High-Performance Graphics (2023)
J. Bikker and C. Gribble (Editors)

Clouds in the Cloud: Efficient Cloud-Based Rendering of Real-Time Volumetric Clouds

A. Weinrauch[1] and S. Lorbek[1] and W. Tatzgern[1] and P. Stadlbauer[1] and M. Steinberger[1,2]

[1]Graz University of Technology, Institute of Computer Graphics and Vision, Austria
[2]Huawei Technologies, Austria

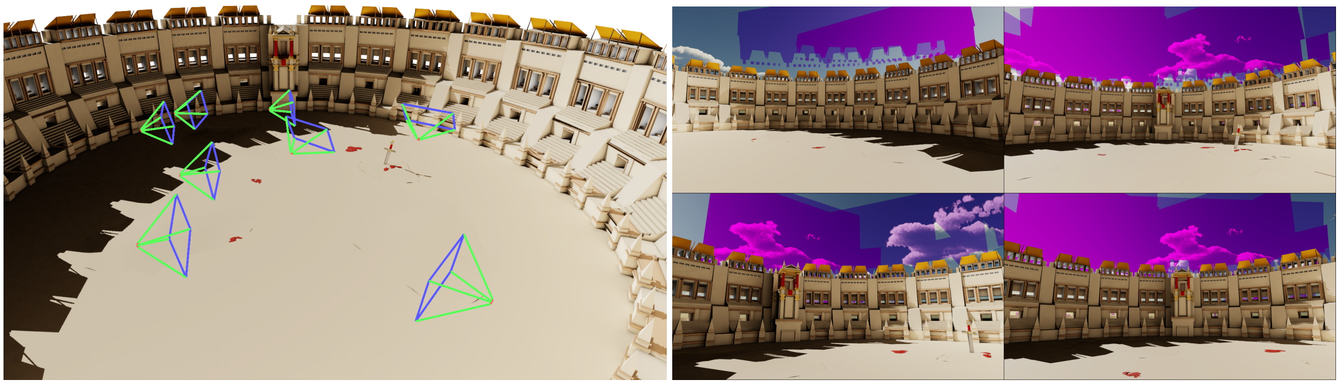

Figure 1: *A multi-viewer scenario wherein viewers are situated in a colosseum environment. The left image depicts the camera frustums, while the succeeding images on the right display the potential for reuse enabled by our method (stronger shades of pink indicating greater overlap). The potentials for significant computational power savings are immense, as vast areas of the sky (highlighted in purple on the right) can be observed by multiple viewers simultaneously.*

Abstract

Volumetric clouds play a crucial role in creating realistic, dynamic, and immersive virtual outdoor environments. However, rendering volumetric clouds in real-time presents a significant computational challenge on end-user devices. In this paper, we investigate the viability of moving computations to remote servers in the cloud and sharing them among many viewers in the same virtual world, without compromising the perceived quality of the final renderings. We propose an efficient rendering method for volumetric clouds and cloud shadows utilizing caches placed in the cloud layers and directly on the surface of objects. Volumetric cloud properties, like density and lightning, are cached on spheres positioned to represent cloud layers at varying heights. Volumetric cloud shadows are cached directly on the surfaces of receiving objects. This allows efficient rendering in scenarios where multiple viewers observe the same cloud formations by sharing redundant calculations and storing them over multiple frames. Due to the placement and structure of our caches, viewers on the ground still perceive plausible parallax under movement on the ground. In a user study, we found that viewers hardly perceive quality reductions even when computations are shared for viewers that are hundreds of meters apart. Due to the smoothness of the appearance of clouds, caching structures can use significantly reduced resolution and as such allow for efficient rendering even in single-viewer scenarios. Our quantitative experiments demonstrate computational cost savings proportional to the number of viewers placed in the scene when relying on our caches compared to traditional rendering.

CCS Concepts

• ***Computing methodologies*** → ***Rendering; Distributed algorithms;***

DOI: 10.2312/hpg.20231138

1. Introduction

Clouds not only cover a significant portion of the field of view in outdoor scenes, but they also play a crucial role in depth perception and creating an immersive experience. Rendering realistic clouds has been a challenging task that has been the subject of ongoing research for the past 30 years. Various approaches have been used to render clouds in games, ranging from static precomputed sky scenery placed as a sphere around the game world to more dynamic methods like placing particles or billboards in the atmosphere. With recent advances in hardware capabilities, game engines have been adopting ray marching approaches, which offer high visual fidelity while being flexible and fully dynamic.

However, these approaches consume a significant part of the rendering budget per frame. To achieve high frame rates, shortcuts such as lowering the resolution or using alternating/checkerboard update patterns are often employed. These shortcuts introduce issues such as edge smearing, tearing, or incomplete shading information, as commonly found in screen-space reprojection techniques. To overcome these issues and elevate the quality of cloud rendering to the next level, more computing power and higher efficiency are required. One potential solution is to shift the computation from end-user devices to remote cloud servers, as commonly found in non-real-time rendering applications.

Server farms with the latest hardware and resource-sharing capabilities can offer cost savings and efficiency gains by allocating resources where needed. Moreover, there is a high potential for increased efficiency when sharing view-independent computations among multiple viewers in the same virtual world, as seen in games, virtual exhibitions, virtual conferences, or VR chats. However, it is important to note that volumetric cloud computations depend on the viewer's location, as the appearance of the sky changes when viewed from different positions, which makes directly sharing them between viewers infeasible.

In this paper, we address the challenge of sharing view-dependent volumetric cloud computations among viewers and thus reducing the required computational power while maintaining visual quality. To achieve this, we propose a rendering pipeline that caches view-dependent dynamic volumetric cloud computations directly on the surface of objects for cloud shadows, and on proxy geometry for in-cloud properties such as density and cloud color. Our method is designed to be easily integrated into existing systems, such as shading streaming pipelines [MVD*18] or future cloud-native rendering pipelines [WTS*23]. The caching approach can also provide benefits when used directly on the end user's device, improving temporal stability with high-frequency occlusions and reducing computational costs, especially in VR applications. To evaluate our approach, we focus on the scenarios mentioned above and investigate how far viewers can be apart while still sharing computations without compromising visual quality. Furthermore, we conducted a user study to determine the impact of the perceived visual quality.

In summary, we make the following contributions:

- A novel cached rendering architecture for cloudscape computations and cloud shadows, which allows for multi-viewer sharing of rendering results and scales with visual overlap.
- Discussion of advantages and disadvantages of different caching setups for certain application types, which directly reflect on the limits for spatiotemporal reuse.
- User study results, shedding light on the ability to reuse cloudscape rendering results under wrong viewing angles for hundreds of meters.

2. Related Work

Our proposed cached cloud rendering pipeline utilizes established research to achieve real-time cloud rendering, with intermediate results stored in caches in object space. This section provides an overview of state-of-the-art rendering methods for volumetric clouds, as well as distributed systems into which our caches can be easily integrated.

2.1. Rendering of clouds

Rendering realistic clouds is a well-researched and complex topic, with various rendering methods explored in the literature, using lookup tables [NDN96], billboards or imposters [DKY*00; HL01; US05], 3D textures [HISL03], slice-based volume rendering [SSEH03], shadow view slices [MDN04], a precomputed octree[DYN01], metaballs [LCL04], radiosity [BNL06], collector area [BNM*08], photon mapping [ERWS12], and various precomputation textures [Yus14]. Recently neural approaches for directly predicting radiance have been used [KMM*17; PN19] and neural networks are even capable of generating 2D cloud appearances [SMDB22; MRI*22]. A recent survey by Goswami et al. [Gos21] provides a comprehensive overview of these state-of-the-art approaches.

Many popular game engines such as Unreal Engine 4 [NBCW18] and Frostbite [Hög16], as well as games like Horizon Zero Dawn [Sch18; Sch22] and Red Dead Redemption 2 [Bau19], have adopted ray marching-based approaches for rendering clouds in real-time. These approaches utilize ray marching to traverse through cloud layers and simulate density, in and out-scattering of light. Additionally, 3-dimensional noise textures are sampled to add details to the clouds, while height-based gradients are employed to shape the overall cloud structure. Our implementation is inspired by the presentations given by Schneider [SV15; Sch22] focusing on performance to achieve high frame rates on less powerful hardware.

