Computer Animation Final Project - Snow simulation using moving least square material point method

Group 34: 110550059 劉珆睿, 110550047 巫廷翰

Outline

- Introduction
- Fundamentals
- Implementations
- Results and discussion
- (Extra) Augmented MPM
- Conclusion
- Member contribution
- Reference
- Files and Links

Introduction

- 雪的物理模擬可以應用於多媒體產業或災害防治等領域，因此，我們想要在這次的期末專案中研究如何模擬雪這種材料。
- 專案中，我們將研究如何使用 MPM(Material Point Method) 來模擬雪。MPM 是一種模擬連續介質的方法，我們將研究其方法以及原理。
- 接著，我們會研究 MLS-MPM(Moving Least Square MPM)，MLS-MPM是對於 MPM 方法的一種改進，可以提升其穩定性以及模擬效率，我們將會研究其如何改善 MPM 。
- 最後，我們會實作使用 MPM 和 MLS-MPM 進行的雪模擬。我們將使用 Taichi 程式語言進行模擬，以其提供的 mpm3d_ggui.py 作為基底生成模擬的粒子動畫，並將其輸入至 Houdini 動畫軟體生成表面，以達成雪的效果。我們將會展示我們的模擬結果，以及不同的參數對模擬結果的影響。
- 除了使用 MPM 和 MLS-MPM 進行雪的模擬之外，我們還另外研究如何處理相變材料融化以及凝固的行為，我們實作了一個方法 Augmented MPM，其可以針對溫度以及相變進行處理，達到模擬的效果。但目前數值系統仍未穩定。我們將會說明 Augmented MPM的原理以及方法。

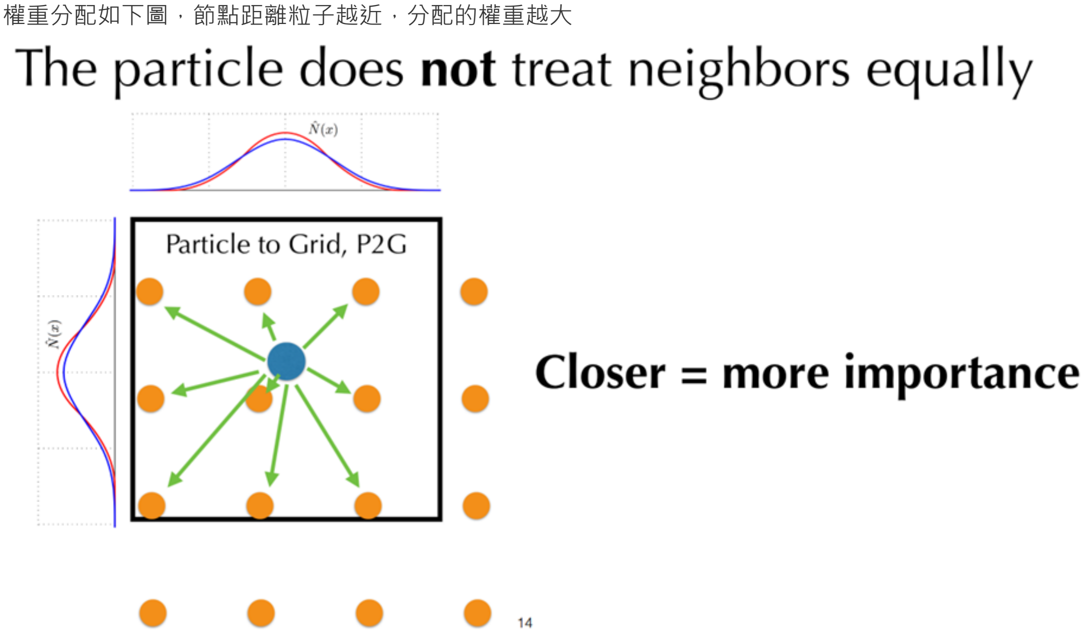
Fundamentals

MPM(Material point method)

