Journal of Korea Multimedia Society Vol. 22, No. 1, January 2019(pp. 99-109)
https://doi.org/10.9717/kmms.2019.22.1.099

사실적인 비 내리는 효과 알고리즘 비교 및 분석

서태욱[†], 성만규[††]

Realistic Rainfall Effect Algorithm Comparison and Analysis

Taeuk Seo[†], Mankyu Sung[††]

ABSTRACT

Realistic rendering of natural phenomena is a difficult problem. Many environmental factors must be considered to simulate this phenomenon. At the same time, we need to think about their computational complexity to be simulated with computer algorithm One of the most difficult problems in creating weather conditions is the rain. To simulate realistic rainy scene, you have to consider the physical properties of rain and the environmental where the rain is falling down as well. In this paper, we survey the modeling and rendering techniques for realistic rainfall scenes from three different aspects. First, we list up techniques for modeling raindrop dynamics. Second, we survey the rendering techniques that render the raindrop in the environment. Third, we take a look at the hybrid methods that combines the rendering the modeling at the same time. For each aspect, we compare the algorithms in terms of implementation and their speciality.

Key words: Rain Rendering, Raindrops, Rain Streak Appearance, Rain Streak Database

1. 서 론

자연장면 시뮬레이션은 컴퓨터 그래픽의 연구 결과 중 어려운 부분이다. 비가 오는 장면을 표현하는 것은 방위 및 군사 텔레비전, 광고, 애니메이션, 영화 등 많은 분야에서 사용되고 있다. 그러나 비가 오는 장면을 촬영하는데 시간과 비용이 많이 들기 때문에 소규모 영화에서는 실행하지 못하는 부분도 있다 [27]. 실제 장면을 촬영하는 것이 가장 사실적이고 확실한 방법이지만 비용적인 측면 및 시간적인 측면에서도 비가 내리는 장면을 렌더링 되는 방법이 효율적이다. 렌더링 방법을 사용하면 계산적인 측면에서는 많은 시간이 들어가지만 가장 현실적인 효과를 도출하고 비용적인 효과도 볼 수 있다. 가상현실 시스템 및 비디오 게임과 같은 분야에서는 실시간적으로 렌더링 되는 방법이 사용된다. 자연현상을 계산하기 위해서 고려사항으로는 안개, 빗방울, 무지개, 구름, 빛 등이 포함되고 이 모든 것을 고려하기에는 어려움이 있다. 이렇게 복잡하고 물리적인 상황을 모두 고려해서 결과에 도출한다. 따라서 컴퓨터 그래픽 분야에서 비가 오는 장면의 사실적인 렌더링에서는 비가 내리는 장면뿐만 아니라 비가 오는 장면의 빗방울 모양의 변화도 연관성이 있다.

본 논문에서는 비가 내리는 장면을 구현하기 위해서 사용된 알고리즘을 비교 및 분석하는 서베이 논문이다. 논문의 목적은 크게 세 가지 분야로 나누어서 비교 및 분석한다. 첫 번째 비가 내리는 장면에서 빗방울에 대한 요소를 설명한다. 두 번째 실제 비가 내

※ Corresponding Author : Mankyu Sung, Address: 1095 Dalgubeol-daero, Dalseo-Gu, Daegu 42601, Republic of Korea, TEL : +82-53-580-6684, E-mail : mksung@kmu.ac.kr
Receipt date : Oct. 19, 2018, Revision date : Dec. 5, 2018
Approval date : Dec. 31, 2018

[†] Dept. of Computer Engineering, Graduate School, Keimyung University
(E-mail : xodnr551@nate.com)
[††] Faculty of Computer Engineering, Keimyung University

리는 장면에서 rain streaks를 추출해서 원하는 이미지 및 동영상에 적용하는 방식을 기반으로 하는 알고리즘에 관해서 설명한다. 세 번째 앞서 설명된 두 가지의 요소를 모두 고려한 알고리즘에 관해서 설명한다. 본 논문의 구성은 다음과 같다. 2장에서는 비가 내리는 장면 렌더링에서 사용되는 메서드를 소개하고 빗방울의 변화 및 움직임에 관해서 설명한다. 3장에서는 Rain Streak Image 데이터를 수집하는 부분과 수집한 이미지 데이터를 활용하는 부분에 관해서 설명한다. 4장에서는 2장과 3장에서 설명한 방법을 합친 하이브리드 방식에 관해서 설명한다. 5장에서는 결론을 도출한다.

2. Raindrop Dynamics

2.1 Physically-based methods

Kaneda이 제안하는 방향은 액체와 고체 사이의 역학관계를 고려하여 창문과 같은 유리판에서 물방울과 물줄기가 흐르는 애니메이션을 생성하는 것이다[1]. 정적의 임계 중량보다 물방울의 중량이 큰 경우에는 물방울은 기울어진 유리판 아래로 흐른다. 물방울의 흐름은 유리판 아래로 똑바로 흐르지 않지만 사행 현상의 본질 때문에 흐른 다음에 물방울이 남습니다. 이렇게 흐르는 물방울은 정적의 임계 중량보다 낮아지는 경우 멈추게 된다. 이러한 방법을 통해서 물방울과 물줄기가 움직이기 위해서 유리판의 표면은 작은 메시로 나누고 물방울은 출발 메시 점에서 다음 메시 점으로 이동한다. 중량이 식 (1)에 만족하면 물방울은 떨어진다. $m_c^s(\phi)$이 나타내는 것은 정적의 임계 중량이고 ϕ이 나타내는 것은 표면의 경사각에 해당한다.

$$m_{i,j} > m_c^s(\phi) \tag{1}$$

Tomoya 제시하는 부분은 물방울의 실시간 애니메이션을 위한 하드웨어 가속화 기법을 제시하고 피사계 심도 효과를 통해서 물체에 초점이 맞지 않는 부분은 흐리게 보이게 한다[2]. 기본이 되는 방법으로는 물방울을 반구형으로 가정한다. 반구는 다각형으로 근사 된다. 그다음 렌더링 된다. 빗방울의 색상을 결정하는 방법으로는 물방울에서 빛이 통과하면 굴절을 통해서 배경에 도달해서 렌더링 된다. 이렇게 각각의 물방울마다 렌더링이 진행되면 물방울에 배경화면이 적용된다.

Wang이 제시하는 알고리즘은 다양한 형태의 변형에 강력하고 정확하게 처리하고 평평한 부분과 그렇지 않은 부분까지 모두 처리할 수 있고 일반적인 물방울, 흐르는 물방울, 분리된 물방울, 떨어지는 물방울과 같은 여러 가지 액체 현상을 현실적으로 시뮬레이션 한다[3].

