Electronic, didactic and innovative platform for learning based on multimedia assets

e-DIPLOMA

Funded by the European Union

D.3.2 Visualization/Interaction Demonstrators

Version No. 1.5
06 March 2026

HISTORY OF CHANGES			
Version*	**Publication date**	**Beneficiaries**	**Changes**
V1.0	09.09.2024	TUD	▪ Initial version of Deliverable Owner
V1.2	25.09.2024	TUD	▪ Added glossary, adjusted numbering of the sections, and extended explanations ▪ Added full technical description outside the main document, added a demonstrator that allows users to switch between modes ▪ Added an option to choose between quality and performance modes for Demonstrators 3 and 4. Added installation instructions and system requirements. ▪ Added a simulation mode to the demonstrators. ▪ Adjusted formatting of the text, updated the content overview, and addressed minor spelling mistakes.
V1.3	29.09.2024	TUD	▪ Formatting changes
V1.4	30.09.2024	TUD	▪ Final Version submitted
V1.5	06.03.2026	TUD	Revised Final Version includes: ● Added license type (MIT) ● Added technical supplements for all demonstrators and included links to all resources in the section Resources ● Extended the “Relation to other documents” section to

			link with deliverables that were released after the original version of this document. In this same section, we included a discussion of why these demonstrators were chosen and how these demonstrators evolved into Prototype 3. • Adjusted formatting

(*) According to the section “Review and Submission of Deliverables” of the Project Handbook

1. Technical References

Project Number	101061424
Project Acronym	e-diploma
Project Title	Electronic, Didactic and Innovative Platform for Learning based On Multimedia Assets
Granting Authority	European Research Executive Agency (REA)
Call	HORIZON-CL2-2021-TRANSFORMATIONS-01
Topic	HORIZON-CL2-2021-TRANSFORMATIONS-01-05
Type of the Action	HORIZON Research and Innovation Actions
Duration	1 September 2022 – 31 October 2025 (36 months)
Entry into force of the Grant	1 September 2022
Project Coordinator	Inmaculada Remolar Quintana

Deliverable No.	D3.2: Demonstrator
Work Package	WP3: Piloting and testing of Innovation procedures and technology enhancement
Task	Visualization and interaction techniques will be included in a set of demonstrators to be integrated in the e-DIPLOMA platform.
Dissemination level*	PU- Public
Type of license:	MIT
Lead beneficiary	Technische Universiteit Delft (TU Delft)
PIC of the Lead beneficiary	999977366
Contributing beneficiary/ies	
PIC of the Contributing beneficiary/ies	

Author(s)	▪ Elmar Eisemann ▪ Ricardo Marroquim
Contributor(s)	
Due date of deliverable	30.09.2024
Actual submission date	30.09.2024
Resubmission date	06.03.2026

2. Table of Contents

3. Introduction

3.1. Executive Summary

All executables of the demonstrators and the technical supplements are available via:

https://surfdrive.surf.nl/s/zNCHwjZft53fqj2

This document gives a brief overview of this deliverable, which consists of 4 demonstrators:

A demonstrator of different visualization methods (Demo1-Visualization.zip)

A demonstrator of a user study, testing the effectiveness of VR abstraction (Demo2-Memory.zip)

A demonstrator letting a user experience various navigation methods (Demo3-Navigation.zip)

A demonstrator letting a user experience various interaction methods (Demo4-Interaction.zip)

Please note that demonstrators 3 and 4 allow users to choose a trade-off between performance and quality. Depending on the hardware that is running the demonstrators, a suitable version can be chosen, which leads to a visual downgrade but a fully functioning experience (see instructions in Section 3.6).

As outlined in the proposal, the demonstrators show existing techniques and novel visualization solutions. The latter is of particular interest, as our hypothesis is that suitable rendering techniques can be beneficial for memorization, recognition and, thus, task effectiveness. In consequence, novel illustration methods could improve learning in future VR applications. To our knowledge, this is the first attempt made to increase memorization and recognition in a VR context by means of changing the illustration method.

The demonstrators have been partially evaluated to gain insights into the effectiveness. Some of these preliminary results are reported where applicable, but they are technically not required for the deliverable. The deliverable was intended to be demonstrators that implement several techniques in a simple setting to show various visualization, interaction, and navigation options. The next period will focus on evaluations in the context of the prototype testing.

3.2. Relation to Other Project Documents

All demonstrators have been adapted and included in Prototype 3, as described in Deliverable 4.3 (Course Prototypes). The focus of each demonstrator was further refined after discussions and co-design sessions.

The selection criteria of these demonstrators were mostly based on the topics to be covered by the course in Prototype 3, which is about teaching Virtual Reality. The demonstrators cover three fundamental aspects of Virtual Reality, and each one served as a further testing ground for refinement.

That is why each topic (visualization, navigation and interaction) had its own preliminary user-study. The preliminary user studies allowed us to better define how they would be adapted and tested within Prototype 3. Some usability issues were already resolved from the preliminary studies, and further issues were analysed in Deliverable 5.3 (System usability and validation report and ergonomics of evaluation measures).

In Deliverable 4.3 (Course Prototypes), the integration of the demonstrators into the Prototype is further described in the following sections:

- Demonstrator 4 (interaction) is further described in Section P3M1 3.2 of deliverable 4.3
- Demonstrator 3 (navigation) is further described in Section P3M1 3.3 of deliverable 4.3
- Demonstrator 2 (visualization user-study) is further described in Section P3M1 3.4 of deliverable 4.3
- Demonstrator 1 (stylization technique) was used as render style for the entire Prototype 3 as described in Section P3M1 5.2 of deliverable 4.3

All demonstrators include associated documents with more detailed descriptions of the technical aspects. The list of associated files is provided in Section 3.7.

3.3. Abbreviation List

Among the acronyms used most often in the present document are, in alphabetical order, the following:

AI: Artificial Intelligence

API: Application Programming Interface

AR: Augmented Reality

GDPR: General Data Protection Regulation

GGX: Shading Model ("ground glass unknown") developed by Walter et al. 2007

GPU: Graphics Processing Unit (often used exchangeably: graphics hardware, graphics card)

HMD: head-mounted display, typically refers to a VR or AR device.

SSAO: Screen-Space Ambient Occlusion

UI: (Graphical) User Interface

URP: Universal Render Pipeline

VM: Virtual Machine

VR: Virtual Reality

VSync: Vertical Synchronization

XDS: XR Device Simulator

3.4. Glossary

Anti-aliasing:	The process of removing aliasing, which can be achieved with a denser sampling pattern or by reusing samples over time (also referred to as temporal anti-aliasing).
Aliasing:	Two signals might appear identical when sampled coarsely. In graphics, it often refers to artifacts that make individual pixels easy to distinguish and is often caused by insufficient sampling of the projected geometry in these pixels.

