Real-Time Ebru Art Process Based on Particle-Based CFD with Parametric Curves

20119 최재열

Contents

Abstract

본 연구에서는 미술 분야에서 유명한 고전 기법인 마블링 아트를 실시간으로 상호작용 가능한 프로세스로 구현하기 위하여 직접적인 유체역학 시뮬레이션을 제작하였고, 외부의 물체와 유체가 맞닿은 지점에서 일어나는 표면 상의 움직임을 예측하였다. 유체 특유의 성질 및 시뮬레이션 오차가 무늬 형성을 방해하지 않도록 잉크 포인트를 별개로 정의한 후, 매개변수 곡선을 활용하여 이들 사이의 부드럽고 예리한 보간법과 움직임을 제시하고 구현하였다.

1 Introduction

1.1 연구 동기

마블링 아트란, 얕은 물로 채워진 수조에 물 위에 뜨는 지용성 잉크를 떨어트려서 만드는 예술 작품이다. 표면을 긋고 도중에 다른 잉크를 새로 떨어트리면서 그림을 변형하고, 결과적으로 복잡하고 화려한 무늬를 종이에 찍어내게 된다. 오늘날 디지털 페인팅이 점차 상용화 되면서 수채화, 유화 등 다양한 기법으로 만들어지는 작품을 시뮬레이션을 통해 간편하고 사실적으로 표현할 수 있게 되었다. 하지만 마블링 아트에 관련된 시뮬레이션은 아직 표현력이나 조작에 있어서 한계를 보이는 점들이 분명히 있었고, 입자 기반 유체역학을 기반으로 하는 보다 개선된 시뮬레이션을 제시해 보고자 한다.

1.2 연구 목적

본 연구에서는 유체역학 시뮬레이션과 매개변수 곡선을 기반으로 실시간 상호작용과 날카로운 곡선의 표현을 동시에 만족할 수 있는 마블링 아트 프로세스를 제안하고자 한다. 특히, 외부에서 일어나는 물리적인 상호작용을 적용하기 간편한 Smoothed Particle Hydrodynamics(SPH)를 통해 외부 조작과 유체 시뮬레이션을 융합할 것이다. 유체의 표면 상에서 마블링 잉크 방울의 변형 과정을 sparse한 점들의 움직임과 베지어 보간법으로 표현하여 마블링 아트를 유체 시뮬레이션에 덧붙여서 자유자재로 시뮬레이션 가능한 형태로 바꿀 것이다.

2 Theoretical Background

2.1 Smoothed Particle Hydrodynamics

Smoothed Particle Hydrodynamics(SPH)는 연속한 유체 흐름을 작은 입자들의 움직임으로 근사하는 방법으로, 각 입자에 적용되는 힘과 속도를 독립적으로 계산하여 다음 프레임을 생성한다. SPH에서는 연속한 유체의 흐름을 입자계로 정밀하게 이산화하기 위한 방법으로 커널 함수를 사용한다. 이는 디랙 델타 함수를 대체하는 함수로, $W(r,h)$을 만족하는 방사형 함수이다. 주어진 영역에서 정의된 스칼라 함수에 대한 항등식을 커널을 사용하여 다음과 같이 이산화할 수 있다.

$$something \tag{1}$$

비슷한 방식으로 스칼라장의 기울기나 벡터장의 발산의 근사식도 얻어낼 수 있다.

$$something \tag{2}$$

이제 위의 원리를 바탕으로 유체 입자의 운동을 해결할 수 있다. 비압축성 유체의 운동방정식은 압력, 점성에 의한 힘과 외력으로 이루어진다.

$$something \tag{3}$$

SPH 시뮬레이션에서 가장 먼저 이루어지는 것은 밀도와 압력의 계산이다. 밀도는 다음과 같이 이산화된 식을 통해 간단하게 구할 수 있고, 압력은 밀도에 비례하는 함수를 사용하며 보편적으로 두 가지 식 중 하나를 적용하곤 한다.

$$something \tag{4}$$

이제 힘을 계산할 차례이다. 사람마다 같은 항을 이산적으로 유도하는 접근이 다양하므로 계산 속도와 정확도 사이의 절충안으로 다양한 대안을 제시한다. 압력에 의한 힘은 대체적으로 (5)를 사용하여 계산한다. 점성은 기본적으로 인접한 유체끼리 속도를 비슷하게 맞추려는 성질을 나타내는데, 점성을 직접 계산하는 여러 방법이 있지만 점성이 크지 않은 유체에서는 점성의 의도를 살리면서 빠르고 안정적인 구현이 가능한 XSPH를 많이 사용한다. XSPH는 (6)과 같이 유체의 속도장을