2.2. Rendering in the cloud

In current cloud rendering systems, the client sends input events to the server, which then streams the generated frame buffers back to the client [CSH*16]. Sharing computations among clients has received limited attention in previous research, with a focus on a small subset of computations such as Indirect Lighting [LJZZ17; LOJZ18; CLM*15; SMBM21] or rendering participating media [SWT*23]. Recently, Weinrauch and Tatzgern *et al.* [WTS*23] proposed On-Surface and World-Space caches to enable sharing of computations across multiple viewers in the same scene, resulting in substantial computational savings by leveraging shading load overlap between viewpoints. While the authors of the paper primarily investigated view-independent effects like Ambient Occlusion or Direction Illumination from analytical light sources which is extended by the work from Stadlbauer *et al.* [SWTS23] to handle surface light computations. In contrast we employ OSC to cache view-dependent cloud computations.

Rather than computing the entire frame inside the clouds, as proposed by Weinrauch and Tatzgern *et al.* [WTS*23], our caches can also be integrated into shading streaming systems, such as the one proposed by Mueller *et al.* [MVD*18] or Neff *et al.* [NMSS22; NBD*23], to offload only volumetric cloud computations to the cloud.

3. Clouds in the Cloud

As outlined in the introduction, the main goal of our approach is to cache expensive calculations for volumetric cloud rendering for improved temporal reuse over screen-space methods and the ability to share them across many viewers in the cloud. This section explains our proposed caching pipeline in three steps. First, we discuss how our caches are designed to store view-dependent cloud computations and discuss the advantages and disadvantages of different setups. Second, we provide a short overview of the ray marching algorithm used in our reference implementation. Lastly, we discuss the requirements that an existing or future rendering system must meet in order to effectively integrate our caches and give an example.

3.1. Cloud caches

To cache calculations over multiple frames and especially among multiple viewers, existing screen-space caching techniques are not ideal. Short occlusions, for just a single frame, result in the erasure of all previously calculated information. Sharing between multiple viewers is another problem, as reprojecting into another viewer's screen space is hardly feasible.

To ease the sharing across time and space, a view-independent cache space is a better choice. One such space explored by previous research is the object space which is often used in combination with texel shading [HY16]. On-Surface-Caches (OSC) [WTS*23] store rendering information in object space.OSC are allocated and updated on demand based on the visibility of object surfaces, making them capable of supporting large scenes while keeping memory consumption low.

However, modern rendering systems do not typically model clouds as triangulated objects, and thus do not expose surfaces to place caches on. To overcome this problem, we add proxy geometry to enable OSC for volumetric clouds.

3.1.1. Single Sphere

One way to add proxy geometry to the scene is to construct a (hemi) sphere around the virtual world similar to traditional sky boxes, depicted as the blue sphere in Fig. 2. Cloud information in the caches is calculated based on the direction from the center of the sphere to the cache location, as indicated by the green dotted lines in Fig. 2. This approach eliminates the dependence on the view direction for cache calculations, resulting in view-independent caches that can

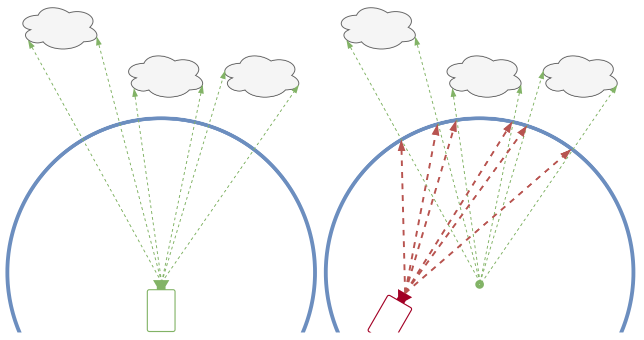

Figure 2: *Visualization of the single spheres (blue) setup. (Left) Perfect aligned view direction and cache update direction (green arrows) producing no distortions. (Right) Camera is moved off-center resulting in differences between the view direction (red arrows) and cache update direction (green arrows) producing distortions.*

be shared across different viewers but introduces visible cloud distortions when viewed off-center. The disparity between the view direction (indicated by red arrows) and the cache update direction (indicated by green arrows) leads to distorted clouds stretching them toward the center of the sphere. Increasing the radius of the sphere reduces the distortion but makes the sky appear overly static, especially if the sphere exceeds the distance to the clouds.

In a single viewer implementation, the sphere can be dynamically moved to always be centered around the camera. However, this setup necessitates reprojecting the caches as the sphere moves, as the cache directions in world space change accordingly. To illustrate this, envision the sphere being moved in Fig. 2 without adjusting the cloud positions, leading to the same cache covering a different section of the sky. Reprojection in cache space offers benefits compared to screen space techniques as the object/cache space only moves along the surface which eliminates rejection of samples and retaining information for short occlusions which increases the temporal stability of the final renderings. As a result, this setup is a valid option to consider for single-viewer applications, particularly in virtual reality (VR) applications, due to the ability to share caches between the images for each eye.

3.1.2. Multiple Spheres

The static single sphere setup may appear unnatural as the parallax due to height differences of clouds is lost. To regain the parallax, additional spheres with different diameters can be added, as shown in Fig. 3. Each sphere stores information about clouds at a specific distance range from its origin, as indicated by the colored background. With multiple spheres, the disparity between the view direction (indicated by red arrows) and the cache update direction (indicated by green arrows) is reduced for clouds cached on the larger sphere.However, the memory consumption of the caches increases linearly with the number of spheres, resulting in a trade-off between the parallax effect and memory usage. Slices of clouds may be stored on spheres if a cloud spans over multiple spheres, as depicted in Fig. 3. Due to the different diameters of the spheres, the slices move differently with camera movement, creating visual arti-

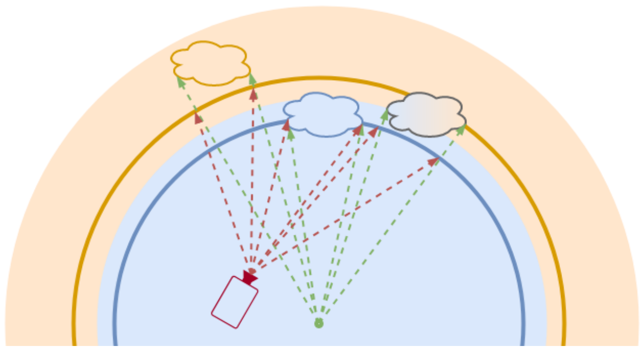

Figure 3: *Setup with two spheres (blue, orange). The color-coded background shows the caching regions of both spheres. Clouds are colored based on their caching sphere with the right-most cloud being cached partially on both spheres.*

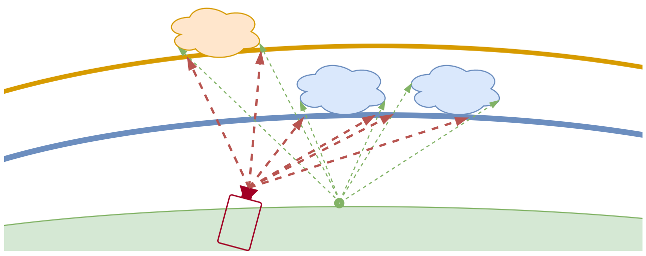

Figure 4: *Visualization of caches placed directly into the cloud layers. The surface of the planet is visualized in green, note that the cache update origin (green) is placed on the surface of a planet.*

facts as the cloud slices get separated. To address this issue, caching computations are extended to the same sphere where a cloud originated, even if its position already falls within a different sphere's region. Likewise, if a cloud is present at the beginning of a sphere's region, the accumulation starts at the end of that cloud.

Multiple spheres produce realistic cloudscapes for large portions of the sky under movement but still face challenges when approaching the horizon. Clouds closer to the horizon are generally farther away compared to clouds directly above the viewer, resulting in them being cached on the largest sphere, which removes the parallax during movement. This may not be noticeable in applications where viewpoints are primarily at ground level with occludes for clouds close to the horizon, such as mountains or tall buildings.

3.1.3. Placing Spheres into the Clouds

To correctly represent the cloud layers we place the center of the spheres at the center of the planet and adjust the radius to account for the radius of the planet and the desired cloud layer height to achieve constant altitude around the planet. This assumes that our planet is a perfect sphere, however, the shape of the caching surface can be modified to model any shape. With multiple spheres for different heights of clouds, as shown in Fig. 4, we can achieve natural-looking parallax for the entire atmosphere. The cache spheres are modeled as an icosphere, trimmed to only cover the potential surface locations when viewed up to a specific height above the ground level. For Earth-like planets, the icosphere covers a large area, which requires a significant amount of triangles to ensure a good fit and cache resolution when stored on the surface of those triangles. Therefore, we generate icospheres with changing geometrical resolution, producing smaller triangles at the zenith and larger triangles towards the horizon, to optimize the cache resolution and rendering overhead while achieving accurate representation of cloud layers.