2.2 Computer vision methods

Starik가 제시하는 방법으로는 모델링과정을 통해서 비가 내리는 장면을 구현하고 구현한 부분을 비디오에 추가하는 방법을 소개한다[4]. 이렇게 하면 비디오 공간에서 시각적인 부분을 유도하고 사실적인 인상을 주게 된다.

Garg는 빗방울의 구면으로부터의 굴절 및 반사에 대한 기하학적 및 광도계 모델을 계발한다[5]. 빗방울의 모양, 크기 및 속도가 상황에 따라 다르다. 일반적으로 빗방울의 모양은 구형이다. 그러나 빗방울의 크기가 커질수록 편평한 회전 타원체가 됩니다. 빗방울에는 많은 크기의 분포가 존재하고 크기에 대해서는 일반적으로 사용되는 Marshall-Palmer 분포이다[6].

$$N(a) = 8 \times 10^{6_e^{-8200^* h^{-0.21}}} \tag{2}$$

여기서 h는 mm/hr단위의 강수량이고, a는 방울의 반경이며, $N(a)$은 간격 $(a, a+da)$내의 크기를 포함하는 단위 부피당 빗방울의 수입니다. 빗방울의 모양은 굴절 및 반사하는 환경조건에 의해서 결정된다. Fig. 1에서 굴절 및 반사가 되는 지오메트리를 보여준다. α의 광선 $\hat{v}$에 카메라의 광학 축 사이의 각이다.

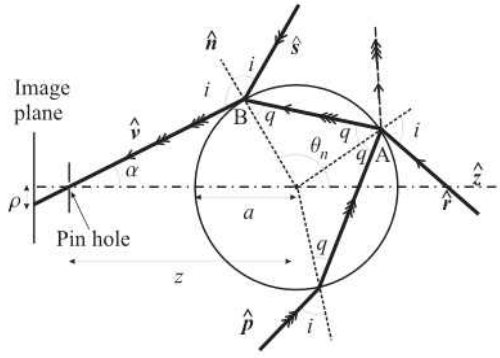


Fig. 1. Refraction, specular reflection and internal reflection by a rain drop[5]

$\hat{r}$은 낙하를 통해서 굴절된 후 카메라에 도달한다. $\hat{s}$는 방울이 내부에 반사된 후에 방울 및 광선 p에서 거울 반사 후 카메라에 도달한다. $\hat{n}$과 B는 표면 법선이고 v카메라의 방향이다. 굴절된 정반사 및 내부 반사 광선은 방울의 같은 점 B에서 나온다. 따라서 카메라 방향의 점 B의 휘도 L은 굴절, 반사 및 내부 반사로 인한 Radiances Lr, Ls, Lp의 합계이다.

Garg는 영상 시스템에 대한 비의 시각적인 효과에 대한 최초의 포괄적인 분석을 제시한다[7]. 비에 의해 생성된 이미지의 복잡한 공간적 및 시간적 강도의 변동은 다음과 같은 요소에 따라 달라진다. (a) 강하 분포 및 속도, (b) 환경 조명 및 배경의 장면, (c) 카메라의 고유 파라미터이다. 가장 먼저 빗방울의 분포와 속도를 기반으로 비의 동역학을 포착하는 상관 모델을 개발했다. 다음으로 비의 줄무늬로 인해 생성되는 밝기를 나타내는 물리 기반 motion blur 모델을 개발했다.

2.3 Other methods

Wang이 제시하는 방법으로는 실제 비가 내리는 장면의 비디오를 오프라인에서 분석과 비를 실시간 입자 기반으로 이미지에 합성하는 두 가지 부분으로 구성된다[8]. 장면의 밝기를 고려하기 위해서 Particle 시스템에 pre-computed radiance transfer (PRT)를 통해서 빗방울을 전달하는 폐쇄형 솔루션을 보여준다. 각 입자는 물의 굴절률, 임의의 빗줄기 및 위치와 속도와 같은 기타 물리적 속성을 가진 구형 모델을 가진다. 비의 색상은 즉석에서 계산되고 빗방울과 다른 물리적 속성은 구형 모델만 가지고 있다. 구현된 시스템에서는 비가 내리는 색상이 정해져 있고 알파 값은 픽셀의 빗방울에 부분적으로 계산된다.

Rousseau가 작성한 논문에서는 빗방울에서 빛에 대한 특성에 대해서는 고려하지 않는다[9]. 다만 기하 광학에 의해 정의된 특성으로 근삿값을 통해서 빛의 단색 광선의 집합으로 간주하며, 서로 다른 경계면에서 굴절과 반사가 된다. 인터페이스에서 반사의 법칙은 반사된 광선의 방향을 설명하고 스넬의 법칙은 굴절된 광선의 방향을 나타낸다.

Hao는 비가 오는 환경과 관련된 특수 효과와 알고리즘 구현을 위해 선택된 Delta 3D에 대해서 소개한다[10]. Delta 3D는 게임, 시뮬레이션 또는 그래픽 응용 프로그램에 사용할 수 있는 오픈소스 엔진이다. 오픈소스 엔진을 통해서 비가 내리는 효과를 생성하는데 두 가지 주요 개념이 있다. 첫 번째 그래픽 Particle 효과 편집기, 두 번째 물리적 속성 기반 방법이다. 첫 번째 Particle 효과에서 빗방울의 질감을 생성하고 뷰어(카메라)와 항상 마주할 수 있는 작은 Particle에 매핑 한다. 다음으로 직사각형 방사체가 특정 높이에서 아래로 빗방울을 무작위로 쏘도록 한다. 마지막으로 낙하 수명, 낙하하는 빗방울의 크기, 빗방울 입자의 생성 수를 정한다. 두 번째 개념은 물리적 특성에 기초로 한다. 작은 빗방울은 거의 구형의 모양을 가지지만 큰 빗방울은 타원체의 모양을 가진다.

MyounJae Lee가 제안하는 방법으로는 빗방울 모양을 생성하기 표면의 장력을 고려하는 방법을 사용했다[11]. 게임에서 비가 오는 장면 렌더링에는 밀크드롭(Milk Drop) 방식과 물체 표면에 빗방울이 맺혀 있는 이미지를 렌더링 되는 방법을 사용한다. 밀크드롭 방식으로는 물체 및 표면에 충돌 시 일어나는 현상을 나타내기 위해서 일정한 패턴을 통해서 미리 제작하고 충돌이 일어날 때마다 미리 제작된 패턴을 렌더링 되는 방식이다. 표면 렌더링 방법 또한 미리 표면에 맺혀있는 것을 미리 이미지로 생성하고 렌더링 되는 방법이다. 시간 면에서는 빠른 방법으로 처리되지만 모든 빗방울이 같은 모양으로 처리되므로 현실감 부분에서는 많은 어려움이 있다.

