움직이는 B-Spline 곡면을 이용한 유체 흐름의 제어

황철현 경민호
아주대학교 미디어 연구실
lndnjc, kyung@ajou.ac.kr

Fluid flow control using animated B-Spline surface

Chul-Hyun Hwang, Min-Ho Kyung
Division of Media, Ajou University

요 약

유체 시뮬레이션은 Navier-Stoke 방정식의 해를 구하는 과정으로 볼 수 있는데, 이 방정식은 초기 조건 및 주변 환경에 따라 매우 민감하게 반응하기 때문에 사용자가 원하는 형태로 제어하는 것이 매우 어려운 일이다. 본 논문에서는 유체의 움직임을 실제 공간에 임베드된 smooth manifold 위로 제한하고, 유체의 움직임을 manifold의 모양에 의해 직관적으로 제어하는 방법을 제안한다. 제어 manifold 안의 유체의 흐름을 자연스럽게 유지하기 위하여 경계에 가상의 중력장을 설정하여 유체가 경계면에서 자연스럽게 내부로 유도되도록 하였다. 본 논문의 유체 제어 방법은 제어 manifold의 모양을 키프레임 보간함으로써 간접적으로 유체 애니메이션의 키프레임 애니메이션으로 만드는 것도 가능하다. 이 과정에서 제어 manifold의 변형에 의한 유체정보를 재구성이 필요한데, 본 연구에서는 그리드의 재샘플링을 통해 해결하는 방법을 제시하였다.

1. 서론

물의 움직임, 기체의 유동현상, 구름의 모양 등 불규칙한 자연현상에 대한 표현은 컴퓨터그래픽에서 흥미롭고도 어려운 분야이다. 하지만, CFD(Computational Fluid Dynamics), 유체의 수치해석적 방법이 컴퓨터 그래픽스와 접목되면서 보다 정확하고 자연현상에 가까운 묘사가 가능하게 되었다. 그러나, 불행히도 보편적인 방법이 일반적으로 모든 자연현상에 적용되는 것은 아니다. 각 물질마다 같은 조건에서 보편적인 힘에 반응하는 형태가 다르기 때문에 물질 고유의 특성을 고려한 지배방정식을 세워주어야 하며, 주위환경에 적합한 수치해석적 방법을 고려해야 한다.

유체의 움직임을 정의하는 방정식이 주어지면 시뮬레이션 결과는 주위 환경의 설정과 초기 조건에 의해서 결정된다. 따라서 유체의 움직임을 원하는 형태로 만들어 주기 위해서는 초기 조건과 환경 설정 값들을 잘 찾아 주어야 하는데, 이는 매우 어려운 일이다. 기존 연구에서는 이를 해결하기 위해 공간에 인위적으로 외부 힘을 배치하거나 중력장을 설정하여 시뮬레이션 중간에 유체의 움직임을 강제로 변화시켜 주는 방법을 사용하였다. 이러한 변화는 자연적인 것이 아니므로 지나치게 크게 일어나면 자연스러운 유체 애니메이션이 만들어지지 않는다. 따라서 기존 방법들은 외부 힘의 간섭을 최소화시키면서 원하는 제어 결과를 얻기 위해 최적화 방법을 사용하였다. 하지만 최적화 방법은 일반적으로 많은 시간이 걸리고 원하는 결과를 항상 보장하지는 않는다. 또한 직관적인 인터페이스를 만들기가 어렵기 때문에 일반 애니메이터들이 사용하기에는 어려운 점이 많다. 본 논문에서는 사용자가 직관적이면서 효과적으로 유체를 제어할 수 있게 해주기 위해, 기존 방법과는 달리 manifold를 이용한 유체 제어 방법을 제시하였다.

본 논문의 아이디어는 Stam[14]에서 얻었다. Stam은 3차원 공간에서 임의의 모양을 가지는 Catmull-Clark곡면 위에서 흐르는 유체 애니메이션을 만들었다. 이를 위해 유체가 존재하는 실제 공간(world space)을 격자로 나누는 기존의 방법들과는 달리, Catmull-Clark곡면을 구성하는 각 patch의 2차원 parameter domain을 그리드 시스템으로 만들고, 여기에 유체의 속성을 저장하였다. 이렇게 함으로서 유체의 흐름은 곡면 위로 제한되고, 3차원 모양을 가지는 곡면임에도 2차원 CFD solver를 이용하여 빠르게 풀 수 있었다. 여기에 사용한 CFD solver는 3차원 공간의 유체 방정식을 풀기 때문에, parameter domain에 저장된 유체 속성을 3차원 공간으로 변환해 주는 과정이 유체 방정식에 추가되게 된다. Stam의 solver는 Catmul-Clark 곡면뿐만 아니라 일반적인 smooth manifold에 대해서도 적용할 수가 있다.

