

# Real-Time Editing of Path-Traced Scenes with Prioritized Re-Rendering

Annalena Ulschmid<sup>1</sup><sup>a</sup>, Bernhard Kerbl<sup>1</sup><sup>b</sup>, Katharina Krösl<sup>1,2</sup><sup>c</sup> and Michael Wimmer<sup>1</sup><sup>d</sup>

<sup>1</sup>TU Wien, Austria

<sup>2</sup>VRVis Forschungs-GmbH, Austria

**Keywords:** Path Tracing, Scene Editing, Adaptive Rendering, Empirical Studies.

**Abstract:** With recent developments in GPU ray tracing performance and (AI-accelerated) noise reduction techniques, Monte Carlo Path Tracing at real-time rates becomes a viable solution for interactive 3D scene editing, with growing support in popular software. However, even for minor edits (e.g., adjusting materials or moving small objects), current solutions usually discard previous samples and the image formation process is started from scratch. In this paper, we present two adaptive, priority-based re-rendering techniques with incremental updates, prioritizing the reconstruction of regions with high importance, before gradually moving to less important regions. The suggested methods automatically identify and schedule sampling and accumulation of immediately affected regions. An extensive user study analyzes whether such prioritized renderings are beneficial to interactive scene editing, comparing them with same-time conventional re-rendering. Our evaluation shows that even with simple priority policies, there is a significant preference for such incremental rendering techniques for interactive editing of small objects over full-screen re-rendering with denoising.


## 1 INTRODUCTION


Monte Carlo (MC) path tracing (Kajiya, 1986; Pharr et al., 2016) is a frequently used physically based rendering technique, as it naturally includes many effects such as global illumination, reflections, soft shadows, and caustics. Offline path tracing has become a staple of the movie industry and production rendering in general. For each pixel of a rendered image, a path is traced into the scene, and over time, multiple of these path samples are accumulated. Each additional sample helps to reduce MC noise, and sampling quality is often referred to via the number of samples per pixel (spp). Due to its stochastic nature, path tracing is unbiased, but a large number of samples is required for a noise-free image. Thus, even in the best-case scenario, renderings require several seconds before an acceptable level of quality is reached. Due to the high cost of path tracing, rendering use cases that require a short feedback loop (e.g., previews during interactive scene editing) rely on real-time approximation methods (often based on rasterization). These solutions closely, but not entirely, mimic the results obtained


by an eventual path-traced render. However, to provide more accurate previews, it is desirable to use path tracing interactively, also during scene editing. Such a solution would provide a more seamless experience and productive workflow for digital artists, as current approximating methods cannot fully capture complex light transport effects.


Schmidt et al. (Schmidt et al., 2014) identify interactive feedback as an open problem in artistic editing. Much has happened since then: Modern graphics hardware supports ray-tracing pipelines next to traditional rasterization. However, in real-time settings (30–60 FPS), current GPU hardware imposes a soft limit of  $\sim 1$  spp per frame. Several powerful noise-reduction techniques have been introduced, allowing not only shorter rendering times for offline rendering, but also path tracing in interactive settings. However, current solutions generally discard all previously accumulated samples as soon as any part of the scene changes (camera movement, object modifications, material edits). Although this is rightly justified by physical correctness (as every change can affect indirect light transport), it results in low image quality during editing and disruptive artifacts, e.g. flickering or blurriness.

Global image updates are reasonable when the scene changes overall, but can be unnecessarily dis-