2.4 비교

사실적인 비가 내리는 효과를 주기 위해서 연구결과가 진행되었다. 앞서 설명된 알고리즘은 비가 내리는 장면 중에서도 빗방울이 어떻게 변형이 되는지를 논의된다. 빗방울은 일상적으로 생각하기에는 원형으로 생각된다. 하지만 빗방울이 낙하하면서 원형의 모습을 유지하는 것이 아니라 타원형의 모습이 된다. 타원형의 크기는 빗방울이 떨어질 때 초기의 크기일 것이다. 모양만 변화되는 것이 아니라 주변 환경에 따른 변화도 생긴다. 주변 환경요소도 사실적인 비가 내리는 효과를 주는데 있어서 어려움을 주고 있다. 그중에서 가장 민감하면서 많이 이야기되는 것이 빛에 대한 이야기다. 빛을 어떻게 활용하나에 따라서 사실적이고 자연스러운 장면이 연출된다. 빗방울을 구현하는 부분에서도 빛을 다루는 부분도 중요하게 이야기된다. 빗방울에 빛이 통과되면서 빛의 굴

절, 흡수, 반사와 같은 모든 조건을 충족해야 한다. 빛을 제외하고도 주변 환경은 존재한다. 나머지 주변 환경 요소로는 안개, 바람, 천둥, 번개와 같은 요소들이 존재한다. 이렇게 환경요소를 추가하므로 인해서 더욱더 자연스러운 장면이 연출된다. 환경요소를 추가하는 부분을 포함해서 빗방울이 물체 및 지면에 닿을 때 생기는 튀는 현상까지 모두 조정하게 되면 사실적인 비가 내리는 모습이 된다. 설명된 알고리즘은 하늘에서 빗방울이 떨어질 때 발생하는 물리적인 부분에 만족하는 알고리즘을 통해서 설명된 부분과 컴퓨터 비전 방식에 만족하는 빗방울 알고리즘 그리고 빗방울이 바닥에 닿아서 생기게 되는 요소들을 소개한다. 모든 방법이 사실적인 비가 내리는 장면을 구현하기 위해서 연구 되었다. 이렇게 소개되는 내용은 Table 1을 통해서 정리되어있다. Wang *et al.* 2006[8]과 Rousseau *et al.* 2006[9]는 Particle 시스템과 실시간으로 렌더링 하는 기법 GPU연산을 사용하는 방법이 모두 같다. 차이점으로는 빛에 대한 역할을 어떻게 처리하는지에 조금은 다르게 나타났다. Wang *et al.* 2006[8]의 논문에서는 PRT를 사용해서 빗방울에 일어날 수 있는 상황을 처리하는 방식을 사용했고 Rousseau *et al.* 2006[9]의 경우에는 빛에 대한 환경적인 요소를 고려하지 않고 단일 빛의 집합과 Physically-based modeling을 통해서 빗방울에 영향을 주었다. Hao 2008[10]에서는 Delta 3D 프로그램을 활용해서 비가 내리는 장면에 대해서 구현했다. 그와 다르게 MyounJae *et al.* 2014[11]에서는 Unity3D를 활용해서 빗방울이 물체 및 지면에 도달할 때 생기는 장력을 고려한 빗방울 생성방법을 제시한다.

3. Rain Scene Rendering

3.1 알고리즘

Garg은 비가 내리는 장면을 렌더링하기 위해서 많은 연구를 했다. 많은 연구 결과 중에서 Rain Streak Image Database를 만들어서 이미지 및 영상에 적용해서 비가 내리는 장면을 렌더링 되는 모습을 보여준다[12]. 사용한 기술로서는 Fig. 3에서 보여주는 Rain Streak Image Database를 사용하지만, Database 안에 모든 상황에 맞는 데이터를 저장할 수 없다. Database 안에는 특정 조건에 맞는 데이터만 저장되어 있다. 활용방법으로는 이미지에 사용할 각도에 근접한 8개의 데이터를 통해서 선형보관을 통해서 사용된다. 카메라에 노출되는 시간에 따라 각각의 크기에 맞는 Rain Streak을 사용하고 잘린 부분에 한해서는 부드럽게 처리하게 된다. 마지막으로 Rain Streak에 조명효과를 입혀서 이미지 및 동영상에 자연스러운 비 효과를 도출하게 된다.

Choudhury도 Garg이 구축한 Rain Streak Database를 활용해서 비가 오는 장면을 구현했다[13]. 목

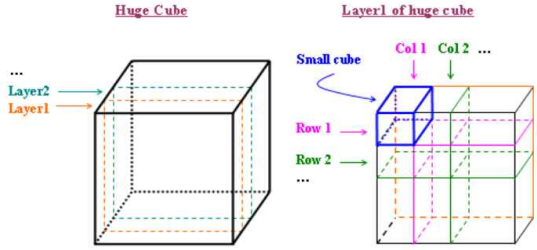


Fig. 2. Division of huge cube in L layers, R rows, and C columns in order to get L*R*C cubes[14]

Table 1. Comparison of Raindrop Dynamic Algorithms

	Implementation			
	Particle System	Real Time Rendering	GPU	Specific technique applied
Kaneda *et al.* 1993 [1]				Ray-tracing
Tomoya *et al.* 2003 [2]		○		Depth Of Field
Wang *et al.* 2005 [3]				Physically Based Animation
Starik *et al.* 2003 [4]	○			Rain Strokes
Wang *et al.* 2006 [8]	○	○	○	PRT
Rousseau *et al.* 2006 [9]	○	○	○	Physically-based modeling
Hao 2008 [10]	○			Delta 3D
1. MyounJae *et al.* 2014 [11]	○			prefab , Milk-Drop

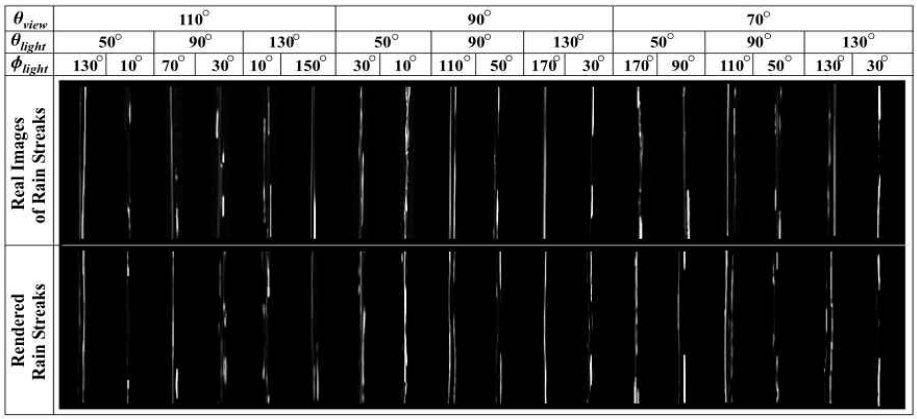


Fig. 3. Appearance of actual rain streaks and rendered rain streaks[12].

