선형 컨벌루션과 경계구를 이용한 물표면과 객체의 실시간 상호작용 생성

Interaction between Water Surface and 3D Object by using Linear Convolution and Bounding Sphere

강경헌, 이현철, 허기택, 김은석
동신대학교 디지털콘텐츠학과

Gyeong-Heon Kang(vagabond@dsu.ac.kr), Hyeon-Cheol Lee(hclee@dsu.ac.kr),
Gi-Taek Hur(gthur@dsu.ac.kr), Eun Seok Kim(eskim@dsu.ac.kr)

요약

컴퓨터 그래픽스에서 물을 애니메이션하거나 다양한 특수효과를 표현하기위해 유체역학의 기술들이 사용되고 있다. 하드웨어성능이 높아지면서 이전에 불가능했던 알고리즘들이 실시간으로 가능해지고 있기는 하지만, 정밀한 표현에는 여전히 많은 시간이 소요되며, 성능과 사실성 사이의 균형을 위한 다양한 기술들이 연구되고 있다. 특히 게임과 같은 문맥을 가지는 곳에서 사용자의 요구에 의해 바다나 호수 같은 넓은 지역의 물표면과 객체의 상호작용을 표현하기 위해서는, 물리적 사실성을 어느 정도 희생하더라도 시각적인 사실성을 유지하는 범위에서 실행 성능을 높이는 것이 우선시 된다. 본 논문에서는 물표면과 객체의 상호작용에 의한 다양한 물표면의 형태변화를 선형 컨벌루션 기법과 경계구를 이용하여 실시간으로 자연스럽게 애니메이션하는 방법을 제안한다.

■ 중심어 : | 유체 애니메이션 | 선형 컨벌루션 | 물표현 | 경계구 |

Abstract

In Computer Graphics, fluid dynamics is used for animating and expressing the various special effects of water. As the hardware performance is getting higher, the several algorithms for fluid dynamics become to be executed in real time. However, it still requires a lot of computational time to get the realistic and detailed results. Therefore, there are many researches on the techniques of balancing between performance and quality. It must give priority to the executive performance preserving the visual reality even though sacrificing the physical reality, specially in applications with the game context which need to express the interaction between 3D objects and the surface of the water such as the sea or a lake. In this paper, we propose a method for the realtime animation of interactions between 3D objects and the surface of the water using the linear convolution of height fields and the bounding spheres of object.

■ keyword : | Fluid Animation | Linear Convolution | Water Expression | Bounding Sphere |

Ⅰ. 서 론

하드웨어의 발전과 함께 컴퓨터 그래픽스에서 유체를 애니메이션하기위해 이전에 불가능했던 시뮬레이션

* 본 연구는 문화관광부 및 한국문화콘텐츠진흥원의 지역문화산업연구센터(CRC) 지원사업의 연구 결과로 수행되었습니다..
접수번호 : #080121-001
접수일자 : 2008년 01월 21일
심사완료일 : 2008년 04월 15일
교신저자 : 강경헌, e-mail : vagabond@dsu.ac.kr

알고리즘들이 수행 가능해지고 있으며, 물, 불, 폭발, 연기, 안개 등의 유체 표현 기술이 성공적으로 적용되어지고 있다. 물은 물리적 처리에 있어 타고난 복잡함을 포함하고 있기 때문에 물리적 법칙을 기반으로 다양한 효과를 실시간 시뮬레이션 하기 매우 어렵다. 결국 게임과 같은 문맥을 가지는 환경에서 물과 상호작용하는 이펙트를 구현할 때는 품질과 성능의 교환이 불가피하며, 호수나 바다와 같은 넓은 지역의 움직임을 실시간으로 시뮬레이션 하기 위해서는 물이 가지는 많은 물리적 속성들을 실시간이 가능할 정도로 배제시키며, 시각적으로 사실성을 유지하면서 근사치를 계산하여 표현하여야 한다. 물과 객체와의 상호작용은 물리적으로 정확한 규칙을 사용하는 비실시간 유체 시뮬레이션으로 표현되어 왔으며, 정적인 대규모 액체와 객체와의 실시간 시뮬레이션에 대한 연구는 많이 이루어지지 않고 있다[18].

본 논문에서는 물표면와 객체의 상호작용을 실시간 시뮬레이션하기 위해 선형컨벌루션과 객체의 경계구를 이용한 물표면 변형 방법을 제안하고자 한다.

II. 관련 연구

주변 환경과 상호작용하는 물의 움직임을 표현하기 위해 많은 연구들이 파티클 및 볼륨모델을 이용하여 유체와 객체의 충돌 및 물방울생성 등을 시뮬레이션 하였다.

1996년 Foster와 Metaxas는 유체 전체 환경을 사각 그리드로 나누어 유체 영역과 장애물이 포함된 영역을 구분하고, 유체의 움직임을 설명하는 Navier Stokes방정식을 유한차분(finite difference) 방법을 이용하여 장애요소와 상호작용하는 충돌하는 물의 움직임을 시뮬레이션 하였다[1]. 2001년 Foster와 Fedkiw는 1999년 유체를 파티클로 모델링하고 파티클의 움직임을 역추적 하여 이동 속도를 계산하는 semi-Lagrangian 방식을 사용하여 기체를 표현하였던 Stam의 연구결과를 기반으로[2], 비관성(inertialess) 파티클과 Level Set을 이용하여 물을 표현하고, 객체의 탄젠트를 따라 유체가 자유롭게 움직이도록 함으로써 움직이는 객체와의 상호작용을 고려하고 애니메이션에서의 활용 가능성을 보여주었다[3]. 2003년 Premoze는 유체 표현에 사용되는 질량, 에너지, 운동량 보존의 식을 파티클 사이의 상호작용 식으로 표현하는 방법을 사용해 유체를 시뮬레이션 하였으며, 서로 다른 유체간의 혼합이나 물체와 객체의 상호작용, 그리고 물의 경계면 깨어짐 현상 등을 표현하였다[4].

그러나 유체를 파티클 또는 격자로 표현되는 볼륨으로 표현하는 경우 계산상의 비용이 크고 넓은 지역을 표현하기에는 더욱 많은 시간이 소요되게 되므로 실시간 시뮬레이션에 적용하기 어렵다는 단점이 있다. 그리하여 넓은 지역을 실시간으로 표현하기위해, 주로 사인파와 같은 정현파의 합성 등을 통해, 물 내부의 모습을 제외하고 물표면 만을 표현하는 기법이 사용하고 있다.

바다나 호수와 같은 수면에서, 고정된 위치에서의 움직임은 있지만 전체적인 움직임이 없는 물표면에 대한 연구가 시작된 것은 1980년대 초반에 높이 필드(height field)를 이용하여 해석학적 함수들로 파도를 표현하면서 본격적으로 시작되었다.

1986년 Fournier와 Reeves는 거스너(Gerstner)모델을 도입하여 해수면을 표현하고 처음으로 파도의 굴절을 표현하였다[5]. 1990년대에 들어서서는 Navier Stokes 식을 적용시켜 각각의 파도가 파동방정식에 의해 표현하던 거스너 모델과 달리 여러 개의 파도를 반사파, 흐름, 경계표현 등의 복잡한 상호작용을 일으킴으로서 수면의 움직임을 시뮬레이션 하였다[6][7]. 1999년 Tessendorf는 실제 바다와 같은 모습을 표현하기위해 Gerstner 모델 기반의 정현파 스펙트럼으로부터 트로코이드(trochoid) 스펙트럼을 추출하여 사용하고, 빠른 푸리에 변환(fast fourier transform : FFT)을 반복적으로 적용하여, 거친 파도를 표현하는 방법을 제안하였다[8]. 2000년 Thon등이 Airy모델과 Gerstner 모델을 혼합하여 전체적인 파도의 모습은 간단한 변위 변화로 표현하고, 세심한 표현은 시간에 따라 위상을 변화시킨 새로운 모델을 제안했었고[9], 이들의 방법을 이용하여 2002년 Hinsinger 등은 해수면을 보이는 스크린의 격자를 역으로 해수면에 투시하면서 필요한 만큼만 메시를

생성하여 시뮬레이션 성능을 개선시켰지만, 깊은 바다의 외형적인 모습만 표현하는데 그쳤으며 해안에서의 굴절 등은 고려하지 않았다[10]. 2002년 Loviscach는 컨벌루션 기반의 알고리즘과 호이겐스의 원리를 이용해 물표면에 배가 지나갔을 때, 물표면에 발생하는 파형을 빠르게 시뮬레이션 하였다[11][12]. 2005년 Mitchell은 스펙트럼 모델을 기반으로 하면서 계산과정을 GPU를 이용함으로써 빠른 속도로 실시간 해수면을 표현하였다. 전체적인 표면을 변형하기위한 높이맵과 세부적인 표현을 위한 법선맵을 푸리에(Fourier)변환을 통해 생성하였으며, 높이맵과 화면에 그려지는 메시가 1:1 대응이 되도록 함으로써 좁은 지역의 수면을 표현하였고 물체와 물표면과의 상호작용을 고려하여 물위에 객체가 지나가는 자국 같은 파형을 표현하였다[13].