<sup>a</sup> <https://orcid.org/0000-0002-0539-9378>

<sup>b</sup> <https://orcid.org/0000-0002-5168-8648>

<sup>c</sup> <https://orcid.org/0000-0002-9939-0517>

<sup>d</sup> <https://orcid.org/0000-0002-9370-2663>

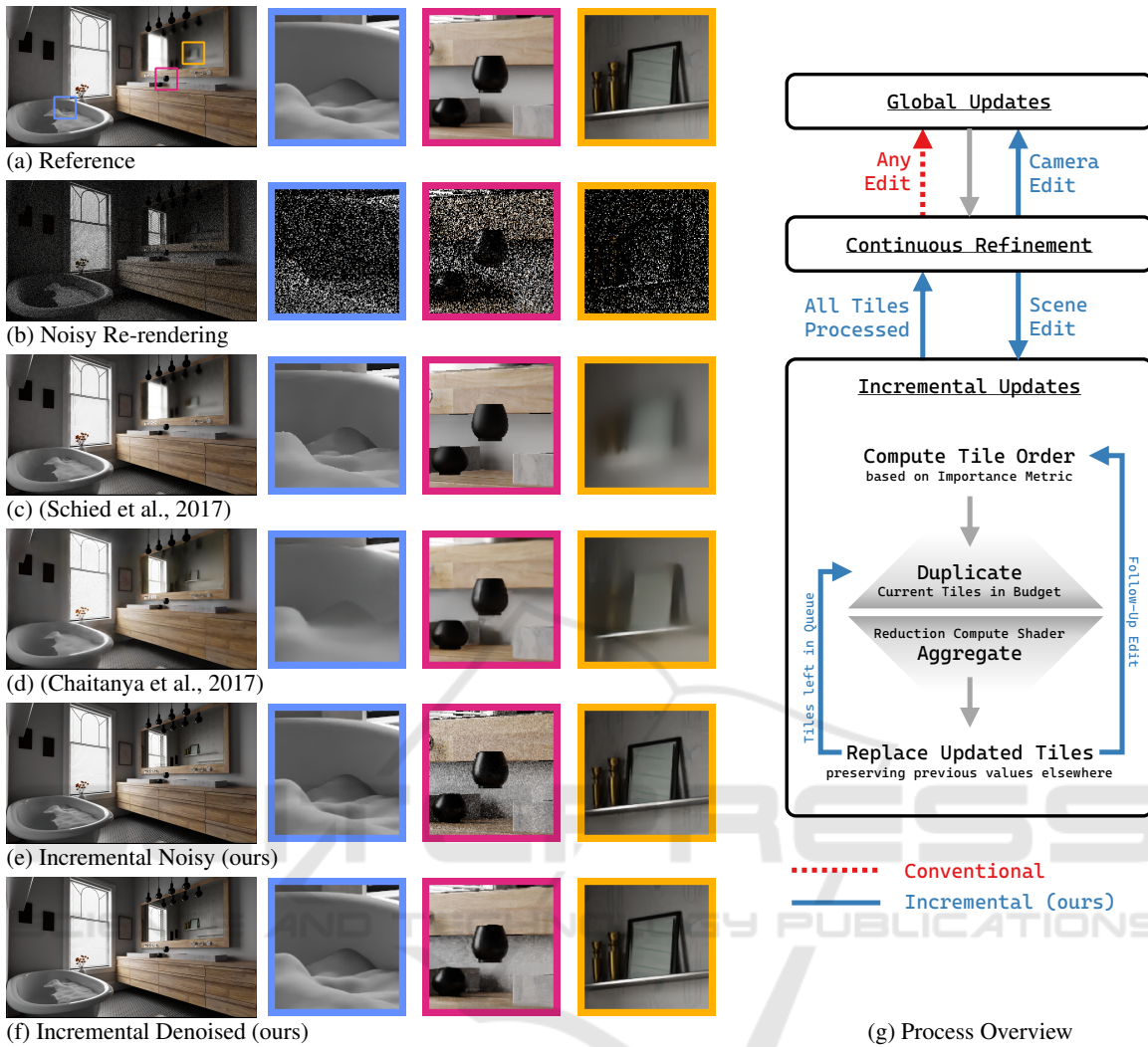


Figure 1: (a) – (f): Comparison of different rendering methods with 1 spp sample budget to the ground truth (a) of an interactively edited scene, after slightly raising the cup. In (b) previously accumulated samples were discarded during the computation of the new visualization and the image is updated globally, resulting in disruptive quality degradation everywhere. (c) and (d) employ denoising to reconstruct high-fidelity images from (b). (e) and (f) use a tile-based approach with incremental updates, concentrating available samples on a subset of screen-space tiles and processing them in order of importance, measured by the Chebyshev distance from the edit on screen. The latter two avoid both noise and overblurring in the entire scene. (g): Diagram of our suggested re-rendering process containing incremental updates.

ruptive for minor modifications that influence only a localized region: The artist must retain their vision of a desired visual change while having to observe an instantaneous regression from converged images to grainy or blurry visuals during the first few moments of image formation. Manually specifying a static region of interest for a high-quality preview, as some production renderers allow, cannot react to dynamic scene changes with large object movements.

In this paper, we propose to automatically identify areas of high importance and, following an edit, re-render these first at higher quality. An importance

policy is used to identify image areas where changes have a lower impact and schedules gradual updates in an incremental fashion. By rendering in the order of importance, we eventually update the entire image to achieve physical correctness but allow an artist to first focus on their modification. To determine importance, we use the Chebyshev distance of pixels from the modified object’s position on screen. We implement two flavors of this approach: One noisy variant, where each MC-integrated image region is presented immediately, and a denoised variant with an additional denoising step. Importantly, we aim to

verify the usability of such incremental re-rendering approaches in an interactive editing setting. Based on our suggested methods, a detailed user study is conducted to evaluate benefits of both the noisy and denoised methods for real-time scene editing. The incremental path tracing solutions are compared to a state-of-the-art denoiser that discards all previously accumulated samples upon edits (conventional re-rendering). Through this user study, we aim to answer the following research questions:

- **RQ1.** Which re-rendering method do artists prefer (w.r.t. perceived workload and focusability)? Are preferences dependent on specific editing scenarios (small objects, large objects, lights)?
- **RQ2.** Which kind of updates (incremental or global) do artists prefer overall?
- **RQ3.** What is their general attitude towards path tracing for scene editing?

By answering these questions, we hope to quantify the benefits of our proposed methods and further gain a better understanding of the artists' needs. Developing techniques adapted to their workflow could enable them to use the same rendering method during scene editing as for the final product and enhance their overall productivity. Our main contributions thus are:

- The design of a tile-based, incremental real-time path tracing pipeline. Scheduled tiles are repeated multiple times in the buffer input of a Path Tracer. This enables replacing conventional, global re-rendering with same-time, adaptive re-rendering, requiring only minimal changes to underlying rendering backends.
- Two implementations of automatic, prioritized re-rendering policies for editing: one displaying the raw values, which may still be noisy, and one with an additional denoising step.
- A detailed user study that investigates the preferred rendering method (conventional or incremental updates) of experienced artists in different scenarios (scene modifications with uniform and non-uniform impact), as well as offering new insights into the artists' workflow and requirements.

## 2 RELATED WORK

The foundations of MC path tracing are provided by Kajiya (Kajiya, 1986). A thorough description of implementing an offline path tracing framework is given by Pharr et al. (Pharr et al., 2016). The recently introduced Turing architecture (Nvidia, 2018) enables

hardware-accelerated accelerated ray-tracing and facilitates real-time path tracing at a low sample count.

### 2.1 Adaptive Sampling

Zwicker et al. (Zwicker et al., 2015) categorize adaptive rendering strategies into a-priori and a-posteriori methods. A-priori approaches incorporate local 3D geometry for example to compute gradients or analyze the light transport equations or analytical BRDF models to decide where to invest in a higher amount of samples. A-posteriori base this decision on statistics such as the variance of samples thus far accumulated or the mean squared error (MSE). Seminal work on adaptive sampling to achieve high quality with reduced sample count was provided by Mitchell (Mitchell, 1987). Select examples of later methods focused on path tracing are presented by Overbeck et al. (Overbeck et al., 2009) (measuring the error based on 2D wavelet approximations) or Hachisuka et al. (Hachisuka et al., 2008) (computing an error based on samples stored in multidimensional path space). However, these works and later follow-up techniques target offline rendering and do not consider interactive aspects. In particular, to the best of our knowledge, none of these adaptive approaches is designed in the context of scene editing. In contrast, we exploit the priors of the editing process (information where changes occurred) to steer the sample distribution.

Strategies for improving importance sampling as pursued by techniques like ReSTIR (Wyman and Panteleev, 2021) constitute a research direction complementary to ours, as both approaches could be combined. The aforementioned paper of Wyman and Panteleev (Wyman and Panteleev, 2021) adapts ReSTIR for production purposes and amongst other things combines it with the ReLAX denoising approach.

### 2.2 Denoising Techniques

Much research has been conducted on improving the computational time of physically-based light-transport simulations to archive real-time frame rates. The main approach is to enhance noisy, but fast low-sample renderings by applying image reconstruction techniques as a post-processing step. Schied et al. (Schied et al., 2017) introduced spatiotemporal variance-guided filtering (SVGF), which strives to output a temporally coherent image sequence by accumulating multiple samples. They later refine this technique to reduce lag and ghosting artifacts by incorporating sparsely sampled gradients in the decision of whether to include a previous sample in a pixels history buffer (Schied et al., 2018).

Nvidia provides a library consisting of three real-time denoisers: ReLAX, ReBLUR, and Sigma (NVIDIA, 2021). Building up on SVGF, ReLAX, inspired by temporal anti-aliasing techniques, clamps to the fast history to improve the treatment of disoccluded areas and deal with ghosting artifacts at the expense of a noisier history buffer. It can be combined with the often more performant ReBLUR (Zhdan, 2021), which is used in computer games such as *Cyberpunk 2077* for enhanced visuals via ray-tracing. Both can separately denoise diffuse and specular signals and handle checkerboarded half-resolution input.