적으로는 빗방울과 관련된 굴절, 반사 및 여러 가지 상황을 고려해서 모델링 한다. 차이점으로는 첫째, 빗방울의 전처리 과정으로 Environment Matting을 사용한다. 둘째, 알고리즘의 변수로는 빗방울의 크기, 수량 및 속도와 같은 조건을 변수로 사용한다. 마지막으로 전처리 단계에서 모든 위치에 빗방울을 매치시키고 조명효과를 적용한다. 새로운 환경에서는 전역 조명을 통해서 각 빗방울에 렌더링 되는 빗방울에 굴절 및 반사를 나타낸다.

Rana의 주요 목적으로는 큐브 맵으로 세계를 모델링하고 환경 매핑 기술을 사용하여 빗방울의 굴절과 반사를 고려하고 사실적인 비가 오는 효과를 실시간으로 구현하는 것이다[14]. 사용 방법으로 전 처리 단계와 처리 단계 두 단계가 있다. 사전처리 단계에서는 거대한 큐브를 생성하고 그 가운데에 카메라를 설정한다. Fig. 2를 확인하면 거대한 큐브에 L개의 계층으로 나누고 각 계층은 R개의 행과 C개의 열로 나누어져 있으므로 거대한 큐브 안에 L * R * C 개의 작은 큐브가 있다. 다음으로 처리 단계에서는 장면의 6개의 이미지를 캡처하여 거대한 큐브 면에 이미지를 적용한다. 그다음으로 빗방울 위치를 확인하고 해당 부분에 빗방울을 렌더링 한다.

Weber이 목표로 하는 부분은 비가 오는 장면의 사실성을 높이기 위해 실시간으로 나무와 비 사이의 복잡한 상호 작용에 대한 시각적 효과를 주는 렌더링을 목표로 한다[15]. 나뭇잎에서 빗방울을 통과 후 나뭇잎에서 떨어지는 빗방울 렌더링 방법을 도입한다. 알고리즘은 효율적인 through fall(TF)을 렌더링하고 사용자 제어를 보장한다. 일단 비가 내리는 강도와 나무 매개 변수가 설정되면, 물의 진행 방향을 나타내는 모델은 각각의 나무 모델에 대해서 물방울이 떨어지는 TF의 시각적, 공간적 및 크기 분류를 on-the-fly로 계산한다.

3.2 Rain Streak Image Database

실제 비가 내리는 화면에서 조명, 방향, 하이라이트 등 물리적으로 많은 부분을 고려해야 한다. 이러한 상황을 고려해서 비가 오는 장면을 구현하기 위한 모델을 개발했다. 다양한 시야, 각도, 방향을 통해서 Fig. 3을 확인하면 Rain Streak Image를 확인할 수 있다. Fig. 3을 설명하기 이전에 Fig. 4에 대해서 설명한다. Database를 구축을 위해서 사용되는 좌표계이다. θ_v 카메라의 시야 방향이고 각도는 (70, 90, 110)으로 설정된다. θ_l 조명의 고도 각이고 각도는 (50, 90, 130)으로 설정된다. ψ_l 방위각이고 각도는 10에서 170까지 20도 단위로 측정한다. 이렇게 설정된 각도

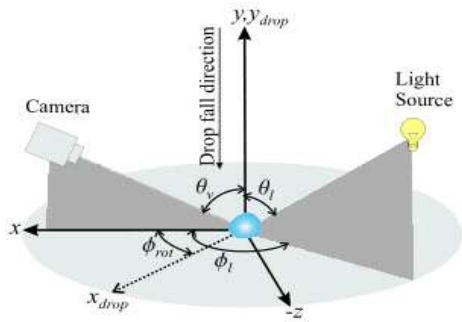


Fig. 4. Coordinate system used for both the measurement and the rendering of rain streaks[12].

를 통해서 Database를 구축한다. Fig. 3으로 돌아가서 첫 번째 표시된 부분은 카메라의 시야 방향을 가리키고 두 번째 줄은 조명의 고도 각도이고 세 번째 줄은 방위각이 된다. 이렇게 설정된 각도의 조건을 통해서 모은 이미지는 Fig. 3 윗부분에서 확인할 수 있다. 수집된 이미지에서는 진동 매개 변수를 고려하지 않고 수집하고 아랫부분은 넓은 범위의 진동 매개 변수를 사용해서 렌더링 이미지이다. 결과적으로 실제 비가 내리는 이미지와 렌더링이 진행된 이미지가 흡사하다는 결과를 도출한다.

