SOFT BODY DYNAMICS

Homepage: https://sites.google.com/site/doc4code/

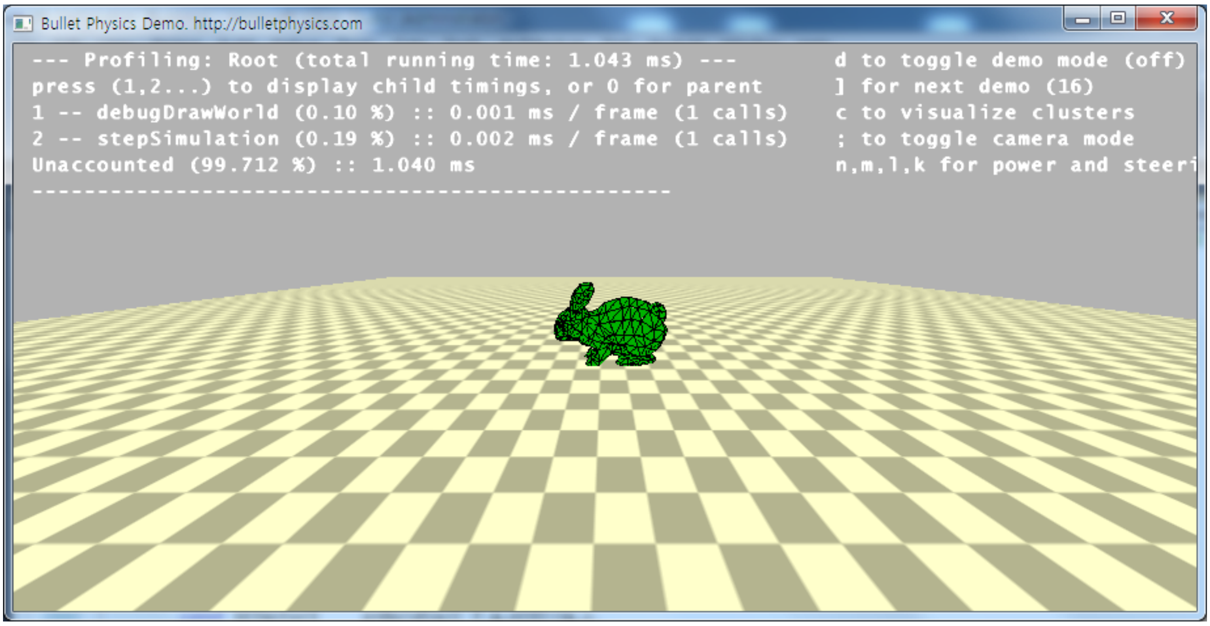
Email: goldpotion@outlook.com

2012/01/13

bullet 엔진의 App_SoftDemo 를 분석해보면 soft body dynamics 의 구현 방법을 알 수 있다. 이 문서에서는 그 구현방법에 대해 간략히 설명하겠다. 어차피, 이 문서를 읽을 수 있는 수준의 사람은 소스 분석도 잘 할 테니까, 자세한 설명은 피하겠다. 백 번 글로 읽는 것보다 한 번 소스를 분석하는 게 나을 것이다. 이 문서에서는 분석하다가 어려움을 느꼈던 부분을 중점적으로 설명하겠다.

VERTEX 방식

소스의 Init_Bunny()를 분석해 보면, soft body 를 점들의 집합으로 구현하고 있다. 이 점들은 물체의 vertex 해당하는데, 각 vertex 들은 가상의 스프링으로 연결되어 있어서 원래 형태를 유지하도록 되어있다.



POSITION BASED STRETCHING

소스의 btSoftBody::PSolve_Links()를 보면, 가상의 스프링이 vertex 간에 원래 길이를 유지하도록 하는 것에 대한 구현 방법을 알 수 있다.

i 번째 vertex 의 위치를 $\mathbf{x}_i(t)$라고 하고, 스프링이 stretching 을 적용하기 이전의 i 번째 vertex 의 위치 $\mathbf{p}_i$라고 하자.

i 번째 vertex 와 j 번째 vertex 의 초기 거리는 $d = |\mathbf{x}_i(0) - \mathbf{x}_j(0)|$로 표기하자.

