Snell의 법칙을 이용한 효율적인 비눗방울 형상 표현 및 배경 합성

정유진[O], 김종현[*]
[O]인하대학교 소프트웨어융합대학 디자인테크놀로지학과,
[*]인하대학교 소프트웨어융합대학 디자인테크놀로지학과
e-mail: jonghyumkiun@inha.ac.kr

Efficient Representation of Soap Bubble Shapes using Snell's Law and Background Synthesis

Yoojin Jeong[O], Jong-Hyun Kim[*]
[O]College of Software and Convergence (Dept. of Design Technology), Inha University,
[*]College of Software and Convergence (Dept. of Design Technology), Inha University

● 요 약 ●

본 논문에서는 비눗방울에 나타나는 반사와 굴절 효과를 상용 게임엔진인 유니티 쉐이더(Unity shader)를 사용하여 구현하고 다양한 배경에서 합성할 수 있는 효율적인 프레임워크를 제안한다. 본 논문에서 제안하는 방법은 계산량이 큰 유체 시뮬레이션을 이용하지 않고, 스넬(Snell)의 법칙을 이용하여 박막 내부의 굴절 벡터를 계산하고, 막 표면의 표현을 위해 다양한 텍스처(Texture)를 적용하였으며, vertex의 조정을 통해 비눗방울 자체의 움직임을 나타낼 수 있다. 결과적으로 실시간으로 높은 품질의 비눗방울을 표현할 수 있기 때문에 게임뿐만 아니라 가상현실 및 다양한 실시간 애플리케이션에 활용될 수 있다.

키워드: 비눗방울(Soap bubble), 유체 시뮬레이션(Fluid simulation), 스넬의 법칙(Snell's law), 유니티3D(Unity3D), 유니티 쉐이더(Unity shader)

I. Introduction

비눗방울은 굉장히 얇은 막인 박막으로 형성되어 있다. 박막의 내부와 외부는 서로 다른 매질로 이루어져 있고, 파동은 매질의 성질에 따라 다른 속도를 가진다[1]. 이때 속도의 차이로 인해 빛의 진행 방향이 변화하게 되는데 이를 굴절 현상이라 한다. 본 논문은 비눗방울 상의 반사와 굴절 현상을 효과적으로 구현하기 위해 쉐이더 프로그램을 활용하는 방법을 기술하고자 한다.

II. The Proposed Scheme

1. Soap bubble shader

1.1 Law of Snell

스넬의 법칙은 광선이 한 매질에서 다른 매질로 입사할 때 굴절되는 현상을 설명하는 물리 법칙이다[2]. 스넬의 법칙에 따르면, 광선이 매질에서 다른 매질로 입사할 때, 두 매질의 굴절률과 입사각, 굴절각 사이에 다음과 같은 관계가 성립한다 (Fig. 1과 수식 1 참조).

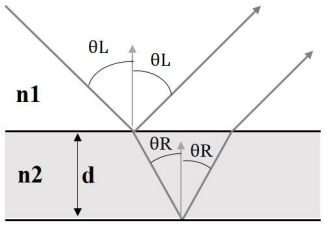

Fig. 1. Snell' s law formulation.

$$\gamma = n_1 \sin(\theta_l) = n_2 \sin(\theta_r) \quad (1)$$

$$\sin(\theta_r) = \frac{n_1}{n_2} \sin(\theta_l) \quad (2)$$

여기서 n_1은 첫 번째 매질의 굴절률, n_2는 두 번째 매질의 굴절률을 의미한다. 막 표면에서의 반사각과 매질 내부의 굴절각은 각각 θ_l과 θ_r이다.

수식 1을 치환하면 수식 2와 같이 나타낼 수 있고, 이 수식은 두 매질의 굴절률과 외부 막의 반사각을 안다고 가정했을 때, 내부

막에서의 굴절각을 계산할 수 있다는 것을 의미한다. 결과적으로 이 식을 통해 광선이 매질 간 경계면을 통과할 때 굴절각이 어떻게 변하는지 계산할 수 있다. 굴절률이 다른 두 매질 사이에서 광선의 진행 방향이 바뀌는 이유는 빛의 속도가 매질에 따라 변하기 때문이다.

