ICG Final Report

B10401006 洪愷希, B10502010 王維勤

Introduction

Position based dynamics (PBD) 是一種在圖形領域的新興物理模型框架，由 Matthias Müller （2007）等人提出 [1]。PBD 並非嚴格意義上的 physics-based simulation，而是一種高效率的近似物理模擬方法，因此主要用於 3D 遊戲等要求效能的場合。PBD 的特點是簡單、穩定、高效，並且容易實現。PBD 的應用範圍廣泛，包括布料模擬、軟體模擬、流體模擬等。本報告將介紹 PBD 的基本原理，並實現布料模擬、軟體模擬等應用。
傳統的物理模擬方法是以力作為核心，即牛頓第二定律：

$$\mathbf{f} = m\mathbf{a} \tag{1}$$

不過，這也就意味者對於軟體、布料及流體等物體的模擬，需要解決大量的微分方程，計算量極大。相對地，PBD 則是以位置作為核心，即直接修改位置以滿足約束，而不是計算力。這樣的方法不僅簡化了計算，並且可以更容易地處理各種約束，如距離約束、體積約束等。我們將在下一小節中介紹 PBD 的基本原理。
PBD 並不是單一的模型，而是一種框架，因此不乏有許多研究者提出延伸的應用以及改進，例如 position based fluid (PBF) 即是以此框架為基礎進行流體模擬。而在我們的專案中，亦參考了由 PBD 團隊提出的 xPBD [2]，作為 PBD 的改進版本。

Position Based Dynamics

Notations. 假設我們有 N 個頂點跟 M 個限制條件, 對於點 $i \in [1, \cdots, N]$ 有質量 m_i、位置 $\mathbf{x}_i$ 以及速度 $\mathbf{v}_i$, 且對一 constraint $j \in [1, \cdots, M]$ 我們有

1. A cardinality n_j
2. A function $C_j : \mathbb{R}^{3n_j} \to \mathbb{R}$.
3. A set of indices $\{i_1, \cdots, i_{n_j}\}, i_k \in [1, N]$.
4. A stiffness parameter $k_j \in [0, \cdots, 1]$.
5. A type pf either equality or inequality.

Algorithm Overviews

基於上述之資料以及 time step Δt, 這個物體可由Figure 1 之演算法描述。在 Figure 1的演算法中給定的約束 $C_1, \ldots, C_M$ 在整個模擬過程中固定不變，以下我們對此演算法進行解釋：在第 5 行我們考慮外力作用，例如重力，第 6 行對速度進行 damping。在第 7 行，計算外力影響後的新位置 $\mathbf{p}_i$。在第 9-11 行通過 Gauss–Seidel 投影約束並求解。在第 13 和 14 行，移動頂點位置並更新速度。在第 16 行則根據摩擦和恢復係數修改碰撞頂點的速度。

Constraint Projection

solver 的 input 為 $M + M_{\text{coll}}$ 的 constraints 和每一點的新位置 $\mathbf{p}_1, \ldots, \mathbf{p}_N$。Solver 將修改這些位置，使其滿足所有 constraints。我們可發現，即使是一個簡單的距離約束 $C(\mathbf{p}_1, \mathbf{p}_2) = |\mathbf{p}_1 - \mathbf{p}_2| - d$ 也會產生非線性方程。
為了解決這樣一組方程和不等式，我們使用類似於 Gauss-Seidel 的方法，逐一獨立解決每個約束。

```
forall vertices i
    initialize x_i = x_i^0, v_i = v_i^0, w_i = 1/m_i
endfor
loop
    forall vertices i do v_i ← v_i + Δt w_i f_ext(x_i)
    dampVelocities(v_1, ..., v_N)
    forall vertices i do p_i ← x_i + Δt v_i
    forall vertices i do generateCollisionConstraints(x_i → p_i)
    loop solverIterations times
        projectConstraints(C_1, ..., C_{M+M_coll}, p_1, ..., p_N)
    endloop
    forall vertices i
        v_i ← (p_i − x_i)/Δt
        x_i ← p_i
    endfor
    velocityUpdate(v_1, ..., v_N)
endloop
```