본 논문에서 제시하는 방법은 유체의 흐름을 실제 공간에 임베드 된 smooth manifold 위에서 흐르게 하고, 이 manifold의 형상을 바꾸어줌으로써 원하는 유체의 움직임을 만드는 것이다. 파라메터 도메인에서 정의된 유체 정보에 대해 유체 방정 식을 푸는 방법은

Stam이 유도한 식을 사용하였다. 유체를 제어 manifold에 의해 제어하는 방법의 여러 가지 장점을 가지고 있다. 첫째, 비선형 최적화 등의 많은 시간이 걸리는 계산 부분이 없기 때문에 기존의 방법들에 비해 매우 빠르다. 또한 일부 방법들에서 사용되는 전처리 과정도 필요 없다. 둘째, 제어 manifold를 키프레임 보간으로 변형시킴으로써 유체의 키프레임 애니메이션을 쉽게 만들 수 있다. 셋째, 원하는 유체의 형상을 제어 manifold의 형상으로부터 유추할 수 있기 때문에 직관적인 인터페이스를 제공해 준다. 기존의 방법들에서는 이러한 장점들을 부분적으로는 가지고 있지만 동시에 제공하지는 못했다.

본 연구에서 제시한 제어 방법이 사실적인 유체 애니메이션을 만들기 위해서는 두 가지 이슈를 해결해야 한다. 첫째, 제어를 위해 사용하는 manifold는 실제 물체가 아니기 때문에, 그 존재가 드러나지 않고 숨겨져 있어야 한다. 그와 동시에 제어 manifold의 경계 형상에 따라 유체 흐름의 형상이 바뀌도록 유도해야 한다. 이 두 가지 모순된 조건들의 타협점으로 우리는 가상의 경계 중력장을 도입하여 경계 제한에 의한 효과가 유체에 점진적으로 작용하도록 하였다. 둘째, 제어 manifold를 시간에 따라 변형시킬 때, parameter domain에 저장된 유체의 속성을 재구성해 줄 필요가 있다. 예를 들어 parameter domain의 유체 속성을 그대로 두고 제어 manifold를 변형하면 실제 공간에서의 유체의 속성이 따라서 변하게 된다. 이 변화는 유체 방정식에 의한 변화가 아니기 때문에 부자연스러운 결과를 낳게 된다. 따라서, manifold 변형을 parameter domain에서 상쇄시키는 것이 필요한데, 이를 위해 본 논문에서는 유체 속성을 재샘플링하는 방법을 제시한다.

논문의 구성은 다음과 같다. 2장에서는 기존 연구에 대한 소개를 하고, 3장에서는 유체 제어 방법의 구체적인 설명을 한다. 4장에서는 구현 방법에 대해 설명하고, 5장에서는 실험 결과를 보여준다. 6장에서는 결론과 앞으로의 연구 방향에 대해 논의한다.

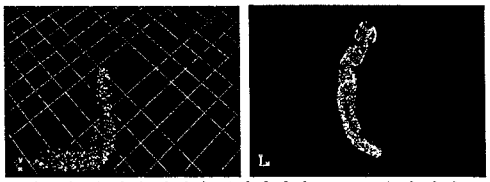
2. 관련 연구

컴퓨터 그래픽에서의 유체유동 시뮬레이션은 새로운 알고리즘과 빠른 하드웨어의 발전으로 그 전성기를 맞고 있다. Kass와 Miller[10]는 3차원상에서 Navier-Stokes 방정식을 풀었으며, Chen과 Lobo[3]는 2차원상에서 압력을 이용한 N-S 방정식의 해법을 마련했다. 또한 Foster와 Metaxas[8]는 이 해법들을 물과 연기에 적용시켰다. 한편, Stam[13]의 semi-Lagrangian 방법을 사용하여 유체 시뮬레이션을 실시간에 근접하게 계산할 수 있음을 보였다. 본 논문에서의 Stam의 유체 해법을 사용하여 유체 제어를 하였다. 이것은 다른 방법보다 계산이 빠르고 구현하기 쉬우며, 연기, 물, 불과 같은 현상들을 비교적 정확하게 시뮬레이션 할 수 있다.