또, 스프링의 원상 회복력 계수(linear stiffness coefficient)를 $\alpha \in [0\ 1]$로 표기하자.

그러면, i 번째 vertex 의 위치 증가분은 다음 공식에 의해 구할 수 있다.

$$\Delta\mathbf{p}_i = -\alpha\frac{\mathbf{m}_i{}^{-1}}{\mathbf{m}_i{}^{-1} + \mathbf{m}_j{}^{-1}}(\mathbf{p}_j - \mathbf{p}_i)\left(d^2 - |\mathbf{p}_i - \mathbf{p}_j|^2\right)/\left(d^2 + |\mathbf{p}_i - \mathbf{p}_j|^2\right)$$

$$\approx -\alpha\frac{\mathbf{m}_i{}^{-1}}{\mathbf{m}_i{}^{-1} + \mathbf{m}_j{}^{-1}}(|\mathbf{p}_i - \mathbf{p}_j| - d)\frac{\mathbf{p}_i - \mathbf{p}_j}{|\mathbf{p}_i - \mathbf{p}_j|}$$

마찬가지로 j 번째 vertex 의 위치 증가분은

$$\Delta\mathbf{p}_j = \alpha\frac{\mathbf{m}_j{}^{-1}}{\mathbf{m}_i{}^{-1} + \mathbf{m}_j{}^{-1}}(\mathbf{p}_j - \mathbf{p}_i)\left(d^2 - |\mathbf{p}_i - \mathbf{p}_j|^2\right)/\left(d^2 + |\mathbf{p}_i - \mathbf{p}_j|^2\right)$$

$$\approx \alpha \frac{\mathbf{m}_j^{-1}}{\mathbf{m}_i^{-1} + \mathbf{m}_j^{-1}} \left(\left| \mathbf{p}_i - \mathbf{p}_j \right| - d \right) \frac{\mathbf{p}_i - \mathbf{p}_j}{\left| \mathbf{p}_i - \mathbf{p}_j \right|}$$

가 된다.

RIGID BODY 와 SOFT BODY 의 충돌

소스의 PSolve_RContacts()에서 rigid body 와 soft body 간의 충돌을 처리한다. 이 때, rigid body 와 soft body 의 각 vertex 간에 충돌로 나누어 처리한다. rigid body A 와 soft body 의 vertex B 간에 충돌했을 때 어떻게 처리하는지 알아보자.

rigid body A 의 무게 중심 위치 $\mathbf{x_A}$라고 하고, soft body vertex B 의 위치 $\mathbf{x_B}$라고 하자.

A 에 가해진 **impulse**(충격량) $\mathbf{F}\Delta t$ 하면, B 에 가해진 **impulse**(충격량)는 -$\mathbf{F}\Delta t$ 가 된다.

A 의 속도 증가분은 다음과 같다.

$$\mathbf{v_A}(t + \Delta t) - \mathbf{v_A}(t) = \dot{\mathbf{v}}_\mathbf{A}(t)\Delta t = m_A{}^{-1}\mathbf{F}\Delta t$$

마찬가지로 B 의 속도 증가분은

$$\mathbf{v_B}(t + \Delta t) - \mathbf{v_B}(t) = \dot{\mathbf{v}}_\mathbf{B}(t)\Delta t = -m_B{}^{-1}\mathbf{F}\Delta t$$

이다.

충돌 지점이 **P** 일 때, A 의 무게중심에 대한 충돌 점의 상대위치를 $\mathbf{r} = \mathbf{P} - \mathbf{x_A}(t)$ 로 표기하면,

A 의 각속도 증가분은

$$\boldsymbol{\omega}_\mathbf{A}(t + \Delta t) - \boldsymbol{\omega}_\mathbf{A}(t) = \dot{\boldsymbol{\omega}}_\mathbf{A}(t)\Delta t \approx I_A(t)^{-1}(\mathbf{r} \times \mathbf{F})\Delta t$$

로 계산된다.