3.2. Cache update procedure

To populate caches, we ray march in the update direction, visualized as green arrows in Fig. 4. The ray marching process occurs between an altitude of 1400m and 6000m above the ground covering Cumulus, Stratus, and Stratocumulus clouds as well as Altocumulus and Altostratus at higher cloud layers. To calculate density values at a specific location, we use the same setup as presented by Schneider [SV15], including a weather map to define where and which cloud might form, two 3-dimensional noise textures with different frequencies to generate the details, as well as shape gradients based on altitude for each cloud type. The in-scatter is calculated by ray marching towards the sun inside a cone. To reduce sampling artifacts, like color banding, we dither our starting position with blue noise [GF16].

3.3. Multi-Viewer Caching

To mitigate the perceived loss of image quality when moving away from the cache update location, viewers are grouped based on their locations, referred to as a view cell throughout this work. A view cell is defined by its location and extends. All viewers within a view cell share the same set of cache spheres, allowing for reuse of computations among viewers looking in similar directions. The size of the view cell is a trade-off between quality and computational savings, where a larger size results in increased distortion at the boundary of the view cell, but also allows for more viewers to be placed in the same view cell.

Depending on the application, view cells can be manually placed at specific regions of interest or dynamically placed based on the distribution of viewers in the virtual world. For example in massively multiplayer online role-playing games, such locations can be town halls, auction houses, battle arenas, or event locations, resulting in many viewers inside a view cell significantly reducing computational cost. In applications or regions without specific points of interest, view cells can be created based on clusters of viewer locations. View cells can then be dynamically moved as viewers move in a general direction *e.g.*, in cooperative games.

3.3.1. Transitioning between view cells

Depending on the distance between view cells clouds can look quite differently when switching from one to another making instantaneous jumps or blending between view cells infeasible. To address this issue, a transition phase can be introduced to smoothly move a viewer to another view cell. During this transition, the viewer renders without caches, while the cloud computation location is linearly interpolated from the old view cell to the new view cell. A fast transition may be noticeable to a stationary viewer as there is typically no cloud movement except for wind-induced motion, necessitating a slower transition. On the other hand, for fast-moving or

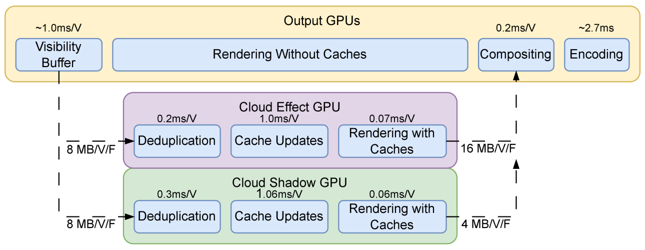

Figure 5: *Overview of our remote rendering architecture for a multi-GPU node. Cloudscape and cloud shadow computations are assigned to dedicated GPUs, while output GPUs handle visibility buffers, traditional rendering, result combination from effect GPUs, and final encoding. Bandwidth requirements between GPUs are reported in megabytes (MB) per viewer (V) and per frame (F). Latencies of different stages are reported in milliseconds (ms) per viewer (V) for 40% overlap and a mip bias of 1 for cloudscapes.*

rotating viewers, the transition can be sped up as changes in movement can help mask the additional changes from transitioning to the new view cell location. By leveraging application-specific knowledge about player intentions, multiple viewers can transition to another view cell together by assigning them to a temporary moving view cell, reducing the overhead of viewers using their own calculations instead of sharing them with other viewers in the scene.

4. Cloud Shadows

Cloud shadows are only dependent on the surface location, making them view-independent and well-suited for caching across multiple views. We adapt OSC to store the cloud shadow directly on the surface of the receiving geometry. Caches are allocated only for surfaces that are not obstructed by other geometry towards the sun. Cloud shadow cache entries are computed by accumulating the density while ray marching toward the sun from the surface. During the final rendering, we sample the cloud density from our caches to scale the incoming sunlight accordingly. If a surface does not have a cache attached to it, we treat the surface as being in the shadow.

5. Integration into existing Rendering Systems

Our system is deliberately designed to be highly adaptable and extendable and allows for easy integration into existing rendering systems.The detection of necessary caches only requires depth information from the current frame, which is commonly available in all modern rendering techniques. To render cloudscapes, the spheres are rasterized from back to front to guarantee correct blending order of the spheres. In the fragment shader, the caches are fetched combined with atmospheric scattering.

In a distributed multi-viewer system, it is necessary to separate the gathering of required cache entries for each viewer, updating required caches once, and rendering with the same caches for multiple viewers. The next subsection explains how this can be achieved in a multi-GPU node in a cloud infrastructure.

5.1. Remote Rendering Architecture

Our proposed remote rendering architecture is specifically designed for cloud ecosystems that encompass numerous nodes, each equipped with multiple high-performance GPUs. Each node possesses its own cache instance to avoid costly inter-node communication.

The efficiency of our caching system scales with the extent of computation overlap among viewers contained within each node. To optimize this overlap and maximize system performance, we group viewers to nodes based on their assigned view cells. Due to the spatial proximity of viewers, our system design offers additional advantages for non-cached traditional rendering such as shared assets when running on the same game instances which is explored by the work of Bhojan *et al.* [BNNO20].

To effectively distribute the workload across GPUs within a node, we divide viewers and assign them to output GPUs responsible for rendering the visibility buffer and non-cloud-related computations. In each node, two or more GPUs are specifically designated for computing our caches: at least one for clouds and another for cloud shadows. This architecture is depicted in Fig. 5.

The designated cache GPUs receive the visibility buffers from all output GPUs and subsequently update the local caches. Once all requested cache entries have been updated, the caches are sampled for each viewer, and the relevant effect information is transmitted back to the output GPUs. This information is sent in the form of screen space textures, with each viewer having its own dedicated texture. While it is possible to optimize bandwidth requirements by sending partial visibility buffer and cache texture updates based on previous frames, for the sake of achieving deterministic results, our focus lies on transmitting full textures between GPUs in this work.

Upon receiving the cache information for each viewer and completing the rendering of non-cached information, the output GPUs combine the results and encode the frame. The encoded frames are then sent over the network to the end user which might involve a copy to the CPU depending on the specific hardware. By following this approach, our rendering system ensures efficient utilization of multiple GPUs and enables the delivery of high-quality frames to the end user on-par with existing cloud services.

6. Evaluation

We evaluate our proposed cached volumetric cloud rendering pipeline in three distinct scenarios, showcasing its sub-linear scaling with the number of viewers while maintaining on-par visual quality as shown by similarity metrics and a conducted user study. The first scenario involves a colosseum scene that simulates a closed environment with restricted movements. The second scenario focuses on a city scene, characterized by substantial occlusions of the sky caused by buildings and trees. Lastly, the third scenario features a landscape scene that simulates open-world scenarios.

6.1. Setup

We evaluate with a resolution of 1920×1080 on a system equipped with an AMD Ryzen 3900x processor, 32GB of system memory, and an NVIDIA RTX 3090 graphics card. Our method was implemented into NVIDIA's Falcor rendering framework [KCK*22] using Vulkan. We compare our approach to a volumetric cloud screen space reference implementation. For ray marching we use 16 steps per cloud layer outside of clouds. Inside of clouds the step size is reduced to $1/8$ of the original step size to capture finer details. While ray marching towards the sun we use 6 steps as explained in Section 3.2. For cloud details we utilize two 3d noise textures with resolutions of $128 \times 128 \times 128$ and $32 \times 32 \times 32$ storing a combination of Perlin [Per85; Per02] and Worley noise [Wor96]. The caching variant of our method utilized two ico-spheres positioned at heights of 1800 meters and 4000 meters. The first sphere caches Cumulus, Stratus, and Stratocumulus clouds, and the second sphere caches Altocumulus and Altostratus clouds.

6.2. User Study

We conducted a controlled user experiment to determine the limits of reusing cached results of cloud rendering. In this study, we set out to answer three questions: **Q1**. How far can the spatial resolution of the cache be reduced before viewers experience a quality loss? **Q2**. What is the maximum distance between the rendered viewpoint and the cache generation viewpoint where viewers cannot tell that the rendering is off? **Q3**. When do viewers recognize that the viewpoint used for generating the cache is moving wrongly compared to their own viewpoint? **Q4**. What is the minimum required bandwidth for streaming cloud information without obvious perceptible compression artifacts to viewers?

