known methods in measurement accuracy, speed, costs; by providing new ways of doing things; or by enabling the execution of basic research concepts theorized previously only on a paper, where the idea had been waiting for the technology to catch on to make such ideas implementable.*

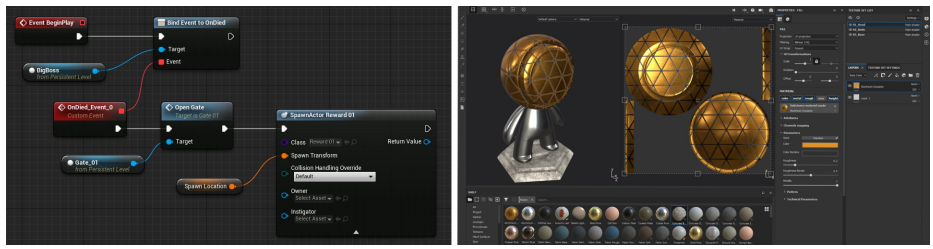


Figure 1.2: The progression in CGI workflows and 3D engine accessibility has spawned tools that allow for rapid, efficient development while taking care of low-level manual processing. On the left: *Unreal Blueprints* – visual scripting interface for programming without coding or for reusing existing scripts as nodes (Romero et al., 2022). On the right: *Substance* – material/texture painting and authoring tools with a real-time 3D interface (Adobe, n.d.).

The new hardware devices and their 3D engine counterparts have spawned, among others, research studies involving the following methodologies or concepts: *controlled 3D/VR experiments, user controller and interaction studies (be it both in a physical and virtual sense), visual fidelity evaluation, understanding of (virtual) space*. Along comes this thesis: a contemplation on operationalizing structures of virtual space with a further intent to streamline user testing in virtual environments (interaction methods), to provide implementations and methodical insights into the subject (user behavior data logging, data evaluation, and data visualization), and to foreshadow future work on the algorithmic evaluation of spatial compositions.

1.1 Thesis conceptualization

The initial scope of this thesis is to look into the ways of creating virtual environments in contemporary real-time GCI – to recognize the experimental possibilities virtual environments offer. Such propensity for experimental research is then described and operationalized. Based on the operationalization and subsequently proposed software architecture design, implementations are provided to release a working set of analytics and tools for user-centric research in virtual environments, to facilitate related user behavior and interaction experiments of our own, and to provide our solutions to other researchers as free software.

Secondarily, the body of this work also hints at a discourse of two methodics: that is, data acquisition and visualization of human behavior in virtual 3D environments (descriptive, observative, interaction-driven) and algorithmic evaluation of such 3D environments (evaluative, heuristic-driven). Subsequently, these two approaches can be brought together & weighted against each other to find ways of bridging and enriching them through a discourse the comparison creates.

While several virtual environments were created in the ongoing research the thesis covers, their visuals and semantics are secondary. So are any experiments that had occurred inside such environments. The thesis intends to design, implement, and provide tools to measure and evaluate common aspects of human behavior in virtual environments, setting the following specific goals:

- To provide a review of 3D virtual structures, virtual environment creation process, 3D object definition, and operationalization. Subsequently, to understand good practices of environment creation.
- To provide functionality for registering and collecting data produced by user behavior, interaction, or autonomous virtual scene actors.
- To facilitate ways of user interaction, be it passive forms of interaction like movement and spatial navigation, or active interactions where the user can change the properties of the environment and other actors in it, or vice versa. To consider the uses of input hardware.
- To offer ways of interpreting exhibited behaviors and acquired data – be it statistical analysis, 2D visualizations, 3D visualizations, or further interpretative or corrective algorithms.

Scientific papers based on such thesis goals would be hardly publishable. Therefore, the published papers (see chapter 3.2 for their content and contribution) are vertical slices of the aforementioned goals – as in, the papers progressively extend the required software implementations or expand on the theory as they delve deeper into the subject (e.g., basic forms of user interaction like movement, to specialized forms like manipulating objects through exhibiting physical movement on the user-side). To depict how the goals of the thesis were met per individual papers (Annex 1-4), see Table 1.1 below:
The concrete, tangible outcomes of the dissertation are the tools and their use: *the software designs, their implementations (scripts, algorithms), the demonstrations and recommendations of their use, their algorithm adaptability, and their inter-usability with other implementations.* If behavioral research in virtual environments is intended, this dissertation can serve as one of the go-to resources explaining the why's and how's of experiment setup, data collection, and subsequent analysis.

The user-side of the dissertation is thoroughly explored and published. The algorithm side is much less fleshed out – partially considered in the last of the four primary papers attached (Annex 4), partially covered in the discussion as a potentiality for future work.

Table 1.1: Thesis goals and how they were met per individual papers (the amount of color saturation signifies the intensity of goal covered)

Goals / papers	Annex 1	Annex 2	Annex 3	Annex 4
3D structures, representations, and related theories	3D engines, pipelines, mesh structures, visual variables			Object size, shape, surface for eye-tracking. Considering spatial metrics.
Data logging		User movement	User-environment interaction	Eye-tracking
User interaction		Hardware interfaces considered	Implementation and consideration of possibilities per existing engine API	
Data evaluation and interpretation		Movement patterns and statistics, movement visualization	Modeling interaction diagrams	3D visualization of eye-tracking and movement, corrective and eval. algorithms

1.2 Structure of the dissertation

The first chapter provides an introduction to the scope of the thesis – the fact that it covers research tools in real-time CGI purposed for behavioral research; that the dissertation follows two subjects that are to converge at the end: *human behavior in virtual environments, and algorithmic calculations in said virtual environments*. Following research questions are asked to be explored: *how do humans interact with virtual objects, compositions, and environments; and what are the computational properties of said objects/environments?* In their particular results, these questions can convert yet again: *does a human activity with/around elements of virtual space correlate with computational importance of said parts?*

The second chapter is a literature and research overview of the existing research fields (semantics and operationalization of virtual shapes and spaces, experimental research in VR, eye-tracking in VR, spatial navigation & decision-making strategies).

The third chapter references the theoretical foundations of this research. Software architecture and software implementation is covered, and so is an introduction to the possibilities of statistical analysis and spatial algorithms worth pursuing. The scope of the four published papers is mentioned, and so is supported research (published elsewhere, with the software solutions covered in this thesis serving as a driver to produce viable experimental setup and results).

The fourth chapter, discussion, serves as a critical reflection of the content covered in the four published papers, along with elaborating on related subjects not mentioned in the papers. Follow-up research prospects emerging from the areas uncovered by the presented studies are considered.

Chapter 2

The current state of the research field

The existing body of knowledge on the subject of user behavior and algorithmic evaluation of virtual 3D environments is merely a loosely coupled collection of inter-related topics – e.g., *spatial cognition and perception, environmental studies & the psychology of space; visual variables, the decomposition of imagery in 2D and 3D, the evolution of 3D representations of imagery, 3D modeling; computational geometry, ray-tracing and real-time 3D graphics pipelines; software architectures and software design*.

Unfortunately, there is no elegant overlapping theory to serve as an umbrella term to interconnect these – the connections between the fields will have to come out in the text by emphasizing how the individual fields serve the bigger picture and interrelate.

2.1 Visual variables and structures in 2D and 3D, graphics engines

Per (carto)graphic theory (Bertin, 1967), the contents of imagery can be decomposed into seven visual variables: *size, shape, color hue, color value, color saturation, orientation, pattern*. Per physiology of visual perception and cognition (Goldstein, 2013), humans identify edges, followed by constructing a shape, giving the shape a form (and obtaining a meaning based on comparing the shape to a library of known shapes) to place it into a context of spatial relations of its surroundings.

There are several caveats with both. First, human visual cognition works like a non-deterministic probabilistic, knowledge-based machine (depth perception is not perceived by the eyes but constructed in the visual cortex), which thus does not give a good starting point on how to model and process 3D space algorithmically. Second, classic cartographic visualizations are static 2D imagery; therefore, these consider neither the issues brought about by the third dimension (Häberling et al., 2008) nor the issues introduced by dynamic/interactive visualizations (Roth, 2013). While the visual variables stand, other issues arise: *object irregularity, where any shape but a sphere changes appearance when rotated; object occlusion, where depth, object placement, and observer position/rotation can cause an object to be hidden by others; navigation in complex 3D spaces using (sub)optimal user interfaces*. As graphics pipelines have gained the ability to post-process and continuously dis-

tort the rendered imagery (Doppioslash, 2018), feature-matching or variable-matching also becomes problematic.

In computer graphics, two distinct approaches to rendering 3D scenes have evolved through the years: the pre-rendered and the real-time approach. With pre-rendered graphics (*think marketing visualizations of products, computer-generated graphics in film, etc.* (Birn, 2013)), there is no constraint on performance, as each individual frame/shot can be rendered for the likes of hours. However, with real-time graphics, the situation is different: only a few (tens of) milliseconds are the budget allocated for a single frame rendering (Akenine-Moller et al., 2018).

Because of this, the repertoire of representations of 3D graphics in real-time is limited to computationally non-intensive forms only. Mathematical curve-based shapes (Schneider, 2014) are nonexistent, and ray-tracing technology has been introduced in a simplified form (Lengyel, 2020). As such, real-time 3D geometry is captured in the format of 3D polygonal meshes (*vertices, edges, polygons*). 3D modeling is based around meshes – be it either traditionally, by manipulating the mesh structures (Vaughan, 2011) or lately by introducing organic modeling (Gladilin et al., 2020), which merely builds its tools on top of existing mesh topology in a performant-enough ecosystem that allows for real-time rendering of organic-resembling structures (which are still based on the mesh structures anyway).

Surfaces of 3D real-time objects are represented as materials. The material consists of several textures (*albedo for basic color, metallic/roughness for reflections, ambient occlusion for shadows, normalmap for light reflection, heightmap for depth simulation, transparency for masking out the invisible, etc.* (McAuley et al., 2013)). The material/textures are stored as 2D imagery and represented as a set of multiple partial projections onto a 3D object. Because real-time scenes can be dynamic and real-time objects can be used in multiple scenes with different lighting/ambient settings, the multiple textures of a material specify how the material is to react to in-scene lighting regardless of the current lighting setup. This is called physically-based rendering (PBR) (Figure 2.1). As lighting, other visual properties, or positioning of items and structures in a virtual environment can dynamically change, this also changes the scene's visual resemblance and, subsequently, its visual evaluation. As for materials based on specific graphics shaders (Figure 2.2), these involve custom, purpose-specific mathematical transformations that can transform the colors, texture, or even shape of a material (*shaders can involve the likes of ocean waves or object disintegration animation*). Therefore, the interpretation difficulty in analyzing virtual visuals per the cartographic theory depends on the complexity of the object that is being analyzed: schematic, flat objects like evacuation signage (Snopková et al., 2021; Bořil, 2020) are easier to consider compared to objects using PBR materials or complex, dynamic shaders like water bodies or animated foliage (see Table 2.1).

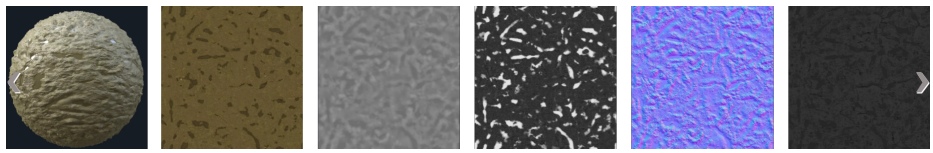


Figure 2.1: An example PBR material of mud (Sharetextures, n.d.). The different texture channels include *albedo (color), heightmap, glossiness map (water reflection), normal map, and specular map (general material reflection)*.

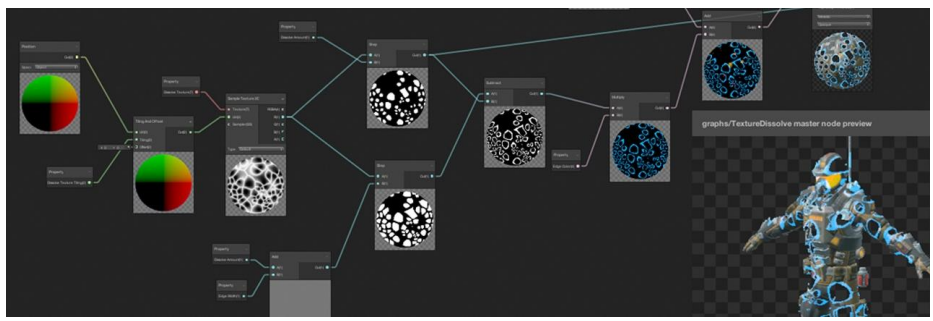


Figure 2.2: Example dynamic texture shader generated in Unity ShaderGraph (Cooper, 2018).

As for 3D mesh structures (objects), their operationalization is unclear. A mesh structure does not have any prescription regarding its size, length, width, scale, detail, consistency of detail (texel and vertex density), etc., prescribed to it. Per visual cognition and linguistics, some objects have well-established boundaries (e.g., man-made items, tools, furniture, objects separated from the earth). Other objects of complex structures, size, or incorporation into other objects/forms have much less clear boundaries (e.g., land structures fused with the earth, groupings of objects). For a virtual mesh-made 3D object to follow a clear separation is merely a good practice or an intent, not a certainty.

Table 2.1: Object interpretation complexity per object shape/texture features

Object feature	**Consequences**	**Interpretation**
Image texture	Color/texture stays the same.	Simple.
PBR material	Reacts to lighting sources and angle. Change in coloring (lighting), small changes in perceived shape (normal map).	Minimum interpretation differences in consistent lighting. Worse with dynamic lighting.
Animated object	Change in shape/size in time (position, deformation).	Depends on animation. None to significant changes in time.
Shader material	Specific mathematical transformations. Can change color and texture (fragment shader), shape (vertex shader).	Depends on implementation. None to significant changes in time.

This is to show that evaluating user perception, cognition, or interaction with a virtual scene or its objects has many caveats that are to be considered and potentially further worked on. Regardless, these aspects of real-

time 3D graphics are incorporated into 3D graphics engines – e.g., Unity or Unreal Engine[1]. The engines are open, to an extent, to further extendability by a scripting/programming language – e.g., C#, or C++. The openness to industry-standard 3D modeling tools (e.g., *Maya, Blender, Zbrush, Substance*) is a must (Gregory, 2018), although the workflows of 3D scene creation and composition can vary.

The first attached paper (Annex 1) covers select issues of working with contemporary virtual 3D environment forms and structures. Other workflows (i.e., ways of establishing, visualizing, and operationalizing virtual environments) are considered in the discussion (chapter 4).

2.2 The utility and validity of virtual environments

A prophetic quote from the 90's states that *everything that can be digital will be digital*. The question is, how does that tie into 3D virtual environment creation, the utility, believably, and experimental validity of such environments?

3D data structures, their representations, and related software

Remote surveying and 3D modeling allow for creating precise 3D representations of geographic environments and buildings that are far away, imaginary, prototyped, or yet to be built. 3D environments consisting of irregular, organic shapes can be digitized using SRTM, LIDAR, photogrammetry (Dedík & Minaroviech, 2017); geometric, man-made structures can be (re)created using CAD, Building Information Modeling (BIM) (Johansson et al., 2015), or traditional 3D modeling software.

Many cases of controlled virtual experiments take place in virtual representations of contemporary man-made indoor environments. Therefore, BIM data sources can be utilized to create such virtual 3D environments. BIM is a term that comes close to a buzzword. Despite it being coined in the 1970s, the involved parties provide their personalized definitions based on the specificity of their software solutions or research focus (Abdirad & Dossick, 2016). Still, the core of the term is as follows: *3D modeling software/tools with extra semantic information to it (materials, properties, segmentation, etc.), to be used by designers, engineers, architects, and the like*. This also means that BIM was pioneered and adopted by the architecture, engineering, and construction industry; this, however, does not invalidate it for other uses (Natephra et al., 2016) – such as taking it for a data source virtual 3D environments can be built upon and adopting it into such a workflow (Kvarda, 2020).

[1]Given the fact the development of current-generation graphics engine is very costly, there are not many 3D engines with adequate support, device compatibility, and extendability to choose from.

From the reductionist perspective of a 3D artist, BIM software (e.g., Autodesk Revit) is just another 3D modeling software with extra libraries of existing 3D models to it. From the reductionist perspective of a researcher interested in subjects like visual/spatial cognition, etc., BIM provides a viable source on which to base experiments. Furthermore, employing VR BIM user testing in applied settings makes sense in validation studies (as in designing new buildings or making alterations to existing ones).

A real-time 3D engine is needed to run and visualize VR scenes (Bamodu & Ye, 2013). Therefore, BIM models are exported to engine-readable formats. This can be done manually; however, with spacious models, this can turn into a repetitive and time-consuming task. For the Unity 3D engine, Tridify (Mezrag & Boton, 2021) has somewhat automated this task. Another similar solution is Unity Reflect (Faure, Prochazka & Menard, 2019).

To prepare BIM environments for some of our research, Autodesk Revit was used; the generated Revit IFC file format was then transformed using the Tridify web service and, as such, imported to Unity (Figure 2.3). Since there is only a partial overlap in the visualization technologies of Revit and Unity, other assets (PBR materials, sounds, 3D objects, etc.) were acquired and imported to Unity to produce experiment-ready versions of intended interior environments. For this, open-source libraries/applications were sourced (free texture depositories, Unity Asset Store, Blender, etc.)



Figure 2.3: BIM geometric building data, as imported into Unity (Mezrag & Boton, 2021).

As for outdoor environments/segments, organic shapes, or digitization of existing spaces, there is terrain modeling, procedural generation, and photogrammetry. Representations of contemporary terrains can be brought to virtual environments through existing GIS datasets (Mat et al., 2014), to be represented as a grayscale heightmap that is to be rendered as a specialized 3D object with a set polygonal distance. This is the traditional way of representing terrains (Foxglove, 2019) in 3D engines – it has been this way since the early 2000s. A newer approach to representing terrains, Voxel-based (i.e.,

three-dimensional matrices of terrain data, incl. vertical variance and potentiality for holes or terrain material layering), is not widely adopted yet.

Terrains can also be artificially generated – either traditionally/artistically by hand or by procedural generation software capable of applying realistic erosion algorithms and the like (WorldMachine, Gaea). A mixture of both workflows is also possible (Foxglove, 2019): from establishing hand-made general terrain shapes to adding detail by stamping the base shape with detail brushes based on real terrain data (Figure 2.4).

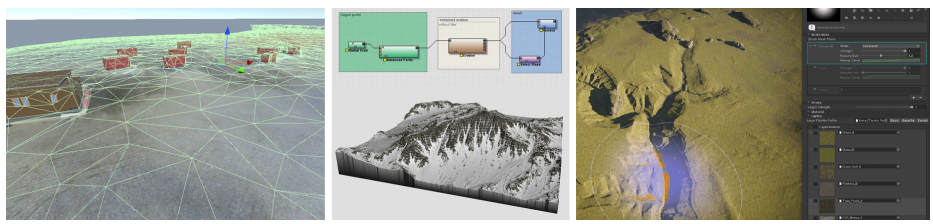


Figure 2.4: Approaches to generating 3D terrains: photogrammetry (left), procedural node-based generation (center) (Schmitt, 2022), manual terrain brush editing (right) (Liu et al., 2022).

Since nature is not fully deterministic or precise, vegetation and foliage can be placed based on a procedurally generated, prescription-based set of rules – e.g., by adding vegetation based on distance, density, altitude, random-generated noise, terrain slope, or terrain-based biome demarcations. This goes for generic detail objects of little significance; for significant objects (per environment design or experiment operationalization goals), these can still be placed by hand. If realistic, organic objects are desired, these can be either created using advanced 3D modeling workflows or sourced from repositories of photogrammetric scans (e.g., Quixel Megascans, Sketchfab).

Virtual environment visuals and validity

Working with BIM software to generate virtual representations of man-made environments is a matter of following a particular graphics workflow (e.g., using an architectonic modeling tool and a 3D engine conversion utility, as mentioned above). There are other viable workflows as well – *as this is merely a recommended or agreed-upon inter-use of related techniques and graphics applications*. Other workflows can also involve the creation of outdoor environments – be it nature or other irregular environments, based on geographic data or pure imagination. Such workflows are discussed in detail in the first paper (Annex 1). The question, for now, is: *what are the virtual environments produced by such workflows suitable for, research-wise?* I.e., can virtual environments hold their own in visual fidelity? Can we extrapolate virtual environment user behavior to retain ecological validity (Kothgassner, Felnhofer, 2020) in real scenarios?

A virtual environment of a certain level of visual fidelity and visual variable depiction (be it realistic or stylized visualization, accentuation of features, lighting and mood, etc.) influences users in a multitude of ways (Figure 2.5). Not only are they to understand the meaning of virtual objects (and the interactivity of said objects, if present) and spatial layouts for what they are – *effectively cutting out the confusion brought about by objects of abstraction or segments of seemingly unclear purpose* – but the overall impression of the virtual environment is also influenced, drawing the user in closer to their authentic, ecologically valid behavior, through immersion (Paljic, 2017). With virtual environment visuals that retain most of the visual cues present in the real world (Chalmers, 1952) and/or visuals that purposefully avoid the uncanny valley of computer graphics (Tinwell, 2014), ecological validity can be closed in. This, however, does not mean one has to resort to some perfect, rigidly optimized environments – on the contrary, virtual environments can introduce some visual clutter (Rosenholtz et al., 2007) and thematicity (Krampen, 2011), as real environments often do. The environments just have to follow the natural law (e.g., gravity, believable proportions, building static load capacity) and retain their context.



Figure 2.5: The "Medieval village" environment (Quixel Blog, 2021) demonstrates a realistic approach to real-time visualization while emphasizing atmosphere and visual storytelling.

Going beyond mere visuals, the overall sensory impression that is to make users believe they are in an environment resembling the real thing has to go the full circle. That is, the visitor of a virtual world need not be just a passive observer; on the contrary, they can actively participate in such environments – that is, to the extent said environment allows (e.g., locomoting, manipulating with virtual objects (with such objects providing some form of feedback), interacting with other users, etc.) Having a story, an activity, or something

that ties the user to the environment does help (Gorini et al., 2011). Furthermore, various state-of-the-art technologies help to raise such a level of fidelity of experience: realistic PBR lighting, including volumetric shadows, global illumination, physical rays & light bounces (Pharr et al., 2017), makes the users perceive the spatial depth of the virtual environment; surround audio engine with in-place effects (Poeschl, 2013) further accentuates this. As for the active component of locomotion, there is an inherent component to human kinesthetics (Kivunja, 2015) that, if captivated well by the controller interface (Caserman et al., 2018; Daria et al., 2018) – ideally to the point where the interface disappears from the conscious awareness of being explicitly felt by the user – lets the user easily fall into immersion, as there is little/no of a user metaphor they have to bridge (as opposed to using bridging interfaces without a glimpse of the motor action required in real-world scenarios – e.g., the keyboard). The kinesthetic experience also coincides with having a reliable (first-person) representation of one's body in the virtual environment (Waltemate et al., 2018; Creem-Regehr, 2015). All this, however, is an ideal scenario; realistically, immersion compromises need to be made due to implementation difficulty, resources, or input hardware limitations (Cuervo et al., 2018, Rietzler et al., 2018).

One can, however, still strive for sensory fidelity and ecological validity in a setup with less-than-ideal immersive capabilities (ideal being Matrix-like brain-computer interfaces (Tan, Nijholt, 2010) that entirely remove intermediary physical interfaces while fully saturating all the senses). Traditionally, the research in this field used to focus on incremental improvements of existing interfaces, real/virtual experimental group testing, and basic research proposing new interfaces; these implementations/interfaces were then field-tested and evaluated by having the users perform in isolated performance tasks and/or by administering self-reported HCI questionnaires (Lazar et al., 2017). This traditional approach still carries its weight, mainly on the physical interface side; on the virtual environment side, good practices of spatial comprehensibility and sensory satiation are key.

2.3 Experimental data collection & virtual interaction

The ability to create custom scripts and functionalities on top of an existing 3D engine became feasible by the late 2000s to the early 2010s. Before that time, most 3D engines (e.g., Unreal Engine 3 and prior, Source, id tech, and older) were closed-source, game-centric solutions with little to no custom scripting functionality or support for it – there was not much room to conduct advanced user studies in 3D[2] (Smith, Trenholme, 2009). Nowadays,

[2] Not without licensing an engine in order to access its source code (oftentimes hardcoded). That, or relying on limited customization features of an engine's level editor or its custom scripting language (e.g., the now deprecated Unreal Engine's Unreal Script).

however, it is feasible to facilitate data logging and controlled interactive events inside an engine, all the way to building custom research applications.

Virtual interaction data collection and interpretation

The basic form of user interaction in virtual space is movement (Roth, 2012). The movement of a user viewpoint (*rotation, locomotion*) can be represented by a so-called travel path (Bowman et al., 2005). As areas of a virtual environment are visited multiple times (be it by a single recurring user or many), heatmaps can be generated to determine movement density (Chittaro et al., 2004; Zanbaka et al., 2005). Area-based (contextual) or node-based (graph theory) segmentation of virtual space can help in predicting, directing, and validating such movement patterns across users. Traveled distance, the amount of rotation, or the number of turns can also be measured, and so can interface usage. When time (efficiency) and traveled distance are measured, the speed of a user's travel can be calculated (Herman et al., 2018).

Other behavioral results and observations acquired in virtual environments allow for examining the influence of an environment's visuals and user interface on human interaction with the virtual world. Variables and variable derivatives can be observed to get this – e.g., user movement speed, movement continuity, the effectiveness of user interface use (*as in experimental tasks where a user has to execute a series of interactions – what is the most efficient way of achieving the desired result, and what is the way a user had chosen*).

Both the quality and the quantity of acquired data can vary based on multiple factors: the features of the physical user interface (and the types of variables each input produces), the efficacy, accessibility, intuitiveness, and distinguishability of mapping such physical device to behaviors in the virtual environment, i.e., the choice and the implementation of interaction metaphors (Hrimech et al., 2011). Also to consider is the designed intent & difficulty of tasks to be executed in the virtual environment. In other words, there are vastly different interaction patterns to be expected while using, e.g., a keyboard, a VR motion controller, or a specialized-purpose interface like a flight simulator cabin. Regardless, manifested user behaviors in a virtual task/scenario can be aggregated into user strategies and analyzed. Should such an analysis be triangulated to see if there are relations among observed variables in users, complex streams of data collected in specific environments can be processed to reveal patterns in human behavior.

Specifics of VR eye-tracking

Eye-tracking in VR can be thought of as a specific form of user interaction, employing its own methodology. It needs to be stressed that the first commercial-grade VR eye-trackers were made available in 2016 (SMI, HTC, Varjo); therefore, this technology is relatively new. As mentioned (Lappi,

2015), the use of VR eye-trackers in virtual 3D environments has many challenges. A standard eye-tracker makes do with a user with a fixated head or body in an experiment that makes them observe a stimulus fixated in place; compared to this, mobile or dynamic stimulus eye-tracking studies tend to be more demanding on experimental control and data processing. VR eye tracking is even more complicated than that – a user with a moving viewport (head), moving body, moving physical controllers, whose movement is translated from a real environment to a virtual environment; thus, eye convergence projection focuses on virtual targets. This makes for a case with many variables to account for.

The existing studies on VR eye-tracking (Clay et al., 2019) have not considered such issues to an extent, keeping to conducting simple research designs.

Virtual interaction functionality

Real-time in-engine interactivity can be based on a simple trigger-event concept where an action originator (a user, a script, an AI entity) triggers an intended reaction on the object(s) of the environment. This functionality can be extended by nested objects and various object states or property changes. The software design patterns this is based on (*command pattern, state pattern* (Baron, 2019)) have been used in computer games and other applications in the past. This is the most common way of establishing interaction in the virtual world. Should artificial intelligence with an intended element of unpredictability be implemented for other interactivity originators than a user, decision-making trees with a factor of variability or randomness can be introduced. Regardless, the base principle of action triggering remains. As for human users, it can be argued that they prefer consistency in the user interface and the consistency of effects their interaction has on the virtual world (Malone, 1982) – as consistency in controls prevents their frustration or confusion with the interaction process.

Existing virtual environment interactivity experiment solutions (Grübel et al., 2017) provide interactions, logging, and evaluation scripts, along with facilitating physiological measurements, questionnaires, and movement tracking. Game-centric open-source solutions (Unity Technologies, 2017; Brookes et al., 2019; Vasser et al., 2017) also provide interaction systems, albeit only partially customizable. For VR controllers, VR UI, and small-scale object manipulation, VR interaction toolkits (Fuad, 2019) (Figure 2.6) can be used, depending on the hardware, vendor, and integrability with 3D engines.

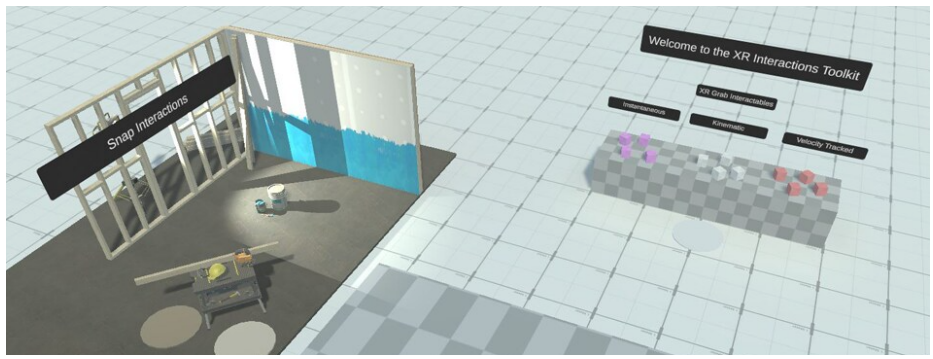


Figure 2.6: The sample environment of the Unity *XR Interaction Toolkit* (Fuad, 2019) facilitates user movement, teleportation, object grabbing, and object interaction. The manual (Unity Manual, 2022) provides more profound insight, including available templates, their implementation in the Unity Editor, and the programming extendability of this interaction toolkit.

What is the palette of ways of interacting with virtual environments? That depends on the accessibility and extendability of a 3D engine's application program interface (API) and the extension libraries likely provided by the vendors of custom physical user interface hardware (be it VR eye-trackers, racing steering wheels, or else). Namely, the following can serve as a means of interaction: *press of a discrete physical interface button, continuous interface gestures, movement in virtual space, interaction with virtual UI, interacting with AI characters or structures, etc.* The interaction responses can also be implemented per the programmability of the 3D engine, with some of the feedback including *video* (*change of textures, lighting, text, animations*), *position* (*virtual movement, teleportation, animation*), *haptics* (*object and physics manipulation*), *audio, a nested series of other events, etc.* Regarding a user triggering an event with the contents of a virtual environment or with other actors in it, this need not always be obvious, intended interaction, as interaction can serve as a (silent) facilitator for controlled/designed changes in the virtual environment. A classification of the aspect of intentionality and user awareness of interaction follows:

- Predictable triggers: Activation of a predictable trigger is intentional and causes an outcome pre-anticipated by the user. As an example, if a user sees a virtual object that resembles a light switch, it probably is a light switch; activating it will, in most cases, yield anticipated results (i.e., turning a light on). This corresponds with Gibson's classic theory of affordances (Volkoff, Strong, 1977).

- Undetected triggers: Such trigger is activated by the user's action; however, the user lacks the knowledge of where, when, or how the trigger is activated (likely a hidden, invisible trigger with no prior and, possibly, no posterior feedback). An example could be entering a restricted

area in the virtual environment followed by an alarm going off – where the environment gives the user no prior explication that triggering is to take place. Furthermore, such an alarm (to keep with this example) can be either an audible or a silent one.

- Uncontrollable triggers: An example of a trigger that cannot be (fully) controlled by the user's intention is a physiological reaction like pupil dilatation representing an involuntary symptom of increased cognitive load, which – if observed through linked eye-tracking hardware – can be implemented as a control mechanism for visual complexity reduction in the environment. This goes likewise for other biofeedback.

- Automated triggers: Activation of such triggers is entirely out of the user's influence, e.g., trigger script that periodically refreshes external sensor data and transforms these into visualizations.

2.4 Spatial strategies and algorithms

People perceive the surrounding environments (be it real or virtual ones) through the filter of their senses – be it visually or otherwise (Hadjiphilippou, 2013). Based on their acquired understanding of the space, they act, move, and interact in the environment (Downs, Stea, 1973). Equipped by their physicality (directly) or user interface (indirectly), they move about the environment or instill change in it.

While such a description stands on the side of basic research (as environments involve, e.g., known/unknown, indoors/outdoors, embedded with contextual tasks, functional constraints, societal norms and behavior expectations, etc.), this is the basis of it in terms of decision making and behavior: spatial cognition, spatial navigation (Denis, 2017). As virtual environments are artificially constructed and situated into virtual (computable) space, user movement can not only be observed (interaction), but based on this data coupled with theoretical knowledge, virtual environment creation can be adapted and analyzed algorithmically to provide intended results for users moving about such virtual spaces – thus going full circle: *either from user interaction with the pre-built environment to evaluation or from environment creation & pre-evaluation to subsequent user interaction*.

Understanding and estimating human navigation behavior

Human navigation strategies tend to follow a pattern of creating spatial knowledge by interconnecting observed (local) spatial nodes and integrating this knowledge to the (global) scale of the navigated environment – be it the former giving a sense of accessibility of a current area (node) leading

to other nodes, the latter providing a sense of general orientation and direction (Siegel, White, 1975). Spatial nodes can be thought of as crossroads or cross-sections or rooms, joined with other nodes along the paths or corridors that connect them; global navigation elements (global landmarks) are represented by significant, distinctive objects in the area (there can be local landmarks as well, albeit of a mere local significance). As the knowledge of an area increases, spatial knowledge extends by encoding the layouts as a sort of associated graph structures in one's memory, with graph vertices representing the nodes and graph edges representing reachable paths between connected nodes (Meilinger, 2008). However, specific circumstances like time pressure, blocked paths, or provided instructions tend to make humans prefer other strategies (Zhou et al., 2020). In unknown areas where previous spatial knowledge is lacking, inter-individual differences can arise as well. Furthermore, there is some inherent difference between outdoor/indoor navigation (Vanclooster, Van de Weghe, De Maeyer, 2016). In a situation where local navigation nodes are lacking (e.g., vast open spaces), global orientation by means of following global landmarks can serve as a source of general direction – to be set and to be adjusted on the go as local obstacles are encountered. Individual differences (e.g., previous memories, aesthetic preferences, cognitive style, or cognitive impairment) can determine why an individual prefers one possible path instead of another (Montello, Sas, 2006); path difficulty (e.g., terrain elevation and roughness) can also be a deciding factor. This entails the basis of human navigation strategies.

As for spatial navigation algorithms in 3D engines, there are two basic approaches to AI movement: the older pathnoding and the newer navmeshing (Tavakoli, 2019). These, however, operate on the same principle (e.g., Dijkstra's algorithm). The only difference being that path nodes represent a graph structure, while a navmesh represents a series of movement planes (similar outcome – just with a potential for motion refinement around an area). This (Figure 2.7), however, is a simple algorithmic "shortest path" solution that ignores the structures of an environment, rememberability of an environment, and the need for human declarative memory (Penn, 2003).

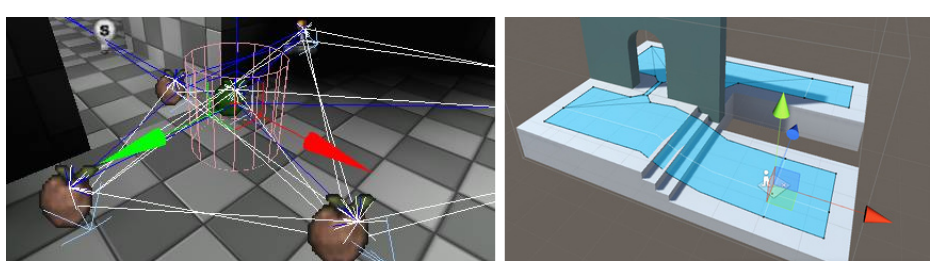


Figure 2.7: PathNode navigation, as defined in the legacy Unreal Engine 3 (left); NavMesh navigation, as generated in Unity (right) ("Unity Scripting Reference, 2021").

Operationalizing virtual space

Algorithms to analyze the spatial features and integration of an environment have been described in 2D analyses, per the Space Syntax theory (Hillier et al., 1976). These involve visibility analysis, interconnection (integration) analysis, distance analysis (per a specified origin point), etc. Similar algorithms have not been widely adopted and implemented in the context of 3D – and criticism is relevant here: analyzing mere 2D floorplans of inbuilt spaces while ignoring existing three-dimensionality of some spaces (*be it most elevated outdoor spaces, or just height-differentiated built spaces like foyers across multiple floors, platforms, stairs, or parts of buildings made of transparent materials*) provides mere simplified results. Algorithms of spatial analysis in 3D do, however, bring about two issues: one of implementability of complex spatial algorithms (which can be facilitated by the 3D engine's API, to an extent), and the second of visualizability of given results.

The existing research in 2D image processing provides some interesting algorithms that can be of use – namely, visual saliency analysis (Zhai, Shah, 2006) or visual clutter detection. While the idea of employing the same on 3D mesh-based geometry in real-time engines is a challenging notion (Alexiou, Xu, Ebrahimi, 2019), such implementations may prove helpful in evaluating the features of virtual environments and the variance of these while moving about the environment.

Some image-processing or mesh-processing algorithms can be executed on the environment as a whole; some of them can be executed on single, separated objects (and then compared to other objects or placed into the environment as a whole to test for isolated/holistic differences). For image processing, visuals of a virtual 3D scene can be rasterized and evaluated (processing the information in 2D, per chosen camera angles); for mesh processing (Lavoué, Larabi, Váša, 2015) this is a matter of computational geometry. While the applicability of this line of research leaves a lot of open questions, it can provide interpretation value on the features and feasibility of virtual environments.

If individual 3D objects are to be analyzed separately, this makes for more straightforward implementations and evaluations compared to object compositions in a virtual scene as a whole – as isolated objects are not occluding other objects, as there is no variance in virtual scene lighting, reflections and postprocessing effects influencing the perception of the presented visuals. 3D mesh processing algorithms can determine the likes of edge detection, detail density, sharpness, convexness, and object irregularity (Corsini et al., 2012). In contrast to that, more "soft" or subjective metrics are difficult to impossible to express by means of an algorithm – e.g., how does one even quantitatively determine the likes of perceived object beauty (Bergen, Ross, 2013)? Even while considering the feasible algorithms only, another open-ended question

is the interpretation of given results – how to aggregate, summarize or classify them? This is another research area with many unknowns.

Chapter 3

Employed methods and reached goals

3.1 Research methods

For the most part, the Unity 3D engine was used to assemble 3D visualizations, conduct experiments, and collect data. The use of this engine in itself constricts and defines other solutions that are to be used, as it employs its own standards – e.g., *the C# programming language, the 3D pipeline, and the format of the virtual 3D geometry that can be used with it, a way of instancing experimental scripts as runtime objects, and the outputs it produces*.

In addition to Unity scenes, scripts, and plugins, third-party scripts (Processing visual language scripts, statistics in R) were created to aid with data cleansing, visualization, and statistical analysis. As for generating 3D objects, the graphics pipeline is quite open, as is today's standard: Blender was used for authoring hand-made 3D models, Meshroom for photogrammetry. To further evaluate 3D space and objects, some spatial computations were proposed or implemented in Unity (see Annex 4).

This puts the accent on the "how" of doing things; the "why" is secondary, as the current trends in the 3D industry determine the methods of representation of 3D models that are created in modeling software and accepted by the 3D engine. Not to mention, the publications with user studies in them (Annex 2, couple of derived studies based on Annex 3) have their own small human-centric research questions to ask and explore. Regardless, having supporting theories (e.g., *human-computer interaction interface design or visual cognition perception of shapes/environments, navigation in space, etc.*) to compare and reference research findings against is crucial so as not to produce ungrounded science.

3D data, 3D software & hardware

All the 3D object models used are 3D meshes stored in the industry-standard FBX format (Wikimedia Foundation, n.d.). They are either sourced from openly available repositories, such as the Unity Asset Store, or modeled using Blender. The definition, quality, and detail of such meshes are set to follow the standard of each individual study in question; this, however, still requires

adhering to the contemporary standards in 3D graphics (be it in *polycount, geometry, topology, textures, or materials*).

Some other sources of 3D meshes and structures that were also used include LIDAR ground scans, Photogrammetry (Jones, Church, 2020), BIM data (Murphy, McGovern, Pavia, 2009), and procedural geometry. Expanding my skill set to the use of these was to ensure versatility in the 3D modeling craft.

As for VR headset eye-tracking extensions, the implementations used in the publications have used the *Pupil-labs VR/AR* (used in Annex 3) and the *Tobii VR* (used in Annex 4). The former is an installable addon to the HTC Vive Pro VR headset; the latter is included in some existing production-grade headsets (e.g., HTC Vive Pro Eye, Pico Neo 2/3 Pro Eye). The *HTC Vive Pro Eye* provided significantly better reliability than the Pupil Labs solution; both solutions came with their own software and Unity plugins that were adapted per experiments' needs.

Both scene composition and programming of runtime scrips/functionalities were done using the Unity 3D engine (Unity 2017 to 2020, long-term support (LTS) versions). As the need to create new virtual environments would arise, a generic 3D modeling production pipeline was followed: *from gathering references/requirements to prototyping an environment, to modeling the structures, to postprocessing and polishing the environment* – as presented in the first paper (see Annex 1), as can be further adjusted or expanded per the needs of research intents.

The 3D hardware is considered on the basis of experimental needs. Standard keyboard and mouse input interfaces were used, and so were VR headsets.

Software architecture of runtime implementations

As explained in the discussion (see chapter 4), the human-centric software implementations introduced in the three papers with elements of user behavioral data capture & evaluation in them (see Annex 2, Annex 3, Annex 4) were designed to work either together or separately of each other (provided there is no feature dependence). The implementations in question have open interfaces that can be reused or adjusted by other scripts or projects.

The implementations are written in C#, along with utilizing existing Unity API functions and its hardware vendor extensions ("Unity Scripting Reference, 2021"); this is necessary for 3D computations or interaction with other native components of the Unity ecosystem. The use of the Unity API synergizes the use of the Unity Editor's Inspector UI to ensure adjustability and reusability of script variables, even for non-programmers. Figure 3.1 explains how the implementations tie into each other. The idea behind figure 3.1 is program logic separation into two distinctive runtimes: *experiment runtime*, to be employed when the users are interacting with the environment

(to produce behavioral data), and, optionally, *evaluation runtime*, to aid a researcher in verifying or visualizing the acquired spatio-temporal data back onto the virtual environment.

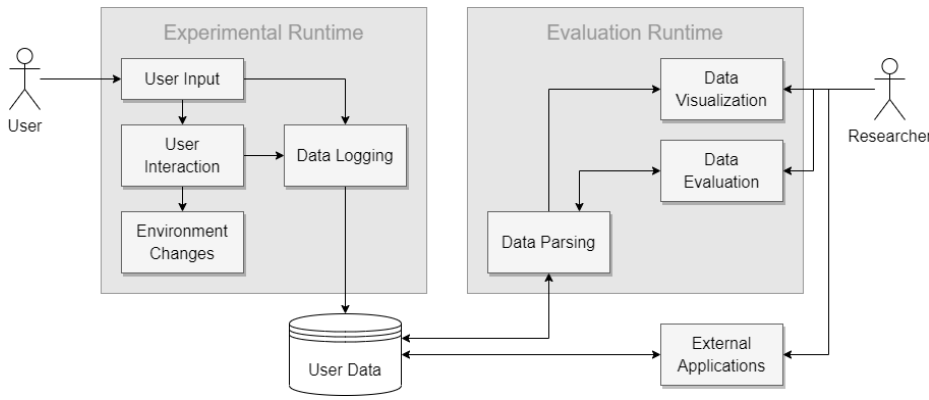


Figure 3.1: A simplified component diagram with user behavior and interaction and components in it. A universal logger implementation can make use of these to write outputs (CSV datafiles here), which, in turn, can be processed by external software or loaded back into Unity to further data processing, evaluation, and visualization.

Software design was outlined and documented using class diagrams, component diagrams, and data flow diagrams (Gomaa, 2011). When process modeling was required, e.g., for conducting experiments with the environment actively directing and/or reacting to the user (see Annex 3), state machine diagrams or the BPMN diagrams were used to model out such interactions. Use case diagrams were not necessary, as the implemented experiments to date were single-user – leaving only one class of "experimental user" human entity type interacting with the runtime of the application.

External programs, statistical analysis

On their output side, the provided Unity scripts produce text files in CSV format. These can be reused by third-party databases, scripts, or applications for further data processing or analysis. To be able to visualize and aggregate movement/interaction data in 2D, an external Processing application was implemented (Ugwitz, Stachoň, Herman, 2019); GIS software was also used for similar purposes (Snopková et al., 2021). To visualize similar data in 3D (while considering the many pros and cons of displaying in 3D), this was done in the 3D engine itself (see Annex 4).

Irregular data with a propensity for error (e.g., eye-tracking) had to be filtered using Processing implementations. Custom R scripts were used to find patterns in data and to compare the calculated data to demographic data and other sources like standardized and/or self-reported questionnaires that measure psychological constructs. For an object-based evaluation of eye-tracking

data (Ugwitz et al., 2017), the Processing algorithms were further expanded to produce per-object and per-object-category statistics (e.g., revisits, total fixation time, scan path).

3.2 Overview of the published papers

Four primary papers were published – three of which appeared in international impacted journals, and the other one was published locally. The papers are supported by other research that uses their technology or methodology (see chapter 3.3). Other papers to follow where the last paper (Annex 4) left off were ideated but not realized at this point; as these ideas, however, provide a potential to validate further and expand the ideas present in the thesis, these are considered in detail in the discussion (chapter 4).

Annex 1: Review imagery structures in 2D and 3D

The intent of the paper was to reflect the ways imagery (de)composition is understood and defined traditionally in the cartographic sense (Bertin, 1967), how this transformed into 3D (Foley et al., 1996), and how it is implemented in contemporary real-time 3D graphics engines (Lengyel, 2020). To know how 3D representations of objects are created and to be aware of the formats they are stored in.

While this paper is arguably the least important one (a review of contemporary technology and 3D industry workflows), other goals were dependent on this – e.g., choosing a 3D engine, producing 3D objects, composing them into iterated and user-tested 3D scenes, and understanding their topology.

Namely, in-engine modeling tools were used (Unity ProBuilder), as was external modeling software (Blender). Other 3D model-authoring sources such as photogrammetry were considered, and the visual level of stylization/realism was discussed.

Annex 2: User virtual movement and data logging

This writeup was to consider the most common PC hardware user interfaces for navigating in virtual 3D environments, human factors, how users move in a virtual environment, and the ways such data can be collected & interpreted.

A virtual outdoor environment of about half a square kilometer was built with the intent to utilize it for a user study. The users were asked to explore the environment in a semi-structured way; this produced a small dataset that was subsequently checked for both individual and group differences (explorative experiment: 20 participants, PC vs. VR interface use, males vs. females).

The scripts for user movement data logging (Unity, producing CSV outputs) and basic movement pattern analysis (Processing 2D visualizations) described and applied in the paper were made available. Subsequent statistical analysis demonstrated another way of dealing with this data, considering the primary movement metrics and their derivates, along with controlling for external variables.