- Continuum Material
  - 連續介質(Continuum Material)泛指各種流體及固體，或是所謂「連續體」，在連續介質力學中，有一個最基本的假設稱為「連續介質假設」，也就是流體以及固體都可以視為連續的介質組成，每個介質之間無空隙，並且介質之間受牛頓力學支配。
  - MPM方法即為模擬連續介質的一種方法，並透過連續介質力學推得整體方法的流程。
- Eulerian view and Lagrangian view
  - 在做物理模擬時，很常使用兩種視角去觀察整個物理系統，Eulerian視角以及Lagrangian視角。

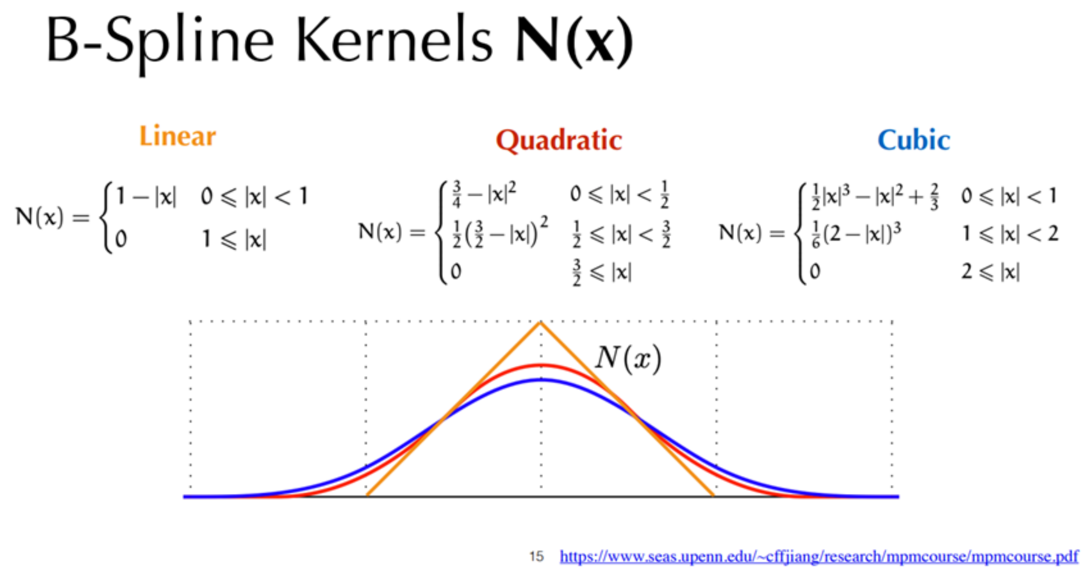
- Eulerian視角中，將整個物理系統切分成固定距離的網格，透過每個固定不動的網格點來觀察整個物理系統。Lagrangian視角則不同，將物質的每一個部分視為一個小小的區域，將這些區域稱為粒子(particle)，有自己的質量、能量以及速度，以這些粒子來觀察整個物理系統。
    - 注意: Lagrangian視角中，粒子並不是真正意義上的粒子，他代表連續體的一個子集，是一個連續的區域。
- 這兩種視角各有優缺點。
    - Eulerian視角中，由於每個網格之間的距離固定，很容易就可以取得相鄰的網格節點的資訊，可以十分有效的計算有關介質之間的物理量(Ex. 壓力)。但是使用Eulerian視角很容易使整體的物理系統能量耗散，難以維持整個物理系統的動能。
    - Lagrangian視角中，由於每個粒子的位置不同，且一個連續體中粒子的數量許多，如果需要計算粒子之間的物理量就會需要查找附近的粒子，效率很差。但相對Eulerian視角來說，由於每個粒子的質量守恆，比較容易維持整個物理系統的能量。
- MPM方法即結合這兩種視角，各取兩種視角的優點。MPM方法本質上是一種Lagrangian視角的方法，大部分的物理量如質量、速度等都以粒子的方式儲存，但在方法進行的過程中，會透過一種稱為P2G(Particle to Grid)的方法轉換至網格，也就是Eulerian視角，在此視角做其擅長的運算之後，再透過一種稱為G2P(Grid to Particle)的方法轉換回粒子，也就是Lagrangian視角，再進行此視角擅長的運算。

