Non-photorealistic ray tracing with paint and toon shading

Nicholas Moon
University of Virginia
nm9nz@virginia.edu

Megan Reddy
University of Virginia
mr8vn@virginia.edu

Luther Tychonievich
University of Virginia
tychonievich@virginia.edu

ABSTRACT

We present a modification to traditional ray tracing that stylistically renders a scene with cartoon and painterly styles. Previous methods rely on post-processing, materials, or textures to achieve a non-photorealistic look. Our method uses a ray tracer to combine cel animation art styles with complex lighting effects, such as reflections, refractions, and global illumination. The ray tracer collects information about objects and their properties to dynamically switch between cartoon and painterly rendering styles. The renderer generates the styles by shooting additional rays for each pixel and collecting information such as normals, distance, slope, object identifiers, and light gradients from neighboring areas of the image. The resulting algorithm produces images with visual and artistic characteristics that allow artists to take advantage of rendering techniques that are not commonly supported in production ray tracers.

CCS CONCEPTS

• **Computing methodologies** → Computer graphics; Rendering; Ray tracing; Computer graphics; Rendering; Non-photorealistic rendering.

KEYWORDS

Cartoon rendering, painterly rendering, physically-based rendering, global illumination

ACM Reference Format:
Nicholas Moon, Megan Reddy, and Luther Tychonievich. 2021. Non-photorealistic ray tracing with paint and toon shading. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Posters (SIGGRAPH '21 Posters), August 09–13, 2021, Virtual Event, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3450618.3469173

Figure 1: Two scenes with paint and toon shading. Reflections and refractions include paint and outlines.

1 INTRODUCTION

Non-photorealistic rendering (NPR) encompasses cartoon and painterly styles. Ray tracing creates photorealistic images with reflections, refractions, and global illumination. Our method combines both techniques by using a ray tracer to stylistically render a 3D scene, allowing interaction between various NPR styles simultaneously.

In order to create a cartoon or painterly look, many algorithms use special materials or textures. In painterly rendering, most methods perform stylization as a post-processing step on an image [Hertzmann, 1998] or determine stroke attributes with shaders [Meier, 1996] instead of acquiring them through the ray tracing process. They also use surface normal cutoffs for feature edge detection [Markosian et al., 1997] or base quantized colors solely on normal and light direction vectors [Anjyo and Hiramitsu, 2003]. Alternatively, work with cartoon ray tracing has yielded results without outlines in reflections and refractions [Choudhury and Parker, 2009].

2 METHOD

The primary ray cast for each pixel intersects with a foreground (toon) or background (paint) object, and uses the its style type to follow either the toon shading or paint generation process (Figure 1).

2.1 Toon Rendering

2.1.1 Quantized Light Intensity. To implement the toon shading style, rays collect illumination at the surface of the closest primary ray intersection, and quantize it after accumulating direct light, indirect light, and bidirectional light samples. The shading process illuminates triangles using the barycentric coordinates of the intersection point, so cutoffs of quantized light intensity values are smooth and continuous for low polygon models.

SIGGRAPH '21 Posters, August 09–13, 2021, Virtual Event, USA

ACM ISBN 978-1-4503-8371-4/21/08.
https://doi.org/10.1145/3450618.3469173

2.1.2 Outline Detection. Outlines are generated in screen space by sending rays around a disc centered on each pixel. These rays trace through the scene, reflecting and refracting, and store information regarding every hit object's identifier, distance, object style type, and total number of objects hit. We use this data to determine if the pixel lies on a transition between two objects (different object identifiers), on a crease in a single object (difference in distance), or on the edge of reflective or refractive surfaces (comparing all hit objects' identifiers). The total number of samples that differ from the primary ray collision determines the resulting intensity of the outline at that point, to enable an outline color gradient.

2.2 Painterly Rendering

2.2.1 Brush Creation. We initialize a circular brush mask and compute its radius as a constant fraction of image resolution. Our brush engine supports two types of brushes, one with a linear falloff and one with no falloff. Brush masks hold paint alpha values, which allow the painterly renderer to perform a simple lookup and use the ray traced color with those values while painting.

2.2.2 Stroke Creation. We create strokes based on data available during ray tracing, following a process similar to those outlined in post-processing painterly techniques [Hertzmann, 1998] and particle system techniques [Hanson and Todd, 2014]. We initialize a constant stroke length based on image resolution. For each pixel, we calculate the gradient direction according to normal and lighting properties at the intersection point. We calculate the rest of the stroke points and add the stroke to a list of strokes for the scene.