Recently, deep learning-based denoisers have been introduced such as Intel’s Open Image Denoise (OIDN) (Intel, 2019) used for example by Unreal Engine or Nvidia’s OptiX denoiser based on the research of Chaitanya et al. (Chaitanya et al., 2017). The latter adopts a recurrent autoencoder additionally using auxiliary buffers to include for example depth information and normals. Kuznetsov et al. propose a joint adaptive and reconstruction approach for low sampling rates based on convolutional neural networks (Kuznetsov et al., 2018). Neural methods, however, while still achieving close to real-time frame rates on recent hardware, currently involve a slightly higher computational cost. They are thus mainly used in offline or interactive rendering, but not in computer games. Thomas et al. recently proposed a combined neural denoising and supersampling approach with comparable runtimes to SVGF, but higher quality results (Thomas et al., 2022). Similarly, (Hasselgren et al., 2020) combine neural denoising with learned spatiotemporal sampling strategies.

In a dynamic context, all of the methods described above discard all previous samples to compute the current frame whenever something in the scene changes and only reuse them via temporal reprojection during the denoising step. Instead of these global updates, we propose an adaptive rendering approach with incremental updates tailored to artists using path tracing during scene editing.

### 2.3 Scene Editing and Perception

(Murakami and Hirota, 1992) introduce an incremental ray-tracing scheme, which tracks changes via a spatial voxel partition and hash indices. For consistent scene editing, (Günther and Grosch, 2015) propose using progressively computed difference images to identify regions where scene modifications have a high impact. Their method is based on stochastic progressive photon mapping, as they focus on slowly converging effects such as caustics, which are more negatively affected by a global reset. (Rous-

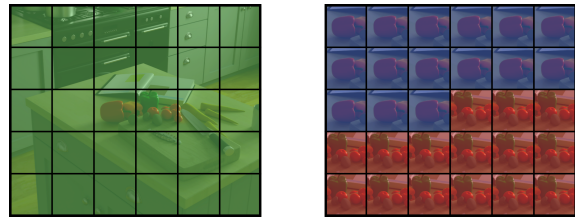


Figure 2: Tile layout representing traced paths of 1 spp renderings. On the left: Standard approach of sampling the entire image once (green). On the right: Our approach of repeatedly sampling only a few selected tiles (in the depicted case two in red and blue) during the incremental updates. The number of repetitions determines sample count within a tile, which we refer to as the *tile quality*. Note that the actual tiles are much smaller than depicted.

selle et al., 2016) reformulate the previous approach into a control-variate integration scheme and explore estimator combinations based on covariance in offline editing scenarios with static scenes and gradient-domain rendering. While they provide a thorough qualitative analysis of their method, they do not evaluate user benefits of such adaptive solutions, e.g., in the context of interactive editing scenarios. Additionally, the above approaches use uniformly distributed and not prioritized, adaptive screen-space samples. In contrast, we outline and thoroughly evaluate re-rendering methods that automatically identify and prioritize regions of interest. Furthermore, our solutions are explicitly designed to allow integration into real-time pipelines for immediate visual feedback.

(Myrodiya, 2021) conducted multiple studies to examine the perception of different noise levels in images rendered with MC path tracing, discovering that participants primarily used their most central vision as well as non-textured and brightest areas of images to detect noise. Foveated rendering for path tracing in Virtual Reality (VR) uses eye-tracking to discover regions of higher interest to the viewer, which then a higher sample count can be devoted to. The basic concept is similar to our idea and could additionally be integrated when editing a scene in VR in case artists focus on a region we identified as having a lower impact. Similarly to foveated rendering with path tracing, we also exploit the fact that unbiased MC rendering allows to straightforwardly average partial results enabling local image areas with higher quality.

## 3 PRIORITIZED RE-RENDERING

We propose two variants for the priority-based re-rendering techniques with incremental updates. To allow integration in real-time pipelines, our solutions are designed as modules in Nvidia’s Falcor frame-

work for rapid prototyping of real-time techniques (Kallweit et al., 2022) (version 5.1.). The code can be found online (Ulschmid et al., 2023). Falcor employs the concept of render graphs consisting of render passes. Our methods adopt the Megakernel Path Tracer and accumulation pass to change between three modes of operation: global updates, continuous refinement, and incremental updates. A global update discards all previously accumulated samples and renders the new frame with 1 spp uniformly. During continuous refinement, per-pixel samples in each tile are accumulated by a moving average. Conventional re-rendering operates exclusively using these two modes, switching to the first whenever the scene is changed or the camera moves. By adding a third mode, we treat scene modifications separately by allowing to update the image in an incremental fashion. During the global updates and continuous refinement step, both variants employ Nvidia’s OptiX (Chaitanya et al., 2017) as a noise-reducing measure. Our methods have the same runtime complexity as conventional re-rendering (aside from a small delta for tile merging and denoising in ours, see below) and achieve real-time performance (30 frames per second at 1080p resolution) on state-of-the-art hardware.

### 3.1 Noisy Incremental Updates

To support incremental updates, we changed the Megakernel Path Tracer to operate on tiles representing disjoint parts of the scene in image space instead of the entire viewport. By repeating a selection of tiles multiple times in the input buffer of the (unbiased) path tracer, we can sample this region at a higher quality without having to adapt the underlying implementation, including random number generation. Our approach is designed to operate on a tile budget, thus we can redirect the original GPU workload (and overall complexity) of full-screen updates to just a few tiles.

After the path tracing procedure, we use a compute shader to aggregate the content of the duplicated tiles with a tree-based averaging reduction. This is possible since we target unbiased MC rendering, which allows for straightforward averaging of partial results to create higher-quality areas. Our approach thus yields a few high-quality tiles in roughly the same time it would take to render a whole scene of the same size at a low sample rate (not accounting for variations in scene complexity over the screen). The buffer layout we use is illustrated by Figure 2. Choosing a higher target tile quality results in fewer tiles being processed per update and thus a smaller region of the image that can be re-rendered with a given budget. The tile size directly influences the visuals and extent

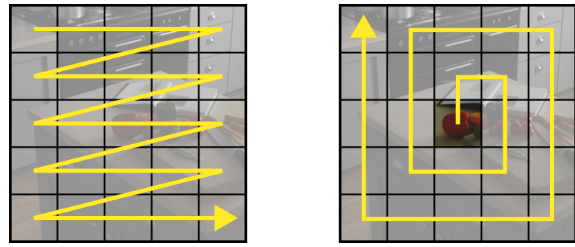


Figure 3: Left: tiles update simultaneously in resting scene. Right: a spiral determines the order of tile updates after edit. Note that the actual tiles are much smaller than depicted.

of each incremental update. As default values, we set the tile quality to 64 spp and the tile size to 16 pixels.

Falcor’s path tracing pipeline includes a sample accumulation stage, which aggregates color values over time as long as scenes remain unchanged and resets them to black otherwise. Consider the case of a small, localized scene edit. In our incremental approaches, only the previously processed tiles are replaced. The values in the other tiles are kept. This way, the regions of the image we identified as most affected by a modification get updated in high quality as soon as possible, while less influenced regions are re-rendered at a later time. Unless the scene is further modified, we switch to the default, full-screen accumulation mode after the entire screen has been processed once with incremental updates. In the case of further modifications, we restart the incremental procedure at the changed object’s screen position. During the global updates and continuous accumulation of samples, the tiles are processed from top to bottom and left to right. If the scene is modified and an incremental update is triggered, the order of tiles is computed by constructing a spiral starting at the object’s position projected to image space (see Figure 3).