* P2G and G2P
    - 要在Lagrangian視角(Particle)以及Eulerian視角(Grid)之間互相轉換，需要透過P2G以及G2P方式轉換。
    - P2G中，會把粒子的質量以及動量轉換到網格節點上，在網格節點中，會先透過節點質量以及動量求出節點速度，並做處理，接著來到G2P階段，會將結點速度轉換回粒子身上。而達成這兩個方式最常見的一種方法稱為PIC(Particle-in-cell)。
    - PIC中，每一個粒子與每一個節點都有一個分配的權重，節點位置距離粒子越近，權重越大。P2G中粒子在轉換質量以及動量給節點時，就是根據這個權重去做轉換，權重越大的節點可以獲得該粒子較大部分的質量以及動量，相同的，G2P中節點在轉換速度給粒子時，權重越大的粒子可以獲得較大部分的速度。而為了保持質量以及動量守恆，粒子對於每個節點的權重和必須為1。在實務上時常使用B-spline kernal來計算分配的權重，且由於B-spline的特性，僅粒子附近的少數幾個節點的權重非零，因此做轉換時僅需處理這些節點即可。
        - 權重分配如下圖，節點距離粒子越近，分配的權重越大



        - 實務上，P2G中會將粒子的質量以及動量分配給節點，稱為scatter，而G2P中會將節點的速度轉移回粒子，或是由粒子去收集每個節點的速度，稱為gather。

- B-spline kernal如下，N(x)表示權重，可以使用Linear, quadratic或是cubic的B-spline kernal



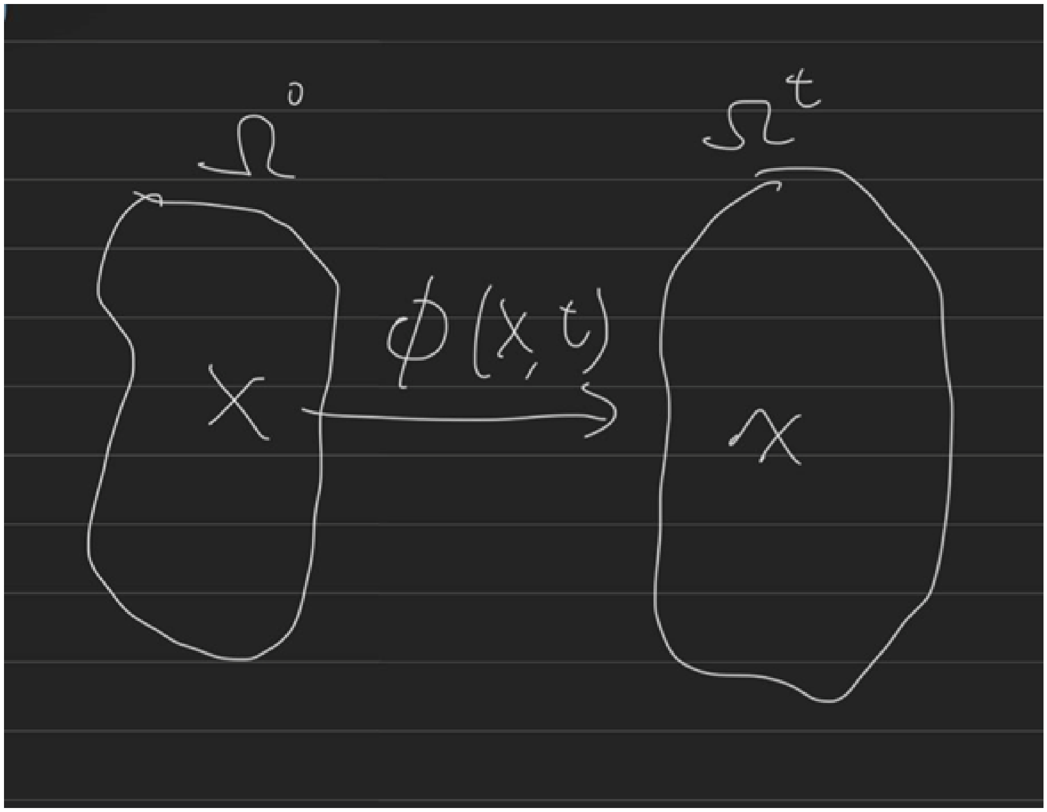
- 注意: 對於多個維度，權重就是每一個維度上的權重相乘，例如假設在三維空間中，粒子P對於節點i的權重$N_i(x_p) = N(\frac{1}{h}(x_p - x_i))N(\frac{1}{h}(y_p - y_i))N(\frac{1}{h}(z_p - z_i))$，其中$(x_p, y_p, z_p)$為粒子的位置，$(x_i, y_i, z_i)$為節點的位置，h為節點之間的間距。

