Computational Visual Media
https://doi.org/10.1007/s41095-018-0113-0 Vol. 4, No. 3, September 2018, 267–276

Research Article

# Instantaneous foveated preview for progressive Monte Carlo rendering

**Matias K. Koskela[1] (✉), Kalle V. Immonen[1], Timo T. Viitanen[1], Pekka O. Jääskeläinen[1], Joonas I. Multanen[1], and Jarmo H. Takala[1]**

**Abstract** Progressive rendering, for example Monte Carlo rendering of 360° content for virtual reality headsets, is a time-consuming task. If the 3D artist notices an error while previewing the rendering, they must return to editing mode, make the required changes, and restart rendering. We propose the use of eye-tracking-based optimization to significantly speed up previewing of the artist's points of interest. The speed of the preview is further improved by sampling with a distribution that closely follows the experimentally measured visual acuity of the human eye, unlike the piecewise linear models used in previous work. In a comprehensive user study, the perceived convergence of our proposed method was 10 times faster than that of a conventional preview, and often appeared to be instantaneous. In addition, the participants rated the method to have only marginally more artifacts in areas where it had to start rendering from scratch, compared to conventional rendering methods that had already generated image content in those areas.

**Keywords** foveated rendering; progressive rendering; Monte Carlo rendering; preview; 360° content

## 1 Introduction

Virtual reality (VR) is increasingly used for both work and entertainment. One challenge posed by VR is the generation of 360° content, especially due to the high resolution requirements of VR devices, and the need for meaningful interesting content in every direction in 3D space. Rendering high resolution images with progressive photorealistic methods typically takes hours to complete, while approximate preliminary results can be produced much faster. Moreover, in *Monte Carlo* rendering, halving the error in the rendered images requires quadrupling the number of rendered samples [1]: the payoff obtained from additional rendering time reduces quickly.

If the artist notices during previewing that something is wrong with the scene, they must abort rendering, make the required changes, and restart rendering all over again. Restarting the rendering process from scratch is required: for example, changing the illumination conditions potentially affects every pixel of the image. In many cases, the artist can create an approximation of the scene with a faster rendering method, but it typically lacks photorealistic effects such as reflections and indirect lighting, which require slow, offline methods to render. If the artist can preview the rendering sooner, it directly improves the speed of the content creation process.

In this paper, we propose a method for speeding up the preview of progressively rendered images by applying *foveated rendering* to reduce the quality in the peripheral regions of vision. Quality can be reduced because visual acuity decreases with increasing eccentricity, as a consequence of drop in the density of rod and cone cells in the retina off-axis [2]. It has been estimated that more than 90% of real-time path tracing samples can be omitted by employing foveated rendering [3].

1 Tampere University of Technology, Tampere, 33720, Finland. E-mail: M. K. Koskela, matias.koskela@tut.fi (✉); K. V. Immonen, kalle.immonen@tut.fi; T. T. Viitanen, timo.2.viitanen@tut.fi; P. O. Jääskeläinen, pekka.jaaskelainen@tut.fi; J. I. Multanen, joonas.multanen@tut.fi; J. H. Takala, jarmo.takala@tut.fi.

Manuscript received: 2017-12-23; accepted: 2018-02-17

We make the following novel contributions in this article:

1. an optimized human visual acuity model, which can be used to accurately generate path tracing samples;
2. an evaluation of the benefits of foveated rendering in speeding up progressive rendering previews, validated by a user study showing that the proposed method is 10 times faster than a conventional previewing approach.

The results of the proposed preview framework are shown in Fig. 1.

We extend our previous work [4] with an additional evaluation of the questionnaire presented to the participants of the user study, and an evaluation of how the participants assessed artifacts in the results.

## 2 Related work

The idea in foveated rendering is to adapt the rendered visual quality to the physiological abilities of the human visual system. Foveated rendering requires predicting or measuring the direction of the user's gaze. Consequently, a real-time requirement is imposed on rendering. There is a large body of work on real-time foveated rendering, which is summarized by a recent comprehensive literature review by Weier et al. [5]. Foveated rendering is very appealing when using *head-mounted displays* (HMDs), which typically have a wider *field of view* (FOV) than desktop monitors, and only a single observer per display [6]. The wider FOV means that the user can see clearly only a proportionally smaller area of the screen. In addition, HMDs require low latency rendering to reduce motion sickness, which calls for greater optimization than for a desktop setup. Moreover, accuracy of eye tracking is better with an HMD setup because the camera used to measure the gaze direction is fixed to the head of the user [7].

One method to perform foveated rendering is to rasterize the scene at multiple resolutions [8]. The system renders only the region centered on the gaze direction at the highest resolution, and uses larger pixels in the user's peripheral vision. Another approach is to include foveated rendering into a deferred shading pipeline by shading only some pixels, and by interpolating results for the remainder of the pixels [9, 10].