Figure 1: Algorithm

我們藉由移動這些點以滿足該 constraint。過程中需滿足動量以及角動量的守恆。令 Δp_i 為 the displacement of vertex i by the projection. 考慮動量守恆

$$\sum_i m_i \Delta \mathbf{p}_i = 0, \tag{2}$$

考慮角動量守恆

$$\sum_i \mathbf{p}_i \times \Delta \mathbf{p}_i = 0. \tag{3}$$

如果投影違反了以上限制，將會產生 ghost force，如同外力一般拖動和旋轉物體。令 $\mathbf{p} = \left[\mathbf{p}_1^\top \cdots \mathbf{p}_n^\top\right]$。可以發現當 $\Delta \mathbf{p}$ 在 $\nabla_p C(\mathbf{p})$ 的方向且每一點質量皆相同時，動量守恆成立。考慮內部約束，對於 $\mathbf{p}$，我們想求得 correction 以滿足 $C(\mathbf{p} + \Delta \mathbf{p}) = 0$。此關係式可以寫成

$$C(\mathbf{p} + \Delta \mathbf{p}) \approx C(\mathbf{p}) + \nabla_p C(\mathbf{p}) \cdot \Delta \mathbf{p} = 0 \tag{4}$$

我們限制 $\Delta \mathbf{p}$ 在 $\nabla_p C(\mathbf{p})$ 的方向，意即

$$\Delta \mathbf{p} = \lambda \nabla_p C(\mathbf{p}) \tag{5}$$

$$\Delta \mathbf{p} = -\frac{C(\mathbf{p})}{\left|\nabla_p C(\mathbf{p})\right|^2} \nabla_p C(\mathbf{p}). \tag{6}$$

對於 individual point $\mathbf{p}_i$ 的 correction，我們有

$$\Delta \mathbf{p}_i = -s \nabla_{\mathbf{p}_i} C(\mathbf{p}_1, \cdots, \mathbf{p}_n) \tag{7}$$

而對於每一點，scaling factor 皆為

$$s = \frac{C(\mathbf{p}_1, \cdots, \mathbf{p}_n)}{\sum_j \left|\nabla_{p_j} C(\mathbf{p}_1, \cdots, \mathbf{p}_n)\right|^2} \tag{8}$$

對於每一點質量不一定相同之情況，我們令 $w_i = 1/m_i$ 於是

$$\Delta \mathbf{p}_i = -s \frac{n \cdot w_i}{\sum_j w_j} \nabla_{p_i} C(\mathbf{p}_1, \cdots, \mathbf{p}_n), \tag{9}$$

舉例來說，我們考慮下列 constraint

$$C(\mathbf{p}_1, \mathbf{p}_2) = |\mathbf{p}_1 - \mathbf{p}_2| - d \tag{10}$$

其微分為

$$\nabla_{p_1} C(\mathbf{p}_1, \mathbf{p}_2) = n \tag{11}$$
$$\nabla_{p_2} C(\mathbf{p}_1, \mathbf{p}_2) = -n \tag{12}$$

其 final corrections 為

$$\Delta \mathbf{p}_1 = -\frac{w_1}{w_1 + w_2} \left(|\mathbf{p}_1 - \mathbf{p}_2| - d\right) \frac{\mathbf{p}_1 - \mathbf{p}_2}{|\mathbf{p}_1 - \mathbf{p}_2|} \tag{13}$$
$$\Delta \mathbf{p}_2 = -\frac{w_2}{w_1 + w_2} \left(|\mathbf{p}_1 - \mathbf{p}_2| - d\right) \frac{\mathbf{p}_1 - \mathbf{p}_2}{|\mathbf{p}_1 - \mathbf{p}_2|} \tag{14}$$