2.2.3 Stroke Placement. Our algorithm sends the list of strokes to the painterly renderer, which composites the strokes onto the canvas with the appropriate brush type. We draw strokes in a random order to create an organic and painterly look. If strokes overlap, the renderer mixes their colors with alpha blending. During the ray tracing process, we keep an object boundary buffer and object type buffer, which allow us to trim strokes so that they do not bleed into neighboring objects or toon objects.

2.3 Style Integration

We also implement techniques for integrating these two features to create a cohesive image. For anti-aliasing, we average ray samples for foreground objects; background objects paint a stroke for each sample, and have anti-aliasing built into the stroke placement itself. Cel-based animation styles typically treat shadows as foreground objects, usually a single dark silhouette applied on top of the painted background. We generate this effect by storing shadow intensity in a buffer during the ray tracing process, and then use the values stored in that buffer to attenuate the color of paint strokes already applied to the scene. We also store outline intensity in a buffer, and apply it over paint strokes and shadows as the last step of the rendering process.

3 DISCUSSION

In Figure 2, all objects are background objects with the exception of the person, plates, and food. In (b), the ground, buildings, tables, and chairs have a stroked appearance instead of normal shading. The foreground objects, such as the person, show color quantization and outline detection. The shadowed parts of background objects have paint stroke color attenuated.

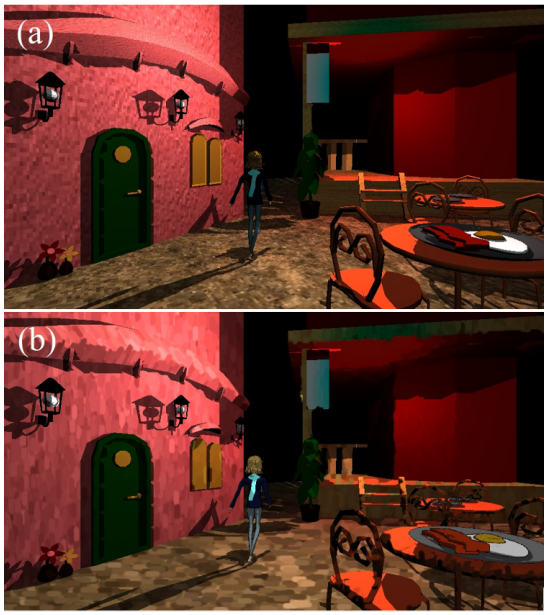

Figure 2: A scene without stylization (a) and with stylization (b).

ACKNOWLEDGMENTS

We would like to thank our animation professor at the University of Virginia, Earl Mark, for his previous guidance. The model of the person was retrieved from free3d.com/3d-model/girl-blind-703979.html

REFERENCES

Ken-ichi Anjyo and Katsuaki Hiramitsu. 2003. Stylized Highlights for Cartoon Rendering and Animation. IEEE Comput. Graph. Appl. 23, 4 (July 2003), 54–61. DOI:https://doi.org/10.1109/MCG.2003.1210865

A. N. M. Imroz Choudhury and Steven G. Parker. 2009. Ray tracing NPR-style feature lines. In Proceedings of the 7th International Symposium on Non-Photorealistic Animation and Rendering (NPAR '09). Association for Computing Machinery, New York, NY, USA, 5–14. DOI:https://doi.org/10.1145/1572614.1572616

Andy Hanson and Scott Todd. 2014. Object-Space Painterly Rendering for WebGL. Final Report, Advanced Graphics. Rensselaer Polytechnic Institute, Troy, NY.

Aaron Hertzmann. 1998. Painterly rendering with curved brush strokes of multiple sizes. In Proceedings of the 25th annual conference on Computer graphics and interactive techniques (SIGGRAPH '98). Association for Computing Machinery, New York, NY, USA, 453–460. DOI:https://doi.org/10.1145/280814.280951

Lee Markosian, Michael A. Kowalski, Daniel Goldstein, Samuel J. Trychin, John F. Hughes, and Lubomir D. Bourdev. 1997. Real-time nonphotorealistic rendering. In Proceedings of the 24th annual conference on Computer graphics and interactive techniques (SIGGRAPH '97). ACM Press/Addison-Wesley Publishing Co., USA, 415–420. DOI:https://doi.org/10.1145/258734.258894

Barbara J. Meier. 1996. Painterly rendering for animation. In Proceedings of the 23rd annual conference on Computer graphics and interactive techniques (SIGGRAPH '96). Association for Computing Machinery, New York, NY, USA, 477–484. DOI:https://doi.org/10.1145/237170.237288