Annex 3: User virtual interaction

Moving on from passive forms of interaction like walking around a virtual environment or merely observing, the purpose of this paper was to consider and typologize forms of interaction where the actors in a virtual scene can actually make alterations – be it to their own or to the properties of their surroundings.

The taxonomy of interaction types and interaction triggers is discussed, and so is our software design and implementation for interactive environments. The software implementation is made available.

Annex 4: User virtual reality eye-tracking

Eye-tracking is a bridge between the abyss of aware and unaware – at least in the context of visual cognition and perception (Popelka, 2018). As such, this form of user-environment interaction deserves extra attention. As users are observing 3D objects and space, patterns are to arise – this also ties into Annex 1 and potential future work that is to consider employing algorithms that aid in analyzing the features of virtual spaces.

Eye-tracking in virtual reality offers a fully isolated field of view and depth perception; it, however, poses significant methodological/implementation challenges (Lappi, 2015). These were discussed in the study, along with providing example implementations for many of the perceived challenges. The issue of data analysis of VR eye-tracking data was also touched on. The eye-tracking code snippets were made available.

Conceptualization of the VR eye-tracking issue and data collection implementation, as described in this paper, had been published as a diploma thesis (Ugwitz, 2020). This was then expanded by a classification, design, and examples/implementations of evaluative/verificative algorithms, and a discussion on future research prospects – as such, the paper was published.

3.3 Supported research

This thesis can be criticized from the angle of *the publications being primarily just a collection of algorithms or software solutions*. While such an argument has its weight, other research based on the contributions of this thesis was

produced as well, and a selection of neighboring issues the research touches on (e.g., spatial behavior in applied settings, cognitive experiments in space) were expanded there.

The knowledge of building and working with virtual 3D spaces, as described in Annex 1 and extended further on, was used to produce other virtual environments to be used in research designs. Aside the environments for Annex 3/4, environments or solutions for student diploma these research were also produced (Chocholáčková, Juřík et al., 2022; Juřík, Ugwitz, Žváčková, 2022; Bořil, 2020); conference papers were published (Ugwitz, Pospíšil, Stachoň, 2020).

User controllers for VR interfaces were created as a portion of other VR research – be it conventional designs with limited user movement per intended experimental variable operationalization (Šašinka et al., 2018) or the other end of the spectrum, including rich movement expression interpretability like motion-capture-based movement (still pending use/publication).

User movement and their interaction with the virtual environment were used in an applied research scenario (Snopková et al., 2021), as were the ways of evaluating the captured behavior. The scenario involved user evacuation from a building interior based on a series of interactive events. On their escape, the users were to rely on evacuation route signage and partial knowledge of the environment. This line of research was later used in an evacuation research project (Technology Agency of the Czech Republic, TL02000103, *Cognitive psychology and spatial syntax in virtual environments for agent models*), along with originating the environment used in Annex 4, plus a cartographic analysis of evacuation plans and signage.

Prototypes of VR eye-tracking solutions were presented at conferences (Ugwitz, 2017; Ugwitz, 2019). There is also a cross-cultural study in preparation that utilizes our eye-tracking and interactivity implementations to look out for visual cognitive preferences across different nationalities (Šašinková, Čeněk, et al., 2022).

Chapter 4

Discussion

4.1 Results

From a low-level perspective, the end results of this work are software implementations intended for capturing and interpreting varieties of user behavior in virtual environments, along with analytical tools and approaches to determine the properties of an environment in a computational manner.

From a high-level outlook, the software should have a timeless design behind it – since the technologies come and go at a rapid pace here, the solutions should be re-implementable or extendable should the need to do so arise. The proper use of the software should also be documented (as intended in the published papers). The implementations should have open interfaces in the sense that they can be used either together or independently of each other, per experimental needs.

Such targets were reached throughout this dissertation – arguably so, to the extent contemporary technologies, specialization, and energy expended by one man allows. The limitations are considered below.

Regarding Annex 1: The advancement of 3D structures and worfklows

When employing 3D graphics workflows with an intent to create a desired standard or stylization of 3D objects, compositions, and visualizations, there is a great deal of variance. When considering graphics production pipelines (per companies or standards) that encapsulate such workflows, this also varies greatly. There are multiple factors to this.

For one, many software packages are out there, specializing in specific workflows and niche functionalities – be it *hard-surface modeling; organic modeling and sculpting, 3D sculpting; retopology; generative, procedural, and semi-procedural generation; a variety of mixed-media efforts; texturing*. Furthermore, the advancements in the software and production pipelines leave much legacy in both the technology and the 3D industry vocations/specializations.

Regardless, the basic building blocks of real-time virtual 3D environments stay the same (as mentioned in chapter 2, these are still outwardly 3D mesh structures with no depiction of their insides). The implementations in this dissertation may have focused on a specific 3D engine, using a specific set of

supporting software, but the overarching ideas stay the same. A review of the software available and a description of the processes used in 3D geometry creation were provided. To know the bounds and good practices of programmability of the Unity 3D engine that can likely be generalized, to an extent – after all, the basis behind this is still linear algebra. How visual variables translate to real-time 3D geometry was considered as well.

The technical aspects of future research may involve mastering other suitable tools – e.g., *Unreal Engine with its virtualized geometry, procedural generators combined with geographic big data repositories, so as to produce convincing virtual representations of geographic spaces*. Should new ways of representing 3D mass virtually (e.g., voxel shapes including voxel physics) find wider adoption in terrain engines or 3D model authoring, these can be incorporated into the workflows and future research on virtual interaction as well – be it destructible environment or perhaps more specialized applications like geomorphologic erosion models or flood simulation (Mei, Decaudin, Hu, 2007).

Regarding Annex 2: User interfaces, behavior & efficacy expressable

The intent of the movement implementation was two-fold. First, to collect user interface data (be it key presses, continuous interface movement, etc.), see how this translates into basic forms of interaction (movement, gaze), and save said data into logger outputs; and second, to provide an outlook into how said data can be visualized and interpreted. Experiments were conducted – to let users explore virtual environments, log the data, evaluate it, and provide a methodical proof of concept.

The limitation of this paper, and perhaps a reminder of how rapidly the technology progresses, is that it is a legacy solution already – considering the fact that the virtual environment used in the experiment was built in 2016-2017, the graphics workflows and fidelity are limited (Unity 2017, no procedural generation of surrounding terrains). The same goes for user interfaces – this was a keyboard & mouse experiment of continuous movement in an outdoor mountain range; the differences in using the point-and-teleport movement solution of contemporary VR controllers are not considered there as in 2017, there were no such devices widely available.

Regardless, the inclusion of other types of physical user interfaces and the ways these translate into actual movement patterns & strategies in a virtual environment is something worth further research and applicability. As VR controllers provide a rich mix of binary/continuous input variables that can be passed along by the user in multiple axes (*two degrees of freedom (DoF) in a movement joystick/trackpad, three DoF in positional tracking while moving in physical space*), there is a potential to achieve more effective and more differentiated ways of interacting, compared to using the keyboard or such. With a physical movement that translates directly into ways of virtual interaction,

new frontiers of human-computer interaction emerge, e.g., in finding efficient ways to 3D modeling while freeing ourselves from confinement to a 2D screen and 2D controls (Lang, 2021). If the future holds mass adoption for devices currently only used as industry-grade or prototypes (e.g., motion capture, haptics), new, highly individually differentiated behavioral modalities expressed by users would point to a plethora of new user studies, findings, and solutions.

Regarding Annex 3: Visual scripting, run-time interactions

While our understanding of the movement interaction mentioned in the previous few paragraphs is *"active user, passive environment"* interaction type (refer to the typology of virtual interactions mentioned in Annex 4 for details), the ability to change the objects, object properties and scene set up in an environment unlocks plenty new ways of functioning virtually. In essence, everything possible in the physical world is to be possible here, too; this can be further accentuated by *the ability to disregard physics, gravity, scale, travel distance & speed, seeing past solid walls and structures by leaning one's head beyond their virtual position, the economic costs of using and creating surrounding spaces, architecture, and the like* – should this be desired. Such limitlessness of virtual environments is, however, a double-edged sword, as defying reality too much can cause a sense of derealization (Peckmann et al., 2022) or simulation sickness. Therefore, any study or implementation of virtual interaction should consider binding factors that ground a user in the virtual experience while maintaining immersion with respect to their physiological indicators.

That being said, the presented study is rather simplistic. It considers the ways users can interact with virtual environments (a trigger-toggle interaction structure), the implementable in-engine features that can respond back to users, and how to plan out & model interactions as process schemas. This can be used in controlled experiments where features of environments change in front of a user in a pre-defined manner or in interaction studies where the user can change the environment to their liking; the combination of both approaches is also possible.

While the trigger-toggle structure presented in the paper provides some accessibility to non-programmers (as facilitated by the Unity Inspector user interface it utilizes), this is merely one of the approaches to provide a toolkit and interface intended for setting up interaction in a 3D engine's editor. Visual scripting of interconnecting interaction nodes (Knutsen, 2021) can serve as a better, more accessible, and more universal way of implementing this – at higher implementation costs, that is. Node-based visual scripting can even be made available in runtime, with a user moving about a virtual environment and placing/interconnecting node-based structures with 3D representations

into the virtual world, proceeding to tie the interactions explicitly onto the virtual objects.

Some of such interactions (be it simple things like manipulating objects or more complex scenarios like creating node-based hierarchies and event structures among objects of a virtual environment) can be adapted to user performance testing. In this regard, intelligence testing can be considered. Over the last century, the subject has been under scientific scrutiny. The existing research has provided classifications of proposed factors of intelligence (Sternberg, 2015); the means of testing intelligence have, however, stayed within the bounds of either pen-and-paper testing or those of cumbersome specialized devices (Witkin, Asch, 1948) or 2D computer screen test platforms. As such, intelligence testing by means of virtual environment interaction (esp. when testing for spatial abilities) can provide a new approach to the old problem.

Regarding Annex 4: More in-depth VR eye-tracking algorithms

The VR eye-tracking paper is, arguably, the grandest scheme of this dissertation, as the previous body of work needed to converge well to create a solid foundation for implementing this: the ability to create and adequately segment virtual environments (Annex 1), to log and visualize user input data (Annex 2), and, possibly, to use eye-tracking inputs (e.g., dwell times) as a means of triggering interactions (Annex 3).

Workflows and algorithms for various stages of eye-tracking in VR were proposed: environment preparation, data collection, data correction, and data visualization. Two levels of data collection/interpretation were proposed: the simple object-based one and the more complex fixation-based one (Ugwitz, Juřík, Šašinka, 2017; Ugwitz, Stachoň, Šašinka, 2019). The latter needs to employ distance gaze vector projection translating into the virtual environment – this implies a novelty approach compared to the methods in traditional 2D eye-tracking and their conservative 3D/VR derivates (Pettersson et al., 2018).

Experiment scene suitability is considered, and so are data visualization challenges. Some of the algorithms presented in this paper (eight in total) are proof of concept only – they leave a lot of optimizations, user interface, or algorithm adjustment to be desired. Similarly, an overarching framework to make use of these is required. Therefore, the potential for future publications is there – this, however, implies highly specialized programming solutions.

4.2 Future work

While Annex 4 mentions proof of concept algorithms that consider spatial arrangements (e.g., the eye-tracking magnetism algorithm, where visual proximity of objects as perceived from a user camera angle is evaluated by pro-

jecting conecast into the direction a user is looking), this is only scratching the surface. Such algorithms are merely selective unidirectional projections; other algorithms are desired as well: omnidirectional projections to capture the features of virtual spaces, heuristic algorithms to estimate the visual significance of separate objects, and object compositions.

As for virtual environments intended for user movement, interaction, or experimental testing, the added value of spatial analytics lies in providing another perspective on the suitability of the virtual space – *to provide their own conclusions and to compare these against behavioral results manifested by users; to find overlaps between the two; extrapolating from this, to be able to provide conclusions regarding the virtual space based on spatial algorithms only (to be in the know without the need for organizing and evaluating yet another user study).*

Spatial algorithms executed on individual objects and on the whole

Per human visual cognition (Blazhenkova, Kozhevnikov, 2009), there are two approaches to making sense of surrounding space: *the holistic approach, where an environment is seen as a composition of positive/negative space, and the analytical approach, where properties of individual objects are considered*. This can be adapted to two distinctive approaches in computational analysis of virtual space – a working definition that can be operationalized and implemented. 3D objects can be created, separated from their surroundings, and analyzed individually; metrics of virtual space can be implemented by generating 2D or 3D grids of probe objects that do the analysis in the defined space and integrate their findings.

Various spatial/geometry/imaging algorithms, e.g., grids of raycasters, navmeshing, complexity analysis, and visual saliency, are to be considered in their utility and potential explanation value. Objects can be evaluated per their decompositions into the structures they are made of (mesh vertices, edges, faces; textures and shaders; object shapes, color grading, and the like). Spatial layouts may require extra defining per referencing or utilizing spatial cognition/navigation theories (Meilinger, 2008).

To be able to compute three-dimensional visibility, a three-dimensional matrix of omnidirectional raycaster nodes can be used across a defined virtual area[1] (Figure 4.1). This way, distances, and visible objects can be considered. Height differentiation can be observed from each node, as can be the roughness/variety of the surrounding terrain. As such, each raycaster node will

[1] A two-dimensional grid of such nodes is, in principle, already implemented and described in space syntax theory/applications as isovist (Hillier et al., 1976) – a solution that does not take into consideration vertically differentiated areas, height differences, elevatable platforms, etc. Furthermore, when implementing the raycaster nodes in 3D space, it makes no sense to use just one (vertical) lineup of nodes, as isovist would do – due to the implementation demands of a virtual 3D space, there would subsequently be no difference in implementing a 2D node matrix compared to a 3D node matrix.

carry a variety of information based on what it gathers from its surroundings. Then, optimal trajectories across an environment can be determined based on creating a pathway through the nodes, all while considering the preferences of acquired parameters that the raycaster nodes hold.

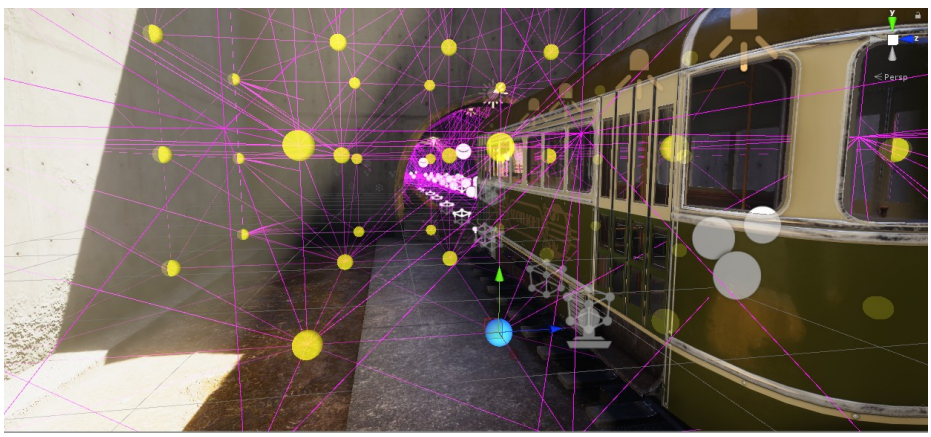


Figure 4.1: An example 3D matrix of nodes. These are Unity light probes for calculating environmental reflections (Unity Technologies, 2019), but the concept stands – 3D spatial analytics nodes would retain similar instantiation in space while running their own algorithms.

As the nodes raycast into their surroundings, they can reach out to objects in the scene visible to them to extract a variety of information included in the objects (as stored in the objects). The idea is to pre-compute the properties of said objects by an external application, followed by importing the results into the 3D engine as data containers attached to instances of objects present in a virtual scene. Some properties of 3D objects can indeed be processed using either computational algorithms or image processing – e.g., object shape convexness/concaveness (Toth, O'Rourke, Goodman, 2017), obejct stretching/borders (Figure 4.2), evenness of object topology detail, visual salience of object texture (Cong et al., 2018), object differentiation across rotations, percentage of the object visible, scene complexity (Rosenholtz et al., 2017). When the nodes present in the space get access to this data along with distance, visibility, size, and the like, they can proceed to draw higher-level conclusions. The spatial node-based algorithms are to be implemented in the Unity 3D engine; object-based analytics will likely have to be carried out using external solutions. It is crucial to review the available algorithms and their utility, as implementing new, mathematically complex solutions of our own may prove to be challenging. In order to undergo this endeavor, it may be fruitful also to consider the existing body of knowledge in the fields of computational geometry (for working with spatial structures), or architecture & psychology of space (for having sound theories to base the evaluations on).

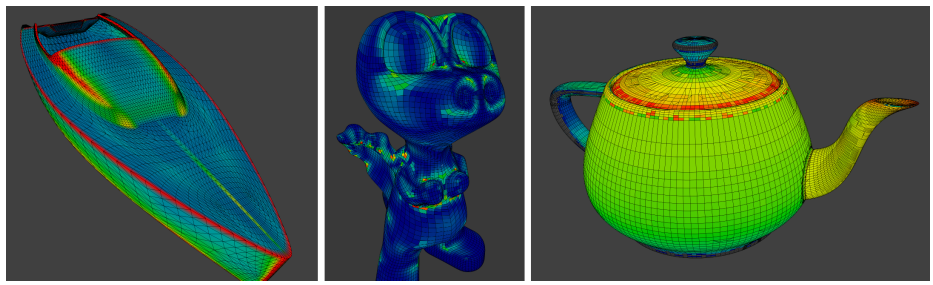


Figure 4.2: This is Blender-based mesh analytics for 3D printing (Blender Foundation, 2022). The figure depicts sharp edge detection (left), polygon skewness detection (center), and surface thickness evaluation (right). Regardless, a similar approach can be borrowed, extended or implemented for our causes.

Comparing of acquired user/algorithmic findings

Experiments in virtual environments with live users produce patterns of behavior related to their virtual surroundings[2]. Algorithmic processing of such surroundings may produce some results as well. So the integrating idea is to determine the significance of overlaps between the two methodologies used and whether interrelating conclusions can be made.

This implies there can be a continuous, iterative body of work ahead – as new ways of evaluating users or virtual environments are developed, as virtual environments are built using new structures/representations, new questions, analyses, and ways of testing will arise – *that is, unless one reaches that positivist state where everything that can be known about user behavior is known and can be predicted, with no further manual user experiment input necessary*. So any studies to be published are to be focused on finding parallels between user-based and algorithm-based spatial findings. Also, risks and limitations with algorithmic assessment need to be considered – e.g., when these spatial algorithms are left to their own devices so as to produce large quantities of data without the researcher's ability to find the underlying factors or groupings that show conclusions of significance, the researcher would be left with nondescript piles of big data missing a classification or consideration on their importance.

Regardless, the experiment included in Annex 2 collected user data. The environment used for eye-tracking per Annex 4 also has user movement data available, and so have other related experiments[3]. Thus, the environments can be reused for algorithmic analysis, and the results of the two methodics can then be compared to provide a discourse worth considering.

[2]The extent and significance of other variables affecting the users is a matter of experiment design, operationalization, researcher demeanor, and controlling for intervening variables.

[3]If experimental data sharing in virtual environment research becomes a trend, virtual environments and data originated by other research teams can also be used.

Literature

Abdirad, H., Dossick, C. S. (2016). BIM curriculum design in architecture, engineering, and construction education: a systematic review. *Journal of Information Technology in Construction, 21*(17), pp. 250-271.

Adobe (n.d.) Substance 3D Documentation. Retreived on June 1, 2022, from: https://substance3d.adobe.com/documentation/home

Akenine-Moller, T., Haines, E., Hoffman, N. (2018). *Real-Time Rendering*, 4th ed. CRC Press.

Alexiou, E., Xu, P., Ebrahimi, T. (2019, September). Towards modelling of visual saliency in point clouds for immersive applications. In *2019 IEEE International Conference on Image Processing (ICIP)*, pp.4325-4329. https://doi.org/10.1109/ICIP.2019.8803479

Bamodu, O., Ye, X. M. (2013). Virtual reality and virtual reality system components. *Advanced materials research, 765*, pp. 1169-1172. Trans Tech Publications Ltd. https://doi.org/10.4028/www.scientific.net/AMR.765-767.1169

Baron, D. (2019). *Hands-on game development patterns with Unity 2019: create engaging games by using industry-standard design patterns with C#*. Birmingham, Packt Publishing.

Bergen, S., Ross, B. J. (2013). Aesthetic 3D model evolution. *Genetic Programming and Evolvable Machines, 14* (3), pp. 339-367. https://doi.org/10.1007/s10710-013-9187-8

Bertin, J. (1967). *Sémiologie graphique. Les diagrammes, les réseaux, les cartes*. Paris: Gauthier-Villars.

Blender Foundation (2022, May). Mesh Analysis. Blender Manual. Retrieved on June 1, 2022, from: https://docs.blender.org/manual/en/latest/modeling/meshes/mesh_analysis.html

Boreskov, A., Shikin, E. (2014). *Computer graphics: from pixels to programmable graphics hardware*. Florida: Crc Press.

Bořil, J. (2020). Kartografická vizualizace pro účely evakuace [Cartographic support of emergency evacuation]. *Bachelor's thesis. Masaryk University, Faculty of Science.*

Bowman, D.A., Kruijff, E., LaViola, J.J., Poupyrev, I. (2005). *3D User Interfaces: Theory and Practice*. Addison Wesley, Longman Publishing, p. 512.

Birn, J. (2013). *Digital Lighting and Rendering*, 3rd ed. New Riders.

Blazhenkova, O., Kozhevnikov, M. (2009). The new object-spatial-verbal cognitive style model: Theory and measurement. *Applied Cognitive Psychology, 23* (5), pp. 638–663. https://doi.org/10.1002/acp.1473

Brookes, J., Warburton, M., Alghadier, M., Mon-Williams, M., Mushtaq, F. (2019). Studying human behavior with virtual reality: The Unity Experiment Framework. *Behavior Research Methods*. https://doi.org/10.3758/s13428-019-01242-0

CANAVERAL Project Report (2021). Department of Geography, Faculty of Science, Masaryk University. *Internal report for the CANAVERAL project*.

Caserman, P., Garcia-Agundez, A., Konrad, R., Göbel, S., Steinmetz, R. (2018). Real-time body tracking in virtual reality using a Vive tracker. *Virtual Reality, 23*(2), pp. 155–168. https://doi.org/10.1007/s10055-018-0374-z

Chalmers, E. L. (1952). Monocular and Binocular Cues in the Perception of Size and Distance. *The American Journal of Psychology, 65*(3). https://doi.org/10.2307/1418762

Chittaro, L., Ieronutti, L. (2004) A Visual Tool for Tracing Users' Behavior in Virtual Environments. *Proceedings of the Working Conference on Advanced Visual Interfaces*, Gallipoli, Italy, pp. 41–47. https://doi.org/10.1145/989863.989868

Clay, V., König, P., König, S. U. (2019). Eye tracking in virtual reality. *Jornal of Eye Movement Research*. https://doi.org/10.16910/jemr.12.1.3

Cong, R., Lei, J., Fu, H., Cheng, M. M., Lin, W., Huang, Q. (2018). Review of visual saliency detection with comprehensive information. *IEEE Transactions on circuits and Systems for Video Technology, 29*(10), pp. 2941-2959. httsp://doi.org/10.1109/TCSVT.2018.2870832

Cooper, T. (2018). Introduction to Shader Graph: Build your shaders with a visual editor. Unity Blog. Retrieved on June 1, 2022, from: https://blog.unity.com/technology/introduction-to-shader-graph-build-your-shaders-with-a-visual-editor

Corsini, M., Larabi, M. C., Lavoué, G., Petřík, O., Váša, L., Wang, K. (2013, February). Perceptual metrics for static and dynamic triangle meshes. In *Computer graphics forum, 32* (1), pp. 101-125. Blackwell Publishing Ltd., Oxford, UK. https://doi.org/10.1111/cgf.12001

Creem-Regehr, S. H., Stefanucci, J. K., Thompson, W. B., Nash, N., Mccardell, M. (2015). Egocentric distance perception in the Oculus Rift (DK2). In *Proceedings of the ACM SIGGRAPH Symposium on Applied Perception*. https://doi.org/10.1145/2804408.2804422

Cuervo, E., Chintalapudi, K., Kotaru, M. (2018). Creating the Perfect Illusion. In *Proceedings of the 19th International Workshop on Mobile Computing Systems & Applications*. https://doi.org/10.1145/3177102.3177115

Daria, B., Martina, C., Alessandro, P., Fabio, S., Valentina, V., Zennaro, I. (2018). Integrating mocap system and immersive reality for efficient human-centred workstation design. *IFAC-PapersOnLine, 51*(11), pp. 188–193. https://doi.org/10.1016/j.ifacol.2018.08.256

Dedík, L., Minaroviech, J. (2017). Project of the Digital Monuments Funds: Digitization of the Cultural Heritage of Slovakia. *Studies in Digital Heritage, 1*(2), pp. 590–606. https://doi.org/10.14434/sdh.v1i2.23286

Denis, M. (2017). *Space and Spatial Cognition*. Routledge, London. https://doi.org/10.4324/9781315103808

Doppioslash, C. (2018). *Physically Based Shader Development for Unity 2017*. Apress, Berkeley

Downs, R. M., Stea, D. (1973). *Image and environment: Cognitive mapping and spatial behavior*. Aldine, Chicago.

Faure, J., Prochazka, D., Menard, D. (2019, November 19). Unity Reflect is here. Unity Blog. Retrieved from https://blog.unity.com/aec/unity-reflect-arrives-in-december

Foley, D. J., van Dam, A., Feiner, S. K., Hughes, J. F. (1996). *Computer Graphics. Principles and practice.* 2nd ed. Addison Wesley, New York.

Foxglove, C. (2019, May 28). Speed up your work with the new Terrain Tools Package. Unity Blog. Retrieved from https://blog.unity.com/technology/speed-up-your-work-with-the-new-terrain-tools-package

Fuad, M. (2019, December 17). XR Interaction Toolkit Preview Package is here. Unity Technologies Blog. Retrieved from https://blogs.unity3d.com/2019/12/17/xr-interaction-toolkit-preview-package-is-here/

Gackenbach, J., Bown, J. (2017). *Boundaries of self and reality online : implications of digitally constructed realities.* Elsevier, London.

Gladilin, V. A., Mandritsa, O., Chalova, A. A., Kazakova, I. N., Zolotukhina, I. V. (2020). Digital Technology in the Design of Nonlinear (Parametric) Objects Environmental Space. *International Journal of Advanced Trends in Computer Science and Engineering*, 9(3). https://doi.org/10.30534/ijatcse/2020/105932020

Goldstein, E. B. (2013). *Sensation and perception*, 9th ed. Wadsworth Cengage Learning, Belmont.

Gomaa, H. (2011). *Software modeling and design: UML, use cases, patterns, and software architectures*. Cambridge University Press, Cambridge, UK.

Gorini, A., Capideville, C. S., Leo, G. D., Mantovani, F., Riva, G. (2011). The Role of Immersion and Narrative in Mediated Presence: The Virtual Hospital Experience. *Cyberpsychology, Behavior, and Social Networking, 14*(3), pp. 99–105. https://doi.org/10.1089/cyber.2010.0100

Gregory, J. (2018) *Game Engine Architecture*, 3rd ed. CRC Press.

Grübel, J., Thrash, T., Hölscher, C., Schinazi, V. R. (2017). Evaluation of a conceptual framework for predicting navigation performance in virtual reality. *PloS one, 12*(9). https://doi.org/10.1371%2Fjournal.pone.0184682

Hadjiphilippou, P. (2013). *The Contribution of the five human senses towards the perception of space*. Department of Architecture, University of Nicosia.

Häberling, C., Bär, H., Hurni, L. (2008). Proposed Cartographic Design Principles for 3D Maps: A Contribution to an Extended Cartographic Theory. *Cartographica. The International Journal for Geographic Information and Geovisualization*, 43(3), pp. 175-188. https://doi.org/10.3138/carto.43.3.175

Herman, L., Řezník, T., Stachoň, Z., Russnák, J. (2018). The Design and Testing of 3DmoveR: An Experimental Tool for Usability Studies of Interactive 3D Maps. *Cartographic Perspectives 9*, pp. 31–63. http://dx.doi.org/10.14714/CP90.1411

Hillier, B., Leaman, A., Stansall, P., Bedford, M. (1976). Space syntax. *Environment and Planning B: Planning and design, 3* (2), pp. 147-185. https://doi.org/10.1068%2Fb030147

Hrimech, H., Alem, L., Merienne, F. (2011). How 3D Interaction Metaphors Affect User Experience in Collaborative Virtual Environment. *Advances in Human-Computer Interaction, 2011*, 1–11. https://doi.org/10.1155/2011/172318

jBPM (2020). Retrieved on June 1, 2022, from: https://www.jbpm.org/

Johansson, M., Roupé, M., Bosch-Sijtsema, P. (2015). Real-time visualization of building information models (BIM). *Automation in Construction, 54*, pp. 69–82. https://doi.org/10.1016/j.autcon.2015.03.018

Jones, C.A., Church, E, (2020). Photogrammetry is for everyone: Structure-from-motion software user experiences in archaeology. *Journal of Archaeological Science: Reports, 30*. https://doi.org/10.1016/j.jasrep.2020.102261

Kivunja, C. (2015). Creative engagement of digital learners with gardner's bodily-kinesthetic intelligence to enhance their critical thinking. *Creative Education, 6*(6). https://doi.org/10.4236/ce.2015.66060

Knutsen, K. Í. (2021). Visual Scripting in Game Development. *Bachelor's thesis. Metropolia University of Applied Sciences.*

Kothgassner, O. D., Felnhofer, A. (2020). Does virtual reality help to cut the Gordian knot between ecological validity and experimental control?. *Annals of the International Communication Association, 44* (3), pp. 210-218. https://doi.org/10.1080/23808985.2020.1792790

Krampen, M. (2011). *Meaning in the urban environment*. Routledge.

Kvarda, O. (2020). Virtuální realita jako prostředek kartografické komunikace [Virtual reality as a means of cartographic communication]. *Master's thesis. Masaryk University, Faculty of Science.*

Lang, B. (2021). Built on "Medium", Adobe's New Substance 3D Modeler Aims to Bring VR Modeling to Pro Workflows. Roat to VR. Retrieved on June 1, 2022, from: https://www.roadtovr.com/adobe-substance-3d-modeler-medium-vr-modeling-pro-workflows/

Lappi, O. (2015). Eye tracking in the wild: the good, the bad and the ugly. *Jornal of Eye Movement Research*. https://doi.org/10.16910/jemr.8.5.1

Lavoué, G., Larabi, M. C., Váša, L. (2015). On the efficiency of image metrics for evaluating the visual quality of 3D models. *IEEE transactions on visualization and computer graphics, 22* (8), pp. 1987-1999. https://doi.org/10.1109/TVCG.2015.2480079

Lazar, J., Feng, J. H., Hochheiser, H. (2017). *Research Methods in Human-Computer Interaction*. Elsevier.

Lengyel, E. (2020). *Foundations of Game Engine Development, Volume 2: Rendering*. Terathon Software LLC.

Liu, X., Shi, S., Campbell, B. (2022, February). Experience the new Unity Terrain Demo scenes for HDRP and URP. Unity Blog. Retrieved on June 1, 2022, from: https://blog.unity.com/games/experience-the-new-unity-terrain-demo-scenes-for-hdrp-and-urp

Malone, T. W. (1982). Heuristics for designing enjoyable user interfaces. *Proceedings of the 1982 Conference on Human Factors in Computing Systems*. https://doi.org/10.1145/800049.801756

Mat, R. C., Shariff, A. R. M., Zulkifli, A. N., Rahim, M. S. M., Mahayudin, M. H. (2014). Using game engine for 3D terrain visualisation of GIS data: A review. *IOP Conference Series: Earth and Environmental Science*, 20. https://doi.org/10.1088/1755-1315/20/1/012037

McAuley, S., Hill, S., Martinez, A., Villemin, R., Pettineo, M., Lazarov, D., Andersson, H. (2013). Physically based shading in theory and practice. In *ACM SIGGRAPH 2013 Courses* (pp. 1-8). http://dx.doi.org/10.1145/2614028.2615431

Mei, X., Decaudin, P., Hu, B. G. (2007, November). Fast hydraulic erosion simulation and visualization on GPU. In *15th Pacific Conference on Computer Graphics and Applications*, pp. 47-56. IEEE. https://doi.org/10.1109/PG.2007.15

Mezrag, Y., Boton, C. (2021). Integrating 4D simulations and virtual reality environments: An innovative prototype. In *International Conference on Cooperative Design, Visualization and Engineering*, pp. 103-114. Springer, Cham.

Meilinger, T. (2008). *Strategies of Orientation in Environmental Spaces*. Logos Verlag, Berlin.

Montello, D., Sas, C. (2006). Human Factors of Wayfinding in Navigation. In Informa Healthcare, Karwowski, W. (Eds.), *International Encyclopedia of Ergonomics and Human Factors* (2nd ed.). CRC Press. http://doi.org/10.1201/9780849375477

Murphy, M., McGovern, E., Pavia, S. (2009). Historic building information modelling (HBIM). *Structural Survey 27*(4), pp. 311-327. https://doi.org/10.1108/02630800910985108

Natephra, W., Motamedi, A., Fukuda, T., Yabuki, N. (2016). Integrating building information modeling and game engine for indoor lighting visualization. In *Proceedings of the 16th International Conference on Construction Applications of Virtual Reality, 11*, pp. 605-617. https://doi.org/10.1186/s40327-017-0058-x

Nelson, F. (2015, March). AMD's LiquidVR Puts Processing Muscle Behind Virtual Reality. Tom's Hardware. Retrieved June 1, 2022, from: https://www.tomshardware.com/news/amd-liquidvr-virtual-reality,28682.html

Paljic, A. (2017). Ecological validity of virtual reality: three use cases. In *International Conference on Image Analysis and Processing*, pp. 301-310. Springer, Cham. http://dx.doi.org/10.1007/978-3-319-70742-6_28

Peckmann, C., Kannen, K., Pensel, M. C., Lux, S., Philipsen, A., Braun, N. (2022). Virtual reality induces symptoms of depersonalization and derealization: A longitudinal randomised control trial. *Computers in Human Behavior, 131*. https://doi.org/10.1016/j.chb.2022.107233

Penn, A. (2003). Space syntax and spatial cognition: Or why the axial line? *Environment and behavior, 35* (1), pp. 30-65. https://doi.org/10.1177%2F0013916502238864

Pettersson, J., Albo, A., Eriksson, J., Larsson, P., Falkman, K. W., Falkman, P. (2018). Cognitive Ability Evaluation using Virtual Reality and Eye Track-

ing. In *International Conference on Computational Intelligence and Virtual Environments for Measurement Systems and Applications (CIVEMSA)*. IEEE. https://doi.org/10.1109/CIVEMSA.2018.8439999

Pharr, M., Jakob, W., Humphreys, G. (2017). *Physically based rendering: from theory to implementation*. Elsevier.

Poeschl, S., Wall, K., Doering, N. (2013). Integration of spatial sound in immersive virtual environments an experimental study on effects of spatial sound on presence. In *2013 IEEE Virtual Reality (VR)*, pp. 129-130. IEEE. https://doi.org/10.1109/VR.2013.6549396

Popelka, S. (2018). *Eye-tracking (nejen) v kognitivní kartografii: praktický průvodce tvorbou a vyhodnocením experimentu.* Univerzita Palackého v Olomouci pro katedru geoinformatiky.

Quixel Blog (2021, February). Medieval Game Environment: Creating Atmospheric Terror in Realtime. Retreived on June 1, 2022, from: https://blog.megascans.se/2021/02/11/medieval-game-environment-creating-atmospheric-terror-in-realtime/

Rietzler, M., Geiselhart, F., Frommel, J., Rukzio, E. (2018). Conveying the Perception of Kinesthetic Feedback in Virtual Reality using State-of-the-Art Hardware. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. https://doi.org/10.1145/3173574.3174034

Romero, M., Sewell, B., Cataldi, L. (2022). *Blueprints Visual Scripting for Unreal Engine 5: Unleash the true power of Blueprints to create impressive games and applications in UE5* (3rd ed.) Packt Publishing.

Rosenholtz, R., Li, Y., Nakano, L. (2007). Measuring visual clutter. *Journal of Vision, 7*(2), 17. https://doi.org/10.1167/7.2.17

Roth, R. E. (2012). Cartographic Interaction Primitives: Framework and Synthesis. *Cartographic Journal, 49*, pp. 376–395. http://dx.doi.org/10.1179/1743277412Y.0000000019

Roth, R. E. (2013). Interactive maps: What we know and what we need to know. *Journal of Spatial Information Science*, (6), pp. 59-115. http://dx.doi.org/10.5311/JOSIS.2013.6.105

Schmitt, S. (2022). World Machine Features for Terrain Generation. World Machine Software LLC. Retrieved on June 1, 2022, from: https://www.world-machine.com/features.php

Schneider, P. (2014). *NURB Curves: A Guide for the Uninitiated*. MACTECH.

Siegel, A. W., White, S. H. (1975). The development of spatial representations of large-scale environments. *Advances in child development and behavior, 10*, pp. 9-55. https://doi.org/10.1016/S0065-2407(08)60007-5

Sharetextures. *Mug Texture*. Retrieved June 1, 2022, from: https://www.sharetextures.com/textures/ground/mug-texture/

Smith, S. P., Trenholme, D. (2009). Rapid prototyping a virtual fire drill environment using computer game technology. *Fire Safety Journal, 44* (4), pp. 559–569. https://doi.org/10.1016/j.firesaf.2008.11.004

Sternberg, R. J. (2015). Multiple intelligences in the new age of thinking. In *Handbook of intelligence*, pp. 229-241. Springer, New York, NY.

Šašinka, Č., Stachoň, Z., Sedlák, M., Chmelík, J., Herman, L., Kubíček, P., Šašinková, A., Doležal, M., Tejkl, H., Urbánek, T., Svatoňová, H., Ugwitz, P., Juřík, V. (2018). Collaborative immersive virtual environments for education in geography. *ISPRS International Journal of Geo-Information, 8*(1). https://doi.org/10.3390/ijgi8010003

Tan, D. S., Nijholt, A. (2010). *Brain-computer interfaces: applying our minds to human-computer interaction*. Springer.

Tavakoli, M. (2019). *Performance Evaluation of Competing Data Structures in Pathfinding*. Doctoral dissertation, University of Windsor, Canada.

Tinwell, A. (2014). *The uncanny valley in games and animation*. CRC Press.

Toth, C. D., O'Rourke, J., Goodman, J. E. (Eds.). (2017). *Handbook of discrete and computational geometry*. CRC press.

Unity Manual (2022, March). XR Interaction Toolkit. Retrieved on June 1, 2022, from: https://docs.unity3d.com/Packages/com.unity.xr.interaction.toolkit@2.1/manual/index.html

Unity Scripting reference (2021). Retrieved one June 1, 2022, from https://docs.unity3d.com/ScriptReference/

Unity Technologies (2017). 2D Game Kit Reference Guide. Unity Learn. Retrieved from https://learn.unity.com/tutorial/2d-game-kit-reference-guide

Unity Technologies (2019). Configuring Light Probes 2019.1. Unity Learn. Retrieved on June 1, 2022, from: https://learn.unity.com/tutorial/configuring-light-probes-2019-1

Vasser, M., Kängsepp, M., Magomedkerimov, M., Kilvits, K., Stafinjak, V., Kivisik, T., Vicente, R., Aru, J. (2017). VREX: an open-source toolbox for creating 3D

virtual reality experiments. *BMC Psychology*. https://doi.org/10.1186/s40359-017-0173-4

Vanclooster, A., Van de Weghe, N., De Maeyer, P. (2016). Integrating indoor and outdoor spaces for pedestrian navigation guidance: A review. *Transactions in GIS, 20* (4), pp. 491-525. https://doi.org/10.1111/tgis.12178

Vaughan, W. (2011). *Digital Modeling*. New Riders.

Volkoff, O., Strong, D. M. (2017). Affordance theory and how to use it in IS research. *The routledge companion to management information systems*, pp. 232-245. http://dx.doi.org/10.4324/9781315619361

Waltemate, T., Gall, D., Roth, D., Botsch, M., Latoschik, M. E. (2018). The Impact of Avatar Personalization and Immersion on Virtual Body Ownership, Presence, and Emotional Response. *IEEE Transactions on Visualization and Computer Graphics, 24*(4), pp. 1643–1652. https://doi.org/10.1109/tvcg.2018.2794629

Wikimedia Foundation (n.d.). FBX. Retrieved March 28, 2022, from: https://en.wikipedia.org/wiki/FBX

Witkin, H. A., Asch, S. E. (1948). Studies in space orientation: IV. Further experiments on perception of the upright with displaced visual fields. *Journal of Experimental Psychology 38*, pp. 762–82. https://psycnet.apa.org/doi/10.1037/h0053671

Zanbaka, C.A., Lok, B.C., Babu, S.V., Ulinsky, A.C., Hodges, L.F. (2005). Comparison of Path Visualizations and Cognitive Measures Relative to Travel Technique in a Virtual Environment. *IEEE Trans. Vis. Comput. Graph. 11*, pp. 694–705. https://doi.org/10.1109/tvcg.2005.92

Zhai, Y., Shah, M. (2006, October). Visual attention detection in video sequences using spatiotemporal cues. In *Proceedings of the 14th ACM international conference on Multimedia*, pp. 815-824. https://doi.org/10.1145/1180639.1180824

Zhang, F., Hu, M., Che, W., Lin, H., Fang, C. (2018). Framework for Virtual Cognitive Experiment in Virtual Geographic Environments. *International Journal of Geo-Information 7*(2). https://doi.org/10.3390/ijgi7010036

Zhou, Y., Pang, Y., Chen, F., Zhang, Y. (2020). Three-Dimensional Indoor Fire Evacuation Routing. *ISPRS International Journal of Geo-Information, 9* (10), 558. https://doi.org/10.3390/ijgi9100558

Part II

Appendices

List of appendices

Annex 1: Paper I 45

Annex 2: Paper II 66

Annex 3: Paper III 89

Annex 4: Paper IV 101

Annex 5: Related publications and sources 132

Annex 6: Curriculum Vitae 134

Annex 1: Paper I

Ugwitz, P., Herman, L., Stachoň, Z. (2021). 3D visualization of historical buildings: Methods, their utility in the tourism industry and beyond. *Regionální rozvoj mezi teorií a praxí*, 2021(1)

3D VISUALIZATION OF HISTORICAL BUILDINGS: METHODS, THEIR UTILITY IN THE TOURISM INDUSTRY AND BEYOND

Pavel Ugwitz, Lukáš Herman, Zdeněk Stachoň

Masaryk University, Faculty of Science, Department of Geography
Kotlářská 267/2, 611 37 Brno
E-.mail: ugwitz.pavel@mail.muni.cz, herman.lu@mail.muni.cz, 14463@mail.muni.cz

Keywords:
3D reconstruction, 3D visualization, historical buildings, virtual tourism

Abstract:
What are the reasons and motivation for virtual reconstruction of buildings? What are the possibilities of current-generation 3D visualization in the context of (re)constructing buildings, especially historical monuments, and how can it be utilized in regional development? Such questions are presented and discussed in the article. The following text narrows down the issues from a general/technological overview to focus on the regional development, the tourism industry, and reconstruction of old buildings. Examples of visualization methods are presented and demonstrated with the provided examples. The discussion follows by highlighting the need for collaboration across multiple specializations, considering the untapped potential for utilizing 3D interactive models in the tourism industry, and the ability to observe and collect user behavior in 3D visualization to understand their behavior/interactions better, and to (re)create the 3D user experience to the optimum of their needs and expectations.

Introduction

Significant monuments of human settlements of the past are considered historically valuable or interesting components of the landscape (e.g., for practical utilization in the context of regional history & tourism industry) if they fulfill one or more criteria: the building in question is a remarkable representation of the architectonic style of an era (Davidson, 2017), it is situated in a strategic location with a (once) significant administrative, martial or cultural function (Ching, Jarzombek, Prakash, 2006). To consider such, e.g., with castles, the architectural styles such monuments are (re)built in change through the centuries, and this leads to the castle changing its shape/function through time, or to ruin. With material destruction, however, the historical value of the building is (somewhat) inadvertently lost. Therefore, the question is whether it is possible and/or desirable to create a virtual reconstruction of historical monuments - be it either to capture their current, preserved form, or to at least estimate the shape and form of a building that is no longer there. 3D visualization and documentation of such historical monuments help to keep the history. Moreover, it supports economic growth by stimulating cultural tourism (Koeva, 2016, Scerri et al., 2016).

Through scientific discoveries in the modern era, the scientific fields of archaeology, architecture, and history of material culture have already proposed ways to achieve historical reconstructions. Since the original appearance of a historical monument is not always preserved well, or nearly at all, or is just (wrongly) assumed, some methods of the aforementioned scientific fields do adjust for this: from aerial archaeology (Raczkowski, 2017), negative remains analysis (Angelo, 2017), experimental archaeology (Ascher, 1961), to visual heuristics

based on knowing building construction practices of an era, the missing pieces can, often, be well estimated to provide a probable resemblance of a monument in a given time.

While the aforementioned sciences and methods allow for the extraction of reconstruction data from historical sites even with little or no remains, the standard mediums of reconstruction methods do have limitations - in visuals or adjustability. E.g., physical models are limited by their scale and transportability; if they are to be present in multiple places, multiple copies of the same have to be created. Depictions and imagery are constrained by angle and scale, i.e., the static depiction of a static scale with details or the big picture often being lost. Animations, while digital, are still a set of prerendered frames that offer little/no user interactivity. Enter interactive 3D visualizations, and consider their advantages:

- they can be outlined at first and then worked out in great geometric detail
- 3D structures are transferable across 3D applications and can be worked on using a broad set of production techniques regarding their 3D geometry
- they are (re)editable and customizable (lighting, texture, post-processing)
- they can be made interactive and served through a variety of media, e.g., PCs, smartphones, web interfaces, augmented/mixed/virtual reality

Furthermore, 3D visualization can serve as a basis for producing and distributing the aforementioned static media visualizations (screenshots, fly-through animations, 3D prints), all from a single, easily managed source.

Photorealistic real-time 3D visualizations are now possible due to past technological progress in computer graphics. The technologies have reached a high standard in resolution, 3D model, detail, texturing, and lighting simulation - e.g., through normal mapping, tessellation, physically-based rendering, or ray-tracing (Singer, 2021). Not only this, but the applications purposed for the creation of 3D assets/compositions have evolved their tools to reflect such advancement: while there is more to be done to produce a higher level of fidelity in 3D graphics, 3D tools have streamlined this process, often automating, randomly generating, or visually aiding (Bertolini, 2018) in what would have required manual input in the past. Furthermore, the openness of current-generation 3D real-time graphics engines to scriptability (often using premade classes of C++ or C# languages) allows for custom, tailored interfaces and user experiences.

There are many workflows to achieve a high level of visual fidelity in 3D. For contemporary buildings, BIM (building information modeling (Murphy, McGovern, Pavia, 2009)) is available. BIM - a 3D method intended to replace traditional 2D construction drawings - is five-dimensional: it contains spatial data (3D), with the fourth and fifth dimensions being construction material properties and building modifications through time. Some 3D software, e.g., Tridify BIM tool for Unity, is already streamlined enough to take construction-purposed BIM data and to transform it into a visualization of a building that is often yet to be constructed. For historical buildings, however, we have no such data available; regardless, BIM methodology as such can serve as an inspiration, as similar results are desired to be implementable for historical sites as well - albeit the methods to obtain three-dimensional data of historical sites are different.