Soft Body Simulation

現在考慮 soft body simulation。將模型以 mesh 的方式建構，可以將它視為多個 tetrahedral 所組成的 3D 結構。為了維持模型的穩定，針對各個 tetrahedral 以及其 vertex，考慮以下兩個 constraints：distance constraint, volume constraint。

Distance Constraint

針對 distance constraint，考慮 mesh 中任兩點 $\mathbf{x}_1$, $\mathbf{x}_2$ 距離 $l = |\mathbf{x}_1 - \mathbf{x}_2|$ 不變，設定 constraint 為

$$C = l - l_0 \tag{15}$$

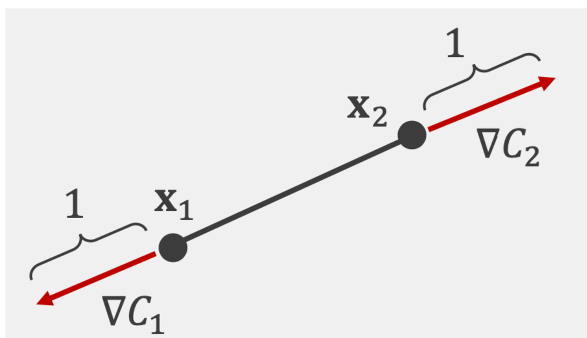

Figure 2: Distance Constraint

而 gradient 即延兩點連線方向，大小為 1

$$\nabla C_1 = \frac{\mathbf{x}_1 - \mathbf{x}_2}{|\mathbf{x}_1 - \mathbf{x}_2|} \tag{16}$$
$$\nabla C_2 = \frac{\mathbf{x}_2 - \mathbf{x}_1}{|\mathbf{x}_2 - \mathbf{x}_1|} \tag{17}$$
$$\lambda = \frac{-(l - l_0)}{w_1 \cdot 1 + w_2 \cdot 1} \tag{18}$$

更新位置為

$$\Delta \mathbf{x}_1 = \lambda w_1 \nabla C_1 = -\frac{w_1}{w_1 + w_2} (l - l_0) \frac{\mathbf{x}_2 - \mathbf{x}_1}{|\mathbf{x}_2 - \mathbf{x}_1|} \tag{19}$$

Volume Constraint

針對 tetrahedral，為保持體積不變，我們設定其 volume constraint 為

$$C = 6(V - V_0) \tag{20}$$

考慮 constraint 的 gradient 方向為體積變化最大方向即垂直底面方向，如Figure 3所示，以底邊外積來計算可以有以下結果

$$\nabla_1 C = (\mathbf{x}_4 - \mathbf{x}_1) \times (\mathbf{x}_3 - \mathbf{x}_2) \tag{21}$$
$$\nabla_2 C = (\mathbf{x}_3 - \mathbf{x}_1) \times (\mathbf{x}_4 - \mathbf{x}_1) \tag{22}$$
$$\nabla_3 C = (\mathbf{x}_4 - \mathbf{x}_1) \times (\mathbf{x}_2 - \mathbf{x}_1) \tag{23}$$
$$\nabla_4 C = (\mathbf{x}_2 - \mathbf{x}_1) \times (\mathbf{x}_3 - \mathbf{x}_1) \tag{24}$$

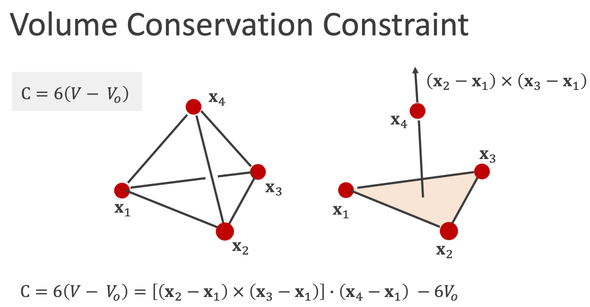

Figure 3: Volume Constraint

$$\lambda = \frac{-6(V - V_0)}{w_1 \left|\nabla C_1\right|^2 + w_2 \left|\nabla C_2\right|^2 + w_3 \left|\nabla C_3\right|^2 + w_4 \left|\nabla C_4\right|^2 + \frac{\alpha}{\Delta t^2}} \tag{25}$$