Current hardware supports only a fixed, predefined resolution for rasterization. Therefore, foveated rendering can be implemented more easily with ray-tracing-based techniques because they support arbitrary sampling patterns in screen space. Consequently, foveated ray tracing has gained academic interest in recent years [11, 12]. An intuitive idea is to distribute samples according to the smallest detectable spatial frequency according to a model of human visual acuity:

$$m(e) = \begin{cases} 1.0, & 0 \leqslant e \leqslant 5.79 \\ 7.49/(0.3e+1)^2, & e > 5.79 \end{cases} \tag{1}$$

where $e$ is the eccentricity angle, i.e., the angle from the gaze direction [13]. This model is derived

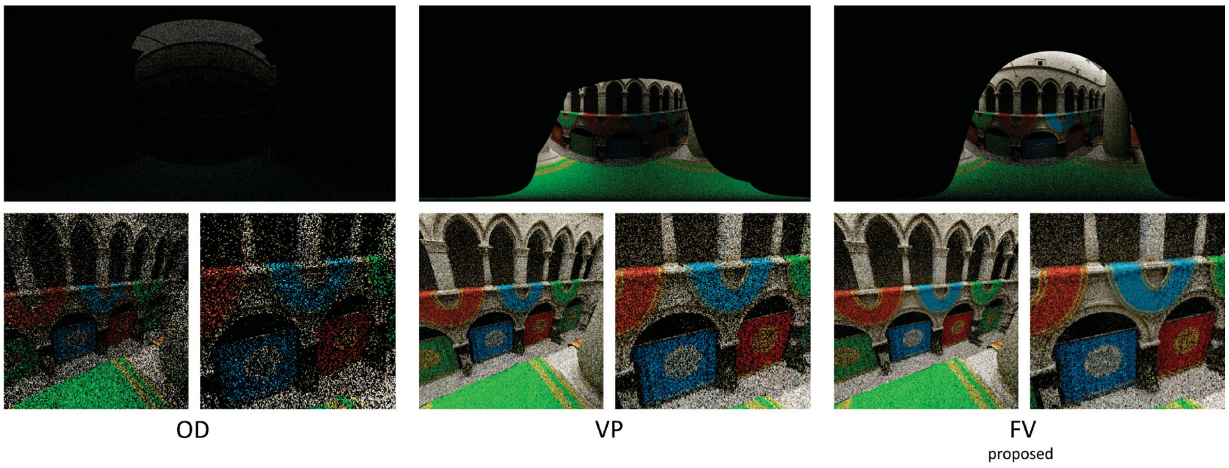


**Fig. 1** Results after two seconds of rendering with a static point of interest. Above: rendering buffer. Below: preview on screen, and close-up of the point of interest. Note how it already starts to converge in our proposed foveated preview approach (FV) which uses eye tracking and a human visual acuity model. On the other hand, the edges in FV are noisier than when uniformly sampling the viewport area (VP). Uniform sampling of the whole 360° image (OD) is noisier than the other methods.

from various psychophysical studies. The equation describes just one radius of visual acuity, and the actual 2D model is obtained by taking a solid of revolution determined by the equation, where the axis of revolution is at $e = 0$.

Due to the complexity of the visual acuity model, linear denominator models may be used instead of the quadratic denominator model in Eq. (1). However, they are not as accurate in the peripheral parts of vision [8]. A simplified version uses a linear fall-off between the maximum and the minimum sampling probability [9–11], or even a static probability with respect to gaze direction [14].

When previewing progressively rendered images, rendering of the region of interest needs to converge as quickly as possible to allow the artist to abort the rendering as soon as possible, when needed, and to make the required changes sooner. One way to vary the convergence rate is to apply a so-called *guided preview*, and have more samples in an area chosen by the artist with a pointing device [15]. Another idea is to select an area of the image where the sample computation is concentrated [16]. Importance masking [17] is an advanced version of area selection.

In this paper, we utilize the idea of a guided preview, and use one of the most intuitive kinds of guidance: the point at which the user is looking. This means that there is no need to manually select the region of interest, and instead the system automatically detects the user's point of interest. Moreover, we use the quadratic denominator visual acuity model instead of coarser models. Compared to coarser models, the more accurate model places fewer samples in the peripheral regions of vision, and therefore allows faster convergence in the fovea. In addition, previous work on foveated rendering has concentrated on real-time rendering, while we propose its use to preview off-line rendering.

## 3 Proposed method

The aim of our previewing method is to render images for VR and to give the artist an instant preview. The method tracks the eye of the user and generates samples according to the visual acuity model. Doing so does not worsen the user experience because resolution can be reduced significantly in the peripheral parts of vision without affecting search task performance [18]. In other words, the user can find the area of interest in equal time compared to when using a conventional preview. However, the area of interest converges to the final result significantly faster than when the image is uniformly sampled.

Sampling the world according to a visual acuity model requires random image space positions to be generated with probability density given by Eq. (1). Note that the equation from Reddy [13] is not directly usable as a probability density function because its integral over the entire space is not equal to one. We show later how to transform the equation to fulfil the constraint in Eq. (6).

Progressive rendering produces the correct color only after averaging many samples. Instead of clamping the model to one sample per pixel, we keep its value as cycles per degree. Using cycles per degree makes sure that the image converges quickly in the gaze direction. In other words, more than one sample may be placed into a single pixel during rendering of the frame. This in turn ensures, for example, that better anti-aliasing occurs faster in the area of interest. Due to the probabilistic nature of sampling, some pixels may be completely unsampled for the first few frames. While unsampled areas could be reconstructed [19], because the pixels are likely to be sampled quickly thereafter, we do not attempt to reconstruct the missing pixels.