Ambient Occlusion:	An approximation of global illumination, based on how much light can statistically reach a point, assuming a uniform illumination from all directions.
Binocular Rivalry:	A mismatch between the perceived image content by the left and the right eye.
Bump mapping:	A texture mapping technique that encodes in each texture pixel a height variation. This height change is then translated into a local normal vector that is used as in normal mapping.
C#:	A programming language
Contour:	A contour is the location on a surface where its normal is orthogonal to the view vector.
Feature Line:	A line on a surface that can be mathematically described by an equation.
Graphics Pipeline:	Typically, it refers to the rasterization process executed on graphics hardware, which transforms virtual object descriptions (typically triangles) into an image.
Image buffer:	A part of memory, typically residing on the GPU, encoding an image, which can be converted into a texture.
Image space:	Working on an image representation of the geometry in a scene
Normal:	A normal at a location P of a surface is the unit vector that is orthogonal to the surface at P underlying surface of
Normal mapping:	A texture mapping technique that encodes in each texture pixel a normal, which is then used to modulate the surface's shading response. Hereby, the illusion of the presence of small-scale geometry can be created without paying the computational costs needed for a fully geometric representation.
Object space:	The actual geometric representation
Rasterization:	The process of transforming projected triangles into pixels.
Rendering:	The process of generating an image
Render scale:	The ratio between calculated and existing pixels on a display. In practice, the lower resolution image is then remapped to fit the target resolution using interpolation techniques.
Reprojection:	Involves unprojecting projected geometry, which means obtaining the original 3D coordinates of a triangle before projection, and then projecting this triangle into a second camera view. For example, taking a pixel in the left eye view and moving it to the corresponding location in the right eye view.
Screen space:	Usually relates to the space after geometry has undergone projection in the

	graphics pipeline.
Shader:	A program targeting the GPU
Shading model:	A computational model to simulate light interaction on a surface
Shading cues:	The intensity gradient induced by a change of orientation of a surface with respect to the light source.
Spatial Coherency:	Consistency in proximity – often used in the context of neighbouring pixels.
Stereo Consistency:	Absence of binocular rivalry.
Temporal Coherency:	Consistent representation over time.
Texture:	An image that resides on the GPU and that can be involved in shader computations. It is often used to add small-scale variations to a surface.
Unity:	A widespread game engine with support for AR/VR applications in C#
URP:	Universal Render Pipeline, one of the choices within the Unity engine for a graphics pipeline. The URP is the most advanced choice for VR applications, giving access to advanced graphics card features.
VSync:	An activated VSync ensures that the display of images is synchronized with the moment, when the display is processing the pixels on the top of the screen.
View vector:	The view vector is the connection between a location and the camera center. In stereo rendering, as required by VR and AR, each camera defines a view vector for a given surface location.

3.5. Reference Documents

See References included at the end of this document.

3.6. System Requirements and Use of Software

All demonstrators rely on the Unity Game Engine and are compiled for Windows. We tested the software on various machines with the main platform being an Intel Core i7-12700K, NVIDIA GeForce RTX 3090, and 32 GB of DDR5-4400 RAM. The demonstrators are easy to install – it is sufficient to unzip the folders and then launch the executable (other than the UnityCrashHandler64.exe) in the main directory of the unzipped folder. Demonstrator 3 and 4 also offer launch scripts in the form of .bat files that enable different customizations of the application and can be launched via a double click. They add parameters to the software to adjust visual style, rendering quality, and level of access. Three default configurations are provided for low, mid, and high performance hardware, and the .bat files are named accordingly, which will impact the appearance of the application.

The software relies on OpenXR, which makes it widely compatible – we tested various VR headsets, including Quest 2, 3, HTC Vive and the HP Omnicept G2. The demonstrators do not even strictly require a VR system to run, as they will then switch to simulator mode, where the VR system is emulated on a

standard display. Of course, the simulation cannot provide the actual experience of an immersive VR system and the use of an actual headset is preferred.

3.7. Resources

Technical descriptions of the demonstrators are available as scientific papers. The papers allow for more in-depth descriptions and reproducibility. They also include more details on the preliminary user studies. In the case of Demonstrator 1 and 2, the final published paper also extended the user study realized for Prototype 3.

However, due to restrictions in Unity's EULA, the source code for Demonstrators 3 and 4 cannot be published because they include assets from the Unity Asset Store (even if they are free). For these two demonstrators, only executables are made available.

Nevertheless, Demonstrator 1 and 2 (regarding visualization) are the most technically challenging ones, and the source code is publicly available. For Demonstrators 3 and 4, there are no specific technical challenges that would hinder the application's reproducibility.

All available source codes are under the MIT license. This is required because the applications are developed in Unity.

All executables and technical supplements are publicly available at:

https://surfdrive.surf.nl/s/zNCHwjZft53fqj2

More information about each demonstrator is available below. With the provided supplementary files, all demonstrators should be reproducible.

Stylisation demonstrators 1 and 2

The complete source code is available under the MIT license on GitHub. https://github.com/amirzaidi/ieeevr

A technical supplement is provided in the public folder. This supplement describes the technique as implemented in the demonstrator. An improved version was submitted and accepted for publication at IEEE VR'26, but has not yet been published. Once it is published, the link to the paper will be available on the same GitHub page as above. The supplement (and to be published paper) includes all technical details for Demonstrator 1 and details of a refined user-study of Demonstrator 2.

Interaction demonstrator

Link to paper: https://dl.acm.org/doi/10.1007/978-3-031-78269-5_39 (also available in the public folder).

Source: Cannot be made public due to Kindergarten Interior Unity Asset. Executable available at https://surfdrive.surf.nl/s/zNCHwjZft53fqj2

Navigation demonstrator

Paper: A technical supplement is provided in the public folder.

Source: Cannot be public due to Polygorn Farm Unity Asset. Executable available at https://surfdrive.surf.nl/s/zNCHwjZft53fqj2

Final Prototype

Source: Cannot be public due to the inclusion of many Unity Assets. However, it is available on request at https://github.com/e-DIPLOMA/Prototype-3

To access the repository, users should email r.marroquim@tudelft.nl with their GitHub username and the reason for requesting access.

The build (executable) is available at https://surfdrive.surf.nl/files/index.php/s/34ITipFVsABwVKk

4. Visualization Demonstrators

This part relates to Demonstrators 1 and 2, as outlined in the executive summary. After a brief overview of the underlying research goals, we present a preliminary study (Section 4.2) that proves the potential effectiveness of abstract rendering. We then discuss the algorithm (Section 4.3), starting with a brief outline of the related work, before addressing methodology and results.

Illustrative visualization techniques have a wide use in education and art. In the context of image abstraction, its potential for increasing memorisation and recognition has been demonstrated, which motivates its use in scientific illustrations, construction manuals, and partially medical and mathematical illustrations. While the power of such representations has been partially explored in a 2D context, the effectiveness of such methods has not been confirmed in a virtual reality or augmented reality setting. We first present a preliminary study (Section 4.2) to investigate this effect. One demonstrator reproduces the setup of this first preliminary study (Demonstrator 2 - Memory). It uses various visualization techniques, including our novel solution. Our novel algorithm focuses on line-art illustrations, which are a rendering stylisation using curves as the primary method to convey a scene through the accentuation of object shapes (Figure 1). For a direct comparison between different methods, Demonstrator 1 can be used, where the techniques can be switched on the fly (as described in the README.txt file).