바다와 같은 대규모의 물표현은 주로 파동방정식을 이용한 정현파들의 합으로 물표면을 형성하거나, 특정 스펙트럼을 추출하여 시뮬레이션 하였으며, 사실감을 더욱 증대시키기 위해 고속 푸리에 변환을 반복적으로 적용해 물표면의 움직임을 표현해 내었다. 하지만 이러한 기법들은 매 프레임마다 처음부터 끝까지 새로운 파형을 만들어내는 연속함수들의 표현이므로 물살을 가르는 배에 의해 난류를 형성하거나, 물체의 운동이 물표면의 경계를 지나가면서 변형시키고 물결을 앞뒤로 요동치게 만드는 것과 같은 물체들과 수면 사이의 상호작용을 표현하기 어려우며, 상호작용이 가능하게 하더라도 상당한 프레임률 저하를 감수해야 한다. 수면의 파동을 전파하는 컨벌루션 기법은 외부환경요소나 물체들과의 상호작용 처리가 손쉬우며 별도의 하드웨어 가속없이 1Ghz의 CPU에서 128x128 크기의 수면격자를 초당 30 fps이상으로 시뮬레이션 하는 것이 가능하다[14].

III. 선형 컨벌루션 기반의 물애니메이션

바다나 호수 등의 정적인 대규모 유체의 가장 역동적인 상황은 물체와의 상호작용에 의해 물표면에 발생하는 물결 및 파도의 변화일 것이며, 유체 시뮬레이션에서 가장 관심을 일으키는 부분이기도 하다. 사인파와 같은 주기 함수들의 계산에 의한 파동 모델은 연속함수이기 때문에, 물표면과 객체의 상호 작용에 의해 지역적으로 급격한 변형을 생성하기 어렵다는 문제점을 갖는다. 따라서 본 논문에서는 사인파와 같은 주기 함수들의 계산에 의한 파동생성이 아닌 물표면을 모델링하는 메시 정점들의 높이 값을 저장한 높이 필드에 선형 컨벌루션을 적용시킴으로써 물표면의 움직임을 생성한다.

1. 높이 필드(Height Field)

높이 필드는 연산량의 감소를 위해 완전한 3D가 아닌 2.5D로써 물표면을 표현한 모델로서, 빠른 2D 계산과 적은 리소스의 사용, 그리고 쉽게 렌더링할 수 있다는 등의 이점을 제공한다. [그림 1]은 높이 필드의 구성요소를 보여주며, 각 정점의 높이를 변경함으로써 효과적으로 물표면의 애니메이션을 생성 할 수 있다[16].

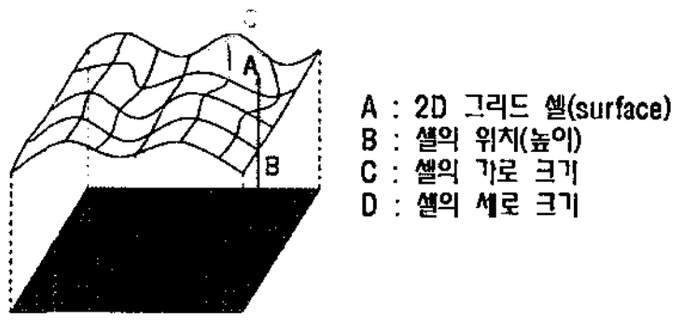


그림 1. 높이 필드구성

물결의 움직임은 시간에 따라 일정 방향으로 이동함으로써 나타낸다. 즉, 현재 물표면 메시의 높이 필드는 이전 시간의 주변의 물결 높이에 영향을 받으며, 자연스러운 물표면 움직임 생성을 위해서는 시간 t와 $t-1$에서의 높이 필드가 모두 필요하다. 따라서 본 논문의 물표면 애니메이션은 연속된 시간에 대한 두 개의 높이 필드를 사용한다.

2. 선형 컨벌루션(Linear Convolution)

컨벌루션은 일반적으로 영상처리에서 선명화, 경계선 검출 등의 영역 기반 처리를 하기위해 사용되는 기법으로, 주어진 위치의 값을 구하기 위해 이웃 값들의 가중치 평균을 이용하는 방법이다[17].

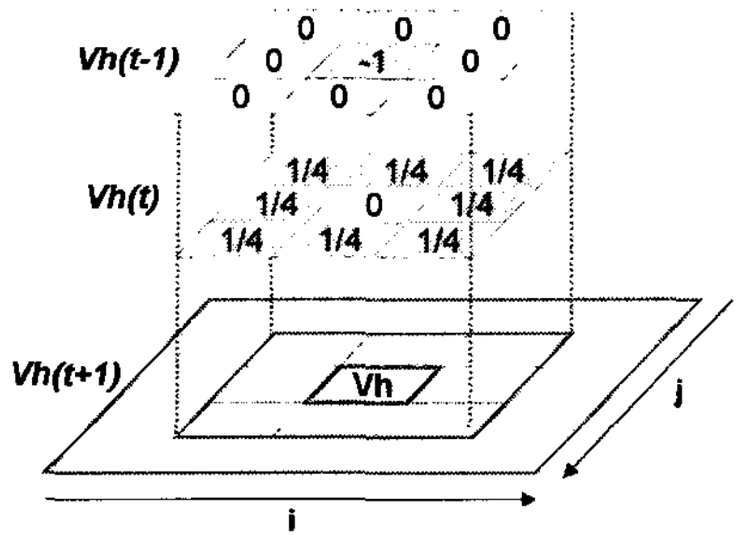


그림 2. 파동 생성을 위한 선형 컨벌루션 영역필터

본 논문에서는 연속적인 물결의 움직임(파동)을 생성하기 위해 [그림 2]와 같은 선형 컨벌루션을 이용하였다. Vh는 특정위치의 높이값으로 시간 t의 높이 필드와 $t-1$에서의 높이 필드에 영역필터를 씌워 계산된, 다음 시간 $t+1$의 높이 필드값의 일부로서 갱신되어야 할 물표면의 높이이다.

선형 컨벌루션을 이용하여 물을 애니메이션 하기위해서는 파동의 지속여부를 조절하기 위한 댐핑(damping) 인자가 필요하다. 댐핑인자는 0에서 1사이의 실수값을 가지며 생성된 파동이 얼마나 빨리 소멸되는지를 결정한다[15]. 댐핑인자가 0인 경우 파동은 다음 시간에 즉시 사라지게 되고, 1인 경우 물의 파동은 절대로 사라지지 않고 계속 유지된다. 선형 컨벌루션을 이용한 파동 애니메이션 생성하기 위한 방법은 다음과 같다. 먼저, 현재시간 t의 높이 필드에 영역필터 마스크를 적용하여 임시변수 $Buff_{(x,z)}$에 저장하고(식1), 시간 $t-1$의 높이값을 차감한 후 댐핑인자 DF를 곱해주어 시간 $t+1$의 높이값을 저장한다(식2). 모든 정점에 대하여 선형 컨벌루션 과정이 끝나면 높이 필드 값을 업데이트한다.

$$Buff_{(x,z)} = \sum_{i=1}^{-1}\sum_{j=1}^{-1}\left(Vh_{(x+i,\,z+j)}(t) \times C(i,j)\right) \quad (1)$$

$$Vh_{(x,z)}(t+1) = \left(Buff_{(x,z)} - Vh_{(x,z)}(t-1)\right) \times DF \quad (2)$$

$C(i,j)$는 파동 애니메이션을 생성하기 위한 선형 컨벌루션의 영역필터로서 다음과 같은 값을 갖는다.

$$C(i,j) = \begin{cases} 0, & \text{if } i=j=0 \\ 1/4, & otherwise \end{cases}, \quad i,j \in -1,0,1$$

표 1. 선형 컨벌루션, 높이 필드 업데이트 과정

```
// 선형 컨벌루션 과정
for(x=xmin; x<xmax; x++)
   for(y=ymin; y<ymax; y++){
          식 (1), (2)를 이용하여 Vh(x,z)(t+1)를 계산
    }
}
```

```
// 높이 필드 업데이트 과정
for(x=xmin; x<xmax; x++)
   for(y=ymin; y<ymax; y++){
         Vh(x,z)(t-1) = Vh(x,z)(t)
         Vh(x,z)(t) = Vh(x,z)(t+1)
    }
}
```

식 (1)과 식 (2)를 이용한 선형 컨벌루션 과정과 높이 필드 업데이트 과정은 [표 1]과 같다. 이 때 xmin와 zmin는 물표면을 나타내는 메쉬의 정점들의 x,z의 최소값을, xmax와 zmax는 최대값을 나타낸다.

[그림 3]은 중심지점에 높이 필드를 급격히 변화시켰을 때 시간의 변화에 따라 선형 컨벌루션에 의해 물표면이 변화하는 모습을 보여준다. 물표면의 파동 애니메이션은 선형 컨벌루션 과정을 통해 이뤄지며, 물표면에 발생하는 효과는 높이 필드를 재정의 하는 방법에 따라 다양하게 생성할 수 있다.