Generating random numbers analytically according to the solid of revolution of Eq. (1) is not feasible for the targeted real-time preview method. Therefore, we simplify the generation by using polar coordinates: a uniformly distributed angular coordinate $\phi$, and a radial coordinate $r$, which is the distance from the center of the vision, i.e., eccentricity angle $e$. The angle $\phi$ can be generated by one of the many algorithms available for quickly generating uniformly distributed random numbers. To achieve correct distribution for $r$, the probability density of Eq. (1) must be modified based on the circumference of circle $2\pi R$ (where $R$ is the radius):

$$g(e) = 2\pi e m(e) = \begin{cases} 2\pi e, & e \leqslant 5.79 \\ 14.98\pi e/(0.3e+1)^2, & e > 5.79 \end{cases} \tag{2}$$

A uniform distribution can be transformed to any other distribution using the *inversion method* [20]:

$$r = f^{-1}(u) \tag{3}$$

where $u$ is a uniformly distributed random number in $[0, 1]$, $f$ is the desired cumulative distribution function, and $r$ is a random number with cumulative distribution $f$. The inversion method requires us to derive the cumulative distribution function from the probability density defined in Eq. (2) by integrating $g(e)$ over the interval $[0, x]$:

$$h(x) = \int_0^x g(e)\mathrm{d}e \tag{4}$$

so

$$h(x) = \begin{cases} \pi x^2, & x \leqslant 5.79 \\ \left(\dfrac{1}{0.3x+1} + \ln(0.3x+1)\right) & x > 5.79 \\ \quad \times 166.4\pi - 612.3, & \end{cases} \tag{5}$$

We chose the upper limit of the function at an eccentricity angle of 80° because at that point the model begins to reach zero. In addition, such an angle suffices to cover all typical HMD FOVs. Finally, the integral needs to be made consistent with the requirement that a cumulative distribution function runs from 0 to 1:

$$f(x) = \frac{h(x)}{G(80)} \tag{6}$$

where $G(e) = \int g(e)\mathrm{d}e$.

Equation (3) requires the inverse of $f$ in Eq. (6). However, it cannot be expressed in terms of standard mathematical functions and the Lambert $W$-function [21] is needed. We simplify the function by approximating it with a fitted fourth-order polynomial determined numerically by least squares regression:

$$f^{-1}(u) \approx \begin{cases} 18.64\sqrt{u}, & u \leqslant 0.0965 \\ 25.09u^4 + 0.1680u^3 & \\ \quad + 27.61u^2 + 23.87u & u > 0.0965 \\ \quad + 3.232, & \end{cases} \tag{7}$$

The maximum approximation error in Eq. (7) is 1.8% and the integral of the difference is less than 0.04%, which are very small, especially in comparison to coarser approximations in previous work. Small error means that the model generates fewer unneeded samples in the peripheral visual regions and more in the center, leading to faster convergence.

In addition to utilizing the sampling pattern shown in Eq. (7), the proposed method allows eye tracking to be frozen. This feature is used if the user wants to look around and still generate most new samples around a certain point of interest.

In the proposed method the users preview the results with a VR HMD with eye tracking capability, but also a desktop setup could be used. We chose an HMD because a VR headset gives better spatial awareness and enjoyment [22] and, therefore, it is likely for an artist to preview the scene with a device similar to the ones used by the consumers of the rendered content. Moreover, future versions of 3D design tools might include user interfaces where the design is done partially or completely in a virtual environment using an HMD [23].

## 4 User study

To measure the subjective performance of our proposed instant preview method, we conducted a user study. It started with a questionnaire on a five-point Likert scale concerning the participant's background in 3D graphics. The questions posed are listed in Table 1.

The study used five different scenes and three different preview methods, in random order. The test scenes were BMW, Classroom, Conference, Sibenik, and Sponza. A sample view of each scene can be seen in Fig. 2. We chose the scenes to represent different 360° rendering scenarios.

### 4.1 Preview methods

Our study compared three different preview methods:

- *Omnidirectional preview* (OD): Samples were distributed uniformly to every possible point in an equirectangular image. This method represents conventional baseline rendering without preview optimization.

**Table 1** Arithmetic mean ($\mu$) and standard deviation ($\sigma$) of answers to the background questionnaire

| Question | $\mu$ | $\sigma$ |
|---|---|---|
| 1. Age? | 28.5 | 4.3 |
| 2. Gender? (5 = female, 1 = male) | 2.3 | 1.9 |
| 3. How much previous experience do you have using applications with 3D graphics (like 3D games)? | 3.5 | 1.3 |
| 4. How much previous experience do you have with offline 3D rendering (like Blender)? | 2.2 | 1.2 |
| 5. How much previous experience do you have with virtual reality or augmented reality devices? | 2.4 | 1.1 |
| 6. Have you experienced virtual reality sickness? | 2.9 | 1.4 |