6.2.1. Task and conditions

To answer all four questions, we prerecorded video frames with different settings and replayed the videos to the participants. Results for Q1-Q3 and Q4 were obtained within two separate studies. Within each section, we randomly shuffled techniques and conditions to avoid learning effects. Videos were presented on a 27" monitor at 3840×2160 resolution and 60 Hz refresh rate. Participants were seated approximately 60 cm from the monitor. Videos were encoded with H.264 using very high-quality settings, ensuring that visual quality was not diminished by encoding artifacts and that playback was running at 60 Hz.

For answering Q1, Q3, and Q4 we set up a virtual scene in a colosseum with a radius of 8 meters with the camera looking slightly towards the sky and the viewer moving on a recorded random trajectory in the colosseum. To answer Q2, we used a simple black terrain consisting of small mountains to provide a reference of scale without distracting from the clouds. For Q1, we reduced the cache resolution by applying a mip bias of $b = [0.0, 0.5, 1.0, 1.5, 2.0, 2.5, 3.0]$. For all biases, we used linear interpolation when sampling from the cache. For all tests, the reference viewpoint used to generate the cache entries was moving with the camera, evaluating only the spatial resolution reduction. For Q2, we evaluated movement directions along the three principle axes, moving left, forward, or up, while keeping the location from which the cache is created unchanged from the start, i.e., only the first frame was rendered with the correct view. For Q3, the average of 16 randomly moving viewers is used as the reference rendering position, yielding cloud renderings that move differently than the viewer. To exaggerate the effect, we scaled the reference movement by $s = [1, 10, 50, 100, 300]$, increasing the effect up to 300-fold. For Q4, we conducted encoding experiments on cloud renderings, utilizing various combinations of video compression formats ($c = [h264, h265]$), bandwidth limits ($b = [4, 8, 16, 25] Mbit/s$), and resolutions ($r = [720p, 1080p]$).

6.2.2. Procedure and participants

After filling out standard participation and consent forms, we recorded demographic information, including each participant's familiarity with computer graphics, virtual worlds, and computer games on a five-point Likert scale [Lik32]. Participants were instructed that the study focused on the quality of cloud rendering and that they should only judge the quality of the rendering of the clouds. For Q1, Q3, and Q4, participants were shown pairs of videos, one being the ground truth, the other being the testing condition, following a pairwise comparison design [KVS*17; MTM12]. We then asked the participants to rate the relative quality of the two video clips from significantly better/worse (+/-2) over slightly better/worse (+/-1) and the same (0). As the order of clips was randomized, they did not know which clip was the ground truth. From the rating, we compute a quality score (Q). Additionally, we compute the probability p_{ref} of choosing the reference over the caching approach. A p_{ref} of 50% indicates that there is no difference between the approaches; p_{ref} of 75% is referred to as just-noticeable-difference (JND) [MTM12]. Staying under 1 JND is considered high quality. For the middle section of the study, we showed single video clips for the respective movement directions and asked the participants to tell when they noticed that the visual quality was reduced or when they noticed that something was off. Again, the order in which individual videos were shown was randomized. Treating the point reported by the participants as jumps in a step function and averaging over all participants yields the probability p_{off} that viewers would judge the rendering to be off—a value akin to p_{ref}.

We recruited 26 participants (aged 25 to 37, 22 males, 4 females) from a local university, all majoring in computer science. All participants had normal or corrected vision and reported no history of visual deficits like color blindness. The average familiarity with computer graphics, virtual worlds, and computer games was 3.55. We computed the Spearman's correlation [Spe87] between Qs over all conditions and familiarity scores, gender as well as age to assess whether there is any correlation with democratic information. We did not find any significant results.

6.2.3. Results and discussion

The quality scores Q and p_{ref} for all experiments (Q1-Q4) are shown in ascending order from left to right in Fig. 6. For Q1, Q3, and Q4 we can compute the probability of the Q to come from a zero-mean distribution using Wilcoxon signed-rank test; i.e., to determine the statistical confidence that our rendering is regarded worse than the ground truth. In general, we focus on p_{ref} and p_{off} to determine JND.

As can be seen in Fig. 6 (Q1), a mip bias of $b \leq 2.0$ was rated below JND (with a maximum of 25% of participants preferring the ground truth). On average participants rated the biased versions

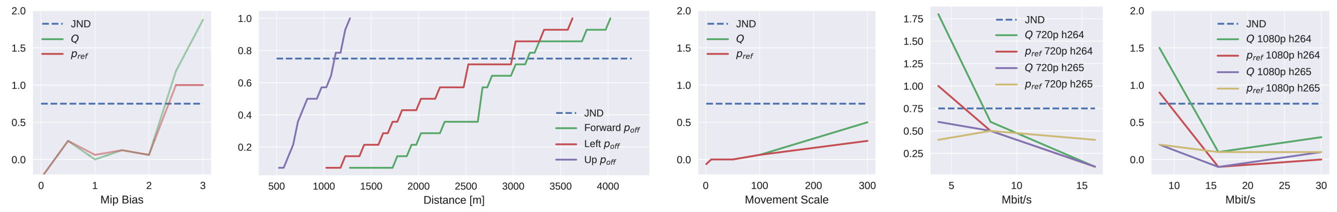

Figure 6: *Summary results of the user study: (first) A mip bias of up to* 2.0 *hardly leads to a reduced quality score Q compared to the ground truth, only for a bias of* 2.5 *both scores move significantly towards the ground truth, introducing a noticeable difference between the cached version and the ground truth ($p_{ref} > 75\%$).(second) Depending on the movement direction our cached results can be used for large distant reference rendering points before reaching 1 JND. (third) Even when exaggerating random movement by a factor of 300 and thus creating dynamic wrong reference points up to 2.4km away, participants still considered the rendering hardly worse than the ground truth (slightly worse equals* 1.0*) and the preference was below 1 JND. (fourth) For a resolution of 720p, encoding the frames using x264 with a maximum bandwidth of 8 Mbits/s or x265 with a maximum bandwidth of 4 Mbits/s does not result in noticeable degradation of the quality of cloud shading information. (fifth) For 1080p resolution, the minimum required bandwidths to maintain good quality increase to 16 Mbit/s for x264 encoding and 8 Mbit/s for x265 encoding, respectively.*

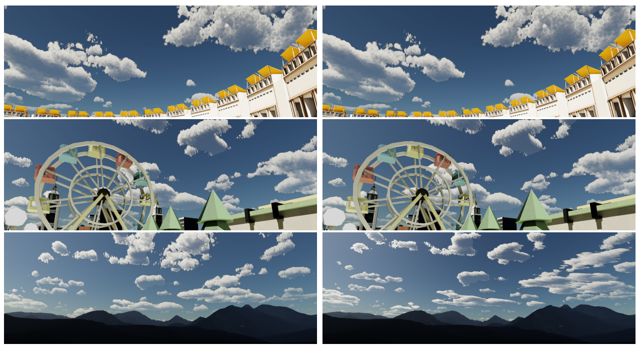

Figure 7: *Visual comparison between screen space (left) and our view cell-based caches (right) at various offsets (from top to bottom 6m, 300m, 2000m) to the view cell's location. The first two image pairs only show a small positional error of the clouds which is not noticeable. Whereas in the third row, the clouds are visually distorted towards the view cell center and start to look flat.*

only marginally worse than the ground truth ($Q < 0.25$). Interestingly, the difference to the ground truth was statistically significant for $b = 0.5$ ($z = .00$, $p < .05$); all other $b \leq 2.0$ were not significant. For $b = 2.5$ and $b = 3.0$, the reported quality and p_{ref} indicate that participants clearly found differences in the rendering ($z = .00$, $p < .001$ in all cases). Clearly, there is a significant jump from $b = 2.0$ to $b = 2.5$. As a result, we assume that a mip bias of 2.0 can safely be used to render clouds.

As can be seen in Fig. 6 (Q2), different distances to the reference rendering position are possible along the different axis. When moving left-right, the first participants noted a quality reduction after 1km of movement, 50% of participants felt that something is off after 2.5km and 1 JND was reached at 3km, and all participants noted issues after 3.6km. Similarly, when moving forward, 1.2km was regarded fine by all participants, 50% noted errors after 2.7km and 1 JND was just reached after 3.3km, with all reporting issues after 4km. Vertical movement was considered more critical, with 500m being the first distance issues were noted by one participant, 50% at 800m and 1 JND after 1.1km, with all noticing issues after 1.3km. Visual examples for distorted clouds are depicted in Fig. 7.

