Real-time Animation of Rain-wet Cloth Material

Young-Joong Kwon
Yonsei University
oreo@yonsei.ac.kr

Dae-Yong Kim
Yonsei University
daeygim@gmail.com

In-Kwon Lee
Yonsei University
iklee@yonsei.ac.kr

Abstract

In this paper, we propose a method to express the change of cloth wetting in rainwater in real time. The method includes generating a mask image for displaying a raindrop region, gradually extending the mask to animate the raindrop, and applying rendering attributes such that the designated region looks like an actual raindrop. This paper focuses on the expression of rain-wet clothing, unlike previous research on the rainy scene. The results of this study make rainy scenes more realistic and can be used for real-time applications such as mobile and online games.

Keywords: rain-wet cloth material, raindrops, noise

1 Introduction

Real-time graphics have evolved rapidly in pursuit of realism. One such effort is a realistic representation of the weather, such as a rainy scene. A study on the expression of rain considering optical characteristics and the process of rain falling on the ground [1, 2, 3] has already been actively carried out, and the result has become difficult to distinguish from reality.

The appearance of the object in the rain should also be appropriately expressed to make the rain scene look more realistic. Similar studies have described the interaction of water and porous materials [4, 5, 6, 7, 8, 9], but they illustrate the process of water contact with a particular area of a porous material from a simulation perspective. These studies show realistic simulations, but simulations take too long and are difficult to use for real-time graphics.

In this paper, we focus on rain-soaked fabrics. We will discuss how to express the wetting process of rainfed material in real time. To do this, we first create a mask image to distinguish between wet and dry areas. Next, over time, an animation that gradually expands the rainprints area of the mask image is created. Finally, the rainprints area has the appropriate rendering properties applied to display the texture of the wet cloth. We will show the effectiveness of the proposed method by comparing the results obtained using commercial plug-ins with those of this paper.

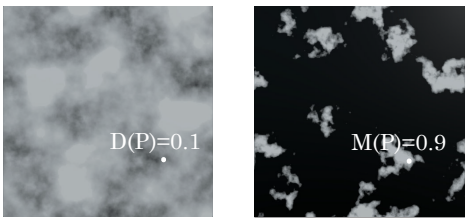
2 Proposed method

2.1 Making mask image

Let D be the 2D Perlin noise texture [10] as shown in Figure 1(a). $D(P)(\in [0,1])$ denotes the gray scale value of the texture D that is mapped according to the texture coordinates (u, v) of an arbitrary point P on the target object O. A threshold value w for determining the width of the rainprints is introduced, and the mask value $M(P)$ is set to $1 - D(P)$ when $0 \leq D(P) \leq w$ is satisfied, zero otherwise. Figure 1(b) shows the resulting rainprints-shaped mask image. For example, $w = 0.72, D(P) = 0.1 < w$, and $M(P) = 1 - 0.1 = 0.9$, which means the point P belongs to the area of the rainprints in Figure 1.

2.2 Animating rainprints

We can create the effect of extending the wetted area in the mask image by increasing the threshold w over time. Let t be the time at which the



(a) (b)

Figure 1: Making mask image: (a) Perlin noise texture D, and (b) Mask image M.

rainprints animation proceeds. The w value for a particular point P at time t can be defined by:

$$w_p(t) = [t + N(x + s_0 t, y + s_1 t, z + s_2 t)] - [N(x, y, z)]^f,$$

where N denotes a 3D Perlin noise function, $P = (x, y, z)$ denotes a position on the 3D, and f denotes a scale factor that controls the generation time difference of the rainprints between each point.

The offset term $[N(x, y, z)]^f$ is introduced to make a difference in the time at which raindrops start to be created, which allows us to generate more natural animations. Also, we use the term $[t + N(x + s_0 t, y + s_1 t, z + s_2 t)]$ to express the diffusion rate of rainwater varying with time. Here, $s_i t (i = 0, 1, 2)$ plays a role of continuously changing the key value of the noise function according to time and controls the change rate of the noise value. By doing this, water droplets can exhibit slightly different rates of water spreading along the fiber, because the density of the elements forming the fibers is not always constant. Figure 2 illustrates an animation in which the rainprints regions are expanded on the mask image.