부드럽게 만들어주어 점성 효과를 인위적으로 형성해준다.

$$something \tag{5}$$

$$something \tag{6}$$

2.2 Mathematical Marbling

마블링 아트에서 이루어지는 기본적인 조작은 바로 선(stroke)이다. 예리한 물체로 유체의 표면에 자유롭게 선을 긋는 방식으로 진행된다. 이 상호작용을 시뮬레이션에 구현하려면 인위적인 조작이 유체의 움직임이 어떤 영향을 주는지 모델링 해야한다. 'Oseen Flow in Paint Marbling'에서는 이 현상을 구현하기 위해 위치 벡터장을 변환하는 함수를 제시한다. 접촉점을 원점으로 하는 극좌표계에서 물리적인 접촉에 의해 발생하는 속도장 F를 정의했을 때, 이는 비압축성 유체에서 다음과 같은 조건을 만족한다.

$$something \tag{7}$$

접촉점에서 물체를 움직이는 속도가 원점에서 F의 크기를 결정할 것이다. 편의상 접촉점이 x축과 나란하게 속력 U로 움직이고 있다고 생각하면 F의 직교분해 $F = something$은 원점에서 다음 조건을 만족하게 된다.

$$something \tag{8}$$

해당 조건들을 모두 만족하는 해는 다음과 같이 삼각함수와 지수함수의 조합으로 나타낼 수 있다. 이때 L은 지수의 차원을 상수로 맞추어 주기 위한 상수로, $L =$으로 정의하였다.

$$something \tag{9}$$

F를 접촉점을 원점으로 하는 카테시안 좌표로 되돌릴 수 있다.

$$something \tag{10}$$

이제 해당 함수를 변형하여, 한 프레임 동안 접촉점이 B에서 E로 이동하였을 때 임의의 좌표 P에 대한 mapping Q를 다음과 같이 일반화할 수 있다.

$$something \tag{11}$$

$$something \tag{12}$$

2.3 Bezier Curve

베지어 곡선은 컴퓨터 그래픽스 및 응용 분야에서 사용되는 parametric curve의 일종이다. 양 끝점과 몇 개의 제어점으로 정의할 수 있고, 이들을 매개변수 t에 대한 다항식으로 보간하여 두 점을 부드러운 곡선으로 이어주게 된다. 일반적으로 3차 다항식을 쓰며, 점 P_0, P_1, P_2, P_3이 순서대로 주어질 때 P_0와 P_3을 잇는 다항식을 다음과 같이 정의할 수 있다.

$$something \tag{13}$$

베지어 곡선의 t에 대한 도함수와 이계도함수는 다음과 같다.

$$something \tag{14}$$

2.4 Tridiagonal Matrix Algorithm

Tridiagonal 행렬의 정의와 이와 관련된 두 개의 알고리즘 소개(Thomas Algorithm, Sherman-Morrison Formula)

3 연구 과정

3.1 마블링 아트를 위한 유체 구현

본 연구에서는 물감의 표현 및 물리적인 조작과 관련된 모든 것이 이루어지는 유체의 인터페이스를 SPH, 특히 XSPH를 통해 구현하였다. 시뮬레이션에는 총 N개의 유체 입자가 존재하며, 매 프레임마다 가상의 시간 간격 $\Delta T = T_n - T_{n-1}$을 두고 갱신한다. 각 입자에는 입자의 2차원 직교좌표 기준 위치 x_i과 속도 v_i, 밀도 ρ_i와 압력 P_i가 할당된다. 이들은 모두 시간 T에 대한 함수이다.

다음은 각 Particle을 SPH에 기반하여 갱신하는 알고리즘의 pseudocode이다.

Algorithm 1 Fluid Simulation Based on SPH

```
something
```

이때, *something*은 실험 외부에서 임의로 결정해줄 수 있는 상수이다. (각 상수에 대한 구체적인 설명). Figure은 *something* 조건 하에서 SPH 시뮬레이션을

진행하였을 때 일어나는 변화이다.

여기에 Figure이 들어가게 됨

하지만 해당 시뮬레이션에는 점성에 의한 효과가 전혀 들어가지 않기 때문에 유체가 쉬지 않고 공간 상에서 흐르게 된다. 이렇게 되면 실제 아트를 만드는 과정에서 물감의 형태를 유지하지 못할 것이다. 그러므로 XSPH, 즉 주변 입자의 흐름과 속도를 비슷하게 조정하는 과정을 추가하여 보다 정적인 유체 인터페이스를 표현한다. 이것이 실제로 큰 점성을 가진 유체는 아니지만, 거의 모든 상호작용이 유체 내부가 아닌 표면장력이 존재하는 표면에서 이루어진다는 점으로 해당 성질을 정당화할 수 있다. 다음은 이를 반영하여 수정한 알고리즘의 pseudocode이다.