1.2 Vector operation in soap bubbles

비눗방울의 반사와 굴절을 표현하는 알고리즘은 다음과 같다. 우선 기본적인 상수인 n_1과 n_2를 설정해야 한다. 비눗방울 시뮬레이션에서 이들은 각각 공기의 굴절률과 비누 막의 굴절률을 의미한다. 또한 비누 막의 두께를 위쪽과 아래쪽으로 구분하여 시간이 흐를수록 위쪽 두께인 μ_{top}은 감소하고, 아래쪽의 두께인 μ_{bottom}는 증가하도록 했다 (수식 3 참조).

$$\mu_{top} = lerp(1, 0.1, \alpha) \\ \mu_{bottom} = lerp(0.1, 1, \alpha) \tag{3}$$

위 수식에서 α는 다음과 같다 : $\alpha = saturate(\mu^{speed} Time.y)$. μ^{speed}는 두께 변화 속도를 조정하는 상수이며, 각각의 두께는 선형 보간을 이용하여 시간이 지남에 따라 변화하게 했다. 막의 두께는 곧 반사와 굴절에 영향을 준다.

다음으로 3D 공간에서 카메라 시점인 뷰 벡터(View vector)와 법선 벡터(Normal vector)를 내적하여 카메라 위치에 따라 표면에 반사되는 빛의 양이 변하도록 계산한다 (수식 4 참조).

$$\gamma_{outside} = N \bullet V \tag{4}$$

여기서 N과 V는 각각 뷰와 법선 벡터를 나타내며, γ는 막 표면 입사각이다. 이 값을 스넬의 법칙에 적용하기 위해 코사인(Cosine)을 사인(Sine)을 변환한 뒤, 수식 2에 대입하여 내부 막에서의 굴절각을 계산한다 (수식 5 참조).

$$\nu_{ouside} = \sqrt{1-\gamma^2} \\ \nu_{inside} = \frac{n_1}{n_2}\nu_{outside} \\ \gamma_{inside} = \sqrt{1-\nu_{inside}^2} \tag{5}$$

여기서 γ는 코사인 값이며, ν는 사인 값이다. ν_{inside}가 1이 되었을 때, 즉 굴절각이 90도가 되었을 경우에는 굴절이 발생하지 않고 표면 전체를 반사하는 전반사가 발생한다. 그렇기 때문에 추가적인 조건을 두어 ν_{inside}가 1보다 작은 경우에만 굴절 현상이 나타나도록 한다.

일반적으로 전반사가 발생한 경우 계산되는 내부 막의 벡터를 구하는 방법은 수식6과 같으며, 입사벡터인 $\gamma_{outside}$와 투영 벡터의 합으로 계산한다.

반적으로 전반사가 발생했을 때 계산되는 내부 막의 벡터를 계산하는 방법과 수식 6과 같지만, 본 논문에서는 입자 벡터인 $\gamma_{outside}$와 투영 벡터의 합으로 계산한다 (수식 6 참조).

$$Refrac_{inside} = \gamma_{outside} + 2N(-\gamma_{outside}N) \tag{6}$$

$$Refrac_{inside} = \left(\frac{n_1}{n_2}\gamma_{outside} + \left(\frac{n_1}{n_2}\gamma_{outside} - \gamma_{inside}\right)N\right)\mu_{bottom} \tag{7}$$

굴절이 진행되는 경우에는 수식 7을 이용하여 굴절 벡터를 계산한다. 위 수식에서 $\frac{n_1}{n_2}\gamma_{outside}$는 현재 매질에서의 입사 벡터로, 외부에서 내부로 들어오는 빛에 대한 굴절을 나타낸다. $\left(\frac{n_1}{n_2}\gamma_{outside}\right)N$는 막 내부의 굴절을 표현하며 굴절 벡터의 방향을 결정하는 항이다. 굴절은 두 요소의 합으로 계산되며 이 항에 μ_{bottom}를 곱하여 두께에 따라 변화하도록 수정했다.