유체의 움직임을 제어하는 연구는 Foster와 Fedkiw[6]가 격자 셀에 속도 값을 정해 유체의 흐름을 제어하는 방법을 제시함으로써 활기를 띠기 시작했다. 그 후, Treuille 등[15]은 유저가 지정한 유체의 키프레임을 만족하도록 가상의 wind force 파라메터를 조정하여 연기를 제어했다. McNamara등[11]은 유사한 방법을 사용하여 유체를 제어하였는데, 많은 제어 파라메터에 대한 최적 해를 효율적으로 구하기 위해 adjoint method를 도입하였다. Fattal과 Lischinski[5]는 주어진 목표 형태로 향하는 유도력을 자동으로 생성하여 이 힘에 의해 유체를 제어하는 방법을 사용하였다. 유사하게 홍정모와 김창헌[9]은 t 목표 형태로부터 유도력 대신 potential field를 생성하여 이를 이용해서 유체가 유도되도록 하였다.

이들 방법은 world 공간 격자 환경에서 원하는 형태를 유도하는 외부힘이나 potential field를 구하는데 있다. 하지만, 우리는 곡면을 이용함으로써, 외부 힘 대신에 곡면에서의 potential 힘을 정의할 것이다. 또한 실제 공간이 아닌 파라메터 도메인을 사용했기 때문에 유체가 존재하지 않는 불필요한 공간이 계산에 포함되지 않는다. 따라서 곡면의 다양한 제어 효과와 더불어 유체의 계산 시간을 훨씬 줄여 준다.

3. 문제정의 및 분석



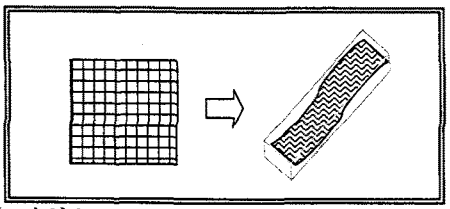
[그림 1] 터널 형태의 곡면상에서 흐르는 유체 파티클

우리는 본 논문에서 [그림 1]과 같은 2D flow의 제어를 구현하였다. 제어 manifold로는 B-spline 곡면을 사용하였다. 3차원인 경우에는 FFD(free-form deformation)에서 많이 사용되어온 B-spline volume을 제어 manifold로 사용할 수 있다. 먼저 B-spline 곡면의 각 패치들의 파라메터 도메인을 유체의 속도장과 밀도 분포를 저장하기 위한 격자로 나눈다. 그리고, 유체 흐름의 모양을 spline 곡면의 경계 모양으로 유도하기 위해 경계선에 인접한 격자 셀마다 경계중력장을 정의하였다. 유체 방정식을 파라메터 도메인에 대해 푸는 것은 Stam의 논문에 소개된 식을 이용하였다. 이에 대한 설명은 3.2에서 하겠다. B-spline 곡면을 키프레임 보간으로 애니메이션시키는 경우 이에 맞도록 각 패치에 저장된 유체 정보가 재샘플링되어져야 한다. 이에 대한 자세한 설명은 3.3에서 하도록 하겠다.

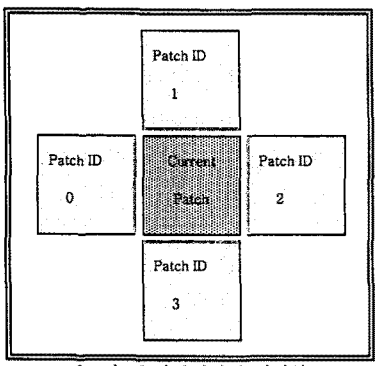
3.1 B-Spline 곡면

B-spline 곡면은 여러 장의 패치로 구성되어 있고, 각각의 패치 p 의 파라메터 (u,v) 는 도메인 $\Omega_p = [0,1]\times[0,1]$ 를 가진다. [그림 2]와 같이 Ω_p 는 $n\times n$ 크기를 가지는 격자로 나누어 지고 격자 셀의 중심에는 유체의 속도와 밀도가 저장된다. B spline 곡면 상의 점은 함수 $B_p : \Omega_p \rightarrow R^2$ 로 정의된다.