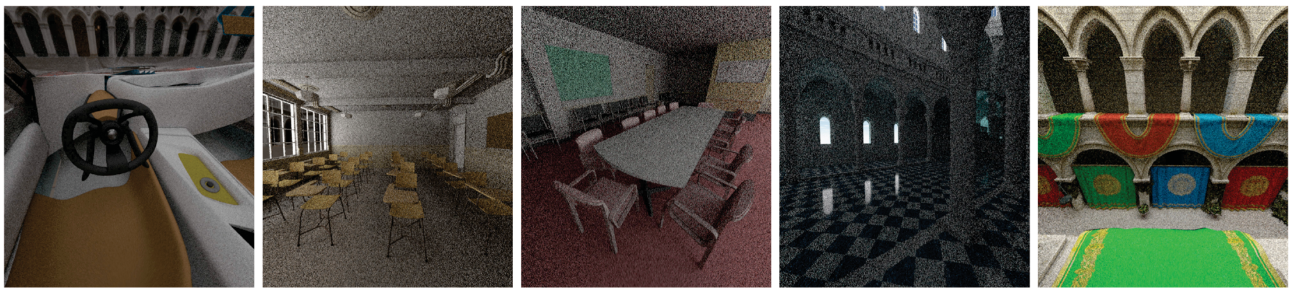


**Fig. 2** Sample views of test scenes used in the user study. Left to right: BMW, Classroom, Conference, Sibenik, and Sponza.

- *Viewport preview* (VP): Samples were distributed uniformly in the area currently viewable with the HMD. The idea was to simulate the rectangular area sampling used in some rendering engines.
- *Foveated preview* (FV): This was the proposed method, which distributed samples according to the visual acuity model centered on the gaze point of the eye-tracked user.

## 4.2 Single scene procedure

The procedure for each 3D scene can be seen in Fig. 3. We asked the participants to play the role of a 3D artist, and to choose an object in the 3D world. They were told that the object represents an object that they have just adjusted. Adjustment could have been, for example, changing the orientation of the object or changing its material parameters. Examples of both can be seen in Fig. 4.

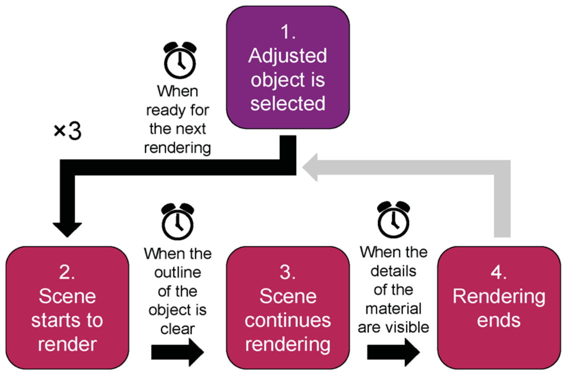
After object selection, the rendering started, and the participants recorded rendering times. The first event was recorded at the point where the participants thought that they could determine if translation or rotation of the object was successful. The second measured time represented the event when the participants were able to determine if the material adjustment was successful. The idea was that at these points the artist could cancel the rendering, go back to editing mode, and make the required changes. In each scene, the procedure described above was repeated for each preview method. The order of preview methods was randomized.

We told the participants that it was important to record the timing at a similar rendering quality in each preview method. If the participant felt that even a single timing failed substantially, the whole rendering method in that scene was timed again.

After time for a single method was measured, we asked the participants to look around in the 360° image and to rate the prevalence of disturbing artifacts in the other areas of the image. The value was recorded on a five-point Likert scale, where one meant no artifacts were present and five meant so many were present that the scene was not discernible at all.

## 4.3 Rendering

We chose unidirectional path tracing with importance sampling as the progressive rendering method. AMD RadeonRays [24] was used for ray traversal and the path tracer ran on an AMD Fury X GPU. The host code ran on an Intel Core i7-6700K CPU with 16 GB of memory. The FOVE 0 VR headset was used as a


1.
Adjusted object is selected
When ready for the next rendering
×3
2.
Scene starts to render
When the outline of the object is clear
3.
Scene continues rendering
When the details of the material are visible
4.
Rendering ends
**Fig. 3** Single scene procedure in the user study. Boxes are stages and arrows are participant's actions triggering transitions to other stages. Clocks represent points where the system saved timing.

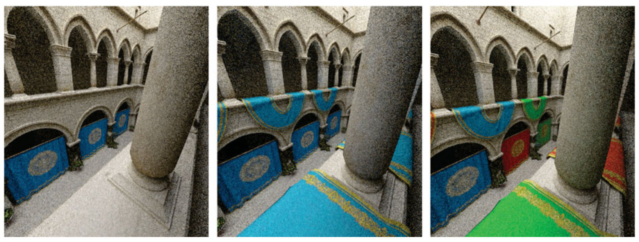


**Fig. 4** Example of a 3D artist's workflow. First the artist places U-shaped cloths, which might require many previews of the positioning with the slow progressive rendering method, especially if the objects are transparent or reflective. Then the artist modifies the material of the objects and previews the changes.

viewing device in the study due to its eye tracking capability, needed for the proposed method.

Translation of the virtual camera was disabled as camera motion would have invalidated the progressive rendering samples. Thus, the assumed starting point was that the 3D artist had already chosen the camera position by utilizing a faster rendering method.