마지막으로 비눗방울 막 표면의 반사 벡터를 계산한 후 내부 벡터와 결합한다 (수식 8 참조).

$$Reflect_{outside} = (\gamma_{outside} + 2N(-\gamma_{outside}N))\mu_{top} \\ Final\,Vector = Reflect_{outside} + Refrac_{inside} \tag{8}$$

외부 막의 반사 벡터는 전반사 계산 과정에 두께 변수를 곱한 것과 동일하다. 최종적으로 반사 벡터와 굴절 벡터를 더하여 비눗방울의 모습을 시각화한다.

1.3 Texture synthesis

비눗방울 막의 표면은 물결무늬를 띄고 있는데, 이러한 물결무늬 효과를 효율적으로 표현하기 위해 본 논문에서는 텍스처 흐름(Texture flow)을 이용하여 시간에 따라 흐르도록 하였다 (수식 9 참조).

$$uv_{offset} = uv_{texture} + float2(\sin(\beta_{speed}\triangle t), \cos(\beta_{speed}\triangle t)) \\ Color = tex2D(texture, uv_{offset}) \tag{9}$$

여기서 $uv_{texture}$는 입력으로 사용하는 텍스처의 기본 uv이며, β_{speed}는 텍스처 흐름의 이동 속도를 제어하는 상수이다. 이 값이 증가할수록 텍스처가 움직이는 주기가 줄어든다. 수식 9를 통해 계산된 색상을 투명도에 적용하여 투명한 부분을 강조했으며, 외곽선을 강조하기 위해 역광인 림 라이트(Rim light)을 추가했다 (수식 10 참조).

$$rim = saturate(pow(1-\gamma_{outside}, rim_{power}) \tag{10}$$

버블의 가장자리는 rim과 pow함수를 사용하여 시점과 표면이 수직일 때 가장 밝은 값을 가지도록 계산했다. 위 수식에서 rim_{power}는 역광의 강도를 결정하는 지수이며, 값이 증가할수록 테두리 쪽 반사가 집중된다.

1.4 Surface deformation of soap bubbles

비눗방울의 운동은 크게 2가지로 분류가능하다. 첫 번째로 비눗방울 자체적으로 가지는 꾸물거리는 움직임이다. 이 특징은 정점 쉐이더

(Vertex shader)에서 위치를 마세하게 조정함으로써 효율적으로 처리했다 (수식 11 참조).

$$\begin{aligned} v.vertex.x &+= \sin(\delta_{speed}\triangle t + v.vertex.x) \\ v.vertex.y &+= \cos(\delta_{speed}\triangle t + v.vertex.y) \\ v.vertex.z &+= \sin(\delta_{speed}\triangle t + v.vertex.z) \end{aligned} \quad (11)$$

위 수식에서 δ_{speed}는 사인과 코사인함수의 주기를 의미하며, 값이 클수록 움직이는 속도가 빨라진다. 두 번째로는 비눗방울의 위치와 관련된 움직임이다. 이는 사용자가 목적지를 지정하고, 현재 위치부터 해당 위치까지 선형 보간을 통해 이동하도록 설정함으로써 쉽게 적용했다.

2. Background

2.1 Camera blending

카메라 블랜드를 통해 여러 텍스처의 조합으로 배경을 나타낼 수 있다. 본 논문에서는 유니티3D에서 제공하는 unlit 쉐이더로 각 텍스처의 강도를 조절하고 lerp함수를 통해 텍스처 혼합이 계산되도록 설계했다 (수식 12 참조).

$$\begin{aligned} blendTex &= tex2D(texture, uv) \\ renderTex &= lerp(renderTex, blendMult, \epsilon) \end{aligned} \quad (12)$$

여기서 $renderTex$는 현재 화면을 나타내고 있는 텍스처이며, $blendMult$는 텍스처간의 수식이고, $opacity$는 카메라 화면에 적용할 정도를 조절하는 상수이다.