  - 假設粒子質量為m_p，粒子速度為v_p，節點質量為m_i節點速度為v_i，P2G可以寫成下面兩式
    - 質量分配為$m_i = \sum_p m_p N_i(x_p)$
    - 動量分配為$(mv)_i = \sum_p m_p v_p N_i(x_p)$
  - G2P可以寫成此式
    - 速度分配為$v_p = \sum_i v_i N_i(x_p)$
  - 由於B-spline權重和為1，也就是$\sum_i N_i(x_p) = 1$因此可以維持質量以及動量守恆
    - 質量守恆: 由於$m_i = \sum_p m_p N_i(x_p)$，因此 $\sum_i m_i = \sum_i \sum_p m_p N_i(x_p) = \sum_p m_p \sum_i N_i(x_p) = \sum_p m_p$
    - 動量守恆: 由於$(mv)_i = \sum_p m_p v_p N_i(x_p)$，因此$\sum_i (mv)_i = \sum_p m_p v_p$

- Material parameters
  - 每一種不同的物質有自己的物質特性，而這些物質特性通常使用一些參數來表示。以下描述幾個在MPM方法中會使用到的物質參數
  - Young's modulus
    - 定義: $E = \frac{\sigma}{\epsilon}$，其中$\sigma$為單位面積受到的應力，$\epsilon$為由於應力受到的應變，也就是沿著應力方向改變的物體長度
    - 如果一個物質的Young's modulus低，則表示應力造成的應變較大，則此物體較有彈性，可以壓縮或伸長，反之則應力造成的應變較小，物體較硬、較不具彈性。
  - Poisson ratio
    - 定義: $\nu = -\frac{\epsilon_x}{\epsilon_z}$，其中$\epsilon_x$表示橫向的應變，$\epsilon_z$表示縱向的應變(假設施予縱向的應力)。
    - 如果一個物質的Poisson ratio低，則表示即使縱向應變很大造成的橫向應變也不大，所以物體就比較不容易型變(伸長或壓扁需要橫向的應變)，物體材質較硬。反之若Poisson ratio高，則物體材質較軟。
  - Lame' parameters
    - 藉由Young's modulus以及Poisson ratio，可以推導出兩個Lame' parameterμ以及λ，在MPM方法中將使用這兩個參數去描述物質的性質。
    - 推導方式如下(假設材質的Young's modulus為E，Poisson ratio為ν)
      - $\lambda = \frac{E\nu}{(1+\nu)(1-2\nu)}$

- $\mu = \frac{E}{2(1+\nu)}$

- Continuum Kinematic Theory
  - 接下來，將會藉由連續介質動力學推導MPM方法。
  - Deformation map
    - 想像一個粒子，原本沒有移動或型變時位置為X，且$X \in \Omega^0$，而在每一個時刻介質都會受到移動或型變，每一個粒子都會移動到另一個位置x，且$x \in \Omega^t$。將一個粒子從靜態位置映射到移動或型變後位置的函數為$\phi(X,t)$，在每一個時刻t，將靜態位置為X的粒子映射到位置$x = \phi(X,t)$，這個映射函數$\phi(X,t)$即稱為Deformation map。
      - 如下圖，deformation map將靜止位置映射至移動或形變後位置。