Our applied importance metric thus defines the priority  $P$  of a tile  $t$  with center  $(x_t, y_t)$  in image space using the Chebyshev distance from the selected object  $o$  with center  $(x_o, y_o)$  as follows:

$$P(t) = \max(|x_o - x_t|, |y_o - y_t|) \quad (1)$$

### 3.2 Denoised Incremental Updates

In addition to the above method, we designed a “denoised” version. In addition to denoising during continuous refinement, it also applies the OptiX denoiser during incremental updates, resizing its domain to the space of already re-rendered tiles. However, as resizing and running a denoiser designed for full-image denoising can be costly, we introduce a configurable threshold parameter. A value of zero means the denoiser is resized and run each frame. Values above zero indicate the number of pixels the updated region

has to grow on either side before the denoiser is resized and executed. In order to not mix noisy and denoised tiles, we separately store two images: One containing all noisy updated tiles and another containing only the region that has so far been updated and denoised. The first one is the same as for the noisy incremental method. If the threshold value is reached, the denoiser is resized to and executed upon the entire region that has been updated so far. For temporal stability, the smaller, previous denoised region is then placed on top of the frame again.

Higher threshold values can lead to slightly more noticeable edges between the old and newly updated regions and less reactive, block-like visuals. Overall, however, it increases the performance of the method such that it only deviates from the noisy incremental approach by a minor delta that does not affect perceived responsiveness during editing. We used a threshold value of 200 as a default.

### 3.3 Interaction and User Control

To enable the user to modify the scene, we added an additional interaction pass to Falcor. It allows selecting one or multiple objects and exposes their transform and material properties via the *Dear ImGUI* user interface (UI). These entries can be manipulated via clicking and dragging or directly entering a value. Changes in the parameters are internally applied to the objects as an animation. The UI stage we added to the Falcor path tracing pipeline also communicates information to the rendering stages, such as selected object center and whether an interaction occurred.

Artifacts can occur when a user interrupts the incremental update procedure by continuously changing the object position with very high tile quality settings. If the resulting tile size is smaller than the object, moving it can result in ghosting artifacts, as parts of the object in its old position persist. In this case, the tile quality can be reduced or the tile size increased to process more tiles in one incremental update. Automating this parameter tuning is left as future work.

## 4 EVALUATION

Especially for smaller edits, reusing tiles from previous, converged images for re-rendering can significantly increase quality as measured by metrics, e.g., PSNR and PSNR-HVS-M, (Ponomarenko et al., 2007; Ponomarenko et al., 2011). Figure 4 illustrates this for the edit in Figure 1, also displayed by the video in the supplementary material. However, our main focus in this work lies on verifying whether

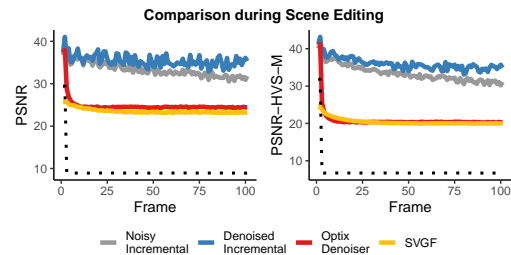


Figure 4: Comparison of different global and incremental methods during the edit in Figure 1 all starting at 1024 spp. The dotted line represents 1 spp baseline without denoising. As our method replaces converged tiles before the edit with new ones, image accuracy degrades gracefully while moving the cup.

such policies also result in an improved usability and interactive editing experience. Thus, we conducted an extensive user study to compare conventional and priority-based re-rendering.

### 4.1 User Study Overview

Referring to our three research questions stated in section 1, we investigate which rendering method artists prefer and if their preference depends on a specific editing scenario (RQ1). The following three scenarios are considered:

1. Small edits (modify localized geometry)
2. Large edits (modify large scene objects)
3. Modify a scene’s light sources

To evaluate the artists’ preferences, we instructed them to execute multiple tasks with global and incremental re-rendering methods under the scenarios listed above (see Table 1). Afterward, we let the participants rate the experienced combination of method and scenario to extract a participant-wise score. We start from the initial premise that edits on small objects primarily affect a local region of the scene and it might thus be obstructive for a fluid workflow to re-render the entire image. On the other hand, light source edits mostly have a global influence and are more likely to require a full image update to convey a complete impression of their effect. Thus, one goal of this study is to evaluate the following two hypotheses:

---

Artists prefer (w.r.t. perceived workload and focusability, measured as a participant-wise score)

1. an incremental rendering method to state-of-the-art solutions (i.e., denoised full-screen updates) when editing small objects.
  2. a global update approach when manipulating light sources.
-

We test these hypotheses via ANOVA and pairwise t-tests. By performing an a-priori power analysis using G\*Power (Faul et al., 2007), we calculated the number of participants needed for a power of  $1 - \beta = 0.8$  with a medium effect of  $\eta_p^2 = 0.06$  as 15 for an ANOVA (repeated measures, within factors) and as 34 for a two-tailed, paired t-test with a medium effect of  $d = 0.5$ .

The preference of users when editing larger objects as well as the remaining aspects are researched in an exploratory fashion. We therefore analyse central tendencies and effect sizes to identify trends in the pairwise differences between the scores. To identify which kind of updates are preferred overall (RQ2), we compute a favored method based on the aforementioned score via majority vote. Furthermore, we directly asked the participants to select their favorite method, as well as to rate them based on whether they would buy them, to construct a metric similar to a Net Promoter Score (NPS). To answer what is the general attitude towards path tracing for scene editing (RQ3), we asked questions about the frequency of working with path tracing and reasons for not using it during scene editing. Moreover, we conducted structured interviews with the artists at the end of the study, asking why they preferred a method, how they estimate the future relevance of physically-based rendering in their field of work, and what they would consider helpful during scene editing with real-time path tracing.

## 4.2 Technical Setup and Participation

The survey was implemented as a self-hosted Drupal Webform and the full questionnaire, as well as the anonymized collected data and R code used for the evaluation, can be found online (Ulschmid et al., 2023). We use three different scenes from the Bitterli Rendering Resources (Bitterli, 2016): *The Contemporary Bathroom*, *the Country Kitchen*, and *The Grey & White Room*. We refer to them simply as *Bathroom*, *Kitchen*, and *Living Room*.

Since hardware-accelerated path tracing requires an Nvidia RTX GPU or comparable, we offered the artists three ways to participate: They could attend in person to perform the study on one of our PCs with an Nvidia RTX 3070 Ti, or online by either executing the program on their own hardware or streaming from our PC by remote controlling it via Parsec (Parsec Cloud Inc., ). Under all three circumstances, we recorded screen and audio for the duration of the study, to make sure the experiment conditions were comparable for all participants, as well as for further evaluation. In the online cases, we supervised the experiment via internet telephony while participants working on their

Table 1: Number of tasks per edit type and scene. Transformational edits include moving and rotating objects. Material edits involve changing albedo and roughness values of objects or emissive parameters of lights/environment maps.