3.3 비교

비가 내리는 환경을 구현하기 위해서 이번에 소개되는 알고리즘으로는 빗방울과 비를 직접 구현하기보다는 실제 비가 내리는 장면을 이미지로 저장해서 이미지를 활용해서 사실적인 비가 내리는 장면을 구현한다. 저장된 이미지는 Database에 저장이 되고 비가 내리는 상황에 맞는 이미지를 사용하게 된다. 실제 비가 내리는 이미지와 Database에 저장된 이미지는 거의 흡사한 결과물을 도출한다. 모든 데이터를 저장하기에는 무리가 있지만, 일정 규칙을 정해서 실제 비가 내리는 장면을 이미지로 저장하고 사용자가 원하는 부분에 의해서는 Database에 저장된 데이터를 통해서 근사한 규칙을 합쳐서 결과 값을 도출한다. 이렇게 저장된 데이터를 토대로 이미지 또는 비디오 환경에서 비가 내리는 장면을 구현할 수 있다. 구축된 Database를 통해서 간단하고 적은 계산 비용으로 비가 내리는 장면이 구현할 수 있다. Rain Streak Database의 기반으로 비가 내리지 않는 일반 이미지 및 동영상에 적용하는 과정에서 주변에 빛을 사용해서 Rain Streak Image에 배경 빛에 대한 색감도 입힐 수 있다. 빛을 사용하는 방법을 넘어서서 비가 내리는 장면의 중요 부분인 빗방울에 대해서도 굴절 및 반사와 같은 상황도 고려된다. 이미지 및 동영상의 공간을 넘어서 더 많이 사용되고 있는 분야인 가상의 공간에서도 구현할 수 있다. 가상의 공간에서 작업하기에 앞서서 사전작업으로는 큰 큐브와 같은 공간을 생성한다. 거기에 일정한 크기의 큐브로 나눈다고 생각을 하고 일정한 크기로 큰 큐브를 나눈다. 그런 다음 큐브에 가상의 공간에 추가할 이미지를 적용하고 이미지에 맞는 환경요소도 고려해서 추가한다. 모든 작업을 통해서 가상의 공간에서도 사실적인 비가 내리는 환경도 구축할 수 있다. 나아가서 가상의 환경에서 다른 상황인 숲과 같이 나무가 많고 고려해야 할 사항이 많은 부분에서도 비가 내리는 장면에 대해서 연구 되었다. 숲과 같은 환경을 구성하기 위해서 나무 모델과 빗방울 모델을 통해서 숲에서 비가 내리는 장면도 구현할 수 있다. 나뭇잎은 빗방울을 흘려보내기만 하는 것이 아닌 빗방울이 투과되어서 떨어지는 요소까지 고려가 필요하다. 이렇게 소개된 환경과 요소를 통해서 조금 더 사실적인 비가 내리는 렌더링 알고리즘이 소개되었다. 소개된 알고리즘은 Table 2를 확인하면 정리되어있다. Grag *et al.* 2006 [12]와 Choudhury *et al.* 2009[13]의 경우에는 사용하는 기법이 같다고 볼 수 있다. 그 이유로는 Grag *et al.* 2006[12]의 기반으로 작성되었다. 차이점으로는 실제 장면에서 일어날 수 있는 상황을 고려해서 작성된 논문이다. Rana 2014[14]는 큐브에 모든 장면을 적용하는 것을 목표로 한다. 그래서 앞에서 설명되는 논문과 다르게 Environment Mapping을 통해서 비가 오는 장면에 대해서 적용한다. Weber *et al.* 2016 [15] 실제 나무가 많은 공간에서 빗방울 렌더링을 통해서 상호작용 되는 장면을 보여준다. 더욱 효율적인 렌더링을 위해서 through fall이라는 메서드를 사용해서 나무가 많은 장면에서 비가 내리는 장면을 구현한다. 앞서 설명된 논문들과 같이 Parti-

Table 2. Comparison of Rain Scene Algorithms

	Implementation				
	Particle System	Texture Database	GPU	Real time Rendering	Specific Technique applied
Garg *et al.* 2006 [12]	○	○			
Choudhury *et al.* 2009 [13]	○	○			
Rana 2014 [14]	○			○	Environment Mapping
Weber *et al.* 2016 [15]	○		○	○	throughfall(TF)

cle 시스템을 사용해서 비가 내리는 장면을 보여주지만 Database를 사용하기 보다는 through fall이라는 메서드를 통해서 구현한다는 다른 점을 가지고 있다.

4. Hybrid Approaches

4.1 알고리즘

Anna이 소개하는 방법으로는 가상 환경에서 시뮬레이션하기 어려운 비 및 대기 현상을 실시간으로 사용되는 방법에 관해서 설명된다[16]. 주요 목표 중에서는 가상 환경이나 게임 속에서 사용자가 계속 움직이는 장면에 적합한 시뮬레이션을 제공하는 것이다. Particle System을 최초에 비 시뮬레이션에 사용하고 GPU를 통해서 빛의 상호 작용 없이 비를 물리적인 움직임 모델을 사용 후 빗방울에 대한 충돌 검사 방법도 제시했다[17,18]. Particle System을 사용에 있어 가장 큰 단점으로는 현실감을 주기 위해서 많은 수의 입자를 시각화하는 것이다. LOD 기술을 통해서 입자의 크기와 위치 수정을 하면서 실제 비가 오는 모습을 유지하게 된다. 사용 단계로는 정의된 구역에서 렌더링을 위한 Particle System을 초기화하고, 범위에서 벗어나는 입자가 있을 때마다 재사용되고 다시 입자의 초기 위치를 정의한다. 사용자의 움직임에 따라 비 컨테이너의 위치 업데이트와 사용자의 위치를 추적한다.

Rousseau은 비를 시뮬레이션하면 많은 문제가 발생할 수 있는 문제를 해결하고 비디오 게임에서 완벽한 비 시뮬레이션이 가능한 프레임 워크를 제공한다[19]. GPU 기반의 입자 애니메이션의 단계로는 첫 번째 Particle 위치는 GPU 메모리 내부의 부동 소수점 Texture에 저장된다. 두 번째 위치는 Fragment Shader를 통해서 각 애니메이션 단계마다 업데이트한다. 세 번째 질감은 입자를 표시할 때 입자의 실제 위치를 얻는 데 사용된다. 다음으로 렌더링 부분에서의 핵심 분야는 미리 굴절되는 방향에 따라 달라지는 광각 Texture 입자를 계산해두고 매핑 하는 것이다.

Coutinho은 Digital terrain model(DTM)을 포함하는 장면에 비가 내리고, 흘러내려 가고, 빗방울이 튀는 시뮬레이션을 통합하여 계산하는 프레임 워크를 설명한다[20]. 사용되는 방법으로는 유체 시뮬레이션을 위한 Navier-Stokes 방정식과 Smooth Particle Hydrodynamics(SPH), 자유 표면의 정규 표현 및 현실적인 비 장면 애니메이션을 위한 입자 기반 프레임 워크이다.

Carles이 소개하는 방법으로는 비가 내리는 줄무늬와 빗방울에 복잡한 조명효과를 실시간으로 주고 안개 후광 및 빛의 광선을 참여 미디어로 사용된다[21]. 목적으로는 예술가가 원하는 비의 분포 및 장면의 다른 영역에서 강도가 다르거나 시간에 따른 비의 분포를 허용한다. 빗방울의 충돌 감지가 가능한 통합 구조를 통해서 여러 가지 상황과 조건에 맞는 효과를 추가할 수 있다. 전 처리 과정에서 실시간 시뮬레이션에 필요한 비가 내리는 모습을 Texture로 구성된다. Particle은 렌더링 중에 떨어지는 빗방울을 애니메이션으로 생성하고 표면에 도달할 때 튀는 효과를 만드는 데 사용된다. 아티스트가 생성하는 분포에 따라서 장면에 배치가 이루어지고 복잡한 조명효과의 경우에는 사전에 계산을 통해서 저장된다. 비가 내리는 줄무늬나 빗방울의 튀김을 렌더링 전에 안개, 후광 및 빛과 같은 처리는 실시간 이미지 기반 효과가 추가된다. Fig. 5를 확인하면 비가 내리는 Particle을 표시하게 된다. 먼저 가장 위에서 내리는 파란색 부분은 비가 내리기 시작하는 부분이고, 분홍색 경계 부분은 비가 표면 또는 물체에 도달하는 부분이다. 비가 내리는 부분이 모두 같은 길이를 가질 수 없기에 표면 또는 물체에 맞은 빗줄기는 주황색으로 표시되어서 다시 파란색 부분으로 가게 돼서 다시 비가 내리게 되는 것이다. 각각의 빗방울이 같은 시간에 작업하는 것이 아니라 상황에 맞게 적용되어서 진행된다.