The system generated equirectangular images because they are common in VR applications, in the authors' experience. The preview method used trilinear filtering to eliminate flickering near the poles. With an unoptimized GPU implementation, generating and sampling mipmaps on the target machine took only about 1 ms of additional time compared to bilinear filtering. This drawback was reasonable since the target was 14 ms per preview frame to achieve the 70 Hz refresh rate needed for FOVE 0.

## 5 Results

The user study included 16 participants, of whom 11 were male and 5 were female. Their ages varied between 22 and 37. Two of the participants knew details of the test set-up beforehand, but their results were so close to the average of the other results that we concluded that this did not affect the results.

### 5.1 Questionnaire

Statistics for answers to the questionnaire can be found in Table 1. The average answer to questions regarding the participants' background in 3D graphics has $P = 0.196$ compared to the speedup of foveated rendering. This $P$ value means that there is no significant correlation in the values, which was expected since the study should test the human visual system and not the person's experience in 3D graphics. In addition, the $P$ value suggests that a user study with random users should give similar results to a user study with actual 360° rendering artists. The questions used in the calculation were questions 3 and 4. The timing used in the calculation was the difference between OD and FV.

### 5.2 User timing

The geometric means of all timing are shown in Fig. 5 and arithmetic means for each scene over all participants are listed in Table 2. The results show that the proposed method required only around 10% of the time required by the baseline method OD. The

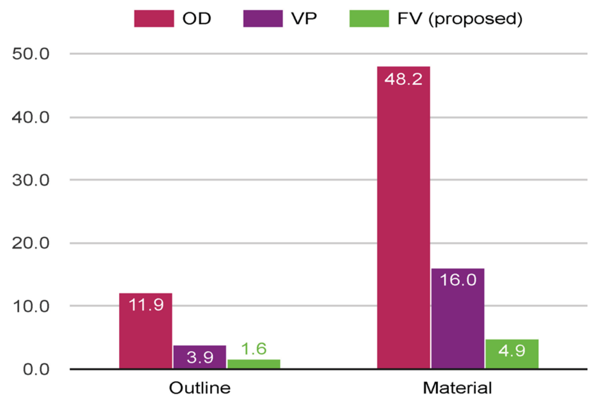


**Fig. 5** Geometric mean of time measured in seconds over all 5 test scenes using the *visible outline* and *visible material* criteria for each of the three preview methods (smaller time is better).

time savings directly translate to the frequency of the artist's feedback loop since rendering can be aborted 10 times faster. An equivalent comparison shows that previewing with VP is 3 times faster than with OD. Likewise, comparison of VP to FV shows that when rendering regular images rather than 360° images, foveated previewing can provide a 3× speedup of the previewing task.

### 5.3 Artifacts

Results of the assessment of artifacts can be seen in Fig. 6. The original idea of this measurement was to assess the reduction in quality for methods other than OD in directions away from the point of interest. These areas are not rendered at all in VP and FV because the user was looking at the point of interest throughout the test. We found out that this measurement was hard to record because every participant had a completely different idea about what should be considered a disturbing artifact. However, from the results we can see that the fast convergence of FV is perceived to have almost the same quality as OD. In contrast, VP clearly has the most artifacts. Note that if the slightly lower quality of other areas in FV is a problem in a progressive rendering system, then the system could be modified to use a hybrid of FV and OD.

### 5.4 Subjective observations

In an open discussion after the test, many participants reported that FV was so fast that it was hard to record the first timing at the right time. They also stated that it felt that the FV method converged instantly. On the other hand, several participants stated that the perceived slowness of OD might

**Table 2** Arithmetic mean ($\mu$) and standard deviations ($\sigma$) of the measured time for each scene in the user study. The results of the FV (proposed) and VP methods are compared to OD; pp stands for percent point. Large values of $\sigma$ in OD are caused by the participants selecting different kinds of objects

| Timing type | Outline visible | | | | | | Material visible | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Preview method | OD | | VP | | FV | | OD | | VP | | FV | |
| Value type | $\mu$ (s) | $\sigma$ (s) | $\mu$ (%) | $\sigma$ (pp) | $\mu$ (%) | $\sigma$ (pp) | $\mu$ (s) | $\sigma$ (s) | $\mu$ (%) | $\sigma$ (pp) | $\mu$ (%) | $\sigma$ (pp) |
| BMW | 6.1 | 7.6 | 41.8 | 9.8 | 20.8 | 1.9 | 29.9 | 16.9 | 33.6 | 15.1 | 11.8 | 5.3 |
| Classroom | 15.6 | 7.7 | 27.1 | 3.1 | 11.6 | 2.1 | 67.9 | 79.2 | 30.8 | 35.7 | 7.9 | 4.6 |
| Conference | 41.9 | 59.7 | 29.7 | 8.6 | 7.6 | 2.3 | 137.2 | 197.4 | 33.9 | 46.0 | 8.2 | 8.6 |
| Sibenik | 24.7 | 20.8 | 29.5 | 9.2 | 11.4 | 3.9 | 76.7 | 55.2 | 34.3 | 29.0 | 10.9 | 10.1 |
| Sponza | 16.1 | 13.8 | 25.9 | 4.9 | 9.1 | 2.1 | 60.4 | 46.8 | 30.1 | 18.9 | 7.5 | 4.0 |