(a) (b) (c)

Figure 2: Extension of rainprints over time: (a) t=0, (b) t=0.6, (c) t=0.7.

2.3 Rendering rainprints

The wet parts of the fabric material have three characteristics that distinguish them from non-wet parts. First, rain wet parts are darker than non rain wet parts. Second, the humidity increases the reflectivity of the rainwater portion and reflects the light better. Finally, the wetted parts show a relatively high reflection value. Therefore, we introduce the rendering equation and use the weighted sum function of Lambert BRDF and Cook-Torrance BRDF as BRDF terms for physically-based rendering [11, 12, 13].

2.3.1 Representing dark colors

The wet part of the fabric is darker than the non-wet part. To achieve this, we blend the original color with the dark color specified by the user and paint the area corresponding to the wet part. Let $M(P)$ be the mask value of the point P, A be the given dark color of the rain-wet part, B be the original color of P, β be the given mixing ratio of B when A and B are mixed. The color of the rain-wet part $T(P)$ is obtained by linear interpolation of A and B according to the ratio β, $T(P) = \beta A + (1 - \beta)B$, and $C(P) = (1 - M(P))B + M(P)T(P)$, where $C(P)$ denotes the color that the point P is to be represented. Now, the value of Lambert BRDF is finally determined by using the $C(P)$ as the surface color of the Lambert BRDF.

2.3.2 Reflectivity and specular value

The part that is wetted by rainwater has a relatively high reflectance due to moisture. To demonstrate this effect, we apply low roughness values to the wetted areas and high roughness values to the dry areas. Let J and K be the roughness value of the completely dry and completely wetted cloth materials, respectively. The final roughness value of point P can be computed as $R(P) = (1 - M(P))J + M(P)K$. Similarly, the final specular value on S can be determined by: $S(P) = (1 - M(P)Y + M(P)Z$, where Y and Z are the specular value of the completely dry and completely wetted cloth materials, respectively.

Figure 3(a) shows that when the roughness value $R = 1$, which means there is no reflec-

tion at the surface of the wet part. On the other hand, Figure 3(b) shows that when $R = 0$, the mirror-like reflection occurs on the surface of wet part. We can also see that in Figure 3(c) with higher specular values, more sharper reflections are obtained than in Figure 3(b). The final BRDF can be obtained by weighting the Cook-Torrance BRDF for the reflectance with the Lambert BRDF for the dark color simulation, and the final pixel value of P can be determined by substituting the final BRDF into the rendering equation.

(a) (b) (c)

Figure 3: Simulating wetted-part by reflectance: (a) $R = 1, S = 0.15$, (b) $R = 0, S = 0.15$, and (c) $R = 0, S = 1$, where R and S represent roughness and specular values, respectively.

3 Result

We implemented our method as a shader on Unreal Engine 4 (UE4)[14], a representative real-time rendering engine. Applying our shader to the object as shown in Figure 4 creates rain marks on the surface of the cloth and increases the area of the rain marks over time. Figure 5 shows a comparison among the commercial visual shader 'VFX Weather Pack' [15], UE4's built-in rain particles only, and our method. Our shader shows wetting clothes during the rain, which is more natural than the other results.

4 Conclusion

In this paper, we discussed how the process of wetting wet cloth can be expressed in real time through shaders. We first created a mask image that differentiates between wet and dry areas and allows wet areas to change over time. Finally, we painstakingly rendered the areas designated as wet areas by adjusting the color, reflection and specular values. In this study, we used a simple method of using a shader to represent rainwater wet cloth in real time. Adding a physical simulation here will make it more realistic, and it will be a good research topic for the future.