We found it surprising that such large distances between rendered viewpoint and used viewpoint are possible. We attribute this fact to the placement of our caches in the cloud layer, leading to plausible motion and parallax for such large movements. As such, cached clouds can be used across very large view cells in multi-viewer rendering scenarios, spanning multiple kilometers of viewpoints on the ground.

It is not surprising that vertical movement reveals issues earlier. When reaching the cloud layer, the projection surfaces collapse into a line, revealing how our caches are built. Clearly, our focus is on caching clouds viewed from the ground. When reaching up into the sky, other billboard or imposter techniques would be more appropriate. Nevertheless, supporting vertical movements of 500m to 800m still allows for significant cache reuse.

As can be seen in Fig. 6 (Q3), moving the reference point, is hardly detected by the participants. 1 JND was not reached in our tests and there was no significant difference revealed by the Wilcoxon test in either condition. According to participant feedback, they often interpreted the reference point movement as movement in the clouds itself. As such, it is also not surprising that some participants rated $s = 300$ higher than the ground truth as they were pleased with a more dynamic cloud setup. These results indicate that it is also possible to simply group viewers that are close and use their mean position for cache generation.

According to the findings presented in Figure 6 (Q4), it is evident that the newer x265 video compression standard outperforms the older x264 standard when lower bandwidths are required. Specifically, for 720p resolution, x265 achieves satisfactory results with a bandwidth of just 4 Mbit/s, while for 1080p resolution, 8 Mbit/s is sufficient to maintain quality below JDN. In cases where higher bandwidths are available, our participants slightly favored x264 over x265. However, both compression standards can deliver acceptable image quality with a bandwidth of 8 Mbit/s for 720p resolution and 16 Mbit/s for 1080p resolution. Our results align with the

Mip bias	0.0	0.5	1.0	2.0	2.5	3.0
FLIP	.0059	.0065	.0088	.0138	.0169	.0224
SSIM	.9980	.9981	.9952	.9855	.9784	.9670

Table 1: *FLIP [ANA*20] and SSIM [WBSS04] error metrics computed for different mip bias settings compared to the reference screen space method. The test setup can be observed in Fig. 8 including the FLIP visualizations.*

practices employed by existing cloud streaming services [DPT*21], further validating the effectiveness and relevance of our findings.

6.3. Visual Quality of volumetric cloud caches

Our first experiment compares the visual quality of our cached volumetric cloud rendering with a screen space reference implementation in a single viewer scenario, where the cache spheres follow the viewer. To ensure a fair comparison, we accumulated 8 samples from both methods before generating the images to remove visible noise introduced from dithering the sample locations during cloud computations.

The quality of the final renderings and FLIP [ANA*20] error visualizations for different mip bias settings are illustrated in Fig. 8. Error metrics for FLIP and SSIM [WBSS04] are reported in Tab. 1. For mip bias values lower than 1.5, the visual quality is similar, with only some loss of information at the boundaries of the cloud which was not noticeable for participants of our user study (Q1). However, as the mip bias exceeds 2.0, noticeable differences arise in all parts of the clouds. To improve visual quality for mip-biased setups, we shift the sampling location in cache space based on the size of a cache pixel, similar to jittering combined with Temporal Antialiasing techniques [Kar14]. The minor differences with a mip bias of 0.0 are caused by the cache locations not being exactly in the center of the screen space pixels, resulting in slight errors in the ray march direction compared to the screen space method.

As previously mentioned in Section 3.1.1 our caches are designed to keep accumulated calculations even during brief occlusion events. This is demonstrated in Fig. 9, which provides zoomed-in examples of regions occluded in the last couple of frames. In these examples, the screen space reference shows significantly more noise in regions that were previously occluded, as it loses all previously stored information. Demonstrating that our cached pipeline offers benefits for applications that involve high-frequency occlusion, such as viewing the sky through foliage.

6.4. Scaling of cloud caches

To show the scaling of our cloud caches, we evaluate our system with varying overlaps, which are defined as the percentage of reused cache computations between viewers. For instance, a 75% overlap requires only 25 computations with caches and 100 without. Viewers are kept in the same position while following a predefined camera path. The amount of overlap is then controlled by rotating the viewers around the y-axis. This keeps our evaluation as general as possible, making our reported numbers transferable to any application assuming overlap values are available.

Overlap %	100	80	60	40	20	0
Runtime (ms)	2.7	4.3	12.2	17.2	26.2	48.6
Memory (MB)	39	80	156	258	455	743

Table 2: *Cloud shadow computation time and memory consumption for 16 viewers placed randomly in the Bistro Exterior scene.*

The timing measurements shown in Fig. 10 were obtained using a simple ground plane, representing a worst-case performance scenario where the entire sky is visible in every frame, demonstrating the strong performance of our caching system in demanding scenarios. Our caching pipeline scales directly with the achieved overlap. Notably, our cached variant performs closely even in the 0% overlap case, indicating that our caches only induce low overhead. Applying a mip bias reduces render timings exponentially.

6.5. Memory Consumption

Memory consumption for 16 viewers with different overlap scenarios are reported in Fig. 11. The memory allocated to caches shrinks proportional to the overlap and mip bias while auxiliary buffers for cache management are allocated only based on the number of viewers. The reported values correspond to two caching spheres and can be scaled accordingly to different sphere setups. Cache entries are freed if not used for more than 20 frames to allow for temporal reuse in case of brief occlusions. Without a fixed number of frames, the system starts to release old cache entries when approaching the assigned memory budget.

6.6. Cloud Shadows visual quality

Fig. 12 shows examples of cloud shadows rendered with our caches. We omitted a comparison to our screen space reference as there are no noticeable differences to our results rendered with unbiased caches. Generally, the smooth shadow of clouds does not introduce noticeable errors when applying a mip bias of up to 1. However, due to the lower resolution of the visibility determination, artifacts appear in the shadows cast by scene geometry, as showcased in Fig. 13. A separation of visibility and cloud shadow computations could be used to allow for mip-biased cloud shadows without decreasing the perceived visual quality.

6.7. Cloud Shadows scaling

The memory consumption and cache update timings are reported in Tab. 2, showing that our cloud shadow caches scale equally well to cloud caches when considering viewer overlap. When rendering with a single viewer, we measured an overhead of approximately $0.15ms$ for cache management, while the update shader took the same amount of time compared to our screen space reference implementation. The memory consumption scales with overlap and the number of viewers. A single viewer requires an average of 40MB of memory when keeping caches allocated for at least 10 frames.

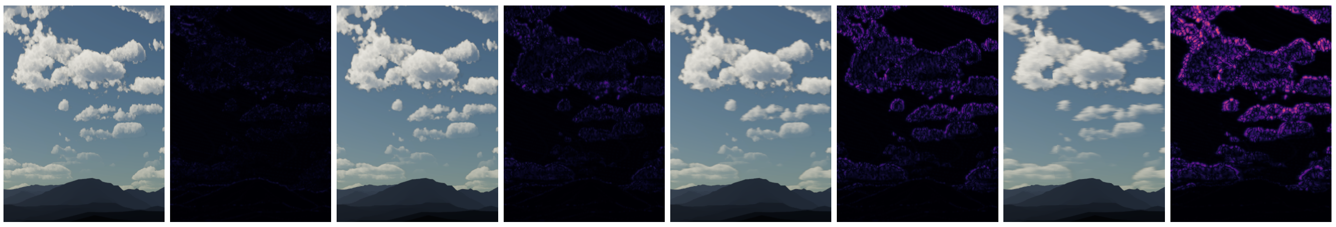

Figure 8: *Quality comparison between different mip bias settings showing pairs of the final image and the calculated FLIP error compared to the screen space implementation. The pairs from left to right correspond to the following mip biases* $[0,1,2,3]$*. With a mip bias less or equal to* 1*, barely any difference is noticeable. With a mip bias of* 2*, the FLIP error starts to show more noticeable differences, which were not detected by our user study participants (Q1).*

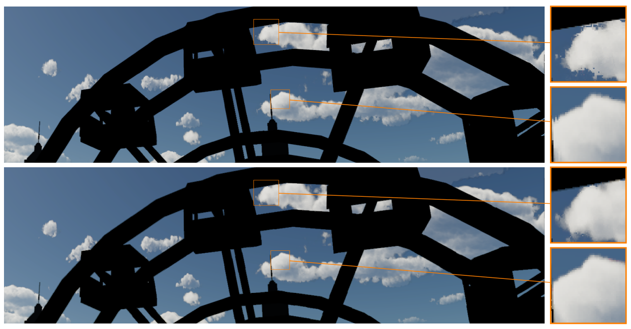

Figure 9: *Comparison of screen space (first row) to our cache approach (second row). While moving the camera to the right, for briefly occluded regions the screen space variant produces more noise than our method since the information from the previous frames is lost.*