Photogrammetry allows for capturing high-detailed 3D meshes of existing (historical) structures. Imaginary, assumed, or otherwise destroyed shapes and structures can then be recreated using either organic modeling (sculpting) or hard surface modeling (Simonds, 2013) using industry-standard 3D software like Blender, Maya, or ZBrush. Generic 3D models can also be reused from an existing 3D model library, e.g., Quixel Megascans, or 3D engine asset

stores (e.g., Unity Store, Unreal Marketplace). The strong suit of a 3D pipeline is that there is no single correct way to (re)create a virtual environment - that is because the 3D techniques and software packages can be used in conjunction with each other. Furthermore, 3D workflows nicely complement and enrich the aforementioned traditional scientific methods. For example, photogrammetry was used in archaeological sites (Jones, Church, 2020), or modeling assumed structures based on known architecture, construction practices, and other (visual) historical sources of an era.

Given the contemporary context of 3D visualization technology and historical visualizations, one has to ask about the utility of such visualizations for the tourism industry. First and foremost, website browsing, mobile browsing/apps, augmented/mixed/virtual reality headsets are all relatively new forms of media that are both becoming increasingly affordable and advanced year by year (Pulli et al., 2007; Dailey-Hebert, Simmons Estes, Choi, 2021). Such mediums can present the user with either the current or the past appearance of a historical site. User viewing can commence either as a supplement to the real visitation of the site or as a substitute/precondition to physically visiting the site (Deng et al., 2020). Provided the user experience and user interface of the virtual application is a good fit for the medium (Weiss et al., 2018) and the targeted types of users (Tomlin, 2018), this can serve to sway the potential visitor to visit the actual site, and/or to enrich the sightseeing experience while physically on site. Therefore, the profitability of such a site can be increased - either indirectly, by converting the potential visitor into an actual future visitor, or directly, by implementing purchases or donations through the virtual application, etc. Disabled or distant potential visitors who would have been otherwise unable to visit a site can also use a virtual visit as a viable alternative - i.e., virtual visitation can represent an improvement in ecology and accessibility.

While the process of creating a virtual representation of a historical site is somewhat universal in terms of 3D graphics (software, supporting reconstruction methods and sources, 3D scene creation pipeline, 3D workflows per specific 3D software), this paper will strive to present such in consideration to the tourism industry, where relevant (user experience, virtual sightseeing commentary).

1. Method

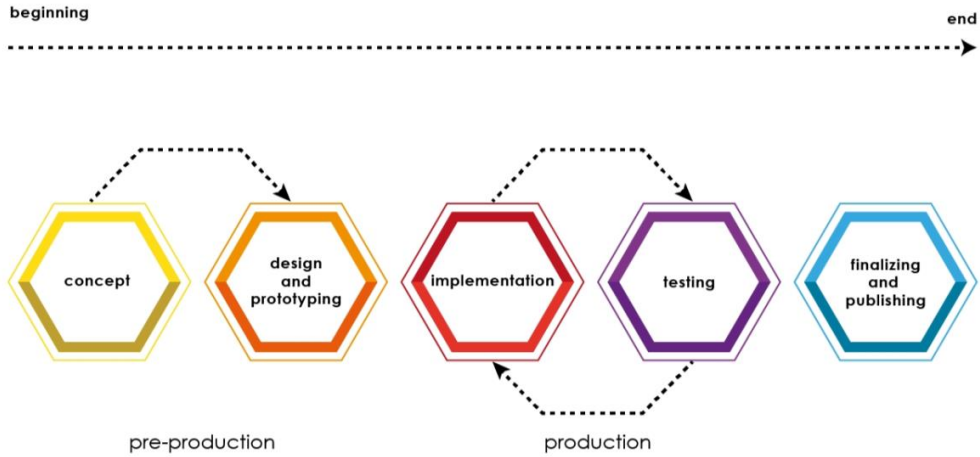
The following process of recreating 3D visualizations of historical buildings, as is to be presented, occurred in the semester of summer 2020 at the Masaryk University, through renovating a master's level college course "3D modelování a vizualizace" (Masaryk University, 2020). The students enrolled in the class were assumed not to have had previous 3D modeling experience; therefore, the semester-long course was intended to cover the fundamentals of 3D modeling. I.e., the students were introduced to what a generic 3D modeling pipeline would look like. To explicate, they were asked to model a castle. Likewise will follow through this article - describing the methods that were taught through the course while merely briefly mentioning the alternatives and advanced approaches.

1.1 3D modeling pipeline as a software development pattern

In the simplest of terms, the process of creating compelling 3D environments can be viewed as following the principles of software development (see Fig. 1). I.e., there is a concept phase that summarizes the ideas that are to be reached. A design/prototyping phase follows to make sure the concepts can be implemented. The implementation phase then carries out the main body of

work - and as it progresses, it can alter back and forth with the testing phase, to ensure the implementation is done to the desired standard. Once such a standard is reached, post-processing and finalizing work is carried out to ensure the results reach a consistent standard. As such, the software is released.

Figure 1: *generic software development pipeline. It can be applied to 3D modeling as well.*



Source: Ferro, Sapio, 2018

In terms of software development models, this can resemble the classic iterative waterfall model, where the whole process goes through from the beginning to the end (McConnell, 1996). More realistically, although the 3D modeling pipeline and its workflows do incorporate some development iterations, e.g., as in an incremental development model, e.g., as in test-based development or SCRUM (Larman, Basili, 2003), since both the software in use and the development/testing of features and visuals can cycle back and forth - especially in case the 3D visualization is a complex one.

1.2 Simplified 3D pipeline in use

To outline the structure of the 3D visualization pipeline, we chose to break it down into four segments, all with verifiable and measurable milestones of their own. The students in our course followed such a structure, too:

1. **Conceptual phase**. This phase was meant for the creator to choose a historical castle to model, and to gather as many relevant visualizations and other relevant historical sources as possible, to have a good estimate of what is there to model. If multiple visualizations depicting the same castle were present (be it either through depicting different periods or multiple interpretations of a single period), this did not impose an issue, as there was an inherent explorative component to this - and only by the time of finalizing the concept phase, our students were asked to choose their direction/vision (be it a single existing visualization or a hybrid of a few sources).
2. **Functional prototype**. The purpose here was to create a simplified, gray-boxed, detail-less 3D representation of the visualization to be. That is, only the structures of the terrain and some very primitive man-made structures were extruded, along with placing a first-person movement controller into such a scene. While there were no details, no textures, and very primitive lighting, the purpose of this segment was to establish a 1:1 scale of

the visualization to be, and to be able to walk around the place to ensure all intended areas are connected, accessible, and correctly scaled.

3. **3D modeling**. In this stage, the simplified models need detailing, texturing, and adding other environmental details. This, along with terrain tweaks and adding lighting to the scene, does resemble the majority of the actual work done on a 3D project.
4. **Finalizing and bug fixing**. The 3D scenes created in the previous step were then evaluated by a third party (in this case, student works were evaluated by a teacher; a similar dynamic would ensure similarly with a graphics designer/director, in a professional environment). The creators of the scene were notified of the main issues found in their visualization (if any) and asked to fix these. Graphical postprocessing was introduced into the 3D scenes, and so were performance-minded optimizations.

This pipeline is presented on a conceptual level. Based on the software used (in our case, just ArcGIS, Unity, and Blender), there can be another operational level pipeline. Regardless, that one is hardly specific, as the use of different software combinations can produce vastly different workflows and interdependencies. Traditional hard-surface 3D software (e.g., Maya, 3DsMax, Blender) has its way of handling things, and so do 3D packages based on sculpting or generative paradigms (e.g. Zbrush, Houdini, WorldMachine). The details production pipeline (e.g., Substance Painter/Designer) introduces its quirks, and so do the 3D engines (Unity, Unreal, CryEngine) with their plugins and affiliates (e.g., MegaScans). Coupled with GIS software, land surveying and photogrammetry, the actual 3D development pipeline one ends up with can be a rather complex one.

In either case, adhering at least to the conceptual structure above can prevent issues with 3D modeling of complex structures/composition later on. E.g., as long as a correct building/space scale and user controller interface are established throughout the functional prototyping phase, the other visual/functional issues one can encounter later on will only be minor ones. Similarly, adding lighting and postprocessing details into a scene only makes sense once after the majority of the scene's geometry has been put into place. In contrast to this, ignoring such 3D scene production principles can lead to wasted efforts, where emerging issues may require redoing a significant portion of the previous body of work.

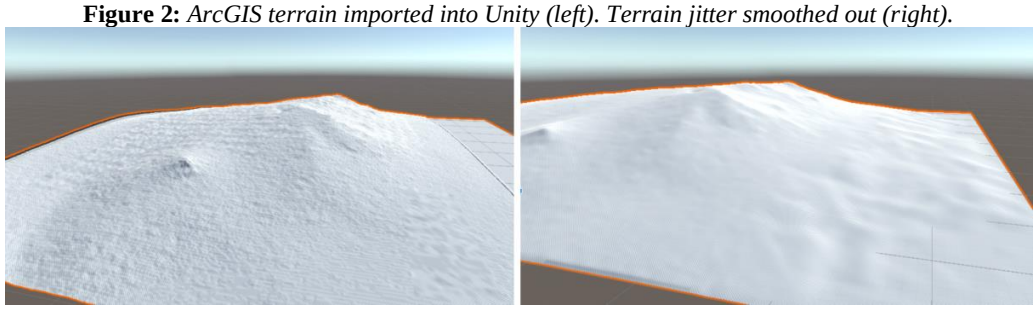
1.3 Terrain data acquisition and adjustment

The 3D terrain models, above which historical monuments were modeled, had been generated from the Digital Terrain Model of the Czech Republic of the 5th generation (DMR 5G). DMR5G was created by the Czech Office for Surveying, Mapping, and Cadastre, based on aerial laser scanning performed by the Military Geographic and Hydrometeorological Office (VGHMÚř).

DMR5G data are available as layer files (.lyr) that are implicitly processed in ArcGIS (in our case version 10.4). In the ArcGIS, the area of interest of the immediate vicinity of the historical monument was extracted (clipped). These extracted Digital Elevation Models (DEMs) were saved as TIFF files).

Upon converting the TIFF DEM into a RAW file, this was then imported into the Unity engine and processed further. As seen in Fig. 2 left, the imported DEM (in Unity terminology, heightmap) did introduce some jitter inaccuracies, given the resolution of the source. Therefore, using the Unity terrain tool (Unity Technologies, 2019a), the jittery terrain was smoothed out a bit to give it natural flow (see Fig. 2 right).

Figure 2: *ArcGIS terrain imported into Unity (left). Terrain jitter smoothed out (right).*



Source: 3D modeling course (Masaryk University, 2020)

Since the acquired terrain is contemporary, it may not completely resemble its historical shape (due to erosion, sedimentation, man-made activities, etc.) Nevertheless, the Unity terrain tools can be used to bring back the desired shape of the terrain, often in conjunction with placing man-made structures onto the terrain.

Since 3D visualization can have a rather large view distance, spanning up to tens of kilometers, the ArcGIS raster created is recommended to be a large one. While importing the terrain into the 3D engine, this can be further separated into two exclusive terrain objects: a near, highly detailed one, upon which the user walks, and a remote, low-detailed one, which covers the view distance. 3D engines can also use a legacy skybox feature (Unity Technologies, 2019b), but using real terrain data (despite being relatively low resolution) is better to preserve the overall visual fidelity.

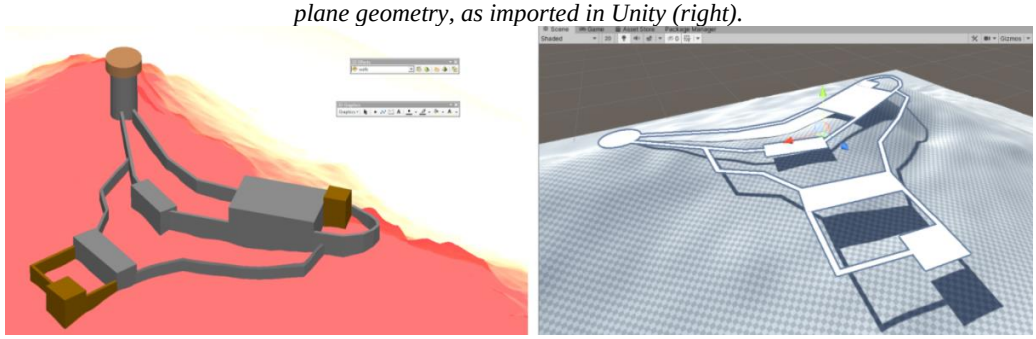
Other limitations of the DEMs can occur in case of the terrain being highly vertically differentiated - e.g., cliffsides, large rock formations. Because the DEM is only two-dimensional with a single height value per pixel, it cannot capture nonconvex variance along the vertical axis. In such cases, it is recommended to complement the DEM with 3D vertical models (e.g., obtained through photogrammetry), or to replace it completely with a voxel-based terrain (e.g. Unity Technologies, 2019c). Voxels, however, are yet not very well supported across the current generation of 3D applications, both in 3D engines and GIS software.

1.4 From acquiring the vector floor plan data to modeling the prototype

Floor plans of the man-made structures were created manually using appropriately visualized DMR5G data and other supporting data sources (geo-referenced plans, reconstruction drawings, photographs, or potential outcomes from a field survey). Using the ArcGIS extension 3D Analyst and their module ArcScene (ArcGIS, 2021), these shapes were then extruded into the third dimension and saved as so-called Multipatch geometries. Fig. 3 left provides an overview of what the reconstructed castle might look like. These were then also imported into Blender/Unity in the form of Shapefiles.

Since the Blender 3D modeling software has a tight integration with the Unity 3D engine, we used that (simplifying the 3D modeling workflow that was to follow by saving Blender files right into the Unity project, with no importing/exporting needed, as this software did such on their own). Blender, however, did not support the 3D data created in ArcScene (with multipatch geometry), making it effectively so that only 2D floor plans of castle structures were imported and had to be modeled into 3D once again (Fig. 3 right).

Figure 3: *original extrusion of the basic castle geometry in ArcGIS using ArcScene (left). Only flat plane geometry, as imported in Unity (right).*

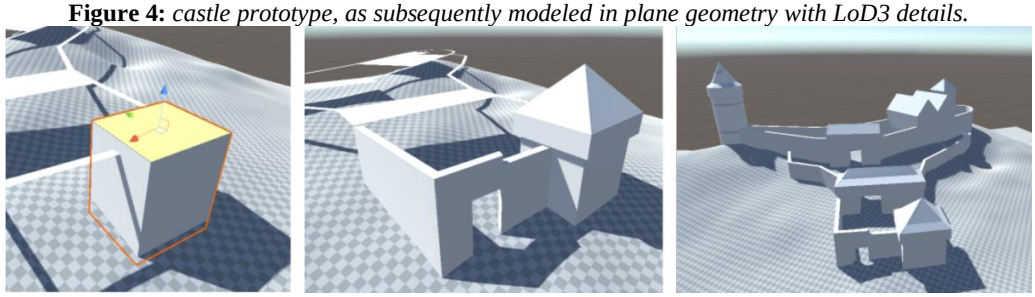


Source: 3D modeling course (Masaryk University, 2020)

Nevertheless, this was not a hindrance for two reasons: one, the 3D geometry to be made required more complexity than ArcScene provided (not just extrusions, but other advanced 3D operations as well), and two, to take this opportunity to teach our 3D modeling class students the principles and basics of the craft.

By modeling the castle from a series of planes with extruded sets of walls and buildings, all with basic doorways and similar physical structures in place, we were able to explain the basics of 3D hard-surface modeling to our students. They tried the different operations available in the vertex/edge/polygon modes and had to think about the desired 3D mesh topology of the objects they were creating (by adding in geometry through extruding beveling, or cutting loops into it; by reducing/optimizing the geometry through deleting polygons and joining vertices; or by using some basic modifiers like mirroring, to aid the modeling process). Such modeling progress is shown in Figure 4.

Figure 4: *castle prototype, as subsequently modeled in plane geometry with LoD3 details.*



Source: 3D modeling course (Masaryk University, 2020)

As seen in Figure 4, such level of detail (based on the level of detail 3 - LoD3, according to the CityGML standard (Kutzner, Chaturvedi, Kolbe, 2020)) was enough for the prototyping phase. We also provided the students with a basic first-person controller (e.g., Halldin, 2018) so that they could walk through their scene prototype and verify their surroundings from the first-person perspective, ensuring that all intended paths were of a correct scale and accessible. In other words, things like oversized fantasy-like towers, oversized stairs, or steep, unwalkable slopes had to be corrected.

While no other interactive scripts were present in our solution (given the already high complexity of the subjects presented to the students), the prototyping phase is also the time to

introduce these. Interactive object functionality, be it animated doors, texts, narration, etc., can be introduced by means of creating a script structure that will carry and deliver their information/interactive content as needed.

Upon finalizing the prototyping phase, we were able to build the first version of an executable, fully walkable 3D visualization application. While our students had different castle assignments to make, they could distribute their prototype builds to their peers to exchange feedback and make adjustments.

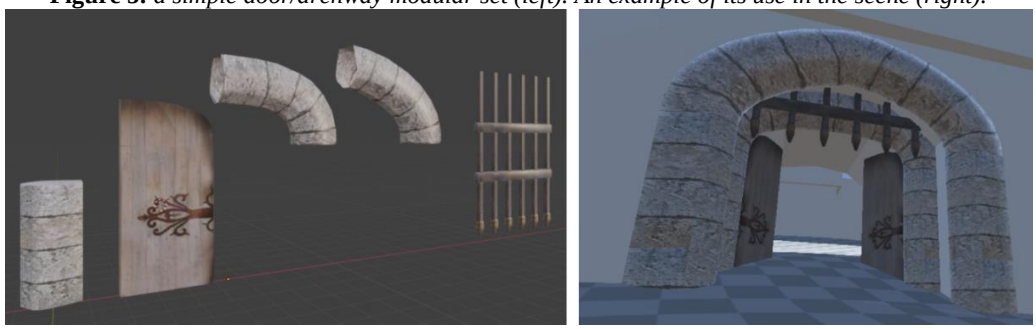
1.5 Modeling details and texturing

While the process of creating detailed small-scale models is in principle the same as LoD prototyping modeling (the models just get more complex at this point), the issues to be faced and the thinking behind the process are more complex.

First of all, modeling, texturing, and deploying small-scale objects into a scene can be the most time-consuming portion of the whole 3D scene creation pipeline. While being rather simple on the execution level (i.e., a repetitive process of recreating 3D assets repeatedly), this can easily take tens hundreds to thousands of man hours. As such, mismanagement of these efforts can be costly - it is, therefore crucial to have a unifying pipeline and approach in detailing a 3D environment. It is recommended to:

- Have a clear standard of what the 3D assets are to look like. That is, their expected level of detail in both 3D meshing and texturing, a (semi)standardized workflow of their creation, and their overall visual appearance/style. In other words, to have an art direction - in the case of historical visualization, this is likely to follow a realistic art style, but in other cases, e.g., education, something more stylized like conceptual, comic, geometric, or hyper-realistic direction can be used too (Kratt et al., 2014).
- Obtain 3D assets from external sources to save time and resources. For such occasions, there are many 3D marketplaces, some free, some paid. To name just a few, there is the Unity Assets Store, Unreal Marketplace, or Quixel Megascans. Similarly, in our 3D modeling class, we recommended the students share their 3D assets with their peers so that some of the common models would not need creating repeatedly.
- Reuse 3D assets and make some of them modular. It is a common practice in 3D scene detailing to reuse existing assets in multiple instances, to save memory, computing power, and asset creation requirements. Especially, organic, semi-random shapes like rocks can be rotated around their axes, scaled (within reason), and grouped with other similar assets to create unique looking vistas. As for mostly geometric, repetitive, man-made shapes and structures, such can be broken down into modular assets - that is, building blocks, often following a grid structure that can be connected to quickly and accurately create compelling man-made compositions. An example of such a library of modular 3D assets is shown in Fig. 5.

Figure 5: *a simple door/archway modular set (left). An example of its use in the scene (right).*

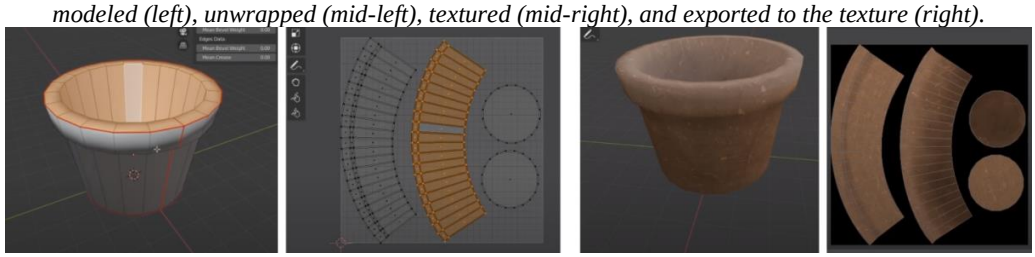


Source: 3D modeling course (Masaryk University, 2020)

In our 3D modeling class, we adhered to a simplified 3D asset modeling workflow that consisted of four steps (for a demonstration, see Fig. 6):

1. 3D mesh creation. Simple objects, just like the flower pot in Fig. 6, were advised to be created from imagination, complex objects to be based on a photoreference (ideally as a series of two to three orthogonal pictures, each per axis - i.e., front, side or top).
2. UV unwrapping. That is, projecting the 3D surfaces of a polygonized model onto strips of 2D planes so that they could be textured later on. Since Blender has integrated Smart UV unwrap, i.e., semiautomated functionality to do this (Selin, 2019), we advised our students to use this.
3. Texturing. This was also done inside Blender, thanks to the texture paint and stencil brush functionality (Blender Foundation, 2021). In principle, this step involves obtaining 2D textures from external sources and projecting them onto a 3D object, moving the 3D object around to obtain better projections in place, and finally, saving the projections represented as a bitmap on top of a UV projection as PNG bitmap files. This created the basic texture (also called albedo).
4. Importing the 3D models of assets into Unity - either through an implicit Blender to Unity converter, or by saving the assets as FBX files (a standardized format to maximize compatibility with other potential 3D modeling software).

Figure 6: *simplified 3D asset modeling workflow, as demonstrated on a flower pot. An object is modeled (left), unwrapped (mid-left), textured (mid-right), and exported to the texture (right).*



Source: Abbit, 2020

While the example workflow described above and shown in Fig. 6 is not exhaustive or producing best-looking 3D assets, it is reasonable enough to be adopted by a 3D modeling beginner, or by a person who strives to produce 3D assets in a reasonable time.

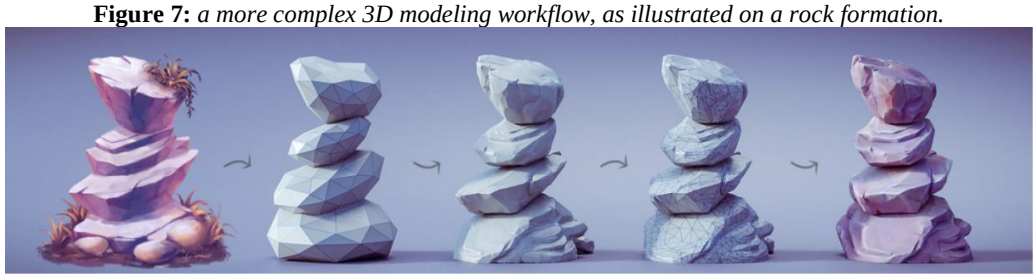
More complex 3D asset creation workflows often also introduce asset sculpting, retopology, baking, and professional-grade texturing (Sairiala, 2015). As illustrated through Fig. 7, a

simplified shape of the desired 3D mesh is created at first; then, it is subdivided and sculpted in high-polygon fine detail, often containing up to tens of millions polygons; a good enough low-polygon retopology of this mesh is then created, and the detail contained in the sculpted mesh is then projected onto it using normal maps. With such baked-in mesh, the texturing process can commence in a professional program like Substance Painter, which assists with procedural texture creation and placement based on the shape and form of the 3D mesh it is being applied on. The stack of specialized textures per 3D asset (i.e, not only albedo for color, but also a normal map for details, a specular map for reflections, an occlusion map for shadow casting, etc.) creates what is known as PBR material (physically-based rendering (Pharr, Jakob, Humphreys, 2016)), which is desired to simulate object material properties across all sorts of lighting settings - provided that scene lighting is crucial, dynamic, or when the purpose of the whole visualization is to reach photorealism (Borg, Paprocki, Madsen, 2014). The price for such a complicated 3D asset creation workflow is the time requirements, which easily span to a tenfold of the simple workflow mentioned above.

Another alternative to 3D asset modeling is their acquisition through photogrammetry (if such props are physically available). With specialized photography equipment (Lachambre, Lagarde, Jover, 2017), the photos can be obtained; then, a specialized photogrammetry software, e.g., Meshroom (Alicevision, 2021) converts these into a 3D mesh; using retopology software like Instant Meshes (Jakob et al., 2015), these can be converted into a mesh that is better suited for editing and 3D engines (all with preserving the normal map and the original texture), corrected to eliminate natural light shadows that may be present in the photos (Unity technologies, 2020), and used.

As for obtaining high-quality textures or materials, there are software packages, e.g., Materialize (Bounding Box software, 2018), that can recreate a whole PBR material based on emulating what would have otherwise required the high-polygon sculpting. While the resulting quality may not be a good fit for all objects and may not always be on par with the sculpting/baking workflow, it sure is a fast way to produce compelling results.

Figure 7: *a more complex 3D modeling workflow, as illustrated on a rock formation.*

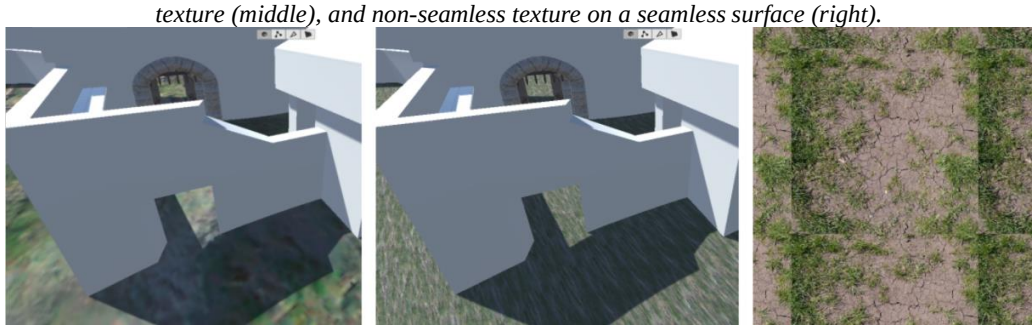


Source: Waterman, 2014

With the texturing workflow, one needs to make sure their surfaces do not suffer from issues - that is, the textures are either under scaled, overscaled, skewed in either X or Y axis, misaligned, having visible seams or repetition issues (see Fig. 8; this applies to both 3D textured objects, and texturing the terrain), or being too different compared to textures of other objects (a matter of consistency and art direction). While 3D objects are textured as aforementioned, terrains have to be texture painted as well - the reason for this is the user visiting a 3D reconstruction from a first-person perspective, where a satellite scan of the land would look just too blurry and oversized (e.g., Fig. 8 left). For terrain texturing, 3D engines like Unity have their terrain painting toolsets (for textures, their strength and blending, and oftentimes for low-height vegetation like grass, too); while the height of a 3D in-engine terrain is usually stored in a

heightmap bitmap, the same goes for terrain textures - functionally speaking, these are just more layers.

Figure 8: *some issues of texture scaling, use, and alignment. An overscaled texture (left), skewed texture (middle), and non-seamless texture on a seamless surface (right).*



Source: 3D modeling course (Masaryk University, 2020)

Completion of a whole 3D scene involves complementing and/or reworking the prototype building by introducing more detailed shapes (unique objects, modular objects, reused objects), terrain heightmap, texturing, vegetation, and clutter. Since man-made structures are oftentimes orderly and geometric, while vegetation and nature form pseudorandom patterns, the contrary of the two can create an interesting dichotomy. By introducing vegetation into a scene, repetitive patterns can be broken. Furthermore, an imperfection in 3D scene composition like texture repetition or 3D model imperfections can be hidden away to an extent. However, artsy, this may sound a good level of integration of all parts present in a scene creating quality environment art (Mon, 2017), which, in turn, makes the user feel comfortable in such a virtual world - i.e. directly improving the immersiveness of the experience (Hudson et al., 2018). What makes or breaks a virtual application is the end feeling of the user/customer. If their impression of a 3D visualization is that a scene is simply “not good enough” (be it because of obsolete or intrusive visuals, cramped space, conflicting user-controller interfaces, etc.), such users would likely not be inclined to visit the real representation of the virtual scene in question. In other words, if a virtual environment is used as a supporting tool to promote tourism, it needs to have a positive conversion rate per potential user/customer (if that is a metric to go by).

This is why the visuals and user interface of a 3D visualization with a real counterpart should cause good impressions in users; this is why they need to be integrated well together. That much can be said for 3D modeling alone; as for user testing for the positivity of such an effect, that is up for discussion.

1.6 Scene lighting, post-processing, and optimization

The purpose of this final stage of the 3D pipeline is two-fold: to make the premodeled 3D scene look presentable considering the context of its use, and to make it run well considering the budget of the hardware it is intended to run on.

Scene lighting is there to accentuate the form and shapes of the composition of the 3D models that were previously put in place. While there surely are dramatic lighting effects like dusk/dawn scenery, photographic magic hours, sun rays (in computer graphics also known as god rays), soft/hard light dichotomies, or reflective, ray-traced scenes (Birn, 2006), which is more of a domain of the film industry, artistic visualizations, or games. When lighting historical

visualizations, a more basic approach can do; from thereon, the question on lighting is whether the scene is situated fully outdoors, or whether it includes indoor space.

Generally speaking, with lighting, it is important to consider light sources. With outdoor spaces, just a single global light source can do - sunlight. This is a light source coming from a distance of infinity, with a global reach across the whole visualization is by default enabled. For a straightforward visualization without a cinematic touch, it is recommended to rotate the sunlight in the vertical axis to achieve an effect of a 3 PM to 5 PM sun, or so. The reason for this is that noon lighting looks bland, not giving the shadows any chance to cast themselves across the existing 3D geometry to accentuate the three-dimensionality of the scene; similarly, early morning or late evening lighting has the issue of casting shadows so long that it is no longer clear what shape originates what shadow and the whole lighting composition gets overlayed by shadows from all sorts of origins (this is especially true when the 3D scene is vertically differentiated - as a castle scene might be).

If a scene contains other light sources, e.g., in indoor spaces with little/no sunlight reach, these are to be included as well. In 3D engines, there are usually two basic kinds of local lights: point lights, which distribute their light in all directions equally, and spotlights, which have a limited angle, in which they cast a light cone (Unity Technologies, 2019d). Such local lights can be adjusted for their limited range and color, among other properties. A deliberate play of light and shadow can give a scene an atmospheric feel, especially in indoor spaces.

Scene postprocessing is a collection of camera filters and effects that can further accentuate the artistic feel of a visualization. Some of them are based on their film or photography counterparts (e.g., color grading, depth of field, vignetting), some others are a domain of computer graphics (e.g., antialiasing, distance fog simulation). While these can be used to hide some imperfections of computer graphics, push the depth of three-dimensionality and photorealism of visualization even further, we do recommend using them sparsely. Not only are they overused in computer graphics media, but they can also be very demanding in terms of performance.

Speaking of performance and scene optimization, it is the beginning of the 3D visualization pipeline (conceptual phase), where the overall requirements for 3D visuals and intended platforms are set. While scene optimization is useful, one does not simply, e.g., convert a lightweight, simplistic scene intended for mobile phones into a photorealistic state of the art virtual reality experience or vice versa. In professional 3D pipelines, the conceptual phase sets a budget (in terms of scene complexity and performance requirements), the finalizing optimizations make sure the budget is not exceeded, along with striving for some reserves. This budget often involves the complexity of 3D geometry (number of polygons simultaneously rendered in a scene), lighting complexity and number of light sources, the size of textures, and the use of postprocessing filters.

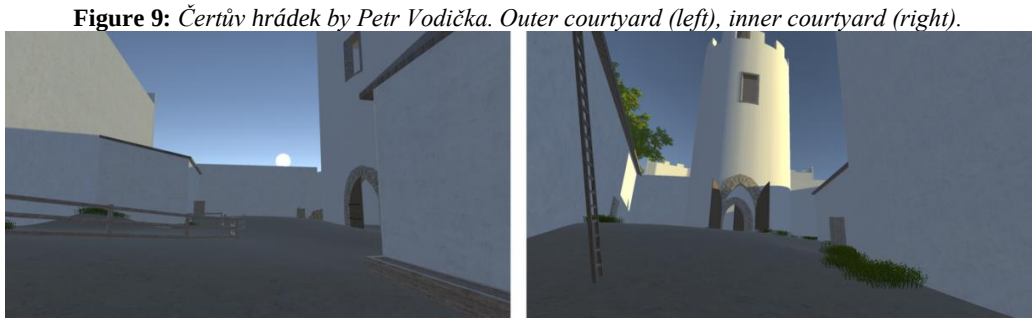
While scene optimization can be a complex subject, and while this exceeds both the depth of this article and the topics covered in our 3D modeling course, as taught to our students, adhering to the simple 3D modeling pipeline, as described in this article, ensures performance limits are unlikely to be reached (i.e. a student presenting their semestral work on a standard PC with a dedicated graphics card).

2. Results

Each student in our 3D modeling class was tasked to pick a unique castle they would model. Through the concept and prototyping phases of our simple 3D modeling pipeline, the students were advised on their progress, so that the amount of work necessary in one semester's timeframe would be manageable, without complications.

Figures 9-12 present screenshots of the final visualizations of four selected students. The screenshots were taken from the first-person perspective while walking through the virtual environments. This is to illustrate what was made possible in a timespan of one semester; it is also apparent that the students did approach the assignment differently, putting their personalities into the end products.

Figure 9: *Čertův hrádek by Petr Vodička. Outer courtyard (left), inner courtyard (right).*



Source: Petr Vodička semestral 3D modeling project (Masaryk University, 2020)

Figure 10: *Lichnice, by Damián Hruban. Outskirts of the castle (left), courtyard (right).*



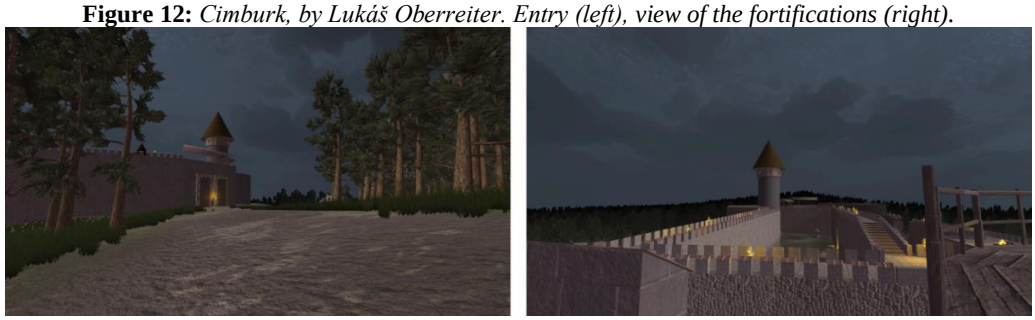
Source: Damián Hruban semestral 3D modeling project (Masaryk University, 2020)

Figure 11: *Zlenice by Rostislav Hořava. Entry gate (left), inner gate (right).*



Source: Rostislav Hořava semestral 3D modeling project (Masaryk University, 2020)

Figure 12: *Cimburk, by Lukáš Oberreiter. Entry (left), view of the fortifications (right).*



Source: Lukáš Oberreiter semestral 3D modeling project (Masaryk University, 2020)

3. Discussion and Conclusions

Throughout the article, we argued for the utility of creating 3D visualizations intended to depict historical buildings (be it as a supplementary to physical tourism, or as an alternative to it). Methods for acquiring depictions of the buildings were mentioned - were it methods applicable to fully preserved or completely derelict historical sites. We then considered the 3D methods intended to bring the buildings onto a virtual three-dimensional canvas.

A simplified 3D modeling pipeline for 3D historical visualization was presented in the body of the article. Its viability, in a pedagogical sense, was also verified by our students from a 3D modeling class, who were tasked to create 3D visualizations of selected castles in a timeframe of one semester.

Although the outcomes of this article are limited, it is so due to the complexity of the subject. Thus, the discussion will consider what are the additional steps should the craft of 3D visualization/modeling be pursued further in the context of geographical sciences and the tourism industry.

3.1 Pedagogical perspective

Were the finalized student visualizations presented in the results chapter success or failure? That depends entirely on the perspective of things.

A professional 3D environment artist would say that the visualizations do not meet today's criteria for the level of detail introduced into the scenes and that the modeling has some apparent errors to it. In other words, in today's market, with consumers and enthusiasts being used to a certain level of quality of 3D graphics, the aforementioned visualizations would likely not impress everyone, was it a commercial market.

The creators of the presented visualizations were, however, no professional environment artists with years of experience; these were university students with no previous background in 3D modeling. Therefore, the perspective here is entirely different: to take someone with no understanding of the subject, and to guide them through it, to make them understand the basics and concepts behind it. While the craft behind 3D modeling does indeed take years to master (e.g., in being able to produce a professional level of quality in organic modeling and anatomy), the castle scene seems to be something even a beginner can handle, with reasonable results. To ensure positive results, however, our students were given feedback along the whole

visualization pipeline; and it needs to be stressed that this kind of teaching is suitable for small groups of students, as there is a lot of feedback to give out (our class totaled 8 students).

The students did spend some 20 to 50 hours each on the most consuming phase of the project - i.e., the modeling phase. While a professional may be somewhat faster, such a time budget seems reasonable for a work of depicted quality. The students were also taught not to waste their time by following the recommended 3D visualization pipeline and its software workflows. If they were to decide to follow up on their 3D education, the subject of our 3D modeling class would serve as a good foundation.

It also needs to be noted that teaching future generations of geographers 3D modeling has its relevance. While 2D mapping has had its tradition and utility (Carbonell-Carrera, Jaeger, Shipley, 2018), an increasing number of geographic applications can and do use 3D implementation and visualizations. It is thus more valuable for the scientific field to have skilled experts who can implement or edit 3D applications, as opposed to mere passive observers.

3.2 Technological perspective

Through the 3D visualization pipeline, there are many advanced techniques, software, and plugins to aid or alter the production workflow. Beyond the covered basics, this article only refers to some of the well-known ones, as of 2020. Even though advanced solutions may be out of the scope of this article or an introductory course in 3D modeling, some of the otherwise advanced stuff may find more utility and adoption in the future. For example, there is now procedural art, even in 3D (Matejka et al., 2018), or artificial intelligence-aided texturing and stylization available (Unity Technologies, 2021); it was not so just 10 years ago. Similarly, virtual reality solutions only had appeared on the scene of 3D graphics in the first half of the previous decade, as prototypes; in the second half of the previous decade, production models of virtual reality hardware were made available, with ever increasing availability, user controllability, and overall comfort. Such technologies are here to stay - if not to evolve.

We do not have a universal answer on the implementation of a great 3D application, regardless of the context. Such an answer will always be difficult to grasp, as it depends on many factors out of our current scope: the technology used, the intended level of interactivity of the application, the expected user profile, etc. This is fit for discussion and for future articles - that is, finding the best solutions, graphics pipelines, and workflows in applied settings, or expanding the methodology.

Methodology-wise, user interactivity and user behavior are topics to be considered. While some interactivity depends on the physical hardware used (e.g., an LCD mouse and a keyboard versus a virtual reality headset with controllers), virtual counterparts of physical interactions can be programmed into the 3D application (Ugwitz et al., 2021). In virtual sightseeing applications, implementing some interactivity may aid in having the user behave as intended (e.g., to make them follow along a designed tour path). As for user behavior in general, it can be logged and analyzed; user behavior also depends on the composition of the virtual environment. Evaluating (and adjusting) the spatial composition of environments with known preferences of human users (Wilson, 1999) can improve user enjoyability regardless of the context - be it virtual sightseeing, or something else.

3.3 Interdisciplinary perspective

This article takes on the task of presenting and considering 3D modeling, in the context of historical visualizations, to be utilized by the tourism industry. And yet, the authors are no star contributors to everything considering 3D modeling, historical reconstruction, or tourism. In other words, there are more relevant scholars or experts out there. Where exactly are they?

Some 3D modelers, environment artists, and concept artists are self-taught, and some others originate from artistic or technical schools/universities; professionally, they tend to gravitate towards game development studios, film, product design, or architecture. Historical reconstruction using modern technology, e.g., photogrammetry in archaeology, is a niche field. Other historians tend to gravitate towards game studios (Bostal, 2019), or to projects of their own (Univerzita Hradec Králové, 2021). Some geographers explore research niches by depicting historical geospatial data in 3D (Tobiáš, Cajthaml, Krejčí, 2017).

If better visualizations are to be produced, more connections between the people involved in the aforementioned fields/professions need to be established. Modeling a few castles with students is one thing, but there is also untapped potential in professional, applied solutions.

References

ABBIT, Grant. *Making Simple Game Assets* [online] 2020 [cit 2021-1-31]. Available from: https://www.youtube.com/watch?v=fEXDylc0Rgg

ALICEVISION. *Meshroom* [online] 2021 [cit 2021-1-31]. Available from: https://alicevision.org/#meshroom

ANGELO, Dante. Histories of a Burnt House. An Archaeology of Negative Spaces and Dispossession. *American Anthropologist 119(2)*. 2017. 253-268. https://doi.org/10.1111/aman.12868

ARCGIS. ArcMap. *3D Analyst and ArcScene* [online] 2021 [cit 2021-1-31]. Available from: https://desktop.arcgis.com/en/arcmap/latest/extensions/3d-analyst/3d-analyst-and-arcscene.htm

ASCHER, Robert. Experimental archeology. *American Anthropologist 63(4)*. 2017, 793–816

BERTOLINI, Lucas. *Hands-On Game Development without Coding: Create 2D and 3D games with Visual Scripting in Unity*. Birmingham: Packt, 2018. 404p, ISBN 978-1789538335

BIRN, Jeremy. *Digital Lighting And Rendering*. New Riders Pub, 2006. 416p, ISBN 978-0321316318

BLENDER FOUNDATION. Blender docs. *Texture Paint* [online] 2021 [cit 2021-1-31]. Available from: https://docs.blender.org/manual/en/latest/sculpt_paint/texture_paint/index.html

BORG, Mathias, PAPROCKI, Martin M., MADSEN, Claus B. Perceptual evaluation of photo-realism in real-time 3D augmented reality. In: *International Conference on Computer Graphics Theory and Applications (GRAPP)*, 2014, 1-10

BOSTAL, Martin. Medieval video games as reenactment of the past: a look at Kingdom Come: Deliverance and its historical claim. In: *Del siglo XIX al XXI. Tendencias y debates: XIV Congreso de la Asociación de Historia Contemporánea*, 2019, 380-394. Available from: http://rua.ua.es/dspace/handle/10045/95829

BOUNDING BOX SOFTWARE. Materialize [online] 2018 [cit 2021-1-31]. Available from: http://boundingboxsoftware.com/materialize/

CARBONELL-CARRERA, Carlos, JAEGER, Allison J., SHIPLEY, Thomas F. 2D Cartography Training: Has the Time Come for a Paradigm Shift? *ISPRS International Journal of Geo-Information 7(5)*. 2018. https://doi.org/10.3390/ijgi7050197

CHING, Francis, JARZOMBEK, Mark M. and PRAKASH, Vikramadi. *A Global History of Architecture*. New York: Wiley, 2006. 864p, ISBN 978-1118981337

DAILEY-HEBERT, Amber, SIMMONS ESTES, Judi, CHOI, Dong Hwa. *Current and Prospective Applications of Virtual Reality in Higher Education*. IGI Global, 2021. http://doi:10.4018/978-1-7998-4960-5.ch001

DAVIDSON CRAGOE, Carol. *How To Read Buildings: A crash course in architecture*. Brighton: Ivy Press, 2017. 256p, ISBN 978-0847831128

DENG, Yilin, HAN, Sang-Yun, LI, Jianyi, RONG, Jinjin, FAN, Wenuy, SUN, Tiancong. The design of tourism product CAD three-dimensional modeling system using VR technology. *PLoS ONE 15(12)*. 2020. https://doi.org/10.1371/journal.pone.0244205

FERRO, Lauren, SAPIO, Francesco. *Unity 2017 2D Game Development Projects*. Packt Publishing, 2018. 338p. ISBN: 978-1786460271

HALLDIN, M. Ludiq. *First Person Controller* [online] 2018 [cit 2021-1-31]. Available from: https://support.ludiq.io/en/communities/5/topics/1832-first-person-controller

HUDSON, Sara, MATSON-BARKAT, Sheila, PALLAMIN, Nico, JEGOU, Guillaume. With or without you? Interaction and immersion in a virtual reality experience. *Journal of Business Research, 100*. 2018. 459-468. https://doi.org/10.1016/j.jbusres.2018.10.062

JAKOB, Wenzel, TARINI, Marco, PANOZZO, Daniele, SORKINE-HORNUNG, Olga. Instant field-aligned meshes. *ACM Transactions on Graphics 34(6)*. 2015. https://doi.org/10.1145/2816795.2818078

JONES, Christine A., CHURCH, Elizabeth. Photogrammetry is for everyone: Structure-from-motion software user experiences in archaeology. *Journal of Archaeological Science: Reports 30*. 2020. https://doi.org/10.1016/j.jasrep.2020.102261

KOEVA, Mila N. 3D modelling and interactive web-based visualization of cultural heritage objects. In: *International Archives of the Photogrammetry, Remote Sensing and Spatial*

Information Sciences, XLI-B5, 2016, 297-303. https://doi.org/10.5194/isprs-archives-XLI-B5-297-2016

KRATT, Julian, EISENKEIL, Ferdinand, PIRK, Sören, SHARF, Andrei, DEUSSEN, Oliver. Non-realistic 3D object stylization. In: *Proceedings of the Workshop on Computational Aesthetics (CAe '14)*, 2014, 67–75. https://doi.org/10.1145/2630099.2630102

KUTZNER, Tatjana, CHATURVEDI, Kanishk, KOLBE, Thomas H. CityGML 3.0: New functions open up new applications. *PFG–Journal of Photogrammetry, Remote Sensing and Geoinformation Science, 88(1)*. 2020. 43-61. https://doi.org/10.1007/s41064-020-00095-z

LACHAMBRE, Sébastien, LAGARDE, Sébastien, JOVER, Cyril. *Photogrammetry Workflow*. Unity Technologies, 2017. 97p. Available from: https://unity3d.com/files/solutions/photogrammetry/Unity-Photogrammetry-Workflow_2017-07_v2.pdf

LARMAN, Craig, BASILI, Victor R. Iterative and incremental developments. A brief history. *Computer 36(6)*. 2003. 47-56. https://doi.org/10.1109/MC.2003.1204375

MASARYK UNIVERSITY. Information System. *Z8311 3D modelování a vizualizace* [online] 2020 [cit 2021-1-31]. Available from: https://is.muni.cz/predmet/sci/jaro2020/Z8311

MATEJKA, Justin, GLUECK, Michael, BRADNER, Erin, HASHEMI, Ali, GROSSMAN, Tovi, FITZMAURICE, George. Dream Lens: Exploration and Visualization of Large-Scale Generative Design Datasets. In: *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18)*, 2018, 1-12. https://doi.org/10.1145/3173574.3173943

MCCONNELL, Steve. *Rapid Development: Taming Wild Software Schedules*. Microsoft Press, 1996. 672p, ISBN 978-1556159008

MON, Sierra. ArtStation Magazine. *Everything You Need to Know to Become a Game Environment Artist* [online] 2017 [cit 2021-1-31]. Available from: https://magazine.artstation.com/2017/03/game-environment-artist/

MURPHY, M., MCGOVERN, E., PAVIA, S. Historic building information modelling (HBIM). *Structural Survey 27(4)*. 2009. 311-327. https://doi.org/10.1108/02630800910985108

PHARR, Matt, JAKOB, Wenzel, HUMPHREYS, Greg. *Physically based rendering: From theory to implementation*. Cambridge: Elsevier, 2016. 1266p. ISBN 978-0128006450

PULLI, Kari, AARNIO, Tomi, MIETTINEN, Ville, ROIMELA, Kimmo, VAARALA, Jani. *Mobile 3D Graphics with OpenGL ES and M3G*. Morgan Kaufmann, 2007. ISBN 978-0-12-373727-4

RACZKOWSKI, Włodzimierz. Aerial Archaeology. In: Encyclopedia of Global Archaeology. Springer, 2017. https://doi.org/10.1007/978-3-319-51726-1_1504-2

SAIRIALA, Joonas. *PBR workflows in Cycles Render Engine*. Tampere University of Applied Sciences, 2015. Bachelor's thesis

SCERRI, Moira, EDWARDS, Deborah, FOLEY, Carmel. The value of architecture to tourism. In *Proceedings of 26th Annual CAUTHE Conference*, 2016, 1-21

SELIN, Erik. Artistic Render. The definitive tutorial to UV mapping in Blender. [online] 2019 [cit 2021-1-31]. Available from: https://artisticrender.com/the-definitive-tutorial-to-uv-mapping-in-blender/

SIMONDS, Ben. Blender Master Class: A Hands-On Guide to Modeling, Sculpting, Materials, and Rendering. No Starch Press, 2013. 288p, ISBN 978-1593274771

SINGER, Graham. *The History of the Modern Graphics Processor* [online] 2021 [cit 2021-1-31]. Available from: https://www.techspot.com/article/650-history-of-the-gpu/

TOBIÁŠ, Pavel, CAJTHAML, Jiří, KREJČÍ, Jiří. Rapid reconstruction of historical urban landscape: The surroundings of Czech chateaux and castles. *Journal of Cultural Heritage 30(2)*, 2017, 1-9. https://doi.org/10.1016/j.culher.2017.09.020

TOMLIN, W. Craig. What's a Persona?. In: UX Optimization. Apress, Berkeley, 2018. ISBN 978-1484238660. https://doi.org/10.1007/978-1-4842-3867-7_2

UGWITZ, Pavel, ŠAŠINKOVÁ, Alžběta, ŠAŠINKA, Čeněk, STACHOŇ, Zdeněk, JUŘÍK, Vojtěch. Toggle Toolkit: A Tool for Conducting Experiments in Unity Virtual Environments. *Behavior Research Methods*. 2021. https://doi.org/10.3758/s13428-020-01510-4

UNITY TECHNOLOGIES. Unity documentation. *Terrain* [online] 2019 [cit 2021-1-31]. Available from: https://docs.unity3d.com/Manual/script-Terrain.html

UNITY TECHNOLOGIES. Unity documentation. *Skyboxes* [online] 2019 [cit 2021-1-31]. Available from: https://docs.unity3d.com/Manual/skyboxes.html

UNITY TECHNOLOGIES. Unity asset store. *Ultimate Terrains - Voxel Terrain Engine* [online] 2019 [cit 2021-1-31]. Available from: https://assetstore.unity.com/packages/tools/terrain/ultimate-terrains-voxel-terrain-engine-31100

UNITY TECHNOLOGIES. Unity documentation. *Types of light* [online] 2019 [cit 2021-1-31]. Available from: https://docs.unity3d.com/Manual/Lighting.html

UNITY TECHNOLOGIES. Unity asset store. *De-lighting tool* [online] 2020 [cit 2021-1-31]. Available from: https://assetstore.unity.com/packages/tools/utilities/de-lighting-tool-99583

UNITY TECHNOLOGIES. *Unity ArtEngine* [online] 2021 [cit 2021-1-31]. Available from: https://unity.com/products/unity-artengine

UNIVERZITA HRADEC KRÁLOVÉ. Věnná města českých královen [online] 2021 [cit 2021-1-31]. Available from: https://www.kralovskavennamesta.cz/index.html

WATERMAN, Patricia. *Trends in the 3D Art Production Pipeline for Games 2014* [online] 2014 [cit 2021-1-31]. Available from: https://watermanp.wordpress.com/trends-in-the-3d-art-production-pipeline-2014/

WEISS, Yannick, HEPPERLE, Daniel, SIESS, Andreas, WOLFEL, Matthias. What User Interface to Use for Virtual Reality? 2D, 3D or Speech–A User Study. In: *2018 International Conference on Cyberworlds (CW)*, Singapore, 2018, 50-57. https://doi.org/10.1109/CW.2018.00021

WILSON, John R. Virtual environments applications and applied ergonomics. *Applied Ergonomics, 30(1)*. 1999, 3–9. https://doi.org/10.1016/s0003-6870(98)00040-4

Acknowledgment

We would like to thank our students from Masaryk University, Faculty of Science, Department of Geography, who participated in the summer 2020 course "3D modelování a vizualizace". Namely: Simona Bočková, Rostislav Hořava, Damián Hruban, Pavel Kubíček, Lukáš Oberreiter, Jan Přikryl, Aneta Ryglová and Petr Vodička.