  - Deformation gradient
    - 定義Deformation gradient $F = \frac{\partial \phi}{\partial X}$，即deformation map的Jacobian，可以將Ω^0中的差值映射至Ω^t中的差值，如下圖

-

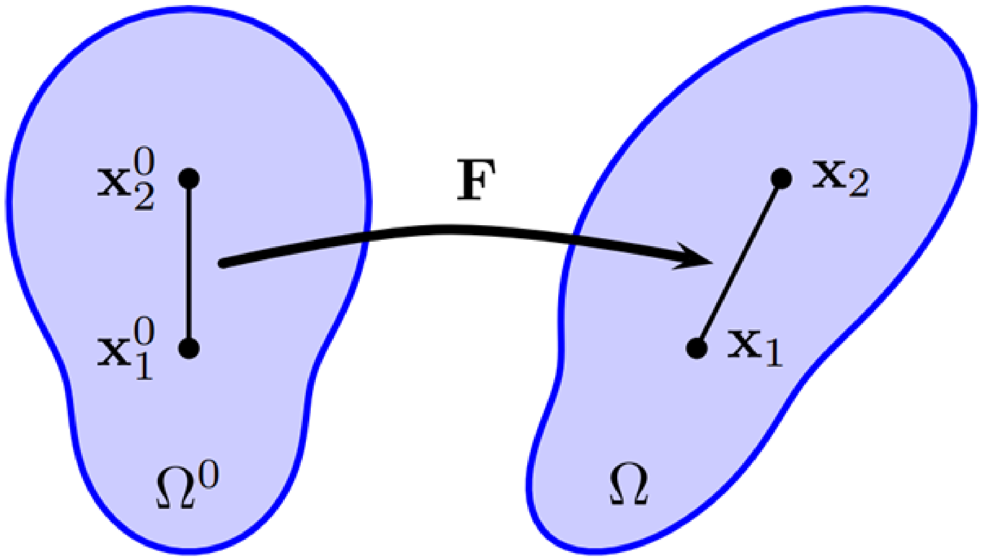


Figure 1: Deformation gradient.

  - 定義Deformation volume ratio $J = det(F)$，J為deformation grandient的determinant，所以J即為介質移動或形變後的體積變化比例。
- Stress-Strain energy density function
  - 當物體受到力時就有可能產生型變，而對於彈性材質，在不彈性疲乏造成物質變化的情況下，會趨向回復靜止的形狀。因此，可以定義一個物質在特定型變下有一個能量，稱為Stress-strain energy density function，在能量大於靜止形狀的能量時，便會透過此能量回復靜止的姿態。
  - 由於能量是由於型變造成的，因此能量應為型變的函數。而在這裡將deformation gradient視為型變，因此定義Energy function為F的函數$\psi(F)$，當F所代表的型變越大，則能量越大。
  - 要考慮F對型變之關係，可以先考慮剛性物體。
    - 對於剛性物體而言，型變為0，其deformation map為$\phi(X, t) = RX + b$，其中R為旋轉矩陣，b為平移量，從靜止位置映射到時刻t時的位置的過程中僅會旋轉和平移，不會使物體產生型變，
    - 此時Deformation gradient $F = \frac{\partial \phi}{\partial X} = R$，Deformation volume ratio = $J = det(F) = det(R) = 1$，體積不會改變。因此可知當F越接近旋轉矩陣R，也就是越接近Orthogonal矩陣，型變越小。而$R^T R = 1$，因此將$F^T F$視為F對於R之偏移量。
    - 所以將能量定義為$\psi(F) = \hat{\psi}(F^T F)$，能量會盡可能使$F^T F$回復為1，減少型變。
- Stress
  - 給定能量ψ以及deformation gradient F，可以將應力定義為$P = \frac{\partial \psi}{\partial F}$，這種應力稱為First Piola-Kirchoff Stress，或簡稱為PK1 stress。
  - 另一種應力的定義方式為$\sigma = \frac{1}{J} P F^T = \frac{1}{det(F)} \frac{\partial \psi}{\partial F} F^T$，這種應力稱為Cauchy stress。
- Hyperelasticity(超彈性)
  - 對於彈性材質，常會使用稱為New-Hookean的能量定義方式
    - 其能量定義為$\psi(F) = \frac{\mu}{2}(tr(F^T F) - d) - \mu \log(J) + \frac{\lambda}{2} \log^2(J)$，其中d表示材料所屬的空間維度(通常為2或3)
    - 其對應之PK1 stress為$P(F) = \frac{\partial \psi}{\partial F}(F) = \mu(F - F^{-T}) + \lambda \log(J) F^{-T}$
  - 也常使用一種稱為Fixed Corotated的能量定義方式
    - 其能量定義為$\psi(F) = \mu \Sigma_{i=1}^{d} (\sigma_i - 1)^2 + \frac{\lambda}{2}(J - 1)^2$，d亦為材料所屬的空間維度，$\sigma_i$則為F的singular value。