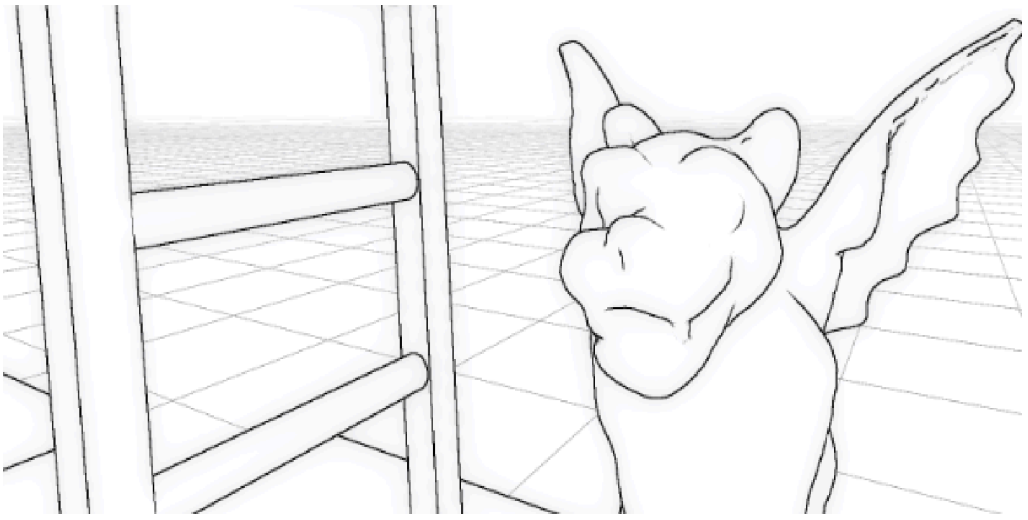

Figure 1: A line-art illustration generated by the proposed algorithm. Besides being robust and efficient to compute, it additionally addresses the consistency between left and right views in a VR context.

Besides being a widespread art style, line art can simplify the representation of an object. For 2D illustrations, it has even been shown that the use of an abstract representation can be beneficial for memorisation and recognition [Winnemoller2006]. Given that we rely increasingly on rendering, e.g., via virtual and augmented reality (VR/AR), for educational purposes, one might wonder if such applications could not also benefit from a visual abstraction.

Certainly, one obstacle to employing such a technique in VR/AR is the absence of existing suitable algorithms to enable clean stereoscopic line drawings. There are various factors that make this task difficult, including two major challenges. First, the difficulty of generating consistent abstractions for both eyes of an observer. Imagine an outline of a sphere, from two different viewpoints, these outlines on the surface will not overlap. In consequence, the 3D impression can be incorrectly interpreted by our visual system. The human visual system bases stereo perception on the matching of identical points perceived in both eyes; their relative displacement in the two views allows us to infer the point's distance. In the case of an abstract representation, an identical point might appear differently in both views, leading to false or even conflicting depth impressions. A second obstacle is that the generation of images in VR requires efficient algorithms, as the target framerates vary between 90-120 images per second, which is required for a smooth and immersive user experience. This leaves very little compute time to execute an abstraction process.

We propose a novel method that is effective in terms of the quality of the line-art representation and efficient in terms of computation. It achieves high framerates by building on a geometric analysis that is executed on an image-based representation of the surface properties to achieve a high-performance analysis of the object's shape, which forms the basis for the placement of (feature) lines that constitute the line-art representation. Further, the algorithm is designed to achieve consistency between different viewpoints and, thereby, ensures compatibility with stereo matching. The method is relatively easy to implement but underwent significant optimizations to ensure efficiency on high-resolution imagery, which is another requirement in VR. It was designed to avoid the noticeable artefacts of existing work.

4.1. Preliminary Study on the Effectiveness of Stylization

To motivate our work towards a line-art rendering style in VR/AR, we first conducted a preliminary user study, in which we analysed the effectiveness of stylized rendering compared to standard rendering (we relied on the GGX shading model as a comparative realistic rendering solution). This study was not strictly required by the deliverable, but it is an important addition to foreshadow the usefulness of the novel rendering technique. A more detailed investigation will be performed within the context of the prototypes.

The study was conducted with 12 participants in the age group of 18—40. Due to privacy concerns of our ethics board, which validated the study, the age was not associated with the individual measurements. In all cases, the participants were naive to the goals of the experiment and provided informed consent without any compensation. Like previous work (i.e., the second experiment conducted by [Winnemoller2006]), we assess the memory retention for different types of stylised images with a memory game.

For this purpose, a wall with a grid of 12 boxes hanging on the wall, organised in 3 rows with 4 columns each, was shown to the participants in VR (Figure 2).

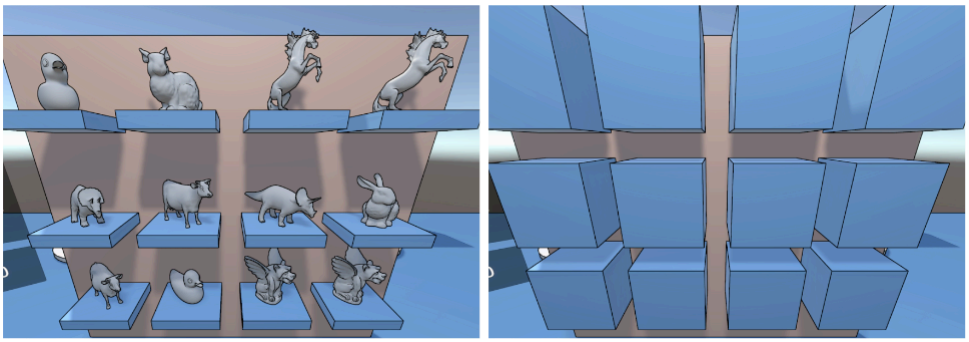

Figure 2. The puzzle layout. On the left, with the boxes open during the time the player should memorize the object locations, on the right, with the boxes closed when the players should select the parts.

Every box on the wall covered a 3D model. Objects are revealed once a participant uses one of their VR controllers to point at the box and presses the grip button.

Participants were tasked to find 6 pairs of matching models by consecutively opening two boxes with a respective match. Our hypothesis is that line art helps recognise models faster, improving recognition and matching timing when line art is added to the rendering output. For this purpose, we first compared an existing simple line-art stylization to a standard rendering using a GGX shading model. Please note that this first abstraction method might still exhibit artifacts due to stereo inconsistencies, which our newly developed method addresses. Some comments pointed at these artifacts, which was an additional motivator for the development of a more-suitable abstraction method.

4.1.1. Procedure

The study proceeds in rounds, each round switching the rendering method.

The participant may memorise the newly revealed models for two seconds before the boxes close and the round starts. Participants repeat this procedure 7 times. The first round is a practice round, and the 6 following rounds alternate in rendering style.

At the end of each round, we record the number of errors made as well as the time needed for completion.

4.1.2. Findings

We removed one outlier that took significantly longer, probably due to the adjustment of the headset during the task and one person who had a vision impairment, which became obvious during the study, as the person indicated that they were unable to recognize the shapes clearly, independent of the rendering style. With the remaining 10 samples, we found that stylized rendering significantly differs in timing ($p < 0.02$) from no stylization. Our assumption included that lines can better highlight geometric detail, which the participants also confirmed in their comments.