*번호는 시간경과에 따른 프레임 순서임

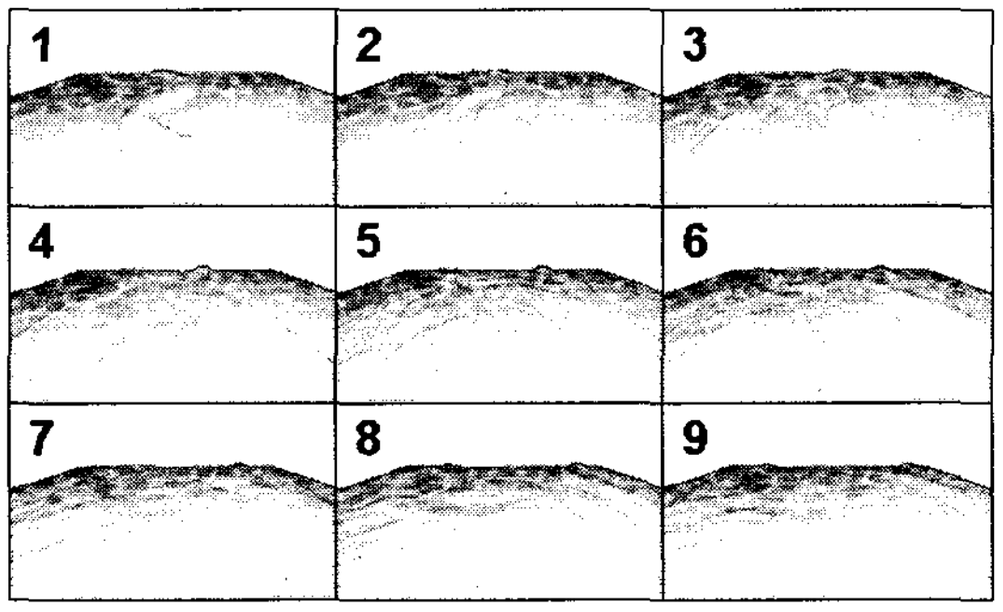


그림 3. 선형 컨벌루션을 이용한 파동 애니메이션

Ⅳ. 물표면과 객체의 상호작용

선형 컨벌루션 기반의 물표현 방법은 시간에 구애받지 않고 자유롭게 물표면을 구성하는 높이 필드를 변형할 수 있다. 변형된 물표면은 선형 컨벌루션이 진행되면서 새로운 파동을 생성하게 되기때문에, 물표면과 객체의 상호작용은 상황에 따라 물표면을 어떻게 변형하느냐에 따라 다양한 결과를 만들어내게 된다.

1. 경계구를(Bounding Sphere) 이용한 파동생성

물의 파동은 호이겐스(Huygens)의 원리를 이용하여 표현할 수 있다. 호이겐스의 원리는 물에 돌멩이를 던지면, 돌멩이가 떨어진 물표면이 원이 되어 퍼져나가는 것과 같이 파동이 퍼져 나갈 때 한점으로부터 퍼져나간다는 이론이다. 물의 파동은 호이겐스의 원리에 따라 파원으로부터 발생하는 구면파 형태의 2차 파형이므로, 높이값을 구의 형태로 변형시켜주면 파동의 움직임을 애니메이션할 수 있다.

본 논문에서는 다양한 물표면을 애니메이션하기 위해, 경계구를 이용하여 현재시간의 물표면의 높이 필드값인 $Vh_{(x,z)}(t)$를 변형시킴으로써 물표면에 다양한 파동을 생성하여 상호작용을 표현해 내었다. 경계구는 위치와 영향력의 범위에 따라 물표면을 변형시키는데, 구의 방향(SD : sphere direction), 구의 크기(SR : sphere radius), 구의 높이비율(SHR : sphere height ratio)을 정의함으로써 형태를 만들어 낼 수 있다.

[그림 4]는 경계구의 크기와 높이비율을 조절하여 만들어진 경계구 영향력의 형태이다. SD는 구의 위/아래 방향을 나타내는데, SD가 1일 경우는 구단면을 반으로 나눈 위쪽부분을, -1일 경우는 아래쪽부분만을 사용한다. SR은 구단면의 반지름의 크기를 나타내며, SHR은 SR에 비례하는 구의 높이비율을 가리킨다. 구의 높이 h가 구의 반지름 SR과 같다면 SHR은 1.0이 되고 h가 SH의 두 배라면 SHR은 2.0이 되며, 이때 경계구의 영향력은 타원의 형태가 된다.

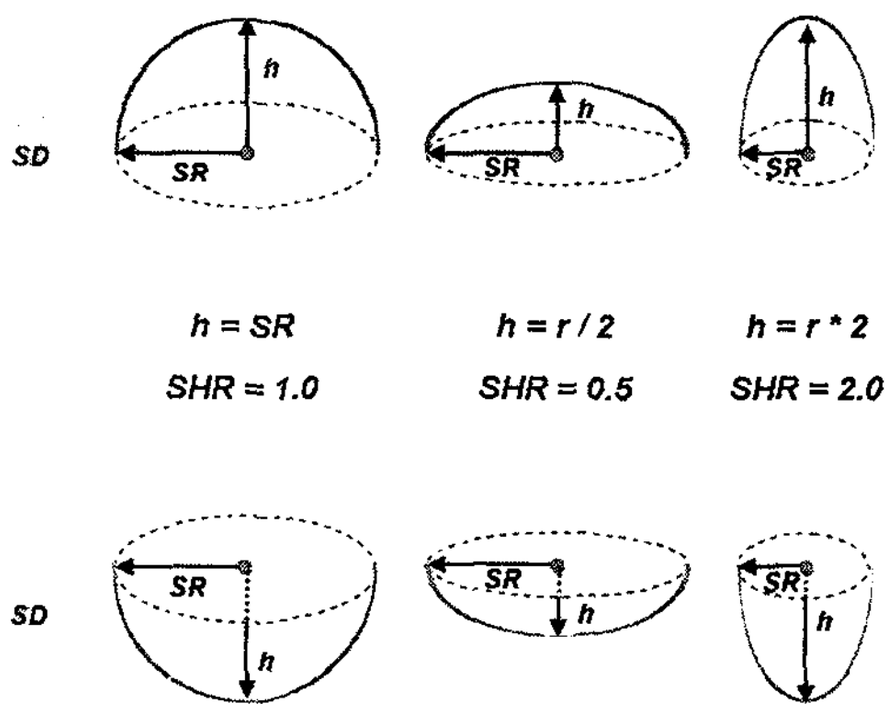


그림 4. 크기와 높이비율에 따른 경계구의 영향력

구단면의 반지름을 나타내는 SR은 높이 필드의 정점과 정점의 간격으로서, SR이 10이라면 $(x-10, z-10)$부터 $(x+10, z+10)$까지의 높이값을 변형하게 된다. [그림 5]는 (x,z)의 위치에 SR=10, SH=1.0으로 경계구를 생성하였을 때, 임의의 위치 (px, pz)의 높이 PH를 나타낸다. (x,z)에서 (px,pz)까지의 거리는 (식3)과 같이 구할 수 있으며, (px,pz)의 높이 PH는 (식4)와 같이 구한다. 이렇게 계산된 PH는 경계구가 구형일 때의 높이이며, SD와 SHR을 적용시켜 최종적으로 구할 $Vh_{(px,pz)}(t)$는 (식5)와 같다.

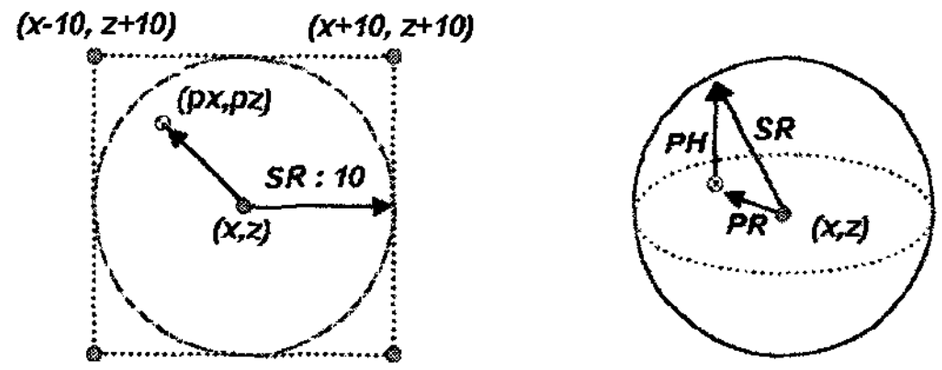


그림 5. (x,z)위치에 생성된 경계구와 (px,pz)위치의 높이 PH

$$PR = \sqrt{(x-px)^2 + (z-pz)^2} \quad (3)$$

$$PH = \sqrt{SR^2 - PR^2} \quad (4)$$

$$Vh_{(px,pz)}(t) = Vh_{(x,z)}(t) + (PH \times SD \times SHR) \quad (5)$$

[그림 6]은 물표면의 움직임이 없는 상태와 움직임이 있는 상태 두 가지의 경우에 대해, 물수면의 높이 0의 위치에 SR은 20으로 고정하고 SD, SHR을 조절하여 경계구를 생성시켰을 때 변형된 물표면을 보여준다.