Algorithm 2 Fixed Fluid Simulation Based on XSPH

```
something
```

이때, ϵ은 유체의 움직임이 얼마나 정적인지를 결정하는 계수이다. Figure은 각각 *something* 조건 하에서 XSPH 시뮬레이션을 진행하였을 때 일어나는 변화이다.

여기에 Figure이 들어가게 됨

3.2 유체의 물리적 조작 및 변형

본 연구에서는 크게 두 가지, stroke와 drop을 시뮬레이션의 주된 조작으로 다루게 된다.

일반적인 방법으로는 stroke에서 요구하는 물리적인 상호작용을 구현하기 까다롭다. 얇은 막대로 유체의 표면에 가하는 stroke는 단순히 특정 입자를 잡고 움직이는 것으로 해결 되는것이 아니기 때문이다. 본 연구에서는 막대가 유체와 상호작용하는 과정을 직접 모델링하지 않고, "Oseen Flow in Mathematical Marbling"에서 제시한 실제 stroke에서 일어나는 변환과 유사한 벡터장을 전체 변환에 추가하는 행위로 표현하였다.

현 문제에서 구현해야 하는 것은 마우스를 클릭한 후 자유롭게 이동하면서 유체에 변화를 주는 것이다. 시뮬레이션에서 반드시 만족해야 할 것은 인터페이스가 끊어지지 않는 자연스러운 변화가 일어나야 한다는 것과, 변형 이후 마우스가 지나간 경로 부근의 영역 이외에는 형태에 큰 변화가 일어나지 않아야 한다는 점이다. 다음은 마우스 포인터의 움직임을 반영한 stroke 표현 알고리즘의 pseudocode이다.

또한 drop, 즉 물감을 떨어트렸을 때 일어나는 현상을 구현해야 한다. 이에 대해서는 이미 퍼져 있는 방울은 사방으로 밀어내고, 자기 자신은 넓게 퍼지려는 현상이 관찰되었다. 본 연구에서는 새로운 위치에 물감을 떨어트렸을 때, 해당 위치 근처에

Algorithm 3 Implementation of Stroke

1: Something
2: something

새로운 유체 입자를 일정 시간 생성하고, 경계에 머물러 있는 입자를 그대로 제거하는 메커니즘을 설계했다. 또한 이는, 경계에 머물러 있는 입자 일부를 드랍이 일어난 위치로 옮기는 더 효율적인 방법이 있음을 발견하였다.

3.3 색의 경계에 대한 베지어 보간법

앞서 구현한 유체 인터페이소로는 마블링 아트를 제대로 재현해 내는 데 한계가 있다. XSPH 모델 특성상 각 입자의 움직인은 부드럽고 정적이고, 물감이 실제와 다르게 유체와 흩어질 가능성이 있다. 얇은 유체 위에 떨어진 잉크는 표면의 입자들과 섞여서 흐름을 차지하는 것이 아니라 표면 위에서 미끄러지며 아래에 있는 유체의 흐름을 따라간다고 생각한다. 서로 다른 물감 방울들이 위에서 겹칠 때 이들을 모두 유체 입자들로 취급하면 처리하기 매우 복잡한 상황이 되고, 크고 작은 오류를 범할 것이다. 그러므로 유체 입자의 움직임과 물감의 형태 변형은 다르게 보아야 한다고 판단했고, 본 연구에서는 물감이 이동하는 영역과 유체의 영역을 다른 두 레이어로 분리하여 생각하였다.

마블링 아트에서 요구하는 사항 중 하나는 각 물감 방울을 렌더링했을때 번짐 없이 깔끔하게 이어지는 곡선이다. 이를 만족시키려면 물감 방울의 테두리가 되는 점들을 정의하고 곡선으로 이어 주어야 한다. 연산 횟수를 효율적으로 절감하기 위하여 본 연구에서는 물감의 입자를 일일이 구현하지 않고, 하나의 긴 테두리를 구성하는 점열을 각각 '색의 경계'로 정의하여 레이어에 추가하여 훨씬 효율적이다. 실시간으로 유체 인터페이스의 입자 움직임이 물감 레이어에서 색의 경계의 움직임을 제어한다.