4.2. New Visualization Algorithm

As a consequence of the observed benefits, we propose Local Surface Approximation Contours (LSA contours), a high-quality and efficient approach to producing stereo-compatible contour-like feature lines for virtual reality. By definition, they avoid viewing discomfort and inconsistencies. Specifically, we made the following contributions. First, we developed a highly efficient automatic screen-space solution for robust contour-like feature lines. Second, we show how it can be adapted to prevent binocular rivalry and

stay temporally coherent, as well as spatially consistent. Third, we evaluated its robustness and effectiveness compared to state-of-the-art methods and illustrated the potential benefit of such representations in a user study. While this user study is not a direct requirement of the deliverable, we added the preliminary results to illustrate the benefit of the novel solution.

4.2.1. Related Work

To place our visualization technique in context and illustrate the importance of the topic, we briefly recap related works. Line art has received much attention in non-photorealistic rendering, and many stylisation systems rely on extracted feature lines [Grabli2004, Eisemann2008]. Here, we only focus on some key methods for extracting feature lines and refer to a recent survey for more details [Benard2019]. Early computer graphics relied on vector graphics, which is a kind of line-art representation, which was considered a useful means for illustration purposes. Already early on, efficiency was an important factor. Besides robust object-space acceleration techniques [Appel1967], screen-space methods [Saito1990], deriving feature lines directly from image buffers, existed early on. In our work, we also ensure robustness and efficiency. Besides the contour generator, where the normal is orthogonal to the viewing direction, many additional feature lines exist. Classic examples are ridges and valleys Lopez et al. [1999], defined as minimum and maximum principal curvature lines, which our solution also naturally extracts.

Other curve definitions include suggestive contours [DeCarlo2003] and apparent ridges [Judd2007]. These curves improve shape perception drastically in the absence of additional shading cues [Cole2009]. In our work, we make use of shading cues following the findings in [Winnemoller2006] and [Winnemoller2007], who underline the benefit of shading cues. Important aspects for virtual reality include performance and coherence of the extracted lines (temporally and regarding both views). Indeed, existing screen-space approaches suffer from poor quality due to occlusion and aliasing.

Further, view-dependent feature lines are often not temporally coherent. Even the relatively slow object-space suggestive contours required a specialized solution [DeCarlo2004], which does not work robustly for the efficient screen-space variant. Finally, even simple contour lines are not stereo-consistent [Bayle2019]; a cube might be seen solely from the front in one eye but from the side in the other, leading to potential discomfort.

Stereo-consistency problems also occur in other VR contexts like Screen-Space Ambient Occlusion (SSAO), where reprojection techniques [Yang2011, Northam2012] can be used to address some of the issues [Shi2022]. Besides reprojection, an alternative is to remove the view-dependency from the feature lines [Kim2013, Bukenberge2018, Benard2019, He2019]. Often, this involves a single view from where the feature line is reprojected, or several views contribute to a canonical curve. Unfortunately, such solutions either involve geometric solutions, which are costly, or image-space solutions still often lead to artefacts due to occlusions and imprecision. Our approach operates in screen-space and ensures consistency in a robust manner.

A hybrid solution between screen-space and object-space is to give the depth buffer a geometric interpretation. While early approaches [Thibault2010] fit a viewport-aligned plane and evaluate how well neighbouring pixels fit on that plane. Recent approaches [Tanaka2024], use a similar concept to fit a general plane. In both cases, pixels that deviate are considered part of a feature curve. Our solution generalizes this concept, achieving higher robustness and view-consistency.

A final remark concerns the compatibility of line extraction with various rendering methods, such as normal, bump or displacement mapping. Implicitly, these techniques do give the illusion of surface details (see Figure 3), which should affect the feature lines. This is not the case for most solutions. Our method will integrate such information. Further, our robust curvature approximation could have applications in other contexts, such as SSAO.

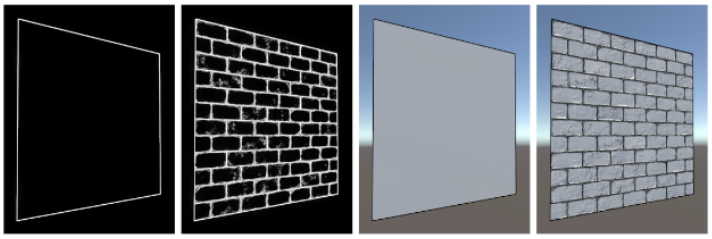

Figure 3. Illustration of compatible rendering methods. From left to right: simple quad as geometry, adding lines from the normal map, simple shading, and adding bump maps.

4.2.2. Methodology

We propose an image-based approach using a local surface approximation (LSA) in combination with a registration error. Our approach aims at determining which pixel might contain a feature line, based on whether the pixels' underlying geometry cannot be registered to an LSA.

Briefly, we fit a local approximation to the 3D positions represented by the geometry captured in the local pixel neighborhood in screen space. We then compute a registration error, which indicates how far each 3D position in the pixels is from the constructed local surface approximation, the LSA. If this error is large, it implies that there exists a discontinuity that should be represented in a line drawing. Even though we use 3D information captured from the involved geometry to achieve a high precision, the entire algorithm is designed to work in image space to make it efficient and, thereby, ready for VR use.

For a detailed and technical description of the method, please refer to the scientific paper provided.

4.2.3. Implementation

Demonstrator 1-3 were implemented in Unity Engine 2023.1.10f1 as a URP Renderer Feature using multiple render steps to capture and process the geometry in screen space and to extract the resulting feature lines. To reduce the memory bandwidth requirements, every shader stage only outputs data in the lowest bit depth and the smallest number of channels needed. The demonstrator further relies on deferred rendering and the main camera uses temporal anti-aliasing.

4.2.4. Performance Results

First, we qualitatively analyse the smoothness and robustness of the generated line art. Second, we quantitatively benchmark the performance and performance variance for various rendering resolutions. The head-mounted display (HMD) used for the evaluation is the HP Reverb G2 Omnicept (G2O), a high-resolution headset that requires an effective render resolution of 3164x3092 pixels per eye. Additionally, the XR Device Simulator (XDS) in the Unity XR Interaction Toolkit was used to avoid the result being influenced by head motion, driver latencies and VSync. The XDS has an effective render resolution of 1512x1680 per eye, which is significantly lower than most HMDs render at by default.

All samples were collected using the XDS. As a state-of-the-art comparison method, we use the solution described in [Honks2019]. In Figure 4, the top row shows the results of this technique. Notice how details in the facial region are lost, and noise occurs, which is particularly visible in VR (yet, a screenshot cannot really convey the stereo mismatch, nor the temporal instability). The row below shows our solution, which remains robust even when low resolutions are used, which makes it a formidable option for accelerating the method further. Our approach faithfully detects details and lines stay clean without resorting to noise, as on the wing in the right example.

Figure 4. Top row, existing solution, which is sensitive to different resolutions and not robust, hence, it misses lines at low resolutions. Bottom row, our method, which is consistent under different resolutions.