충돌로 인한 A 의 충돌 점의 추가 위치 증가분은

$$\mathbf{x_A}(t + \Delta t) - (\mathbf{x_A}(t) + \mathbf{v_A}(t)\Delta t + \boldsymbol{\omega}_\mathbf{A}(t) \times \mathbf{r}\Delta t)$$

$$= \left(\mathbf{v_A}(t + \Delta t) - \mathbf{v_A}(t)\right)\Delta t + \left(\boldsymbol{\omega}_\mathbf{A}(t + \Delta t) - \boldsymbol{\omega}_\mathbf{A}(t)\right) \times \mathbf{r}\Delta t$$

$$= m_A{}^{-1}\mathbf{F}\Delta t^2 + (I_A(t)^{-1}(\mathbf{r} \times \mathbf{F})\Delta t) \times \mathbf{r}\Delta t$$

$$= m_A{}^{-1}\mathbf{F}\Delta t^2 - \mathrm{Skew}(\mathbf{r})(I_A(t)^{-1}\mathrm{Skew}(\mathbf{r})\mathbf{F})\Delta t^2$$

$$= \left(m_A{}^{-1}E - \mathrm{Skew}(\mathbf{r})I_A(t)^{-1}\mathrm{Skew}(\mathbf{r})\right)\mathbf{F}\Delta t^2$$

가 된다.

충돌로 인한 B 의 추가 위치 증가분은

$$\mathbf{x_B}(t+\Delta t) - (\mathbf{x_B}(t) + \mathbf{v_B}(t)\Delta t) = \left(\mathbf{v_B}(t+\Delta t) - \mathbf{v_B}(t)\right)\Delta t = -m_B{}^{-1}\mathbf{F}\Delta t^2$$

의해 계산한다.

충돌을 해소하기 위해 필요한 B 에 대한 A 의 충돌 점의 상대 위치 증가분이 $\mathbf{v_{Rel}}$이라고 한다면

$$\mathbf{v_{Rel}} = (m_A{}^{-1}E - \mathrm{Skew}(\mathbf{r})I_A(t)^{-1}\mathrm{Skew}(\mathbf{r}) + m_B{}^{-1}E)\mathbf{F}\Delta t^2$$

를 만족해야 한다.

그러므로, 구하는 **impulse**(충격량)은

$$\mathbf{F}\Delta t = (m_A{}^{-1}E - \mathrm{Skew}(\mathbf{r})I_A(t)^{-1}\mathrm{Skew}(\mathbf{r}) + m_B{}^{-1}E)^{-1}\mathbf{v_{Rel}}$$

에 의해 구할 수 있다.

이렇게 구한 충격량을 물체 A 와 vertex B 에 적용하면 된다.

그런데, vertex 방식에서는 soft body 의 face 와 rigid body 의 vertex 간에 충돌을 처리하지 않는 단점이 있다.