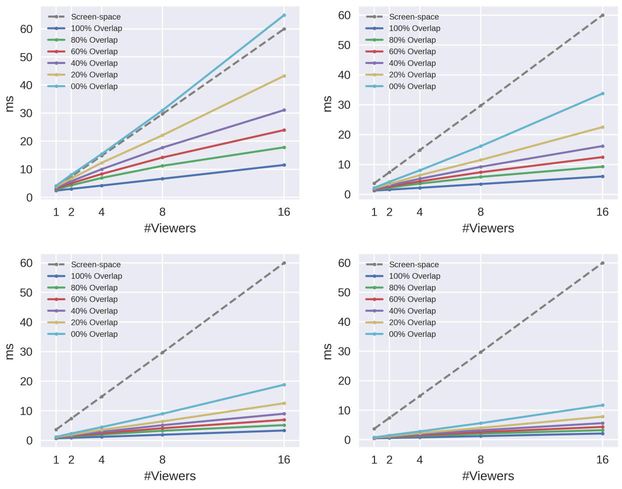

Figure 10: *Runtime of our cache pipeline for various numbers of viewers with different overlap scenarios compared to the screen space reference implementation. The four visualizations differ in the applied mip biases, 0.0 (top-left), 1.0 (top-right), 2.0 (bottom-left), 3.0 (bottom-right), showing sublinear scaling with an increasing number of viewers.*

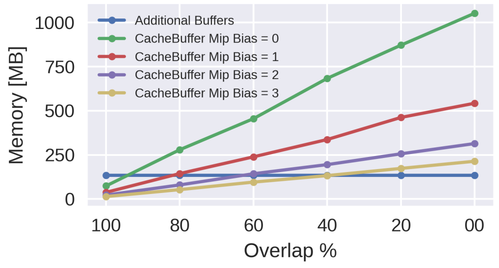

Figure 11: *Memory consumption for 16 viewers at various overlaps and mip biases. Note that our caching system allocates a fixed-size buffer based on the number of viewers for auxiliary data structures.*

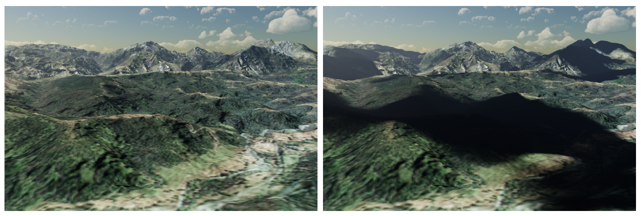

Figure 12: *Comparison of the landscape scenario without cloud shadows (left) and cloud shadows enabled (right).*

6.8. Compression Quality

The encoding and decoding processes were carried out using hardware acceleration on the GPU with a fixed bit rate and a GOP size of 5. we evaluated the performance for 720p and 1080p resolutions, simulating small form factor and large form factor screens, respectively. We did not conduct evaluations at 4K resolution due to limitations in GPU memory when handling multiple viewers on current consumer hardware. In Fig. 14 large errors are present primarily in high-frequency regions, particularly at cloud boundaries. Although these errors are partially mitigated with higher bandwidths, they do not completely disappear. Furthermore, the flip visualizations reveal smaller systematic errors in clear sky regions which noticeably decrease when the bandwidth is increased. Participants in our user study mentioned the presence of visible color banding and blocky

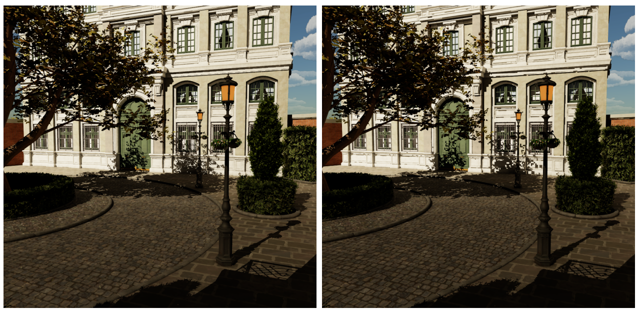

Figure 13: *Cloud shadows in Bistro Exterior with mip bias of* 0 *(left) and* 1 *(right). Low-resolution artifacts are present on the house wall due to the lower-resolution visibility determination.*

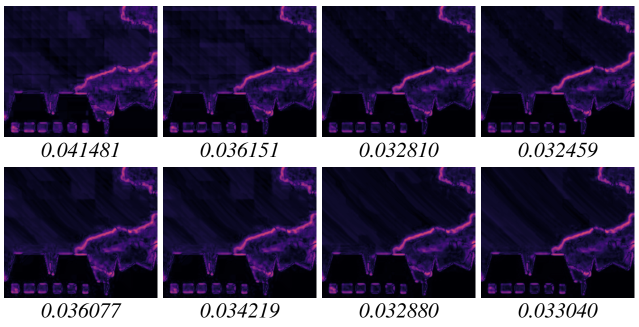

Figure 14: *Flip error visualization and metrics for locally rendered ground truth and decoded frames in 1080p from the remote server. The top row shows results for x264, while the bottom row displays results for x265. Bandwidths are constrained to* $[4,8,16,25]$ *Mbit/s from left to right.*

artifacts in these clear low-frequency regions, highlighting the importance of these areas in terms of perceived visual quality.

6.9. Latency and bandwidth between GPUs

Fig. 5 includes the bandwidth and latency requirements for our proposed remote rendering pipeline for 1080p and can be scaled to account for different pixel counts. For each viewer, the visibility buffer is sent to the effect GPUs, with each pixel requiring 4 bytes for the unique triangle index. This amounts to $2 * 8MB$ per viewer per frame, resulting in a data transfer rate of $480MB/s$ to achieve 60FPS.

To transmit the computed effect data back to the output GPUs, 16MB (RGBA16Float) are required for cloudscapes and 4MB (R16Float) for cloud shadows, resulting in $1.2GB/s$ data transferred for each viewer. Our tests were carried out on a single GPU node containing 8 NVIDIA RTX A6000 GPUs with a maximum measured bi-directional PCIe system bandwidth of $37GB/s$. In our GPU server configuration, we assign 12 viewers to a node, with 6 output GPUs assigned to handle 2 viewers each. This leads to a PCIe data transfer rate of $11.6GB/s$ from the output GPUs to the two effect GPUs and $14.4GB/s$ from the effect GPUs to the output GPUs. With this setup, we could accommodate more than 12 viewers on a node without exhausting the available bandwidth, although this may lead to increased latency and frame times as discussed in the following.

The visibility stage takes around $1ms/V$, resulting in a latency of $2ms$ since each GPU handles two viewers. The rendering of non-cached information is not considered in this evaluation, as it depends heavily on the scene and rendering quality. However, these computations can be performed simultaneously with our caches. The reported values were measured with a 40% overlap but can be scaled according to Fig. 10 and Table 2 for different overlaps. This results in a maximum latency of $16ms$ to compute the caches for 12 viewers. The compositing stage takes $0.2ms/V$, resulting in a total of $0.4ms$. For the encoding stage, we report average numbers for the encoding and transfer to the CPU. The x264 encoding was carried out by the GPU with a fixed bandwidth of $25Mbit/s$. This results in a total latency of $2.7ms$ since the GPU can handle multiple encoding tasks simultaneously [NVI23]. In this scenario our system introduces a latency of $21ms$ for 12 viewers, allowing us to handle 1080p resolution at constant 30 frames per second. By parallelizing the visibility buffer generation and encoding, and applying a slight bias to cloud shadows, 60 frames per second are not out of reach.

7. Conclusions

We have presented a caching setup that efficiently shares computations across viewers in the same virtual world, resulting in sub-linear rendering times as user counts increase. Our evaluation demonstrates that our system scales directly with the overlap between viewers, without compromising rendering quality as evidenced by error metrics and user study results. Moreover, our user study provides valuable insights for cache setups in various use case scenarios, providing initial guidelines for the use of caches in cloud rendering. Furthermore, our study indicates that users are not sensitive to errors in view-dependent shading when cache structures are chosen appropriately, which may further open the door for more aggressive caching solution for view-dependent multi-viewer computations. Additionally, we have shown that our system also scales well in single-viewer applications, improving temporal stability with low-performance overhead compared to traditional implementations. This makes our approach flexible and adaptable, capable of scaling from single viewer applications to cloud-based rendering pipelines, such as offloading cloud computations or integration into fully cloud-native rendering pipelines in the future.