We evaluated the performance of our method on both the XDS and G2O with a desktop PC containing an Intel Core i7-12700K, NVIDIA GeForce RTX 3090, and 32 GB of DDR5-4400 RAM. All hardware components were running at stock clock speeds. The metric measured is the mean frame time of 300 frames collected using the Unity Frame Analyser tool (see Figure 5).

On the XDS, the performance impact is consistent with the number of pixels processed (approximately 3.5e6 px/ms) after the resolution passes 1890x2100. Similarly, on the G2O, the performance impact is consistent (between 3e6 and 4e6 px/ms).

The method cannot maintain 90 FPS on the G2O at 1.0x render scale (stereo 3164x3092) without any optimisation, only achieving 70 FPS. At 0.85x render scale (stereo 2690x2630), however, the method can maintain 90 FPS consistently without dropped frames. Our more efficient variant that is described in the accompanying technical paper, can maintain a stable 90 FPS at 1.0x render scale.

4.2.5. Preliminary Study on Quality

We used the same memory game setup as is provided as a demonstrator. We asked 18 participants aged 18-40 to create a preference ranking of the three styles in the context of geometric understanding based on the question “Which style lets you best understand the geometric properties of objects in the scene, notably their shape?”.

Participants are allowed to move through the scene and switch between styles without a time limit. Once a participant has determined their preference order, we record the answer. The results of the geometric understanding preference order can be found in Tables 1 and 2. First, out of the 18 participants for whom we could compare our method to no stylization, 15 preferred our method. Second, out of the 13 participants, where we additionally compared the depth-normal difference, our method was still the highest-ranked method.

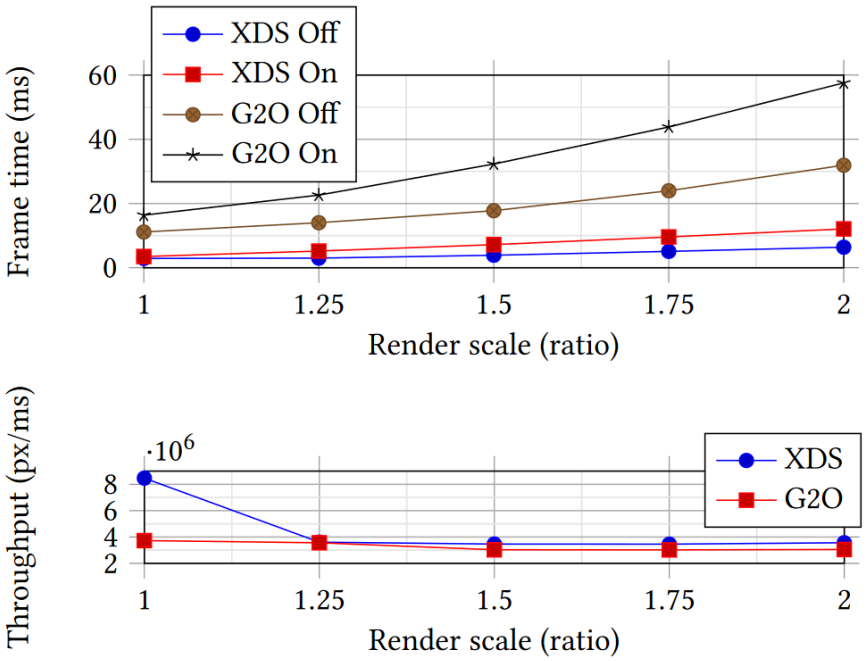

Figure 5. Timings for the XDS and the Omnicept device (G2O).

Ranking		Percentage
LR > Off	15	83.8%
Off > LR	3	16.7%

Table 1: Preferred geometric understanding: our method (LR) and no stylization (Off).

Ranking		Percentage
L > DN > Off	7	53.8%
DN > L > Off	3	23.1%
L > Off > DN	1	7.7%
DN > Off > L	1	7.7%
Off > DN > L	1	7.7%
Off > L > DN	0	0%

Table 2: Geometric understanding: ours (L), [Honks2019] (DN), and no stylization (Off).

4.2.6. Conclusion

Our demonstrator showed that suitable visualization techniques can improve interaction, memorization and recognition in a virtual-reality context. Our LSA method was integrated in Demonstrator 1 next to other comparative solutions. Our algorithm is a general method, adding line art to a rendering engine by detecting geometric discontinuities in screen space, requiring no actual geometric preprocessing and supporting normal- and bump-maps. The method can run in real-time at high resolutions and automatically reduces stereoscopic inconsistencies through reprojection. Results of our user study indicate that our method is preferred over existing methods.

5. Navigation Demonstrator

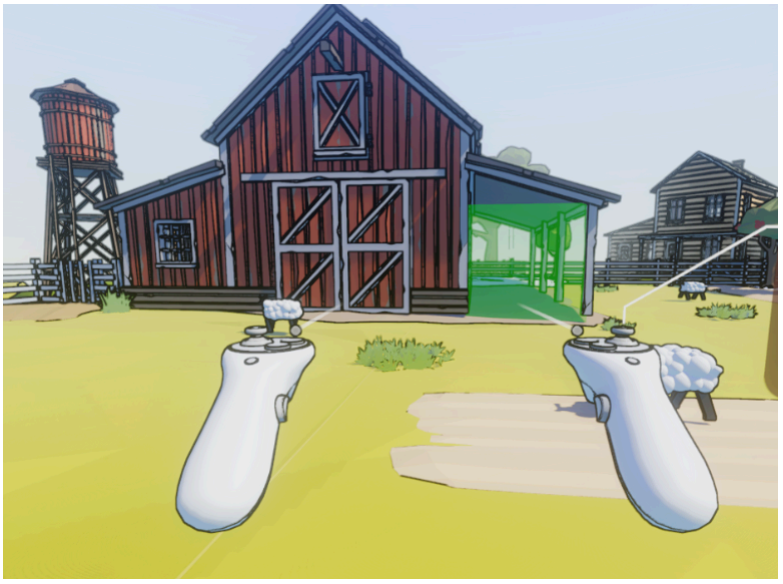

Figure 6: Screenshot of our navigation demonstrator (Demonstrator 4)

This part relates to Demonstrator 3, as outlined in the executive summary. We give a short overview of the motivation and development of the demonstrator and give insights into preliminary user testing.

Newcomers often find VR difficult to use. One particular skill that proves unintuitive to new users is navigating a scene or the effective use of movement methods available in a virtual environment. To this end, we developed SHEPHERD, which is delivered as Demonstrator 4 (see Figure 6). It is a demonstrator in the form of a game, in which players are tasked with corralling a herd of sheep. It is a gamified way for new users to engage with VR technology and learn common navigation techniques.

The main goal driving the design decisions for the game was teaching newcomers how to navigate in VR efficiently using four of the most commonly used VR navigation techniques. To lower the learning difficulty and increase user engagement, this teaching process was gamified, directly linking different elements of movement with an entertaining and relaxed experience.