2.2 Surface wave

비눗방울과 배경을 효율적으로 합성하기 위해 쿼드(Quad)를 생성하고, 일렁이는 무늬를 표현한다. 법선 텍스처(Normal texture) 2개와 알파 텍스처를 추가하여 시간에 따라 노말 텍스처를 이동시켰다. 이를 통해 물속에서 햇빛이 비추는 듯한 텍스처 흐름을 표현할 수 있도록 했다.

III. Results

본 논문에서는 스넬의 법칙을 활용하여 굴절벡터를 계산하고 이를 반사벡터와 결합하여 비눗방울의 최종 벡터를 추출했다. 또한 카메라 블랜드와 법선 매핑 기법으로 바닷속과 햇빛의 일렁거림을 연출했고, 이를 비눗방울 시뮬레이션과 통합했다. 모든 시뮬레이션 결과에서 사용한 상수 n_1은 1, n_2는 1.33, δ_{speed}는 2, 텍스처의 β_{speed}는 0.4, $rimPower$는 2.8로 설정했다.

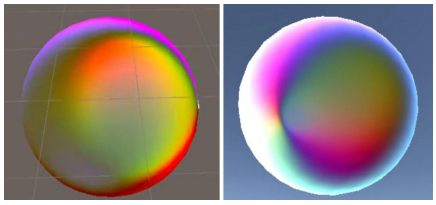

Fig. 2. Final vector of soap bubble with our method.

Fig. 2는 최종 벡터를 계산했을 때 관찰자 위치(카메라 위치)의 변화에 따라 색감이 달라지는 결과를 보여준다.

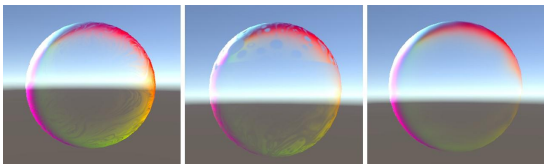

Fig. 3. Applying texture to the surface of soap bubble.

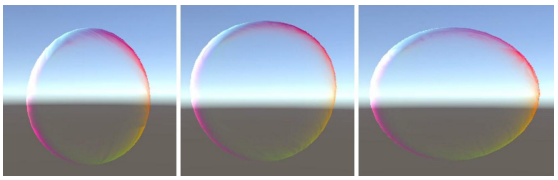

Fig. 4. Surface deformation of the soap bubble.

Fig. 3은 각기 다른 텍스처를 매핑했을 때의 비눗방울 모습을 보여주는 결과이며, Fig. 4는 비눗방울의 표면을 시간에 따라 변형시킨 결과이다. Fig. 5는 카메라 블랜드 모드와 뒷배경을 합성한 비눗방울 결과이다. 이 시뮬레이션에서는 비눗방울의 알베도(Albedo)와 에미션(Emission)의 값을 3배 이상 증가시켜 배경 속에서도 비눗방울 표면이 명확하게 보이도록 했다.

Fig. 5. Representation of soap bubbles with background synthesis.

Ⅳ. Conclusions

본 논문에서는 스넬의 법칙을 활용하여 비눗방울의 반사과 굴절 현상을 게임엔진 내 쉐이더를 통해 효율적이고 실시간으로 표현할 수 있는 프레임워크를 제시했다. 상대적으로 계산양이 큰 유체 방정식을 적용하지 않았기 때문에 버블 표면에서 발생하는 공기 저항이나 표면장력은 표현되지 않았지만, 빛과의 상호작용을 실시간으로 시뮬레이션 했다는 점에서 가상현실, 게임, 메타버스 등 다양한 물리 기반 가상환경 구축에 활용될 수 있을 거라 기대한다.

REFERENCES

[1] Huang, Weizhen, Julian Iseringhausen, Tom Kneiphof, Ziyin Qu, Chenfanfu Jiang, and Matthias B. Hullin. "Chemomechanical simulation of soap film flow on spherical bubbles." ACM Transactions on Graphics (TOG) 39, no. 4 (2020): 41-1.

[2] Bryant, F. "Snell's law of refraction." Physics Bulletin 9, no. 12 (1958): 317.