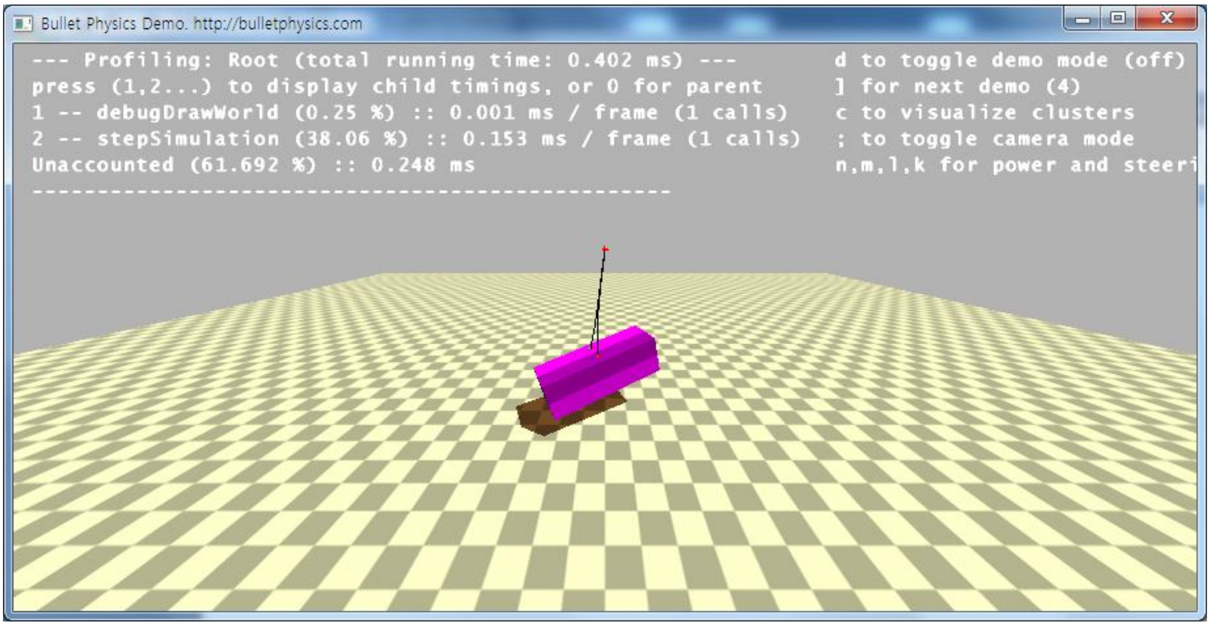
SOFT BODY 와 SOFT BODY 의 충돌

소스의 PSolve_SContacts()에서 soft body 와 soft body 간의 충돌을 처리한다.

soft Body A 와 soft Body B 의 충돌을 처리할 때는 두 부류로 나누어 처리한다. 하나는 물체 A 의 각 vertex 와 물체 B 의 각 face 간에 충돌을 처리한다. 또 하나는 물체 A 의 각 face 와 물체 B 의 각 vertex 간의 충돌을 처리한다.

줄(ROPE)

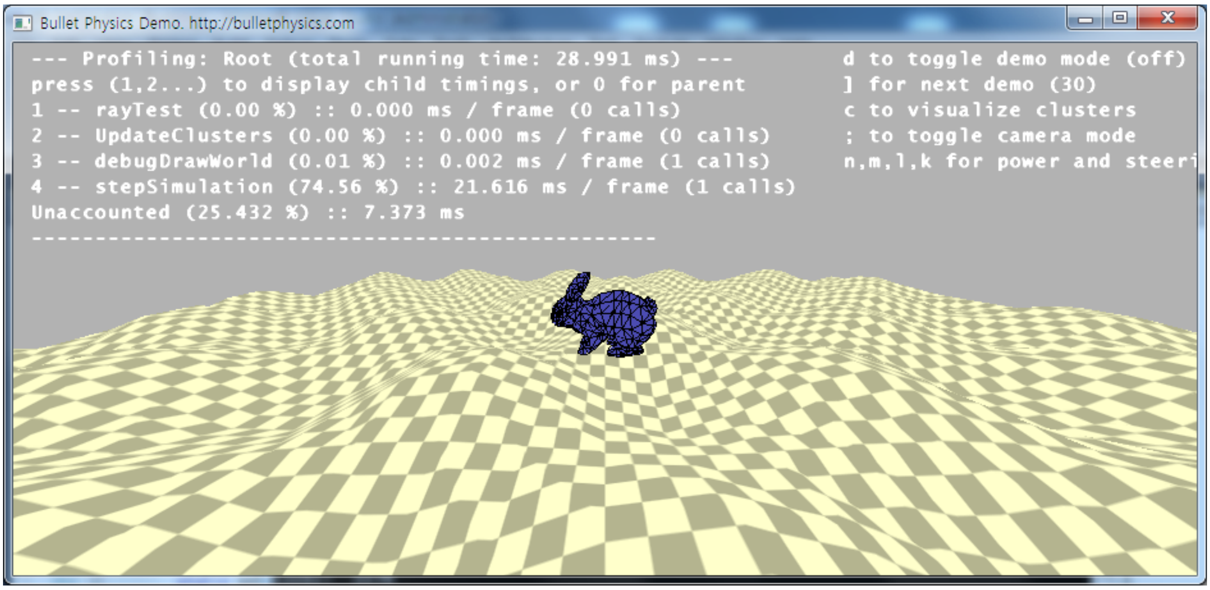
소스의 Init_RopeAttach()를 보면, 로프를 구현하는 방법을 알 수 있다. 로프는 여러 개의 점으로 이루어지는데, 각 점을 가상의 스프링으로 연결한다. 그리고, rope 끝에 box 형태의 rigid body 를 매달았는데, 이 때 rope 와 box 는 anchor 를 이용해 연결한다.