5.1. Concept

The game is structured around the activity of corralling sheep towards a desired location in a level-based sequence. When the player enters a level, they start on a farm with sheep placed around. By utilizing one of four available navigation methods, they are able to direct the sheep towards a highlighted area until every sheep is herded into the designated area. Afterwards, the level is complete, and they progress to the next one.

The sheep are influenced by the movement of the player. More specifically, they always react by moving away from the player when the player gets close. The player is given an open area to explore and a simple goal achievable through movement. The game concept relies on the player's drive to complete a level to learn to move in a VR environment. Due to the nature of the game, repetitive movements are needed, both to navigate around the level and to direct all the sheep. As there are no other imposed

restrictions, the player can safely and impulsively learn how to use the presented navigation method through trial and error.

5.2. Setting and Art Style

The setting and activity of corralling sheep on a farm were chosen due to the activity's reliance on movement to be performed. This allows the game to focus its teaching on navigation, without requiring any additional VR skills. Further, the setting of a farm provided a stress-free setting in which players can safely get introduced to the movement methods. A simplified and cartoon-like visual style aids in the creation of a peaceful, welcoming environment for VR newcomers.

5.3. Level Design

The game is organized into four levels that each introduce one new navigation technique. Each level is designed around the advantages and disadvantages of the corresponding technique.

The first level, which acts as an introduction to the game, teaches teleportation. The play area is a small field with only a few sheep to herd. Additionally, the control scheme for rotating the player's vision is introduced. The in-game implementation of teleportation is to press the joystick in a given direction, resulting in an arc with a landing spot appearing on the screen. Once the joystick is released, the player is teleported to this location. The direction in which the joystick is pressed influences the viewing direction once teleported.

The second level teaches dashing. The play area is elongated and invites the player to move across it multiple times, allowing the player to practice moving both long and short distances with the method. Our implementation of dashing has the player pressing the joystick in a given direction for a certain amount of time. The time for which the joystick is held determines the distance for which the player is moved, and this distance is visualised by a vector that extends from the player in the direction the player will be travelling. To prevent the player from moving in a direction they are not looking into, which could cause motion sickness, we make the player's vision rotate if the dash vector threatens to leave the player's field-of-view. During playtesting, this measure tended to reduce motion-sickness to a minimum. It therefore seems a suitable element that should be adopted by other applications. With this measure, we could even avoid the very common use of vignetting (a tunnel-like reduction of the player's field of view) during the dash movement.

The third level teaches our implementation of WIM called the mini world. The player has to move between multiple enclosed areas over a long distance to corral all the sheep. This teaches the player that a mini world can help traverse larger distances over disconnected areas, and how the miniature world can help get a quick overview of their location. It will also teach them that it is less useful for smaller distances because of the time it takes to locate yourself in the miniature. The in-game implementation of the mini world involves the player holding one of the triggers, which makes a map appear. Once the map has appeared, the player can then use the other controller to point at the map and press the trigger to teleport the player to this point of the map.

The fourth level teaches one-handed flying. The sheep are on different elevations, which the player must fly up to in order to corral the sheep. The in-game implementation of flying works by pressing a button, which triggers the movement in the direction the player is looking. The speed is controlled based on the distance at which the player holds their hands in front of them. We chose to implement the movement method this way because of its resemblance to and the intuitiveness of doing a superhero-styled flying pose, which many players indeed adapted intuitively.

5.4. Difficulty

The difficulty and length of the game were kept in consideration. Preliminary testing was used to find a good number of sheep to match the difficulty and length of the game. To keep the game from becoming too challenging to newcomers, the time spent in each level was limited by having only a small number of sheep that need corralling, and only allowing one type of movement at a time. This decision comes at a cost of engagement and replayability, but these aspects were not the main focus of this game.

5.5. Implementation

The Unity Engine version 2022.3.13f1 was used for the development of this demonstrator. The used assets are a combination of purpose-built assets and store-bought. A variety of models used for the scenery were taken from the asset pack by Synty Studios. The sheep and shepherd models were created and animated using Autodesk Maya version 2023.3 and then imported into Unity.

5.6. Conclusion

The demonstrator covers the most important navigation systems in a playful manner. Preliminary testing showed that it is indeed a useful environment to learn VR interaction because beginners feel easily challenged, while more advanced users can finish the tasks relatively easily. It further provides a useful overview for educators of various VR navigation methods and recalls their use.

6. Interaction Demonstrator

Figure 7: Screenshot of the interaction demonstrator (Demonstrator 5)

This part relates to Demonstrator 4, as outlined in the executive summary. We give a short overview of the motivation and development of the demonstrator and give insights into preliminary user testing.

Demonstrator 5 (see Figure 7) covers controller interaction with objects. It also takes the form of a game called Puzzle Playground, which builds familiarity with VR by teaching object interactions through puzzles in an interactive experience tailored for VR newcomers. Players gradually learn VR interactions by completing various puzzle levels. A preliminary evaluation study was conducted to determine whether learning with Puzzle Playground was more efficient than learning with alternative teaching methods. The study indicates that users who learned with Puzzle Playground grasped VR interactions faster than those who learned with alternative teaching methods. Our contribution is twofold: (i) we identify the common basic interactions and controls in most VR applications, and (ii) propose Puzzle Playground, a game designed to introduce inexperienced VR users to interaction mechanisms. Preliminary evaluation of the game suggests that the learning retention and performance of its players is higher than that of other participants, who learned with either a written explanation or a recorded video.

6.1. Introduction

We designed and developed the demonstrator Puzzle Playground, a first-person puzzle game in a sandbox playground centered around common VR interactions. It consists of several levels, each with a single objective to complete. For each level, players learn one VR interaction type and must subsequently apply it to solve the task at hand. The levels' setup allows the players to make mistakes and easily recover from them, encouraging them to explore the interactions until they have fully mastered them.

As the game is aimed at beginners, the number of interaction mechanisms is kept small. Based on the most popular VR games and applications, we identified the following interaction categories as the most important to learn:

- grabbing and displacing objects
- rotating objects
- scaling objects
- bringing distant objects into your hand (informally known as ‘tele-grab’)
-

As Puzzle Playground includes UI elements to navigate through the menu, players should also get familiar with how to interact with these.

The game scenario is a stylized kindergarten environment. This environment is beneficial for the learning objectives for two reasons. First, a kindergarten creates an analogy to the inherent learning experiences associated with early education. A kindergarten is typically the first educational institution. The goal of Puzzle Playground is to educate inexperienced players through interactions in VR. A concrete example is stacking blocks to practice grabbing and moving objects. Stacking blocks is a typical activity in a kindergarten, thus players who understand the analogy could intuitively succeed in the learning process. Second, a kindergarten has the benefit of being a fairly simple environment. This reduces distractions for the player, as opposed to being indulged in a more thrilling environment, which might then drive attention away from the puzzles and the learning goals.

To focus on interactions, there are no navigation mechanisms, meaning that the player cannot move around the environment.

In the following, we discuss the level design. The introductory levels present one of the four key interactions and are expected to be easy to complete. Since the players are introduced to the

mechanics one by one, they can fully focus on and achieve the learning objectives. The challenging levels test the player's knowledge by requiring them to combine multiple interactions to complete the level. These levels reinforce and deepen the players' knowledge through repetition and recall in different contexts.

