KIPS Tr. Software and Data Eng.
Vol.6, No.12 pp.581~586 pISSN: 2287-5905

https://doi.org/10.3745/KTSDE.2017.6.12.581

Real-Time Water Surface Simulation on GPU

Mankyu Sung† · DeokHo Kwon†† · JaeSung Lee††

ABSTRACT

This paper proposes a GPU based water surface animation and rendering technique for interactive applications such as games. On the water surface, a lot of physical phenomenon occurs including reflection and refraction depending on the viewing direction. When we represent the water surface, not only showing them in real time, but also make them adjusted automatically. In our implementation, we are able to capture the reflection and refraction through render-to-texture technique and then modify the texture coordinates for applying separate DU/DV map. Also, we make the amount of ratio between reflection and refraction change automatically based on Fresnel formula. All proposed method are implemented using OpenGL 3D graphics API.

Keywords : Water Surface Simulation, GPU Shader

GPU기반 실시간 물 표면 시뮬레이션

성 만 규† · 권 덕 호†† · 이 재 성††

요 약

본 연구는 게임을 비롯한 많은 콘텐츠에서 활용하기 위한 GPU기반 사실적 물 애니메이션 기법을 제안한다. 물 표면은 반사 및 굴절과 같은 물리적 현상이 일어나며, 시점에 따른 반사와 굴절의 정도가 자동적으로 조절되어야 한다. 본 논문에서는 GPU 프레임 버퍼를 이용한 렌더투텍스처 방법(RenderToTexture)을 이용하여 반사 및 굴절결과를 텍스처로 저장하며, 이 저장된 데이터에 대한 텍스처 좌표 값을 변형함으로서, 자연스러운 물결의 모습을 표현한다. 또한 표면에 나타나는 굴절 및 반사의 정도가 프레넬(Fresnel) 공식을 통해 시점과 물 표면과의 각도에 따라 자동적으로 계산되도록 한다. 제안된 알고리즘은 윈도우 기반위에서 OpenGL API와 GLSL쉐이더 언어를 통해 구현되었다. 구현 결과, 물결의 자연스러운 움직임이 확인 되었으며, 30 프레임 이상의 속도로 렌더링 되었음이 확인 되었다.

키워드 : 물표면 시뮬레이션, 쉐이더

1. 서 론

최근 GPU성능의 발전에 따라 실시간 처리가 가장 중요한 게임에서도 자연스럽고 사실적인 3차원 공간에 대한 렌더링이 가능하였다. 하지만, 여전이 물이나 불, 혹은 옷감과 같은 복잡한 객체에 대한 표현은 수작업을 통한 표현이 불가능하며, 수작업을 통해 표현이 가능하더라도 자연스러운 움직임을 표현하는 데에는 한계가 있다. 본 논문에서는 GPU 쉐이더를 이용하여 자연스러운 물 표면을 표현하는 기법을 구현한다[1]. 본 연구에서는 GPU기반의 쉐이더를 이용하였으며, 렌더투텍스처를(RenderToTexture) 이용하여 굴절과 반사를 각각 하나의 텍스처로 저장한 후, 이를 블랜딩하여 물 표면을 나타내었다. 또한, 시간데이타를 이용하여 텍스처 좌표값을 변형함으로서, 물결 표현을 하였으며, 반사광(specular light)를 이용하여 표면위의 광원의 모습을 렌더링 하였다. 또한, 시점의 각도에 따라, 굴절과 반사의 모습이 변화하도록 구현되었다. 제안한 방법은 기존의 일반적인 유체시뮬레이션을 위한 나비어-스톡스 방정식(Navier-Stokes)에 기반한 알고리즘[1, 2, 3] 혹은 위치기반 시뮬레이션[5]보다 구현이 간단하며, 실시간 애니메이션 및 구현이 가능하도록 하였다. 제안한 방법은 3D 게임 등 실시간 애니메이션 및 렌더링이 필요한 콘텐츠에 쉽게 적용된다.