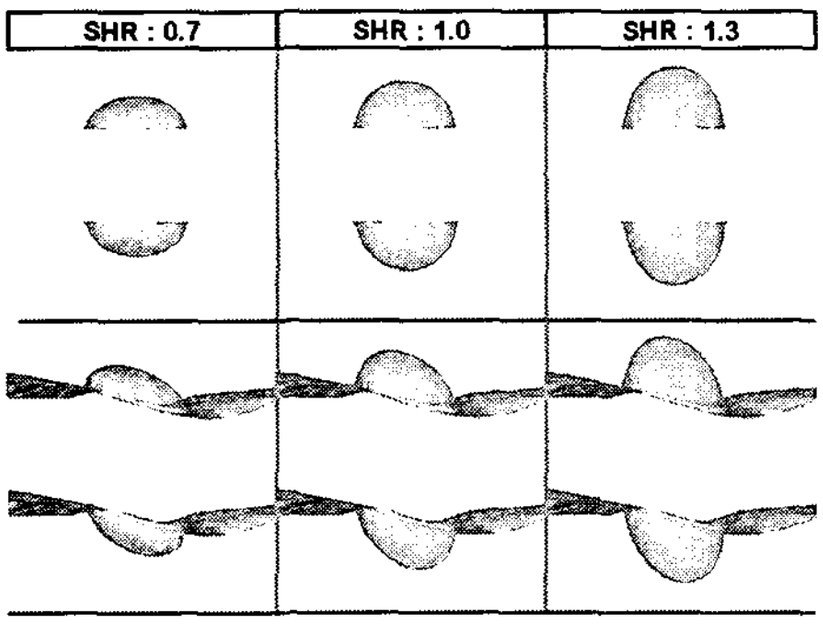


그림 6. SHR 값에 따른 물표면 변화

2. 이동하는 객체와 물표면의 상호작용

물의 파동은 호이겐스의 원리에 따라 파원으로부터 발생하는 구면파 형태의 2차 파형이므로, 높이값을 구의형태로 변형시켜주면 파동의 움직임을 애니메이션 할 수 있다. 본 논문에서는 객체와 물표면과의 상호작용을 위해 객체를 하나 또는 복수의 경계구 집합으로 정의하고 경계구의 크기와 위치에 따라 물표면의 높이 필드값 $Vh_{(x,z)}(t)$를 변형시킴으로써 상호작용을 표현한다. [그림 7]은 객체의 경계구가 물표면을 향해 움직일 때 물표면과의 충돌에 의해 물표면이 변형된 모습을 보여준다.

그림 7. 경계구에 의한 물표면 변형

2.1 가속 인자(Acceleration Factor)

동일한 크기를 가진 객체가 물표면에 충돌할 때, 물표면은 경계구가 가진 무게와 속도에 따라 변형이 달라질 것이다. 경계구의 크기와 위치는 현재 시간 t의 물표면의 높이 필드값에 영향을 준다. 그러나 파동 애니메이션은 선형 컨벌루션을 통해 t에서의 높이값과 $t-1$에서의 높이값을 조합하여 $t+1$의 높이값을 계산하므로, $t-1$에서의 높이 필드값 역시 조정이 필요하다. $t-1$의 높이는 (식1),(식2)에서와 같이 t의 높이값에 차감되어 $t+1$의 높이에 반영되므로, $t-1$의 높이를 객체의 특성에 따라 증감시킨다면 발생되는 파동의 반발력을 생성할 수 있게 된다. 본 논문에서는 이러한 특성을 고려하여 객체의 무게와 속도에 비례하는 가속인자를 정의하고 $t-1$에서의 높이필드 값을 결정하는데 이용하였다.

가속인자 AF는 사용자가 정의하는 기준객체와 대상객체의 특성에 따라 결정된다. 기준객체를 B, 대상 객체를 O, 무게를 w, 속도를 v, 크기를(SR) s라고 했을 때, AF는 (식6)과 같이 얻어낼 수 있다.

$$AF = \frac{(Ow \times Ov) \div Os}{(Bw \times Bv) \div Bs} - 1 \qquad (6)$$

AF는 $t-1$의 높이 필드값을 변형시켜 대체되어 애니메이션시 객체의 움직임에 따른 물표면의 변화를 유도할 수 있게 한다. 경계구가 물의 현재 높이 필드 $Vh_{(x,z)}(t)$를 변형시킬 때, 객체의 특성에 따라 AF를 적용하여 이전 시간의 높이 필드 $Vh_{(x,z)}(t-1)$을 변형시키는 식는 다음과 같다.

$$Vh_{(x,z)}(t-1) = Vh_{(x,z)}(t) - Vh_{(x,z)}(t) \times AF \qquad (7)$$

[그림 8]은 경계구가 물 표면에 충돌하였을 때, AF값에 따른 물표면의 변화를 보여준다. (식6)에 의해 구해진 AF는 실수 값을 가지며 0을 기준으로 양의 값을 가지면 파동의 반발력이 커지고, 음수 값을 가지면 파동 변화의 감쇠가 빨라져 원래의 높이로 빨리 복원된다. AF는 파동에 대한 반발력을 정의하는 것이고, DF는 파동의 수명을 정의하는 것으로서, AF값에 의한 감쇠는 DF와는 다르게 나타난다.

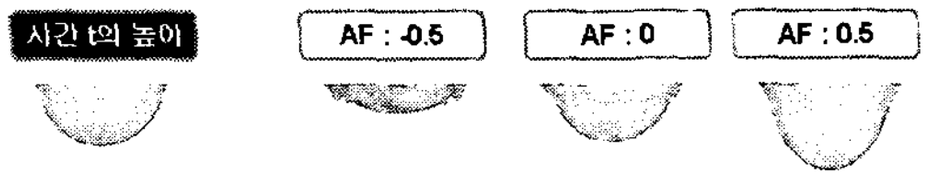


그림 8. AF값에 의해 변형된 t+1의 물표면

2.2 물표면 변경시점

[그림 8]는 경계구가 수직방향으로 떨어졌을 때를 기준으로 시뮬레이션한 결과로서, 움직임이 수직이 아닌 방향성을 가진 객체의 경우 이를 고려하여 높이 필드를 변경하여야 한다. 높이 필드를 변경하는 시점은 [그림 9]와 같이 경계구가 물표면에 닿는 순간부터 완전히 잠길 때까지로 정의할 수 있다.

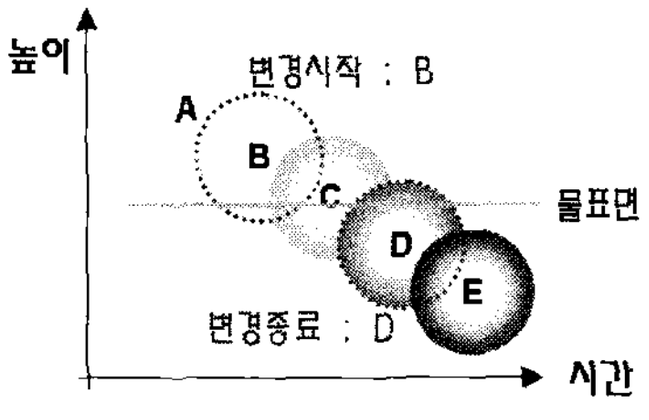


그림 9. 객체에 의한 물표면의 높이 필드 변경

경계구의 위치는 시간에 따라 변화하게 되며, [그림 9]처럼 수면과 충돌 시작시 높이 필드를 변형하기 시작한다. 즉, 변경 시작 시점 B에서의 물표면 높이를 기준으로 경계구가 완전히 잠기게 되는 시점인 D이후에 높이 필드 변형을 종료하여야 한다. 지금까지의 방법들을 이용하여 물표면과 객체의 상호작용을 고려한 물표면 변형 알고리즘은 다음과 같다.

표 2. 물표면 변형 알고리즘

```
While(Simulation)
{
    Step1. if(특정경계구가 최초로 수면에 닿았는지 체크)
               fixHeight[][] = 수면의 경계 저장
    for each class of bounding sphere
    {
      Step2. if(경계구가 fixHeight[][] 아래로 잠기면)
                 Goto Step7.
      Step3. if(fixHeight[][]와 경계구 표면이 충돌하면)
             {
                 Step4. 경계구 형태에 따라 Vh(r,:)(t) 변경
                 Step5. 정의된 무게,속도비례에 따라 AF계산
                 Step6. Vh(r,:)(t - 1) 변경
             }
      Step7. 경계구의 다음 위치 계산과 이동
    }
    Step8. Height Field에 선형 컨벌루션 적용
}
```

V. 실험 결과

본 논문에서는 제안한 방법을 이용하여 포탄 발사, 점핑 물고기, 항해하는 배를 시뮬레이션하고 성능결과를 검증하였다.

1. 포탄 발사 시뮬레이션

경계구의 기본 모양은 경계구의 높이비율 SHR이 1.0의 값을 가지는 구의 형태이다. 포탄 발사 시뮬레이션은 포탄의 형태와 경계구의 형태가 일치하므로, 움직이는 객체를 경계구로 정의하여 물표면을 변형하는 본 논문의 제안방법에 가장 적합한 예이다.