Every level contains instructions in the form of a floating UI element. The UI element contains helpful explanations on accomplishing the interaction(s) needed for the level. The instructional UI elements shown at the start of a level are only meant to be read and do not require particular interaction. Once a level is completed, the UI element shows two buttons to let the player move on to the next level or return to the main menu. Instructions on how to use these buttons are available in the game's main menu, where the player also needs to press the 'Start Level' button to begin the game.

The game consists of six levels, with one elementary puzzle per level. Out of the six levels, two are dedicated to picking up and placing objects: one to introduce the player to the interaction and another to complement with a more precise placement of objects. These two are followed by one level dedicated to grabbing distant objects (`tele-grabbing'). Grabbing and tele-grabbing are the interactions that are most likely to be included in a VR game or application. Therefore, the first three levels at the start of the game are dedicated to teaching these mechanisms. The following levels teach less popular interactions. These may not be found in games and applications as often as the first two but may be helpful in a classroom setting. The third interaction is rotation. Two options are presented to the player on how to rotate objects. They can either use the orientation of their hand to rotate an object they are holding or use a stick on a VR controller to rotate it. Finally, the last two levels teach the use of gestures in the context of object scaling.

6.2. Implementation

The Unity Engine version 2022.3.13f1 was used. The used assets are a combination of purpose-built assets and assets from a pack by Synty Studios.

6.3. Evaluation

In order to assess the effectiveness of Puzzle Playground, we conducted a preliminary evaluation. This section describes the methodology and results of this process. We conducted a formative user study to gauge how well participants mastered interactions within a virtual environment. Specifically, we wanted to test whether participants would learn better with a VR application or with traditional methods.

To achieve this, the participants were divided into three groups, each undergoing a distinct learning approach: one group engaged in the game, another watched a recorded presentation, and the third read a written explanation. After the learning phase, participants played an external VR application to evaluate how well they learned the interaction mechanisms. Each participant was allocated a 30-minute slot, divided into 15 minutes for the learning phase and 15 minutes for the evaluation phase. The testing phase was designed to evaluate how well participants would perform in a realistic VR scenario. For this, we chose the VR game Job Simulator because it contains a large subset of the interaction methods taught in Puzzle Playground.

Additionally, the game lacked manual locomotion, which is consistent with Puzzle Playground, in which the player does not walk around. Finally, Job Simulator contained levels with many sub-tasks, making it easier to evaluate players' performance in the game quantitatively.

Before the study, participants were asked for their age group and self-reported proficiency levels on a scale of 1 to 5 for technology in general, phones, and virtual reality (VR). After the learning phase, we asked the participants what their confidence level was with interactions in VR. After the testing phase, we asked participants how sufficient the learning phase was to complete the task. The primary metric recorded during the study was the number of tasks and sub-tasks the participants completed during the allocated 15 minutes. The number of times the participants asked for help during their play session was also recorded.

Our user study had 9 participants, all between the ages of 20 and 30 and with no or little VR knowledge. An overview of the groups and average scores can be seen in Table 3.

	Puzzle Playground	Reading	Presentation
# Participants	4	3	2
Prior VR knowledge (1-5)	2	1.33	2
Avg. confidence	4.25	2.67	3.5
Avg. tasks	8	7.67	5
Avg. help	0.25	0.33	6
Avg. sufficient	4.75	2.33	4.5

Table 3: Average score of the questionnaire and tasks per group.

Three aspects of the data indicate that Puzzle Playground is an effective introduction to VR. First, the confidence of the group that played Puzzle Playground is higher than that of the control groups. Second, the group that played Puzzle Playground completed the most tasks on average in the external evaluation game and needed the least amount of help. Third, candidates who played Puzzle Playground felt most sufficiently prepared to complete the evaluation tasks. Unfortunately, the presentation control group had lower self-reported technological and video game knowledge than the other groups. This likely led to a significant difference in performance between them and the other groups. However, even the least proficient member of the game group performed better in the evaluation than in the presentation group.

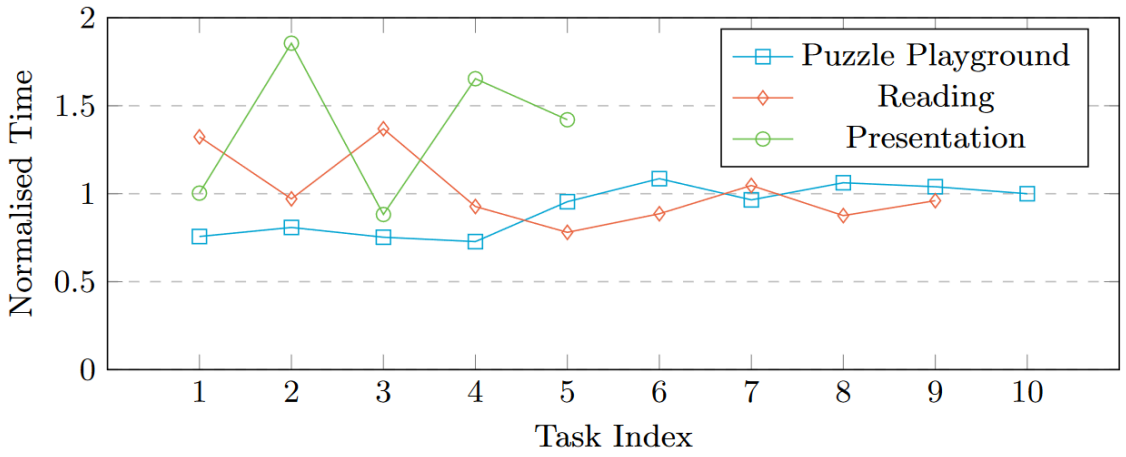

Figure 8: Normalised average task completion time per group.

Figure 8 shows the normalised average task completion time per group. This was obtained by averaging the task completion times per group, and normalising by the overall average time for a given task. As only a few participants passed Task 7, we expect values after Task 7 to be close to one. In the first few tasks, the game group starts out consistently faster than the other two groups. Members of the reading group stumbled often in the early tasks but would get the hang of the controls after about 5 minutes in VR. Even though the reading group manages to catch up to the game group after a trial-and-error phase in the evaluation tasks, Puzzle Playground appears to be an effective introduction to VR, as it lets newcomers skip this trial-and-error phase without requiring additional time compared to a written or recorded introduction.

6.4. Conclusion

There is a significant lack of VR experience among the population. For many, the bar to enter VR is high and gaining familiarity in safe and simple applications can improve accessibility. This observation also holds for educators who are new to VR, and solving this technological hiccup can potentially drastically improve education in the near future. Puzzle Playground is a VR game that is expressly designed and developed to teach players how to interact with objects in VR. The game approaches interaction in a structured, step-by-step manner, guiding players through basic VR interactions and letting them play through various puzzle levels.

Our preliminary user study was limited to a small number of participants, but still showed that learning VR interactions through Puzzle Playground is more time-efficient than the other tested alternative methods, such as a presentation or written explanations. Further investigations are intended for the upcoming period, as outlined in the planning of the e-Diploma project.