Thanks to your input and consideration, we were able to refine the taught subject, which in turn allowed us to publish it in this form.

Annex 2: Paper II

Ugwitz, P., Juřík, V., Herman, L., Stachoň, Z., Kubíček, P., Šašinka, Č. (2019). Spatial analysis of navigation in virtual geographic environments. *Applied Sciences*, 9(9). https://doi.org/10.3390/app9091873

applied sciences

MDPI

Article

Spatial Analysis of Navigation in Virtual Geographic Environments

Pavel Ugwitz [1,3,*], Vojtěch Juřík [2,3,*], Lukáš Herman [1], Zdeněk Stachoň [1,3], Petr Kubíček [1] and Čeněk Šašinka [3]

[1] Department of Geography, Faculty of Science, Masaryk University, Kotlářská 267/2, 61137 Brno, Czech Republic; herman.lu@mail.muni.cz (L.H.); zstachon@geogr.muni.cz (Z.S.); kubicek@geogr.muni.cz (P.K.)

[2] Centre for Experimental Psychology and Cognitive Sciences, Department of Psychology, Faculty of Arts, Masaryk University, Arna Nováka 1/1, 60200 Brno, Czech Republic

[3] Division of Information and Library Studies, Faculty of Arts, Masaryk University, Arna Nováka 1/1, 60200 Brno, Czech Republic; cenek.sasinka@mail.muni.cz

* Correspondence: ugwitz.pavel@mail.muni.cz (P.U.); jurik.vojtech@mail.muni.cz (V.J.); Tel.: +420-549-49-7608 (P.U. & V.J.)

Received: 6 April 2019; Accepted: 30 April 2019; Published: 7 May 2019



Featured Application: Methods of analyzing user behavior in virtual environments, such as spatial movement and interaction patterns in individuals or within groups, user interface usage.

Abstract: Human performance and navigation activity in virtual environments can be measured and assessed with the aim to draw specific conclusions about human cognition. This paper presents an original virtual geographic environment (VGE) designed and used for this purpose. The presented research is rooted in an interdisciplinary approach combining knowledge and principles from the fields of psychology, cartography, and information technologies. The VGE was embedded with user logging functionality to provide a basis from which conclusions about human cognitive processes in a VGE could be drawn. The scope of this solution is introduced, described, and discussed under a behavioral measurement framework. An exploratory research design was adopted to demonstrate the environment's utility in proof-of-concept user testing. Twenty participants were observed in interactive, semi-interactive and non-interactive tasks, their performance and individual differences were analyzed. The behavioral measurements were supplemented by Object-Spatial Imagery and a Verbal Questionnaire to determine the participants' cognitive styles. In this sample, significant differences in exploration strategies between men and women were detected. Differences between experienced and non-experienced users were also found in their ability to identify spatial relations in virtual scenes. Finally, areas for future research areas and development were pinpointed.

Keywords: spatial cognition; spatial navigation; virtual geographic environment; visual cognitive styles; spatial data collection; spatial data analysis; exploration strategies

1. Introduction

Virtual reality (VR) elements are increasingly used in daily life. After a recent period of disillusionment with virtual reality [1], technological progress (displays, graphics) is now proving VR's strengths and has established it as a useful and practical tool in many areas of human endeavor. VR can be used to construct virtual geographic environments (VGEs), i.e., specific computer-based environments designed for studying geographic and cognitive phenomena and solving problems in geo-related tasks [2–4]. There is a considerable trend in exploring interactive VGEs and discovering

details about their features and using their benefits to create, customize and optimize geographic products. Some researchers [3–6] consider VGEs as a new generation of geographic analytical tools that provide an open digital window into the real world as a means of understanding human cognition and behavior in geographical issues. VGEs can be relevant tools in fields such as urban planning [7,8], crisis management [5,9,10], air traffic control [11], or generally speaking, fields that require support for quick and adequate human decision-making processes based on spatial reasoning.

The term virtual geographic environment (VGE) is rather broad in its definition and usually represents 3D geo-visualization with an enormous range of functionalities. VGEs represent, for example, multi-user, multi-faceted, dynamic geographical products designed for conducting geo-analysis, supporting collaboration, and making predictions, etc. [3]. As such, VGE is a specific type of virtual environment with complex options suitable for specific tasks and extended with different features (e.g., control options, view modes, texture pattern controls). Recent efforts to bring the effect of presence (i.e., the feeling of being surrounded by and/or a part of a (visual) stimulus) to immersive VGEs [12] reveal more valid data about users' real behaviors. These environments are presented in first-person view in a manner that resembles first-person computer games. In respect of the above-mentioned arguments, the user aspects of VGEs should be subject to specific research and testing and their possibilities and limitations further explored. In this paper, we discuss the context of the use of VGEs and present an original VGE with embedded functionalities for data collection. The environment was tested in relation to user behavioral activity and cognitive ability. We also discuss the possibilities, limitations, and potential implications of VGEs in research and practice.

1.1. User Testing and Evaluation of VGEs

In many areas, current VGEs are still lacking various properties (e.g., fidelity of computer graphics, sensory quality and immersion, ease-of-use of UIs, etc.). Even though VR is continually evolving, research on VGEs was once (and to some extent still is) constrained by huge technological limits [13]. Previous studies (pre-2010's) on this issue are both rare and questionable in their topicality. Recent strides in VR have provided a great opportunity to make use of realistic and usable simulations for accurate feedback and an unprecedented level of immersion (e.g., 3D visualization). However, the delivery of a persuasive stimuli exposition must also come with detailed recording and subsequent behavioral input analysis (i.e., user interaction with VGE), which plays an important role in the human perception, understanding, and subsequent processing (via interaction) of tasks [14]. From an experimental point of view, VGEs can be considered two-way products. One thing is that VGEs can serve as a controlled experimental stimuli exposition tool, even for potentially complex properties or visual patterns (especially dynamic animations, interactive objects, realism-inducing simulations, etc.); the other advantage is that the users interaction with such controlled virtual environment can be precisely measured/recorded and specific (inter)action patterns can be identified there. This opens up many potential lines of research in view of the fact that VR may help increase the ecological validity of research and experimental control.

With a specifically designed VGE in mind, different types of VR settings (e.g., types of visualization) can be measured and compared. This was previously explored in several studies examining the use of 3D technology in geo-visualizations, for example, References [15–19]. In geography especially, the effects of different visualization types have been compared, for example, between interactive versus static 3D visualizations [16,20], photorealistic versus non-photorealistic 3D visualizations [21,22] and eventually visualizations with different ratios of realistic/abstract elements [23–25]. How suitable a visualization is to a user depends heavily on the purpose of the task. Beyond that, the individual differences between users (or groups of users) can also be studied. The approach of measuring individual differences has a long tradition, stemming largely from psychology and behavioral sciences [26]. We reason that a well-structured VGE with proper functionality and supplemented by psychological testing can effectively and quickly provide a large amount of complex user behavioral data, and as such may be used to develop strategies in user interface (UI) optimization and individual customization.

The process of adding more than one method of user testing and monitoring to achieve complex insight into user activity is covered by the concept of methodological/research triangulation.

1.2. Spatial Analysis of User Navigation and Interaction

In recent years, progress in the field of geo-visualization has brought about studies on user strategies that make use of cartographic visualizations. For example, Manson et al. [27] explored the usability of web mapping navigation in different interfaces. Using the combined metrics from a computer mouse and eye-tracking, they found significant differences between different types of interface and spoke in favor of the standard GIS approach of rectangle zoom, followed closely by wheel zoom. Wilkening and Fabrikant [28] employed a similar approach to evaluate the effect of time constraints on the usability of a 3D geo-visualization (Google Earth displayed on a computer screen and controlled with a computer mouse). Participants dealt with practical tasks such as assessing the elevation of given points. The proportionality of applied movement interaction types was monitored—this constituted panning (the most common in both experimental conditions), zooming, tilting, and rotating. Interaction with geo-visualizations can also be analyzed in more detail; You et al. [29] compared different pan and zoom function settings. Individual types of virtual movement can also be compared in control devices other than conventional computer mice, i.e., a touch screen was used by Glawischnig and Mahdavi [30] and a Wii remote controller was used by Juřík et al. [14].

The result of interaction with a VGE is usually movement within this environment, which is defined as navigational (or viewpoint) interactivity [31]. Movement of a viewpoint can be represented by a so-called travel path [32]. Analysis of user travel paths is a research method used primarily in studies of human-computer interaction (HCI) [33,34] and 3D UIs [35]. Treves et al. [36] used and analyzed travel paths to distinguish between two types of geo-visualizations (non-guided and guided) in the Google Earth software. Herman and Stachoň [20] and Herman et al. [16] visualized user travel paths as trajectories when differentiating individual types of movement (zoom, pan, rotation) in a 3D geo-visualization. Additional visual or GIS-related analysis can be applied to user trajectories. Zanbaka et al. [34] and Cirio et al. [35] used visualizations of the paths of multiple users. Chittaro and Ieronutti [33] and Zanbaka et al. [34] used heatmaps to depict presence density. As Bowman et al. [32] suggested, traveled distance and the amount of rotation or turns can be measured. When time (efficiency) and traveled distance is measured, the speed of a user's travel can be calculated [16,20,37].

The use of VGEs lets us examine the influence of specific visualization UI settings on human interaction with the virtual world, especially considering user fluency/speed (efficiency) and correctness (effectiveness) in various perceptual and decision-making tasks. To better understand the whole process of interaction, user strategies must be analyzed while following defined goals. To precisely describe a user's strategy, we need to capture a wide range of behavioral information, such as user movement trajectory (distances or average speed) or control types used for navigation (e.g., pan, zoom or rotation of the viewpoint). Based on the principle of triangulation, which is conditioned by the controlled setting of a virtual environment, data collected in such VGEs are complex and can help reveal specific patterns in human behavior.

1.3. Cognitive Styles and Route Preferences in VR Navigation

According to previous psychological research, an individual's spatial skills such as navigation and orientation in space are determined by a specific cognitive style [38]. Cognitive style was originally defined as a psychological dimension representing the consistency of an individual's information processing [39]. Currently, we understand the concept as a way of acquiring, organizing, and processing information [40], or as a manifestation of the means of personality traits [41]. Blazhenkova and Kozhevnikov [42] discussed cognitive styles from the perspective of visual processing ability and suggested that people can distinguish between object visualizers and spatial visualizers (that these are the two diverging cognitive approaches in processing visual data). The existence of spatially-oriented people or those who are more object-oriented or verbally-oriented was previously explored in several

studies [43–45] and highlighted the existence of inter-individual differences between people in these cognitive styles. We can see a cognitive style as a specific mental setting that may influence the perception and cognitive elaboration of specific perceived content. Through this process, active human behavior (e.g., navigation, locomotion in an environment, judgement, etc.) can also be affected. The construct of visual cognitive styles can be measured, for example, by the Object-Spatial Imagery and Verbal Questionnaire (OSIVQ) developed by Blazhenkova and Kozhevnikov [42]. In this study, we explored the relationship between participants' self-reported visual cognitive styles and their performance in the VGE concerning their movement trajectories (among others). The data obtained concerning the participants' cognitive styles and behavior in the VGE was analyzed.

The understanding of spatial relations, spatial context and orientation in space was covered under the process of wayfinding [46]. Previous research explored how real spatial relations translated into internalized spatial knowledge, i.e., cognitive maps [47,48]. Other research described the typology of space according to which this knowledge was created [49,50]. Exploration of physical space is an ongoing process that integrates and links recently acquired spatial knowledge with knowledge that is already known [51]. Superficially, this is the process of route selection [52] and has been explored in many contexts, for example, behavior during emergencies [53], vehicle navigation [54], animal behavior [55], child behavior [56], and others. The VGEs presented in this paper were purposefully designed to accommodate for the aforementioned theories. This applies to overall spatial layouts, sectioned areas, paths, crossroads, and even the props of the environment—all of which draw from the concepts covered by wayfinding and the visual cognitive styles theory. Even though the VGE visuals are made out of freely available resources and can be shared across different installations using the same software, the need to create a controlled environment that would follow along the theories to an adequate level of complexity had to be reached. In this study, we collected and visualized participants' movements to obtain information about individual and generalized route selection in the context of the presented VGE.

2. Methods and Technologies

The study examined user behavior in a virtual environment and the psychological concepts explaining participant behavior. Since users were engaged with the environment in great detail, the technological solutions for the experiment were considered a key aspect of this study. The original user and data logging functionalities were implemented into the VGE setting and the entire experiment was conducted based on this.

2.1. Visualization Engine

To make sure the experimental design was effectively constructed, an optimized virtual environment creation workflow was devised, and compatible hardware was found. Combined with sufficient software implementation of real-time data-logging algorithms and a user interface tailored for ease of use, a valid process for obtaining data was established.

Hardware—The experiment was designed for potential use with both Pseudo-3D [57] visualization interfaces (monoscopic devices, e.g., PC monitors) and Real-3D [57] interfaces (stereoscopic devices, e.g., 3D monitors with shutter glasses or head-mounted displays). In our case, participants were outfitted with a head-mounted display, as the research primarily focused on user behavior in a virtual environment with a full VR experience (i.e., not employing multiple visualization devices, because this would have introduced extra experimental variables). All participants used the same kind of device with the same parameters, calibration settings, etc. The participants were given an Oculus Rift DKII head-mounted display (Full HD resolution, 75 frames per second, 110 deg. field of view). The head-mounted display was also outfitted with an aftermarket SMI iView-NG eye-tracker (with 60 Hz binocular sampling frequency), though eye-tracking data were not analyzed in this paper. The experimental scenes were rendered on an Intel i7 5820 K system (6 × 3.3 GHz), outfitted with

an nVidia GeForce GTX 980Ti graphics card—a system powerful enough to prevent any potential framerate drops.

Software—Unity version 5.4.3 was used as the visualization engine and central point of our application development. Unity provides free-to-use resources and customizable plugins that facilitate relatively quick application development. Thanks to the open and well-documented application program interface (API) of Unity, we were able to utilize or tweak some of the functionalities of the engine, but also to implement some others of our own. The API is written in the C# programming language, through which we were able to complement to the utility of the VGE to fit our purposes (e.g., user behavior logging, or teleporting users from one experimental scene to another). Unity offers support for the latest devices, including head-mounted displays. All the virtual environments participants were exposed to were created in Unity. To compose the environments, a combination of freely available 3D objects (Unity Store) was used, along with our own objects created in external applications (e.g., Photoshop textures, Blender 3D models) for each scene with respect to their intended saturation as visual stimuli.

A universal user interaction logging component was implemented and recorded the following input:

- virtual movement (trajectory of movement in virtual space)
- mouse movement (looking around and/or changing direction of movement)
- head movement (looking around using a head-mounted display)
- keyboard interaction (eliciting movement in virtual space)
- participant–object collisions in the virtual world (interaction with features in virtual space)
- (optionally) eye-tracking data (participants' eye fixations on objects)

Virtual movement, mouse movement and head movement were recorded with a frequency of 2 Hz (one record per 500 ms). Keyboard interaction and participant-object collisions were recorded as these events occurred. All participants' interaction data were saved as CSV outputs (Data S1) to allow additional analysis and visualization using statistical software or external visualization applications.

VGE—The software fully facilitated creation of the environment. No external tools (except for those data logging ones created by us) were used, as none were needed. The 3D objects which established the VGE were either free-to-use items obtained from the Unity Asset Store (an official depository of Unity packages) or created by us. The VGE creation process was streamlined: first, following the visual cognition theory, a subject matter (an outdoor, mountain-like environment) was chosen, along with conceptualizing a suggested spatial layout draft (the size of the environment, main routes, and key landmarks). Then, a rough 3D terrain of the experimental space was created. After that, the environment was iterated upon—either by more detailed passes (e.g., large vegetation to small vegetation) or by thematic passes (e.g., when man-made structures were added). Sunlight was introduced into the environment. Finally, any imperfections of the environment were searched for and fixed (visual or performance issues). It was ensured that the visuals of the VGE would run well on the hardware provided and that they would be distributed evenly.

2.2. OSIVQ

Based on previous findings [44], proponents of the object/spatial dichotomy [45] developed the OSIQ, which is a self-assessment questionnaire (Object-Spatial Imagery Questionnaire) consisting of two general scales. The object scale reveals preferences for clear, colorful representations of detailed objects. The spatial scale shows tendencies for representations involving object relationships and spatial transformations. The questionnaire was subject to validity and reliability testing. Results have shown that the questionnaire is reliable in both scales (object scale Cronbach's $\alpha = 0.83$, spatial scale Cronbach's $\alpha = 0.79$). Unlike previous questionnaires in this area, OSIQ is not impaired by low prediction validity—the object scale correlates with performance tests of object imagery; similarly, the spatial scale correlates significantly with performance tests focusing on spatial imagery [45]. As an extension to 2006's OSIQ, Blazhenkova and Kozhevnikov [42] developed the OSIVQ—adding a third,

verbal scale to the object/spatial scales, which remained unchanged. This extended version was used in this study.

3. User Study

This paper presents an exploratory study to demonstrate the functionality of a virtual environment and suggests topics suitable for future research. A correlation analysis focusing on the relationship between self-reported individual cognitive styles and participants' behavioral performance in a virtual environment was conducted as a demonstration of psychological application of the tool. Data collected in the VR experimental scenario (distances walked, pauses, time) were correlated and compared to the information acquired from participants' self-reports (the OSIVQ, participants' sex, and participants' experience with map/PC/VR use). Since this study dealt with an original implementation of a specific VR testing tool, the basic features of an experimental VR environment and variables regarding future experimental designs (e.g., sex or differences in expertise) were also analyzed to demonstrate the tool's potential for assessing individual differences.

The analysis was done in R using RStudio. All graphs were created in RStudio. Spatial visualizations were created in the Processing visualization language.

3.1. Procedure

The experimental tasks were presented after participants were briefed, introduced to and trained in the devices and familiarized with the experiment. In the initial briefing, participants were questioned about any visual disorders that could potentially interfere with testing. They were also asked about their experience in using computers, 3D applications and maps. Regarding ethical standards, participants were informed about the nature of the experiment (spatial cognition) and their right to withdraw from the experiment if they experienced any motion sickness or for any other reason.

The participants were seated in a non-swivel chair facing an LCD screen with a keyboard and a mouse; on top of the LCD screen, a pre-calibrated HMD motion tracking camera was placed. The participants were then instructed on the principles the controller interface worked: mouse movement allowed them to move their whole virtual body on the horizontal axis, effectively changing the direction they were going; in contrast to this, head movement while outfitted with the HMD translated to head-only movement in VR (that is, looking around in both axes, all while not altering movement direction). To move through the VGE, keyboard arrow keys were provided. To get accustomed to the interface prior to the experiment, the users were first introduced to a training environment. (Figure 1b, map B).

To summarize, all participants underwent the same procedure (in the following order):

1. Inquiry into their demographical data and possible health limitations regarding vision and motion sickness.
2. Exploration of a small-scale interactive training environment with a task (Figure 1b, map B).
3. Free exploration of an interactive experimental virtual environment (Figure 1a, map A) and memorization of its features.
4. Observation of another semi-interactive scene (Figure 1c, map C) and answering questions about the spatial layout of the specific components in the scene.
5. Choice of the correct appearance of various objects in the semi-interactive scene (Figure 1c, map C) in a multiple-choice questionnaire.
6. Completion of the OSIVQ questionnaire.



Figure 1. Isometric, elevated overview of map A (**a**), map B (**b**), and map C (**c**).

3.2. Participants

We engaged 25 participants in the experiment. The sample of 25 psychology students was recruited from the Department of Psychology, Faculty of Arts, Masaryk University, where we expected a rather low general experience with head-mounted display (HMD) technologies. We initially excluded two individuals with visual impairments. Three other participants were excluded because of incomplete experimental runs caused by VR-induced motion sickness. The final number of analyzed participants was 20 (10F/10M; aged 20–26, $m = 21.33$, $sd = 1.602$).

3.3. Stimuli and Tasks

For the purpose of this study, a realistic visualization of a rural mountain range was created (Figure 2). This environment deliberately included key visual components according to spatial landmark theories such as trails, districts, environmental borders/edges, key landmark objects [50,51], etc. This included detailed and unique items scattered across the environment. The user interface consisted of head-mounted display, keyboard and mouse. A small, purposefully bland tropical island (Figure 1b, map B) was created for a trial task to familiarize participants with the control devices and VR technology. The overall visual complexity of the trial island was much lower than the one of the experimental environment, as the emphasis in there was on the user interface. To learn the principles of the interface, participants were asked to walk around the island in a first-person perspective and collect five objects (virtual coconuts) scattered across the terrain. Since some of the objects were harder to find than the others, it took the participants about 2–5 min to find them. Individual differences in the trial task can be attributed to participants' visual attention and their initial mastery of the interface. In spite of a brief exposure, this task ensured the participants with no prior VR/3D experience grasped the necessary skills to navigate in a VGE.



Figure 2. An elevated view of the experimental environment (map A) the participants were asked to explore.

After completing the trial task, participants were brought to the experimental mountain range environment (Figure 1a, map A; Figure 2). This environment was much bigger, spanning half a square kilometer. Here, participants were instructed to freely explore the features of the environment. An undisclosed time limit (10 min) applied to exploration (if they exceeded this limit, participants were then asked to finish exploring); this limit was not disclosed to participants in order not to induce any feelings of constraint and to keep the process of exploration natural to reflect the participants' specific visual cognitive styles. As they explored, data was collected on participants' movement patterns, stops/times spent pausing, and use of the interface.

3.3.1. Spatial Relations Identification Task (SRIT)

After participants were notified or once the ten-minute period had expired, participants were transferred to a semi-interactive scenario (Figure 1c, map C), where they were asked to observe a small castle. They were allowed to look around freely, but their spatial movement/locomotion was disabled.

After an initial period of free observation of the castle, participants were asked ten questions concerning the spatial distances in the exposition, for example, the distance from their own position to a specific object in the scene, the width/height of an object or the distance between two other objects (Figure 3 for scene appearance, Appendix A for the questions asked in the task). This task's objective was to mirror the self-reported spatial scale on the OSIVQ, specifically focusing on participants' spatial orientation skills.



Figure 3. An example of spatial relations identification tasks as seen from the participant's perspective; (red line—distance between the chair and the tree stump; blue line—height of the door).

3.3.2. Object Visual Recognition Test (OVRT)

The final task was designed to evaluate participants' visual cognition object-scale abilities in relation to the aforementioned OSIVQ, specifically focusing on participants' spatial orientation skills. At this point, the VR headset was already set aside; this task was executed digitally on a standard 24" PC screen. In a series of full-screen slides, participants were given ten visual discrimination questions. In each of these questions, four cut-out images of a portion of the scene shown in the previous task were presented, one of which depicted an object that was present in the scene. The other three were distractors, i.e., intentionally modified or substituted objects (Figure 4 shows an example item, Appendix B for the entire test).

The participants were asked to report the correct answer for each question and then proceed to the next question by pressing a key. There was no time limit for answers, and participants were allowed to progress through this test on their own. After each keypress, the next question's stimuli appeared with no delay. Once participants had answered the final OVRT question, the testing concluded.

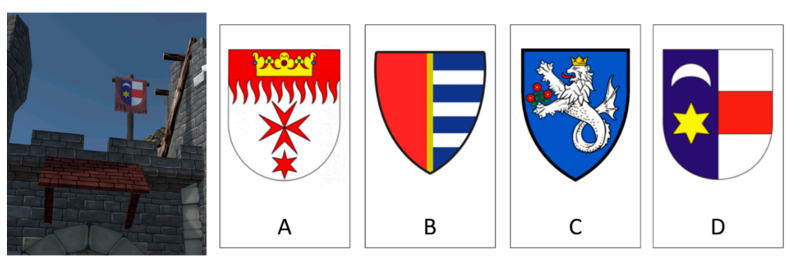


Figure 4. Example of an object visual recognition test item–coat of arms on a flag. A, B, C: distractor objects; D: correct object.

4. Results

4.1. Exploration of the Virtual Environment–Descriptive Data

4.1.1. Overall Required Time

On average, participants took 396 s to explore the mountain range environment. Participants walked a distance of 1180 meters on average. Both of these trends pass the Shapiro–Wilk

test for normal distribution (time, $p = 0.72$, distance, $p = 0.99$), and the experimental environment therefore met the requirements for further parametric analyses. Only three of the 20 participants exceeded 10 min in their exploration.

4.1.2. Visualization of Exploration Strategies Based on User Logs

Some differences in the participants' exploration strategies were observed during user testing. Some walked almost the entire time while exploring; others tended to stop and look around. Corresponding data was measured in our visualization engine (see Section 2.1) and then visualized as virtual trajectories to provide an understanding of participants' exploration approaches (see the comparison of virtual trajectories of two participants employing different exploration strategies in Figure 5 and an overview of all participants in Figure 6a,b). To compute and visualize this data, we used an external script in the Processing programming language (version 3.5) to load virtual movement data collected in the experimental procedure. Figure 6a is a merger of all the individual paths travelled by the participants (each red circle represents a single instance of a participant stopping in a place). Figure 6b also illustrates the use of the Gridded AoI (Areas of Interest) method. Gridded AoIs were created as squares. In each, the number of participant occurrences were determined in order to show a generalized representation of the most prominently visited areas of the environment (to improve comprehensibility, data in the grid cells were relativized). Figure 6a,b show that participants had some prominent route preferences in common; these preferences are further considered in the discussion.

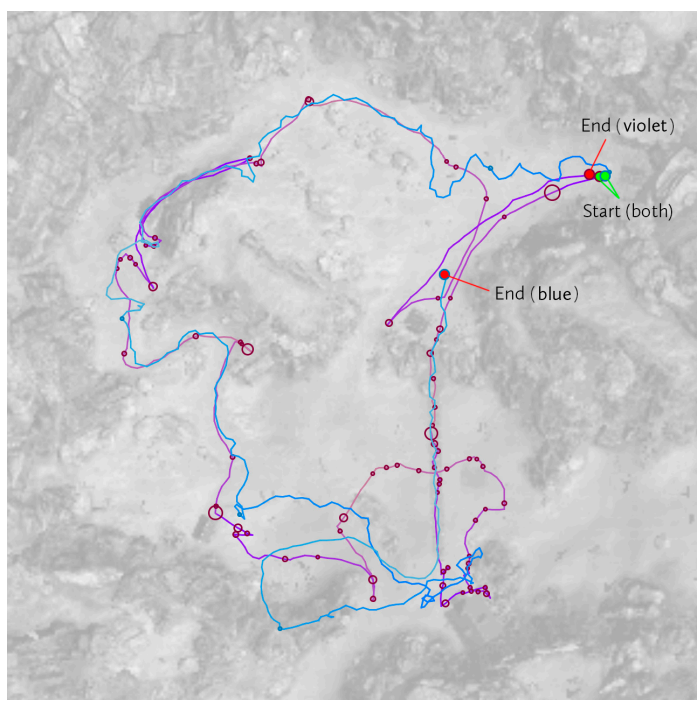
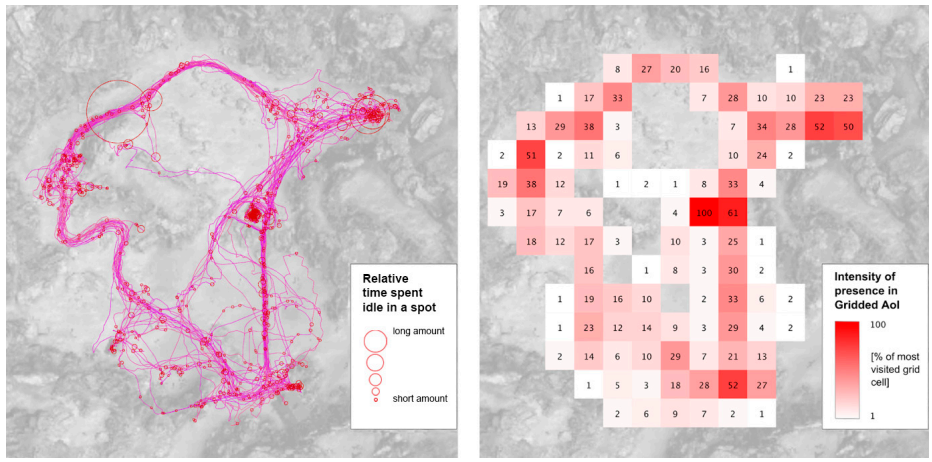


Figure 5. The participant indicated by the blue line walked through the environment with few pauses along their virtual trajectory (path) in a rather jagged fashion (side-stepping left/right, using the keyboard interface frequently). This contrasted with the participant indicated by the violet line, who moved at a slower exploration pace primarily in a forward direction and paused more often (visualized as circles of varying size, larger circles indicating a longer pause).



(**a**) (**b**)

Figure 6. Virtual trajectories (paths) of all individual participants combined (**a**), Gridded AoI average for all paths (**b**).

4.1.3. Statistical Analysis of User Logs and Mental Measurements

Using the experiment's logged data (Data S1), we extracted the following virtual movement parameters:

- exploration time (the total time a participant spent in the environment, in seconds)
- time spent pausing (the amount of time spent not walking, in seconds)
- average exploration speed (walking/pausing time considered, in metres per second)
- travel distance (the overall distance a participant travelled, in metres)
- number of stops (the number of times a participant paused exploration)
- number of keypresses (raw movement interface usage)
- number of direction changes (the number of times a participant changed their keyboard-driven movement direction)

Participants also self-reported their visual cognitive styles (using objects/spatial scales) and were tested in their visual recognition and spatial relations identification abilities. We extracted the following mental scales:

- self-reported OSIVQ object (on a 0–5 scale)
- self-reported OSIVQ spatial (on a 0–5 scale)
- Object Visual Recognition Test (on a 0–10 scale)
- Spatial Relations Identification Task (on a 0–10 scale)

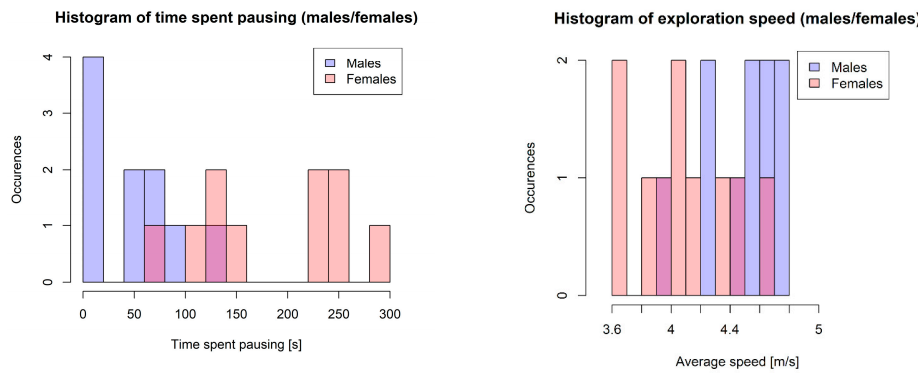
Differences between sexes—We explored the differences between sexes in the extracted virtual movement parameters and mental scales. The Wilcoxon rank-sum test identified differences between males and females in their Exploration Time, Time Spent Pausing, and Exploration Speed (Table 1).

Table 1. Analysis of measured variables according to sex.

	Sex	Mean	Std. Dev.	Median	1st Qu.	3rd Qu.	Shapiro–Wilk Test	*p*-Value	Wilcoxon W	Cohen's d
Exploration Time [s]	M	316.2	141.27	314	216.8	429	0.78	0.03 *	21.5	1.15
	F	475.2	136.03	450.5	371	600.8	0.11			
Time Spent Pausing [s]	M	48.3	41.65	47	11	70	0.4	0.0001 **	4	2.30
	F	184.4	72.76	187.5	124.2	244.5	0.17			
Exploration Speed [m/s]	M	4.47	0.26	4.55	4.31	4.66	0.36	0.012 *	83	1.33
	F	4.09	0.31	4.05	3.91	4.32	0.85			
Direction Changes [Count]	M	377.4	309	316.5	246.2	426.2	0.89	0.019 *	81.5	1.12
	F	111.8	133.29	54	19	173.8	0.8			

Significance levels: * 0.05; ** 0.001.

The average participant stood still for approximately a quarter of the total exploration time (making 37 stops on average, idling for 116 s). The previous analyses of the sample revealed two opposite subsets consisting of "runners" (mostly males) and "lookers" (mostly females)—see Figure 7a,b. The controller's walkthrough speed in the experimental interface was set to a constant 5 m/s. By pausing, men reduced their average walkthrough speed to 4.47 m/s, women to 4.09 m/s. Participants who exceeded the 10-min exploration limit were all females that had spent a considerable time (>4 min) pausing. Males tended to utilize their keyboard movement interface much more often, making about three times more movement direction changes on average compared to females.



(**a**) (**b**)

Figure 7. Histogram comparing males/females in time spent pausing (**a**). Histogram comparing males/females in average exploration speed (**b**).

Differences in VR expertise—We compared participants with respect to their self-reported VR experience. Participants reported themselves as either experienced or not experienced with VR use, and these two groups were compared. Significant differences were identified between non-proficient and proficient VR users in the ability to identify spatial relations in the virtual scene (Table 2).

Table 2. Analysis of measured variables according to VR expertise.

	VR Users	Mean	Std. Dev.	Median	1st Qu.	3rd Qu.	Shapiro–Wilk Test	*p*-Value	Wilcoxon W	Cohen's d
Spatial Relations Identification Task [0–10]	Non-proficient	6.12	0.88	5.95	5.6	6.76	0.85	0.009 **	6	0.01
	Proficient	7.37	0.59	7.37	6.99	7.86	0.6			

Significance levels: ** 0.001.

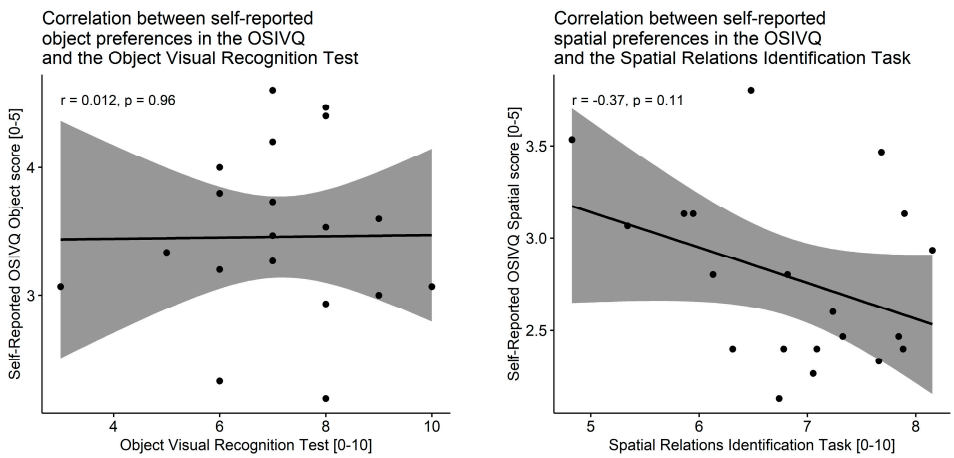
4.2. Correlation Analyses

4.2.1. Self-Reported OSIVQ Object Score and Object Visual Recognition Test

According to the visual cognitive styles theory, in the context of a non-interactive object recognition task, self-reported object visualizers are expected to perform better than self-reported spatial visualizers. However, no statistically significant correlation between the self-reported preferences in the OSIVQ and the participants' scores in the Object Visual Recognition Test were found (Figure 8a, Table 2). No relationship could be observed between a specific self-reported cognitive style and the ability to recognize a specific object that was previously presented in the virtual environment.

4.2.2. Self-Reported OSIVQ Spatial Score and Spatial Relations Identification Task

As in the previous case, spatial visualizers are expected to perform better at estimating distances in a spatial layout. However, the correlation showed a negative, non-significant trend (Figure 8b, Table 2). A non-significant trend was observed in participants who reported themselves as spatially-oriented, performing worse than on specific spatial identification tasks.



(**a**) (**b**)

Figure 8. Correlation between self-reported object preferences in the OSIVQ and the Object Visual Recognition Test (**a**). Correlation between self-reported spatial preferences in the OSIVQ and the Spatial Relations Identification Task (**b**).

In regard to correlating the aforementioned with other continuous variables (Figure 9), only one with any relevance was found: the self-reported spatial score from the OSIVQ correlated positively with exploration speed ($r = 0.48$, $p = 0.03$), i.e., the spatially-oriented participants had a mild tendency to walk through the virtual environment more quickly. A positive correlation between exploration distance and exploration time, and a negative correlation between exploration time and exploration speed are both easily explained, as these variables were heavily dependent on each other. No other correlation approached a statistical significance.

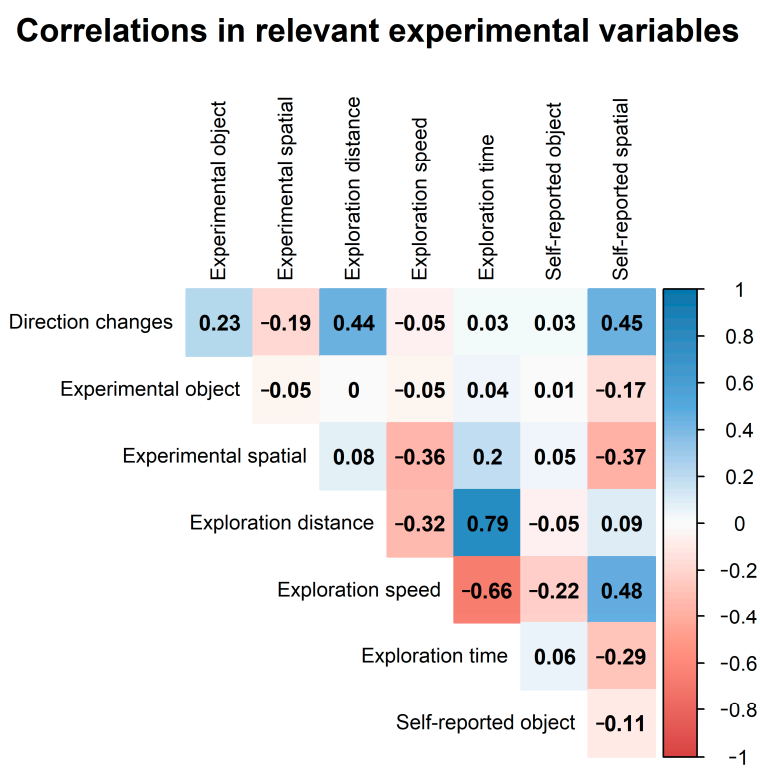


Figure 9. Correlations in relevant experimental variables. Color represents the correlation direction, color intensity is correlation strength, *p*-values are in squares (rounded to two decimals).

5. Discussion and Conclusions

5.1. Methods for Analyzing and Visualizing User Strategies

The virtual movement parameters and visualizations of exploration strategies described in this paper can be categorized or compared according to different criteria. Classification of these methods according to five criteria was suggested (Table 3). This classification can be further used to select the most suitable methods for analyzing the users' strategies in the virtual environment.

The first criterion is preservation of the spatial component of the data. When the spatial component is preserved, the results of the analysis can be applied or visualized to a certain part of the 3D scene (stimulus). Otherwise, the results focus on the non-spatial aspects of user strategies. An example of the results of a non-spatial method are measurements calculated from the virtual trajectory (its distance, average speed).

The second criterion is the attitude to the temporal component of the data. Three ways of handling the temporal aspect of user strategy data were identified. The temporal component can be preserved, and the dynamics of user interactions throughout the entire task can be analyzed (virtual trajectory visualization). Other methods aggregate characteristics to one value for the entire user interaction (Gridded AoI), and some methods also focus on specific moments. That is, for example, the visualization of pauses in virtual movement.

The data aggregation criterion describes whether the result of the method is more suitable to aggregation for all users (or specific groups of users) or if the user interaction records are distinguishable only for certain participants. An example of an aggregating method is Gridded AoI, which does not allow individual users to be identified. A contrasting example is a virtual trajectory visualization

method that is more suitable for studying the virtual movement of individual users or comparing pairs of users.

The fourth criterion describes the potential for numerical or statistical analysis of user-selected aspects of strategies in participants or groups. Some methods are designed specifically for this comparison, or their output may be a numerical comparison of certain characteristics of user interaction or virtual movement (number of pauses in movement, distance of virtual trajectory, average speed). Other methods allow only visual comparison, for example, screenshots (Figure 3), or screen video (see Video S2), which allow visual (qualitative) analysis.

The last criterion is the dimensionality of the analyzed data. In the experiment, it was only possible to walk on terrain (an alternative to a flat plane), so in this case 2D visualizations of virtual trajectories and Gridded AoI could be suitably used. Other virtual worlds can employ free virtual movement in all three dimensions [58]. However, in some special cases this criterion cannot be used, when the method, for example, focuses on non-spatial aspects of user interaction.

The suggested classification can be used to compare the virtual movement parameters and visualizations mentioned in Section 1.2. Other methods that can be used according to the proposed scheme are described by, for example, Chittaro and Ieronutti [33], Zanbaka et al. [34], Cirio et al. [35], Treves et al. [36], Herman and Stachoň [20], and Herman et al. [16,59].

In future studies, researchers can employ methods and types of data visualization other than those demonstrated in this experimental design. Virtual trajectory recording combined with interface interaction data, eye-tracking and possibly think aloud protocol employed during environment exploration could provide detailed insight into the cognitive and behavioral strategies people use for specific tasks. Tracking of interface interactions and eye movements (use of the keyboard and mouse, head movement tracking and eye-tracking) permits effective quantitative assessment of exploration efficiency, both at the level of interindividual differences and the level of evaluating interface and visualization settings. For example, the pilot results of using this method to compare different types of visualization (photorealistic and non-photorealistic) was described by Stachoň et al. [21].