물감과 유체의 상호작용을 다루기 이전에, 본 연구에서는 색의 경계를 하나의 부드러운 곡선으로 렌더링하는 방법을 제시한다. 기존의 베지어 곡선은 곡선이 지나지 않는 부수적인 점인 두 제어점을 필요로 한다. 하지만 현재 문제에서는 제어점 대신에 둘 이상의 인접한 베지어 곡선이 서로 부드럽게 이어져야 한다는 조건이 추가되었다. 그러므로 각 곡선 Γ_i를 이루는 두 제어점을 미지수 α_i, β_i를 통해 $\Gamma_i(t) = something$로 표현하고 서로 만나는 두 곡선의 매개변수 t에 대한 도함수와 이계도함수가 같다는 식을 사용하여 해를 구할 것이다.

색의 경계에는 환형(Ring Form)과 선형(Linear Form) 두 가지가 존재하는데,

이들의 차이는 경계의 첫 점과 마지막 점의 경계조건에 있다. 경계의 양 끝 점이 하나로 모일 때 매끄럽게 만나야 한다면 환형, 그렇지 않고 첨점(sharp point)의 형태로 미분 불가능하게 만나면 선형 경계가 된다. 이들은 연립방정식의 경계 조건에 영향을 주어, 해결 방법에 다소 큰 차이를 주게 된다. 환형 경계는 주로 drop 직후의 기본 형태에서, 선형은 stroke가 생겨 형태가 뾰족하게 왜곡될 때 발생하게 된다. 임의의 경계에서 어느 한 지점의 각의 크기가 60도보다 작으면 그 부분이 sharp point로 끊어지게 된다. 하나의 물감 방울의 테두리가 두 개 이상의 선형 경계로 이루어져 있을 수 있다.

α_i, β_i에 대한 연립방정식을 세우는 과정은 다음과 같다. (something). 이렇게 선형 경계의 경우 Tridiagonal Matrix 형태를 만족하므로 Thomas 알고리즘을 사용하여 간단한 소거법으로 해결할 수 있다. 환형 경계의 경우, Sherman-Morrison Formula을 활용하여 행렬을 Tridiagonal한 것과 남은 자투리 행렬로 나누어서 문제를 해결할 수 있다. (어떻게 하는지에 대한 자세한 설명). 다음 그림은 환형 경계와 선형 경계로 이루어진 하나의 물감 방울에 대한 예시이다.

우선, 새로운 drop이 일어나면 해당 위치를 중심으로 하는 작은 반경의 정10 20각형의 환형 경계를 생성한다. 3.2에서 제시한 메커니즘에 따라 드랍이 막 일어난 동안에는 물감 방울의 크기가 서서히 커질 것이다. 또한 매개변수 t에 대한 곡선을 모든 색의 경계에 대해 구하였다면 물감을 먼저 드랍한 순서대로 내부를 칠하면 된다. 나중에 떨어진 물감이 대개 먼저 떨어진 물감 층을 덮을 것이다.

3.4 유체와 물감의 상호작용

색의 경계의 움직임은 전적으로 유체 인터페이스의 변화에 영향을 받게 된다. 본 연구에서 색의 경계의 각 점은 SPH에서 사용한 kernel 함수를 "유체의 평균 속도"를 구하는 데 적용하여 해당 지점의 속도로 활용한다. 단, smoothing distance(h)를 유체 인터페이스에서 사용한 값과 다르게 사용할 수 있고, 구한 속도에 1보다 큰 상수를 곱하여 해당 점의 속도로 사용하면 비교적 실제에 가까운 물감 방울의 형태 변환이 드러나게 된다. 비로소 유체 인터페이스를 따라서 움직이는 물감 방울이 만들어졌다.

색의 경계의 각 점끼리는 렌더링을 제외하고 서로 영향을 주지 않는다. 단, 경계에서 인접한 점 사이의 거리가 특정 길이 이상일 경우 복잡하게 변하는 모양의 처리가 힘들어지므로 해당 위치의 베지어 곡선에서 $t = 1/3, 2/3$ 지점에 새로운 점을 추가하여 균형을 맞추게 된다. 반대로 두 인접한 점이 너무 가까워지게 되면 해당 위치에서 곡선이 매끄럽게 나오지 않을 수 있기 때문에 제거가 일어나기도 한다.

4 Results

여기에는 여러 조건에서 실험한 시뮬레이션을 자유롭게 사진으로 올리도록 하자. 최종 결과 이외의 단계적인 결과들은 다 연구 과정에 하나씩 집어넣어야 보는 사람이 더 이해가 잘 됨. 또한, 실험 전에 설정하는 상수가 있다면 어떤 값을 사용했는지 제시하여 보는 사람이 믿을 수 있는 논문을 만들자.

5 Conclusion

여기에는 결론을 적자

6 References

https://en.wikipedia.org/wiki/Bezier$_{C}urve$

https://arxiv.org/pdf/1702.02106.pdf

https://arxiv.org/pdf/astro-ph/0204118.pdf

https://arxiv.org/pdf/1007.1245.pdf