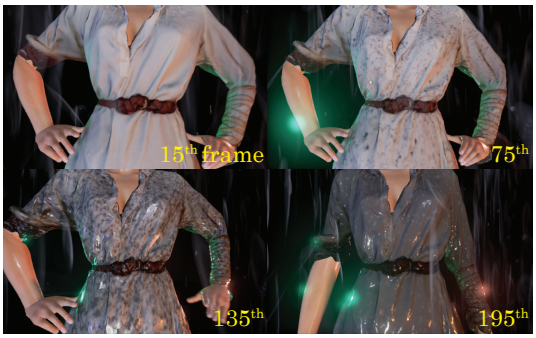


Figure 4: Animated screenshots made by applying a rain-wet shader to a character's clothes, taken at several animation frames. We can see that the wet part of the rain gradually expands over time.

Acknowledgements

This work was supported by Samsung Research Funding Center of Samsung Electronics under Project Number SRFC-IT1601-04.

References

[1] Kshitiz Garg and Shree K Nayar. Photorealistic rendering of rain streaks. In *ACM Transactions on Graphics (TOG)*, volume 25, pages 996–1002. ACM, 2006.

[2] Wang Changbo, Zhangye Wang, Xin Zhang, Lei Huang, Zhiliang Yang, and Qunsheng Peng. Real-time modeling and rendering of raining scenes. *The Visual Computer*, 24(7):605–616, 2008.

[3] Natalya Tatarchuk. Artist-directable real-time rain rendering in city environments. In *ACM SIGGRAPH 2006 Courses*, pages 23–64. ACM, 2006.

[4] Markus Huber, Simon Pabst, and Wolfgang Straßer. Wet cloth simulation. In

Figure 5: Comparison among the results from: (a) applying the VFX Weather Pack [15], the plugin of Unreal Engine 4 (UE4) [14], (b) the UE4 built-in rain particle, and (c) our shader with the UE4 built-in rain particle.

ACM SIGGRAPH 2011 Posters, page 10. ACM, 2011.

[5] Witawat Rungjiratananon, Yoshihiro Kanamori, and Tomoyuki Nishita. Wetting effects in hair simulation. In *Computer Graphics Forum*, volume 31, pages 1993–2002. Wiley Online Library, 2012.

[6] Yujun Chen, Nadia Magnenat Thalmann, and Brian Foster Allen. Physical simulation of wet clothing for virtual humans. *The Visual Computer*, 28(6-8):765–774, 2012.

[7] Mi You, Taekwon Jang, Seunghoon Cha, Jihwan Kim, and Junyong Noh. Realistic paint simulation based on fluidity, diffusion, and absorption. *Computer Animation and Virtual Worlds*, 24(3-4):297–306, 2013.

[8] Kiwon Um, Tae-Yong Kim, Youngdon Kwon, and JungHyun Han. Porous deformable shell simulation with surface water flow and saturation. *Computer Animation and Virtual Worlds*, 24(3-4):247–254, 2013.

[9] Saket Patkar and Parag Chaudhuri. Wetting of porous solids. *IEEE transactions on visualization and computer graphics*, 19(9):1592–1604, 2013.

[10] K. Perlin. An image synthesizer. In *Proceedings of SIGGRAPH*, volume 19, 1985.

[11] James T Kajiya. The rendering equation. In *ACM Siggraph Computer Graphics*, volume 20, pages 143–150. ACM, 1986.

[12] Robert L Cook and Kenneth E Torrance. A reflectance model for computer graphics. In *ACM Siggraph Computer Graphics*, volume 15, pages 307–316. ACM, 1981.

[13] Robert L Cook and Kenneth E. Torrance. A reflectance model for computer graphics. *ACM Transactions on Graphics (TOG)*, 1(1):7–24, 1982.

[14] EPIC Games. Unreal engine, https://www.unrealengine.com/.

[15] Vfx weather pack, https://www.unrealengine.com/marketplace/vfx-weather-pack.