※ 본 연구는 2016년도 계명대학교 비사 일반 연구기금으로 이루어졌음.
† 비 회 원 : 계명대학교 게임모바일공학전공 조교수
†† 준 회 원 : 계명대학교 게임모바일공학전공 학사과정
Manuscript Received : July 26, 2017
Accepted : September 12, 2017
* Corresponding Author : Mankyu Sung(mksung@kmu.ac.kr)

2. 관련 연구

최근 컴퓨터 애니메이션 기법들이 많은 발전을 이루어 왔으나, 유체 및 기체, 옷감의 표현들은 여전히 가장 구현하기 어려운 객체들로 인식되어오고 있다. 이에 대한 가장 오래된 연구인 [1]에서는 나비어 스톡스 방정식을 2차원 얕은 물에 적용하여 표현하였다. 이 방법은 나비어 스톡스 방정식을 단순화 하여 하나의 height 필드를 구축하여 이를 이용하여 물 표면을 표현하였다. 또한 이를 확장하여 외부 개체와 유체와의 상호작용이 가능한 수학적 모델 또한 제안되었다[2]. 기술발전을 통해 3차원 유체의 움직임 또한 개발되었는데 나비어 스톡스 방정식의 이산화를 통해 구현되었으며, 이 연구를 통해 좀 더 자연스러운 유체와 강체와의 상호작용 및 유체에 대한 제어가 가능하였다[3]. 위 연구의 가장 큰 단점은 실시간 애니메이션이 불가능 하다는 점으로 게임과 같은 콘텐츠에 직접 적용하기에는 어려움이 있었다. 최근에는 SPH(Smoothed Particle Hydrodynamics)에 기반한 유체 시뮬레이션 기법들이 제안되었다. 이 방법을 통해 스플래싱(Splashing), 버블(Bubbling) 등과 같은 유체 효과들이 실시간으로 표현될 수 있었다[4]. 이 방법은 기본적으로 파티클의 밀도, 압력 값 등을 SPH 기반의 근사 보간 함수로 계산하며 상대적으로 적은 수의 파티클 갯수로도 시뮬레이션이 가능하다. 또한 미분 값이 근사 함수에 의해 미리 계산되어 있기 때문에 연산 시간에서 잇점이 있으므로, 실시간 시뮬레이션이 가능하다. 이와 더불어 Muller등은 위치기반 다이나믹(Position based dynamics) 기법을 통한 새로운 유체 움직임 생성기법을 제안하였다[5]. 이 방법은 SPH기반 방법이 밀도의 변화에 민감하여 때에 따라 비압축형(imcompressibility)유체를 표현하기 위해서는 많은 계산이 필요하다는 단점을 극복하기 위해 파티클의 위치 제약조건을 iterative한 방법을 통해 해결함으로서 큰 타임스텝을 얻을 수 있었다. 기존의 연구들이 유체의 움직임을 생성하기 위한 시뮬레이션에 초점을 맞추었다면, 본 논문에서 제안하는 방법은 물의 표면을 실시간으로 렌더링 하기 위한 기법을 제안하고 이를 GPU를 통해 구현하는데 초점을 맞추었다. 제안하는 논문의 물 표면의 움직임은 간단한 텍스처 애니메이션을 통해 구현되었으며, 물속의 모습을 위한 굴절, 주변환경의 물 표면 반사를 위한 기법, 그리고 이에 대한 자동 파라미터 설정을 제안한다.

3. 알고리즘

전체 알고리즘은 Fig. 1의 순서에 따른다. OpenGL에서 제공하는 렌더투텍스처 기법을 이용하여 반사 텍스처와 굴절 텍스처를 각각 생성하며, 이를 입력된 시간데이타를 이용하여 텍스처 좌표를 변형한다. 후에 카메라의 위치를 고려하여 프레넬 공식을 이용한 반사 텍스처의 가중치와 굴절 텍스처의 가중치를 계산하며, 이를 블랜딩하여 물 표면을 나타낸다. 이 후에 반사광 색상을 추가함으로서 최종 렌더링 결과를 얻는다. Fig. 1은 전체적인 알고리즘 순서를 나타낸다.

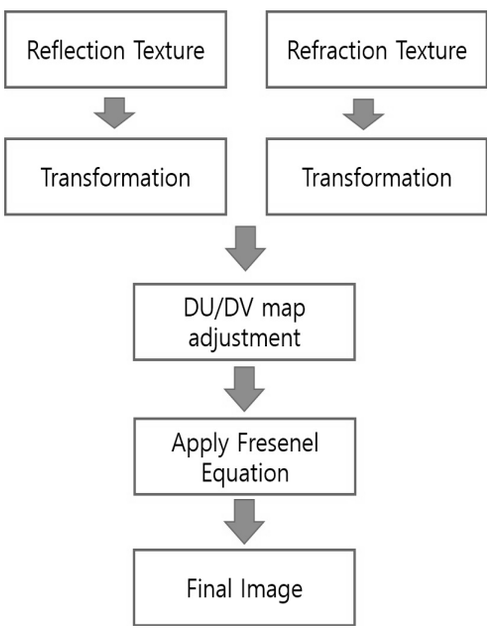


Fig. 1. Algorithm Overview

3.1 프레임 버퍼 오브젝트

반사 및 굴절텍스처는 OpenGL의 프레임버퍼오브젝트(FrameBufferObject)를 이용하여 구현된다[6]. 프레임 버퍼 오브젝트는 화면상의 나타내는 컬러정보와 뎁스 정보를 동시에 하나의 텍스처 형태로 저장 가능하도록 해주며, 실제로 모니터에 나타나지 않고도 GPU상의 메모리 버퍼에 저장할 수 있다. Fig. 2는 컬러와 뎁스 텍스처를 가진 프레임버퍼오브젝트를 나타낸다.