곡면 상의 유체는 패치와 패치사이를 자유롭게 흘러가기 때문에, 패치의 경계 셀에서는 인접 패치의 경계 셀을 같이 포함하여 계산해야 한다. 인접 패치를 쉽게 찾기 위해 [그림 3]과 같이 패치마다 각 방향 인접 패치에 대한 인덱스를 PatchID[0..3]에 저장한다. 예를 들어 현재 패치의 왼쪽에 인접한 패치가 없다면 PatchID[0]에는 곡면 경계로 표시하고, 패치 p 가 인접하고 있다면 패치 인덱스 p 가 저장된다. B spline 곡면이 선형 밴드라면 패치들이 u,v 방향으로 차례로 정렬되기 때문에 현재 패치의 인덱스만으로 인접 패치를 간단히 구할 수 있다. 하지만 다양한 유체 흐름을 만들기 위해 곡면을 T형이나 O형의 구조와 같이 다양한 구조로 만드는 경우는 인접 패치에 대한 아이디 정보를 PatchID 에 저장하는 것이 편리하다.



[그림 2] 2D fluid solver(u,v), Surface control f(u,v)



[그림 3] 인접패치의 레이블

3.2 Stable Fluids Solver

유체 시뮬레이션은 Stam[13,14]의 Stable Fluids Solver를 사용한다. 이 solver는 다음과 같은 비압축성 Navier-Stokes 방정식의 해를 네 개의 단계를 거쳐서 구한다:

$$\frac{\partial u}{\partial t} = P\left\{-(u\cdot\nabla)u + \nu\nabla^2 u + f\right\}$$
$$\frac{\partial \rho}{\partial t} = \left\{-(u\cdot\nabla)\rho + k\nabla^2 u + S\right\}$$

u는 유체의 속도를 나타내고, ρ는 밀도, f는 유체에 작용하는 외부 힘을 나타낸다. 이 방정식은 다음의 네 단계를 거쳐서 풀게 된다:

addforce → diffuse → advect → project

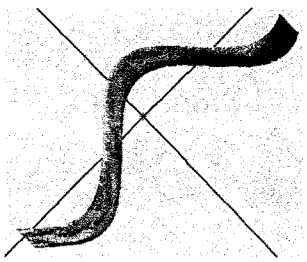
각 단계에 대한 자세한 내용은 [10]에 설명되어 있다.

위 유체 방정식을 곡면 파라메터에 대하여 풀기 위해서는 파라메터 공간에서의 유체 속도를 곡면 위에서의 속도로 변환하는 것이 필요하다. 이 변환 식은 파라메터 u,v에 대한 곡면 B_p의 미분에 관한 선형식으로 나타낼 수 있다. 이때 이 미분 행렬의 determinant를 g-metric이라고 부른다:

$$g_{i,j} = \sum_{k=1}^{3} \frac{\partial y^k \partial y^k}{\partial x^i \partial x^j}, \qquad i,j = 1,2\ ,$$
$$g = \det(g_{i,j}) = g_{1,1}g_{2,2} - g_{1,2}^2 \ > 0$$

곡면 위에서 흐르는 유체의 움직임을 구하기 위해서는 stable fluid solve의 네 단계들에 g-metric을 적용하여야 한다. g-metric을 적용하여 얻어진 각 단계별 계산식은 [1,14]에 설명되어 있다.

3.3 경계 중력장



[그림 4]경계에서의 중력장을 적용하지 않은 유체의 결과

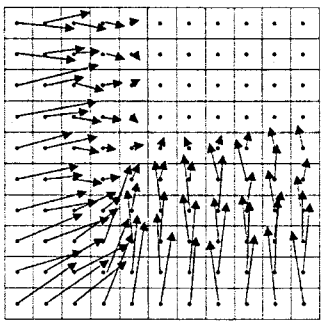
제어 곡면의 파라메터 도메인에서 정의된 유체는 일정한 시간이 지나면 곡면 전체를 채우게 된다. 일반적으로 유체가 곡면의 경계에 도달하면 경계 제한이 있는 닫힌 공간인 경우 공간 내부로 유도되거나, 또는 경계 제한이 없는 열린 공간에서는 경계 밖으로 나가

면서 소멸되어 격자 내부로 다시 돌아오지 않게 된다. 본 연구에서는 곡면 경계가 유체의 형상을 제어하는 역할을 하기 때문에 격자를 닫힌 공간으로 정의한다. 그런데, 이 경우 [그림 4]에서 볼 수 있듯이, 유체 경계의 자연스러운 불규칙성이 없어지고, 매끄러운 경계면을 가지게 되어 제어 곡면을 그대로 드러내게 된다. 이 것은 매우 자연스럽지 않은 결과로, 이를 해결하기 위해 유체가 제어 곡면의 경계에 도달하기 전에 내부로 자연스럽게 유도해 주는 가상의 중력장을 설정하였다.