포탄으로 정의된 경계구의 SR은 5.0, 반발력을 결정하는 AF는 0, 경계구의 높이비율 SHR은 1.0, DF는 0.95로 설정하였으며, [그림 10]은 객체가 물표면을 변화시키는 과정을 자세하게 보여주기 위하여 물표면의 움직임이 없는 상태에서 시뮬레이션한 결과이다. 2,3,4,5번에서 물표면과 포탄과의 상호작용에 의해 높이 필드가 변형된 후 선형 컨벌루션이 진행되면서 물표면과의 상호작용을 만들어 내었다.

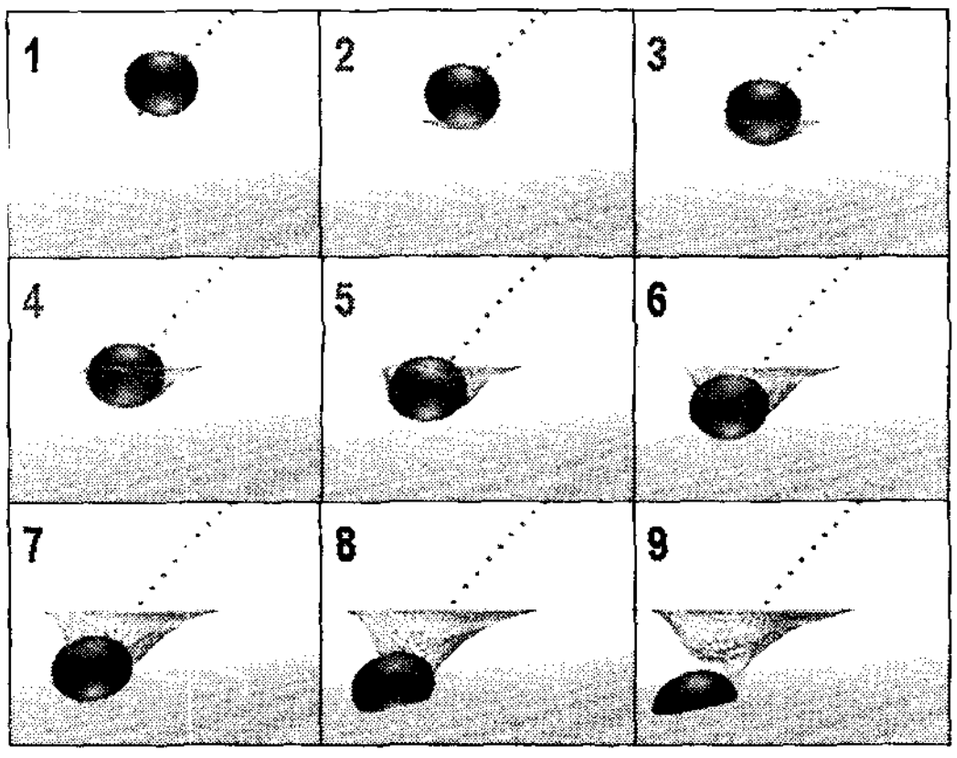


그림 10. 포탄이 변형시킨 물표면

[그림 11]은 [그림 10]의 시뮬레이션을 물표면의 측면과 위쪽방향에서 바라본 것으로서, 포탄이 물에 떨어지면서 물웅덩이가 생겼다가 물기둥으로 솟아오르는 모습과, 포탄의 진행 방향으로 이동하는 물결의 형태까지 자연스럽게 표현되었다.

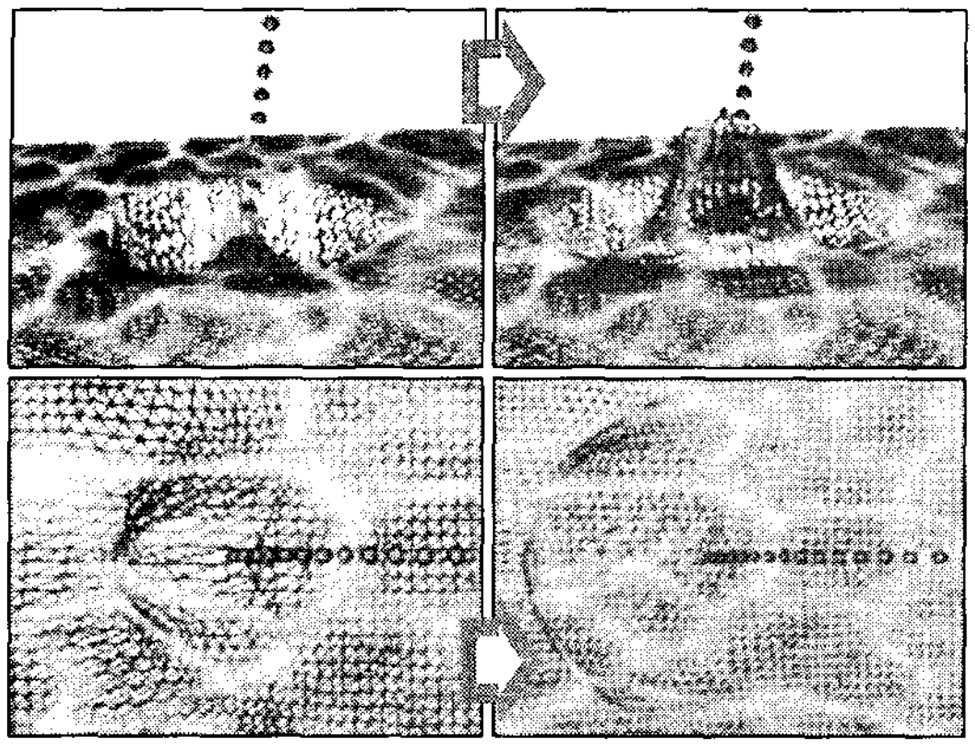


그림 11. 포탄발사 시뮬레이션 결과

[그림 12]의 A는 단순하게 생성한 긴 물결에 포탄이 떨어지는 상황을 시뮬레이션 한 모습이다. 포탄이 떨어지면서 물결의 표면을 변형하였고, 일그러진 물결이 포탄이 발생시킨 물기둥과 주변 물결에 의해 점차 복귀되는 모습을 보여준다. [그림 12]의 B는 포탄 발사 시뮬레이션의 마지막 결과로 포탄에 의해 발생된 파동이 힘을 가지고 진행하는 긴 물결에 의해 일그러지면서 소멸되는 형태를 보여준다.

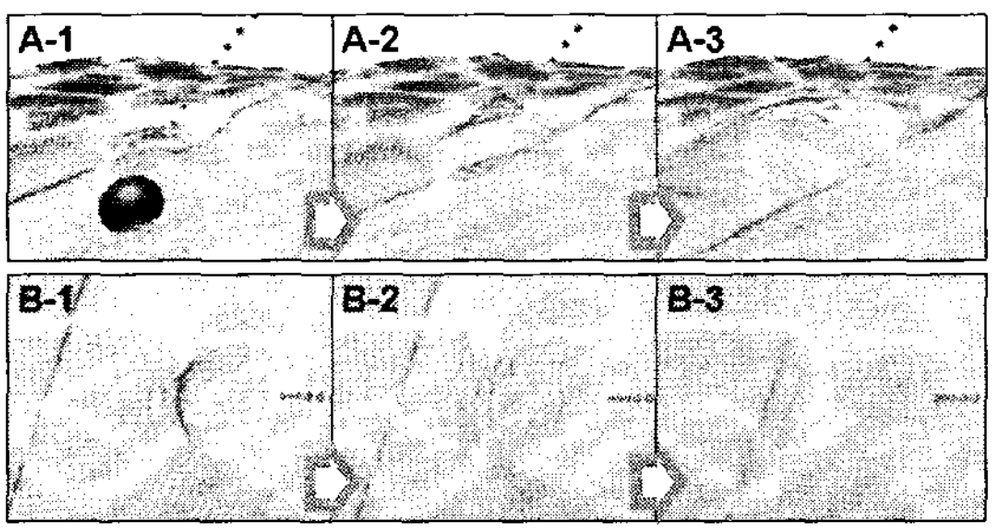


그림 12. 긴 물결과 포탄의 상호작용

2. 점핑 물고기 시뮬레이션

기본 경계구의 형태와 유사한 포탄보다 좀 더 복잡한 형태의 객체와 물표면의 상호작용을 위해 물고기 객체를 이용하여 수면에서 점핑하는 시뮬레이션을 수행하였다. [그림 13]은 시뮬레이션에 사용된 물고기 객체와 설정된 경계구의 모습이다.