8. Future Work

A first step to further expand the usefulness of caches for multi-viewer rendering is storing view-dependent information in caches to allow using larger view cells. For example, through the use of spherical harmonics [See66]. This could potentially improve the visual quality when viewing clouds from large offsets to their computation reference. Furthermore, alternative cache setups are needed when viewers are moving in the cloud layer.

Exploring ways to mitigate the latency introduced when combining our caches with streaming rendering systems is another area that requires further investigation. Given the low update frequency resulting from slow cloud movement, there are several options on how to hide transmission latency. For example, determining the potential visibility of islands, as shown by the work of Voglreiter *et al.* [VKW*23], to allocate and fill our caches ahead of time could allow sending updates even before they are needed.

References

[ANA*20] ANDERSSON, PONTUS, NILSSON, JIM, AKENINE-MÖLLER, TOMAS, et al. "FLIP: A Difference Evaluator for Alternating Images". *Proc. ACM Comput. Graph. Interact. Tech.* 3.2 (Aug. 2020). DOI: 10.1145/3406183 9.

[Bau19] BAUER, FABIAN. *Creating the Atmospheric World of Red Dead Redemption 2: A Complete and Integrated Solution*. [Online; accessed 15-April-2023]. 2019. URL: https://advances.realtimerendering.com/s2019/index.htm 3.

[BNL06] BOUTHORS, ANTOINE, NEYRET, FABRICE, and LEFEBVRE, SYLVAIN. "Real-time realistic illumination and shading of stratiform clouds". *Eurographics Workshop on Natural Phenomena*. 2006 3.

[BNM*08] BOUTHORS, ANTOINE, NEYRET, FABRICE, MAX, NELSON, et al. "Interactive Multiple Anisotropic Scattering in Clouds". *Proceedings of the 2008 Symposium on Interactive 3D Graphics and Games*. I3D '08. Redwood City, California: Association for Computing Machinery, 2008, 173–182. ISBN: 9781595939838. DOI: 10.1145/1342250.1342277 3.

[BNNO20] BHOJAN, ANAND, NG, SIANG PING, NG, JOEL, and OOI, WEI TSANG. "CloudyGame: Enabling cloud gaming on the edge with dynamic asset streaming and shared game instances". *Multimedia Tools and Applications* 79 (2020), 32503–32523 6.

[CLM*15] CRASSIN, CYRIL, LUEBKE, DAVID, MARA, MICHAEL, et al. "CloudLight: A System for Amortizing Indirect Lighting in Real-Time Rendering". *Journal of Computer Graphics Techniques (JCGT)* 4.4 (Oct. 2015), 1–27. ISSN: 2331-7418 3.

[CSH*16] CAI, WEI, SHEA, RYAN, HUANG, CHUN-YING, et al. "A Survey on Cloud Gaming: Future of Computer Games". *IEEE Access* 4 (2016), 7605–7620. DOI: 10.1109/ACCESS.2016.2590500 3.

[DKY*00] DOBASHI, YOSHINORI, KANEDA, KAZUFUMI, YAMASHITA, HIDEO, et al. "A Simple, Efficient Method for Realistic Animation of Clouds". *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '00. USA: ACM Press/Addison-Wesley Publishing Co., 2000, 19–28. ISBN: 1581132085. DOI: 10.1145/344779.344795 3.

[DPT*21] DI DOMENICO, ANDREA, PERNA, GIANLUCA, TREVISAN, MARTINO, et al. "A Network Analysis on Cloud Gaming: Stadia, GeForce Now and PSNow". *Network* 1.3 (2021), 247–260. ISSN: 2673-8732. DOI: 10.3390/network1030015 9.

[DYN01] DOBASHI, YOSHINORI, YAMAMOTO, TSUYOSHI, and NISHITA, TOMOYUKI. "Efficient rendering of lightning taking into account scattering effects due to clouds and atmospheric particles". *Proceedings Ninth Pacific Conference on Computer Graphics and Applications. Pacific Graphics 2001*. IEEE. 2001, 390–399 3.

[ERWS12] ELEK, OSKAR, RITSCHEL, TOBIAS, WILKIE, ALEXANDER, and SEIDEL, HANS-PETER. "Interactive cloud rendering using temporally coherent photon mapping". *Computers & Graphics* 36.8 (2012). Graphics Interaction Virtual Environments and Applications 2012, 1109–1118. ISSN: 0097-8493. DOI: https://doi.org/10.1016/j.cag.2012.10.002 3.

[GF16] GEORGIEV, ILIYAN and FAJARDO, MARCOS. "Blue-Noise Dithered Sampling". *ACM SIGGRAPH 2016 Talks*. SIGGRAPH '16. Anaheim, California: Association for Computing Machinery, 2016. ISBN: 9781450342827. DOI: 10.1145/2897839.2927430 5.

[Gos21] GOSWAMI, PRASHANT. "A survey of modeling, rendering and animation of clouds in computer graphics". *Vis. Comput.* 37.7 (2021), 1931–1948. DOI: 10.1007/s00371-020-01953-y 3.

[HISL03] HARRIS, MARK J., III, WILLIAM V. BAXTER, SCHEUERMANN, THORSTEN, and LASTRA, ANSELMO. "Simulation of Cloud Dynamics on Graphics Hardware". *Graphics Hardware*. Ed. by DOGGETT, M., HEIDRICH, W., MARK, W., and SCHILLING, A. The Eurographics Association, 2003. DOI: 10.2312/EGGH03/092-101 3.

[HL01] HARRIS, MARK J. and LASTRA, ANSELMO. "Real-Time Cloud Rendering". *Computer Graphics Forum* 20.3 (2001), 76–85. DOI: https://doi.org/10.1111/1467-8659.00500 3.

[Hög16] HÖGFELDT, RURIK. "Convincing Cloud Rendering–An Implementation of Real-Time Dynamic Volumetric Clouds in Frostbite". (2016) 3.

[HY16] HILLESLAND, K. E. and YANG, J. C. "Texel Shading". *Proceedings of the 37th Annual Conference of the European Association for Computer Graphics: Short Papers*. EG '16. Lisbon, Portugal: Eurographics Association, 2016, 73–76 4.

[Kar14] KARIS, BRIAN. *High-Quality Temporal Supersampling*. [Online; accessed 15-April-2023]. 2014. URL: https://advances.realtimerendering.com/s2014/ 9.

[KCK*22] KALLWEIT, SIMON, CLARBERG, PETRIK, KOLB, CRAIG, et al. *The Falcor Rendering Framework*. https://github.com/NVIDIAGameWorks/Falcor. Aug. 2022. URL: https://github.com/NVIDIAGameWorks/Falcor 7.

[KMM*17] KALLWEIT, SIMON, MÜLLER, THOMAS, MCWILLIAMS, BRIAN, et al. "Deep Scattering: Rendering Atmospheric Clouds with Radiance-Predicting Neural Networks". *ACM Trans. Graph.* 36.6 (Nov. 2017). ISSN: 0730-0301. DOI: 10.1145/3130800.3130880 3.

[KVS*17] KIRAN ADHIKARLA, VAMSI, VINKLER, MAREK, SUMIN, DENIS, et al. "Towards a Quality Metric for Dense Light Fields". *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. July 2017 7.

[LCL04] LIAO, HORNG-SHYANG, CHUANG, JUNG-HONG, and LIN, CHENG-CHUNG. "Efficient Rendering of Dynamic Clouds". *Proceedings of the 2004 ACM SIGGRAPH International Conference on Virtual Reality Continuum and Its Applications in Industry*. VRCAI '04. Singapore: Association for Computing Machinery, 2004, 19–25. ISBN: 1581138849. DOI: 10.1145/1044588.1044591 3.

[Lik32] LIKERT, RENSIS. "A technique for the measurement of attitudes." *Archives of Psychology* (1932) 7.

[LJZZ17] LIU, CHANG, JIA, JINYUAN, ZHANG, QIAN, and ZHAO, LEI. "Lightweight WebSIM Rendering Framework Based on Cloud-Baking". SIGSIM-PADS '17. Singapore, Republic of Singapore: Association for Computing Machinery, 2017, 221–229. ISBN: 9781450344890. DOI: 10.1145/3064911.3064933 3.

[LOJZ18] LIU, CHANG, OOI, WEI TSANG, JIA, JINYUAN, and ZHAO, LEI. "Cloud Baking: Collaborative Scene Illumination for Dynamic Web3D Scenes". *ACM Trans. Multimedia Comput. Commun. Appl.* 14.3s (June 2018). ISSN: 1551-6857. DOI: 10.1145/3206431 3.