7. Interaction Demonstrator

Arthur Appel. 1967. The notion of quantitative invisibility and the machine rendering of solids. In Proceedings of the 1967 22nd national conference (ACM '67). Association for Computing Machinery, New York, NY, USA, 387–393. https://doi.org/10.1145/800196.806007

Elodie Bayle, Esetelle Guilbaud, Sylvain Hourlier, Sylvie Lelandais, Laure Leroy, Justin Plantier, and Pascaline Neveu. 2019. Binocular rivalry in monocular augmented reality devices: a review. In Situation Awareness in Degraded Environments 2019, Vol. 11019. SPIE, 136–149. https://doi.org/10.1117/12.2518928

Dennis R. Bukenberger, Katharina Schwarz, and Hendrik P. A. Lensch. 2018. Stereo-Consistent Contours in Object Space. Computer Graphics Forum 37, 1 (2018), 301–312. https://doi.org/10.1111/cgf.13291

Pierre Bénard and Aaron Hertzmann. 2019. Line Drawings from 3D Models. Foundations and Trends® in Computer Graphics and Vision 11, 1-2 (2019), 1–159. https://doi.org/10.1561/0600000075 arXiv:1810.01175 [cs].

Forrester Cole, Kevin Sanik, Doug DeCarlo, Adam Finkelstein, Thomas Funkhouser, Szymon Rusinkiewicz, and Manish Singh. 2009. How well do line drawings depict shape?. In ACM SIGGRAPH 2009 papers (SIGGRAPH '09). Association for Computing Machinery, New York, NY, USA, 1–9. https://doi.org/10.1145/1576246.1531334

Doug DeCarlo, Adam Finkelstein, and Szymon Rusinkiewicz. 2004. Interactive rendering of suggestive contours with temporal coherence. In Proceedings of the 3rd international symposium on Non-photorealistic animation and rendering (NPAR '04). Association for Computing Machinery, New York, NY, USA, 15–145. https://doi.org/10.1145/987657.987661

Doug DeCarlo, Adam Finkelstein, Szymon Rusinkiewicz, and Anthony Santella. 2003. Suggestive contours for conveying shape. ACM Transactions on Graphics 22, 3 (July 2003), 848–855. https://doi.org/10.1145/882262.882354

Elmar Eisemann, Holger Winnemöller, John C. Hart, and David Salesin. 2008. Stylized Vector Art from 3D Models with Region Support. Computer Graphics Forum 27, 4 (2008), 1199–1207. https://doi.org/10.1111/j.1467-8659.2008.01258.x

Epic Games. [n. d.]. Bump Mapping Without Tangent Space In Unreal Engine | Unreal Engine 5.4 Documentation.
https://dev.epicgames.com/documentation/en-us/unreal-engine/bump-mapping-without-tangent-space-in-unreal-engine

Stéphane Grabli, Emmanuel Turquin, Frédo Durand, and François X. Sillion. 2004. Programmable Style for NPR Line Drawing. The Eurographics Association. http://dx.doi.org/10.2312/EGWR/EGSR04/033-044 ISSN: 1727-3463.

Dejing He, Rui Wang, and Hujun Bao. 2019. Real-Time Rendering of Stereo-Consistent Contours. In 2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR). 81–87. https://doi.org/10.1109/VR.2019.8797990 ISSN: 2642-5254.

Roystan Honks. 2019. Unity Outline Shader Tutorial at Roystan. https://roystan.net/articles/outline-shader/

Tilke Judd, Frédo Durand, and Edward Adelson. 2007. Apparent ridges for line drawing. ACM Transactions on Graphics 26, 3 (July 2007), 19–es. https://doi.org/10.1145/1276377.1276401

Yongjin Kim, Yunjin Lee, Henry Kang, and Seungyong Lee. 2013. Stereoscopic 3D line drawing. ACM Transactions on Graphics 32, 4 (July 2013), 57:1–57:13. https://doi.org/10.1145/2461912.2462001

A.M. Lopez, F. Lumbreras, J. Serrat, and J.J. Villanueva. 1999. Evaluation of methods for ridge and valley detection. IEEE Transactions on Pattern Analysis and Machine Intelligence 21, 4 (April 1999), 327–335. https://doi.org/10.1109/34.761263 Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence.

Lesley Northam, Paul Asente, and Craig S. Kaplan. 2012. Consistent stylization and painterly rendering of stereoscopic 3D images. In Proceedings of the Symposium on Non-Photorealistic Animation and Rendering (NPAR '12). Eurographics Association, Goslar, DEU, 47–56.

Takafumi Saito and Tokiichiro Takahashi. 1990. Comprehensible rendering of 3-D shapes. ACM SIGGRAPH Computer Graphics 24, 4 (Sept. 1990), 197–206. https://doi.org/10.1145/97880.97901

Peiteng Shi, Markus Billeter, and Elmar Eisemann. 2022. Stereo-consistent screen-space ambient occlusion. Proceedings of the ACM on Computer Graphics and Interactive Techniques 5, 1 (May 2022), 2:1–2:12. https://doi.org/10.1145/3522614

Kosuke Tanaka and Takashi Komada. 2024. 3D Toon Rendering in 'Hi-Fi RUSH'. https://gdcvault.com/play/1034330/3D-Toon-Rendering-in-Hi

Aaron Thibault and Sean Cavanaugh. 2010. Making Concept Art Real for Borderlands. https://stylized.realtimerendering.com/#borderlands

Unity Technologies. [n. d.]. Unity - Manual: Normal map (Bump mapping). https://docs.unity3d.com/Manual/StandardShaderMaterialParameterNormalMap.html

Unity Technologies. 2024. Unity - XR Interaction Toolkit 3.0.3. https://docs.unity3d.com/Packages/com.unity.xr.interaction.toolkit@3.0/manual/index.html

Holger Winnemöller, David Feng, Bruce Gooch, and Satoru Suzuki. 2007. Using NPR to evaluate perceptual shape cues in dynamic environments. In Proceedings of the 5th international symposium on Non-photorealistic animation and rendering (NPAR '07). Association for Computing Machinery, New York, NY, USA, 85–92. https://doi.org/10.1145/1274871.1274885

Holger Winnemöller, Sven C. Olsen, and Bruce Gooch. 2006. Real-time video abstraction. In ACM SIGGRAPH 2006 Papers (SIGGRAPH '06). Association for Computing Machinery, New York, NY, USA, 1221–1226. https://doi.org/10.1145/1179352.1142018

Lei Yang, Yu-Chiu Tse, Pedro V. Sander, Jason Lawrence, Diego Nehab, Hugues Hoppe, and Clara L. Wilkins. 2011. Image-based bidirectional scene reprojection. ACM Transactions on Graphics 30, 6 (Dec. 2011), 1–10. https://doi.org/10.1145/2070781.2024184

Bruce Walter, Stephen R. Marschner, Hongsong Li, Kenneth E. Torrance, Microfacet Models for Refraction through Rough Surfaces, Eurographics Symposium on Rendering (2007)

e-DIPLOMA

Funded by
the European Union