- 其對應之PK1 stress為$P(F)=\frac{\partial\psi}{\partial F}(F)=2\mu(F-R)+\lambda(J-1)JF^{-T}$，注意這裡的R為F做Polar decomposition後的rotation matrix，也就是若先對F做SVD，$F=U\Sigma V^T$，可以進一步寫成$F=UV^TV\Sigma V^T=RS$，$R=UV^T$，$S=V\Sigma V^T$，R就代表F中旋轉的部分。

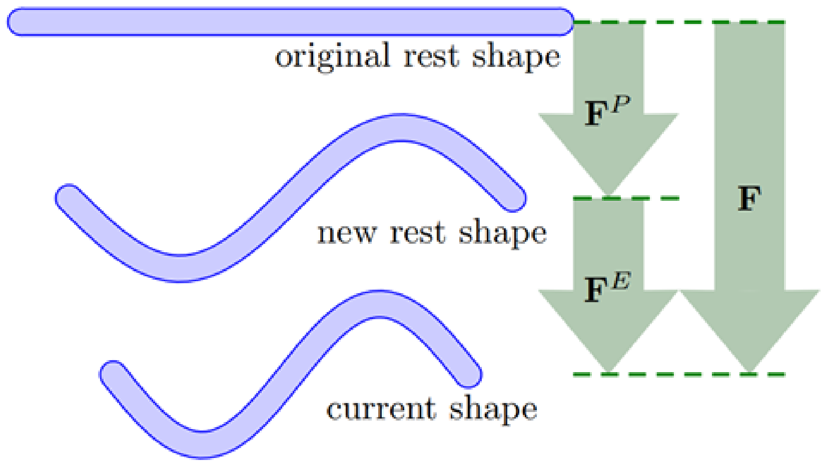
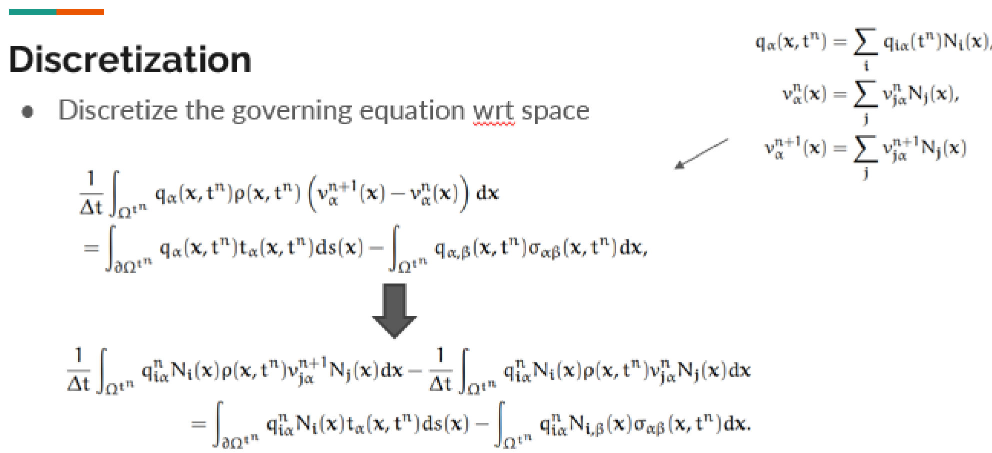
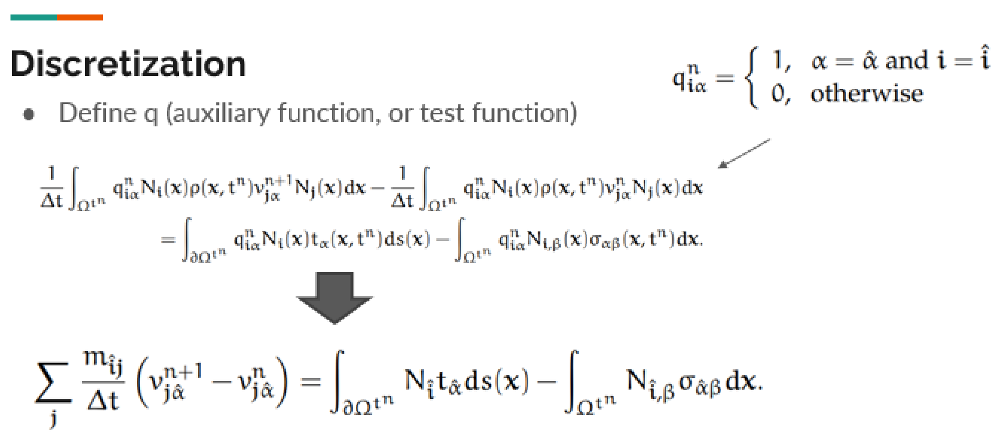
- Snow plasticity
  - 對於雪或流體這種材質，並不是全然的彈性，如果型變過大則會造成斷裂，也就是不可回復的型變，這種性質稱為plasticity(塑性)，相對的，可回復的彈性則稱為elasticity(彈性)
  - 在模擬的過程中，需要處理彈性型變過大導致塑性型變的過程。由於我們使用deformation gradient來表示型變，因此可以將F拆分為塑性以及彈性兩個部分，如下圖，$F=F_PF_E$，可以想像成先透過不可回復的塑性型變F_P轉移到一個新的靜止位置後，再透過可回復的彈性型變F_E映射至形變後的位置，只是這裡的彈性及為相對於新的靜止位置的彈性，會回復至新的靜止位置。
    - 