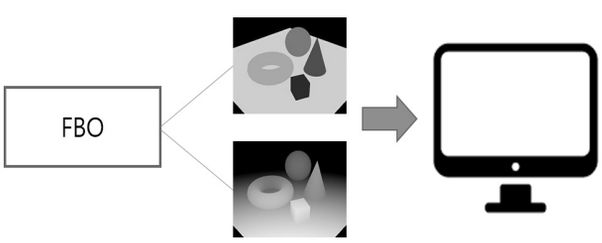


Fig. 2. Frame Buffer Object

3.2 절단 플레인을 이용한 굴절 및 반사 텍스처

반사와 굴절되는 이미지를 얻기 위해서는 렌더링을 수행할 때 물 표면을 기준으로 아래 부분과 윗 부분을 구분하여 따로 따로 랜더링 해야 한다. 즉, 반사이미지를 얻기 위해는 물 표면 윗 부분만을 렌더링하고, 굴절 이미지를 얻기 위해서는 물 표면 아래 부분만을 랜더링 하여 프레임 버퍼오브젝트에 저장한다. 가장 대표적인 3D그래픽스 API인 OpenGL에서는 gl_ClipDistanc라는 쉐이더 내장 변수를 제공해 준다[6]. 본 연구에서는 이를 이용한 평면의 방정식을 통해 반사와 굴절을 구분한다.

$$ax + by + cz + d = 0 \tag{1}$$

이 방정식에서 (a, b, c)는 평면의 normal이며, d는 원점까지의 거리를 나타낸다. 대 부분 평면은 horizontal하기 때문에 (a, b ,c) = (0,1,0)이며 d는 물 표면의 높이를 의미한다. 이 관계를 이용하면 gl_ClipDistace값은 쉽게 물 표면을 나타내는 사각형 각 정점의 월드 좌표 p_{world}와 평 방정식의 4개의 coefficient인 p=(a,b,c,d)와의 내적을 통해 계산된다. ($p, v_{world} \in R^3$)

$$\text{gl\_ClipDistace} = p \bullet v_{world} \tag{2}$$

반사와 굴절은 바로 p값을 조절함으로써 쉽게 설정할 수 있다. 이 때 반사 텍스처를 위해서는 카메라의 위치를 물 속 안으로 바꾸어야 한다. 카메라 위치를 C라고 하고 물표면의 y축값을 d라고 한다면 C의 높이는 다음과 같이 계산된다.

$$\begin{aligned} \triangle &= 2(C_y - d) \\ C_y &= C_y - \triangle \end{aligned} \tag{3}$$

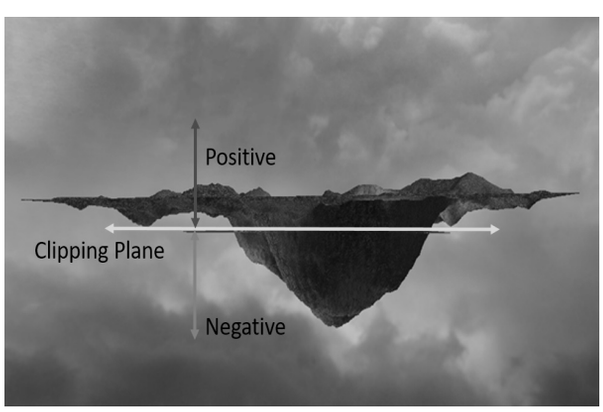


Fig. 3. Clipping Plane

Fig. 3은 ClipDistance의 값에 따른 공간에 대한 분할을 의미한다. ClipDistance가 0보다 클 경우에는 반사를 위한 렌더링, 아닐 경우에는 굴절을 위한 렌더링을 수행할 수 있다.