Scene	Edit	Small Objects	Large Objects	Lights
<i>Kitchen</i>	Transform	1	1	-
	Material	1	1	2
<i>Living Room</i>	Transform	1	1	-
<i>Room</i>	Material	1	-	1
<b>Sum</b>		<b>4</b>	<b>3</b>	<b>3</b>

own hardware instead of streaming from ours were sharing their screens.

The artists were asked to voice their thoughts during the study, and we conducted a structured interview directly after the experiment. For evaluation purposes, the voice recordings were transcribed using OpenAIs speech-to-text system Whisper (Radford et al., 2022).

Throughout the entire questionnaire as well as in the Falcor framework we obfuscated the names of the rendering methods using letters. We also hid all UI elements except the needed transformational, material, and rendering parameters.

## 4.3 User Study Design

For the representative of conventional re-rendering with denoising, we use the AI-accelerated Nvidia OptiX denoiser (Chaitanya et al., 2017), as it produced higher-quality results than SVGF in our quantitative analysis, while still being interactive.

Before beginning the experiment, participants were asked to sign a consent form. The study itself consists of three parts: an introductory tutorial, the task-based evaluation of the three methods (denoiser, noisy incremental, denoised incremental) in three different editing scenarios (small objects, large objects, lights), as well as some general and demographic questions at the end. We intentionally searched for artists from diverse backgrounds and different experience levels to achieve better generalizability.

**Tutorial.** The tutorial explained these necessary elements of the UI to the participants (see section 3) and allowed them to practice by executing four exemplary tasks in the *Bathroom* scene using the basic 1 spp rendering without any additional features enabled. The artists were also instructed to use predefined camera viewports but were allowed to move the camera if they really needed it to not interrupt their artistic workflow. However, we ensured that they would always stay at the same distance from the object. The

Table 2: Format of the Likert items for the adapted scale used for the extended NASA-TLX questions and their numerical mapping applied during the evaluation.

Worse than 1spp	Same as 1spp	Slightly better than 1spp	Better, but still flawed	Mostly like converged	Almost equivalent	Same as converged
-1	0	1	2	3	4	5

tutorial also provided a short text explaining the current limitations of real-time path tracing and showed two prerecorded videos of a cup moving in the *Bathroom* scene, one with a 1 spp noisy re-rendering and one with 4096 spp plus denoising with OptiX. The latter was supposed to serve as a reference for an imaginary approach that instantly converges.

**Task-based Evaluation.** During the task-based evaluation, participants performed multiple editing tasks in both the *Kitchen* and *Living Room* scene (see Table 1) for each of the three editing scenarios and the three rendering methods, yielding nine combinations in total. After each condition, the artists were asked to rate the experienced method. For the incremental approaches, we additionally asked which rendering parameters (tile quality/size, threshold) they changed and which settings they preferred (see subsection 3.1).

The order of the tasks and scenarios was fixed, whereas we randomized the order of the rendering methods so participants would either first encounter the conventional or the incremental techniques (in the order noisy then denoised).

To rate the experienced combination of scenario and method, we used a questionnaire based on the standardized NASA Task Load Index (NASA-TLX) (Hart and Staveland, 1988), which is comprised of six questions measuring the perceived workload of tasks. We added two questions to assess focus and distraction and dropped the question on physical demand as we deemed it less relevant for our purposes, resulting in seven questions in total.

**Scale Design.** After conducting a small pilot study with four participants, we also decided to change the scale of the answers from the original range with 21 gradations to Likert items with seven levels as can be seen in Table 2 following the construction guidelines in (South et al., 2022). The pilot participants were confused about the comparison point to use for the first encountered method. In response to this, we introduced the tutorial, where they experience the conventional 1 spp noisy re-rendering approach themselves and watch a video of how one of the editing tasks would look like if we could produce high-quality renderings in real-time. We requested the artists to rate the experienced rendering methods

in comparison to the basic 1 spp approach and the imaginary converged solution. In using these descriptions for our adapted seven-point scale, the participants were provided with the reference points they reported as missing during the pilot study.

## 4.4 Results

We use an alpha level of  $\alpha = 0.05$  for all statistical tests and, in case of multiple testing, report adjusted p-values for better readability. As effect sizes we report, depending on applicability, Cohen’s  $d$  ( $< 0.5$  small,  $[0.5, 0.8]$  medium,  $> 0.8$  large), Cramér’s  $V$  (0.1 small, 0.3 medium, 0.5 large) and the generalized eta squared  $\eta_g^2$  (Cohen, 1988). As we did not find any similar studies on our topic, we use the standard interpretation of  $\eta^2$  for  $\eta_g^2$  ( $< 0.06$  small,  $[0.06, 0.14]$  medium,  $> 0.14$  large).

For the task-based evaluation, following the original formulation of Likert-typed scales (South et al., 2022), we aggregated the extended NASA-TLX questions by averaging the individual Likert items. The aggregation is computed for each combination of scenario and method on participant level, resulting in nine scores for each participant. This procedure is also referred to as Raw TLX (Hart, 2006). As the same participant ranked all of the combinations during the study and the individual ratings are thus dependent on the subject, we use paired or repeated measure methods.

**Study Population.** Out of the 21 artists ( $m=15, f=4, d=2$ ) who completed the survey, ten were professionals from the industry, and eleven were students from universities. Their experience measured in hours spent using 3D rendering software ranged from ‘50-99’ to ‘10,000+’ hours (Median  $Mdn$ : ‘1,000-4,999’, interquartile range  $IQR$ : [‘100-499’, ‘5,000-9,999’]) and participants’ ages ranged from 21 to 51 (Mean  $M$ : 30.48, standard deviation  $SD$ : 7.79).

Eleven executed the tasks with the denoiser first and ten always the incremental methods first. Eight artists participated via Parsec or on their own hardware, while five performed the tasks in person at our office. The results of further questions about the study population are reported in Table 3. Most artists completed the study in roughly one hour.