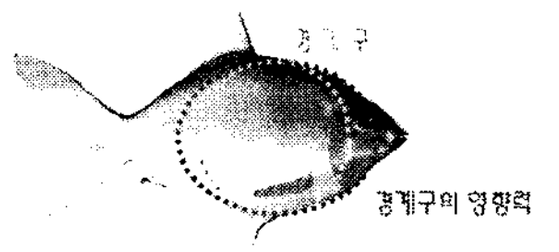


그림 13. 물고기 객체의 경계구 설정

물고기 객체는 수면 아래에서 점핑하여 수면 위로 올라간 뒤 다시 수면 아래로 들어가는 점핑 동작을 수행하는데, 수면 위로 올라갈 때는 *SD*를 1로 설정하고 수면 아래로 내려갈 때는 *SD*를 -1로 설정하여 물표면과의 상호작용을 구현하였다. 물고기에 적용된 경계구는 물고기 몸통보다 적은 형태로 설정하였기 때문에 *SHR*의 비율을 1.4로 늘려 물표면에 충돌하였을 때 변형되는 수면이 더욱 깊거나 높아지도록 경계구의 영향력을 [그림 13]처럼 타원체로 만들어 내었다. 또한 물고기는 앞서 실험한 포탄에 비해 가벼우므로 발생된 물파동이 빠르게 변형되도록 가속인자 *AF*값을 -0.3으로 설정하고, 생성된 파동이 빠르게 소멸되도록 *DF*는 0.93으로 조정하였다. [그림 14]는 물고기 점핑시뮬레이션의 결과로서 수면 밑으로 내려갈 때 나타난 구는 설정된 경계구이다.

물고기의 지느러미나 꼬리부분이 물표면에 일으키는 영향은 적을 것이며, 물고기 객체 모두를 포함하는 경계구로 설정할 경우 객체가 포함되지 않은 영역을 많이 포함하여 물표면과의 상호작용을 어색하게 일으킬 수 있기 때문에, 적용된 경계구의 크기는 물고기 객체보다 적은 크기로 설정되었다. 경계구를 객체 형태에 적합한 사이즈가 다른 여러 개의 경계구 집합으로 설정하여 물표면 변형을 수행한다면 객체의 영향에 따라 물표면을 좀 더 세밀하게 표현할 수 있을 것이다.

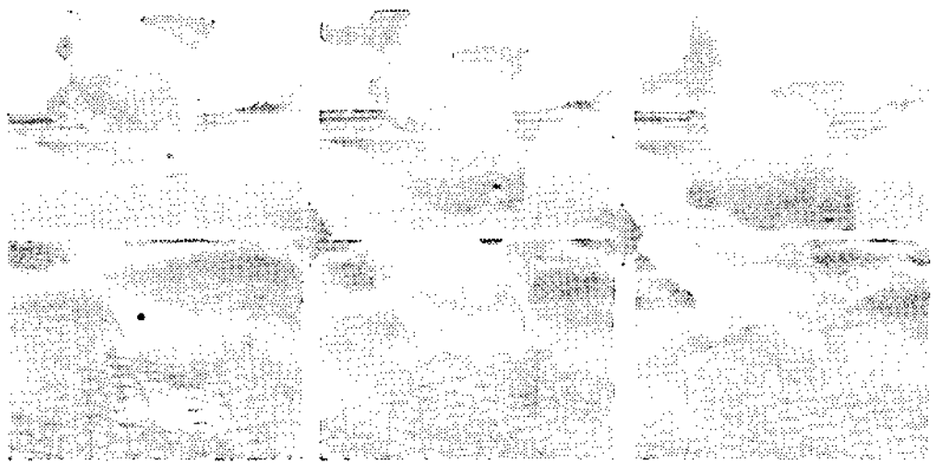


그림 14. 물고기 점핑 시뮬레이션 결과

3. 항해하는 배 시뮬레이션

포탄발사 시뮬레이션이나 물고기 점핑 시뮬레이션은 하나의 경계구를 사용하여 수면 밑으로 완전히 들어가거나 수면 밖으로 점프하면서 상호작용을 만들어내었다. 수면의 경계에 위치하여 지속적인 물표면의 변형을 만들어내는 모습을 표현하기 위하여 6개의 경계구를 사용하여 수면위에서 돌아다니는 배와 상호작용하는 물의 모습을 시뮬레이션 하였다. [그림 15]는 시뮬레이션에 사용된 배의 이미지와 설정된 경계구(점선원)를 나타낸다.

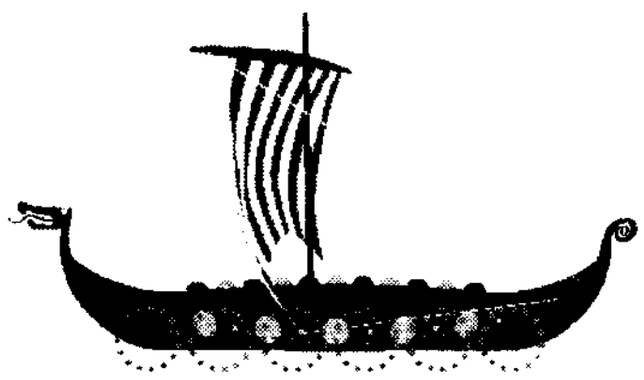


그림 15. 배에 적용할 6개의 경계구

배는 수면위에 떠있는 상태로 움직이기 때문에 배에 적용된 6개의 경계구는 항상 물표면을 변형하게 된다. 즉, 배가 이동하는 속도에 따라서 물표면과의 상호작용도 다르게 표현될 수 있다. 경계구의 *SR*은 7로 지정하고, 물표면의 깊이 변형이 크지 않도록 *SHR*은 0.2로 설정하였으며, 표면에서 변형되는 깊이에 대한 영향력을 줄이는 대신 전체적인 파동의 움직임이 많아지도록 *AF*는 0.3, *DF*는 0.96으로 설정하였다. [그림 16]은 배의 이동속도에 따라 만들어진 물의 표면을 측면에서 보여주는 것으로, A는 정지상태, B는 조금씩 움직이고 있는 순간의 모습, C는 B에 비해 5배가량 빠르게 움직였을 때의 물표면 모습이다.

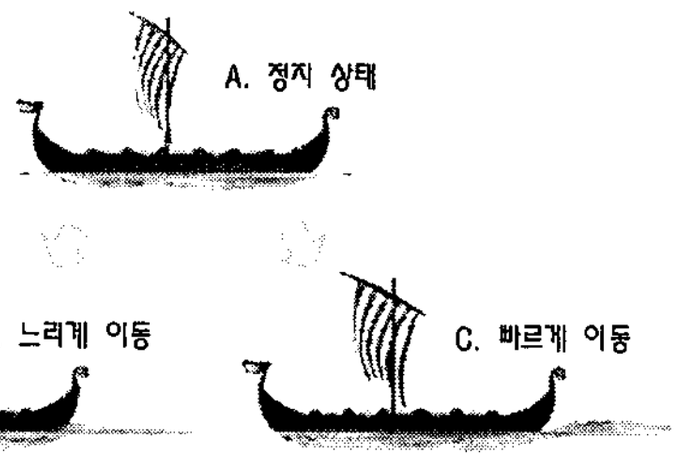


그림 16. 배의 이동속도에 따른 물표면 변화

[그림 16]의 결과와 같이 정지상태일 때는 항상 같은 위치에 있는 경계구로 인하여 물표면의 모습은 변형되었지만, 선형 컨벌루션이 진행되어도 다른 움직임이 만들어지지 않았다. 반면 느린 속도로 이동시켰을 경우에는 배의 후미 쪽에서 파동의 반발이 표현되었고, 빠르게 이동시켰을 때는 파동의 반발이 표현되는 지점을 빠르게 벗어남으로써 느리게 이동했을 때보다 기다란 파동의 모습이 표현되었다. [그림 17]은 카메라가 고정된 상태에서 배가 카메라 쪽을 향해 이동하고 있을 때를 나타내는데, 가장 앞쪽에 있는 경구계에 의해 발생된 파동만이 2차 파형의 영향을 받지 않아 파동의 경계가 물결의 갈라짐으로 표현되었다. 그리고 [그림 18]은 배가 이동하면서 발생시킨 파동이 난류를 생성하는 모습으로 지나간 자리에 거친 부서짐이 생성되는 현상을 보여준다.