最終更新位置

$$\Delta \mathbf{x}_i = \lambda w_i \nabla C_i \tag{26}$$

Cloth Simulation

Bending Constraint

對於布料的模擬，我們使用了兩種 constraint：distance constraint 和 bending constraint。Distance constraint 用於保持同一個三角形上的點之間的距離，而 bending constraint 用於保持相鄰三角形的角度。由於 distance constraint 已經在 soft body simulation 中介紹過，這裡我們將重點放在 bending constraint 上。
在 Figure 4 中，兩個相鄰三角形的頂點為 $\mathbf{p_1}, \mathbf{p_2}, \mathbf{p_3}, \mathbf{p_4}$。則兩個面的法向量分別為

$$\mathbf{n_1} = \frac{(\mathbf{p_2} - \mathbf{p_1}) \times (\mathbf{p_3} - \mathbf{p_1})}{|(\mathbf{p_2} - \mathbf{p_1}) \times (\mathbf{p_3} - \mathbf{p_1})|} \tag{27}$$
$$\mathbf{n_2} = \frac{(\mathbf{p_2} - \mathbf{p_1}) \times (\mathbf{p_4} - \mathbf{p_1})}{|(\mathbf{p_2} - \mathbf{p_1}) \times (\mathbf{p_4} - \mathbf{p_1})|} \tag{28}$$

因此對於兩者夾角的 constraint 可表示為

$$C_{bend}(\mathbf{p_1}, \mathbf{p_2}, \mathbf{p_3}, \mathbf{p_4}) = \arccos \mathbf{n_1} \cdot \mathbf{n_2} - \phi_0$$

其中 ϕ_0 為初始角度。

Collision

對於物體之間的碰撞，我們使用 distance constraint 來處理。當兩個質點的距離小於一定值時，我們施加 15 中的 constraint，並且令 l_0 為事先訂好的參數。如此一來在 constraint solver 中即可處理碰撞問題。
若是對每兩個質點都判斷是否碰撞，計算量將會過大。因此我們使用 spatial hashing 的技巧，將空間分成一個個的格子，並且將每個質點放入對應的格子中。這樣一來，我們只需要對每個格子中的質點進行碰撞判斷，大大減少了計算量。