      Figure 2: Multiplicative decomposition of the deformation gradient.
  - 要將過大的彈性型變轉換至塑性型變，可以藉由限制F_E的singular value來達到。根據SVD，$F_E=U_E\Sigma_EV_E^T$，其中U_E,V_E^T為旋轉的部分，Σ_E為伸縮部分，因此整個F_E對位置的影響可以分為旋轉(U_E,V_E^T)以及伸縮部分(Σ_E)，而會使彈性型變轉換成塑性型變的原因，就是因為伸縮太大，導致介質斷裂，因此若F_E之Singular value超過一定範圍，則需要將超過的部分轉移至塑性型變
  - 因此，會給定一個Singular value的範圍$[1-\theta_c,1+\theta_s]$，θ_c稱為critical compression，影響可以壓縮的最小範圍，θ_s稱為critical stretch，影響可以伸長的最大範圍。
  - 在整個MPM方法中，每個time step會去迭代每個粒子的deformation gradient。假設已經知道此粒子的下一個時刻的deformation gradient為F^{n+1}
    - 下一個時刻的彈性型變$F_E^{n+1}=U_E\hat{\Sigma}_EV_E^T$，其中$F_E^n=U_E\Sigma_EV_E^T$，且$\hat{\Sigma}_E=clamp(\Sigma_E,[1-\theta_c,1+\theta_s])$，也就是$\Sigma_E$中的每個singular value都會被限制在這個範圍內。
    - 而由於$F_PF_E=F$，因此下一個時刻的塑性型變$F_P^{n+1}=(F_E^{n+1})^{-1}F^{n+1}=V_E\hat{\Sigma}_E^{-1}U_E^TF^{n+1}$，如此，就可以將超出範圍的的Singular value轉移到塑性型變上。
  - 而新的塑性型變會造成新的靜止位置，進而造成新的物質特性。例如雪在很集中時較堅韌，較分開時較易碎。因此需要使物質的參數根據塑性的型變作調整。因此每一個time step更新完塑性型變F_P後，便會先取得其體積變化比例$J_P=det(F_P)$，並將lame parameter更新為$\mu(F_p)=\mu_0^{\xi(1-J_P)},\lambda(F_P)=\lambda_0e^{\xi(1-JP)}$，其中$\mu_0,\lambda_0$為物質的初始lame' parameter，$\xi$為指定的hardening coefficient，透過調整物質參數便可以模擬物質在不同的靜止位置時不同的特性。