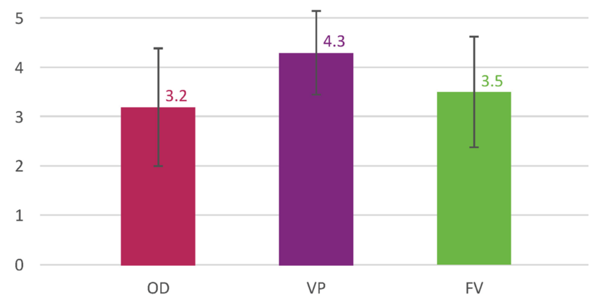


**Fig. 6** Arithmetic means and the standard deviations of the amount of artifacts the participants saw when looking around. These numbers were measured after the users were satisfied with the rendering of their point of interest.

have caused them to get bored, inducing them to mark timing at a lower quality than with the other methods. These participants simply did not have enough patience to wait for the image to converge to the same level as with the other measurements. Hence, the results are skewed in favor of OD. Most of the participants also stated that they did not realize that eye tracking was used, and instead thought that the actual rendering was somehow faster. Not noticing the eye tracking indicates that the visual acuity model is a good way to distribute the samples.

## 5.5 Performance

All three methods showed similar computational performance. On the target machine, according to AMD CodeXL, it takes around 0.17 ms to launch 65,536 primary rays with all preview methods. This includes generating random pixel coordinates for the rays and calculating the ray origin and direction based on the random pixel coordinate. In the case of our proposed FV method, modifying the random number distribution with the inversion method requires some extra work (Eq. (7)). However, our measurements showed that the extra work done in FV to generate non-uniform random numbers is entirely hidden by the latencies of memory accesses and the kernel launch.

The ray tracing performance is dependent on the user's gaze or head direction with the FV and VP methods, respectively. In contrast, OD has the same ray tracing performance independent of where the user is looking at. While OD yields a larger number of samples per second than the other methods, this result can be misleading because many of the rays are sent to directions that are easier to ray trace, e.g., directly to a skybox.

## 5.6 Latency

Because the users preview the content with an HMD device, the latency should be kept low enough to not hinder the rendering experience. None of the participants of the user study mentioned any issues with latency. Only a few reported some minor VR motion sickness, but they had also had similar symptoms in other VR experiments. Moreover, for most participants, the latency was low enough for them not to realize that the system reacted to their gaze direction.

The components adding up to the total rendering latency are shown in Fig. 7. The HMD device has a screen refresh rate of 70 Hz, which means that we need to have a new preview frame ready every 14.3 ms. Otherwise the HMD displays the same frame for 28.6 ms, thus causing a *frame drop*. To avoid reducing the quality of experience, our code is designed so that it always meets the 14 ms target. All work other than progressive rendering itself takes around 4 ms in our code. This work includes, for example, updating the UI, checking inputs, generating the mipmap, sampling it, and sending the image to the screen buffer with DirectX. In the 10 ms time left from the 14 ms target after all other work, the progressive renderer is able to path trace a batch of approximately 100,000 samples.

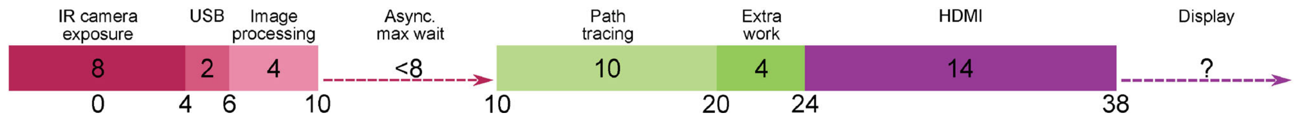


**Fig. 7** Breakdown of the system latency in milliseconds. Best-case latency is approximately 38 ms. Worst-case latency is hard to determine, since we do not know all the internals of the HMD used. Moreover, green timing is the only one we can affect without changing the HMD.

The actual end-to-end latency of foveated rendering, from eye movement to pixels being illuminated in the HMD display, is hard to determine because we do not know all the internals of the FOVE 0 driver and hardware. To the best of our knowledge, the exposure time of the eye-tracking camera is 8 ms, transferring the data to the driver takes 2 ms, and processing the data takes approximately 4 ms [25]. However, the eye-tracking device used in FOVE 0 has a refresh rate of 120 Hz [26], meaning that the exposure and processing of different frames occur in parallel. At the beginning of every frame our code queries the driver for the latest eye position, which means that if the frames of the display and eye tracking are not synchronized by the driver, in the worst case the eye position data used is that from an image processing phase that ended 8 ms ago.

It should also be noted that the image captured by the eye-tracking camera may show motion blur due to the movement of the eye. To simplify analysis, we assumed the eye to be moving at a constant speed, and also assumed that the eye position estimate produced by the image processing phase corresponds to the eye's position at the midpoint of the exposure interval. Consequently, we started our latency timing 4 ms after the start of the exposure.

After processing ends, our code swaps the image to the displays. We have not measured how long it takes for the FOVE 0 display to illuminate the pixels after the buffer swap. Since we meet the 14 ms timing requirement it is likely that the frame is moved almost immediately to the screen. FOVE 0 moves the frame via HDMI 1.4 [26], and with typical transfer speeds it should take 14 ms to move the frame to the display.