경계 중력장은 유체를 곡면 내부로 유도하기 위하여 곡면 내부를 향하는 방향으로 설정된다. 곡면의 내부는 파라메터 도메인에서 쉽게 결정된다. 패치의 i번째 방향이 곡면 경계라면 곡면의 내부 방향은 $((i+2) \bmod 4)$방향이 된다. 중력장의 크기는 곡면 경계로부터의 거리에 따라 감소하도록 만들었다. 따라서 곡면 경계에서 일정 거리 이상으로 떨어지게 되면 경계 중력장의 영향을 거의 받지 않게 되어 유체 흐름에 대한 제한이 사라지게 된다. 격자 셀 i, j에서의 경계 중력장을 수식으로 표현하면

$$f_{p,u}[i][j] = \text{sign}_u \cdot k \cdot e^{-ci} \cdot R_u(t)$$
$$f_{p,v}[i][j] = \text{sign}_v \cdot k \cdot e^{-cj} \cdot R_v(t).$$

sign_u, sign_v는 곡면 경계의 방향에 반대 방향으로 결정된다. k는 중력장의 세기를 결정하는 상수로 이 값이 증가하면 유체의 분포가 곡면 내부로 집중되고, 감소하면 유체가 곡면 경계에 가깝게 분포하게 된다. 상수 c는 중력장의 영향이 미치는 범위를 정하는 상수이다. $R(t)$는 중력장에 불규칙성을 추가하여 경계 중력장에 의한 영향이 불규칙하게 변하도록 하여 유체의 유도가 자연스럽게 이루어지도록 하였다. [그림 5]는 왼쪽과 오른쪽이 제어 곡면의 경계가 되는 격자에서 경계 중력장이 만들어진 예이다.



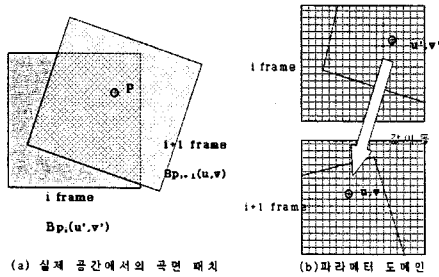
[그림 5] 중력장의 예. 그리드의 왼쪽과 오른쪽 경계가 제어 곡면의 경계가 된다.

3.4 유체의 키프레임 애니메이션

제어 곡면을 이용하여 유체의 키프레임 애니메이션을 쉽게 구현할 수 있다. 제어 곡면의 모양을 키프레임마다 설정하고 적절한 보간법을 이용하여 곡면을 보간하면 곡면의 애니메이션이 만들어진다. 이렇게 만들어진 움직이는 곡면 위에 유체를 흐르게 하면 유체의 모양이 자동적으로 키프레임 보간되어 만들어지게 된다. 하지만 이 방법은 실제로 좋은 결과를 만들어 주지 못한다. 왜냐하면 곡면의 변화가 실제 공간에서의 유체의 움직임에 더해지기 때문에, 유체 방정식에 지배되는 자연스러운 유체로 보이지 않게 된다. 간단한 예로 제어 곡면을 단순 평행 이동 시킬 경우를 보면, 곡면 위의 유체도 동시에 평행 이동되어 실제 공간과 완전히 다른 계에서 움직이는 것으로 보이게 된다. 이 문제를 해결하기 위해서는 제어 곡면의 변형을 상쇄시켜주는 과정이 필요하다. 본 연구에서는 이것을 재샘플링으로 해결하였다.

곡면의 변형은 parameter domain의 점이 대응되는 실제 공간에서의 위치가 변하는 것이다. 따라서, 유체 속성이 저장된 격자 셀이 대응되는 위치도 변하게 되어, 저장된 유체 속성이 실제 공간에서의 유체 속성과 어긋나게 된다. [그림 6]를 보면 어떤 셀 c'의 i프레임에서 대응되는 실제 공간의 위치와 $i+1$프레임에서 대응되는 위치가 달라졌다. 따라서, 곡면의 변형이 유체의 흐름을 바꾸지 않기 위해서는, 셀 c의 유체 속성이 [그림 6](b)처럼 동일한 실제 공간으로 대응되는 $i+1$프레임의 셀로 이동되어야 한다. 이 과정은 다음과 같이 수행된다. $i+1$프레임에서 격자 셀 c의 중심을 (u,v)라고 했을 때, 실제 공간에 대응되는 위치는 $B_p^{i+1}(u,v)$가 된다. 따라서, 셀 c에 저장되어야 할 유체의 속성은 i 프레임에서 $B_p^{i+1}(u,v) = B_q^i(u',v')$을 만족하는 패치 q 의 (u',v') 에 저장된 값이 된다. (u',v')에 저장된 유체의 속성은 주변 셀들의 값을 bilinear interpolation을 하여 구하게 된다.