[MDN04] MIYAZAKI, RYO, DOBASHI, YOSHINORI, and NISHITA, TOMOYUKI. "A fast rendering method of clouds using shadow-view slices". *Proc. CGIM* (2004), 93–98 3.

[MRI*22] MIRBAUER, MARTIN, RITTIG, TOBIAS, ISER, TOMÁŠ, et al. "SkyGAN: Towards Realistic Cloud Imagery for Image Based Lighting". *Eurographics Symposium on Rendering*. Ed. by GHOSH, ABHIJEET and WEI, LI-YI. The Eurographics Association, 2022. ISBN: 978-3-03868-187-8. DOI: 10.2312/sr.20221151 3.

[MTM12] MANTIUK, RAFAŁ K., TOMASZEWSKA, ANNA, and MANTIUK, RADOSŁAW. "Comparison of Four Subjective Methods for Image Quality Assessment". Vol. 31. 8. 2012, 2478–2491. DOI: https://doi.org/10.1111/j.1467-8659.2012.03188.x. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1467-8659.2012.03188.x 7.

[MVD*18] MUELLER, JOERG H., VOGLREITER, PHILIP, DOKTER, MARK, et al. "Shading Atlas Streaming". *ACM Trans. Graph.* 37.6 (Dec. 2018). ISSN: 0730-0301. DOI: 10.1145/3272127.3275087 3, 4.

[NBCW18] Nowak, Lukasz, Bak, Artur, Czajkowski, Tomasz S., and Wojciechowski, Konrad. "Modeling and Rendering of Volumetric Clouds in Real-Time with Unreal Engine 4". *Computer Vision and Graphics - International Conference, ICCVG 2018, Warsaw, Poland, September 17-19, 2018, Proceedings*. Ed. by Chmielewski, Leszek J., Kozera, Ryszard, Orlowski, Arkadiusz, et al. Vol. 11114. Lecture Notes in Computer Science. Springer, 2018, 68–78. doi: 10.1007/978-3-030-00692-1_7 3.

[NBD*23] Neff, Thomas, Budge, Brian, Dong, Zhao, et al. "PSAO: Point-Based Split Rendering for Ambient Occlusion". *Computer Graphics Forum* 42.8 (2023). doi: 10.1111/cgf.14864. url: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.14864 4.

[NDN96] Nishita, Tomoyuki, Dobashi, Yoshinori, and Nakamae, Eihachiro. "Display of Clouds Taking into Account Multiple Anisotropic Scattering and Sky Light". *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '96. New York, NY, USA: Association for Computing Machinery, 1996, 379–386. isbn: 0897917464. doi: 10.1145/237170.237277 3.

[NMSS22] Neff, T., Mueller, J. H., Steinberger, M., and Schmalstieg, D. "Meshlets and How to Shade Them: A Study on Texture-Space Shading". *Computer Graphics Forum* 41.2 (2022), 277–287. doi: https://doi.org/10.1111/cgf.14474. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.14474. url: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.14474 4.

[NVI23] NVIDIA. *Video Encode and Decode GPU Support Matrix*. [Online; accessed 20-May-2023]. 2023. url: https://developer.nvidia.com/video-encode-and-decode-gpu-support-matrix-new 11.

[Per02] Perlin, Ken. "Improving Noise". *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '02. San Antonio, Texas: Association for Computing Machinery, 2002, 681–682. isbn: 1581135211. doi: 10.1145/566570.566636 7.

[Per85] Perlin, Ken. "An Image Synthesizer". *SIGGRAPH Comput. Graph.* 19.3 (July 1985), 287–296. issn: 0097-8930. doi: 10.1145/325165.325247 7.

[PN19] Panin, Mikhail and Nikolenko, Sergey. "Faster RPNN: Rendering Clouds with Latent Space Light Probes". *SIGGRAPH Asia 2019 Technical Briefs*. SA '19. Brisbane, QLD, Australia: Association for Computing Machinery, 2019, 21–24. isbn: 9781450369459. doi: 10.1145/3355088.3365150 3.

[Sch18] Schneider, Andrew. "Real-time volumetric cloudscapes". *GPU Pro 360 Guide to Lighting*. AK Peters/CRC Press, 2018, 473–504 3.

[Sch22] Schneider, Andrew. *Nubis, Evolved: Real-Time Volumetric Clouds for Skies, Environments, and VFX*. [Online; accessed 22-May-2023]. 2022. url: https://advances.realtimerendering.com/s2022/index.html 3.

[See66] Seeley, R. T. *Spherical Harmonics*. Vol. 73. 4P2. Taylor & Francis, 1966, 115–121. doi: 10.1080/00029890.1966.11970927 11.

[SMBM21] Stengel, Michael, Majercik, Zander, Boudaoud, Benjamin, and McGuire, Morgan. "A Distributed, Decoupled System for Losslessly Streaming Dynamic Light Probes to Thin Clients". *Proceedings of the 12th ACM Multimedia Systems Conference*. MMSys '21. Istanbul, Turkey: Association for Computing Machinery, 2021, 159–172. isbn: 9781450384346. doi: 10.1145/3458305.3463379 3.

[SMDB22] Satilmis, Pinar, Marnerides, Demetris, Debattista, Kurt, and Bashford-Rogers, Thomas. "Deep Synthesis of Cloud Lighting". *IEEE Computer Graphics and Applications* 42.5 (2022), 8–18 3.

[Spe87] Spearman, C. "The Proof and Measurement of Association between Two Things". *The American Journal of Psychology* 100.3/4 (1987), 441–471. issn: 00029556 7.

[SSEH03] Schpok, Joshua, Simons, Joseph, Ebert, David S., and Hansen, Charles. "A Real-Time Cloud Modeling, Rendering, and Animation System". *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. SCA '03. San Diego, California: Eurographics Association, 2003, 160–166. isbn: 1581136595 3.

[SV15] Schneider, Andrew and Vos, Nathan. *The Real-time Volumetric Cloudscapes of Horizon: Zero Dawn*. [Online; accessed 15-April-2023]. 2015. url: https://advances.realtimerendering.com/s2015/index.html 3, 5.

[SWT*23] Stojanovic, Robert, Weinrauch, Alexander, Tatzgern, Wolfgang, et al. "Efficient Rendering of Participating Media for Multiple Viewpoints". *Computer Graphics Forum* 42.8 (2023). doi: 10.1111/cgf.14874. url: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.14874 3.

[SWTS23] Stadlbauer, Pascal, Weinrauch, Alexander, Tatzgern, Wolfgang, and Steinberger, Markus. "Surface Light Cones: Sharing Direct Illumination for Efficient Multi-viewer Rendering". *Computer Graphics Forum* 42.8 (2023). doi: 10.1111/cgf.14875. url: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.14875 4.

[US05] Umenhoffer, Tamás and Szirmay-Kalos, László. "Real-Time Rendering of Cloudy Natural Phenomena with Hierarchical Depth Impostors." *Eurographics (Short Presentations)*. 2005, 65–68 3.

[VKW*23] Voglreiter, Philip, Kerbl, Bernhard, Weinrauch, Alexander, et al. "Trim Regions for Online Computation of From-Region Potentially Visible Sets". *ACM Trans. Graph.* 42.4 (Jan. 2023) 12.

[WBSS04] Wang, Zhou, Bovik, A.C., Sheikh, H.R., and Simoncelli, E.P. "Image quality assessment: from error visibility to structural similarity". *IEEE Transactions on Image Processing* 13.4 (2004), 600–612 9.

[Wor96] Worley, Steven. "A Cellular Texture Basis Function". *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '96. New York, NY, USA: Association for Computing Machinery, 1996, 291–294. isbn: 0897917464. doi: 10.1145/237170.237267 7.

[WTS*23] Weinrauch, Alexander, Tatzgern, Wolfgang, Stadlbauer, Pascal, et al. "Effect-based Multi-viewer Caching for Cloud-native Rendering". *ACM Trans. Graph.* 42.4 (Jan. 2023) 3, 4.

[Yus14] Yusov, Egor. "High-Performance Rendering of Realistic Cumulus Clouds Using Pre-computed Lighting". *Eurographics/ ACM SIGGRAPH Symposium on High Performance Graphics*. Ed. by Wald, Ingo and Ragan-Kelley, Jonathan. The Eurographics Association, 2014. isbn: 978-3-905674-60-6. doi: 10.2312/hpg.20141101 3.