Table 3. Comparison of methods of analysis or visualization of user interaction and virtual movement according to their application.

Method	Preservation of Spatial Component	Attitude to Temporal Component	Suitable for Groups of Users	Statistical or Numerical Comparison	Dimensionality of Data
Gridded AoI	yes	aggregation	yes	yes	2D or 3D
Statistics	no	aggregation or selected moments	yes	yes	2D or 3D
Video	yes	preservation	only individual users	only visual	3D
Views (screenshots)	yes	preservation	only individual users	only visual	3D
Virtual trajectory	yes	preservation	only individual users	only visual	2D or 3D

5.2. Exploration of User Activities and Strategies

Since the experimental group's exploration times and times spent pausing appeared to follow normal distribution, we can assume that the experimental conditions are adequate for collecting data in larger samples. This would only need to be reconsidered if a more spacious/complex virtual environment were employed. The option to visualize the individual paths of participants' walkthroughs as virtual trajectories in different ways (Figure 5; Figure 6a,b, and potentially by other means) also allows us to conduct precise qualitative analyses to understand exploration strategies, while numerical results provide the opportunity to harvest a large amount of data for testing hypotheses with inferential statistics. To visualize our data, we used the Processing programming language. Since this language contains an internal data structure that can load tables (e.g., CSV files), the data could be analyzed rather easily. Based on the collected data, the following visual outputs could be produced: use of the interface by the participant, behavioral patterns and temporal/spatial movement through the virtual space (as a 2D map or an interactive 3D render).

When the paths traveled by all the participants were combined (Figure 6a,b), for the most part participants followed existing pathways/routes present in the environment, only rarely straying off the path. We hypothesize that this behavior may tie to Gibson's affordance theory [60] with consideration to the surrounding environment (e.g., Hadavi et al. [61])—in other words, where a man-made path is meant to be walked, regardless of the path being or not being the optimal route to travel through said environment. Other surroundings and patterns in the terrain may also have predetermined participants' behaviors.

Overall, the participants preferred to visit open and flat areas, even though all areas up to 45 degrees of terrain elevation in the VGE were accessible and the participants did not need to invest any real physical effort to reach them (e.g., climbing a hill—a false affordance [60] in a virtual setting). When the surrounding terrain was constrained or elevated rapidly (height differences, sprawling vegetation, large objects occluding the view), straying from the path was far less frequent. Travelling along the centers of areas may have served the participants as the most efficient strategy in acquiring the most visual information from their surroundings and allowed connections to other potential routes [62]. Similarly, the areas off the path may have provided little to no incentive (relevant information) for exploration. If, however, the accessible surroundings had looked inviting and were furnished with interesting and detailed objects, they may have been inclined to explore the area in order to add these stimuli to their growing spatial knowledge [63].

Follow-up research to verify these route selection hypotheses is required. For this purpose, experimental environments with various (un)interesting areas populated by deliberately pre-placed objects of interest may be created. Such environments may or may not include man-made roads. Logging user route selection in these environments would allow valid behavioral data to be captured. To monitor participant movement across an area, Gridded AoI can be used (as shown in Figure 6b, or by defining some other customized examples). To hypothesize according to these data, algorithms that approximate user behavior can then be created, for example, a 360-degree raycasting visual algorithm (to provide direction-specific attractiveness in an area surrounding a vantage point based on the algorithm targeting visible objects and accessible areas of interest). If multiple 360-degree "cameras" such as these were placed into an environment as de facto route nodes, algorithmic conclusions on route selection can be produced and compared to real user behavior.

Figure 9 shows the overall correlational trends across the variables we measured. It also demonstrates a possible method for further exploration in follow-up studies. In our procedure, no relationship between the participants' self-reported scores on the object/spatial visual scale (OSIVQ) and their objective performance in the VGE was found. No trend in the relationship between individual object-oriented visual abilities and visual recognition measured in the Object Visual Recognition Task (OVRT) was observed (Figure 8a). Contrary to our expectations, a negative non-significant trend was observed between spatial abilities and the Spatial Relations Identification Task (SRIT) (Figure 8b). In this matter, we ought to consider the small sample size of our experiment and suggest that this idea be further explored using larger samples. Reviewing the nature of experimental tasks should be also considered in order to better fulfill the theoretical requirements.

Significant differences between males and females were observed in exploration time, speed and time spent pausing (Table 1). This observation may represent valuable indicators of user tendencies in VGEs and should be further explored in larger samples in respect of not only difference in sex, but also potential customization of the VGE. We also observed that people experienced in VR use were more able to identify features in the virtual reality scene measured by the Spatial Relations Identification Task (Table 2). We can assume that previous experience may possibly decrease the level of distraction in wearing an HMD, have a lower novelty effect, and reduce motion sickness and discomfort.

However, it still needs to be considered that all participant movement happened within the pre-set context of user interface chosen to navigate the environment. Were the participants outfitted with a different kind of an interface (e.g., a motion-tracked controller with teleportation movement style, as opposed to a continuous-walkthrough keyboard and mouse solution), the results may have

told a different story. Since a motion-tracked controller can be considered more of an extension of a human arm, along with a less abstract usability pattern compared to a keyboard, this would, e.g., acc. to Norman's model [64], allow for quicker mastering of the interface by the participants, and thus having to deal with less of an extreme of beginners/experts in user distribution. Not only was it previously shown that interfaces matter [14], but there are studies that are primarily concerned with correctness of fit of VR interfaces [65]. However, this can be disputed, to an extent, since the majority of the population is familiar with the use of computer mouse, and VR interfaces have to deal with the novelty effect. While this study is not primarily focused on user interface, repeating the experiment with different interfaces (and controlling this by the means of a questionnaire) may be a worthy follow-up. However, the act of introducing different controllers would introduce further challenges, as a different concept of movement would also change user movement patterns and the level of interactibility with the environment (effectively needing a distinction in data interpretation and implementation). Furthermore, different interfaces also bring other challenges into research design and methodology—that is, a setup with motion-tracked controllers cannot be deployed to collect data on bigger samples (e.g., in computer classrooms), and it does not offer the same level of experimental control as a keyboard and a mouse do (six degrees of freedom in rotational/translational movements, as opposed to only two).

5.3. Impact of Findings

This paper introduced and described a specific VGE that can be used to measure the behavioral activity of human users in a first-person point-of-view interactive geovisualization. A technological solution was outlined, and potential obstacles were reported. A thorough testing procedure supplemented by and compared to real performance data from the VGE was also designed to reveal individual differences in visual cognitive styles.

The experimental scene appeared natural and was filled with objects of different shape, color and a range of other visual properties. During the estimation of distance, the operation of object boundary differentiation may have not been universally clear (or understood) by participants. To eliminate the potential intervening variables, we therefore suggest re-creating the experiment with more simplistic objects (e.g., a collection of unified gray boxes—a virtual implementation of research from Šikl and Šimeček [66]). The lens of the VR headset may have also introduced some distortion in the perception of distance. The research was exploratory in character, and it may have also become apparent that not all the variables were controlled and analyzed. However, based on our observations, a discussed construct such as "spatial cognitive style" seems to be rather difficult to grasp under a single experimental scenario, since we can further differentiate spatial cognitive style as egocentric and allocentric (i.e., self-centered and scene-centered [67]) design of tasks. In our future research, a larger research sample will be engaged to detect the suggested relationship between cognitive style and observed behavior in VR. Regarding the presented solution and preliminary findings, specific experimental designs for rigorous testing of hypotheses will also be used.

To further address the issue of our small, homogenous sample of participants, follow-up research could check for cross-cultural differences by comparing behavior and exploration strategies of geographically or culturally different groups, such as in Stachoň et al. [68]. Similarly, to alleviate the potential shortcomings of the user interface used in this experiment, a follow-up research employing more humanly intuitive interfaces may be proposed.

Because of the exploratory nature of the study and the low number of participants, we do not draw any general conclusions from this initial research. However, the results suggest the need to assess different levels of VGE realism and to better explore the individual differences between various user groups in order to customize and optimize the VGE. The importance of this suggestion will only grow in the future, especially with respect to the increasing trend of using immersive 3D virtual visualizations in practice.

Regarding the technological solution and the way exploratory design presented in this paper was conducted, positive conclusions can be made. By laying out a behavioral/interaction data logging framework, we have created a basis (a structure in Unity that can be appended to a virtual representation of a participant inside the VGE, along with virtual objects they interact with) from which other possible/future data can be extracted. This also applies to future interfaces—all depending on how open the interfaces are to reading and logging their interaction data. When such data is extracted, data visualization framework, for which we have also laid a foundation, can be extended. Depending on the shape of the VGE in question, the potential question of how an interface is used or the nature of the data that is to be presented, visualizations can be adapted. Since all the technologies on which our solutions are built are easily accessible, free (software) or affordable (hardware), they are easily customizable and/or deployable. Furthermore, even the process of creating VGEs themselves can be streamlined, to an extent, based on the experience gained and described through this paper.

Supplementary Materials: The following materials are available online at http://www.mdpi.com/2076-3417/9/9/1873/s1. Data S1: Experiment data, Video S2: Environment walkthrough.

Author Contributions: Conceptualization, Z.S., P.K., V.J. and Č.Š.; methodology, Č.Š.; software, P.U.; validation, P.U., V.J. and L.H.; formal analysis, P.U.; investigation, P.U.; resources, P.U.; data curation, P.U. and L.H.; writing—original draft preparation, V.J. and L.H.; writing—review and editing, V.J., L.H. and P.U.; visualization, P.U.; supervision, Z.S., P.K., and Č.Š.; project administration, Č.Š.; funding acquisition, Č.Š. and P.K.

Funding: This research was funded by a grant from the Czech Science Foundation agreement No. GC19-09265J called "The influence of socio-cultural factors and writing systems on perception and cognition of complex visual stimuli", and Pavel Ugwitz was also supported by a grant of the Masaryk University No. MUNI/A/1576/2018 called "Complex research of the geographical environment of the planet Earth".

Acknowledgments: This work was supported by the research infrastructure HUME Lab Experimental Humanities Laboratory, Faculty of Arts, Masaryk University. Also, our thanks go to all participants, proof readers and reviewers.

Conflicts of Interest: The authors declare no conflict of interest in relation to this study.

Appendix A Spatial Relations Identification Task

In this VR task (Figure 1c, map C), participants were asked to verbally answer ten questions regarding spatial relations dealing with vertical or horizontal distances, derived from both egocentric and allocentric frames of reference. The questions were as follows:

1. How tall is the ladder in the background?
2. How tall is the tree stump?
3. How tall is the door?
4. How tall is the other ladder?
5. How far away from you is the chair?
6. How far away is the doorway?
7. What is the distance between the chair and the flag with the skull?
8. How far away is the chair from the tree stump?
9. How high up from the ground are the two flags on the battlements?
10. (This is the second value for the previous question, which asks for two values)

Appendix B Object Visual Recognition Test

This test (Figure A1) followed the Spatial Relations Identification Task and had one main difference: the items were presented as a static slideshow on an LCD monitor (not in VR). Each task was presented on a separate screen, with four object variants (single correct answer, three distractors).

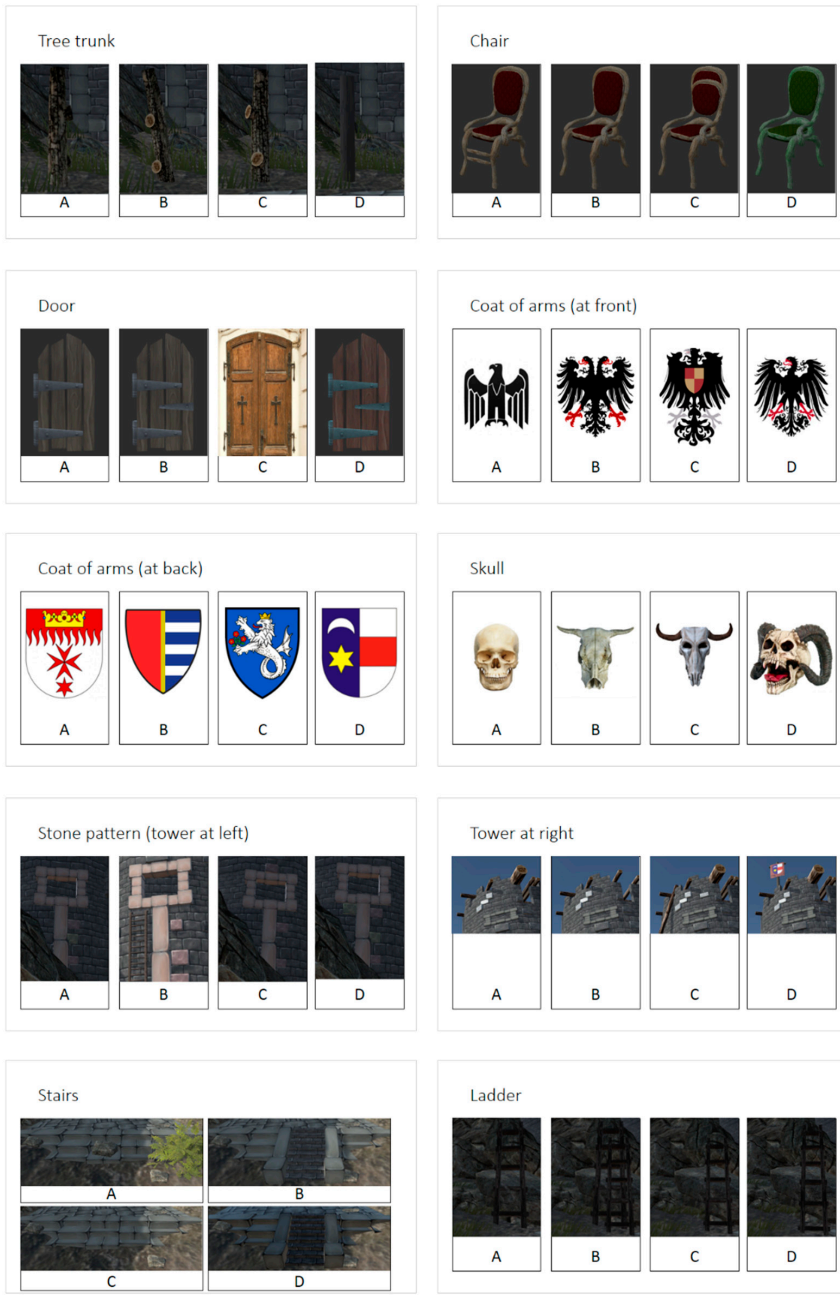


Figure A1. The ten sets of items for the Object Visual Recognition Test.

References

1. Robertson, A.; Zelenko, M. Voices from a Virtual Past. An Oral History of a Technology Whose Time Has Come Again. Available online: http://www.theverge.com/a/virtual-reality (accessed on 1 October 2018).
2. Lin, H.; Gong, J. Exploring Virtual Geographic Environments. *Geogr. Inf. Sci.* **2001**, *7*, 1–7. [CrossRef]

3. Lin, H.; Chen, M.; Lu, G.; Zhu, Q.; Gong, J.; You, X.; Wen, Y.; Xu, B.; Hu, M. Virtual Geographic Environments (VGEs): A New Generation of Geographic Analysis Tool. *Earth Sci. Rev.* **2013**, *126*, 74–84. [CrossRef]
4. Voinov, A.; Çöltekin, A.; Chen, M.; Beydoun, G. Virtual geographic environments in socio-environmental modeling: A fancy distraction or a key to communication? *Int. J. Digit. Earth* **2017**, *11*, 408–419. [CrossRef]
5. Konečný, M. Cartography: Challenges and Potentials in Virtual Geographic Environments Era. *Ann. GIS* **2011**, *17*, 135–146. [CrossRef]
6. Lin, H.; Batty, M.; Jørgensen, S.E.; Fu, B.; Konečný, M.; Voinov, A.; Torrens, P.; Lu, G.; Zhu, A.X.; Wilson, J.P.; Gong, J. Virtual Environments Begin to Embrace Process-based Geographic Analysis. *Trans. GIS* **2015**, *19*, 493–498. [CrossRef]
7. Thompson, E.M.; Horne, M.; Fleming, D. Virtual Reality Urban Modeling—An Overview. In Proceedings of the International Conference on Construction Applications of Virtual Reality, Orlando, FL, USA, 3–4 August 2006.
8. Jamei, E.; Mortimer, M.; Seyedmahmoudian, M.; Horan, B.; Stojcevski, A. Investigating the Role of Virtual Reality in Planning for Sustainable Smart Cities. *Sustainability* **2017**, *9*, 2006. [CrossRef]
9. Moreno, A.; Segura, Á.; Zlatanova, S.; Posada, J.; García-Alonso, A. Benefit of the integration of semantic 3D models in a fire-fighting VR simulator. *Appl. Geomat.* **2012**, *4*, 143–153. [CrossRef]
10. Havenith, H.B.; Cerfontaine, P.; Mreyen, A.S. How Virtual Reality Can help Visualise and Assess Geohazards. *Int. J. Digit. Earth* **2017**, *12*, 173–189. [CrossRef]
11. Maggi, S.; Fabrikant, S.I.; Imbert, J.P.; Hurter, C. How do Display Design and User Characteristics Matter in Animations? An Empirical Study with Air Traffific Control Displays. *Cartographica* **2015**, *51*, 25–37. [CrossRef]
12. Slater, M.; Usoh, M. Presence in immersive virtual environments. In Proceedings of the IEEE Conference—Virtual Reality Annual International Symposium, San Jose, CA, USA, 18–22 September 1993.
13. Tamasi, T. The Evolution of Computer Graphics. In Proceedings of the NVISION 08, San Jose, CA, USA, 25–28 August 2008.
14. Juřík, V.; Herman, L.; Šašinka, Č.; Stachoň, Z.; Chmelík, J. When the Display Matters: A Multifaceted Perspective on 3D Geovisualizations. *Open Geosci.* **2017**, *9*, 89–100. [CrossRef]
15. Zhang, F.; Hu, M.; Che, W.; Lin, H.; Fang, C. Framework for Virtual Cognitive Experiment in Virtual Geographic Environments. *ISPRS Int. J. Geo Inf.* **2018**, *7*, 36. [CrossRef]
16. Herman, L.; Řezník, T.; Stachoň, Z.; Russnák, J. The Design and Testing of 3DmoveR: An Experimental Tool for Usability Studies of Interactive 3D Maps. *Cartogr. Perspect.* **2018**, *9*, 31–63. [CrossRef]
17. Kubíček, P.; Šašinka, Č.; Stachoň, Z.; Herman, L.; Juřík, V.; Urbánek, T.; Chmelík, J. Identification of altitude profiles in 3D geovisualizations: The role of interaction and spatial abilities. *Int. J. Digit. Earth* **2019**, *12*, 156–172. [CrossRef]
18. Juřík, V.; Herman, L.; Kubíček, P.; Stachoň, Z.; Šašinka, Č. Cognitive Aspects of Collaboration in 3D Virtual Environments. In *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*; ISPRS: Leopoldshöhe, Germany, 2016; Volume XLI-B2, pp. 663–670. [CrossRef]
19. Špriňarová, K.; Juřík, V.; Šašinka, Č.; Herman, L.; Štěrba, Z.; Stachoň, Z.; Chmelík, J.; Kozlíková, B. Human-computer Interaction in Real 3D and Pseudo-3D Cartographic Visualization: A Comparative Study. In *Cartography-Maps Connecting the World*; Springer: Cham, Switzerland, 2015; pp. 59–73. [CrossRef]
20. Herman, L.; Stachoň, Z. Comparison of User Performance with Interactive and Static 3D Visualization —Pilot Study. In *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*; ISPRS: Leopoldshöhe, Germany, 2016; Volume XLI-B2, pp. 655–661. [CrossRef]
21. Stachoň, Z.; Kubíček, P.; Málek, F.; Krejčí, M.; Herman, L. The Role of Hue and Realism in Virtual Reality. In *7th International Conference on Cartography and GIS*; Bandrova, T., Konečný, M., Eds.; Bulgarian Cartographic Association: Sofia, Bulgaria, 2018; Volume 2, pp. 932–941.
22. Plesa, M.A.; Cartwright, W. Evaluating the Effectiveness of Non-Realistic 3D Maps for Navigation with Mobile Devices. In *Lecture Notes in Geoinformation and Cartography Map-Based Mobile Services*; Meng, L., Ed.; Springer: Berlin/Heidelberg, Germany, 2008; pp. 80–104. [CrossRef]
23. Lokka, I.E.; Çöltekin, A.; Wiener, J.; Fabrikant, S.I.; Röcke, C. Virtual environments as memory training devices in navigational tasks for older adults. *Sci. Rep.* **2018**, *8*. [CrossRef] [PubMed]

24. Popelka, S.; Dědková, P. Extinct Village 3D visualization and its Evaluation with Eye-Movement Recording. In *Computational Science and Its Applications*; Springer: Cham, Switzerland, 2014; pp. 786–795. [CrossRef]
25. Zanola, S.; Fabrikant, S.I.; Coltekin, A. The Effect of Realism on the Confidence in Spatial Data Quality in Stereoscopic 3D Displays. In Proceedings of the 24th International Cartography Conference, Santiago, Chile, 15–21 November 2009. [CrossRef]
26. Gregory, R.J. *Psychological Testing: History, Principles, and Applications*; Pearson: Boston, MA, USA, 2016.
27. Manson, S.; Kne, L.; Dyke, K.; Shannon, J.; Eria, S. Using Eye-tracking and Mouse Metrics to Test Usability of Web Mapping Navigation. *Cartogr. Geogr. Inf. Sci.* **2012**, *39*, 48–60. [CrossRef]
28. Wilkening, J.; Fabrikant, S.I. How Users Interact with a 3D Geo-browser under Time Pressure. *Cartogr. Geogr. Inf. Sci.* **2013**, *40*, 40–52. [CrossRef]
29. You, M.; Chen, C.W.; Liu, H.; Lin, H. A Usability Evaluation of web Map Zoom and Pan Functions. *Int. J. Des.* **2007**, *1*, 15–25.
30. Glawischnig, S.; Mahdavi, A. Human Interface Devices and Building Information Systems—A Usability Study. *IADIS Int. J.* **2013**, *11*, 29–142.
31. Roth, R.E. Cartographic Interaction Primitives: Framework and Synthesis. *Cartogr. J.* **2012**, *49*, 376–395. [CrossRef]
32. Bowman, D.A.; Kruijff, E.; LaViola, J.J.; Poupyrev, I. *3D User Interfaces: Theory and Practice*; Addison Wesley Longman Publishing: Redwood City, CA, USA, 2005; p. 512.
33. Chittaro, L.; Ieronutti, L. A Visual Tool for Tracing Users' Behavior in Virtual Environments. In Proceedings of the Working Conference on Advanced Visual Interfaces, Gallipoli, Italy, 25–28 May 2004; pp. 41–47.
34. Zanbaka, C.A.; Lok, B.C.; Babu, S.V.; Ulinsky, A.C.; Hodges, L.F. Comparison of Path Visualizations and Cognitive Measures Relative to Travel Technique in a Virtual Environment. *IEEE Trans. Vis. Comput. Graph.* **2005**, *11*, 694–705. [CrossRef]
35. Cirio, G.; Marchal, M.; Olivier, A.H.; Pettré, J. Kinematic Evaluation of Virtual Walking Trajectories. *IEEE Trans. Vis. Comput. Graph.* **2013**, *19*, 671–680. [CrossRef]
36. Treves, R.; Viterbo, P.; Haklay, M. Footprints in the sky: Using student tracklogs from a "bird's eye view" virtual field trip to enhance learning. *J. Geogr. High. Educ.* **2015**, *39*, 97–110. [CrossRef]
37. Herman, L.; Stachoň, Z. Controlling 3D Geovisualizations on Touch Screen—The Role of Users Age and Gestures Intuitiveness. In *7th International Conference on Cartography and GIS*; Bandrova, T., Konečný, M., Eds.; Bulgarian Cartographic Association: Sofia, Bulgaria, 2018; Volume 1, pp. 473–480.
38. Paivio, A.; Csapo, K. Short-term sequential memory for pictures and words. *Psychon. Sci.* **1971**, *24*, 50–51. [CrossRef]
39. Ausburn, L.J.; Ausburn, F.B. Cognitive styles: Some information and implications for instructional design. *Educ. Technol. Res. Dev.* **1978**, *26*, 337–354.
40. Martinsen, O.; Kaufmann, G. Cognitive style and creativity. In *Encyclopedia of Creativity*; Runco, M.A., Pritsker, S.R., Eds.; Academic Press: New York, NY, USA, 1999; Volume 1, pp. 273–282.
41. Kirton, M.J. *Adaption-Innovation: In the Context of Diversity and Change*; Routledge: New York, NY, USA, 2008.
42. Blazhenkova, O.; Kozhevnikov, M. The New Object-Spatial-Verbal Cognitive Style Model: Theory and Measurement. *Appl. Cogn. Psychol.* **2009**, *23*, 638–663. [CrossRef]
43. Peterson, E.R.; Deary, I.J.; Austin, E.J. A New Measure of Verbal–Imagery Cognitive Style: VICS. *Personal. Individ. Differ.* **2005**, *38*, 1269–1281. [CrossRef]
44. Kozhevnikov, M.; Kosslyn, S.; Shephard, J. Spatial versus object visualizers: A new characterization of visual cognitive style. *Mem. Cogn.* **2005**, *33*, 710–726. [CrossRef]
45. Blajenkova, O.; Kozhevnikov, M.; Motes, M.A. Object-Spatial Imagery: A New Self-Report Imagery Questionnaire. *Appl. Cogn. Psychol.* **2006**, *20*, 239–263. [CrossRef]
46. Jonassen, D.H.; Grabowski, B.L. *Handbook of Individual Differences, Learning, and Instruction*; Routledge: Abingdon, UK, 2012.
47. Golledge, R.G. Human Wayfinding and Cognitive Maps. In *The Colonization of Unfamiliar Landscapes: The Archaeology of Adaptation*; Routledge: New York, NY, USA, 2003; pp. 25–43.
48. Schiller, D.; Eichenbaum, H.; Buffalo, E.A.; Davachi, L.; Foster, D.J.; Leutgeb, S.; Ranganath, C. Memory and Space: Towards an Understanding of the Cognitive Map. *J. Neurosci.* **2015**, *35*, 13904–13911. [CrossRef]

49. Epstein, R.A.; Patai, E.Z.; Julian, J.B.; Spiers, H.J. The cognitive map in humans: Spatial navigation and beyond. *Nat. Neurosci.* **2017**, *20*, 1504–1513. [CrossRef]
50. Lynch, K. *The Image of the City*; MIT Press: Cambridge, MA, USA, 1960.
51. Gartner, G.; Huang, H. *Progress in Location-Based Services 2016*; Lecture Notes in Geoinformation and Cartography; Springer: Berlin, Germany, 2017.
52. Franz, M.O.; Mallot, H. Biomimetic robot navigation. *Robot. Auton. Syst.* **2000**, *30*, 133–153. [CrossRef]
53. Golledge, R.G. Path selection and route preference in human navigation: A progress report. In *Spatial Information Theory A Theoretical Basis for GIS. Lecture Notes in Computer Science*; Frank, A.U., Kuhn, W., Eds.; Springer: Berlin, Germany, 1995. [CrossRef]
54. Vilar, E.; Rebelo, F.; Noriega, P.; Teles, J.; Mayhorn, C. The influence of environmental features on route selection in an emergency situation. *Appl. Ergon.* **2013**, *44*, 618–627. [CrossRef]
55. Duckham, M.; Kulik, L. "Simplest" Paths: Automated Route Selection for Navigation. In Proceedings of the International Conference on Spatial Information Theory, Ittingen, Switzerland, 24–28 September 2003.
56. Collett, T.S.; Zeil, J. The selection and use of landmarks by insects. In *Orientation and Communication in Arthropods*; Birkhäuser: Basel, Switzerland, 1997; p. 84. [CrossRef]
57. Nys, M.; Gyselinck, V.; Orriols, E.; Hickmann, M. Landmark and route knowledge in children's spatial representation of a virtual environment. *Front. Psychol.* **2015**, *5*, 1522. [CrossRef]
58. Buchroithner, M.F.; Knust, C. True-3D in cartography—Current hard and softcopy developments. In *Geospatial Visualisation*; Moore, A., Drecki, I., Eds.; Springer: Heidelberg, Germany, 2013; pp. 41–65.
59. Herman, L.; Stachoň, Z.; Stuchlík, R.; Hladík, J.; Kubíček, P. Touch Interaction with 3D Geographical Visualization on Web: Selected Technological and User Issues. In *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*; Dimopoulou, E., van Oosterom, P., Eds.; ISPRS: Leopoldshöhe, Germany, 2016; Volume XLII-2/W2, pp. 33–40. [CrossRef]
60. Gibson, J.J. *The Senses Considered as Perceptual Systems*; Allen and Unwin: London, UK, 1966.
61. Hadavi, S.; Kaplan, R.; Hunter, M.C.R. Environmental affordances: A practical approach for design of nearby outdoor settings in urban residential areas. *Landsc. Urban Plan.* **2014**, *134*, 19–32. [CrossRef]
62. Wiener, J.M.; Schnee, A.; Mallot, H.A. Use and interaction of navigation strategies in regionalized environments. *J. Environ. Psychol.* **2004**, *4*, 475–493. [CrossRef]
63. Meilinger, T. *Strategies of Orinetation in Environmental Spaces*; Logos Verlag: Berlin, Germany, 2008; pp. 72–80.
64. Sutcliffe, A.G.; Kaur, K.D. Evaluating the usability of virtual reality user interfaces. *Behav. Inf. Technol.* **2000**, *19*, 415–426. [CrossRef]
65. Nguyen-Vo, T.; Riecke, B.E.; Stuerzlinger, W.; Pham, D.M.; Kruijff, E. Do We Need Actual Walking in VR? Learning with Actual Rotation Might Suffice for Efficient Locomotion. In Proceedings of the 11th International Conference on Spatial Cognition, Tübingen, Germany, 5–8 September 2018.
66. Šikl, R.; Šimeček, M. Visual space perception at different levels of depth description. *Atten. Percept. Psychophys.* **2015**, *77*, 2098–2107. [CrossRef]
67. Kozhevnikov, M.; Hegarty, M. A dissociation between object manipulation spatial ability and spatial orientation ability. *Mem. Cogn.* **2001**, *29*, 745–756. [CrossRef]
68. Stachoň, Z.; Šašinka, Č.; Čeněk, J.; Štěrba, Z.; Angsuesser, S.; Fabrikant, S.I.; Štampach, R.; Morong, K. Cross-cultural differences in figure–ground perception of cartographic stimuli. *Cartogr. Geogr. Inf. Sci.* **2018**, *46*, 82–94. [CrossRef]



© 2019 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).

Annex 3: Paper III

Ugwitz, P., Šašinková, A., Šašinka, Č., Stachoň, Z., Juřík, V. (2021). Toggle Toolkit: A Tool for Conducting Experiments in Unity Virtual Environments. *Behavior Research Methods*. https://doi.org/10.3758/s13428-020-01510-4

Behavior Research Methods (2021) 53:1581–1591
https://doi.org/10.3758/s13428-020-01510-4

Toggle toolkit: A tool for conducting experiments in unity virtual environments

Pavel Ugwitz[1,2] · Alžběta Šašinková[2] · Čeněk Šašinka[2] · Zdeněk Stachoň[1,2] · Vojtěch Juřík[2,3]

Accepted: 9 November 2020 / Published online: 6 January 2021
© The Psychonomic Society, Inc. 2021

Abstract
This article presents and offers Toggle Toolkit, which is an original collection of Unity scripts designed to control various aspects of interactive 3D experiments. The toolkit enables researchers in different fields to design, conduct and evaluate experiments and include interactive elements in immersive virtual environments. This was achieved by using the internal functionalities of the Unity engine and solutions of our own design. The structure of Toggle Toolkit allows triggers and toggles to be allocated to existing virtual objects and throughout the Unity scene. Once a trigger is executed (with a pre-described action, such as colliding with a virtual object, pressing a key, gazing at an object, etc.), the toggles associated with the trigger are activated and then change the attributes or behaviors of linked objects. All interactive behavior is logged and made available for further statistical analysis. Examples of applications in research are presented and discussed. The Toggle Toolkit's utility lies in its simplicity and modularity. The Toolkit was especially produced for experimenters with few coding skills and high customization requirements in their experiments. The tool is freely available for use in research and can be enhanced with custom scripts. A video tutorial is provided to facilitate use of the tool. The paper aims to not only introduce beginners to experimentation with VR but also offers more experienced researchers who are potentially interested in using and adjusting the features the Toolkit a deeper insight into its structure.

Keywords Virtual environment · Virtual reality · Interactivity · Dynamic visualization · 3D visualization · Behavioral analysis · Experiment design · Event logger

✉ Pavel Ugwitz
ugwitz.pavel@mail.muni.cz

✉ Alžběta Šašinková
st.betty@mail.muni.cz

Čeněk Šašinka
cenek.sasinka@mail.muni.cz

Zdeněk Stachoň
zstachon@geogr.muni.cz

Vojtěch Juřík
jurik.vojtech@gmail.com

[1] Department of Geography, Faculty of Science, Masaryk University, Kotlářská 267/2, 611 37 Brno, Czech Republic

[2] Department of Information and Library Studies, Faculty of Arts, Masaryk University, Arna Nováka 1/1, 602 00 Brno, Czech Republic

[3] Centre for Experimental Psychology and Cognitive Sciences, Department of Psychology, Faculty of Arts, Masaryk University, Arna Nováka 1/1, 602 00 Brno, Czech Republic

Introduction

The rapid development of technologies for virtual reality (VR), especially visual head-mounted displays, computer graphics, and general computing power (Singer, 2013; Cuervo et al., 2018), have paved the way for the creation of dynamic, controllable, replicable, customizable and complex stimuli that offer experiences which resemble real phenomena.

It is not a new phenomenon in science that experiments shift to the world of technology. In 1D experiments, many instruments have been successfully introduced and applied (e.g., Šašinka, Morong & Stachoň, 2017; NeuroBS, 2017; Cornelissen, Peters & Palmer, 2002; Paradigm, 2017; Mathôt, Schreij & Theeuwes, 2012; etc.).

VR technologies have been suggested as stimulating brain mechanisms in a similar manner to real-world experiences (Alcañiz et al., 2009). The opportunity to explore human behavior and cognition in realistic, yet well-controlled experimental settings produced with VR represents an important improvement in research over the natural trade-off in science

between experimental control and ecological validity, which has so far been difficult to overcome (Loomis et al., 1999).

So far, the necessity to possess programming skills has prevented many people from designing and conducting experiments in virtual environments unless they are members of profoundly transdisciplinary teams. This difficulty could be partly offset by deploying a visual scripting tool (Sewell, 2015; D., 2015), although if no scripts are available to base simplified scripting on, no solution is offered. Toggle Toolkit was developed for the needs of our own interdisciplinary research team in applying the Unity editor's user interface (Unity Inspector) to simplify experimental design, data collection and subsequent data analysis. Toggle Toolkit enables virtual users to interact with a virtual environment and the environment to react to a user. The toolkit also aids in tracking the behavior of virtual users, including their locomotion, gestures, manipulation of virtual objects, or even eye movements. Examples of the toolkit's use in producing a VR experiment are described in Chapter 3, Experimental Utility. To understand the workflow, see the linked tutorial video, which presents a simple example of an experiment managed using Toggle Toolkit: https://drive.google.com/file/d/1cMfmbaKH1pQfmNA3IW8BCSyhrXLTB-E9

Toggle Toolkit is based on a trigger-event concept extended by nested objects and states. The software design patterns that the kit is based on (command pattern, state pattern; Baron, 2019) have been applied in many 3D games and applications over the past 25 years. These functionalities, however, have not always been easily accessible to users because of limited functionality, previous-generation 3D engine architecture (e.g., Unreal Engine 2/3, Source, id Tech engines, and older), in-engine editors, the inability to customize closed-source solutions, or toolkits that did not match the needs of creating an experiment since they were only game-oriented.

Naturally, Toggle Toolkit is not the first ever software produced to enhance the process of designing experiments in VR. However, none of the other tools available to date are equivalent to Toggle Toolkit. EVE (Experiments in Virtual Environments) (Grübel et al., 2017) is a valuable tool which responds to similar research needs but approaches the problem from a different perspective. EVE differs from Toggle Toolkit in its range of functionality, its architecture and method of use. EVE includes inherent interaction, logging and evaluation scripts, and has utilities for physiological measurements, questionnaires and movement tracking. However, the tool's actual interactivity, as far as we understand from its GitHub repository[1], consists solely in concealing objects and displaying text. It is a type of "superstructure" laid over a finished and compiled Unity project, unlike Toggle Toolkit, which is applied during the process of creating a Unity project. Toggle Toolkit also contains a greater focus on detail in its modularity: the tool enables dependency chains to be created between individual objects and non-trivial sequences of events. Unlike in EVE, every Unity object can become an interactive factor, and the variety of states any object can attain is limited only by its features. The EVE environment is likely easier to use for a common researcher (e.g., it does not require knowledge of Unity editor), however its adjustability compared to Toggle Toolkit is limited, which makes it less able to meet higher experimental design requirements.

Besides EVE, there are some partially analogous tools that possess similar functionalities but differ in other aspects. The reference *Unity 2D Game Kit* ("2D Game Kit" 2017) has its own interaction system for in-game objects that react to players. This game-centric solution is only partially customizable, but its trigger-event functionality follows the same principles as Toggle Toolkit. Although focusing primarily on VR controllers, VR UI, and small-scale object manipulation, the *Unity XR Interaction Toolkit*(Fuad, 2019) has some similarity in being able to alter virtual objects and the surrounding environment. The Unity Experiment Framework (Brookes et al., 2019) provides a comprehensive structure for behavioral experiments in virtual settings; direct access to Unity in-engine functionalities, however, is intentionally not offered and must be programmed manually. The Virtual Reality Experiments (VREX) Toolbox (Vasser et al., 2017) allows the creation of basic experiments situated in interior spaces that consist of a square grid of interconnected rooms generated in the researcher's user interface and detailed procedurally. While this is a useful function for some experiments, it is unnecessary to limit ourselves to this alone.

We chose to remain with Unity engine in our research, as Unity is relatively easy in programming functionality and has an existing support infrastructure (forums, tutorials, videos, documentation), component system (Unity Manual: Components, 2020), , accessibility through the generic C# language and custom C# extensions (Unity Scripting Reference, 2020), extension packages, versatility for use in different fields, and developer maintenance (as opposed to purely research-oriented 3D engines such as Vizard (Vizard, 2020) or older in-house implementations such as Vrecko (Kovalčík et al., 2012)).

Concerning published studies situated in virtual environments, use of multi-modal, complex dynamic implementations for research is currently highly unsatisfactory. Basic research is more prevalent, with designs that frequently rely on fully static, non-interactive environments or that merely implement a single function (e.g., user teleportation (Moghadam et al., 2018), object collection (Grübel et al., 2017), object manipulation (Caputo et al., 2018), movement measurements (Diersch, Wolbers, 2019), and animated passive viewing (Parong, Mayer, 2018)). In older experiments with closed-source 3D engines, the technical aspects of experimental procedures were not mentioned (Smith & Trenholme, 2009; Gould et al., 2007). Assuming no automated in-engine interactivity or measuring

[1] https://github.com/cog-ethz/EVE/tree/master/Assets/EVE/Scripts

took place in these studies, the analysis may have been transferred to a human researcher or post hoc protocol.

Toggle Toolkit requires few coding skills and offers a comprehensive solution that can be adjusted to suit the needs of various experimental designs and various technologies used (e.g. HMD, CAVE etc.). Compared to other similar tools and because of its modularity and focus on detail, Toggle Toolkit allows elements of very high interactivity and enhances a great degree of control in an experiment. The software is freely available and supplemented by an instructional video to provide information to less experienced VR researchers. The toolkit can significantly improve the effectiveness of an experiment's design process since it is based on the real needs of researchers.

The following chapter is especially interesting to researchers who have more advanced coding skills and whose aim is to gain insight into the toolkit's architecture and options for further development. Readers less interested in how the toolkit precisely functions may find the first section of this chapter sufficient and may skim through the Trigger and Toggle Script Module paragraphs to explore the figures and their descriptions or skip them entirely. Readers may then proceed to the Experiment Utilities section, which presents concrete examples of the toolkit's application in research.

Software description

Toggle Toolkit's main function is in predefining controlled changes encountered in a virtual environment through real-time 3D visualization experiments, i.e., altering the graphics and functionality and enabling dynamic visualizations and interactive elements. That is achieved by creating links between the Trigger and Toggle scripts included in the toolkit (for the principle of the toolkit's architecture, see Fig. 1). A stimulus activates a trigger script. This trigger script is linked to a toggle script appended to an object in the VR scene. Activating a trigger causes a linked toggle to change the state of its respective object. The change in the target object due to its associated toggle can be either instantaneous (e.g., a change of color) or continuous (e.g., an animation).

The following scenario (as depicted in Figs. 2 and 3) represents an example case of the aforementioned concepts: a user situated in a hotel backyard is instructed to enter the hotel. For clarity, the design process of this example is presented in the tutorial video mentioned above (https://drive.google.com/file/d/1cMfmbaKH1pQfmNA3IW8BCSyhrXLTB-E9). The hotel door (i.e., object to be affected) is surrounded by an invisible collider (highlighted in Figs. 2 and 3 by a thin green line). Once the user touches the collider (i.e., activates the trigger), the hotel door opens (i.e., the game object toggles itself from *closed* to *open*). When the user leaves the area marked by the collider, the trigger is once again activated and the door returns to its original state (i.e., closes). This example illustrates a simple concept using Toggle Toolkit as an "on/off switch", where the game object has only two predefined states and repeated trigger activation leads to the object returning to its original state.

Toggle Toolkit components can be added to the objects or native components of a Unity scene and their values can be set up using the Unity editor through the Unity inspector GUI. Toggle Toolkit can thus access and alter Unity object attributes; while object attribute manipulation is native to the Unity editor inspector interface and Unity scripting, the tool permits unified, streamlined, continuous changes from a single interface, which is the toolkit's chief utility. When a scene setup and its desired Toggle Toolkit functionalities are finalized, the scene can be compiled and deployed for experimental purposes as a standalone application.

The toolkit was designed as a modular system: some of its components can reference other components, and the spectrum of application is broad. If additional components are

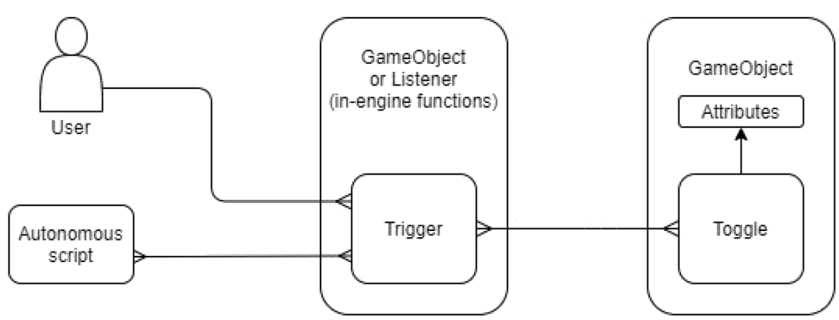


Fig. 1 Basic overview of Toggle Toolkit system architecture. *Note.* Usually, an external entity (a user or script) activates the trigger script(s), which, by extension, activates the toggle script(s) on associated object(s). When toggled, the object(s) change(s) their attributes, feeding back to the dynamics of the scene. This scheme implies that multiple objects can be interconnected, sometimes with dependencies or parent–object relationships. For further explanation of advanced concepts, see Chapter 2.2.2

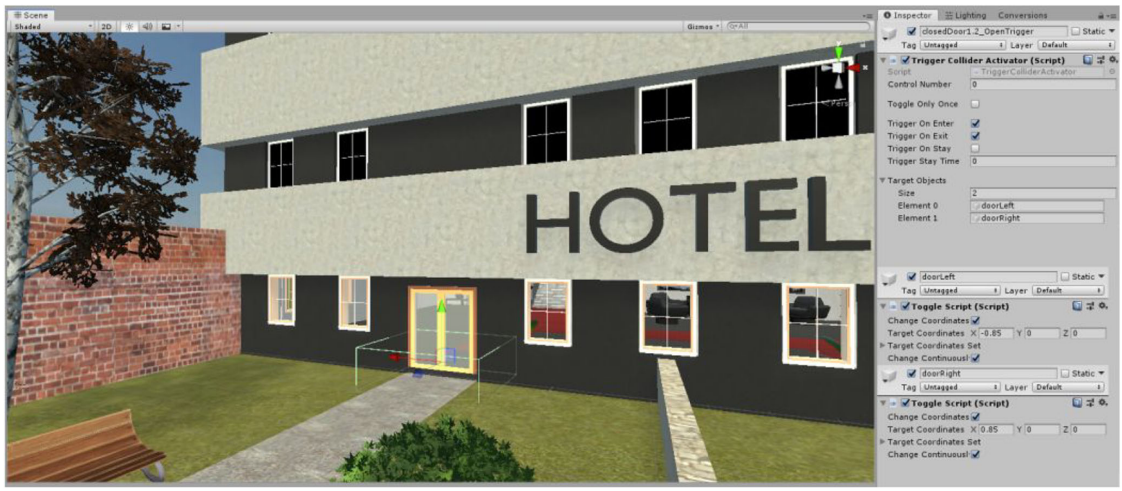


Fig. 2 Example of setting up a simple two-state (on/off) trigger. *Note.* A collider trigger is attached to the hotel door object; the trigger is set up to activate when an external entity (the user) collides with it and trigger again when the user no longer collides with it. The GameObjects associated with this trigger are the two door wings, and their toggle behavior moves the door wings 85 cm to the left and right, respectively. This behavior ensures the door opens in front of the user and then closes behind as soon as the user leaves the defined vicinity. To see the user's perspective, see below (Fig. 3)

implemented, they can draw from the functionality of existing components (see chapter Discussion).

Trigger script modules

Trigger scripts have many variations, simply because each type of user controller interface or automated script that activates a trigger (Fig. 1) is unique and must be adapted according to its programming, controller logic or good HCI practices.

A trigger may be accessed from either the real, physical world (e.g., pressing a key on a keyboard or a controller button) or a virtual environment (e.g., stepping on a pressure board in a virtual room or passing through an automated revolving door in a virtual building, as illustrated in Figs. 2 and 3).

The user may or needn't be aware of launching a trigger since triggers can be of various types. An example of a trigger is a virtual light switch that actually turns on the lights. Another example is pupil dilation in the user's eye caused by increased cognitive load; this can be used to activate a reduced visual complexity mode of a scene to increase user comfort.

Trigger mechanisms

The trigger scripts below are those currently available. Note, that the concrete scripts are listed without any environmental context.