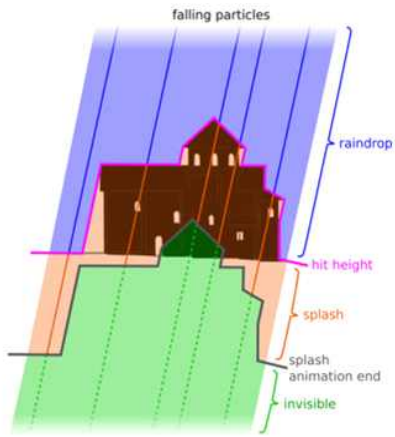


Fig. 5. Particle States[21].

Esther이 설명하고자 하는 부분은 비와 눈이 내리는 부분을 애니메이션으로 표현하고 프로젝트에 사용되는 방법을 제시한다[22]. Puig-Centelles이 제시한 방법은 사용자가 입자 시스템을 사용해서 계속 움직이는 비가 내리는 Rain Streak을 실시간으로 묘사한다[23]. 세부적으로 확인을 하면 시각에서 멀어지면 빗방울의 개체의 수는 줄어든다. LOD의 경우 비가 내리는 장면을 시뮬레이션하기 위해서 사용되는 입자의 숫자를 제어하는 역할을 한다. 사실적인 비가 내리는 효과를 주기 위해서 빗방울의 입자를 수직의 줄무늬로 재현성 한다. 그 이유는 인간의 눈은 종종 비가 내리는 모습을 빗방울을 보고 확인하기 보다는 비가 내리는 줄무늬를 확인해서 비가 내리는 것을 감지하기 때문이다. 이렇게 형성된 줄무늬의 빗방울에 내부의 반사, 굴절, 조명 방향등 외적인 조건을 주어서 렌더링 한다.

Wang이 추구하는 방향으로는 현실성, 실시간성, 자원의 소비에 대한 최적화를 하는 것을 목표로 한다[24]. 실제의 자연현상 중에서 비가 내리는 장면은 매우 복잡하다. 수십억 가지의 빗방울이 있고, 빗줄기가 각기 다른 방향과 반경으로 서로 다른 모양의 빗방울과 모양을 가지고 주변의 상황 따라 각기 다르게 변화하는 부분까지 모두 고려해야 한다. 입자 시스템에서는 단일 Voxels를 통해서 빗방울 입자를 여러 가지 상황에 맞게 시뮬레이션하고 서로 다른 비의 장면을 생성하는 것을 제어했다.

Yoann의 목표로는 비가 오는 장면에서 지역 속성 및 전역 속성을 고려한 실시간 렌더링을 위해서 일관된 멀티 스케일 모델을 제시한다[25]. 실제 비가 내리는 장면에서 패턴의 스펙트럼 분석을 기반으로 이미지 공간의 절차적인 줄무늬 생성 및 비가 내리는 강도의 정확한 제어를 위해서 눈에 보이는 줄무늬만을 관리한다. 시야에 보이는 비가 내리는 줄무늬의 밀도와 시야에서 멀어져서 가시성이 약화하는 물리적 상관관계를 사용하고, 빗방울의 밀도를 평가하고 장면에서 가시성이 손실되는 부분에 한 번만 배포가 되어 사용된다.

4.2 비교

Hybrid Algorithms은 기존의 비가 내리는 알고리즘과 다르게 비가 내리는 환경요소인 빛의 굴절, 주변 환경의 빛, 자연현상과 같은 요소와 빗방울이 내려오면서 변화되는 부분과 Rain Streaks Image를 활용해서 비가 내리는 장면에 적용하는 부분까지 모두 고려한다. 이렇게 고려된 환경에 사실적인 비가 내리는 이미지를 Particle에 어떻게 적용하는 부분에 대해서 많은 연구 결과를 진행했다. 단순하게 비가 내리는 이미지를 Particle에 적용하는 것이 아니라 적용 후 사실적으로 보여주고 시각적인 부분에서도 큰 노력을 했다. Particle에 이미지를 적용하고 GPU에서는 환경적인 요소인 빛, 주변의 배경에서 나오는 색상, 안개등과 같은 요소를 계산해서 추가했다. 다른 부분으로는 빗방울이 떨어지고 튀기면서 발생하는 모든 부분에 대한 프레임 워크를 제시해서 비가 내리는 장면을 구현하는 방법을 사용하는 모습도 보여준다. 비가 내리는 화면에서 빗방울이 물체 및 지면에 도달하는 시간은 각각 다르다. 모든 빗방울을 같은 시간에 같이 처리하기보다는 각각의 빗방울이 물체 및 지면에 도달하면 다시 하늘에서 떨어지는 방식을 통해서 조금 더 사실적인 비가 내리는 장면과 계산 시간을 줄이는 방법을 제시하기도 했다. 이렇게 여러 가지 방면에서 많은 연구 결과를 통해서 지금까지 사실적인 시각을 주기 어려운 분야인 비가 내리는 장면을 구현했다. 이로 인해서 많은 연구 결과는 사실적이고 실시간성을 모두 가진 연구 결과 결과를 도출했다. 모든 연구 결과는 기존의 연구에서 해결하지 못한 부분을 서로 다른 측면에서 해결책을 제시했다. 이러한 연구 결과를 통해서 사실적인 비가 내리는 시뮬레이션이 가능하게 되었다. 이렇게 많은 연구 결과와 좋은 방식의 알고리즘이 나오기까지 가장 큰 이유로는 이전과 다르게 많이 발전된 컴퓨터 그래픽 카드의 성능이 큰 역할을 했다. 이로 인해서 비가 내리는 장면의 시뮬레이션에서 사용되는 메모리 관리와 이미지 및 시뮬레이션에 렌더링 되는 알고리즘의 계산 시간도 단축이 많이 되었다. 이렇게 사용되는 알고리즘과 성능이 좋은 그래픽카드로 인해서 많은 연구 결과는 실시간성과 사실성이 증가하는 결과를 도출했다. 이번 장에서 설명된 대부분 알고리즘은 GPU 연산을 통해서 실시간성을 높이는데 많은 이바지를 했다. 다음으로 이번 장에 소개된 알고리즘 분석한 부분을 확인하기 위해서는 Table 3을 확인하면 된다. 소개된 알고리즘 모두는 Particle 시스템을 활용해서 비가 오는 장면을 렌더링하고 GPU 연산을 통해서 비가 내리는 장면에 해당하는 요소를 계산한

Table 3. Comparison of Hybrid Algorithms.