**Assumption Testing.** The applied statistical techniques require the data to fulfill the following assumptions: no significant outliers, normality, and sphericity. The sphericity assumption requires that the variances of the pairwise differences are equal. To test



Pairwise Differences between the Rendering Methods by Editing Scenario

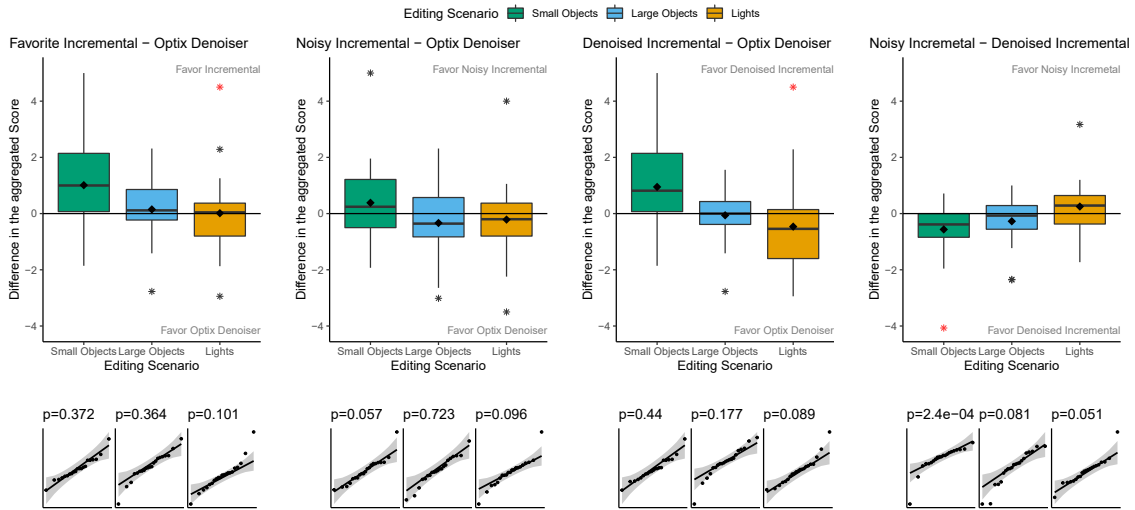


Figure 5: Top row: Box plots of the pairwise differences in the averaged score of the task-based evaluation. If the difference between two methods is positive, the method named first was favored. If it is negative, the method named second was favored. The respective means are depicted with a diamond shape and outliers as stars, whereas extreme outliers are colored red. Bottom row: quantile-quantile (QQ) Plots of the pairwise differences by editing scenario and results of Shapiro-Wilk tests. If the  $p$ -value is larger than the assumed significance level of  $\alpha = 0.05$ , we may assume a normal distribution.

Table 3: Artistic background of the study population. For all questions, except the path tracer usage, selecting multiple answers was possible.

Results of the general questions					
Field of Work	Used Denoisers	Path Tracer Usage			
Games	14	OptiX	12	Never	4
Digital Art	11	Intel	5	Occasionally	9
Animation	9	None	2	Regularly	7
Architecture Vis.	6	Vdenoise	2	Every time	1
Other	9	Other	5		
Used Software	Reasons for not using a Path Tracer				
Blender	18	Waiting time for clear image too long	16		
Unity 3D	15	Real-time approx. good enough	10		
Unreal Engine	10	Used software has no Path Tracer	7		
Maya	9	Only using for final image	6		
3ds Max	7	Initial noise/blurriness is distracting	5		
ZBrush	6	Necessary hardware not available	2		
Other	14				

this we use Mauchly’s Test, which is significant ( $p = 0.041$ ,  $\epsilon = 0.743$ ), meaning the assumption is violated.

There are no outliers on the aggregated values. However, when looking at the pairwise differences between the methods by participant and scenario depicted in Figure 5, we found two significant outliers: one on the difference between noisy incremental and denoised incremental for small objects and one on the difference between the denoiser and denoised incremental for lights. As removing the respective values led to similar overall results, we left the outliers in the data for the main analysis, but reported both results where appropriate.

To test if the data is normally distributed we use a Shapiro-Wilk test on the pairwise differences. The

resulting  $p$ -values, as well as QQ plots, can be seen in Figure 5. The only pairwise difference violating the normality assumption is between noisy incremental and denoised incremental for small objects. By looking at the QQ plot we identify the same outlier as above as being responsible. Removing it would lead to a value of  $p = 0.527$ , however with the same reasoning as before we decided to leave it in the data.

**ANOVA and t-Test Results.** To analyze the interaction between the rendering method and editing scenario on the score aggregated by participants, we performed a two-way repeated measures ANOVA with the scenario as the focal variable and the used method as the moderator variable. As the sphericity assumption is violated, we apply the conservative Greenhouse–Geisser correction. We found a statistically significant interaction between the rendering approach and editing scenario on the aggregated value ( $F(2.97, 59.42) = 5.29$ ,  $p = 0.003$ ,  $\eta_g^2 = 0.03$ ).

Therefore, we analyzed the effect of the used method for each scenario, reporting  $p$ -values adjusted using the Bonferroni multiple testing correction method. Using one-way ANOVAs, the simple main effect of the rendering approach was found to be only significant for small objects ( $p = 0.03$ ,  $\eta_g^2 = 0.06$ ). Pairwise comparisons using paired t-tests show that for small objects, the mean aggregated value was significantly different between using the denoiser and the denoised incremental method ( $p = 0.026$ ,  $d = 0.63$ ) and between the noisy and denoised incremental

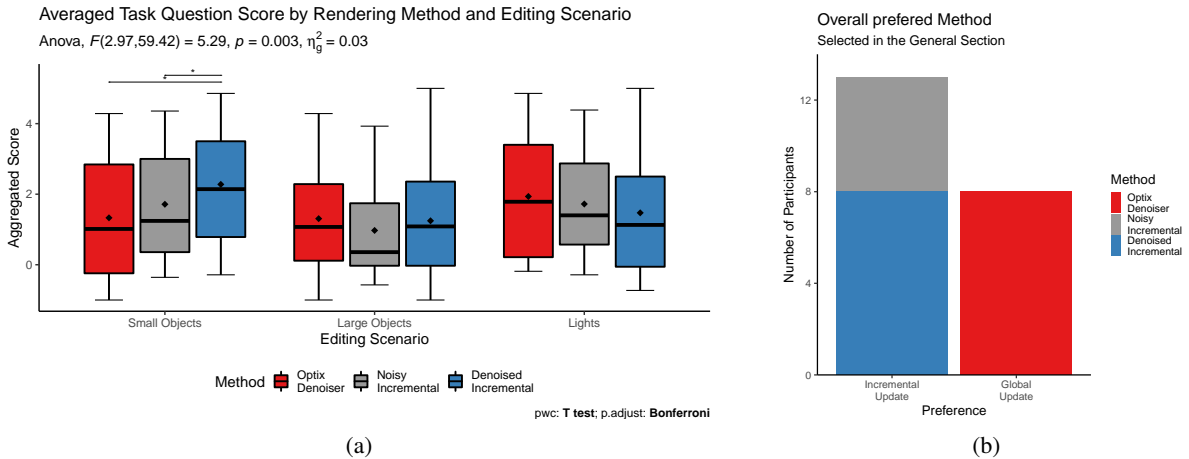


Figure 6: (a) Box plot of the aggregated score of the task-based evaluation part of the survey (♦: mean, \*:  $\alpha < 0.05$ ). (b) Bar plot showing the favorite rendering approach selected in the general section at the end of the questionnaire.

method ( $p = 0.05$ ,  $d = 0.57$ ; with outlier removed  $p = 0.025$ ,  $d = 0.66$ ). There is no significant difference for light edits between using the denoiser and the noisy ( $p = 1$ ,  $d = 0.15$ ) or denoised ( $p = 0.69$ ,  $d = 0.27$ ; with outlier removed  $p = 0.079$ ,  $d = 0.54$ ) incremental rendering technique (see Figure 6a).