- **Autonomous trigger**. Any trigger can be set with a time delay for execution or reset a number of times (once, *N* times, indefinitely). An autonomous trigger does this

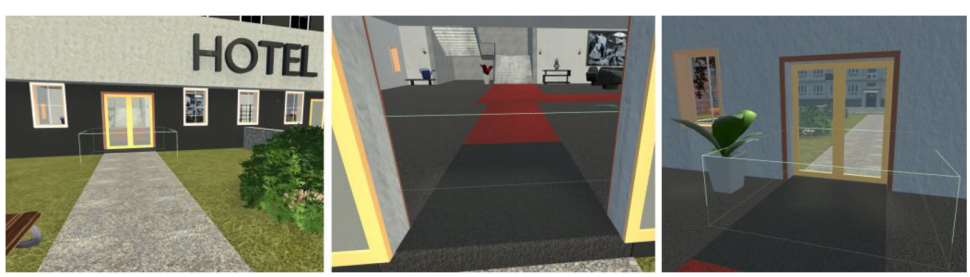


Fig. 3 Behavior example of a two-state trigger. *Note.* As the user approaches the sliding door, it remains in its default ("off") state. In this particular example, this state is defined as a closed-door system. To emulate the functionality of a real, automated door photosensor, both door wings open when the user enters the defined collider area (highlighted in light green). This activates the trigger and toggles the doors into an "on" (opened) state. When the user leaves the collider area, the trigger again toggles the door system and returns it to the default "off" (closed) state

independently. For each instance in a specified period (seconds) during the visualization's run-time, the trigger toggles its targeted object(s). In the virtual environment, this usually translates into self-perpetuating or external entity-driven objects, such as traffic lights, animated looping billboards, etc.

- **Key trigger**. A key trigger defines a list of keys (taken from a generic keyboard or other key-outfitted controller devices), and for each key, a list of objects to trigger. During run-time, the trigger script listens to user key input; if a key existing in its list is pressed, triggering commences.
- **Timer trigger**. A timer trigger is executed using a key and then follows a predefined sequence of times/objects to trigger. Multi-step object animation or a series of slides in a presentation can be executed using this trigger. This is a key trigger that does not execute a single action but a timed sequence of actions.
- **Collider trigger**. A triggerable collider is a defined section of virtual space that registers a user's encroachment. If a user enters, exits or remains within the boundaries of this collider for a specified time, the trigger executes. Standard colliders are invisible to the user (undetected trigger type). They may, however, be suggested or implied in the trigger's virtual space environment that an active component is indeed present (e.g., by adding a button, switch, or pressure plate within the collider's range and animating it when a user approaches).
- **Collider-key trigger.** A combination of the key and collider triggers. Users can only activate a trigger with a keypress if they are present in a certain section of the virtual environment (useful in applying multiple functionalities to the same controller key over multiple virtual areas).
- **Distance trigger**. This trigger observes the distance between an entity (another user or an (animated) object). If the entity is within a predefined distance interval, the trigger is activated. Multiple intervals can be defined, and if the entity repeatedly moves between them, the script can continue triggering its target objects. In principle, this trigger has similar spatial applications to collider triggers; it can also be used, for example, to implement a custom level of detail (LoD) or scaling on objects (Unity Manual: LoD, 2020).
- **Eye-tracking trigger**. With integrated VR eye-tracking, a user's gaze can intersect with a virtual object or collider that has an eye-tracking trigger attached to it. The principle resembles a collider trigger: if a gaze enters, exits or fixates for a specified interval, the trigger is activated. Some good practices with eye-tracking should be considered: due to the human eye's rapid searching speed and also the relative inaccuracy of contemporary integrated VR eye-tracker hardware (Clay et al., 2019), some delay in gaze enter and fault tolerance on gaze stay should be specified. However, triggers controlled with eye-tracking are one of the significant advantages offered by Toggle Toolkit since the mechanism could potentially solve the critical question of analyzing eye-tracking data collected in VR.

Toggle script modules

In contrast to the many trigger scripts, the toolkit's basic toggle script is a universal solution. Using the Unity inspector UI, each instance of the script is attached to the object whose attributes or visuals it will modify.

When toggled, the object transforms into its changed state. When toggled again, it returns to its default state. Such on/off dynamics are the simplest form of the toggle script's functionality and may be compared to the behavior of a light switch or an automated door system (for practical examples, see Figs. 2 and 3; for a schematic depiction, see Fig. 4, left). A togglable object, however, may possess more than two defined states. In such cases, the trigger script executing the toggle script must specify the state that the toggle should switch to (by sending a control number – see Fig. 4, right).

Even in more complex toggle systems (multiple triggers, toggles or toggle states, as described in Chapter 2.2.2), the number of possible states of each object is limited by its attributes and components. Some of these attributes (e.g., position, rotation and scale of an object in virtual space) are universal, but the set of applicable attributes per object is determined by what the specific object allows. If an object does not have a certain component (e.g., when attempting to change the text value of an object that does not have a Unity "Text" component attached), then changing it in the toggle script has no effect.

Toggle functionality

The sets of attributes currently accessible using a toggle script are listed below. If extra Unity object attributes are required in the future, they may be added to the available toggle script or implemented as a new script, as with components appended to objects in Unity inspector.

- **Autonomous behavior**. When toggled, the script can be set to execute only once (effectively freezing the object in its changed state). The toggle script can trigger itself as soon as the virtual scene is initiated. It can also delay toggling or toggle itself to its default state once a specified time passes after toggling by a trigger.
- **User manipulation (teleportation)**. An empty object in virtual space with an attached toggle script (with 3D coordinates regardless) can serve as a teleporter. When triggered, it teleports the user to its coordinates. It can also

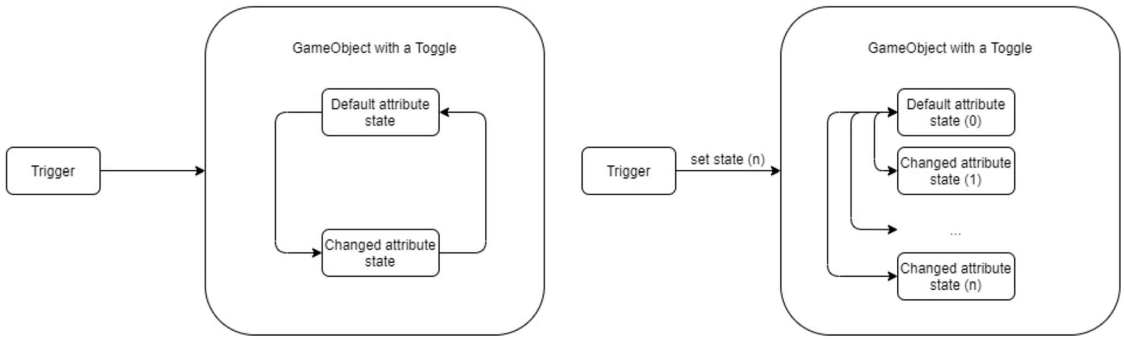


Fig. 4 Overview of simple and complex triggers, with object state transitions. *Note.* Left – A simple two-state (on/off) toggle. Continuous triggering of the toggle results in switching between the object's default and changed states. Right – A toggle containing more than two predefined object states. For this to function effectively, the trigger executing the toggle must specify the intended state to switch to (by sending a control number). When activated, the toggle script switches the object to the state specified by the control number

access the user's viewport and change their direction of view (camera angle).

- **Light manipulation**. If an object has a light component attached to it (any of Unity light), this allows the light to be switched on/off or its intensity and color to be gradually changed.
- **Material and text manipulation**. As long as the object has a visual representation (a mesh), its material or texture properties can be altered. A simple color change in the object can function as a highlighter, and while this does not change the underlying material, the sudden difference in object tint may be enough to draw the user's attention. Different shaders or Unity materials can also be assigned to the object, changing its visuals more comprehensively.

  In some cases, changing an object's text (by using the Unity "Text" component) could be an interchangeable alternative to materials or textures; for example, if the user is exposed to a series of slides.
- **Object manipulation**. By enabling or disabling its mesh, an object can be visually hidden from a scene. Any object can also be initiated in a scene as a hidden object, only to appear later once it is enabled with a trigger. A set of objects can also be cycled, only one appearing at any given time.

  An object in space can be transformed using one or more of the three basic operations: move, scale and rotate. This functionality can transform an object in an instant transition or with a gradual, animated transition. Objects can be transformed relative to their current position/orientation or according to absolute values. A combination of these can produce simple animations. A specialized implementation allows objects to be grabbed/dropped by the user; combined with object physics, this offers a foundation for various complex experimental designs and user task-solving scenarios.
- **Audio manipulation**. With an AudioSource component attached to an object, audio files can be played. Toggle scripts can store audio files in a list. When an audio file is played, the sound originates from the position (center) of the object. This can add a variety of possibilities to a scene, from environmental sounds to music or narrated instructions.
- **External program execution**. The script can specify an external program to run, including command-line parameters. For this to work, the external program or its shortcut must be included in one of the operating system's executable directories.

Advanced toggle states

The basic toggle script is a simple solution: an object is either presented in its default state or a changed state (once triggered). If a trigger toggles this object multiple times, the object would simply cycle between its default and changed states (Figs. 2 and 3). If, however, a more complex state change system is required (e.g., a visual presentation requiring a variety of states to display its slides), a different method must be applied.

A single trigger script can alter multiple togglable objects; similarly, a single togglable object can be triggered with multiple triggers (i.e., an N:N relationship exists between triggers and toggles; for a visual explanation, see Fig. 5, left/right). While linking a single trigger script to an actor's action (e.g., a keypress) may be considered good practice, in some complex cases this may not be desired since the use of multiple triggers features different toggle settings per trigger (multiple triggers used in a manner that complements the dynamic behavior of a targeted object).

Multiple triggers can toggle the same object; this is a feature of our toolkit's design. In addition, each trigger script can

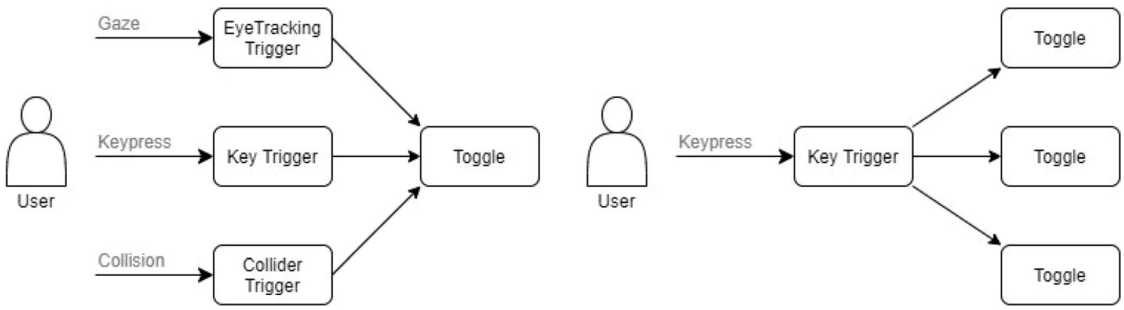


Fig. 5 Relationship between triggers and toggles. *Note.* Left – Multiple triggers can toggle the same object. Right – A single trigger can toggle multiple objects

also pass a unique control number to its linked toggle script(s) (Fig. 4, right). Since many of the aforementioned toggle functionalities store data in ordered lists (e.g., textures, text fields, audio files), invoking a trigger with a specific control number toggles the object while targeting this number in the list(s). The same can be achieved using a single trigger script, such as a timer trigger, which can be set with multiple triggering actions to trigger the same object with different control numbers over time.

Another use case is in designing experimental scenarios. Some objects may require multiple conditions to be met before they can be triggered. This is solved by introducing an iterator script which serves as an intermediary. The sequence in this setup is: Trigger –Iterator – Toggle. The linked triggers therefore first send their trigger command to an iterator. The conditions that must be satisfied before an iterator passes the trigger to a toggle are defined in the Unity Inspector GUI (e.g., the number of times a trigger must be activated, or a collection of control numbers it receives from triggering). Once these conditions are satisfied, triggering is finally passed to a standard toggle.

Another type of logical dependency is an experimental design that specifies when a second action should follow once a first action completes (A → B sequence, i.e., where trigger A must be activated first, and trigger B remains disabled until trigger A is activated). This is achieved by attaching the second trigger to a disabled parent GameObject; the first trigger simply enables the disabled parent, and the A → B sequence may follow.

Data logging modules

Toggle Toolkit is provided with PathScript, a data logger implementation for experimental applications. When the Unity application initializes, the logger reads the program's settings for the logs it should produce. According to these settings, the logger creates new text file logs with prescribed data formats (CSV or other). The logger has default, predefined log output options, such as user movement or interface use (keyboard log, eye-tracking log, etc.) (Fig. 6). The logger automatically and internally manages these predefined log outputs. It also provides logging functionality to other, external scripts. Third-party scripts (e.g., Toggle Toolkit scripts) can access the logger to create their own log files and populate them with their own data (Fig. 7).

Once the experimental application runs, new log entries are written, provided that notable events occur. Some logging is performed periodically (e.g., eye-tracking coordinates), while other logging is per-event only (e.g., key presses, user-initiated triggers).

Understanding the behaviors and events in an interactive toolkit may be a daunting task, especially if a researcher is interested in the interrelations between multiple items. This, however, is not the logger's task. The logger simply outputs logically separated events into separate files or lines. Nevertheless, a researcher may need to know which actor triggered a certain object, which interface was used, or how it affected the virtual environment. This analysis can be done either post hoc, in an external application, or in real-

logId	timestamp	hour	min	sec	ms	eventInfo
0	1574418894	11	34	54	805	ColliderTrigger point-212-entryPrompt1 triggered Canvas-extra
1	1574418895	11	34	55	462	ColliderTrigger point-212-entry triggered point-212-entry
2	1574418896	11	34	56	722	ColliderTrigger point-212-entryPrompt2 triggered Canvas
3	1574418897	11	34	57	384	ColliderTrigger point-212-entryToKitchen triggered point-212-entryToKitchen
4	1574418897	11	34	57	904	ColliderTrigger WateringCanPrefab triggered Cube.001
5	1574418897	11	34	57	909	ColliderTrigger WateringCanPrefab triggered Canvas
6	1574418900	11	35	00	443	ColliderTrigger plant2-kitchen triggered Box030
7	1574418900	11	35	00	444	ColliderTrigger plant2-kitchen triggered Box030 (1)
8	1574418900	11	35	00	445	ColliderTrigger plant2-kitchen triggered alarmCountTrigger

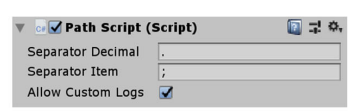


Fig. 6 An example of a standard data output log. *Note.* An example data output using the internal PathScript logging functionality, showing data for a user's movement through a virtual environment. On the right are the logging settings, set in Unity inspector

logId	timestamp	hour	min	sec	ms	xpos	ypos	zpos	xrot	yrot	zrot	keyPressed
0	1574430232	14	43	52	57	31.90038	4.598608	31.13223	353.2997	136.2027	0	up(w)
1	1574430232	14	43	52	679	32.50599	4.598608	30.55055	348.2997	133.3027	0	up(w)
2	1574430233	14	43	53	285	33.09337	4.598608	30.142	329.9997	137.7027	0	down(s)
3	1574430234	14	43	54	296	32.58818	4.598608	30.78666	5.499656	70.40266	0	up(w)
4	1574430236	14	43	56	705	39.34562	4.598608	33.22945	10.89966	86.80267	0	right(d)
5	1574430237	14	43	57	343	39.47714	4.598608	31.5505	355.6996	176.2027	0	right(d)
6	1574430239	14	43	59	797	38.49111	4.587609	24.83524	4.899662	172.7027	0	right(d)
7	1574430241	14	44	1	342	37.7785	4.587609	21.82908	3.699663	259.5026	0	left(a)
8	1574430245	14	44	5	38	27.29935	4.587609	20.00946	316.5997	286.5026	0	right(d)

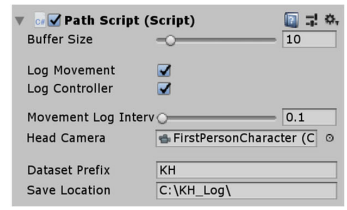


Fig. 7 An example of a custom Toggle Toolkit output log. *Note.* In this example, the collider trigger execution is logged. At a glance, the structure of the log at left is the same as in the log shown in Fig. 6 because all the logs have a single, predefined data structure, regardless of internal or external origin, as shown in the Unity inspector logging format settings on the right

timethrough a custom script that aggregates this type of logging. In any case, the experimenter must decide on the most suitable method. If any data is processed, a specific logic applies to loading or processing different log types separately or cohesively.

External scripts such as those found in the Toggle Toolkit can access and make use of the PathScript logger to produce custom log files (atomic logs or aggregates of multiple modalities). The existing Toggle Toolkit scripts use PathScript to output their integrated basic logging functionality (e.g., logging trigger/toggle state changes, Fig. 7).

Experimental utility

The range of options that Toggle Toolkit brings to experimental design is difficult to present in a comprehensive overview. Two case studies are therefore presented in the following sections, with illustrative examples of Toggle Toolkit's utility in experimental research and psychodiagnostics: one depicts a rather simple use of the Toggle Toolkit and the other presents a more complex application. The paper is supplemented with an example of data cleaning and processing session to complete the presentation of how to use Toggle Toolkit.

Complex 3D Scene Processing

In 2020, our colleagues from Masaryk University conducted an empirical study on processing complex scenes in 1D settings (Čeněk et al., draft). The experiment's design employed eye-tracking, and the experiment's scenery presented to participants consisted of focal objects and a less salient background (a foreground-background task). The experimental hypothesis questioned the intercultural differences in cognitive processing of visual stimuli (Czech and Taiwanese populations).

The aim of this research was to recreate the experiment in immersive 3D and use a VR headset and integrated VR eye-tracking to determine whether different sensory input (binocular imaging with depth perception) would create differences to the original findings by Čeněk et al. (2020). We therefore designed a set of complex virtual scenes consisting of a background and two focal objects (Fig. 8).

Toggle Toolkit was used to manage the flow of the experiment. Each of the 30 virtual scenes was separated from the next with an eye-tracking fixation cross. Toggle Toolkit was used to control and time the experiment. The experimenter had several key shortcuts at their disposal to control the experiment. To validate eye-tracker calibration accuracy, users were shown a trial



Fig. 8 3D re-creation of an experiment with two focal objects. *Note.* 3D re-creation of an experiment by Čeněk et al. (2020), with two focal objects, conducted in combination with a VR headset and VR eye-tracking (one of 30 subsequent scenes, this scene displays a radar and helicopter as focal objects, with terrain as a background)

scene and asked to fixate on certain objects; the exact coordinates of their fixations could then be visualized. With another keypress, the experimenter could then commence the experiment.

Evacuation behavior analysis

A very practical issue arises in evacuation behavior experiments: no easy way exists to evacuate real buildings without major cost, administration and other investments; in addition, people treat drills differently to real emergencies. We therefore conducted a VR evacuation experiment as a proposed ecologically valid alternative to a real evacuation (Snopková et al., 2020); the interactive segment of the experiment was based on Toggle Toolkit. The entire range of implemented dynamic functionalities is shown in the following video demonstration:

https://drive.google.com/open?id=1AOhyH9ES08z0JVCgb1QwRxdbtwUHN8Zf

The experiment recreated the layout of an existing hotel in a 3D model (Figs. 2, 3, and 9). Users were outfitted with a VR headset and directed to explore the building. As users navigated, a fire alarm sounded, and users were instructed to exit the building. The experiment was managed by a complex set of spatial triggers, instructions and events, all based on Toggle Toolkit. This allowed providing the users with text instructions while they navigated through the building and give them an interactive task (watering withered plants in one of the hotel's apartments), followed by a fire alarm. Four experimental groups participated in the experiment. Each group was immersed in a slightly different spatial layout of the hotel (differences in corridor width and corridor arrangement). The route inside the hotel was predetermined according to the text instructions, while the subsequent escape route during the fire alarm depended fully on the participants' decisions and strategies. The experiment's hypothesis explored the roles of evacuation signs and spatial layouts on the participants' behavior. Aggregate data for this scenario was logged by placing invisible Toggle Toolkit collider triggers into the environment. Data acquired in this manner later accelerated the process of data analysis.

Discussion

As illustrated by the schematics and examples in the Software Description chapter and its Toggle Script Modules subchapter, the process of creating a trigger controller logic can be rather complex, especially when multiple triggers are placed in a scene where trigger sequence or dependency is required. When realizing a complex experimental project, we recommend modeling the flow of these scripts and any (in)active objects and toggle states using either the depiction of automata (e.g., UML state machine diagram (Samek, 2008)) or a process modeling tool (jBPM, 2020). This allows the researcher to plan and document their solution before implementation, and if required, review and change it afterwards.

It is worth noting that although only few coding skills are sufficient to use Toggle Toolkit, designing an experiment with this tool provides an opportunity to learn good basics in coding logic since it is necessary to comprehend and well plan trigger–toggle sessions.

Can an experimental design be too complex and involve too many trigger/toggle scripts? Yes, but producing this terminal level of complexity may be beyond most experimental needs. Furthermore, such complexity may be considered from two perspectives: Firstly, it is computational limitations – script implementation resulting in costly processor run-time may reduce the fluency of the 3D experience. Secondly, it is human limitations – it is essential that script dependencies, in-editor clarity, etc., must always be comprehensible to the researcher. Some knowledge of Unity editor is required to design an experiment based on Toggle Toolkit, although this obstacle is not difficult to overcome. Another limitation occurs for experiments that require extreme temporal accuracy,



Fig. 9 Overview of the second floor of the hotel in the evacuation experiment. *Note.* Participants were required to navigate to and subsequently escape from a room on the second floor

for instance those employing EEG. However, this limit is rooted in the features of the Unity engine, not the toolkit itself. For details on the temporal accuracy, see the supplements including data log examples.

We recommend Toggle Toolkit for use in experimental designs of simple to medium complexity, mainly because of the software's current architecture as described in the present paper. Very complex, specialized implementations with many interconnected functionalities should not be implemented in this manner since a general solution such as Toggle Toolkit will eventually reach the limits of its own software design.

Toggle Toolkit is freely available in a GitHub repository, available at the URL https://github.com/kim-dotcom/toggleScript. Toggle Toolkit is an open-source product for application in third-party research on the condition that sources are appropriately credited. Toggle Toolkit is also updated, maintained and developed via this GitHub project page. The toolkit will be kept updated for compatibility with new Unity LTS (i.e., long-term support, annual revisions). Thus, the toolkit will be guaranteed to function for at least several years.

Creating custom scripts to extend the scope of the toolkit's functionality is welcomed. Third-party GitHub submissions will be merged on the condition that they follow the aforementioned logic and good practices. If an experimental design with specific needs arises, extensions made to the toolkit for this purpose will also be merged retrospectively. The toolkit still has scope for future development, however, even its current version offers the opportunity to exploit the potential of immersive VR in research on human behavior with little effort and further training on the side of researchers.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.3758/s13428-020-01510-4.

Acknowledgements This publication was supported by the Czech Science Foundation (GC19-09265J: The influence of socio-cultural factors and writing system on perception and cognition of complex visual stimuli).

Toggle Toolkit programming, debugging and pilot testing of experiments was made available by the infrastructure of Virtual Geographic Environments Lab, Department of Geography, Faculty of Science, Masaryk University.

Experiment preparation and data collection were aided by the infrastructure of the Grey Lab, Department of Psychology, Faculty of Arts, Masaryk University.

This work was supported by the research infrastructure HUME Lab Experimental Humanities Laboratory, Faculty of Arts, Masaryk University.

References

2D Game Kit Reference Guide (2017). Retrieved from https://learn.unity.com/tutorial/2d-game-kit-reference-guide

Alcañiz, M., Rey, B., Tembl, J., Parkhutik, V. (2009). A Neuroscience Approach to Virtual Reality Experience Using Transcranial Doppler Monitoring. Presence: Virtual and Augmented Reality 18(2), 97-111. https://doi.org/10.1162/pres.18.2.97

Baron, D. (2019). Hands-on game development patterns with Unity 2019: create engaging games by using industry-standard design patterns with C#. Birmingham, UK: Packt Publishing.

Brookes, J., Warburton, M., Alghadier, M., Mon-Williams, M., & Mushtaq, F. (2019). Studying human behavior with virtual reality: The Unity Experiment Framework. Behavior Research Methods. https://doi.org/10.3758/s13428-019-01242-0

Caputo, F. M., Emporio, M., & Giachetti, A. (2018). The Smart Pin: An effective tool for object manipulation in immersive virtual reality environments. Computers & Graphics, 74, 225–233. https://doi.org/10.1016/j.cag.2018.05.019

Čeněk, J., Tsai, JL., Šašinka, Č. (2020). *Cultural Variations in Global and Local Attention and Eye-Movement Patterns during the Perception of Complex Visual Scenes: Comparison of Czech and Taiwanese University Students*. PLoS ONE 15(11). https://doi.org/10.1371/journal.pone.0242501

Clay, V., König, P., & König, S. (2019). Eye Tracking in Virtual Reality. Journal of Eye Movement Research, 12(1), 1–18. https://doi-org.ezproxy.muni.cz/10.16910/jemr.12.1.3

Cornelissen, F.W.; Peters, E.M.; Palmer, J. The EyeLink Toolbox: Eye tracking with MATLAB and the Psychophysics Toolbox. *Behav. Res. Method. Instrum. Comput.* 2002, *34*, 613–671.

Cuervo, E., Chintalapudi, K., & Kotaru, M. (2018). Creating the Perfect Illusion. Proceedings of the 19th International Workshop on Mobile Computing Systems & Applications - HotMobile 18. https://doi.org/10.1145/3177102.3177115

D., R., (2015, September 29). The maturity of visual programming. Retrieved from https://craft.ai/blog/the-maturity-of-visual-programming

Diersch, N., & Wolbers, T. (2019). The potential of virtual reality for spatial navigation research across the adult lifespan. The Journal of Experimental Biology, 222(Suppl 1). https://doi.org/10.1242/jeb.187252

Fuad, M. (2019, December 17). XR Interaction Toolkit Preview Package is here - Unity Technologies Blog. Unity Technologies Blog. Retrieved from https://blogs.unity3d.com/2019/12/17/xr-interaction-toolkit-preview-package-is-here/

Gibson, J. J. (1977). The theory of affordances. Hilldale, USA, 1(2).

Gould, N. F., Holmes, M. K., Fantie, B. D., Luckenbaugh, D. A., Pine, D. S., Gould, T. D., & Zarate, C. A. (2007). Performance on a virtual reality spatial memory navigation task in depressed patients. American Journal of Psychiatry, 164(3), 516–519. https://doi.org/10.1176/ajp.2007.164.3.516.

Grübel, J., Thrash, T., Hölscher, C., & Schinazi, V. R. (2017). Evaluation of a conceptual framework for predicting navigation performance in virtual reality. PLoS one, 12(9), e0184682. doi:https://doi.org/10.1371/journal.pone.0184682

jBPM (2020). Retrieved from https://www.jbpm.org/

Kovalčík, V., Chmelík, J., Bezděka, M., & Sochor, J. (2012). Virtual reality system as a tool for education. In Václav Skala (Ed.), paper presented at the *WSCG Poster proceedings* (15-18). Plzen: Union Agency.

Kubíček, P., Šašinka, Č., Stachoň, Z., Herman, L., Juřík, V., Urbánek, T., & Chmelík, J. (2017). Identification of altitude profiles in 3D geovisualizations: the role of interaction and spatial abilities. International Journal of Digital Earth, 12(2), 156–172. https://doi.org/10.1080/17538947.2017.1382581

Loomis, J. M., Blascovich, J. J., & Beall, A. C. (1999). Immersive virtual environment technology as a basic research tool in psychology. Behavior Research Methods, Instruments, & Computers, 31(4), 557–564. https://doi.org/10.3758/bf03200735

Mathôt, S.; Schreij, D.; Theeuwes, J. OpenSesame: An open-source, graphical experiment builder for the social sciences. *Behavior Research Method.* 2012, *44*, 314–324.

Moghadam, K. R., Banigan, C., & Ragan, E. D. (2018). Scene Transitions and Teleportation in Virtual Reality and the Implications for Spatial Awareness and Sickness. IEEE Transactions on Visualization and Computer Graphics, 1–1. https://doi.org/10.1109/tvcg.2018.2884468

Neuro behavioral systems, Inc. Available online: www.neurobs.com (accessed on 24 February 2017).

Paradigm. Available online: http://www.paradigmexperiments.com (accessed on 24 February 2017).

Parong, J., & Mayer, R. E. (2018). Learning science in immersive virtual reality. Journal of Educational Psychology, 110(6), 785–797. https://doi.org/10.1037/edu0000241

Samek, M. (2008). Practical UML Statecharts in C/C++, Second Edition: Event-Driven Programming for Embedded Systems. London, UK: Routledge.

Sewell, B. (2015). Blueprints visual scripting for unreal engine: build professional 3D games with Unreal Engine 4s Visual scripting system. Birmingham: Packt Publishing.

Šašinka, Č., Morong, K., Stachoň, Z. (2017). The Hypothesis Platform: An Online Tool for Experimental Research into Work with Maps and Behavior in Electronic Environments. International Journal of Geo-Information, 6(12). https://doi.org/10.3390/ijgi6120407

Singer, G. (2013). The History of the Modern Graphics Processor. Retrieved from: http://www.techspot.com/article/650-history-of-the-gpu/

Smith, S. P., & Trenholme, D. (2009). Rapid prototyping a virtual fire drill environment using computer game technology. Fire Safety Journal, 44(4), 559–569. https://doi.org/10.1016/j.firesaf.2008.11.004

Snopková, D., Ugwitz, P., Stachoň, Z., Hladík, J., Kvarda, O., & Kubíček, P. (2020). Can increased visual accessibility and signage suppress retracing evacuation tendencies in unfamiliar buildings? A Virtual reality game-based experiment. Manuscript submitted for publication.

Unity Manual: Components (2020). Retrieved from: https://docs.unity3d.com/Manual/Components.html

Unity Manual: LoD (2020). Retrieved from https://docs.unity3d.com/Manual/LevelOfDetail.html

Unity Scripting Reference (2020). Retrieved from: https://docs.unity3d.com/ScriptReference/

Vasser, M., Kängsepp, M., Magomedkerimov, M., Kilvits, K., Stafinjak, V., Kivisik, T., Vicente, R., & Aru, J. (2017). VREX: an open-source toolbox for creating 3D virtual reality experiments. BMC Psychology. https://doi.org/10.1186/s40359-017-0173-4

Vizard. Virtual Reality software for researchers (2020). Retrieved from https://worldviz.com/vizard-virtual-reality-software

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Annex 4: Paper IV

Ugwitz, P., Kvarda, O., Juříková, Z., Šašinka, Č., Tamm, S. (2022). Eye-Tracking in Interactive Virtual Environments: Implementation and Evaluation. *Applied Sciences*, 12(3). https://doi.org/10.3390/app12031027

applied sciences

MDPI

Article

Eye-Tracking in Interactive Virtual Environments: Implementation and Evaluation

Pavel Ugwitz [1], Ondřej Kvarda [1,*], Zuzana Juříková [2], Čeněk Šašinka [3] and Sascha Tamm [4]

[1] Department of Geography, Faculty of Science, Masaryk University, Kotlářská 267/2, 611 37 Brno, Czech Republic; ugwitz.pavel@mail.muni.cz
[2] Department of Educational Sciences, Faculty of Arts, Masaryk University, Arna Nováka 1, 602 00 Brno, Czech Republic; jurikova@mail.muni.cz
[3] Department of Information and Library Studies, Faculty of Arts, Masaryk University, Arna Nováka 1, 602 00 Brno, Czech Republic; cenek.sasinka@mail.muni.cz
[4] Institute of Psychology, Freie Universität Berlin, Kaiserswerther Straße 16-18, 14195 Berlin, Germany; tamm@zedat.fu-berlin.de
* Correspondence: kvarda.ondrej@mail.muni.cz

Abstract: Not all eye-tracking methodology and data processing are equal. While the use of eye-tracking is intricate because of its grounding in visual physiology, traditional 2D eye-tracking methods are supported by software, tools, and reference studies. This is not so true for eye-tracking methods applied in virtual reality (imaginary 3D environments). Previous research regarded the domain of eye-tracking in 3D virtual reality as an untamed realm with unaddressed issues. The present paper explores these issues, discusses possible solutions at a theoretical level, and offers example implementations. The paper also proposes a workflow and software architecture that encompasses an entire experimental scenario, including virtual scene preparation and operationalization of visual stimuli, experimental data collection and considerations for ambiguous visual stimuli, post-hoc data correction, data aggregation, and visualization. The paper is accompanied by examples of eye-tracking data collection and evaluation based on ongoing research of indoor evacuation behavior.

Keywords: eye-tracking; virtual reality; eye-tracking algorithms; data collection; data visualization; dynamic environments; interactive environments

check for updates

Citation: Ugwitz, P.; Kvarda, O.; Juříková, Z.; Šašinka, Č.; Tamm, S. Eye-Tracking in Interactive Virtual Environments: Implementation and Evaluation. *Appl. Sci.* **2022**, *12*, 1027. https://doi.org/10.3390/app12031027

Academic Editor: Enrico Vezzetti

Received: 4 December 2021
Accepted: 17 January 2022
Published: 19 January 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Researchers are generally fond of valid, reliable measurement methods and tools that allow the capture of variables containing rich information and high interpretative value. Eye-tracking may be considered such a profitable method because it captures eye movements and their derived patterns and finds application in the analysis of many types of spatial and visual compositions.

1.1. The Argument for Eye-Tracking

One could argue that the measurement variables acquired by eye-tracking have broad applicability, especially in cognitive science and its applications. The argument defines two types of variables: extrinsic and intrinsic variables [1]. Extrinsic variables (e.g., size) can be measured straightforwardly without the need for expert knowledge in constructing or operating the measuring instrument, whereas intrinsic variables are more challenging to access and measure, as their measurement tools make use of other variables or physics-based conversions to acquire the intrinsic value (e.g., measurement of blood pressure). In addition, not all phenomena can be measured by a device situated in the physical realm; for example, stress levels are not only expressed physically by blood pressure, heartbeat, pupil dilation, etc., but also by high-level cognitive and neurological factors such as stimulus habituation, nervous system excitability, etc. These are called constructs [2] and generally

involve cognitive processing, and unlike variables, are not necessarily expressed by a singular quantitative value.

Therein lies the utility of eye-tracking: not only can it measure eye movements indirectly (non-invasively), i.e., intrinsically, it also lies at the threshold between variables and constructs. A raw data set of gaze coordinate entries (i.e., where a person looks at through a series of time segments) can be extrapolated into visual focus on stimuli (fixations), sequences of such focus (fixation scanpath and fixation revisits), and, subsequently, to provide an understanding of cognitive processing of a given visual scenery [3]. In general, the length of eye-tracking fixations, the objects that are fixated, the scanpath, and the frequency of fixation revisits all provide an insight into a person's cognitive processing (e.g., task difficulty and task-solving strategy). In specific eye-tracking tasks (e.g., scene perception, visual search, reading, and task-driven action), eye patterns exhibited by users can be analyzed in further detail depending on the task methodology. In some instances, other findings can be extracted from eye movements and eye physiology—e.g., different cognitive styles or abnormal personalities [4–6]. Given that the eyes are an outward extension of the brain [7], and that many nonverbal [8] and physiological signals [9] are communicated through them and different processing patterns are used for different visual stimuli [10,11], they are useful data to obtain. These data can also be built on in future and in emerging technologies such as artificial intelligence [12], computer graphics, and virtual reality [13].

1.2. Eye-Tracking in VR

The present paper explores the possible interpretative value provided by eye-tracking in combination with VR technology. To describe the utility and potential of this combination of methods, an insight into their respective technologies follows.

While research into eye-tracking has a history of around a hundred years, it has only gained traction over the last four decades because the technology has matured into relatively simple to use, and easily interpretable, tools. The very first eye-tracking solutions were crude, painful extrinsic measures, which attached direct pen-like transcribers to the eyeball. In 1935, Buswell published a major eye-tracking study based on manual transcription of eye movements captured on film [14,15]. Indirect eye-tracking based on tracking light reflections in the eye, also known as Purkinje images [16], was first implemented in the 1970s and 1980s [17]. This technology uses the infrared light spectrum to obtain pupil center corneal reflections (PCCR), and is the most used eye-tracking method to this day.

Mobile eye-trackers also have a history, albeit a significantly shorter one. Given the technical requirements for processing corneal reflection video feeds and video encoding in such a small mobile device, the technology first had to miniaturize and improve power and performance [18,19] before it became a feasible tool. Mobile eye-trackers have therefore emerged only in the last decade or so; although, they are still less accurate and have less performance in frames per second captured compared to their stationary counterparts.

Along with its application in video-capturing mobile glasses [20], mobile eye-tracking technology has also been implemented in VR headsets. This coincides with the so-called "virtual reality revolution" of the current generation of VR headsets [21,22]. The first viable VR headsets of this generation appeared in 2013–2016 (Oculus Rift DK1 and DK2, and the first HTC Vive), and the first VR eye-trackers followed in 2016 (SMI VR ET addon to the abovementioned headsets). Currently, several vendors offer either inbuilt eye-trackers in VR headsets of their own production or eye-tracking add-ons intended as extensions to existing VR headsets. Advancements in 3D graphics rendering and VR display technology [23] allow the visual quality of VR displays to approach the physiological limits of human visual perception.

What is the practical utility in combining VR and eye-tracking? Perhaps it is the synergy of this combination. VR is known for its ability to render visuals of projected, proposed, or imaginary spaces [24,25], and facilitate interactivity with these imaginary worlds [26] and other people [27]. VR can thus transcend the limitations of time and space. Once the VR visuals are rendered, they can be combined with eye-tracking for numerous

purposes, for example, validation of the spatial arrangement of a virtual environment [28] or user behavior in a virtual scene [29], reduction in graphics performance requirements by applying foveated rendering [30], or the use of eye-tracking as an interactive interface [31].

This can be effected at a fraction of the cost of preparing a real scenario. The software and application program interfaces (APIs), which facilitate graphics processing, and rendering and spatial algorithms are also computed relatively quickly, some even in real-time. Real spatial data, pre-made 3D objects, and graphics pipeline workflows [32] can be employed to create plausible virtual spatial compositions that can be supplemented with spatial data visualizations and evaluations [33,34].

1.3. Current State of the Research Field

The topic of eye-tracking data processing and interpretation is complex. Eye physiology must be considered, and algorithms must be built around it [35]. When eye-tracking is applied in virtual 3D space, new layers of complexity are introduced. Virtual space itself is accessible through interaction metaphors executed in real-world user interfaces. In the virtual world, the eye-tracked gaze is non-deterministically projected from the real world into the virtual world as the user moves about. Dynamic and interactive objects in a virtual scene can further complicate the interpretability of the given visuals. Suitable designs of VR experimental scenes that employ eye-tracking should therefore be considered, and the resulting eye-tracking challenges in evaluating the gaze in VR require a solution.

Prior research [36] and 3D industry implementations have considered some of these issues. Lappi discusses the problems of eye-tracking in VR at a theoretical level [37]. Clay et al. demonstrated the basic use of eye-tracking in experimental VR settings; however, the study lacks an in-depth perspective of processing data in the 3D environment [38]. Tobii XR SDK enables eye-tracking in basic scenarios [31]. A unifying approach to solving the problems related to eye-tracking in 3D VR, however, is currently unavailable.

2. Methodological Issues

2.1. Eye-Tracking Technology Applied in VR

To establish the common ground in all video-based eye-tracking technology, whether VR or otherwise, the similarities in the hardware and software of most eye-trackers must first be examined.

The commonalities in the hardware include the type of technology used and the metrics for sampling rate, field of view, accuracy, and precision. Given the limited space of the interior of a VR headset, the eye-trackers used in VR employ tiny cameras that implement infrared PCCR, which is the most used eye-tracking technology today. The sampling rate, accuracy, and precision of all non-stationary eye-trackers, whether VR or mobile, is limited by the size, layout, and performance constraints of the hardware, resulting in worse parameters than stationary eye-trackers (eye-trackers used in VR sample at 75–120 Hz, at about 3–5 degrees of accuracy, whereas stationary eye-trackers are superior, sampling at, for example, 2000 Hz at 0.15 degrees of accuracy [39]). Most VR headsets use an LED display behind an optical lens to cover a visual field of 100–110 degrees to accommodate most of the clearly perceived human vision; the range covered by eye-trackers used in VR has greater limits due to the optical limits of the system [40].

Eye-tracking software for VR (APIs, calibration procedures, and 3D engine extensions) is tailored to the virtual 3D world. The captured data contain two levels: the raw X/Y coordinate data, as captured through the eye-tracking cameras, and the transformed 3D data, as projected into the virtual world. Whether the raw data are accessible or correctible, and how it interfaces with third-party applications such as 3D engines, depends on the implementation of an eye-tracker API for VR or a 3D engine-specific software development kit (SDK) that draws functionality from the API. At the level of the 3D engine, it would be beneficial if the VR eye-tracking vendor provided an engine-specific SDK or a preconfigured extension (plugin) so that the API link between the eye-tracker and the engine would be well established, out of the box, with official support. This would allow the hardware-

capable features to be utilized in the engine without the need for deep understanding of hardware and low-level programming to access the functionality. All Tobii XR eye-trackers (HTC Vive Pro Eye, Pico Neo 2/3 Eye, HP Reverb G2 Omnicept Edition) [31] and enterprise solutions such as VRgineers [41] and Varjo [42] provide engine-specific plugins for eye-tracking in VR. Behavioral data inputs, such as eye-tracking gaze, fixations, dwell time, and object-to-object transitions, can thus be both logged and used as a means of interacting with the objects in a virtual world. Custom programming functionality in 3D engines can expand on the eye-tracker's extensions and functionality. All this, as depicted in Figure 1, provides a full picture of how an eye-tracking solution integrates with a 3D engine and a virtual world. The industry is even showing an inclination to develop standardized eye-tracking interfaces for VR [31] and SDKs [43].

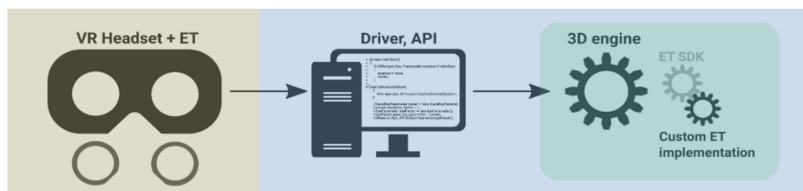


Figure 1. Interconnection of technology. The VR headset hardware is augmented with eye-tracking hardware (depicted in yellow) connected to a PC (depicted in blue). This is facilitated through a driver and a VR/eye-tracking API. The API is then applied in a 3D engine/application (depicted in green), which expands on the API by applying an engine-specific SDK or custom implementation.

2.2. Levels of Interpretation

Traditional 2D eye-tracking has two distinct levels of interpretation, one through eye-tracking coordinates (*point of regard*) and the other through defined areas of interest (AoI). As with eye-tracking in 2D, eye-tracking in 3D VR can observe coordinates in virtual 3D space (*point of fixation (PoF)*—a positional vector coordinate) and virtual 3D objects (as visually defined by 3D meshes). The informational value obtained from eye-tracking in 3D is similar to its 2D counterparts; although, differences exist because of the three-dimensionality of the virtual coordinates and the predefined surface area of the in-engine 3D meshes. Since the virtual 3D space is defined in exact mathematical coordinates, this allows for executing spatial algorithms on top of it.

A virtual 3D world is composed of virtual objects (3D objects and 3D meshes, depending on the terminology and technology of the 3D engine). The eye-trackable 3D objects have physics boundaries (collision geometry) attached to them. At a detailed interpretation level, the 3D gaze coordinate can be obtained when the user's eye-tracking raycast hits the object collision geometry at a particular positional vector. At the simplified interpretation level, the information that a user is looking at the object, is aggregated.

The problem with 3D coordinate tracking is that the subsequent data filtering and analysis is demanding on the required implementation, algorithmization, and processing time. Similarly, the problem with 3D object tracking is loss in measurement accuracy and the inherent issues of object-based geometry, for example, the non-uniform sizes and scales of objects, their frequently illogical segmentation/separation, or their partial visibility in a scene.

Nevertheless, processing eye-tracking data from 3D VR at either the coordinate level or object level does not need to be antagonistic. One approach can complement the other, and hybrids can be implemented. This is discussed in the implementation section (e.g., fixation magnetism, distance-based object segmentation, and fixation algorithms).

2.3. The Complex World behind the Projection

The virtual world presented to a user through the VR headset display does not have any physical dimensionality. It is instead rendered onto the two-dimensional plane of a VR

LED display. Depth perception of a virtual 3D world is maintained by simulating some of the visual cues present in the real world. Monocular visual cues that can be applied in 3D graphics include perspective, 3D lighting, and postprocessing filters for recomputing light occlusion and atmospheric perspective [44]. At this level, *pseudo 3D* level of depth perception is produced (e.g., as achieved by 3D graphics projected onto an LCD screen). By adding the binocular disparity visual cue, i.e., by shifting near object positions to the side for left/right display (as would be naturally perceived by the left/right eye in physical space), *real 3D* depth perception is attained.

Regardless of whether eye-tracking is stationary, mobile, or VR, or whether the stimulus is presented in 2D, pseudo 3D, or real 3D, eye-tracking device will always internally capture eye coordinates as 2D X/Y coordinates of the normalized projection plane where the capture commences (*point of regard, PoR*). The VR headset display is one such projection plane, and it serves as a portal between the real and the virtual world:

- **Head movement** is tracked by the VR headset (or a motion capture setup for whole body movement). Unless otherwise limited, its six degrees of freedom (6DoF) [45] allows direct rotation (3DoF) and movement (3DoF) in real space. User kinetic movement is translated directly into the virtual space, altering the camera's rotation and position.
- **Locomotion user interfaces** allow translating movement into the virtual world—be it through abstraction, movement metaphors, or actual movement tracking. Up to six degrees of freedom are also possible, depending on the complexity and usability of the physical user interface (e.g., keyboard and mouse vs. a VR controller, treadmill, or other (experimental) devices [46]). There is a trade-off between the complexity of a locomotion user interface and the quality of sensory input it offers (e.g., keyboard facilitates on/off movement along one axis at a constant pace—it is a simple interface, but the lack of bodily kinesthetic feedback makes it susceptible to motion sickness [26]; a complex interface such as full-body motion capture allows for realistic, precise movement sensation—at a cost of setting up the interface, physical space demands, and significant purchasing costs). Locomotion user interfaces influence users' perceived ease of movement and distance estimation [47]; they can even facilitate new ways of working with digital content [48,49]. They may also influence viewing behavior—this, however, has not been addressed in any research to date.
- **Eye gaze** coordinates are transformed into raycasts, which extend into the virtual world, i.e., by using the *point of regard* eye coordinate and camera position/rotation in the virtual world. As the ray is cast, it lands onto a spatial coordinate in the virtual world (*point of fixation, PoF*), provided something is in the ray's path to land on (3D objects or surfaces).

In geometry and 3D engine terminology, a raycast is a beam with a positional vector origin and a directional vector angle [50]. In eye-tracking in VR, the origin is the virtual camera's position and rotation as aligned by real-world user movement or the locomotion user interface, and the directional vector is the eye-tracking coordinate angle relative to the camera's position/rotation. The specificity of processing and evaluating eye-tracking in 3D VR is dependent on the applied device(s), algorithm(s), and virtual environment(s). The transition to a virtual world is an extra calculation that traditional 2D eye-trackers need not be concerned about.

The validity and feasibility of eye-tracking analysis in 3D VR therefore becomes complicated (a greater number of measured and intervening variables, and variable transition rules). This complexity has various degrees, however, and was first described by Lappi [37]: while a stationary eye-tracker needs only to observe eye movements, such movements in VR may be occurring in addition to or counter to the user's head or body movements (performed either directly, kinesthetically, or indirectly through a user interface), hand movements, user controller movements, interactions with other objects/users/entities in dynamic virtual scenes, and so on.