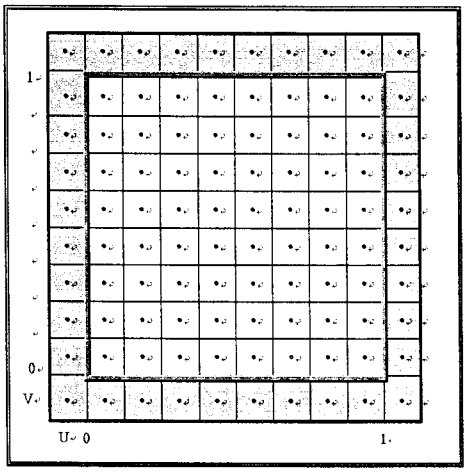
유체의 속성 중 밀도는 셀이 대응되는 곡면의 실제 면적과 관련되어 있다. 다시 말해 곡면 위의 유체 밀도가 고르게 분포되어 있더라도, 각 셀의 대응되는 곡면 면적이 다르기 때문에 셀에 저장되는 유체 밀도는 다른 값을 가지게 된다. 따라서, $i+1$프레임에서의 셀 c의 유체 밀도를 구할 때는 i프레임에서 대응되는 면적의 고려하여야 한다. 본 연구에서는 셀 c의 네 꼭지점에 대응되는 파라메터 점들을 i프레임에서 구한 다음, 그 점들로 구성되는 사각형과 i프레임 격자와의 교차 영역을 계산한다. 보통 교차 영역은 네 개의 격자 셀에 걸쳐 있게 된다. 다음은 교차 영역의 비율을 가중치로 하여 교차하는 셀의 밀도를 더해서 이 값을 셀 c에 저장한다.



[그림 6] i프레임에서 i+1프레임의
각 그리드에 값을 재설정한다

4. 구 현

여기서 우리는 Stable Fluid와 B-Spline Surface를 구현할 것이다. 다른 기능들과 확장성을 고려해 3D 프로그램인 maya에 기본 플랫폼을 맞추고, Plug-in형태로 구현했다.



[그림 7] 음영의 셀들에 인접패치의 경계값들이 저장된다.각 셀의 중심엔 속도와 밀도값이 저장된다

우선, 격자형태의 (N+2)*(N+2) 셀을 사용한다. 이 격자 셀들의 경계를 둘러싸고 있는 셀들은 인접 셀들의 값이 저장된다. [그림 7]에서 보듯이 모든 값은 그 셀 중앙에 저장된다. 이 격자 셀들은 u,v 도메인으로 나타낼 것이며, B-Spline 패치를 구성하는 함수 Bp(u,v)에 사용될 것이다. 각 패치는 격자 셀로 각각 이산화되며, 각 격자의 중앙에는 속도와 밀도값이 저장된다. 따라서 Stable Fluid 단계를 적용해 나가면서 그 값들을 매 프레임마다 이 격자 셀에 갱신해주면 된다.

Pseudo Code

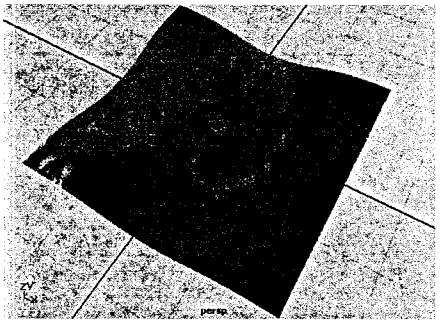
```
Start
    ControlMeshAnalysist();
    CreateBSplinePatch();
    ApplyParametricSF();

    If(per frame)
        GetAnimationMeshCV();
        RefineBSplineFunction();
        CreateBSplinePatch();
        BoundaryConditionSetting();
        SimulationByUsingSFSolver();
End
```

ControlMeshAnalysist()는 B-Spline 곡면을 생성하는 16개의 컨트롤포인트를 곡면으로부터 분류하고 인접 패치의 인덱싱작업을 수행한다. CreateBSplinePatch() 함수는 B-Spline Patch를 생성한다. 또한 ApplyParametricSF()는 이 곡면에 Stable Solver로 구한 값들을 매칭한다.