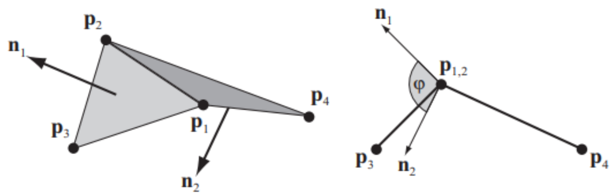

Figure 4: Bending constrint 示意圖

Results

我們以每個 step 有 10 個 substep 進行模擬，畫面中約有 6000 個三角形。

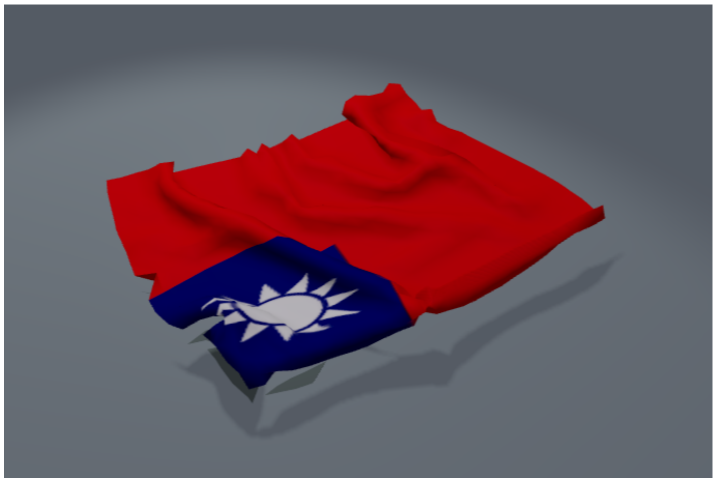

Figure 5: 模擬布料掉落地面，可見有褶皺的現象

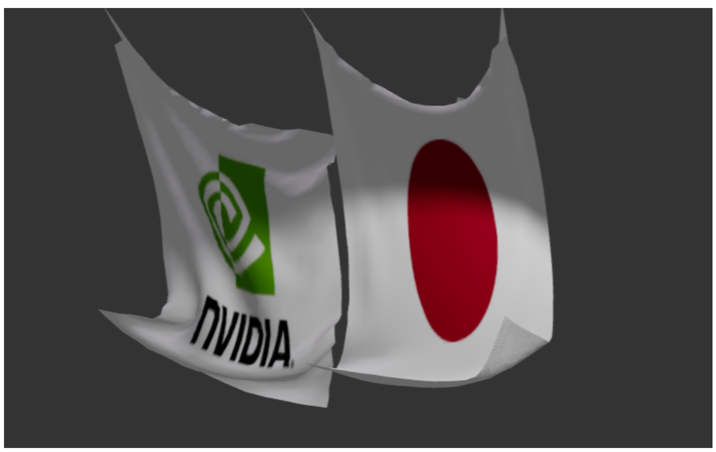

Figure 6: 模擬布料懸掛空中，並加上水平的風力

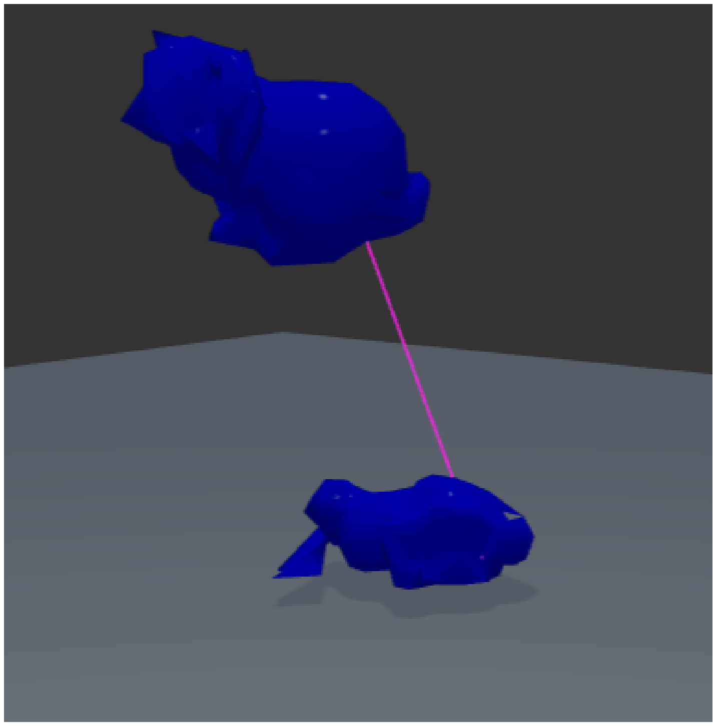

Figure 7: 以兔子模型模擬 soft body。另外在兩個兔子間加上距離的 constraint

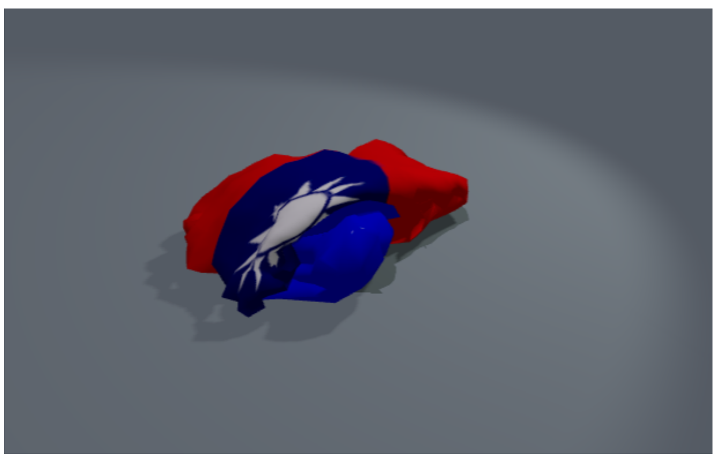

Figure 8: 布料與兔子間的碰撞

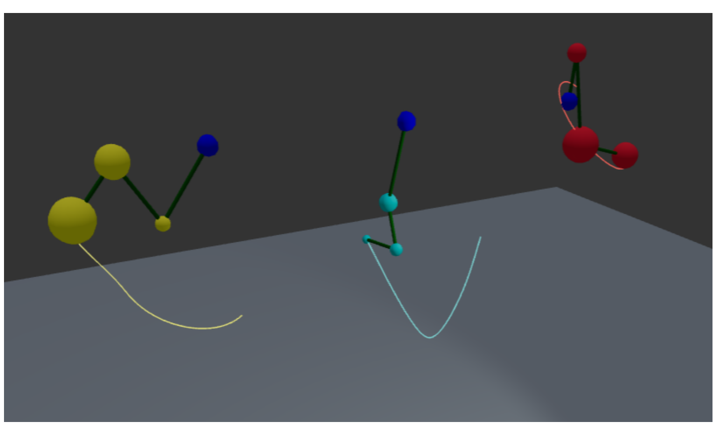

Figure 9: 模擬多重單擺串接，起始畫面

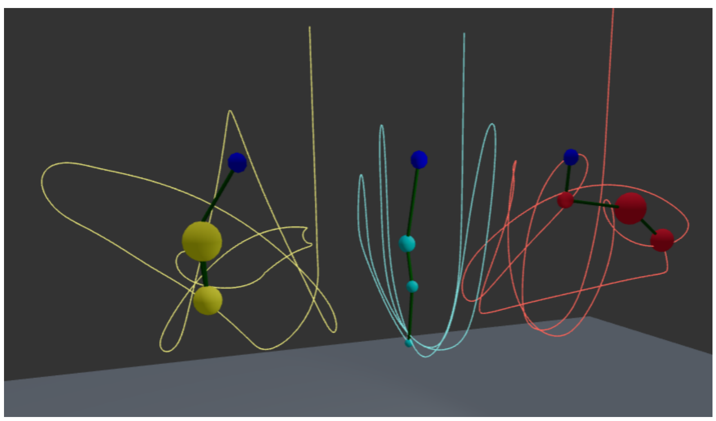

Figure 10: 模擬多重單擺串接，運行數秒後之軌跡

References

[1] Matthias Müller et al. “Position based dynamics”. en. In: *J. Vis. Commun. Image Represent.* 18.2 (Apr. 2007), pp. 109–118.

[2] Miles Macklin, Matthias Müller, and Nuttapong Chentanez. “XPBD”. In: *Proceedings of the 9th International Conference on Motion in Games*. Burlingame California: ACM, Oct. 2016.

[3] Miles Macklin et al. “Unified particle physics for real-time applications”. en. In: *ACM Trans. Graph.* 33.4 (July 2014), pp. 1–12.

[4] Jan Bender, Matthias Müller, and Miles Macklin. *Position-based simulation methods in computer graphics*. 2015.

[5] *Ten Minute Physics*. https://matthias-research.github.io/pages/tenMinutePhysics/index.html. Accessed: 2024-6-9.