Brown CS2240 Final Project Proposal
Foo

TIANMU LAN, RUOLAN TANG, and KAI WANG

Fig. 1. An in game render of the game *Okami* [2]. We want to create renders with similar art styles by combining existing non-photorealistic rendering techniques.

1 INTRODUCTION

We propose to create a non-photorealistic renderer with a watercolor and ink style. *Okami* [2] is a good example of a game with such a render style, though we do not aim to replicate it exactly. To achieve such an effect, two different components need to be implemented. The first component handles the watercolor-style fills, and the second component handles the pencil/ink contours. We will explore ways to handle each component, and to combine them into a stylistically coherent renderer.

2 FEATURES

2.1 Watercolor

The simplest way to create a watercolor effect will be a cel-shader. However, a lot of additional features can be implemented to improve the results. [4] provides a good set of such features, including:

- Object-space effects, such as color bleeding technique and hand tremors effects.
- Watercolor reflectance model and pigment turbulence.
- Image-space effects, such as edge darkening, paper distortion and granulation.

We might explore addtional options as well, both from other papers and technical reports/discussions of games with similar styles.

2.2 Pencil/Ink Contour

Creating contours with a style of ink/pencil is a much more well studied field. There are studies that concerns how people draw those contours [1], as well as researches that concerns actual implementation [5] [3]. Though we have not decided upon what to do exactly, here is a list of features that we are considering:

- Contour detection and contour shaking.
- Multiple contour drawing effects.
- Pencil texture generation.
- Pencil texture rotation and 3-way blending.

2.3 Extensions

Naturally, combining methods from different works will involve a lot of occasions where we have to modify existing methods, especially if we want to create a final product that is stylistically conherent. These modification will be the major extensions that we attempt to make.

In addition, we might also try to design methods on our own, after experimenting with existing methods. Finally, it would be great if we could make the renderer real-time.

3 TIMELINE

1st week (Apr. 7th - 13th): Read relevant works and figure out mathematical details and pseudocodes for the methods we want to try out.

2nd week (Apr. 14th - 20th): Implement individual components, (likely) find more methods if previous ones don't work as intended.

3rd week (Apr 21st - 27th): Combine individual components, perform the necessary modifications to each of them.

4th week (Apr 28th - May 4th): Debug and refine the results. Make appropriate interfaces, generate results an prepare for final report and presentation.

4 DIVISION OF LABOR

As our proposed project involves many individual components, it would be straightforward to parallelize the implementation process, as each group member can implement a selection of components. Prior to this, though, we plan to read and discuss all the methods together, to ensure that all of us have a good understanding of the entire pipeline.

5 SKILLS AND EXPECTATION

We all have the necessary knowledge of the programming language to be used (C++, OpenGL) and relevant graphics concepts. We hope to develop a more in depth knowledge of NPR, and improve our skills in reading research papers and designing complex rendering systems.

REFERENCES

[1] ACM 2008. *Where do people draw lines?* Vol. 27. ACM.

[2] Capcom. 2006. Okami. (2006).

[3] Hyunjun Lee, Sungtae Kwon, and Seungyong Lee. 2006. Real-time pencil rendering. In *Proceedings of the 4th international symposium on Non-photorealistic animation and rendering*. ACM, 37–45.

[4] Santiago E Montesdeoca, Hock Soon Seah, and H-M Rall. 2016. Art-directed watercolor rendered animation. In *Proceedings of the Joint Symposium on Computational Aesthetics and Sketch Based Interfaces and Modeling and Non-Photorealistic Animation and Rendering*. Eurographics Association, 51–58.

[5] Emil Praun, Hugues Hoppe, Matthew Webb, and Adam Finkelstein. 2001. Real-time hatching. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. ACM, 581.