A stationary eye-tracker with a chinrest only tracks the user's eyeballs, which move along two axes and have a limited angular range of rotation and can only be projected

against a fixed observed plane. Mobile eye-trackers may have to account for changing surroundings while the user moves about. Beyond the physical dimension, if a subject is rendered in VR, the projection must be converted, for example, by translation of the virtual coordinates into seemingly physical ones. Some stationary VR experiences have limited or no degrees of freedom, for example, 360 video or driving simulators [51]. Many VR devices, however, make full use of the mobility aspect to utilize the full potential of VR as a medium [52]. Eye-tracking in VR scenarios is therefore interpretation-heavy. Below is a description of the interpretation complexity in scenes when considering two control variables:

- **Static camera, static scene**. The VR user observes the virtual world through a stationary, immovable virtual camera with zero degrees of freedom, and the only registered physical movement is eye movement. The user's eye movements have no dynamic or interactive function attached, effectively equating this interpretation level to that of traditional 2D eye-tracking with static stimuli and the user resting on a chinrest. This is the simplest eye-tracking interpretation level, as the *point of regard* is the only relevant variable. The *point of fixation* transition is unnecessary since all the virtual coordinates seen on the screen are the only coordinates there are. Even if the presented scene is a 3D visualization with varying depth, the *point of regard* data set will always be transformed to the same *point of fixation* data set, with 1:1 projection. The acquired data can therefore be reduced to *point of regard* or worked with as if the stimulus were a static 2D image.
  Level of interpretation complexity: eye coordinates over time.
- **Static camera, dynamic scene**. The user has no control over the virtual camera position or rotation, giving them no possibility to change the perspective from which the presented stimuli are seen. The experimental scene can, however, dynamically change in front of the user's eyes. In 3D, this would involve a predefined animation or other actors (AI, or other users with dynamic movement enabled). In classic eye-tracking studies, this can be equated to presenting a video. The majority of experimental control is still retained, i.e., the camera's rotation/position is shown through the virtual camera, but knowing which content is shown, and the order in which it is shown, depends on whether the dynamics are deterministic (e.g., predefined animation vs. another user with dynamic controls). Regardless, this type of scene must be evaluated frame-by-frame. In these types of continuous scene, the positions of objects and their AoI coordinates change in time.
  Level of interpretation complexity: eye coordinates over time with scene movement.
- **Dynamic camera, static scene**. All interaction occurring in this type of scene is movement interaction [52], i.e., the user manipulates the virtual camera. The degree of complexity in recording and evaluating such movement increases significantly since the researcher abdicates control of the camera to the user, and thereby loses control of what is shown to the user, when it is shown, the order in which it is shown, from what angle and distance it is shown, and the overall scene composition. The user's ability to navigate through the virtual scene depends on the physical controller and how the physical controller translates into the virtual scene (in-engine character controller [53]), and how many degrees of freedom such a controller provides, i.e., whether the camera can be rotated and moved and the extent of this movement. Even with simple camera dynamics, the three-dimensionality of a scene, i.e., the virtual space coordinates at and from which the user looks, and the distance between the user camera and the *point of regard*, must be considered for eye-tracking evaluation purposes. Free movement of the camera may cause 3D objects to occlude other objects, creating difficulties in interpretation [54]. Given the rather low precision of VR eye-trackers, measurement artifacts can originate at the border of multiple neighboring objects. The depth irregularity of a 3D scene and the potential measurement error is also projected irregularly onto objects in the 3D scene (objects at greater/varying distances accumulate more absolute-distance measurement errors than near objects of similar size since the eye-tracking precision

error is based on angles, and the absolute distances between angle-based raycasts increase with depth). This level of eye-tracking complexity in 3D VR has no parallel in traditional eye-tracking in 2D.
Level of interpretation complexity: eye movements over time with user movement and scene movement (synchronously).

- **Dynamic camera, dynamic scene**. All the interpretation issues mentioned above apply to this classification. In this case, other actors besides the user can also manipulate the projection and composition of the scene. At the user interaction level, these may be other (concurrent) user interfaces, controllers, or interaction metaphors, which allow selection, manipulation, systemic control, or symbolic input. Scene manipulation may occur at the level of individual objects, groups of objects [52], or the entire virtual scene [55]. At the execution level, events happening in the scene may be user-driven (interaction) or autonomous (AI or script-based), with or without feedback (visible, audible, or unregistrable), with or without affordances [56], (communicating or not communicating to the user that events or objects are potentially (inter)actible). Other users (multiplayer), AI, or interactive scripts may also instill changes within the scene. All these factors raise a new set of computational and visualization questions in the interpretation and visualization of a series of *point of regard* coordinates that target moving objects (especially when the user is concurrently moving in a different direction, or using a different controller).
Level of interpretation complexity: eye movements over time with user movement and scene movement (asynchronously or a multitude of concurrent movements).

The examples as described above of a static, fixed user and a dynamic, moving user who observes a complex virtual scene, are depicted below in Figure 2, sourced from Lappi [37]. These examples do not consider the dynamics of the stimulus, regardless of whether it is a flat 2D video or an interactive multi-actor 3D scene. The movement diagram depicted by Lappi does not include other devices (e.g., motion controllers) with their own fields of reference.

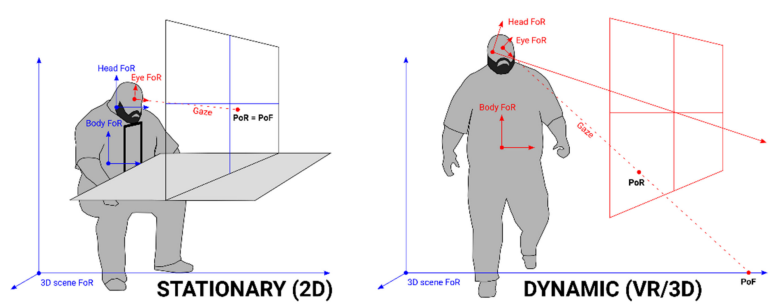


Figure 2. A comparison of interpretation references in 2D stationary eye-tracking and dynamic camera eye-tracking in VR (measured (red) and not measured variables (blue); FoR = *field of reference*, PoR = *point of reference*, PoF = *point of fixation*).

Beyond research interests and theoretical approaches, the potential and limitations of eye-tracking in VR are also applicable in practical implementation, i.e., in algorithms and the optimization, evaluation, and visualization of VR eye-tracking data in virtual 3D scenarios. Previous studies and implementations have considered this only to an extent. The present paper regards implementation as its principal contribution. A discussion and demonstration are given in the following sections.

2.4. The State of Other Implementations

Other studies and implementations of eye-tracking in VR should also be considered to determine their links with the theory described above, and whether they are able to solve

the aforementioned issues. Unfortunately, only a few solutions exist for comparison, and we are reluctant to apply these existing implementations because they interpret eye-tracking data at a rather shallow level or in specific, easy-to-interpret scenarios, only.

A 2018 study [57] employs eye-tracking in VR but reduces all evaluated eye-tracking data to those observed on a flat orthogonal wall in front of a user (effectively staying in the realm of traditional 2D eye-tracking). A 2019 study [37] goes further by visualizing the raycastHit.hit coordinate returned by a basic projection function in the Unity engine API. This is the 3D coordinate where the raycast hits an object collision geometry. The authors visualized this with a sphere, and then colored the sphere according to the distance from which it was gazed upon (the raycastHit.distance numeric value according to the Unity API). The remainder of the article discussed working with this basic eye-tracking data using external statistics, visualization software, and user movement parameters. The in-engine eye-tracking data itself were not processed or corrected into derived variables, and therefore the specifics of eye-tracking methodology in VR, such as data filtering, scene complexity, or fixation processing, remained unaddressed.

The Tobii XR SDK, which is an extension of the SRanipal API used by some VR eye-trackers, implements some eye-tracking logging and visualization functionality [58], such as standard object-level data logging metrics (first dwell time, dwell time, and dwell count) and heatmap-based visualization projected onto supported objects using a dynamic graphics shader [59] to draw the gaze heatmap as an additive texture over already fixated objects/textures. While the solution behind this implementation does provide some value and ingenuity, it is only applicable to qualitative scenes with a low complexity in the 3D geometric variance, such as gazing at a static car object situated in the simple scene of a showroom. If eye-tracking is performed in an environment where, for example, a vast, dynamic cityscape with varying distances and overlapping vistas is present, using a heatmap-based eye-tracking visualization would not be practical or acceptable. Similar issues would arise when qualitative evaluation of the acquired eye-tracking data is intended.

The Vizard virtual reality software extension for researchers from WorldViz, SightLab VR [60] offers features similar to the aforementioned solutions (gaze intersections, fixations, fixations per object, average time of fixation, etc.), which are automatically logged to a text file. It also features a record function that lets the user replay certain sessions with graphical overlays of the gaze paths, fixations, fixation spheres, heatmaps, or intersect points. Vizard and SightLab VR, however, are developed as proprietary software, and hence the functions in the free version of Vizard are strictly limited, or in case of the SightLab VR, must be purchased.

The aim of the present paper is therefore to develop algorithms for eye-tracking in VR that facilitate data logging, filtering, correction, analysis, and visualization. This solution is to function with scenarios of high complexity, providing better methods of managing eye-tracking data in complex scenes, all while respecting the methodology.

3. Implementation

To give some context to VR eye-tracking algorithms that aid in dealing with the interpretative difficulties of virtual 3D environments, their unifying technological foundations should first be explained. This section therefore describes the technology we worked with, the software architecture and design of the proposed implementation, the process of obtaining and working with eye-tracking data, and the desired 3D scene setup for use in an eye-tracking experiment. Based on this foundation, the algorithms that can eliminate some of the issues with 3D VR eye-tracking data are described and implemented. Together with visualizations of the processed data, the solution is able to offer clearer interpretative value.

Partial software implementation of the VR eye-tracking solutions mentioned throughout this section are available at the following GitHub repository: https://github.com/VGE-lab-MUNI/vr-et, accessed on 16 January 2022. This is a demonstration implementation for

the Unity engine version 2019 LTS and newer, using the HTC Vive Pro Eye VR headset and SRanipal API (the technology is explained in Section 3.1).

3.1. The Technology That Is Available and the Technology That Is Used

The technology applied in eye-tracking and VR forms a layered architecture consisting of VR and eye-tracking hardware and low-level drivers (firmware) to facilitate the hardware's basic functionality. This translates into an operating system level driver and API used in a 3D engine. Additional work over the API can be tailored into a premade 3D engine extension to cut implementation costs for the users of the engine. In the end, a set of eye-tracking-related in-engine functions are made available for use.

Due to the current market fragmentation (many vendors of VR headsets, many third party vendors of VR eye-trackers, and multiple 3D engines), the future will likely include a standard for extended/virtual/mixed/augmented reality devices, i.e., multiple APIs under a single SDK [43] that interconnects the hardware with 3D engines and other applications. Standardization of eye-tracking for VR is also expected [61].

The inter-compatibility of devices under a single SDK is significant because it eliminates researcher dependence on a single hardware vendor or device. This is relevant, as hardware dependence could become an issue in ongoing research. For example, in long-term research design, the original vendor-centric hardware could become obsolete or unusable over time (lack of support, feature-breaking update, or hardware error). API/SDK standardization also enables designing and programming of eye-tracking evaluation scripts in a manner that allows their re-use, either immediately or with minor tweaks only, with different VR eye-trackers (within the constraints of the programming language of the selected 3D engine).

The present paper demonstrates an implementation and interpretation of eye-tracking for VR in a virtual indoor evacuation behavior experiment (to observe user decision-making and navigation strategies based on locomotion and eye-tracking). The experiment employed an HTC Vive Pro Eye headset with an integrated eye-tracker (Tobii). VR and eye-tracking functionality was enabled in the engine with the SteamVR application (as required for all HTC or Valve headsets) and applied with the SRanipal API for Tobii eye-trackers [62]. Upon importing the SRanipal API into Unity, the 3D engine of our choice, basic eye-tracking functionality, such as positional tracking and a calibration procedure, was available out of the box. The remainder was implemented using algorithms of our own design.

The currently most used 3D engines are Unreal engine [63] and Unity [64]. We selected Unity for the experiment. The criteria to determine the suitability of an engine to one's (experimental) needs include the programming language and the range of functionality of the provided programming API, the customizability, scalability and extensibility of the engine and its scene editor, available extensions, supported 3D formats and available 3D model repositories, platform compatibility, and graphics fidelity. In other words, a 3D engine is a middleware with various functionalities and quality of life additions, and the end-product software (whether a research application or otherwise) should make the best use of the 3D engine's features to reduce the potential implementation costs (man-hours, budget, workflows, or technical knowledge). Unity provides a C# programming language API, along with its own repository, *Asset Store* (to obtain editor extensions and 3D models). Comprehensive documentation is available, and it has the support of a strong community. However, the level of its graphics fidelity is lacking, in direct contrast with Unreal engine, which is written in C++, has a less widespread community, yet state-of-the-art graphics. It is noteworthy that Unreal engine has been driving advances in the 3D industry by providing feature additions and improvements such as a photogrammetric 3D object library [65], virtualized geometry [66], and a visual scripting language [67]. As such, the engine is well-established for excellence in the future.

Although we used Unity throughout the implementation phase of our paper, the examples are described in a manner so that they can be applied or implemented in other 3D engines.

3.2. The Program Architecture

The core of the proposed solution, as implemented partially in the Unity engine and partially by applying external scripts and applications (data log file processing and statistical analysis), is straightforward and in accordance with the intended program architecture. Most of the provided programming language classes are instantiated in engine run-time either as single instances (singletons) or script components attached to 3D objects existing in the virtual world. This is acceptable, since only one eye-tracking raycaster script, one data logger script, one data processing script, one data visualization script, etc., are required at a time. This also enables extensibility and interchangeability in the provided algorithms, and if the need arises, their reimplementation in other 3D engines.

The complexity of the implementation is evident from the eye-tracking algorithms. These must be optimized for efficiency, i.e., reasonable processing time and memory allocation. This applies especially to algorithms executed in real time, i.e., with a severely limited processing budget (a few milliseconds). Each algorithm must also follow the eye-tracking methodology and apply workarounds to its (and the engine's) limitations.

Figure 3 illustrates the proposed program logic: *experimental runtime* and *evaluation runtime*. The program architecture of the proposed solution follows this logic because the two runtimes frequently apply algorithms that are oftentimes mutually exclusive.

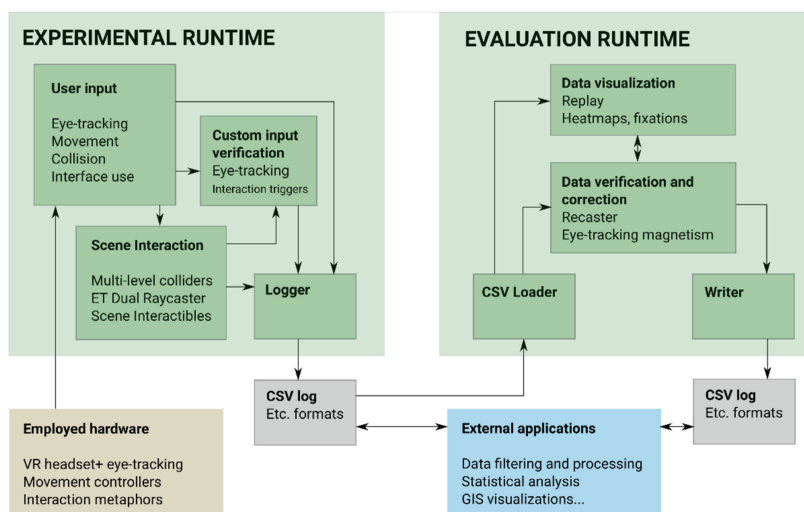


Figure 3. The VR eye-tracking software architecture, as implemented in a Unity project (a series of C# scripts and engine API functionalities). The implementation, although integrated into a single Unity project, separates the program logic into two use-cases: experimental runtime (**left**) for experimental data acquisition, and evaluation runtime (**right**), which is used by the researcher afterwards to process/visualize the data. The data objects outside both runtimes (artifacts) are experimental data logs and aggregated data that are produced by the experimental runtime or reprocessed by the evaluation runtime or external applications.

Experimental runtime should be included in any experimental scenario purposed for data acquisition. The scenario should contain fully realized visuals in the virtual scene, a user controller logic allowing users to move around the environment or interact with their surroundings as intended, and a data logger script that produces the desired data outputs. The user experience here is critical: the user should perceive the experimental scenario as it

was intended. Scene interactivity scripts and data logging should therefore be performed in the background, without disrupting the flow of the experience.

Evaluation runtime is an ex-post set of tools for researchers: i.e., for processing, visualizing, and verifying the movement, interaction, and eye-tracking data produced by the user. The runtime still contains the virtual experimental scenario (allowing the researchers to move through it), but data logging or user experience is not the intent. The user behavior data produced in experimental runtime are then loaded by the evaluation runtime for visualization or processing into aggregated values (then visualized or sent for statistical analysis). In addition, algorithms that are either inefficient for implementation in the experimental phase, too visually intrusive, non-relevant, or experiment-tampering, are applied and used here. These include eye-tracking fixation calculations in 3D space, eye-tracking correction algorithms, visualizations and projections of user behavior, visualization filtration, and user behavior replays.

3.3. Levels of Working with Eye-Tracking Data

Eye-tracking data can be acquired and reprocessed (recomputed and aggregated). This can be performed either non-destructively, by retaining the full information provided by the eye-tracker hardware, or destructively, by reducing the information resolution. This relies on obtaining data from the eye-tracking hardware, the ability to access the data through the eye-tracking API, and the in-engine implementation and virtual scene definition (explained in the following subsections).

What is the minimum required level of distinction in the acquired data for it to be fully useful for evaluation? That depends on the degrees of freedom of the provided VR hardware (the headset and controllers, if any). Generally, it requires:

- Eye-tracking calibration data (success/failure, or more detailed accuracy value);
- Timestamp for each log entry (time, up to milliseconds);
- User coordinates (positional Vector3 in the virtual world);
- User camera rotation (rotational Vector3 in the virtual world);
- Eye-tracking rotation relative to the user camera rotation or the *point of fixation* (Vector3).

This could be considered the bare minimum set of variables to be acquired during experimental runtime. Other derived variables may also be obtained. The question is whether it is better to acquire the derived variables during runtime (for convenience) or to recompute them later, outside experimental runtime (to conserve computational power during the runtime). At a minimum, these derived variables include:

- Eye-tracking rotation relative to the user camera rotation and *point of fixation*—both;
- Gazed 3D object name (object.name value);
- Gaze distance from the user camera to the *point of fixation* (in meters);
- Data loss metric, i.e., a check for missing eye inputs;
- Fixation metric, i.e., a check for whether eye-tracking fixation is occurring, based on the eye-tracker movement velocity and its near-time history, all relative to camera movement;
- Other potential physiological measurements of the eyes, if supported by the API (e.g., pupil dilation size or measurement accuracy);
- Other potential derived variables.

Beyond eye-tracking, experimental runtime can also process other algorithms. To measure user behavior, other logs can be produced, such as user movement logs, interface usage logs, interaction logs, etc. Such interaction logs (event logs) are important, as their execution (based on a temporal event, spatial event, etc.) allows segmentation of the eye-tracking data. An event log is a demarcation that states that *from this event here until another event there, an experimental phase occurs*, and, therefore, *this eye-tracking data section is suitable for further extraction and processing*.

3.4. Setting Up the 3D Environment for Eye-Tracking

Depending on the hardware, software, visual art direction/stylization, and intended scene use, various types of scenes can originate from a number of graphics pipelines and workflows [68]. A graphics pipeline takes into consideration the available rendering technology, for example, the rendering library, 3D modeling software, external sources like photogrammetry, AI-assisted pattern generation, and 3D engine scripting extensions. A graphics workflow is a set of agreed rules and best practices of working with the 3D sources and technologies in the context of a specific project, incorporating the 3D content into a scene in a (semi) streamlined fashion. Namely, a scene can be generated fully from the author's imagination, while the opposite approach is the generation of a virtual scene strictly based on existing geometry (e.g., a replica of an existing real environment). An example of an approach situated between these two is the use of pre-generated 3D objects acquired from real geometry to produce layouts of new spatial compositions. Given the possibilities in graphics shaders and graphics post-processing [69], it is possible to visualize a scene in various styles, for example, realistic, semi-realistic, surrealistic, simplistic, and cartoonish. Depending on the artistic or research intentions of the creator, all these approaches and stylizations have some merit.

For the present paper, we created a digital substitute of an existing building based on its floor plan. The scene was then digitized into a 3D standard used in the construction industry, followed by porting into a 3D engine. The scene was originally created to serve as a digital copy of a real environment with the intent that an experiment could be run simultaneously in VR/real counterparts.

The virtual scene was created based on BIM (Building Information Modeling) documentation presented in the IFC data format. BIM is a commonly used digital representation of the physical and functional characteristics of a building's entire life cycle, with a focus on large-scale details that are relevant in architecture, civil engineering, construction, and project management [70,71]. At the level of implementation, BIM is a container for five-dimensional data, which describes space, material type, and building changes through time. For 3D engine use, the format is excessive and requires simplification.

Because we initiated the conversion process from BIM to the Unity engine, the building's model was converted from the IFC format to the Unity compatible DAE format using the Tridify Convert service [72]. The Unity editor was extended with the Tridify BIM Tools extension [73], which allowed conversion of the DAE data into standard 3D objects native to the Unity ecosystem. However, the imported 3D objects of the building did not satisfy the requirements for graphics fidelity, model organization, and model optimization. Many more manual optimizations were required before the 3D scene was usable in the subsequent eye-tracking experiment.

To objectively recreate the environment in high fidelity, we visited the physical site and acquired reference photos, textures and a 360° spherical video for modeling and texturing the building's environment (a four-story interior with an exterior entry) accordingly. We focused on capturing the physical and visual characteristics (e.g., approximate dimensions, shapes, materials, and colors) and the overall layout of indoor objects so that we could suitably recreate them. The overall physical dimensions (without the textures and optimizations) had already been incorporated in the imported BIM data. Small, decorative objects not included in the BIM data (e.g., furniture and indoor plants) were created from scratch. The general steps in producing a realistic visualization (from establishing dimensions, to detailing and texturing geometry, to lighting) is depicted in Figure 4.

The imported BIM objects required extensive rework and optimization. The hierarchical structure and naming convention of the individual floor objects and construction elements were in disarray after importing, and required rearrangement into a more orderly, logical structure in which objects were grouped according to their affiliation to a certain floor and object type. Not only did this aid in working with the structures of the virtual environment in the Unity editor, it also assisted in the subsequent eye-tracking evaluation (i.e., having unique, structured object names as AoIs that could later be interpreted accord-

ing to the acquired eye-tracking gaze log). For easier interpretation, the interior space was also partitioned into logical spatial polygons (as shown in Figure 5). An object naming convention (e.g., *4A_Plant_3*) thus developed; the example mentioned, representing the third small plant object contained in the "A" spatial polygon on the fourth floor.



Figure 4. Examples of each step in creating the environment: basic BIM model as imported into the 3D engine (**top**), re-creation of scene geometry and textures (**middle**), and lighting generation (**bottom**).

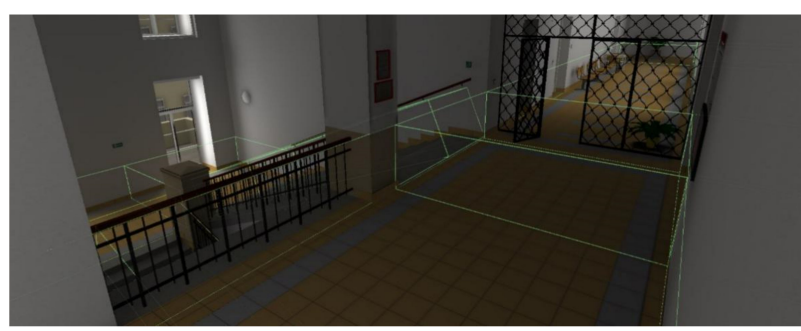


Figure 5. Example of spatial polygon (green outlines) partitions in the area near a staircase.

Many of the pre-existing objects and construction elements imported from the BIM model contained erroneous or somewhat complex geometry (consisting of too many polygons or individual parts). These had to be remodeled since they would cause performance issues in real-time 3D rendering (Figure 6). Other objects that were redundant in the final solution were discarded. Remodeling of unsuitable objects was performed either directly in Unity, using the ProBuilder extension [74] or externally, using the Blender modeling software [75]. ProBuilder's integration with the Unity engine was useful in remodeling the building's construction elements, whereas Blender was superior in maintaining good practices in creating 3D model topology (i.e., detailed objects). Some 3D models were created with two levels of detail (LoD) [76] to reduce the rendering load on the GPU. The result was

a reasonable polycount (i.e., the total number of triangles present in the user viewport of the scene at any one time), which ensured stable performance with no visual frame drops (given the performance of the GPU). Frame drops would adversely affect the user's overall perception of the scene. A high level of in-scene object detail (i.e., high polycount) also reaches a point of diminishing return (there is a significant difference in visual fidelity in observing a 3D model of a detailed object composed of a hundred polygons versus the same object composed of a thousand polygons; the visual difference between a thousand polygons and ten thousand polygons is nowhere as prominent, yet the increased rendering demands are much more significant).

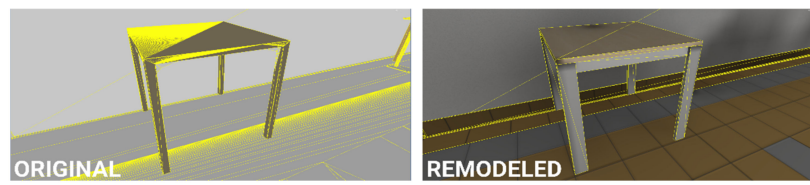


Figure 6. A comparison of 3D model wireframes in the original and remodeled environments.

Regarding the suitability of the final virtual scene for eye-tracking, the important components are not the models and 3D meshes themselves, but rather their physical collision (interaction) with the user's gaze. These physical collisions between objects are mediated by colliders, which are invisible, simplified representations of the actual 3D models. Correct definition of such colliders is important for two reasons: (i) to register the user's eye-tracking gaze (since colliders are the 3D equivalents of eye-tracking AoIs); (ii) to block the user from walking through objects that were not intended to be walked through (e.g., walls or furniture). Inspecting and testing the 3D scene at this point in development is crucial, because we want to ensure the virtual 3D scene retains the features of a controlled experiment with no unexpected issues. We check whether eye-tracking registers object colliders as intended, and that users can only move around within the bounds of a defined area while experiencing no walk-through glitches, fall-through glitches, or other, incorrect 3D geometry-related visual/interaction glitches (gaps or holes in 3D object models, flickering textures, objects defying reality by hanging in space or being unnaturally crammed together, etc.).

Implementing custom scripts into a virtual 3D scene (to provide user interactivity with objects and their surroundings, facilitate events, visuals, or animations, etc., between interactable actors in a scene) has its own challenges. While this paper focuses mainly on scripts related to eye-tracking, some comments about scripting in general should be offered.

Any 3D engine should have an analytics tool for logging and reporting the performance of a 3D scene, i.e., for both visual rendering and script execution. A badly written script that slows down the entire scene can therefore be discovered. In the case of Unity, analytics is provided through the Unity Profiler [77]. There are two kinds of scripts according to their execution time: on-event or loop-based. An on-event script is generally not a problem since the script is executed only once or sporadically (e.g., upon user-object collision). A loop-based script is executed frequently, up to once per every rendered frame. A heavy, poorly programmed script such as this can degrade the performance of the entire application. General script optimization recommendations [77] suggest (re)implementing as many scripts into the on-event category as possible and refraining from using execution-time heavy functions, such as hard drive read/write operations. While some scripts are execution-heavy by design (e.g., continuous user eye-tracking and other activity logging), there are still good and bad ways of implementing them. This is discussed in the next section. The overall performance budget of real-time rendering is always limited: if VR scene visualization and eye-tracking data renders and processes on a common VR device at 90 frames per second, it means we have a real time budget of 11 ms for each frame (1000 ms/90 = 11 ms). The scene geometry rendering and script processing must be

performed below this time budget to ensure a smooth, continuous experience and valid data logging.

Because visuals, scripting implementations, and analytics are all combined, a standardized and automated performance test can verify the performance targets of the scene (as in software integration testing). To test the performance demands of a 3D scene, a user camera can be animated to fly through the scene. When the frame rate is logged during the fly-through, weak areas in the scene with frame rate drops can be discovered. The weak points can then be fixed by adjusting the 3D model polycount and then verified using the same camera fly-through. Standardized performance testing is achieved in this manner.

3.5. Data Acquisition and Related Algorithms

These implementations occur in experimental runtime and involve the capture of mainly raw eye-tracking data. While corrective eye-tracking measures can be employed (described in the following sections), it is economical to acquire the best possible measurements immediately from the experiment as opposed to correcting them later, provided the measurements can be corrected at that point.

An in-engine eye-tracking data acquisition script must execute continuously, producing a stream of dozens of entries per second. The frame rate for logging eye-tracking can be set according to either the VR headset display refresh rate or the sampling rate of the eye-tracker's hardware. Logging eye-tracking in VR with a sampling rate that differs from the VR display refresh rate may need to be handled separately. In Unity, this is solved by the *update* and *fixedUpdate* functions. The *update* function executes scripts at the speed of the framerate (i.e., while visuals are rendered), and the *fixedUpdate* function can be used to process physics, engine program logic, and script extensions at a different rate [78]. With different display/eye-tracker frequencies, it is prudent to employ the *fixedUpdate* function. When this function is used, however, a new visual frame does not always render each time an eye-tracking measurement is taken (or vice versa). This is a minor problem for visual cognition, as these logging/rendering frequencies are sufficiently high to begin with [79]. Regardless, for the ease of the methodology, implementation, and data evaluation, it is preferable to use hardware in which the display/eye-tracker frequencies match and synchronize.

Implementation of user behavior logging is based on a generic logging script with public functions (i.e., functions accessible by any other script). This generic logging functionality can then be used by specific scripts, for example, eye-tracking logging scripts, user movement logging scripts, custom event logging scripts, etc. Once a VR application is launched, the generic logging script waits for other scripts to subscribe to it and specify their own logging formats, i.e., the VR eye-tracking logger calls the generic logger to create a dedicated CSV data file for the eye-tracking log output, along with specifying the log header (the variables). When the VR application is running, the subscribed scripts can pass their own data to the generic logging script whenever required. In the case of VR eye-tracking, this occurs at a pre-specified rate in the fixedUpdate loop. The generic logging script handles writing of the passed-on data into CSV files, along with implementing a writing cache for each of the subscribed loggers to reduce the overall demand on hard drive write performance.

This is the basis behind single-user logging. Implementations with multiple users in a scene (see Section 2.3, *dynamic camera*, *dynamic scene* variable level), may require an extra network layer to ensure correct logging functionality, i.e., the generic logging script must be implemented on a server, and all the user logging scripts of all involved users (separate devices—clients) must subscribe to the server. This ensures synchronization of all the user logs under a single, server-side time. A stable, low-latency and low packet loss local network is also a critical requirement, especially if eye-tracking is logged. For a related issue, refer to Section 3.7 discussing implementation challenges of a robust *replay script*.

For a solid, operationalized understanding of where a user's eye-tracking gaze dwells in uncertain situations in real time during the *experimental runtime*, we propose two algorithmic workarounds: dual raycaster and multi-level collider segmentation.

Dual raycaster—Regarding the fidelity of graphics in modern 3D rendering, an issue with transparent objects arises. *Is the user looking at the transparent object or is the user looking through it and focusing on something more distant?* In any 3D scene, this type of ambiguity in interpretation can arise in two forms: the object is either area-transparent, i.e., not convex, containing gaps/holes (e.g., plants, segmented mesh structures, etc.), or the object is surface-transparent, i.e., see-through texture (glass, liquids, etc.). While it can be difficult to nearly impossible to determine the actual object a user is looking at from a purely algorithmic decision (since the physical cue of eye accommodation present in perceiving near/far objects in the real world is non-existent in VR), it can at least be beneficial to log both cases of the ambiguity, i.e., simultaneously register both the transparent object and what is behind it (Figure 7). This is solved by assigning all transparent objects/colliders into a separate semi-pass-through object layer, followed by implementing a dual eye-tracking raycaster: one raycast is permitted to pass through the transparent layer while the other is not, and both are logged simultaneously.

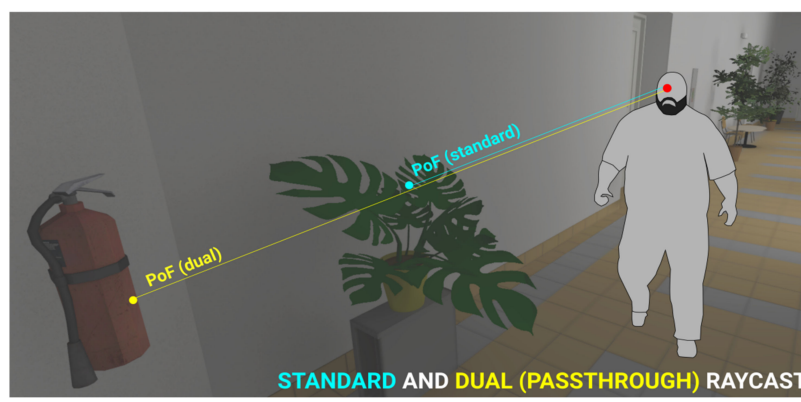


Figure 7. Dual raycaster principle: if an object is flagged as transparent (as the plant in the example), one ray passes through it, and the other hits the object.

Multi-level collider segmentation—The interpretation of dwelling on distant objects can be challenging. The resolution of even the most advanced VR headsets of 2021 is noticeably less than the resolution provided by the fovea of the eye [23], and therefore distant and sufficiently small objects or a combination of these (i.e., objects of a low relative size) will be represented by just a few pixels when the virtual image is rendered. The issue may concern entire objects or only subsections of objects, provided the division of objects into segments is desired. A researcher interpreting such problematic data may ask whether they represent a valid eye-tracking gaze upon a relatively small object or just noise in the data. The inherent inaccuracy in all current eye-tracking technology also represents a problem in interpretation. A superior method of handling this is operationalization, i.e., defining the distances from which an object is relevant as a potential gaze target. At a global level, this can be defined by a maximum permitted eye-tracking raycast distance (calculated as the distance of a straight line from the user's virtual camera position to the potential *point of fixation* on a 3D object). The global maximum raycast distance does not, however, take into account relative object size. Any 3D object in a scene relevant to eye-tracking can therefore be segmented according to a custom-distance multi-level collider distinction; for example, *non-registered distance* (sufficiently far-away to be disregarded by the eye-tracker), *standard distance* (standard eye-tracking collider (AoI) registration), *detailed distance(s)* (sufficiently close for the single object collider to be broken into segmented

parts). Established in this manner according to each relevant object of interest in the scene, the setup checks for user-object distance, and upon crossing a threshold distance, the colliders assigned to the object are interchanged or enabled/disabled (Figure 8). This can be implemented in two ways: the simple implementation works with predefined threshold distances (specified manually for each object), whereas the advanced implementation uses algorithmic detection of relative object size, as seen by the user, to set/disable object colliders dynamically.

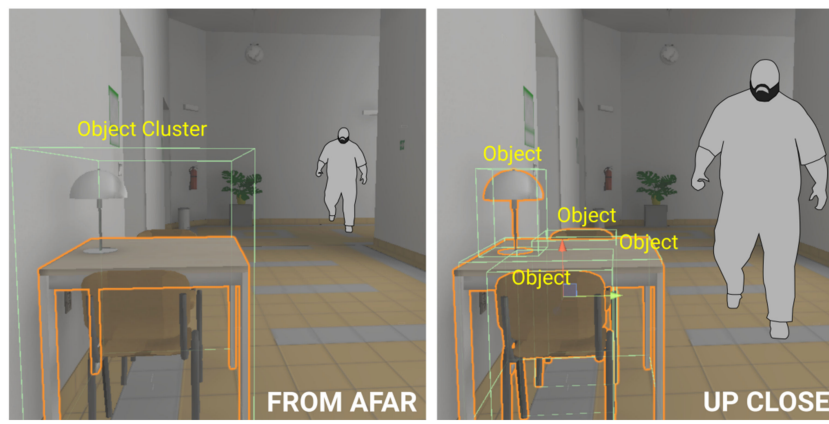


Figure 8. Multi-level collider segmentation principle. As the user moves in/out of the intervals where the object is registered for eye-tracking (with a different segmentation detail), the related collision geometry is altered.

A specialized collider adjustment scenario arises when objects of significance are tracked in a scene where the surrounding objects can be disregarded. In this type of scenario, the collider can be enlarged or shrunk continuously and linearly (as opposed to the aforementioned discrete collider level transitions) to adjust to the user distance and eye-tracker accuracy threshold. This, however, is a computationally intensive operation, one that should be used only for a select few objects, if at all.

Both the dual raycaster and multi-level collider algorithms require some manual input and testing from the researcher before application in a 3D eye-tracking experimental scene. The algorithms can, however, contribute to more intelligible data output. A hybrid of the two algorithms can also be implemented, incorporating a dual raycaster solution in which objects of a virtual environment are tracked in higher/lower detail/segmentation simultaneously. This approach can offer multiple levels of interpretation.

3.6. Data Cleaning, Curation, and Related Algorithms

The data obtained in *experimental runtime* may contain some inaccuracies introduced either by user physiology (brief sudden eye-movements and data loss caused by blinking, or miscalibration) or the eye-tracker (inaccuracy). Some of these inaccuracies can be corrected, while some should be discarded. The algorithms proposed for this purpose adopt two possible data curation paths: (i) clean the data and send them to an external application for statistical processing; (ii) clean/correct the data and reload them into the 3D engine for more processing or visualization (i.e., *evaluation runtime*).

Some eye-tracker APIs inherently report data loss, i.e., moments when no eye-tracking pupil or corneal signal could be registered. If reported by the API, it is best to implement this into the logger so that the data loss value (Boolean true/false) is included in the log. If data loss is already present, the data cleaning process may simply erase the data entries flagged for data loss. If, however, no data loss is present in the log (whether due to a missing implementation or API feature), data loss events can be recomputed using an

external data processing script. A data loss event may manifest as the eye-tracking raycast angle not moving relative to the user camera rotation. In this case, detection and removal of the data loss is accomplished by computing the difference between the user's camera angle and the user's eye-tracking raycast angle. If the difference remains static across a time period (i.e., subsequent logs), this can be considered data loss and removed.

A similar data cleaning procedure entails identifying eye-tracking artifacts (eye saccade movements so unnatural that they would be physically impossible to perform), for example, a measurement error caused by blinking, in which the eye-tracker measures an incorrect eye-position while the eyelid is partially closed and in motion and where the assumed eye coordinate is projected into a radically different direction in a very short period of time (this would be physiologically impossible). Such incorrectly measured data can be erased simply by computing the differences in angles between subsequent logs and removing the instances that contain large, outlier numbers.

These types of correction are rather primitive since they rely on deleting incorrect data. However, corrective approaches can also be considered and implemented by repairing, interpolating, or reinterpreting data. This is often feasible, as one of the advantages of eye-tracking in VR is that the virtual world is geometrically defined and therefore fully measurable and computable using linear algebra [80].

Recaster loads an existing eye-tracking log acquired in *experimental runtime* into the 3D engine and once again raycasts the data onto the environment, i.e., by accessing each subsequent eye-tracking log entry, user camera position (raycast origin) and raycast angle to reproduce each raycast hit or *point of fixation*. This is useful when the naming convention or atomicity of the objects in the scene is redefined after data collection, for example, to obtain more accurate data resolution or for corrective reasons. Recasting the raycast does not alter the time, user position, head rotation, or eye-tracking gaze angle data in the eye-tracking log, only the information concerning the raycast-hit object's name. Coordinate and distance may also be overwritten if objects or colliders in a scene are altered. However, it must be emphasized that while redefining invisible object colliders or their names seems logical, the act of significantly rearranging the visual layout of the 3D models that constitute the scene would be scientifically incorrect, producing either skewed data, or representing a process similar to statistical data tampering.

Eye-tracking magnetism is an implementation for correcting eye-tracker inaccuracies based on machine-estimation of the importance of objects in the focal area. It takes into account a predefined level of potential inaccuracy, depending on the hardware and user eyesight (often a circular spread of five to ten degrees is considered a potential focal area [81]). Then, instead of only a raycast, the algorithm considers a conecast (a conical projection with a diameter equivalent to the specified range). On initialization, the script loads user data that have already been collected, as with the recaster implementation, but now the raycast angle is the central point of the conecast. The algorithm then considers all the objects captured within the cone's diameter (Figure 9). Depending on the setup and the algorithm, the magnetism feature can estimate and attempt to correct what the user is looking at. *The magnetism principle specifies that if a user's gaze dwells on an insignificant area near a significant object, the algorithm will snap-correct the gaze onto the important object*. However, what determines an object's importance is a matter of context and manual setting-up. For example, if the conecast catches the user gazing at a bland-looking white wall with a sufficiently nearby contextual object of importance (e.g., a navigation sign), the algorithm will assume this is a measurement error and snap-correct the gaze onto the object of importance. The object, however, must be manually pre-set as important (or a complex prediction algorithm capable of spontaneously determining the object's importance can be employed).

Nevertheless, correcting the user's gaze onto a single important object in close vicinity is a simple example. The eye-tracking magnetism evaluation process becomes more complicated as multiple objects of importance are captured within the conecast. When this occurs, the question arises as to what type of heuristic the algorithm should employ. Does

it consider the object whose prominence is closest to the cone's center? Does it consider the object with the highest percentage of its volume within the conecast? Does it consider the object with the highest significance rating (as defined by the object's shape and texture, relative to its surroundings)? Or does it consider a combination of all these factors, and if so, what significance do individual factors play in this combination? As seen from this line of thought, the conecast algorithm is only a proof-of-concept at this point, with full implementation depending on other algorithms or heuristic processing.

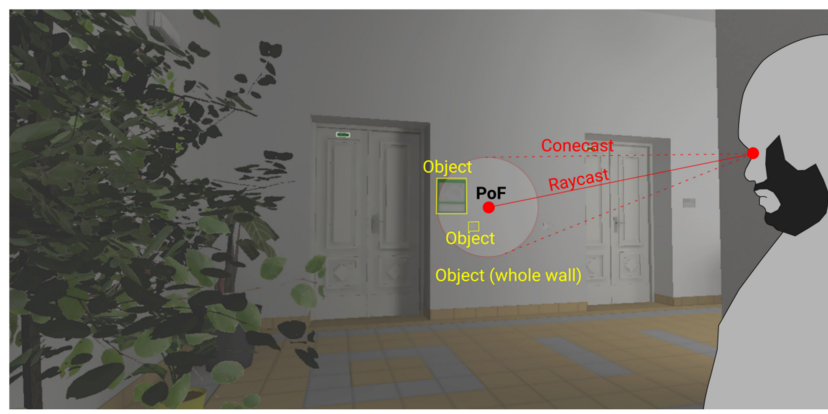


Figure 9. Principle of eye-tracking magnetism. When multiple objects are detected within the defined conecast (e.g., wall, timetable, and light switch), the magnetism heuristics determine which object it should potentially snap to.

3.7. Data Visualization and Related Algorithms

Reimporting CSV data logs into a 3D engine for visualization purposes can be achieved by programming language data structures that efficiently load and work with database-like logs (e.g., *List*, *Dictionary*, and *HashTable*). A challenge is the amount of data that are loaded and the engine's capacity to process and visualize these data. A 3D engine is ill-equipped to deal with thousands of atomic log entries, each represented as a single coordinate object, a *point of fixation* (as acquired according to the eye-tracker's logging rate). Additionally, visualizing raw data brings about less clairvoyant interpretative value compared to the value processed data offers. Some analytics may therefore require pre-computing the raw CSV data by using external or in-engine scripts (which apply spatial computations).

Regardless, processing the data in *evaluation runtime* implies that some evaluation slowdowns are acceptable (since a substantial loss in framerate or an algorithm-induced lag due to instantiating and handling thousands of small objects is a likelihood). It is therefore acceptable to first instantiate all the data log entries in the engine, process them while considering the existing 3D geometry, and cut out and simplify them afterwards, provided sufficient computer memory is available to manage all the log entries and computations.

Before algorithmic data processing commences, another method of comparing the data is useful, for example, manual verification facilitated by dynamic replay functionality. For this purpose, a **replay script** can be implemented. This contains two perspectives for consideration: the programmer's perspective (i.e., the format of the input data, and what the script can do with such data), and the user's perspective (i.e., the user interface, presentation of the data in a comprehensible manner, researcher control, and convenience of the data and related visualizations, and the utility of this).

Given the list of minimum content required in eye-tracking data logs (Section 3.3), a data source such as this is sufficient. The replay script should, however, be designed and implemented so that different data logs (if produced by different eye-tracking logging scripts) can be loaded, i.e., the replay script should not rely on hard-coded, unspecifiable

naming of variables. Ideally, it should also include algorithms for recomputing a missing variable from an included variable, in which the former can be derived from the latter.

In the user interface, the basic implementation is a temporal one: the replay script takes control of the user's camera and uses the loaded CSV positional and rotational data to replay the user's movements, and gazes exactly as the user moved about in the experimental environment. Visualization of the eye-tracking data can also be added to the replay. The eye-tracking coordinate is visualized as a small object (e.g., a sphere) and rendered at the *point of fixation* coordinates that were acquired at a specific time, according to the existing CSV log entries.

Beyond the basics, the replay script's functionality and presented visual information can also be extended. To glimpse the cognitive processes behind the user who is observing the environment, a trailing fade-out of the last few previous eye-tracking coordinates can be rendered. The replay functionality can be slowed down, accelerated, rewound, or paused. To control this feature, basic GUI (graphics user interface) or keyboard shortcuts can be implemented. The GUI can also display a spontaneous overview of eye-tracking information, which includes the name of the currently fixated object, the amount of time spent dwelling on the currently fixated object, etc.

A robust replay script can also visualize multiple inputs from various perspectives, i.e., if data are collected during a multi-user (multiplayer) experiment, multiple sources of simultaneous user behavior will be available. Similarly, interactable or autonomous objects present in the virtual scene would need data logs of their own, as they may move about, change their properties or states, or be acted upon by a user. If such an enormous amount of interaction occurs in the experimental scene, replay functionality implemented by manipulating a single user's camera would not suffice. It would then be necessary to supplement the replay GUI/shortcuts with another, free-to-move observer camera that the researcher can use to observe multiple users/objects from their own perspective or projection. Such a robust multi-actor replay script, however, is beyond the scope of this paper, but having such a utility developed and available would lead to further advancements in the research capability of VR interaction.

Eye-tracking data can be rendered as a static visualization. Upon loading a CSV eye-tracking data file into the 3D engine, the data are then portrayed in the scene. It can be portrayed either fully (raw, unprocessed data), or further processed by an external script or an in-engine eye-tracking aggregation algorithm. Other related eye-tracking data (e.g., user movement, interaction) can also be visualized.