3.3 텍스처 변형

윗 단계에서 얻은 굴절과 반사텍스처는 텍스처 좌표에 대한 쉬프팅을 통해 변형한다. 이 변형단계는 물 표면의 어른거림을 표현하기 위함이다. 본 연구에서는 DU/DV맵을 이용하여 수정하였다. DU/DV map는 텍스처 좌표를 바꾸기 위한 레퍼런스 테이블을 의미하며 원래의 텍스처 좌표에 대한 옵셋으로 사용된다.. 반사와 굴절의 텍스처 좌표를 각각 $UV_{reflection}$과 $UV_{refraction}$이라고 하고,

DU/DV map의 값을 DV라고 한다면, 두 텍스처 좌표는 아래와 같이 변형된다.

$$\begin{aligned} UV_{reflection} &+= DV \\ UV_{refraction} &+= DV \end{aligned} \tag{4}$$

Fig. 4는 물 표면 변형을 위한 DU/DV 맵을 나타낸다.

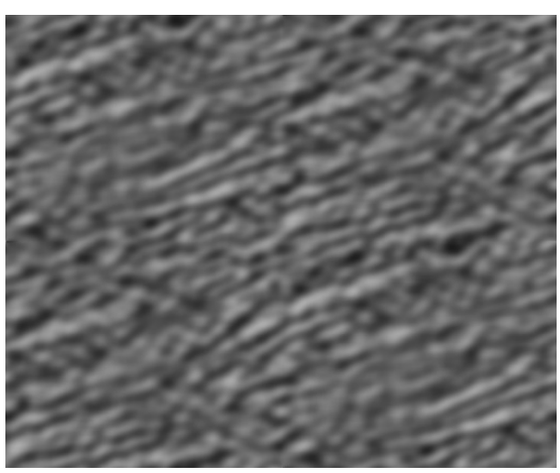


Fig. 4. DU/DV Map

3.4 물결 애니메이션

DU/DV map을 이용해 변형된 텍스처 좌표값을 시간축으로 계속 반복하면 물결의 효과를 줄 수 있다. Equation (5)는 이를 나타내며, 이 수식에서 t는 시간 축에 따라 (0,1)사이를 반복한다.

$$\begin{aligned} UV_{reflection} &+= DV*t \\ UV_{refraction} &+= DV*t \end{aligned} \tag{5}$$

시간 축 데이터는 어플리케이션으로부터 얻은 후에, GPU 쉐이더에 지속적으로 보내며, 이 보내는 시간 간격을 조절함으로서, 물결의 흐름속도를 제어할 수 있다.

3.5 프레넬(Fresnel) 계산

프레넬 값은 현재 카메라의 시점 방향을 이용하여 물 표면에 얼마나 반사와 굴절이미지가 중요한지에 대한 가중치를 구하기 위해 필요하다. 이를 위해 물 표면의 normal 벡터(N)와 시점 벡터(V)에 대한 각도 값을 이용한다. 이를 위해 굴절의 정도를 나타내는 파라미터인 $r_{refract}$은 아래와 같이 계산된다.

$$r_{refract} = \theta^r \tag{6}$$

이 수식에서 r는 반사의 정도를 강조하기 위한 상수 값이며 보통 0.5를 설정한다.

3.6 물 표면 색상 계산

반사 텍스처와 굴절 텍스처를 통해 얻은 색상을 각각 $c_{reflect}$와 $c_{refract}$라고 한다면, 이 색상에 반사광의 색상과 물 고유의 색상을 블랜딩하여 최종 픽셀의 색상을 결정한다. 물 고유의 색상은 보통 물을 나타내는 푸른색을 사용하며, 물의 깊이에 따른 따라 색상의 변화까지 고려한다. 반사광은 보통 퐁 광원모델의 반사광을 사용한다. 반사광을 c_h라고 하고, 해당 픽셀의 물 깊이를 d, 물 고유의 색상을 c_o라고 한다면, 최종 색상 f는 아래와 같이 계산된다.

$$f = mix(c_o, mix(c_{reflect}, c_{refract}, r_{refract}), e^{-d}) + c_h \tag{7}$$

이 수식에서 mix함수는 가중치 w를 이용한 두 색상 A, B의 선형 보간하는 함수를 의미한다.

4. 실 험

위에서 설명한 알고리즘을 Visual Studio 2013을 이용하여 Windows 7위에서 구현하였다. 윈도우 생성을 위해 FLTK UI 라이브러리를 이용하였으며 렌더링을 위하여 OpenGL를 3D 그래픽스 API를 이용하였다. 지형 모델 데이터는 무료의 3D OBJ포맷의 데이터를 이용하였으며 쉐이더 구현은 GLSL를 이용하였다.

Fig. 5는 구현 후, 실험을 통해 얻은 반사 텍스처를 나타내며 Fig. 6은 굴절 텍스처를, Fig. 7는 이 두 개의 텍스처를 혼합한 최종 렌더링 된 모습을 나타낸다.