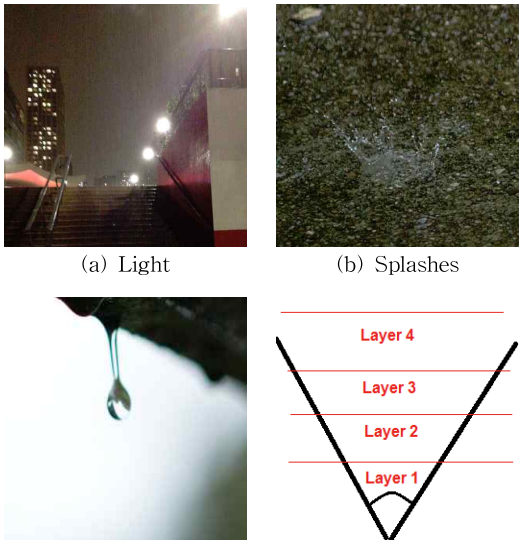
	Implementation					Performance
	Particle System	Texture Database	GPU	Level Of Details	Ray Tracing	
Anna *et al.* 2008 [16]	○	○	○	○		
Rousseau *et al.* 2009 [19]	○		○			
Coutinho *et al.* 2010 [20]	○		○			
Carles *et al.* 2013 [21]	○	○	○			
Esther 2014 [22]	○	○	○	○		
Wang *et al.* 2015 [24]	○	○	○		○	Real Time
Yoann *et al.* 2015 [25]	○	○	○			Real Time

다. 추가로 대부분 알고리즘은 3장에서 소개된 데이터베이스를 사용한다. Anna *et al.* 2008[16]에서 사용된 LOD는 입자의 크기와 위치를 수정하는 데 사용하고 Esther 2014[22]에서는 비가 내리는 장면에서 입자의 숫자를 제어하는 역할을 한다. 이렇게 같은 기법을 사용하여도 해결하고자 하는 방향에 따라서 사용되는 역할이 달라진다. Carles *et al.* 2013[21]은 Texture Database를 확장해서 비가 내리는 이미지를 구축했다. 빛에 대한 작용과 실제 비가 내릴 때 나타날 수 있는 상황을 모두 고려해서 작업했다. 추가로 아티스트가 쉽게 사용하고 같이 비가 내리기보다는 영역마다 비가 다르게 내리는 상황까지 만들었다. Wang *et al.* 2015[24]에서는 다른 알고리즘과 복잡한 빗방울처리와 사실적인 비가 내리는 장면을 위해서 번개와 바람 같은 요소를 추가하고 빛 처리를 위해서 고전적인 기법인 Ray-tracing 기법을 사용했다.

5. 결 론

본 논문에서는 자연현상 중에서 많은 어려움과 문제점을 가지고 있는 부분인 비가 내리는 효과 렌더링에 관해서 설명한다. 기상효과 중에서 비가 내리는 효과는 많이 사용되지만, 빗방울이 떨어지는 화면을 사실적으로 보여주는 부분에 대해서 어려움이 있었다. 비가 내리는 효과에서 비 줄무늬와 함께 빗방울에 생기는 여러 가지 제약조건과 수많은 처리가 이루어져야 한다. 서로 다른 방향으로 내리는 빗줄기와 빗방울이 표면에 충돌이 이루어지는지 실제 비가 내리는 수많은 상황에 대해서 고려한다. 이렇게 비가 내리는 장면을 렌더링하기 위해서 가장 많은 부분을 차지하는 부분은 실제 비가 내리는 장면에서 얻은 정보를 토대로 작성된 Rain Steak Image Database이다. Database를 통해서 비가 내리는 효과를 주는 장면을 렌더링 한다. 이렇게 구축된 데이터는 조금 더 사실적인 상황을 연출했다. 이렇게 기존의 Database만을 활용하는 것이 아닌 주어진 상황에 맞는 여건을 충족시키면서 점차 연구 결과를 진행했다. 비가 내리는 효과에서도 촬영하는 장소 및 시야에 따라서 비가 내리는 모습 또한 많은 변화가 생기게 된다. 가까이에 있는 Rain Steak는 확대한 듯 보일 것이고 시야에서 멀어지는 장면에서는 Rain Steak 또한 축소되는 효과와 함께 흐리게 보여 준다. Rain Steak의 효과를 통해서 시야에 보이는 부분이 완성되었다면 비가 내리는 효과를 한층 더 시각화하는 부분에서는 빗방울이 떨어지면서 굴절, 반사, 빛의 영향에 한 상황을 모두 고려 후 물체 및 표면에 빗방울이 충돌하게 되면 나타나는 효과도 주어야 한다. 이렇게 모든 효과와 상황을 고려하게 되면 사실적인 비가 내리는 렌더링이 완성된다. 컴퓨터 그래픽 분야에서 현재까지 기초연구 결과를 통해서 가장 사실적인 비가 내리는 장면이 렌더링 되는 알고리즘을 설명했다.

Fig. 6을 확인하면 비가 내리는 장면을 구현하기 위해서 사용되는 환경적인 요소를 확인할 수 있다. (a)는 실제 환경에서 비가 내리는 모습이다. 비가 내리는 장면에서 조명은 밤에 많은 영향을 준다. (b)는 물방울이 표면에 닿아서 생기는 모습이다. 비가 내리는 장면에서는 튀김 현상을 일일이 처리 하지 못하므로 미리 여러 가지의 상황을 생성해서 사용한다. (c)는 물방울의 모습이다. 비가 내리는 장면에서 자세하게 봐야 보이고 실제 많은 사람은 비가 내리는 모습



(a) Light (b) Splashes

(c) Raindrops (d) Layer

Fig. 6. Environmental factor.

Fig. 7. Raining scenes applied to the game environment [26].