**Analysis of Differences.** When looking at the pairwise differences between the rendering methods at per-participant level for each scenario depicted in Figure 5, we observed a trend in central tendency between the OptiX denoiser and the denoised incremental method that the participants of our study seemed to favor the denoised incremental method for small objects ( $M: 0.948$ ,  $SD: 1.50$ ) and the denoiser for light edits ( $M: -0.464$ ,  $SD: 1.72$ ). By looking at the differences between the noisy and denoised incremental approach, however, it can be noticed that while for small objects the denoised method was preferred ( $M: -0.564$ ,  $SD: 0.988$ ), for light edits the noisy method was favored slightly more ( $M: 0.25$ ,  $SD: 0.958$ ).

To analyze in a more general fashion whether a global or incremental update method is preferred, we extracted the favorite incremental method per participant and scenario by choosing the one with the highest aggregated score. We then compared this favorite to the denoiser. The results for edits on small objects approximately remain the same, whereas for light edits the differences become closer to zero ( $M: 0.014$ ,  $SD: 1.53$ ). When editing large objects, the differences between the rendering approaches are roughly zero-centered for all combinations.

**Overall Preference.** We furthermore computed which rendering method artists preferred overall according to the task-based evaluation. For this, we first

extracted the favored method for each of the three editing scenarios by taking the one with the highest aggregated score. Then the overall favorite is determined by a majority vote. The results can be seen in Figure 7a. Most participants preferred incremental over global updates, which is confirmed by directly asking the artists at the end of the study as well, as can be seen in Figure 6b. There is, however, no statistically significant relation between the two favorites (Fisher’s exact test (FET),  $p = 0.46$ ,  $V = 0.37$ ). We identified a significant association between the favorite for the light editing scenario and the overall preferred method stated at the end of the survey (FET,  $p = 0.043$ ,  $V = 0.47$ ). The majority vote favorite is also significantly associated with the preference for edits on large objects (FET,  $p = 0.004$ ,  $V = 0.65$ ) and lights (FET,  $p = 0.004$ ,  $V = 0.68$ ). These values are illustrated in Figure 7c.

**Net Promoter Score.** To compute a metric similar to a NPS based on the question if artists would buy the respective rendering approach as a feature in a 3D software, we subtracted the percentage of participants rating higher than seven on a scale from zero to ten from those rating lower than six. We decided to use a slightly lower threshold than for a normal NPS, as the overall ratings were also rather low. The computed scores are depicted in Figure 7b.

**Parameter Settings.** Regarding the rendering parameters, participants favored higher tile qualities for editing smaller objects (noisy  $Mdn: 64$ , denoised  $Mdn: 128$ ) and lower quality for larger objects and light edits ( $Mdn: 16$ ). The median setting for tile size was 16, except for the denoised incremental method in the small objects scenario, where it was 24. For

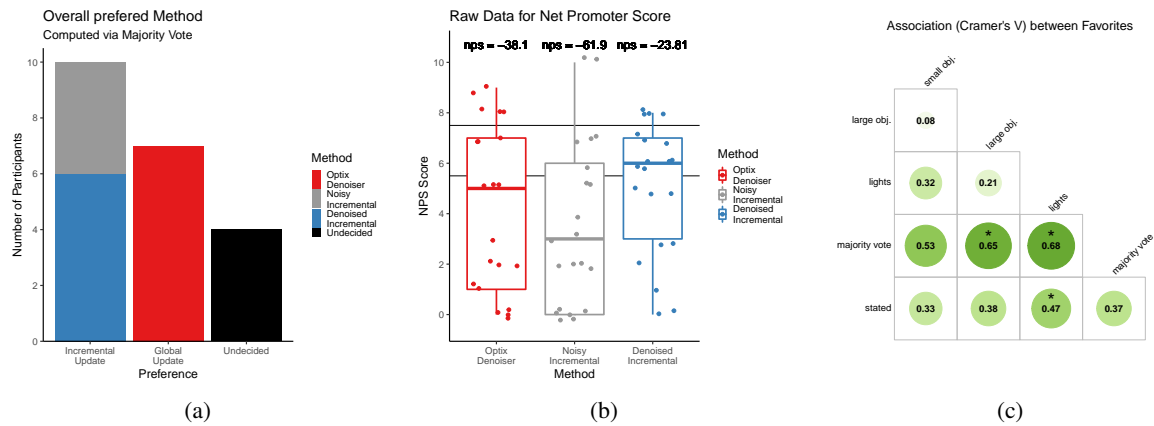


Figure 7: (a) Overall preferred rendering approach of the artists extracted by a majority vote between the three methods with the highest average score for each editing scenario. (b) Box plot showing the distribution of ratings when asking artists whether they would buy the respective rendering method as a feature in a 3D software. Based on these ratings we computed a metric similar to a NPS, which is stated on top of the box plots. (c) Association computed via Cramer's V between the favorite method for a certain task derived from the averaged scores (Figure 6a) and the overall favorite either calculated via a majority vote or stated by the artists themselves in the survey.

the threshold parameter, larger values were preferred (*Mdn*: '101-200').

## 5 DISCUSSION & CONCLUSION

Regarding our first research question **RQ1**, we found a statistically significant increase in a score measuring participant-wise preference for using the denoised incremental approach when editing small objects by performing an ANOVA and pairwise t-tests. For larger objects, no clear favorite method could be determined. In the case of editing the lighting of a scene, there is a minor, however not statistically significant trend in central tendencies to favor the technique with global updates. The study is slightly underpowered w.r.t. the a-priori power analysis for pairwise t-tests. However, we observed larger effect sizes than assumed.

Overall and in answer to our second research question **RQ2**, rendering with incremental updates seems to be favored according to a preference both computed via majority vote between the three investigated scenarios and extracted by directly asking the participants at the end, as well as to a NPS-like metric. On the other hand, artists mentioned during a structured interview that none of the methods perfectly caters to their needs so far and they would thus still mainly edit scenes using less ideal real-time approximations, resolving **RQ3**. However, the future relevance of accurate, physically based methods for final render results is still high for the movie industry and even infiltrates high-performance businesses such as computer games.

In summary, we presented two adaptive, priority-based re-rendering approaches designed for interactively editing a 3D scene with MC path tracing. Using a tile-based approach, we can efficiently update image regions with higher quality in an incremental fashion, applying an importance metric based on the Chebyshev distance. Our quantitative and qualitative evaluation shows that for localized edits, in particular, our incremental approach reduces noise levels and is preferred by artists. While we tried to choose a diverse study population with different backgrounds, as well as various editing scenarios in multiple scenes, our results may not generalize. Future work could extend the building blocks of the incremental methods by combining them with higher quality denoisers, such as Intel's OIDN (Intel, 2019). Furthermore, the priority policy could be refined, based on fast approximations of global illumination, which would allow for better incremental updates for indirect (lighting) effects. Gradient-based techniques and difference images could be integrated to enable automatic parameter selection (Rousselle et al., 2016). Overall, our evaluation provides a strong motivation to pursue further methods for automatic, priority-based, and interactive re-rendering techniques for creative processes.