## 6 Conclusions

In this paper we have proposed a foveation-based preview system for progressive rendering. The system tracks the user's gaze and distributes samples according to a visual acuity model. Generating the sample locations with the proposed method did not show a measurable overhead in computational performance.

We measured the benefits of the system in a user study with 16 participants, who were asked to indicate how fast the different preview methods reached specified levels of quality. The targets used in the study were (i) when the users could detect if a change in the transformation of an object was successful, and (ii) when they could detect if a change in material parameters was successful.

The results showed that the rendered image converges at the user's point of interest 10 times faster than with conventional uniform sampling over the whole 360° image area. In practice this means that the 3D artist can abort rendering 10 times earlier, shortening the artist's feedback loop time and thereby improving working efficiency significantly.

Most of the user study participants did not realize that eye tracking was used, and instead thought that the rendering process itself was faster, which was the desired end result. In addition, participants rated the proposed system to have only slightly more artifacts than in areas where conventional rendering has already rendered image content progressively for several seconds and the proposed method needs to start from scratch. This is likely due to the speed of the proposed method, which is supported by the fact that many participants stated that the proposed method appears to converge instantly. The perception of foveated rendering did not have significant correlation with the participant's background in 3D graphics.

## Acknowledgements

The authors would like to thank the creators of the 3D models used in the user study: Christophe Seux for Classroom, Anat Grynberg and Greg Ward for Conference, Marko Dabrovic for Sibenik (License: CC BY-NC), and Frank Meinl for Crytek Sponza (License: CC BY 3.0). In addition, the authors would like to thank Heli Väätäjä, Chelsea Kelling, and Otto Kauhanen for helpful discussions.

The work was financially supported by the TUT Graduate School, Nokia Foundation, Emil Aaltonen Foundation, Finnish Foundation for Technology Promotion, Academy of Finland (funding decision 297548), Finnish Funding Agency for Technology and Innovation (funding decision 40142/14, FiDiPro-StreamPro) and ARTEMIS joint undertaking under grant agreement no 621439 (ALMARVI).

**Electronic Supplementary Material** Supplementary material is available in the online version of this article at https://doi.org/10.1007/s41095-018-0113-0.

## References

[1] Pharr, M.; Jakob, W.; Humphreys, G. *Physically Based Rendering: From Theory to Implementation,* 2nd edn. Morgan Kaufmann, 2010.

[2] Strasburger, H.; Rentschler, I.; Jüttner, M. Peripheral vision and pattern recognition: A review. *Journal of Vision* Vol. 11, No. 5, 13, 2011.

[3] Koskela, M.; Viitanen, T.; Jääskeläinen, P.; Takala, J. Foveated path tracing. In: *Advances in Visual Computing. Lecture Notes in Computer Science, Vol. 10072.* Bebis, G.; Boyle, R.; Parvin, B. et al. Eds. Springer Cham, 723–732, 2016.

[4] Koskela, M.; Immonen, K.; Viitanen, T.; Jääskeläinen, P.; Multanen, J.; Takala, J. Foveated instant preview for progressive rendering. In: Proceedings of the SIGGRAPH Asia 2017 Technical Briefs, Article No. 10, 2017.

[5] Weier, M.; Stengel, M.; Roth, T.; Didyk, P.; Eisemann, E.; Eisemann, M.; Grogorick, S.; Hinkenjann, A.; Kruijff, E.; Magnor, M.; Myszkowski, K.; Slusallek, P. Perception-driven accelerated rendering. *Computer Graphics Forum* Vol. 36, No. 2, 611–643, 2017.

[6] Shibata, T. Head mounted display. *Displays* Vol. 23, Nos. 1–2, 57–64, 2002.

[7] Lee, E. C.; Park, K. R. A robust eye gaze tracking method based on a virtual eyeball model. *Machine Vision and Applications* Vol. 20, No. 5, 319–337, 2009.

[8] Guenter, B.; Finch, M.; Drucker, S.; Tan, D.; Snyder, J. Foveated 3D graphics. *ACM Transactions on Graphics* Vol. 31, No. 6, Article No. 164, 2012.

[9] Vaidyanathan, K.; Salvi, M.; Toth, R.; Foley, T.; Akenine-Möller, T.; Nilsson, J.; Munkberg, J.; Hasselgren, J.; Sugihara, M.; Clarberg, P.; Janczak, T.; Lefohn, A. Coarse pixel shading. In: Proceedings of High Performance Graphics, 9–18, 2014.

[10] Stengel, M.; Grogorick, S.; Eisemann, M.; Magnor, M. Adaptive image-space sampling for gaze-contingent real-time rendering. *Computer Graphics Forum* Vol. 35, No. 4, 129–139, 2016.

[11] Weier, M.; Roth, T.; Kruijff, E.; Hinkenjann, A.; Pérard-Gayot, A.; Slusallek, P.; Li, Y. Foveated real-time ray tracing for head-mounted displays. *Computer Graphics Forum* Vol. 35, No. 7, 289–298, 2016.