을 빗줄기로 생각한다. (d)는 사실적인 비가 내리는 장면을 구현하기 위해서 가장 필수적이다. 각 layer에 사전에 정의된 같은 빗방울 motion을 사용하고 layer마다 다른 시차를 주어서 사실적인 효과를 준다. 이렇게 설명된 속성을 모두 고려된 장면이 Fig. 7에 해당한다.

REFERENCE

[1] K. Kaneda, T. Kagawa, and H. Yamashita, "Animation of Water Droplets on a Glass Plate," *Models and Techniques in Computer Animation*, pp. 177-189, 1993.

[2] T. Sato, Y. Dobashi, and T. Yamamoto, "A Method for Real-time Rendering of Water Droplets Taking into Account Interactive Depth of Field Effects," *Entertainment Computing*, pp. 125-132, 2003.

[3] H. Wang, P.J. Mucha, and G. Turk, "Water Drops on Surfaces," *Association for Computing Machinery Transactions on Graphics*, Vol. 24, No. 3, pp. 921-929, 2005.

[4] S. Starik and M. Werman, "Simulation of Rain in Videos," *Proceeding of International Conference on Computer Vision*, Vol. 2, pp. 95-100, 2003.

[5] K. Garg and S.K. Nayar, "Photometric Model of a Rain Drop," *CMU Technical Report*, 2003.

[6] J.S. Marshall and W.M.K. Palmer, "The Distribution of Raindrops with Sizes," *Journal of Meterology,* Vol. 70, No. 327, pp. 165-166, 1948.

[7] K. Garg and S.K. Nayar, "Detection and Removal of Rain from Videos," *Proceeding of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition,* Vol. 1, pp. 1-1, 2004.

[8] L. Wang, Z. Lin, T. Fang, X. Yang, X. Yu, and S.B. Kang, *Real-time Rendering of Realistic Rain*, Association for Computing Machinery's Special Interest Group on Computer Graphics and Interactive Techniques, pp. 156, 2006.

[9] P. Rousseau, J. Vincent, and G. Djamchid, "Realistic Real-time Rain Rendering," *Computers and Graphics,* Vol. 30 No. 4, pp. 507-518, 2006.

[10] H. Pisith, *Algorithms for Atmospheric Special Effects in Graphics and Their Implementation*, PhD Thesis. Indian Institute of Technology, Bombay Mumbai, 2008.

[11] M.J. Lee and K. Kim, "Implementation of Raindrop Rendering Using Unity 3D Engine," *Journal of Digital Convergence,* Vol. 20 No. 1, pp. 519-524, 2014.

[12] K. Garg and S.K. Nayar, "Photorealistic Rendering of Rain Streaks," *Association for*

Computing Machinery Transactions on Graphics, Vol. 25. No. 3. pp. 996-1002, 2006.

[13] B. Choudhury, H. Pisith, and C. Sharat, "Real-time Droplet Modeling Using Color-space Environment Matting," *Association for Computing Machinery's Special Interest Group on Computer Graphics and Interactive Techniques, Posters,* pp. 78, 2009.

[14] A. RANA, "Rain: Graphical Rendering," *International Journal of Interdlisciplinary Research and Innovations,* Vol. 2, No. 2, pp. 30-38, 2014.

[15] Y. Weber, V. Jolivet, G. Gilet, K. Nanko, and D. Ghazanfarpour, "A Phenomenological Model for Throughfall Rendering in Real Time," *Computer Graphics Forum,* Vol. 35, No. 4, pp. 13-23, 2016.

[16] P.C. Anna, O. Ripolles, and M. Chover, "Multiresolution Techniques for Rain Rendering in Virtual Environments," *Proceeding of International Symposium on Computer and Information Sciences,* pp. 1-4, 2008.

[17] T. Yoshihiko, K. Kensuke, and T. Katsumi, "A Method for Rendering Realistic Rain-fall Animation with Motion of View," *Information Processing Society of Japan Special Interest Group*, No. 105-005, pp. 21-26, 2001.

[18] Z.X. Feng, M. Tang, J.X. Dong, and S.C. Chou, "Real-time Rain Simulation," *Proceeding of International Conference on Computer Supported Cooperative Work in Design, Springer,* pp. 626-635, 2005.

[19] P. Rousseau, V. Jolivet, and D. Ghazanfarpour, "Gpu Rainfall," *Journal of Graphics Tools*, Vol. 13, No. 4, pp. 17-33, 2008.

[20] B.B. Coutinho, A.A. Oliveira, Y.P. Atencio, and G.A. Giraldi, "Rain Scene Animation Through Particle Systems and Surface Flow Simulation by SPH," *Proceeding of Brazilian Conference on Graphics, Patterns and Image Processing,* pp. 255-262, 2010.

[21] C. Creus and G.A. Patow, "R4: Realistic Rain Rendering in Realtime," *Computers and Graphics,* Vol. 37, No. 1-2, pp. 33-40, 2013.

[22] E. de Jong, "Rain and Snow Flurries," Msc Computer Animation and Visual Effects, 2014.

[23] A. Puig-Centelles, O. Ripolles, and M. Chover, "Creation and Control of Rain in Virtual Environments," *The Visual Computer*, Vol. 25, No. 11, pp. 1037-1052, 2009.

[24] C. Wang, M. Yang, X. Liu, and G. Yang, "Realistic Simulation for Rainy Scene," *Journal of Software*, vol. 10, no. 1, pp. 106-115, 2015.

[25] Y. Weber, V. Jolivet, G. Gilet, and D. Ghazanfarpour, "A Multiscale Model for Rain Rendering in Real-time," *Computers and Graphics*, Vol. 50, pp. 61-70, 2015.

[26] Game Environments-part B : Rain, https://www.fxguide.com/featured/game-environments-partb/ (accessed June, 5, 2013).

[27] 윤정수, 윤성의, "물리적 특성을 고려한 빠른 번개 렌더링", 한국컴퓨터그래픽스학회 2016년 KCGS 학술대회 논문집, 2016.7, pp. 133-141.

서 태 욱

2016년 2월 Tarlac State University, 학사
2016년 3월~현재 계명대학교 컴퓨터공학전공 석사과정
관심분야 : 컴퓨터 그래픽스, HCI

성 만 규

1993년 2월 충남대학교 전산학과 학사
2005년 12월 (미) 위스콘신대학 컴퓨터사이언스 석사
2005년 12월 (미) 위스콘신대학 컴퓨터사이언스 박사
2006년 2월~2012년 2월 한국전자통신연구원(ETRI) 선임연구원
2012년 3월~현재 계명대학교 게임모바일공학전공 부교수
관심분야 : 컴퓨터 그래픽스, 컴퓨터 애니메이션, 컴퓨터 게임, HCI