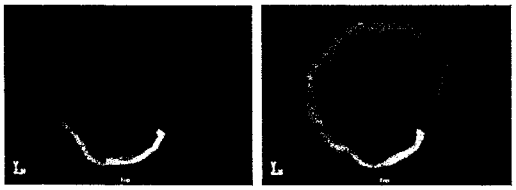
매 프레임마다 GetAnimationMeshCV()는 MeshCV를 갱신하고, RefineBSplineFunction()는 B-Spline 곡면을 생성하는 함수를 재정의한다. 그 다음 CreateBSplinePatch()로 곡면을 생성하고 경계 값들 셋팅한 다음 (BoundaryConditionSetting()), SimulationByUsingSFSolver()로 Stable Solver를 적용한다.

5. 실험 결과

본 논문에선 두 가지 형태의 연기 시뮬레이션을 실험했다. 우선, 연기가 어떤 경로를 따라 흐르는 경우와 일정 형태에서 다른 형태로 변하는 경우이다. 첫 번째는 경계 중력장만을 적용하여 실험하였고, 두 번째는 여기에 유체 속성의 재샘플링을 적용하여 실험하였다.



[그림 8] 곡면을 흐르는 연기

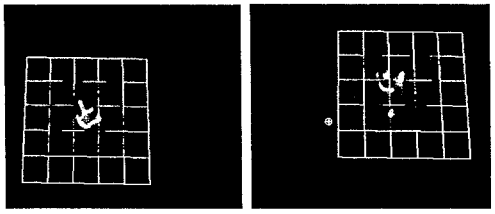


[그림 9] α 를 형성하는 연기

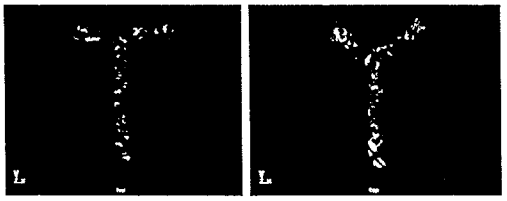
[그림 9]은 α 를 형성하는 Smokeflow를 보여주고 있다. 이것은 곡면을 따라 흐르는 유체가 경계에서 작용하는 힘에 의해 원하는 경로로 움직이는 것이다. 동영상은 37초의 길이를 가지고, 결과를 얻는 데는 50초 정도가 소요되었다. 따라서, 시뮬레이션 속도는 22.2fps로 실시간에 근접한다.

[그림 10]는 애니메이션 되는 곡면에 의해 움직이는 연기를 보여준다. 재샘플링은 매 프레임마다 stable fluid solver를 실행하기 전에 수행된다. 동영상은 11초 길이를 가지고, 결과를 얻는 데는 36초 정도가 소요되었다. 시뮬레이션속도는 9.1fps 가 나왔다.

[그림 11]은 연기가 어떤 특정형태로 변형하는 것을 보여준다. 이것은 동영상 4초의 길이를 가지고, 결과는 2분 정도가 소요되었다. 시뮬레이션속도는 1.0fps 정도로, 이렇게 느린 이유는 사용된 패치의 수가 증가하였기 때문이다.



[그림 10]곡면애니메이션에 의해 이동하는 연기



[그림 11] 움직이는 곡면에 따라 제어되는 연기

6. 결 론

이 논문은 유체를 보다 빠르고 효율적으로 제어하기 위해 제어 곡면을 사용하였다. 유체의 흐름이 제어 곡면에 제한되도록 유체의 속성을 제어 곡면의 파라메터 도메인에서 나타내었다. 따라서, 유체의 흐름은 제어 곡면의 형상에 의해서 결정된다. 제어 곡면에 의해 제어되는 유체의 모양을 자연스럽게 해주기 위해서 제어 곡면의 내부로 유체를 유도해 주는 경계 중력장을 도입하였고, 움직이는 제어 곡면의 변형을 고려하여 격자에 저장된 유체 속성을 매 프레임마다 재샘플링해 주는 방법을 적용하였다. 이 방법을 마야의 plug-in 프로그램으로 개발하였고 여러 가지 실험을 통해 그 유효성을 보였다.