[12] Murphy, H. A.; Duchowski, A. T.; Tyrrell, R. A. Hybrid image/model-based gaze-contingent rendering. *ACM Transactions on Applied Perception* Vol. 5, No. 4, Article No. 22, 2009.

[13] Reddy, M. Perceptually optimized 3D graphics. *IEEE Computer Graphics and Applications* Vol. 21, No. 5, 68–75, 2001.

[14] Pohl, D.; Zhang, X.; Bulling, A. Combining eye tracking with optimizations for lens astigmatism in modern wide-angle HMDs. In: Proceedings of the IEEE Virtual Reality, 269–270, 2016.

[15] Roth, T.; Weier, M.; Maiero, J.; Hinkenjann, A.; Li, Y. Guided high-quality rendering. In: *Advances in Visual Computing. Lecture Notes in Computer Science, Vol. 9475.* Bebis, G.; Boyle, R.; Parvin, B. et al. Eds. Springer Cham, 115–125, 2015.

[16] Pixar. Renderman 20 documentation: Rendering efficiently. 2017. Available at `https://renderman.pixar.com/resources/RenderMan_20/tutorialRenderingEfficiently.html`.

[17] The community of LuxRender. LuxRender documentation: Refine brush. 2013. Available at `http://www.luxrender.net/wiki/Refine_Brush`.

[18] Duchowski, A. T.; Bate, D.; Stringfellow, P.; Thakur, K.; Melloy, B. J.; Gramopadhye, A. K. On spatiochromatic visual sensitivity and peripheral color LOD management. *ACM Transactions on Applied Perception* Vol. 6, No. 2, Article No. 9, 2009.

[19] Viitanen, T.; Koskela, M.; Immonen, K.; Mäkitalo, M.; Jääskeläinen, P.; Takala, J. Sparse sampling for real-time ray tracing. In: Proceedings of the 13th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications, Vol. 1, 295–302, 2018.

[20] Devroye, L. *Non-Uniform Random Variate Generation.* Springer-Verlag, 1986.

[21] Weisstein, E. Lambert W-function. Available at `http://mathworld.wolfram.com/LambertW-Function.html`.

[22] MacQuarrie, A.; Steed, A. Cinematic virtual reality: Evaluating the effect of display type on the viewing experience for panoramic video. In: Proceedings of the IEEE Virtual Reality, 45–54, 2017.

[23] Stark, R.; Israel, J. H.; Wöhler, T. Towards hybrid modelling environments—Merging desktop-CAD and virtual reality-technologies. *CIRP Annals* Vol. 59, No. 1, 179–182, 2010.

[24] AMD. Radeon Rays SDK. Available at `https://github.com/GPUOpen-LibrariesAndSDKs/RadeonRays_SDK`.

[25] Rodrigo. What is the latency of FOVE eye tracking? Available at `https://support.getfove.com/hc/en-us/articles/115000733714-What-is-the-Latency-of-FOVE-Eye-Tracking-`.

[26] FOVE Inc. Tech specs. Available at `https://www.getfov.com/`.

**Matias K. Koskela** received his bachelor and master degrees with honors from Tampere University of Technology in 2014 and 2015, respectively. His research interests include optimizations and parallelism in real-time rendering, especially in real-time ray tracing.

**Kalle V. Immonen** received his M.Sc. (Tech.) degree in pervasive systems from Tampere University of Technology (TUT) in 2017. He is now working as a project researcher at TUT. His research interests include computer graphics methods and algorithms.

**Timo T. Viitanen** received his M.Sc. degree in embedded systems from Tampere University of Technology (TUT) in 2013, and is now a graduate student in the Laboratory of Pervasive Computing, TUT. He is the recipient of a TUT graduate school position and was awarded a Nokia Scholarship in 2014. His research interests include computer architecture and computer graphics.

**Pekka O. Jääskeläinen** (Adjunct Professor) has worked on heterogeneous platform customization and programming since the early 2000s. He has published over 60 academic papers in journals and conferences, and is an active contributor to various heterogeneous parallel platform related open source projects. In addition to his ongoing research on methods and tools to reduce the engineering effort involved in design and programming of heterogeneous platforms, he is interested in next generation programmable graphics architectures for photorealistic real-time rendering in the context of small form factor VR/AR products of the future.

**Joonas I. Multanen** received his M.Sc. degree in electrical engineering from Tampere University of Technology (TUT) in 2015. He is currently a graduate student at the Laboratory of Pervasive Computing, TUT. His research interests include energy efficient computer architectures and computer graphics.

**Jarmo H. Takala** received his M.Sc. (hons) degree in electrical engineering and Dr.Tech. degree in information technology from Tampere University of Technology (TUT), Tampere, Finland, in 1987 and 1999, respectively. From 1992 to 1995, he was a research scientist at VTT-Automation, Tampere, Finland. Between 1995 and 1996, he was a senior research engineer at Nokia Research Center, Tampere, Finland. From 1996 to 1999, he was a researcher at TUT. Since 2000, he has been a professor in computer engineering at TUT and is currently vice president of TUT. Dr. Takala is Co-Editor-in-Chief for *Journal of Signal Processing Systems*. During 2007–2011 he was Associate Editor and Area Editor for *IEEE Transactions on Signal Processing*. His research interests include circuit techniques, parallel architectures, and design methodologies.