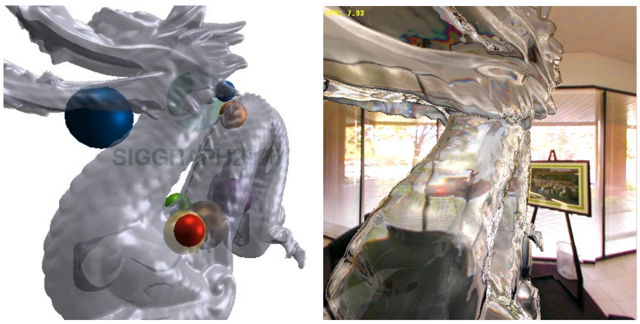
GPU-Accelerated Transparent Point-Based Rendering

Yanci Zhang* Renato Pajarola†

Visualization and MultiMedia Lab
Department of Informatics, University of Zürich



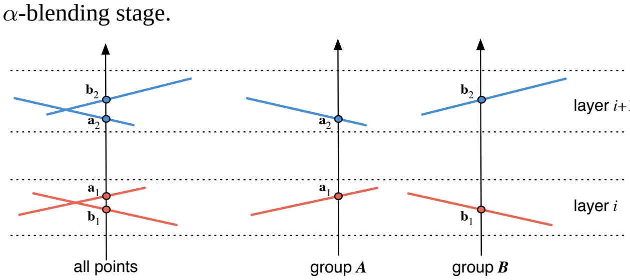
a) b)

Figure 1: Rendering results. a) Opaque and transparent objects; b) Refractive and reflective environment mapping, including both the Fresnel effect and chromatic dispersion.

1 Overview

The main difficulty of rendering transparent point-based objects as shown in Figure 1 is to solve the confliction between two following blending operations: *PBR-interpolation*, which is used to blend screen-space overlapping splats within the same surface layer to achieve smooth surface rendering, and *transparency-compositing*, which is used to α-blend surface layers in a back-to-front order to generate the effect of transparency.

We present a novel algorithm to separate the two blending operations into different rendering passes as indicated in Figure 2. Our method is based on a graph coloring algorithm that divides the point set into groups with minimal overlap. For points within a single group, PBR-blending is omitted as the overlap between splats is small. Instead, transparency-blending is performed in the first phase for each group separately. In a post-pass, PBR-blending is eventually achieved by combining the images from the transparent α-blending stage.



a) b)

Figure 2: a) Rendering of transparent points must distinguish per fragment blending for point interpolation as well as transparent α-blending. b) We divide points into groups; where a_1, b_1 are α-blended for transparency with a_2, b_2 within their group respectively, and then PBR-interpolation is performed in a post-pass.

2 Preprocessing

The purpose of our preprocessing stage is to divide points into groups to achieve the following two targets: (1) Minimize overlaps among splats in individual groups so that PBR-blending can be deferred to an image compositing post-pass; (2) Each group provides a good coverage of the surface to allow for transparent α-blending.

The minimal overlap grouping can be formulated as a K-colors *Weighted Graph Coloring* (WGC) problem. Let us define the weighted graph $G(\mathcal{S}, E)$ over the points $\mathcal{S} = \{\mathbf{p}_{1\ldots n}\}$ with edges $E = \{e_{ij} | \text{iff } \mathbf{p}_i \text{ and } \mathbf{p}_j \text{ have non-zero overlap}\}$ and weights $W = \{w_{ij} | \text{overlap between } \mathbf{p}_i \text{ and } \mathbf{p}_j\}$. The goal of K-colors WGC is to assign one of K colors to each node in G to generate K subgraphs while minimizing the sum of weights in the subgraphs. This problem can be approached by a two-step greedy strategy including an initialization and an optimization step. In order to improve the surface coverage, some splats may be duplicated between groups or splat radii can be enlarged.

3 Rendering

Note that PBR blending kernels can not be used as the interpolation weights interfere with the transparent α-blending. Hence each fragment contributes equally to the final smooth point interpolation. The artifacts introduced by this simplified PBR-blending can be masked visually to be unnoticeable by separately considering the nearest of the transparent layers due to the following observations: The first fact is that visual artifacts are reduced dramatically behind transparent surface layers. The second fact is that with current 8-bit color and α resolutions any errors below a value of $1/256$ have no effect. Therefore, we can render the nearest layer exclusively and separately in high quality using PBR blending kernels as there is no α-blending within the same layer. The following 3-pass algorithm describes the basic approach of our transparent PBR solution.

1. *Geometry Pass for Nearest Layer*: Render point groups $\mathcal{S}_k$, generated by the WGC algorithm, into K target images F_k using high-quality PBR blending, including the depth of the nearest fragments (depth-buffer Z) and blending kernel weight information.
2. *Geometry Pass for Other Layers*: Render point groups $\mathcal{S}_k$ using back-to-front transparency α-blending into K target images O_k, but ignoring all fragments from the nearest layer using the depth-mask Z from the previous pass.
3. *Compositing Pass*: Combine images F_k together according to the per fragment PBR kernel weights, resulting in a smoothly interpolated image C_F for the nearest visible layer. Average images O_k into C_O for the other layers. High-quality transparency is finally achieved by α compositing C_F and C_O.

Our approach allows for mixed opaque and transparent point surfaces correctly rendered together as shown in Figure 1 a). We note here that opaque surfaces and basic transparent point rendering can be achieved by a modified algorithm, which only uses one pass over the point geometry, not further discussed here but demonstrated in the accompanying video. High-quality transparent PBR is shown in Figure 1 b). In fact, our algorithm can also simulate visual effects such as multi-layer refraction and reflection, also demonstrated in the video sequence.

*e-mail: zhang@ifi.unizh.ch
†e-mail: pajarola@acm.org