제어 곡면을 이용한 유체 제어의 장점은 유체의 제어를 최적화 방법에 비해 매우 직관적인 곡면 모델링의 관점으로 바꾸어서 애니메이터가 유체의 움직임을 쉽게 디자인할 수 있도록 하였다. 특히 기존 애니메이션 시스템의 키프레임 기능을 그대로 사용할 수 있어 기존 시스템에 접목시키기가 매우 쉽다.

앞으로의 연구로는 제어 곡면의 보간을 유체의 움직임과 연동하여 시각적으로 최적의 결과를 얻도록 하는 방법을 찾는 것이다. 현재는 제어 곡면의 보간을 유체 흐름과 독립적으로 먼저 계산한 후, 유체의 흐름을 곡면 변화에 맞도록 계산하였다. 하지만, 유체의 움직임에 비해 곡면의 변화가 매우 빠른 경우 유체가 자연스럽게 반응할 수 있는 충분한 시간이 없어서 부자연스러운 흐름이 만들어진다. 이를 해결하는 것은 유체의 흐름을 분석하여 곡면의 보간 속도를 조절해 주는 것이 필요하다.

현재는 2차원 상에서의 유체에 대해서만 구현이 되어 있고, 3차원 유체의 구현은 계속 진행되고 있다. 이를 위해서는 B-spline volume을 사용하려고 한다. 또한 연기 외에 불이나 물과 같은 유체의 제어에도 제어 manifold를 적용할 수 있을 것이다.

감사의 글

본 연구는 정보통신부 및 정보통신연구진흥원의 대학 IT연구센터 육성 지원 사업의 연구 결과로 수행되었습니다.

[참고논문]

1. Aris, R. 1989. *Vectors, Tensors and the Basic Equations of Fluid Mechanics.* Dover, New York, 1989.
2. Catmull, E., and Clark, J. 1978. Recursively Generated B-Spline Surfaces On Arbitary Topological Meshes. *Computer Aided Design* 10(6), 350-355, 1978.

3. Chen, J. X., Lobo, N. V., Hughes, C. E., and Moshell, J. M. Real-time Fluid Simulation in a Dynamic Virtual Environment. *IEEE Computer Graphics and Applications*, 17(3), 52-61, 1997.
4. Enright, D., Marchner, S., and Fedkiw, R. Animation and Rendering of Complex Water Surfaces. *ACM Computer Graphics (Proc. Of SIGGRAPH '02)*, 736-744, 2002.
5. Fattal, R., and Lischinski, D., Target-driven smoke animation. *ACM Computer Graphics (Proc. Of SIGGRAPH '04)*, 23(3), 441-448, 2004.
6. Fedkiw,R., Stam, J., And Jensen, H. Visual Simulation of Smoke. *ACM Computer Graphics (Proc. Of SIGGRAPH '01)*, 15-22, 2001.
7. Foster, N., and Fedkiw, R. Practical Animation of Liquids. Graphical Models and Image Processing 58(5), 471-483, 2001.
8. Foster, N., and Metaxas, D. Modeling the Motion of a Hot, Turbulent Gas. *ACM Computer Graphics (Proc. Of SIGGRAPH '97)*, 181-188, 1997.
9. Hong, J.-M., and Kim, C.-H. Interactive Control of Fluid Animation, *Computer Animation and Virtual Worlds*, 15(3-4), 147-157, 2004.
10. Kass, M., and Miller, G. Rapid, stable fluid dynamics for computer graphics, *ACM Computer Graphics (Proc. Of SIGGRAPH '90)*, 49-57, 1990.
11. McNamara, A., Treuille, A., Popovic, Z., and Stam, J. Fluid control using the adjoint method. *ACM Computer Graphics (Proc. Of SIGGRAPH '04)*, 23(3), 449-456, 2004.
12. Stam, J. Exact Evaluation of Catmull-Clark Subdivision Surfaces at Arbitrary Parameter Values. *ACM Computer Graphics (Proc. Of SIGGRAPH '98)*, 395-404, 1998.
13. Stam, J. Stable Fluids. *ACM Computer Graphics (Proc. Of SIGGRAPH '99)*, 121-128, 1999.
14. Stam, J. Flows on Surfaces of Arbitrary Topology. *ACM Computer Graphics (Proc. Of SIGGRAPH '03)*, 22(3), 724-731, 2003.
15. Treuille, A., McNamara, A., Popovic, Z., and Stam, J. KeyFrame Control of Smoke Simulations. *ACM Computer Graphics (Proc. Of SIGGRAPH '03)*, 22(3), 716-723, 2003.