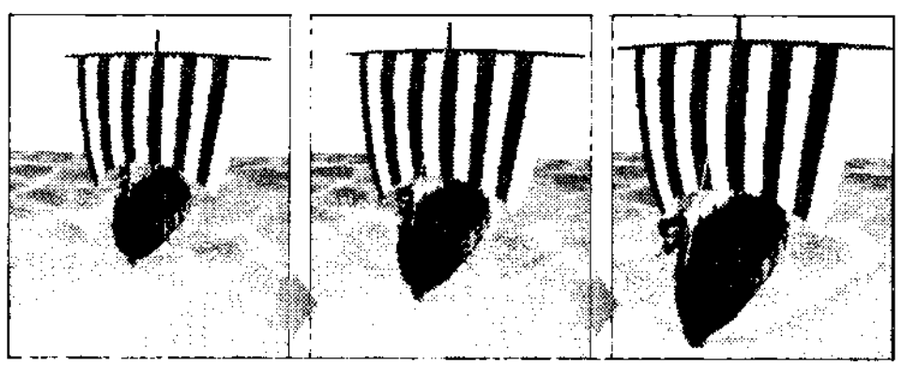


그림 17. 배의 앞쪽에서 발생되는 파동에 의해 갈라지는 물결

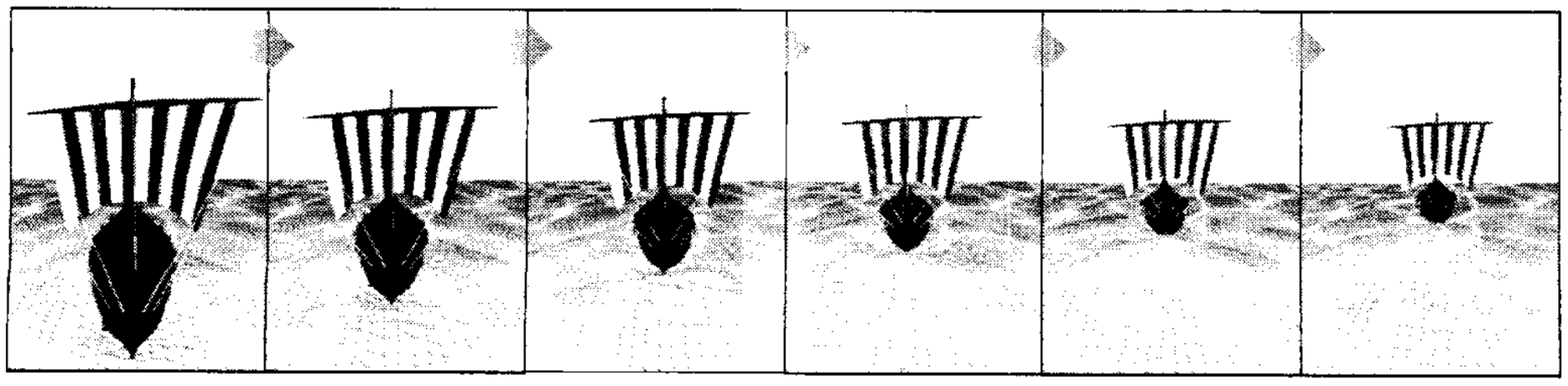


그림 18. 물표면에 표현된 난류와 배가 지나간 자리에 생긴 부서짐

시뮬레이션에 사용된 배는 수면에서 요동치거나 회전하는 것과 같은 움직임을 고려하지 않았으며, 배에 사용된 6개의 경계구는 모두 같은 크기와 같은 특성을 적용시켰다. 하지만 모터보트와 같이 수면에서 심하게 요동치거나 배의 앞부분이 일정 부분 떠서 이동하는 것 같은 여러가지 상황에서는 사용되는 각 경계구의 특성을 상황에 맞게 조절시킴으로써 다양한 상호작용을 표현해 낼 수 있을 것이다.

4. 시뮬레이션 성능

본 논문에서 제안한 방법들을 이용한 물결, 포탄발사, 점핑하는 물고기, 움직이는 배에 대한 시뮬레이션은 CPU 2.6GHz, Main Memory 1GB, GPU 600MHz의 시스템에서 DirectX SDK 9.0을 이용하여 구현되었으며, 스크린의 크기는 800×600의 해상도에서 이뤄졌다.

[표 3]은 높이 필드 크기에 따른 시뮬레이션 성능을 나타내는데, 일반적인 실시간 시뮬레이션의 요구 성능인 15fps를 월등히 뛰어넘는 성능을 보여주었다. 시뮬레이션은 초당 50여개의 경계구를 물표면에 생성하고 방향성을 주어 물결과 같은 형태로 지속적인 물표면의 움직임을 표현하고 있는 상황에서 이루어졌다. [표 3]의 음영영역은 객체와 물표면과의 상호작용에 대한 순수한 알고리즘 처리능력을 확인하기 위하여 객체의 렌더링은 제외한 채 경계구만 나타내고 상호작용을 유지하였을 때의 결과이고, 동시 사용은 포탄, 물고기, 배 모두를 동시에 시뮬레이션 하였을 때의 결과를 나타낸다. 포탄 발사 시뮬레이션은 포탄이 발사되고 지형 아래로 자취를 감출 때 다시 새로운 포탄을 자동으로 발사하도록 하였고, 물고기 점핑시뮬레이션은 지정된 영역에서 원을 그리며 점핑동작을 계속 수행하도록 하였으며, 움직이는 배는 물표면에 위치하여 이동하면서 지속적으로 물표면에 변형을 만들어내도록 하였다.

표 3. 시뮬레이션 성능(fps)

높이 필드 해상도	시뮬레이션 분류							
	포탄 발사		점핑 물고기		항해하는 배		동시 사용	
64×64	91	114	84	113	74	113	45	112
128×128	41	43	41	43	32	42	31	42

[표 4]는 물표면 표현 및 상호작용 등을 모두 제외하고, 시뮬레이션에 사용된 객체의 렌더링만을 진행하였을 때의 성능을 나타낸다. 단독으로 렌더링 시켰을 때는 움직이는 배에서 가장 낮은 성능을 보여주었는데, 이것은 배에 사용된 재질의 특성이 좀 더 복잡하고 정점의 수가 다른 객체에 비해 많았기 때문이다. 즉, 물표현과 상호작용을 표현하는 알고리즘 계산시간과는 별도로 시뮬레이션에 사용된 객체가 많고 복잡할수록 렌더링 시간을 더욱 소요시키면서 성능에 영향을 끼치게 된다고 할 수 있다.

표 4. 시뮬레이션에 사용된 객체의 렌더링 성능(fps)

포탄 발사	점핑 물고기	움직이는 배	동시 사용
124	112	94	48

30fps의 성능을 나타내는 동일한 상황을 가정하고 비교하였을 때, 점핑 물고기 시뮬레이션과 움직이는 배에 대한 시뮬레이션은 1초당 물고기는 6개의 경계구가 물표면을 변형하게 되고, 배는 180개의 경계구가 물표면을 변형하게 된다. 시간에 따라 작용하는 경계구의 양은 30배에 달하지만 본 논문에서 제안하는 방법의 순수한 처리능력을 보여주는 [표 3]의 음영영역처럼 점핑물고기와 움직이는 배의 시뮬레이션 결과는 거의 같음을 확인할 수 있다.

시뮬레이션에 사용된 경계구의 개수에 따른 시뮬레이션 성능변화를 좀 더 명확하게 하기 위하여 점핑 물고기 시뮬레이션에 다수의 물고기를 생성하여 실험을 하였다. 실험은 점핑 물고기 시뮬레이션의 경우와 동일하게 경계구와 물고기를 1대 1로 매칭하였고, 수면 위아래를 지속적으로 움직이게 하였다. 또한 물고기에 설정된 경계구는 크기에 따라 물표면을 변형할 범위가 달라져 성능에 영향을 끼칠 수 있기 때문에 경계구의 SR을 모두 20으로 고정시켰으며, 정확한 측정을 위하여 물고기 렌더링은 제외하였다. 사용한 경계구(물고기)의 개수에 따른 실험결과는 [표 5], [그림 19]와 같으며, 경계구 1개를 사용한 경우와 1000개를 사용한 경우 2.1fps의 미세한 성능차이가 있었다.

표 5. 경계구의 개수에 따른 시뮬레이션 성능

경계구 수 (물고기 수)	fps	경계구 수 (물고기 수)	fps
1	39.2	1000	37.1
10	39.1	3000	32.5
30	38.9	10000	23.7
100	38.8	30000	13.3
300	38.4	100000	5.4

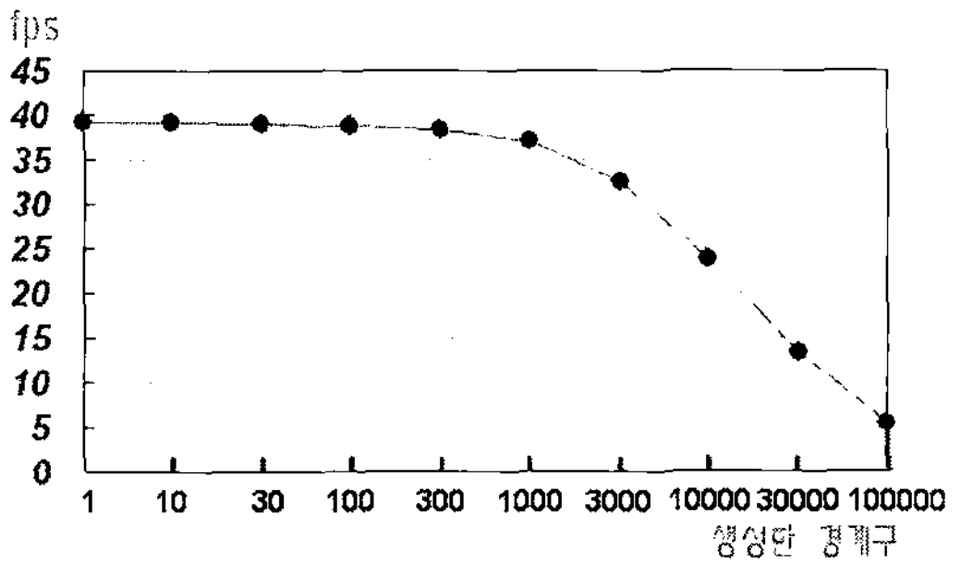


그림 19. 경계구 개수에 따른 시뮬레이션 성능그래프

결국 시뮬레이션 성능은 물표면에 사용한 높이 필드 크기가 결정하게 된다. 또한 다양한 상황에 대하여 더욱 많은 경계구를 추가하여 상호작용을 만들어내더라도 시뮬레이션 성능에는 큰 영향을 끼치지 않는다고 할 수 있으며, 대화식 문맥을 요구하는 실시간 시뮬레이션에서 복잡한 상황이 발생하더라도 다른 외부 렌더링에 대한 성능 고려만 이뤄진다면 추가적인 성능저하 없이 시뮬레이션이 가능할 것이다.