소스의 btSoftBody::PSolve_Anchor()를 보면 anchor 를 어떻게 구현하는지 알 수 있다. 앞에서 설명한 rigid body 와의 충돌 처리방법과 비슷하다. Rope 의 끝 점 vertex 을 box 의 지정된 위치로 이동시키키기 위한 $\mathbf{v}_{\mathbf{Rel}}$를 구해서, 그 때의 **impulse** 를 계산한다. 그 **impulse** 을 줄의 끝점 vertex 와 box 에 적용시키면 된다.

CLUSTER 방식

소스의 Init_TetraBunny ()를 분석해 보면, cluster 방식으로 soft body 를 구현하는데 vertex 방식과 거의 비슷하게 처리한다. 여기서도 cluster 의 vertex 들이 가상의 스프링으로 연결되어 있다. 다른 점은 충돌을 처리하는 할 때, vertex 방식은 점인 vertex 단위로 처리하지만, 이 방식은 사면체(tetrahedron)인 cluster 단위로 처리한다.



소스에서 btSoftColliders::ClusterBase::SolveContact()와 btSoftBody::CJoint::Solve()를 참조하면 된다.

Vertex 방식에서는 점과 rigid body 간의 충돌을 처리하지만, cluster 방식에서는 cluster 와 rigid body 간의 충돌을 처리하는 점을 제외하고는 거의 유사하다. Cluster 방식에서도 vertex 방식에서처럼 충격량을 계산한 후, 그 충격량을 cluster 와 상대 물체에 적용한다.

ROTATION MATRIX 계산

Cluster 단위로 처리를 하기 위해서는 각 cluster 마다, mass, inertia tensor, center of mass 등을 계산해야 한다. 또, 좌표계가 어떻게 바뀌는지 알아야 하므로, cluster 의 회전 행렬을 구해야 한다. cluster 에 속한 vertex 의 위치를 가지고, 회전 행렬을 구하는 방법을 알아보자.

Cluster 에 있는 vertex 의 개수가 n 이라고 하고,

i 번째 vertex 의 위치를 $\mathbf{x}_i(t)$라고 하면,

cluster 의 무게 중심은 $\mathbf{x}(t) = \sum_i^n m_i \mathbf{x}_i(t) / \sum_i^n m_i$가 된다.

무게중심에 대한 i 번째 vertex 의 상대 위치를 $\mathbf{a}_i = \mathbf{x}_i(t) - \mathbf{x}(t)$로 표기하고,

초기 상태에서의 상대 위치를 $\mathbf{b}_i = \mathbf{x}_i(0) - \mathbf{x}(0)$로 표기하자.

그리고, $3 \times n$ matrix $A = [\mathbf{a}_1 \quad \mathbf{a}_2 \quad \cdots \quad \mathbf{a}_n]$ 라고 하고, $3 \times n$ matrix $B = [\mathbf{b}_1 \quad \mathbf{b}_2 \quad \cdots \quad \mathbf{b}_n]$ 라고 하자.

3×3 orthogonal matrix U 에 대해 $\sum_{i=1}^{n}|\mathbf{a}_i - U\mathbf{b}_i|^2$ 의 값이 최소가 되는 U 가 우리가 구하는 회전행렬이다.

이 때, AB^T가 **orthogonal matrix** Q 와 **positive semidefinite symmetric matrix** S 로 **polar decomposition** 된다고 하면,

$$AB^T = QS$$

$$\arg\min_U \sum_{i=1}^{n}|\mathbf{a}_i - U\mathbf{b}_i|^2 = \arg\min_U \|A - UB\|_F{}^2 = Q$$

이것에 대한 증명은 조금 후에 하겠다. 어쨌든, Q 가 우리가 구하는 회전행렬이다.