When the individual log entries are loaded and instantiated into the 3D engine as 3D objects, it is good practice to attach a data container to each object. While a data container merely copies or recomputes the data in the CSV log entry from which the 3D object is created (e.g., log id, log time, and raycast distance), this procedure serves optimization purposes. Instead of searching the log table data structure for each 3D computation of the log-originated 3D objects, the data can now be accessed directly from each individual 3D object according to its spatial dimension. This also aids in linking the instantiated eye-tracking objects with other instantiated objects (e.g., eye-tracking coordinates to movement coordinates).

Another key aspect of processing static eye-tracking data is optimization of the algorithm. A comparison of the entire data set of coordinates, entry-to-entry, constitutes an exponentially growing computational difficulty (factorial). Some demanding algorithms may therefore encounter a limit in available memory or in the sheer computational time required for processing. Implementing spatial algorithm optimizations, such as spatial partitioning [82], or at least a basic form of data prefiltering to disregard the irrelevant entries, is recommended.

Heatmap algorithms (absolute distance) are a basic form of visualizing spatial attention. While the interpretative value of this method is already limited in traditional eye-tracking due to a lack of the temporal factor [83], the situation is worse with 3D eye-tracking, especially with differentiating depth in a highly varied 3D environment. The

standard heatmap algorithm considers the absolute distance of all the eye-tracking *point of fixation* coordinates and determines how closely they are lumped together. Differences in spatial depth, however, introduce differences in absolute point-to-point distances. The standard heatmap algorithm is therefore not suitable since it must be corrected for relative distance and (optionally) also the temporal factor. We mention this algorithm because it serves as the basis for all other static visualization algorithms in eye-tracking. The algorithm checks for the distances of other points and how close they are to the *point of fixation* currently being processed. Based on the number or proximity of neighboring points deemed close enough to the current point, the current point is assigned a heatmap index value. Then, as all the individual points are processed, they are colored with a color gradient according to their assigned values (e.g., gray/invisible = below the threshold of consideration; light gray = minimum amount of gaze concentration; red = maximum amount of gaze concentration).

The heatmap visualization can take the form of colored geometric primitives (e.g., a small sphere for each *point of fixation*, as depicted in Figure 10), which is a simple, albeit crude implementation. A more challenging implementation entails visualizing the heatmap as an overlay texture or as a custom graphics shader (i.e., decal system or decal shader [84]). This, however, requires a complex and constrained implementation, because most 3D scenes are composed of multiple objects that contain multiple overlapping textures, some of which use complex texture shaders (the object heatmap overlay is a shader that can interfere with other shaders). Since designing a 3D scene for use with a heatmap overlay texture requires adapting the scene creation process to the heatmap's constraints and involves increasing implementation costs, the present paper disregards the use of overlay heatmap textures.



Figure 10. A basic, geometric primitive-based heatmap visualization algorithm. The point of fixation coordinates in the eye-tracking log are visualized into the scene as geometric primitives (spheres). These are colored according to the number of other points in the vicinity of each point (gray = lowest concentration, red = highest concentration).

Corrected heatmap algorithms (relative distance) are more suitable for obtaining spatial information in VR eye-tracking data with user movement in the virtual space. This is a valid correction in which point-to-point distances are determined according to an observed angular distance rather than an absolute metric distance.

Interpreting absolute point-to-point distances projected onto varying spatial depth in a non-corrected heatmap algorithm can be alleviated, to an extent, by a simple implementation of cut-out distances. Computing an absolute-distance, uncorrected heatmap while including only the points of fixation that were observed within a specified distance interval will discard the remainder of the points from processing and visualization. While this procedure is simple, it does not solve the root cause of the problem—the need to interpret sequences of *PoF* data with a high variety of absolute distances (e.g., just like looking down the corridor in Figure 10). Thus, environments with high spatial variance (or

with user-induced movement causing such variance) are not fit to be interpreted by the uncorrected (absolute distance) algorithm.

The corrected heatmap works well for all distances since it uses conecast. Cone projection from the user's position covers the eye-tracking *point of fixation* coordinate and its surroundings. The angular width of this conecast can be specified according to the algorithm's parameters. Only the surrounding points captured within the cone are then considered for the corrected heatmap computation (this also helps tremendously with the issue of optimizing the otherwise exponentially rising difficulty for the algorithm, where each point would require measurement of its distance from all the other points). However, because the points are compared to each other in relative distances, it implies that the difference in user movement will lead to different results. *For example, comparing points A–B may determine that they are within a sufficiently close distance, whereas comparing points B–A may yield the opposite, especially when the gaze produced to generate log entry B was taken from a shorter distance, where the cone's angular distance would not protrude sufficiently far from point B to include point A.* Another issue is that of occlusion, in which a varying camera angle/position conecast may not register some points previously looked at. Hence, this may introduce variation in the heatmap's visualization intensity/coloring, which can be either kept as is, or the algorithm can be expanded to smoothen such A–B/B–A differences.

Corrected fixation algorithms (relative distance–time) are an extension of the previous algorithm, with the ability to compute eye-tracking fixations in virtual 3D space. The angular distances between subsequent points can effectively distinguish between eye-tracking fixations and saccades; the minimum or maximum duration of a fixation can be set and visualized (the algorithm is now a sequential pass-through on the eye-tracking data log). Similar algorithm (including discussing suitable time and angle thresholds) is proposed and experimentally verified by an existing study [85].

Depicting eye-tracking fixations, saccades, and scan paths is also a visualization issue. The output visualization can produce many fixations that overlay each other, as a user may observe a virtual 3D object of interest from many different angles and distances (we can argue that not only do 3D visual stimuli offer more information than their 2D counterparts [86], they also require more cognitive processing [87], i.e., dwell time). It is also relevant that the user moves about in the 3D environment. Subsequently, the following must be visualized: eye-tracking scan path with fixations, user movement path, and associated eye positions at any given time. While this is easily implementable in a 3D engine (using a series of geometric primitives to depict the *points of fixation* and basic geometric lines (e.g., Unity's *LineRenderer* class) to connect the subsequent points into a path), the comprehensibility of the resulting visualization may be worth further consideration, filtering, or simplification, etc., especially for cases of prolonged interaction or multi-user visualizations.

4. Discussion and Conclusions

The paper discussed the processes and challenges of eye-tracking in spatially complex or interactive VR environments with high degrees of user freedom. The paper explored and applied these challenges at a theoretical level to define a desired level of interpretation in eye-tracking for the design and production of virtual environments that adhere to this level, to clarify the data processing requirements and partially automate data processing, and to explore methods for applying corrective actions to the acquired data (facilitated by the proposed algorithms).

4.1. Contributions to Eye-Tracking in VR

The paper provided an overview of working with eye-tracking data for experiments or applied scenarios built around the theory and proposed algorithms—all the processes and data flow related to eye-tracking in virtual environments are proposed to be regarded as a software architecture and a structured approach (Section 3.2, Figure 3). Specifics of designing and building 3D virtual environments optimized for eye-tracking research are

covered (Section 3.4). During data collection, a raw eye-tracking data stream (Section 3.3) is projected against the virtual environment and saved into a CSV file (Section 3.5). The raw data are then processed through an external script and cleaned of noise. The data can then be further processed externally to obtain basic statistical metrics, or the data can be imported back into the 3D engine for the 3D/in-engine algorithms to execute spatial corrective processing (Section 3.6). The reloaded or reprocessed data can be visualized and interpreted in-engine, over the experimental environment (Section 3.7).

Given the breadth of the subject, the paper is not exhaustive. Future research will address the paper's shortcomings and extend its potential—mainly by extending the provided algorithm examples into polished solutions that are applicable in experimental settings and verifiable for their algorithm and interpretation efficacy. While the linked GitHub repository is documented in both its readme file and source code comments, future work predetermines the repository to become a living text with updates and additions. Regardless, the GitHub algorithm source code examples published along with this paper will serve as a basis for future implementations. The algorithms were implemented as complements to the algorithm proposals described in Sections 3.5–3.7. The included algorithms are listed and described as follows:

- **experiment\PathScript.cs** is our custom-built logger. On its own, it implements generic logging functionality (i.e., user movement, user interface usage, and user collision with other objects (e.g., spatial polygons depicted in Figure 5)). Moreover, custom logs to be passed to it, to be written into CSV files. Eye-tracking data can be one of such custom inputs.
- **experiment\DualRaycaster.cs** is an extended implementation of the raycaster function provided by the SRanipal API [62]. The second raycaster can be specified to pass through some virtual environment object layer(s) (Section 3.5, *Dual Raycaster*). It utilizes the PathScript logger to produce eye-tracking CSV logs. The logged variables are derived from the variable listing needed for further eye-tracking data processing (Section 3.3).
- **experiment\MultiLevelColliderClient.cs** and **experiment\MultiLevelColliderServer.cs** is a server-client solution for segmenting eye-tracking colliders (AoI) according to the user-object distance (Section 3.5, *Multi-level collider segmentation*). The server-client separation is intended so that the collider switching logic can be customized regardless of the collider-switchable objects present in a 3D scene—meaning that multiple clients subscribe to one server. Depending on server implementation and set-up (performance and evaluation heuristics), multi-level eye-tracking colliders are switched through the experimental runtime.
- **verification\CSVReader.cs** is an auxiliary script that loads CSV files back into the evaluation runtime (Section 3.2), to be converted into a List<Dictionary<>> data structure. Such data structures can be processed further by data cleaning (Section 3.6) and data visualization (Section 3.7) algorithms.
- **verification\Recaster.cs** is a re-raycasting script intended to reprocess existing eye-tracking data onto a scene with post-hoc altered 3D object and 3D object collider naming/structuring (Section 3.6, *Recaster*). The script utilizes CSVReader to load existing data; the 3D coordinates (user positional, user camera rotation, and eye-tracking angle) are used to re-raycast eye-tracking gaze onto the altered virtual environment, and the acquired results are saved, again, using PathScript.
- **verification\et_recalculation.pde** is an external script written in Processing programming language. The script processes eye-tracking CSV files produced by the raycaster/logger, to filter out junk data, data loss, and gaze segments too brief to be considered fixations. The provided **verification\et_recalculation_sample** directory also contains example input data and their reprocessed counterparts.
- **visualization\ReplayScript.cs** provides replays on acquired user behavior data including eye-tracking data, if existent (Section 3.7, *Replay script*). As the script loads a user's CSV log using CSVReader, it takes control of the evaluation runtime cam-

era to move and rotate through the virtual scene in real-time in a same way as the user (as specified by the positional and rotational vectors included in the CSV file). Eye-tracking data can be visualized, e.g., as a small white sphere rendered at the *PoF* positional vector, as hit by the eye-tracking raycast back in the experimental runtime.

- **visualization\HeatmapVisualizer.cs** and **visualization\PathVisualizer.cs** are two examples of visualizing existing CSV data (Section 3.7, *Heatmap algorithms*). PathVisualizer uses Unity LineRenderer class to show user movement trajectory through the virtual environment. HeatmapVisualizer processes eye-tracking data (Figure 10). Given the computational difficulty of processing large quantities of eye-tracking data, the example HeatmapVisualizer already contains some optimizations: a setting to process only interval subsets of eye-tracking data, and another setting to process only area subsets of eye-tracking data (the area can be defined by assigning a spatial polygon).

DualRaycaster.cs is an extension of an existing raycasting script included in the SRanipal API [62]. CSVReader.cs is a generic data file processing script adopted from an external source [88]. Original authors of both scripts are mentioned in their respective source codes. All other source code is a work of our own. As for algorithms that were merely proposed in the paper and not implemented (due to their complexity and the need for further work), this concerns Section 3.6, *Eye-tracking magnetism*, Section 3.7, *Corrected heatmap algorithms*, and Section 3.7, *Corrected fixation algorithms*.

The main point this paper drives is that while virtual 3D environments prove to be a challenge in collecting and evaluating eye-tracking data using existing conventional 2D eye-tracking methodics (especially if said 3D environments are highly spatially and structurally complex, dynamic, and interactive), virtual 3D environments also are explicitly defined in their 3D vectors and structures. As such, virtual 3D environments are mathematically computable, and a good-enough spatial algorithm should then be able to make sense of the complications arising from introducing the third dimension into eye-tracking. Therein lies the utility of the provided and proposed algorithms. This is not to say that the existing body of knowledge in traditional 2D eye-tracking (or mobile eye-tracking, for that matter) is of no use here—it is, in cases where spatial 3D computations are not relevant (e.g., when dealing with *PoR* or aggregated data). When filtering out junk data from the eye-tracking stream or working with eye-tracking colliders (AoIs), existing 2D methodics [89] can be used. However, when data are interpreted at 3D projection heatmap/fixation level (*PoF*), the computations have to be executed in a 3D-specific way.

Throughout the paper, data structures are mentioned (from CSV files to *Lists* and *HashTables*), to carry eye-tracking data in and out of a 3D engine. Computer graphics raytracing [90] classes (*Raycast* and *Conecast*) are used as part of the available API. 3D engine-specific components like *Layers* or *Colliders* are utilized. All this is nothing new in graphics programming. However, the value added by this paper lies in the ingenuity of combining these classes to obtain viable 3D eye-tracking concepts or working solutions per methodological arguments laid out in Sections 1 and 2.

4.2. Future Prospects

The provided example algorithms leave room for improvement—to be expanded further to alleviate some of the issues of eye-tracking in 3D space (per Section 2.3). Some other related issues (e.g., visual evaluation and operationalization of space, 4D visualization in 3D space, and external application use) are merely glanced over in the paper. This concluding section is to elaborate on both the subsets of issues.

Evaluating virtual scene visuals can be achieved algorithmically, to prepare and optimize a scene for experimental purposes (Section 3.5). If spatial and visual properties (e.g., visual saliency, scene variety/complexity, or view distance) can be evaluated by image processing and spatial computation algorithms [91] prior to, or in addition to, user eye-tracking feedback, this can validate virtual experimental conditions. There is the problem of specifying fitting eye-tracking object colliders (AoI); since 3D objects acquired by means like photogrammetry [32,92] or procedural 3D asset generation do not have clearly defined

AoI separations, segmenting objects into eye-tracking colliders is an open question. In static 2D imagery, saliency algorithms can be used for machine AoI detection [93,94]. If such algorithms can be used with 3D environments (e.g., by averaging multiple pictures taken from multiple virtual positions/angles), 3D AoI segmentation can be achieved.

Beyond mere passive logging of gazed-upon coordinates, eye-tracking can also be used to trigger interactive elements in virtual scenes. Compared to passive eye-tracking logging, eye-tracking user-controllable interfaces can leave feedback for the user in visual or auditory cues, interface-changing rules, etc. This can range from very simplistic solutions, such as an object lighting up when it is gazed upon, to complex, composite interactions of multiple interfaces or in-scene actors. Such interactions, by their own design, make the scene a dynamic one, thereby increasing the complexity of evaluating user behavior around them (described in Section 2.3). While interactive eye-tracking applications involve user experience and cognitive processing of interactive tasks, which is beyond the scope of the present paper, it is worth emphasizing that eye-tracking need not always be used merely for passive data-acquisition, and that the technology can be tailored to fit interactive needs [95] and scenarios [96].

Presenting multi-dimensional data, or a large quantity of data, is a computer visualization/design challenge [97]. Eye-tracking in VR may demand either four-dimensional or five-dimensional visualizations (i.e., a three-dimensional visualization extended with factor(s) for user/object movement in time and user/object state/property changes). When user camera(s), user-to-user interaction, user-to-object interaction, or object placement/properties change through time, the number of variables and values that require tracking expands and must be managed with operationalization, implementation, logging, rendering, and replaying. The included replay script is only a simple prototype solution with a basic visualization feature. Similarly, the provided heatmap visualization script can become an illegible depiction, should long-enough eye-tracking logs of complex-enough spaces be processed. Existing studies [38] or solutions [31] have also not addressed this issue; the complexity of the issue may need to be solved through software suites with GUI and layered approach to viewing, filtering, and processing different kinds or segments of data. Thus, proposing a classification system of virtual environment behavioral data may be needed; similarly, implementing VR user interfaces to work with viewing and processing these data may also be of value.

Processing and visualizing all the user behavior data in a 3D engine may seem clumsy and computationally intensive on resources, but existing external visualization applications may be considered in balance, with both advantages and disadvantages. The utility of a 3D engine lies in its accessibility to programming language and the API used in the experiment, to the point of accessing object behavior and object components related to that experiment. However, spatial data algorithms such as movement heatmaps or path averages are not implemented in a 3D engine by default. Instead of reimplementing them, they can be visualized by external GIS software [98] (which, in turn, only accepts a graphically and functionally simplified version of an experimental scene).

On a general note, VR scenarios offer high interpretative value, provided the constraints of the platform are taken into account in preparing an experiment and implementing an environment. The implementation and evaluation difficulty of dynamic scenes with many degrees of freedom of movement is balanced by the inherent advantage of VR as a medium, i.e., by its ability to present complex stimuli while maintaining (some) experimental control over the scene and isolating users from the types of unwanted, disruptive, and intervening variables that tend to plague real-world scenarios. As mentioned in a previous overview study [37], experiments in eye-tracking in VR have both advantages and disadvantages compared to traditional eye-tracking research. These manifest at the level of experimental stimuli, experimental tasks, and measured behavior. Freedom in user movement and action can facilitate the emergence of individual differences in cognitive processing of scenes, interaction with objects, user controller use, and their overarching strategies [99,100]. Applied experimental settings of different types, which implement

various, often customized user controllers, can make use of this in simulators for engineering [101], military [102], medical purposes [103], experimental cartography [104], collective learning [105], etc. Furthermore, the principles and the algorithms of eye-tracking in VR can be used in the aforementioned fields to add to the interpretation value of the overall acquired data.

In closing, any implemented corrective eye-tracking and related algorithms must be considered critically. If they are employed for interpreting complex environments, a key question is the extent of such corrections. Is there such a thing as an experimental eye-tracking scene so complex that it is not worth interpreting? And where exactly does that threshold lie? Future research can take this direction and verify the efficiency of corrective eye-tracking algorithms, and possibly also provide newer, improved versions, pushing the efficiency of the corrective solutions further. In parallel, eye-tracking in VR environments re-created from their real counterparts can be compared with eye-tracking in the original (i.e., by comparing eye-tracking in VR with mobile eye-tracking [106] or 2D eye-tracking studies [107]), to test for the ecological validity of the virtual method and visualization.

Author Contributions: Conceptualization, Z.J., S.T. and P.U.; methodology, S.T., P.U. and Z.J.; software, P.U. and O.K.; data curation, P.U.; writing—original draft preparation, P.U. and O.K.; writing—review and editing, Z.J. and S.T.; visualization, P.U. and O.K.; supervision, Č.Š. and S.T.; funding acquisition, Č.Š. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by a grant from the Technology Agency of the Czech Republic No. TL02000103 called "Cognitive psychology and space syntax in virtual environments for agent modeling", and Pavel Ugwitz was also supported by a grant of the Masaryk University No. MUNI/A/1576/2018 called "Complex research of the geographical environment of the planet Earth".

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: This work was supported by the research infrastructure HUME Lab Experimental Humanities Laboratory, Faculty of Arts, Masaryk University. Our thanks also go to proof readers and reviewers.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Gliner, J.A.; Morgan, G.A.; Leech, N.L. *Research Methods in Applied Settings: An Integrated Approach to Design and Analysis*; Routledge: London, UK, 2017.
2. Strube, M.J. Psychometric principles: From physiological data to psychological constructs. In *Principles of Psychophysiology: Physical, Social, and Inferential Elements*; Cacioppo, J.T., Tassinary, L.G., Eds.; Cambridge University Press: Cambridge, UK, 1990; pp. 34–57.
3. Rayner, K. The 35th Sir Frederick Bartlett Lecture: Eye Movements and Attention in Reading, Scene Perception, and Visual Search. *Q. J. Exp. Psychol.* **2009**, *62*, 1457–1506. [CrossRef] [PubMed]
4. Hooft, E.A.J.V.; Born, M.P. Intentional response distortion on personality tests: Using eye-tracking to understand response processes when faking. *J. Appl. Psychol.* **2012**, *97*, 301–316. [CrossRef] [PubMed]
5. Tsianos, N.; Germanakos, P.; Lekkas, Z.; Mourlas, C.; Samaras, G. Eye-tracking users' behavior in relation to cognitive style with inane-learning environment. In Proceedings of the Ninth IEEE International Conference on Advanced Learning Technologies, Riga, Latvia, 15–17 July 2009. [CrossRef]
6. Sasson, N.J.; Elison, J.T. Eye tracking young children with autism. *J. Vis. Exp.* **2012**, *61*, 3675. [CrossRef] [PubMed]
7. Coetzee, R.; Regel, S. Eye movement desensitisation and reprocessing: An update. *Adv. Psychiatr. Treat.* **2015**, *11*, 347–354. [CrossRef]
8. Ekman, P. *Emotion in the Human Face*; Malor Books: San Jose, CA, USA, 2015.
9. Changing Minds. Eyes: Body Language. 2012. Available online: http://changingminds.org/techniques/body/parts_body_language/eyes_body_language.htm (accessed on 31 July 2021).
10. Yarbus, A.L. *Eye Movements and Vision*; Plenum Press: New York, NY, USA, 1967.
11. Falck-Ytter, T. Gaze performance during face-to-face communication: A live eye tracking study of typical children and children with autism. *Res. Autism Spectr. Disord.* **2015**, *17*, 78–85. [CrossRef]

12. Sharma, K.; Giannakos, M.; Dillenbourg, P. Eye-tracking and artificial intelligence to enhance motivation and learning. *Smart Learn. Environ.* **2020**, *7*, 1–19. [CrossRef]
13. Shipman, K. MetaHumans—A look past the Hype. Puget Systems. 2021. Available online: https://www.pugetsystems.com/labs/articles/MetaHumans---a-look-past-the-Hype-2139 (accessed on 31 July 2021).
14. Buswell, G.T. *How People Look at Pictures: A Study of the Psychology and Perception in Art*; University of Chicago Press: Chicago, IL, USA, 1935.
15. Wade, N.J. Pioneers of eye movement research. *i-Perception* **2010**, *1*, 33–68. [CrossRef]
16. Cornsweet, T.N.; Crane, H.D. Accurate Two-Dimensional Eye Tracker Using First and Fourth Purkinje Images. *J. Opt. Soc. Am.* **1973**, *63*, 921–928. [CrossRef]
17. Charlier, J.; Hache, J. New instrument for monitoring eye fixation and pupil size during the visual field examination. *Med. Biol. Eng. Comput.* **1982**, *20*, 23–28. [CrossRef]
18. Van de Kerkhof, M.; Benschop, J.; Banine, V. Lithography for now and the future. *Solid-State Electron.* **2019**, *155*, 20–26. [CrossRef]
19. Safin, R.; Garipova, E.; Lavrenov, R.; Li, H.; Svinin, M.; Magid, E. Hardware and software video encoding comparison. In Proceedings of the 59th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE), Chiang Mai, Thailand, 23–26 September 2020; pp. 924–929. [CrossRef]
20. Cognolato, M.; Atzori, M.; Müller, H. Head-Mounted Eye Gaze Tracking Devices: An Overview of Modern Devices and Recent Advances. *J. Rehabil. Assist. Technol. Eng.* **2018**, *5*, 1–13. [CrossRef] [PubMed]
21. Ewalt, D.M. *Defying Reality: The inside Story of the Virtual Reality Revolution*; Blue Rider Press: New York, NY, USA, 2018.
22. Levy, R.M. The virtual reality revolution: The vision and the reality. In *Virtual Reality—Human Computer Interaction*; Xinxign, T., Ed.; InTech: Šid, Croatia, 2012; pp. 21–36.
23. Zhan, T.; Yin, K.; Xiong, J.; He, Z.; Wu, S.-T. Augmented Reality and Virtual Reality Displays: Perspectives and Challenges. *iScience* **2020**, *23*, 101397. [CrossRef] [PubMed]
24. Portman, M.E.; Natapov, A.; Fisher-Gewirtzman, D. To Go Where No Man Has Gone Before: Virtual Reality in Architecture, Landscape Architecture and Environmental Planning. *Comput. Environ. Urban Syst.* **2015**, *54*, 376–384. [CrossRef]
25. Griliopoulos, D. The Virtual Reality of Virtual Worlds—Improbable. 2017. Available online: https://www.improbable.io/blog/the-virtual-reality-of-virtual-worlds (accessed on 21 September 2021).
26. Mayor, J.; Raya, L.; Sanchez, A. A Comparative Study of Virtual Reality Methods of Interaction and Locomotion Based on Presence, Cybersickness and Usability. *IEEE Trans. Emerg. Topics Comput.* **2021**, *9*, 1542–1553. [CrossRef]
27. Pan, X.; Hamilton, A.F. Why and How to Use Virtual Reality to Study Human Social Interaction: The Challenges of Exploring a New Research Landscape. *Br. J. Psychol.* **2018**, *109*, 395–417. [CrossRef]
28. Sussman, A.; Hollander, J.B. *Cognitive Architecture: Designing for How We Respond to the Built Environment*; Routledge: London, UK, 2015.
29. Kiefer, P.; Giannopoulos, I.; Raubal, M.; Duchowski, A. Eye Tracking for Spatial Research: Cognition, Computation, Challenges. *Spat. Cogn. Comput.* **2017**, *17*, 1–19. [CrossRef]
30. Patney, A.; Kim, J.; Salvi, M.; Kaplanyan, A.; Wyman, C.; Benty, N.; Lefohn, A.; Luebke, D. Perceptually-Based Foveated Virtual Reality. In Proceedings of the ACM SIGGRAPH 2016 Emerging Technologies, Anaheim, CA, USA, 24–28 July 2016.
31. Tobii. Tobii XR SDK, 2021a. Available online: https://vr.tobii.com/sdk/ (accessed on 25 August 2021).
32. Statham, N. Use of Photogrammetry in Video Games: A Historical Overview. *Games Cult.* **2020**, *15*, 289–307. [CrossRef]
33. Yan, J.; Kensek, K.; Konis, K.; Noble, D. CFD Visualization in a Virtual Reality Environment Using Building Information Modeling Tools. *Buildings* **2020**, *10*, 229. [CrossRef]
34. Zhao, J.; Wallgrün, J.O.; LaFemina, P.C.; Normandeau, J.; Klippel, A. Harnessing the Power of Immersive Virtual Reality—Visualization and Analysis of 3D Earth Science Data Sets. *Geo-Spat. Inf. Sci.* **2019**, *22*, 237–250. [CrossRef]
35. Krejtz, K.; Duchowski, A.T.; Niedzielska, A.; Biele, C.; Krejtz, I. Eye Tracking Cognitive Load Using Pupil Diameter and Microsaccades with Fixed Gaze. *PLoS ONE* **2018**, *13*, e0203629. [CrossRef]
36. Duchowski, A. *Eye Tracking Methodology: Theory and Practice*, 3rd ed.; Springer International Publishing: Cham, Switzerland, 2017.
37. Lappi, O. Eye Tracking in the Wild: The Good, the Bad and the Ugly. *J. Eye Mov. Res.* **2015**, *8*, 1–21. [CrossRef]
38. Clay, V.; König, P.; König, S. Eye Tracking in Virtual Reality. *J. Eye Mov. Res.* **2019**, *12*. [CrossRef] [PubMed]
39. SR Research. EyeLink 1000 Plus Technical Specifications. 2017. Available online: https://www.sr-research.com/wp-content/uploads/2017/11/eyelink-1000-plus-specifications.pdf (accessed on 18 October 2021).
40. Sipatchin, A.; Wahl, S.; Rifai, K. Eye-Tracking for Clinical Ophthalmology with Virtual Reality (VR): A Case Study of the HTC Vive pro Eye's Usability. *Healthcare* **2021**, *9*, 180. [CrossRef] [PubMed]
41. VRgineers. XTAL Technical Specification. 2021. Available online: https://vrgineers.com/xtal/technical-specification/ (accessed on 25 October 2021).
42. Varjo. Getting Started with Varjo Plugin for Unity. 2021. Available online: https://developer.varjo.com/docs/v2.1.0/unity/getting-started-with-varjo-plugin-for-unity (accessed on 25 October 2021).
43. Khronos Group. The Open XR Specification. 2021. Available online: https://www.khronos.org/registry/OpenXR/specs/1.0/pdf/xrspec.pdf (accessed on 25 August 2021).
44. Hornsey, R.L.; Hibbard, P.B. Contributions of Pictorial and Binocular Cues to the Perception of Distance in Virtual Reality. *Virtual Real.* **2021**, *25*, 1087–1103. [CrossRef]

45. Chandrasekera, T.; Fernando, K.; Puig, L. Effect of Degrees of Freedom on the Sense of Presence Generated by Virtual Reality (VR) Head-Mounted Display Systems: A Case Study on the Use of VR in Early Design Studios. *J. Educ. Technol. Syst.* **2019**, *47*, 513–522. [CrossRef]
46. Kitson, A.; Hashemian, A.M.; Stepanova, E.R.; Kruijff, E.; Riecke, B.E. Comparing leaning-based motion cueing interfaces for virtual reality locomotion. In Proceedings of the 2017 IEEE Symposium on 3D User Interfaces, Los Angeles, CA, USA, 18–19 March 2017; pp. 73–82.
47. Keil, J.; Edler, D.; O'Meara, D.; Korte, A.; Dickmann, F. Effects of Virtual Reality Locomotion Techniques on Distance Estimations. *ISPRS Int. J. Geo-Inf.* **2021**, *10*, 150. [CrossRef]
48. Berger, M. Exploring and Transforming Spaces through High-Dimensional Gestural Interactions. *Adv. Cartogr. GIScience ICA* **2021**, *3*, 1–8. [CrossRef]
49. Adobe Inc. Medium. Top 3D Sculpting Tools for Virtual Reality Authoring. 2021. Available online: https://www.adobe.com/products/medium.html (accessed on 23 December 2021).
50. Pietroszek, K. Raycasting in Virtual Reality. In *Encyclopedia of Computer Graphics and Games*; Lee, N., Ed.; Springer: Cham, Switzerland, 2018; pp. 1–3.
51. Ihemedu-Steinke, Q.C.; Erbach, R.; Halady, P.; Meixner, G.; Weber, M. Virtual Reality Driving Simulator Based on Head-Mounted Displays. In *Automotive User Interfaces*; Meixner, G., Müller, C., Eds.; Springer: Berlin/Heidelberg, Germany, 2017; pp. 401–428.
52. Gillies, M. What Is Movement Interaction in Virtual Reality For? In Proceedings of the 3rd International Symposium on Movement and Computing, Thessaloniki, Greece, 5–6 July 2016; pp. 1–4.
53. Unity Technologies. Character Controller. 2021. Available online: https://docs.unity3d.com/Manual/class-CharacterController.html (accessed on 25 August 2021).
54. Motamedi, A.; Wang, Z.; Yabuki, N.; Fukuda, T.; Michikawa, T. Signage Visibility Analysis and Optimization System Using BIM-Enabled Virtual Reality (VR) Environments. *Adv. Eng. Inform.* **2017**, *32*, 248–262. [CrossRef]
55. Ugwitz, P.; Stachoň, Z.; Pospíšil, P. Rendering a series of 3D dynamic visualizations in (geographic) experimental tasks. In Proceedings of the 8th International Conference on Cartography and GIS, Nessebar, Bulgaria, 20–25 June 2020; pp. 628–634.
56. Steffen, J.H.; Gaskin, J.E.; Meservy, T.O.; Jenkins, J.L.; Wolman, I. Framework of Affordances for Virtual Reality and Augmented Reality. *J. Manag. Inf. Syst.* **2019**, *36*, 683–729. [CrossRef]
57. Pettersson, J.; Albo, A.; Eriksson, J.; Larsson, P.; Falkman, K.W.; Falkman, P. Cognitive Ability Evaluation Using Virtual Reality and Eye Tracking. In Proceedings of the 2018 IEEE International Conference on Computational Intelligence and Virtual Environments for Measurement Systems and Applications (CIVEMSA), Ottawa, ON, Canada, 12–13 June 2018.
58. Tobii. Eye-Tracking Analytics. 2021. Available online: https://vr.tobii.com/sdk/learn/analytics/ (accessed on 18 October 2021).
59. Halladay, K. *Practical Shader Development: Vertex and Fragment Shaders for Game Developers*; Apress: New York, NY, USA, 2019.
60. WorldViz Inc. Sightlab VR. 2021. Available online: https://www.worldviz.com/virtual-reality-eye-tracking-for-research-solutions (accessed on 26 August 2021).
61. Rönngren, D. OpenXR Announces the First Extension for Eye Tracking. 2020. Available online: https://blog.tobii.com/openxr-announces-the-first-extension-for-eye-tracking (accessed on 31 July 2021).
62. Tobii. Get Started Developing with the HTC VIVE Pro Eye Using Tobii XR SDK. 2021. Available online: https://vr.tobii.com/sdk/develop/unity/getting-started/vive-pro-eye/ (accessed on 18 October 2021).
63. Epic Games. The Most Powerful Real-Time 3D Creation Tool—Unreal Engine. 2021. Available online: https://www.unrealengine.com/ (accessed on 18 October 2021).
64. Unity Technologies. Unity Real-Time Development Platform. 2021. Available online: https://unity.com/releases/release-overview (accessed on 18 October 2021).
65. Bergsman Lind, T. Quixel Joins Forces with Epic Games. 2019. Available online: https://quixel.com/blog/2019/11/12/quixel-joins-forces-with-epic-games (accessed on 26 September 2021).
66. Beardsall, R. Unreal Engine 5 And Nanite Virtualized Geometry—What Does It Mean For Content Creators? 2020. Available online: https://medium.com/xrlo-extended-reality-lowdown/unreal-engine-5-and-nanite-virtualized-geometry-what-does-it-mean-for-content-creators-b4106accd306 (accessed on 26 September 2021).
67. Romero, M.F.; Sewell, B. *Blueprints Visual Scripting for Unreal Engine: The Faster Way to Build Games Using UE4 Blueprints*; Packt Publishing: Birmingham, UK, 2019.
68. Ugwitz, P.; Herman, L.; Stachoň, Z. 3D Visualization of Historical Buildings: Methods, Their Utility in the Tourism Industry and Beyond. *Regionální Rozvoj Mezi Teorií a Praxí* **2021**, *1*, 43–62.
69. Unity Technologies. Post-Processing. 2021. Available online: https://docs.unity3d.com/Manual/PostProcessingOverview.html (accessed on 25 August 2021).
70. Volk, R.; Stengel, J.; Schultmann, F. Building Information Modeling (BIM) for existing buildings—Literature review and future needs. *Autom. Constr.* **2014**, *38*, 109–127. [CrossRef]
71. Gu, N.; London, K. Understanding and facilitating BIM adoption in the AEC industry. *Autom. Constr.* **2010**, *19*, 988–999. [CrossRef]
72. Tridify Ltd. Tridify Convert. 2021. Available online: https://www.tridify.com/ (accessed on 25 August 2021).
73. Tridify Ltd. Tridify BIM Tools. 2021. Available online: https://assetstore.unity.com/packages/tools/utilities/tridify-bim-tools-125779 (accessed on 25 August 2021).

74. Unity Technologies. ProBuilder. 2021. Available online: https://unity.com/features/probuilder (accessed on 25 August 2021).
75. Blender Foundation. Blender—A 3D Modelling and Rendering Package. 2021. Available online: https://www.blender.org/ (accessed on 25 August 2021).
76. Luebke, D.P. *Level of Detail for 3D Graphics*; Morgan Kaufmann: Amsterdam, The Netherlands, 2003.
77. Aversa, D.; Dickinson, C. *Unity Game Optimization: Enhance and Extend the Performance of All Aspects of Your Unity Games*, 3rd ed.; Packt Publishing: Birmingham, UK, 2019.
78. Austin, J. Fix your (Unity) Timestep. 2019. Available online: https://johnaustin.io/articles/2019/fix-your-unity-timestep (accessed on 29 September 2021).
79. Farrell, J.E. Fitting Physical Screen Parameters to the Human Eye. In *The Man-Machine Interface*; Macmillan: Basingstoke, UK, 1991; pp. 7–23.
80. Lengyel, E. *Foundations of Game Engine Development*; Terathon Software LLC: Lincoln, CA, USA, 2016; Volume 1.
81. Guenter, B.; Finch, M.; Drucker, S.; Tan, D.; Snyder, J. Foveated 3D Graphics. *ACM Trans. Graph.* **2012**, *31*, 1–10. [CrossRef]
82. Nystrom, R. Game Programming Patterns. Genever Benning. 2014, pp. 321–353. Available online: https://gameprogrammingpatterns.com/ (accessed on 16 January 2022).
83. Kurzhals, K.; Weiskopf, D. Space-Time Visual Analytics of Eye-Tracking Data for Dynamic Stimuli. *IEEE Trans. Vis. Comput. Graph.* **2013**, *19*, 2129–2138. [CrossRef]
84. Unreal Engine. Decals. An Overview of the Decals Example Level. 2022. Available online: https://docs.unrealengine.com/4.27/en-US/Resources/ContentExamples/Decals/ (accessed on 5 January 2022).
85. Llanes-Jurado, J.; Marín-Morales, J.; Guixeres, J.; Alcañiz, M. Development and Calibration of an Eye-Tracking Fixation Identification Algorithm for Immersive Virtual Reality. *Sensors* **2020**, *20*, 4956. [CrossRef]
86. Saleem, W.; Belyaev, A.; Wang, D.; Seidel, H.-P. On Visual Complexity of 3D Shapes. *Comput. Graph.* **2011**, *35*, 580–585. [CrossRef]
87. Bulthoff, H.H.; Edelman, S.Y.; Tarr, M.J. How Are Three-Dimensional Objects Represented in the Brain? *Cereb. Cortex* **1995**, *5*, 247–260. [CrossRef]
88. Ikonen, T. Lightweight CSV Reader for Unity. 2014. Available online: https://bravenewmethod.com/2014/09/13/lightweight-csv-reader-for-unity/ (accessed on 5 January 2022).
89. SensoMotoric Instruments. Event Detection. In *BeGaze Manual Version 3.4*; SensoMotoric Instruments: Teltow, Germany, 2014; pp. 315–325.
90. Hofmann, G.R. Who Invented Ray Tracing? *Vis. Comput.* **1990**, *6*, 120–124. [CrossRef]
91. Yamu, C.; van Nes, A.; Garau, C. Bill Hillier's Legacy: Space Syntax—A Synopsis of Basic Concepts, Measures, and Empirical Application. *Sustainability* **2021**, *13*, 3394. [CrossRef]
92. Lachambre, S.; Lagarde, S.; Jover, C. *Unity Photogrammetry Workflow*; Unity Technologies: San Francisco, CA, USA, 2017.
93. Fuhl, W.; Kuebler, T.; Santini, T.; Kasneci, E. Automatic Generation of Saliency-Based Areas of Interest for the Visualization and Analysis of Eye-Tracking Data. In Proceedings of the Conference on Vision, Modeling, and Visualization, Stuttgart, Germany, 10–12 October 2018; pp. 47–54.
94. John, B.; Raiturkar, P.; Le Meur, O.; Jain, E. A Benchmark of Four Methods for Generating 360 Saliency Maps from Eye Tracking Data. *Int. J. Semant. Comput.* **2019**, *13*, 329–341. [CrossRef]
95. Soler-Dominguez, J.L.; Camba, J.D.; Contero, M.; Alcañiz, M. A Proposal for the Selection of Eye-Tracking Metrics for the Implementation of Adaptive Gameplay in Virtual Reality Based Games. In *International Conference on Virtual, Augmented and Mixed Reality*; Springer: Cham, Switzerland, 2017; pp. 369–380.
96. Ugwitz, P.; Šašinková, A.; Šašinka, Č.; Stachoň, Z.; Juřík, V. Toggle Toolkit: A Tool for Conducting Experiments in Unity Virtual Environments. *Behav. Res. Methods* **2021**, *53*, 1581–1591. [CrossRef]
97. Ware, C. *Information Visualization: Perception for Design*; Morgan Kaufmann: Burlington, MA, USA, 2020.
98. ESRI. ArcGIS 3D Analyst. 2022. Available online: https://www.esri.com/en-us/arcgis/products/arcgis-3d-analyst/overview (accessed on 5 January 2022).
99. Parsons, T.D.; Rizzo, A.A. Initial Validation of a Virtual Environment for Assessment of Memory Functioning: Virtual Reality Cognitive Performance Assessment Test. *CyberPsychology Behav.* **2008**, *11*, 17–25. [CrossRef]
100. Hanus, A.; Hoover, M.; Lim, A.; Miller, J. A Collaborative Virtual Reality Escape Room with Passive Haptics. In Proceedings of the 2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR), Osaka, Japan, 23–27 March 2019; pp. 1413–1414. [CrossRef]
101. Delgado, J.M.D.; Oyedele, L.; Demian, P.; Beach, T. A Research Agenda for Augmented and Virtual Reality in Architecture, Engineering and Construction. *Adv. Eng. Inform.* **2020**, *45*, 101122. [CrossRef]
102. Lele, A. Virtual Reality and Its Military Utility. *J. Ambient. Intell. Humaniz. Comput.* **2011**, *4*, 17–26. [CrossRef]
103. Visser, H.; Watson, M.O.; Salvado, O.; Passenger, J.D. Progress in Virtual Reality Simulators for Surgical Training and Certification. *Med. J. Aust.* **2011**, *194*, S38–S40. [CrossRef]
104. Herman, L.; Popelka, S.; Hejlová, V. Eye-Tracking Analysis of Interactive 3D Geovisualization. *J. Eye Mov. Res.* **2017**, 10. [CrossRef]
105. Šašinka, Č.; Stachoň, Z.; Sedlák, M.; Chmelík, J.; Herman, L.; Kubíček, P.; Šašinková, A.; Doležal, M.; Tejkl, H.; Urbánek, T.; et al. Collaborative Immersive Virtual Environments for Education in Geography. *ISPRS Int. J. Geo-Inf.* **2018**, *8*, 3. [CrossRef]

106. Šašinková, A.; Ugwitz, P. Cross-Cultural Differences in the Processing of Complex Virtual 3D Scenes. In Proceedings of the 32nd International Congress of Psychology, Prague, Czech Republic, 18–23 July 2021.
107. Čeněk, J.; Tsai, J.-L.; Šašinka, Č. Cultural Variations in Global and Local Attention and Eye-Movement Patterns during the Perception of Complex Visual Scenes: Comparison of Czech and Taiwanese University Students. *PLoS ONE* **2020**, *15*, e0242501. [CrossRef]

Annex 5: Related publications and sources

Papers in impact journals

Snopková, D. Ugwitz, P., Stachoň, Z., Hladík, J., Juřík, V., Kvarda, O., Kubíček, P. (2022). Retracing evacuation strategy: A virtual reality game-based investigation into the influence of building's spatial configuration in an emergency. *Spatial Cognition & Computation* (22), pp. 30-50. https://doi.org/10.1080/13875868.2021.1913497

Conference papers

Ugwitz, P., Stachoň, Z., Pospíšil, P. (2020). Rendering a Series of 3D Dynamic Visualizations in (Geographic) Experimental Tasks. *8th International Conference on Cartography & GIS: Proceedings(1), Nesebar.*

Ugwitz, P., Stachoň, Z., Herman, L. (2019) Prototyp nástroje TrailScript pro evaluaci chování ve virtuálním prostředí. *Kognícia a umelý Kognícia a umelý život 2019, Bratislava.*

Ugwitz, P., Stachoň, Z. (2018). Visual Elements in Virtual Environments: Finding Parallels Between CGI Visualizations and Human Perception. *7th International Conference on Cartography & GIS: Proceedings(1), Sozopol.*

Ugwitz, P., Juřík, V., Šašinka, Č. (2017). Visual cognitive styles in virtual environments: Constructing and evaluating adequate task. *Kognicia a umelý život 2017, Trenčianské Teplice.*

Others

Ugwitz, P. (2020). Využití eye-trackingu ve virtuální realitě [Utilizing Eye-Tracking in Context of Virtual Reality]. *Master's thesis. Masaryk University, Faculty of Informatics.*

Ugwitz, P., Stachoň, Z., Šašinka, Č. (2019). Eye-Tracking in VR: From Heatmaps to Object Aols. *14th International Conference on Spatial Information Theory, Regensburg.*

Unpublished research

Chocholáčková, M., Juřík, V., Ugwitz, P., Jelínek, M. (2022). *Context-dependent memory recall in computer-generated virtual environments* [Unpublished manuscript]. Department of Psychology, Masaryk University.

Juřík, V., Ugwitz, P., Žváčková, Š. (2022). *The influence of 2D and 3D visualization on contextual memory recollection* [Unpublished manuscript]. Department of Psychology, Masaryk University.

Šašinková, A., Čeněk, J., Ugwitz, P., Tsai, J., Giannopoulos, J., Lacko, D., Stachoň, Z., Šašinka, Č. (2022). *Variations of eye-movement patterns during the perception of 3D complex visual scenes: Comparison of five cultural samples* [Unpublished manuscript]. Department of Information and Library Studies, Masaryk University.

Source codes

The source codes implementing the technical solutions of the second, third and fourth papers (see Annex 2-4) are mentioned in each paper. However, since some of the programming solutions overlap, and since source codes of some smaller publications are also included, let us mention these below exhaustively:

- The repository at https://github.com/kim-dotcom/
- The repository at https://github.com/VGE-lab-MUNI/ (Virtual Geographic Environments laboratory at Faculty of Science, Masaryk University; a backup of the repository above)

The repositories are documented on GitHub for usability (see projects' *readme.md*); the code itself comments on implementation and programming logic. Most of the codebase is written in C#, native to the Unity engine (unless stated otherwise). Regarding what does what, the listing follows:

- /toggleScript/PathScript.cs is a universal user behaviour logger, as used in Annex 2.
- /pathScript/ is a 2D visualization script (Processing) that takes logger data and saves them into PNG outputs – used in Annex 2 and (Ugwitz, Herman, Stachoň 2019).
- /toggleScript/ provides interactive functionality, as used in Annex 3.
- /vr-et/ provides example VR eye-tracking algorithms, as described or demonstrated in Annex 4. Scripts for cleansing the acquired eye-tracking data (Processing) are also included – see /vr-et/verification/
- /etc/ includes auxiliary Unity scripts to be used in any Unity project.

Annex 6: Curriculum Vitae

Pavel Ugwitz
Brno, Rekreační 150/22
ugwitz.pavel@mail.muni.cz

Education

Masaryk University, Faculty of Science, 2017 - *ongoing*
Geoinformatics and remote surveying (doctoral)

Masaryk University, Faculty of Informatics, 2017 - 2021
Applied informatics (masters)
Thesis: *Utilizing Eye-Tracking in Context of Virtual Reality*

Masaryk University, Faculty of Arts, 2011 - 2017
Psychology (masters)
Thesis: *Experimental Possibilities in Virtual Reality*

Masaryk University, Faculty of Informatics, 2010 - 2013
Applied informatics (bachelors)
Thesis: *Level Design*

Projects, work experience

Cognitive psychology and spatial syntax in virtual environments for agent models
TAČR TL02000103, 1/2020-12/2022

Geografický výzkum dynamiky přírodních a společen. prostorových procesů.
MUNI/A/1570/2020, 1/2021–12/2021

The influence of socio-cultural factors and writing system on perception and cognition of complex visual stimuli.
GACŘ GC19-09265J, 1/2019–1/2021

Výzkum proměn geografických procesů a vztahů v prostoru a čase.
MUNI/A/1356/2019, 1/2020–12/2020

Komplexní výzkum geografického prostředí planety Země.
MUNI/A/1576/2018, 1/2019–12/2019

In Brno, on June 6, 2022