Fig. 5. Reflection Texture

Fig. 6. Refraction Texture

Fig. 7. Final Rendered Water Surface

그림에서 나타나 있듯이, 물 표면은 자연스럽게 반사와 굴절의 모습을 나타냈으며, 시간에 따른 자연스러운 물결 흐름을 나타내었다. 성능 면에서도 모든 렌더링은 30프레임 이상의 실시간 속도를 나타내었다. 또한, 시점 방향의 변화에 따라 반사와 굴절의 정도가 바뀜을 확인하였다. GPU 쉐이더를 이용할 경우, RenderToTexture를 위하여 프레임 버퍼를 개별적으로 생성 및 조작이 가능하므로, 제안한 방법에 대한 구현이 가능했으나, 전통적인 CPU기반의 경우, 프

레임 버퍼에 대한 조작이 불가능하므로, 제안한 방법을 구현 가능하지 않았으며, 이에 따른 성능비교를 수행할 수 없었다. 하지만, 논문 [7]에 따르면 일반적인 컴퓨팅 알고리즘의 경우 GPU구현이 12-13배 빠른 것으로 알려져 있다.

5. 결 론

본 논문은 실시간으로 물 표면을 애니메이션하고 렌더링하기 위한 기법을 제안하였고, 이를 OpenGL를 쉐이더 언어(GLSL)를 통해 구현하였다. 본 논문에서 제안한 애니메이션 방법은 단순히 텍스처의 좌표를 이동함으로서 물결을 표현하였으므로, 강체와의 자연스러운 상호작용은 되지 않는다. SPH 혹은 위치기반 다이나믹스와 같은 사실적인 유체 애니메이션기법을 본 논문에서 제안한 렌더링 기법과 결합한다면 좀 더 효과적인 물 표면이 되리라고 생각된다.

References

[1] M. Kass and G. Miller, “Rapid, Stable Fluid Dynamics for Computer Graphics,” *The Proceeding of SIGGRAPH ’90 in Computer Graphics*, Vol.24, No.4, pp.49-57, 1990.

[2] J. F. O’Brien and J. K. Hodgins, “Dynamic simulation of Splashing Fluids,” *Computer Animation ’95, Proceedings*, IEEE, pp.198-205, 1995.

[3] N. Foster and R. Fedkiw, “Practical Animation of Liquids,” *Proceeding of SIGGRAPH ’01*, pp.23-30, 2001.

[4] T. Weaver and Z. Xiao, “Fluid Simulation by the Smoothed Particle Hydrodynamics Method: A Survey,” *GRAPP 2016 Proceedings of the 11th Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications Volume 1: GRAPP*, pp.215-225, 2016.

[5] M. Macklin and M. Muller, “Position Based Fluids,” *ACM TOG*, Vol.32. No.4, 2013.

[6] D. Wolf, “OpenGL 4.0 Shading Lanuage Cookbook,” PACT publishing, 2011.

[7] J. Krüger and R. Westermann, “Linear Algebra Operators for GPU Implementation of Numerical Algorithms,” *ACM Transactions on Graphics (TOG)*, Vol.22, Issue 3, pp.908-916, 2003.

성 만 규

http://orcid.org/0000-0001-5807-6719

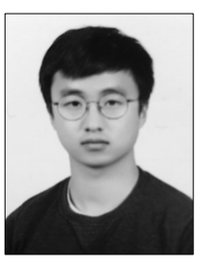
e-mail : mksung@kmu.ac.kr

1993년 충남대학교 전산학과(학사)

2005년 (미)위스콘신대학-메디슨 Computer Sciences(석・박사)

2012년～현 재 계명대학교 게임모바일공학전공 조교수

관심분야 : 컴퓨터그래픽스, 컴퓨터 애니메이션, HCI



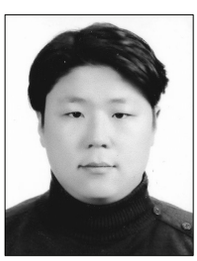
권 덕 호

http://orcid.org/0000-0001-7539-8883

e-mail : kon9383@naver.com

2012년～현 재 계명대학교 게임모바일 공학전공 학사과정

관심분야 : 컴퓨터그래픽스, 컴퓨터게임



이 재 성

http://orcid.org/0000-0001-9001-1691

e-mail : dau9999@kmu.ac.kr

2012년～현 재 계명대학교 게임모바일 공학전공 학사과정

관심분야 : 컴퓨터그래픽스, 컴퓨터게임