6-9-2008

Direct Shader Sampling in Painterly Rendering for Animation

Matt Kuruc
University of Pennsylvania, mkuruc@seas.upenn.edu

Chris Czyzewicz
University of Pennsylvania, skiz@seas.upenn.edu

Vijay Nair
University of Pennsylvania, vnair2@seas.upenn.edu

Norman I. Badler
University of Pennsylvania, badler@seas.upenn.edu

Direct Shader Sampling in Painterly Rendering for Animation

Abstract
We present a technique for generating stroke parameters in particlebased painterly rendering algorithms that guarantees temporal coherence in animation through directly sampling shaders. To determine the appropriate color for a brush stroke, techniques like [Meier 1996] render the scene into reference pictures using traditional techniques. This reference picture is then queried by applying the camera transform to the particle's position. This transform will not always map to the correct pixel which can lead to noticeable temporal discontinuities when there is a significant color difference between two neighboring pixels. When a stroke particle lies on the edge of an object, the brush stroke will sometimes flicker between the two neighboring colors. Previously, as described in [Meier 1996], scenes would be broken up into different layers, and image processing effects were applied to reference pictures to ensure that the right color is sampled. Our method requires no post processing effects to ensure temporal coherence around these edge cases.

Disciplines
Computer Sciences | Engineering | Graphics and Human Computer Interfaces

Direct Shader Sampling in Painterly Rendering for Animation

Matt Kuruc* Chris Czyzewicz* Vijay Nair* Norman I. Badler*

University of Pennsylvania

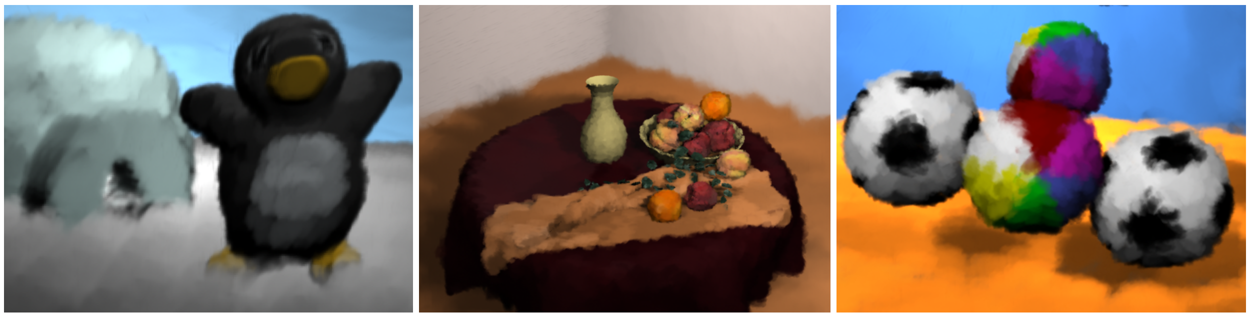

Figure 1: *Animations remain temporally coherent over animations of characters, camera movement, and colliding objects.*

1 Introduction

We present a technique for generating stroke parameters in particle-based painterly rendering algorithms that guarantees temporal coherence in animation through directly sampling shaders. To determine the appropriate color for a brush stroke, techniques like [Meier 1996] render the scene into reference pictures using traditional techniques. This reference picture is then queried by applying the camera transform to the particle's position. This transform will not always map to the correct pixel which can lead to noticeable temporal discontinuities when there is a significant color difference between two neighboring pixels. When a stroke particle lies on the edge of an object, the brush stroke will sometimes flicker between the two neighboring colors. Previously, as described in [Meier 1996], scenes would be broken up into different layers, and image processing effects were applied to reference pictures to ensure that the right color is sampled. Our method requires no post processing effects to ensure temporal coherence around these edge cases.

2 Implementation Details

As in other implementations, a particle system is employed to represent brush strokes. Particles are distributed across the meshes of triangulated geometry in the scene. Every frame, particle attributes (ie. position, color, etc.) are adjusted to reflect changes in the scene. Particles are rendered as 2-D textured billboards, composing the painterly abstraction. With the particle approach, artists have greater control over the appearance of individual objects when compared to recent video-based methods like [Bousseau et al. 2007]. These methods can be applied to animation by rendering out a complete sequence but do not offer any control over how individual objects are abstracted.

In implementations which forgo any attempts to correct the sampled color, when the camera is static and a light is moving through the scene, the imagery generated generally is temporally coherent. However, camera movement will introduce flickering behavior if there is no correction step (such as the post-processing technique referenced earlier). For this reason, we aimed to remove the camera transform from the color sampling procedure in the algorithm. Instead of the indirect method of reference pictures which could sample a pixel generated by a different shader, we directly evaluate the shader of the face on which the particle is attached. This ensures that the color assigned to the particle will be generated by the same shader every frame. Effects like shadows can be generated using shadow maps in combination with direct shader sampling.

For our offline renderer, we implemented a Maya Plug-in. The Maya API provides functionality to sample a shader given a point in space and other important properties in the shading calculations (such as surface normals). Any material generated in Maya can be sampled, giving our renderer a wide variety of ways to describe a surface.

We also generated a simple GPU implementation which evaluates basic shading calculations. With an Nvidia GeForce 8800 card, we achieved an interactive frame rate (456,750 particles, 18 FPS).

3 Conclusion

The movies we generate are temporally coherent without reliance on additional techniques to correct the sampled color. Many implementations sacrifice temporal coherence for the complexity of setting up this additional step in renders. By removing the need to apply post-processing effects to rendered layers, our method generates comparable imagery to previous techniques and is easy to implement and use for both offline and interactive applications.

Acknowledgements

Thanks to Grace Fong and Kendra Gibbons for models and animations, Pixar Animation Studios and Autodesk for the donation of software licenses for research, and the Center for Human Modeling and Simulation for its support.

References

BOUSSEAU, A., NEYRET, F., THOLLOT, J., AND SALESIN, D. 2007. Video watercolorization using bidirectional texture advection. *ACM Transaction on Graphics (Proceedings of SIGGRAPH 2007) 26*, 3.

MEIER, B. J. 1996. Painterly rendering for animation. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, ACM, New York, NY, USA, 477–484.

*e-mail: {mkuruc, skiz, vnair2, badler}@seas.upenn.edu