TRACE

증명을 하기 전에 trace 라는 것에 대해 공부해 보자.

$m \times n$ matrix $A = \begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1n} \\ A_{21} & A_{22} & \cdots & A_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ A_{mn} & A_{mn} & \cdots & A_{mn} \end{bmatrix}$에 대해, A 의 **trace** 를 다음과 같이 정의한다.

m >= n 이면

$$tr(A) = \sum_{i=1}^{n} A_{ii}$$

으로 정의한다.

m <= n 이면

$$tr(A) = \sum_{i=1}^{m} A_{ii}$$

으로 정의한다.

그러면, transpose 에 대한 값은

$$tr(A^T) = tr(A)$$

로 구할 수 있다.

또, $m \times n$ matrix A 와 $m \times n$ matrix B 에 대해,

$$\mathrm{tr}(A + B) = \mathrm{tr}(A) + \mathrm{tr}(A)$$

라는 식이 성립한다.

$m \times n$ matrix $A = \begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1n} \\ A_{21} & A_{22} & \cdots & A_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ A_{m1} & A_{m2} & \cdots & A_{mn} \end{bmatrix}$, $n \times m$ matrix $B = \begin{bmatrix} B_{11} & B_{12} & \cdots & B_{1m} \\ B_{21} & B_{22} & \cdots & B_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ B_{n1} & B_{n2} & \cdots & B_{nm} \end{bmatrix}$에 대해서는

$$\mathrm{tr}(AB) = \sum_{i=1}^{m}\sum_{j=1}^{n} A_{ij}B_{ji} = \sum_{j=1}^{n}\sum_{i=1}^{m} B_{ji}A_{ij} = \mathrm{tr}(BA)$$

가 성립한다.

만약, **square matrix** A 가 **eigendecomposition**(**spectral decomposition**)해서 **invertible matrix** U 와 **diagonal matrix** Λ로 분해된다면

$$A = U\Lambda U^{-1}$$

$$\mathrm{tr}(A) = \mathrm{tr}(U\Lambda U^{-1}) = \mathrm{tr}\big(U(\Lambda U^{-1})\big) = \mathrm{tr}\big((\Lambda U^{-1})U\big) = \mathrm{tr}(\Lambda)$$

이 된다.

즉 $\mathrm{tr}(A)$는 **eigenvalues** 의 합과 같다.

$m \times n$ matrix $A = \begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1n} \\ A_{21} & A_{22} & \cdots & A_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ A_{m1} & A_{m2} & \cdots & A_{mn} \end{bmatrix}$라고 할 때

$$\|A\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} {A_{ij}}^2} = \sqrt{\mathrm{tr}(A^T A)}$$

이라는 것도 쉽게 알 수 있다.

증명

자, 이제 AB^T가 **orthogonal matrix** Q 와 **positive semidefinite symmetric matrix** S 로 분해된다고 하면,

$$AB^T = QS$$

$$\arg\min_U \|A - UB\|_F{}^2 = Q$$

라는 것을 증명하자.

Positive semidefinite diagonal matrix $\Sigma = \begin{bmatrix} \Sigma_{11} & 0 & \cdots & 0 \\ 0 & \Sigma_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \Sigma_{nn} \end{bmatrix}$, **orthogonal matrix** $U = \begin{bmatrix} U_{11} & U_{12} & \cdots & U_{1n} \\ U_{21} & U_{22} & \cdots & U_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ U_{n1} & U_{n2} & \cdots & U_{nn} \end{bmatrix}$에 대해

$$tr(U\Sigma) = \sum_{i=1}^{n} U_{ii}\Sigma_{ii} \le \sum_{i=1}^{n} \Sigma_{ii}$$

위에서, Σ_{ii}는 모두 음이 아닌 실수이므로 U_{ii}가 1 일 때 최대값을 가진다. 그러므로

$$\arg\max_U \ tr(U\Sigma) = I$$

이다.