## ACKNOWLEDGEMENTS

This research was funded by FWF project F77 (SFB *Advanced Computational Design* SP4) and by WWTF project ICT22-055 (*Instant Visualization and Interaction for Large Point Clouds*). We also thank our students Pascal Hann and Nina Semmelrath for their contributions.

## REFERENCES

- Bitterli, B. (2016). Rendering resources. <https://benedikt-bitterli.me/resources/>.
- Chaitanya, C. R. A., Kaplanyan, A. S., Schied, C., Salvi, M., Lefohn, A., Nowrouzezahrai, D., and Aila, T. (2017). Interactive reconstruction of monte carlo image sequences using a recurrent denoising autoencoder. *ACM Trans. Graph.*, 36(4).
- Cohen, J. (1988). *Statistical Power Analysis for the Behavioral Sciences*. Lawrence Erlbaum Associates, 2nd edition.
- Faul, F., Erdfelder, E., Lang, A.-G., and Buchner, A. (2007). G\*power 3: A flexible statistical power analysis program for the social, behavior, and biomedical sciences. *Behavior Research Methods Instruments & Computers*, 39:175–191.
- Günther, T. and Grosch, T. (2015). Consistent Scene Editing by Progressive Difference Images. *Computer Graphics Forum*.
- Hachisuka, T., Jarosz, W., Weistroffer, R. P., Dale, K., Humphreys, G., Zwicker, M., and Jensen, H. W. (2008). Multidimensional adaptive sampling and reconstruction for ray tracing. *ACM Trans. Graph.*, 27(3):1–10.
- Hart, S. (2006). Nasa-task load index (nasa-tlx); 20 years later. volume 50.
- Hart, S. G. and Staveland, L. E. (1988). Development of nasa-tlx (task load index): Results of empirical and theoretical research. In *Human Mental Workload*, volume 52 of *Advances in Psychology*, pages 139–183.
- Hasselgren, J., Munkberg, J., Salvi, M., Patney, A., and Lefohn, A. (2020). Neural temporal adaptive sampling and denoising. *Computer Graphics Forum*, 39:147–155.
- Intel (2019). Open image denoise. <https://www.openimagedenoise.org/>.
- Kajiya, J. T. (1986). The rendering equation. *SIGGRAPH Comput. Graph.*, 20(4):143–150.
- Kallweit, S., Clarberg, P., Kolb, C., Davidovič, T., Yao, K.-H., Foley, T., He, Y., Wu, L., Chen, L., Akenine-Möller, T., Wyman, C., Crassin, C., and Benty, N. (2022). The Falcor rendering framework. <https://github.com/NVIDIAGameWorks/Falcor>.
- Kuznetsov, A., Kalantari, N. K., and Ramamoorthi, R. (2018). Deep adaptive sampling for low sample count rendering. *Computer Graphics Forum*, 37.
- Mitchell, D. P. (1987). Generating antialiased images at low sampling densities. *SIGGRAPH Comput. Graph.*, 21(4):65–72.
- Murakami, K. and Hirota, K. (1992). *Incremental Ray Tracing*, pages 17–32. Springer Berlin Heidelberg.
- Myrodia, V. (2021). *Psychophysical studies on Monte Carlo rendering-noise visual perception*. PhD thesis.
- Nvidia (2018). NVIDIA TURING GPU ARCHITECTURE. Technical report.
- NVIDIA (2021). Nvidia real-time denoisers. <https://developer.nvidia.com/rtx/ray-tracing/rt-denoisers>.
- Overbeck, R. S., Donner, C., and Ramamoorthi, R. (2009). Adaptive wavelet rendering. *ACM Trans. Graph.*, 28(5):1–12.
- Parsec Cloud Inc. Parsec. <https://parsec.app/>.
- Pharr, M., Jakob, W., and Humphreys, G. (2016). *Physically Based Rendering: From Theory to Implementation*. Morgan Kaufmann Pub. Inc., 3rd edition.
- Ponomarenko, N., Ieremeiev, O., Lukin, V., Egiazarian, K., and Carli, M. (2011). Modified image visual quality metrics for contrast change and mean shift accounting. In *The Experience of Designing and Application of CAD Systems in Microelectronics*, pages 305–311.
- Ponomarenko, N., Silvestri, F., Egiazarian, K., Carli, M., Astola, J., and Lukin, V. (2007). On between-coefficient contrast masking of dct basis functions. *Proc. of the 3rd Int. Workshop on Video Processing and Quality Metrics for Consumer Electronics*.
- Radford, A., Kim, J. W., Xu, T., Brockman, G., McLeavey, C., and Sutskever, I. (2022). Robust speech recognition via large-scale weak supervision.
- Rousselle, F., Jarosz, W., and Novák, J. (2016). Image-space control variates for rendering. *ACM Trans. Graph.*, 35(6).
- Schied, C., Kaplanyan, A., Wyman, C., Patney, A., Chaitanya, C. R. A., Burgess, J., Liu, S., Dachsbacher, C., Lefohn, A., and Salvi, M. (2017). Spatiotemporal variance-guided filtering: Real-time reconstruction for path-traced global illumination. In *Proc. of High Performance Graphics*. ACM.
- Schied, C., Peters, C., and Dachsbacher, C. (2018). Gradient estimation for real-time adaptive temporal filtering. *Proc. ACM Comput. Graph. Interact. Tech.*, 1(2).
- Schmidt, T.-W., Pellacini, F., Nowrouzezahrai, D., Jarosz, W., and Dachsbacher, C. (2014). State of the art in artistic editing of appearance, lighting, and material. *Computer Graphics Forum*, 35.
- South, L., Saffo, D., Vitek, O., Dunne, C., and Borkin, M. A. (2022). Effective Use of Likert Scales in Visualization Evaluations: A Systematic Review. *CGF*.
- Thomas, M. M., Liktov, G., Peters, C., Kim, S., Vaidyanathan, K., and Forbes, A. G. (2022). Temporally stable real-time joint neural denoising and super-sampling. *Proc. ACM Comput. Graph. Interact. Tech.*, 5(3).
- Ulschmid, A., Kerbl, B., Krösl, K., and Wimmer, M. (2023). <https://github.com/cg-tuwien/Prioritized-ReRendering>.
- Wyman, C. and Pantelev, A. (2021). Re-architecting spatiotemporal resampling for production. In *High-Performance Graphics - Symposium Papers*. The Eurographics Association.
- Zhdan, D. (2021). Reblur: A hierarchical recurrent denoiser. In *Ray Tracing Gems II*, pages 823–844.
- Zwicker, M., Jarosz, W., Lehtinen, J., Moon, B., Ramamoorthi, R., Rousselle, F., Sen, P., Soler, C., and Yoon, S.-E. (2015). Recent advances in adaptive sampling and reconstruction for monte carlo rendering. *Comput. Graph. Forum*, 34(2):667–681.