VI. 결 론

본 논문에서는 선형 컨벌루션을 이용하여 물표면을 애니메이션 하였고, 경계구 모델을 정의하여 물파동을 생성하고 물표면과 객체와의 상호작용을 실시간으로 표현해 내었다.

기존의 넓은 지역의 물표면을 표현하기 위한 기법들은 주로 파동방정식을 기반으로 하여 지역적 제어가 어렵다는 문제점을 갖기 때문에 실시간으로 상호작용을 만들어내기가 쉽지 않다. 제안된 방법은 객체를 하나의 경계구 또는 경계구의 집합으로 설정하고 경계구를 이동시키면서 물표면을 변형함으로써 객체와의 상호작용에 의한 수면 밑이나 수면위의 모습도 자연스럽게 표현할 수 있었다. 또한 먼저 발생한 파가 뒤이어 발생한 파와 충돌하여 파형을 상쇄시키면서 진행방향으로 파동이 이동하는 것처럼 보이게 되는 호이겐스의 원리에 따라 파동 애니메이션을 생성하였다.

시뮬레이션 성능은 높이 필드 크기에 따라 결정되었으며 실시간 시뮬레이션에 무리없는 높은 성능은 보여주었다. 그리고 경계구를 추가하여 상호작용을 늘리더라도 시뮬레이션 성능에 큰 영향을 끼치지 않는 장점은 실시간 시뮬레이션에서의 한정된 상호작용의 제약사항을 해결할 수 있을 것이다. 또한 물과 외부환경과의 상호작용 생산성이 높아짐에 따라 게임과 같은 사용자 대화식 시뮬레이션에서 이전에 생각하지 못했던 새로운 게임플레이 요소를 만들어 낼 수 있을 것으로 기대된다.

향후 디테일한 물표면의 모습을 표현하도록 경계구의 결합과 DF를 조절하여 다양한 파동을 생성하고 결합하는 방법을 찾고, 더욱 사실성을 높이기 위하여 파동의 부서짐에 대하여 파티클로 표현해야 할 것이다. 또한 객체의 크기와 무게, 속도 및 재질 특성에 따른 가속인자 설정 방법을 연구하고, 세밀하고 사실적인 물표면 애니메이션을 생성하기 위한 복잡한 형태를 갖는 객체에 대응하는 경계구 생성을 위한 공간분할 기법에 대한 연구가 필요하다. 그리고 다양한 주변 환경에 따른 상호작용 인자를 추가함으로써 사실적인 실시간 물표면 애니메이션을 생성하고자한다.

높이 필드 크기에 의존적인 시뮬레이션 성능은 지형관리에서 사용되는 LOD(Level of Detail)기법을 높이 필드에 적용하고, 선형 컨벌루션 과정을 MMX(multimedia extension), SSE(streaming SIMD extensions) 등을 사용하여 CPU를 효율적으로 운용하고, 알고리즘의 일부를 GPU와 함께 병행하여 처리한다면 시스템 성능을 더욱 향상시킬 수 있을 것이다.

참 고 문 헌

[1] N. Foster and D. Metaxas, "Realistic Animation of liquids," Graphical Models and Image Proc,

Vol.58, No.5, pp.471-483, 1996.

[2] J. Stam, "Stable Fluids," in Proc. of SIGGRAPH, pp.121-128, 1999.

[3] N. Foster and R. Fedkiw, "Practical Animation of Liquids," in Proc. of SIGGRAPH, pp.23-30, 2001.

[4] S. Premoze, T. Tasdizen, J. Bigler, A. Lefohn, and R. Whitaker, "Particle -Based Simulation of Fluids," Computer Graphi cs Forum, Vol.22, No.3, pp.401-410, 2003.

[5] A. Fournier and W. Reeves, "A Simple Model of Ocean Waves," in Proc. of SIGGRAPH '86, pp.75-84, 1986(8).

[6] M. Kass and G. Miller, "Rapid, Stable Fluid Dynamics for Computer Graphics," in proc. of SIGGRAPH '90, pp.49-57, 1990.

[7] J. Chen and N. Lobo, "Toward Interactiverate Simulation of Fluids with Moving Obstacles Using Navier-Stokes Equations," Graphical Mo dels and Image Proc., Vol.57, No.2, pp.107-116, 1995(3).

[8] J. Tessendorf, "Simulating Ocean water," In SIG GRAPH Course Notes, Addison-Wesley, 1999.

[9] S. Thon, J. Dischler, and D. Ghazanfarpour, "Ocean Waves Synthesis Using a Spectrum-Ba sed Turbulence-Function," in Proc. of the Int'l Conf. on Com-puter Graphics, p.65, 2000(6).

[10] D. Hinsinger, F. Neyret, and M. Cani, "Interacti ve Animation of Ocean Waves," in Proc. of the 2002 ACM SIGGRAPH/Eurographics Symp. on Computer Animation, pp.161-166, 2002.

[11] J. Loviscah, "A Convolution-Based Algorithm for Animated Water Waves, "Eurographics 200 2 Short Paper Presentations, pp.381-389, 2002.

[12] J. Loviscah, "Complex Water Effects at Interac -tive Frame Rates," in Proc. of WSCG, 2003.

[13] J. L. Mitchell, "Real-Time Synthesis and Rend ering of Ocean Water," ATI Research Technical Report, 2005.

[14] Andrew Kirmse, *Game Programming Gems4*, 2004.

[15] M. Mason, *Special Effects Game Programming with DirectX*, 2001.

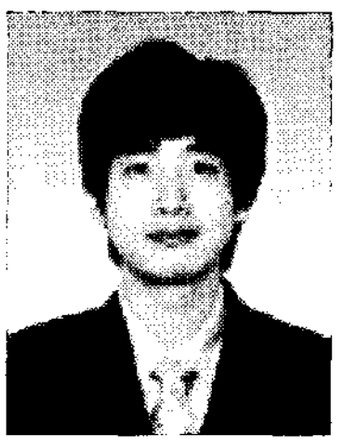
[16]http://www.gamedev.net/reference/articles/article2138.asp

[17]http://www.gamedev.net/reference/articles/article915.asp

[18] 표순형, 구본기, "CG 유체 표현 기술 동향", 전자통신동향 분석, 제20권, 제4호, pp29-45, 2005.

저 자 소 개

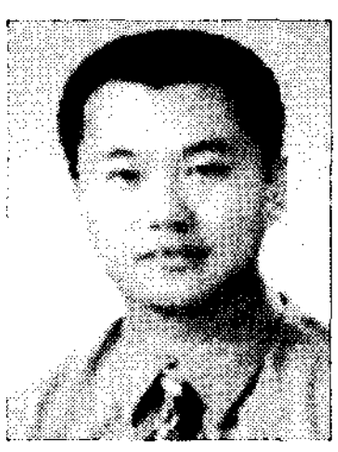
강 경 헌(Gyeong-Heon Kang) 준회원



- 2006년 2월 : 동신대학교 멀티미디어콘텐츠학과(이학사)
- 2008년 2월 : 동신대학교 디지털콘텐츠학과(이학석사)
- 2008년 2월 ~ 현재 : 동신대학교 디지털콘텐츠협동연구센터 연구원

<관심분야> : 디지털콘텐츠, 모바일콘텐츠, 유체애니메이션, 입체영상

이 현 철 (Hyun-Cheol Lee) 정회원



- 1996년 2월 :동신대학교 컴퓨터학과(이학사)
- 1998년 2월 :동신대학교 컴퓨터학과(이학석사)
- 2003년 2월 :동신대학교 컴퓨터학과(이학박사)
- 2005년 ~ 현재 : 동신대학교 디지털콘텐츠학과 전임강사

<관심분야> : 멀티미디어통신, 유체애니메이션

허 기 택(Gi-Taek Hur) 종신회원

- 1984년 2월 : 전남대학교 계산통계학과(이학사)
- 1986년 2월 : 전남대학교 계산통계학과(이학석사)
- 1994년 2월 : 광운대학교 전자계산학과(이학박사)
- 1989년 ~ 현재 : 동신대학교 디지털콘텐츠학과 교수

<관심분야> : 영상처리, 유체역학, 디지털콘텐츠

김 은 석(Eun Seok Kim) 정회원

- 1995년 2월 : 전남대학교 전산학과(이학사)
- 1997년 2월 : 전남대학교 전산통계학과(이학석사)
- 2001년 2월 : 전남대학교 전산통계학과(이학박사)
- 2002년 3월 ~ 현재 : 동신대학교 디지털콘텐츠학과 조교수

<관심분야> : CG, 디지털콘텐츠, 애니메이션