그리고, $m \times n$ matrix A, $m \times n$ matrix B, $m \times m$ **orthogonal matrix** U 에 대해

$$\|A - UB\|_F{}^2 = tr\big((A - UB)^T(A - UB)\big) = tr\big((A^T - B^TU^T)(A - UB)\big)$$
$$= tr(A^TA) - tr(A^TUB) - tr(B^TU^TA) + tr(B^TB)$$
$$= \|A\|_F{}^2 + \|B\|_F{}^2 - 2tr(U^TAB^T)$$

가 된다.

AB^T가 $m \times m$ **orthogonal matrix** Q, $m \times m$ **positive semidefinite symmetric matrix** $W\Sigma W^T$로 **polar decomposition** 된다면,

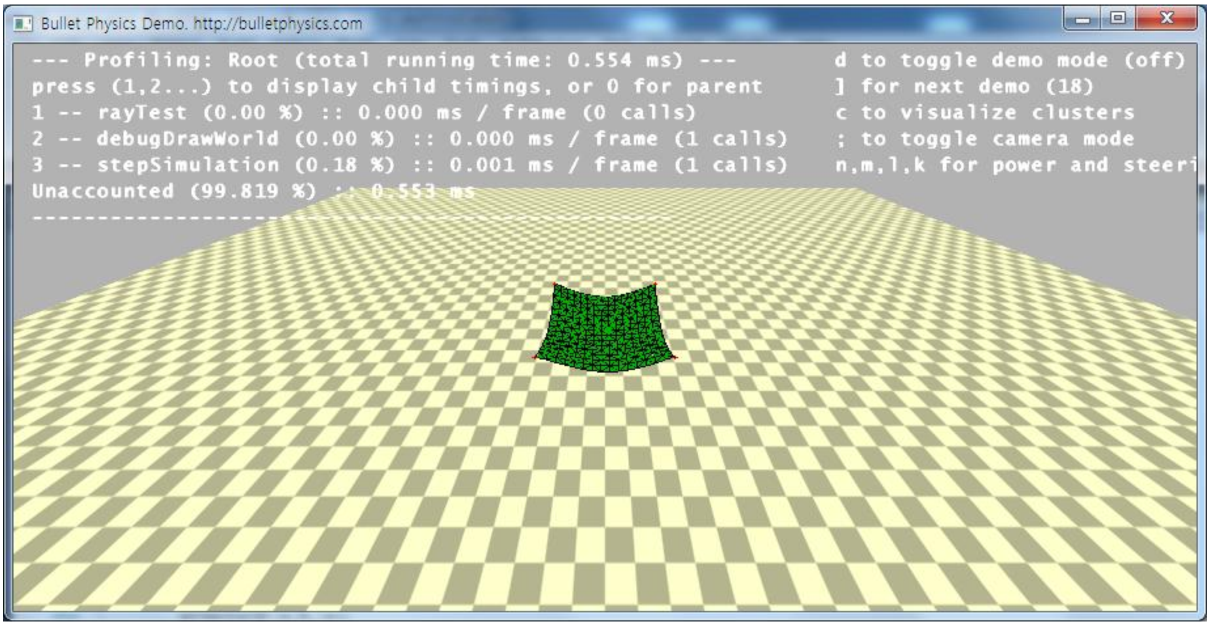
$$\arg\min_U \|A - UB\|_F{}^2 = \arg\max_U tr(U^TAB^T) = \arg\max_U tr\big(U^TQ(W\Sigma W^T)\big)$$
$$= \arg\max_U tr\big((W^TU^TQW)\Sigma\big)$$

위 식에서 W^TU^TQW가 I 일 때 최대값을 가지므로

$$\arg\min_U \|A - UB\|_F{}^2 = Q$$

천(CLOTH)

소스의 Init_Cutting1()을 분석해 보면, 천을 cluster 방식을 이용해 구현한다.
이 때는, 각 삼각형 면(face)이 하나의 cluster가 된다.



참고 문서

[1] Real Time Physics Class Notes http://www.matthiasmueller.info/realtimephysics/index.html

이 문서의 3 장 Mass Spring Systems 과 5 장 Position Based Dynamics 는 이 문서를 이해하는 데 도움이 될 것이다.