- Governing equation

- 藉由連續介質力學中的質量守恆以及動量守恆，可以推導力平衡方程式: $\int_{\Omega^t} q_i(x,t)\rho(x,t)a_i(x,t)dx = \int_{\partial\Omega^t} q_i t_i ds(x) - \int_{\Omega^t} q_{i,k}\sigma_{i,k}dx$，其中$q_i(x,t)$為定義在每個節點的輔助函數，可自行定義，$\rho(x,t)$為密度，$a_i(x,t)$為加速度，$t_i$為邊界條件，$ds(x)$為極小區域面積，$\sigma$為Cauchy stress。
- 將此力平衡方程式對時間進行離散化，也就是使$a_i(x,t) \approx \frac{v_\alpha^{n+1}(x)-v_\alpha^n(x)}{\Delta t}$，則力平衡方程式則變為 $\int_{\Omega^t} q_i(x,t^n)\rho(x,t^n)(v_\alpha^{n+1}(x) - v_\alpha^n(x))dx = \int_{\partial\Omega^t} q_\alpha(x,t^n)t_\alpha(x,t^n)ds(x) - \int_{\Omega^t} q_{\alpha,\beta}(x,t^n)\sigma_{\alpha,\beta}(x,t^n)dx$，注意$\alpha$為每個維度上的分量
- 接著將方程式對空間進行離散化。這是MPM方法中需要使用Grid的地方，將粒子轉為網格，就是對空間進行離散化。將輔助函數以及速度寫為網格形式，就可以將方程式寫成對空間離散的形式。如下圖
  - 

- 接著定義輔助函數，即可推導出現在時刻t^n以及下一個時刻t^{n+1}的動量差，就可以得到更新節點動量，也就是更新速度的方式。如下圖。
  - 

- 而$\int_{\Omega^{t^n}} N_{i,\beta}(x)\sigma_{\alpha\beta}(x,t^n)dx$經過轉換可以寫成$\sum_p P_{p_{\alpha\gamma}}^n F_{p_{\beta\gamma}}^n N_{i,\beta}(x_p^n)V_p^0$，於是整個方程式可以寫成以下形式
  - $$\boxed{\frac{\left((mv)_{i\alpha}^{n+1} - (mv)_{i\alpha}^n\right)}{\Delta t} = \int_{\partial\Omega^{t^n}} N_i(\mathbf{x})t_\alpha(\mathbf{x},t^n)ds(\mathbf{x}) - \sum_p P_{p\,\alpha\gamma}^n F_{p\,\beta\gamma}^n N_{i,\beta}(\mathbf{x}_p^n)V_p^0}.$$
  - 等式右側的積分項在設定Neumann boundary condition的情況下，會變成0，因此只需計算$\sum_p P_{p_{\alpha\gamma}}^n F_{p_{\beta\gamma}}^n N_{i,\beta}(x_p^n)V_p^0$
  - 所以在每一個time step更新速度時，需要知道每個粒子的PK1 stress, deformation gradient, 權重函數，以及初始體積。權重函數以及初始體積為固定值，所以每一個time step必須計算出PK1 stress以及下一個時刻的deformation gradient。

- 要更新Deformation gradient，需要將其對時間進行離散化。
  - 先將$F(X, t^{n+1})$對時間進行偏微分: $\frac{\partial}{\partial t}F(X_p, t^{n+1}) = \frac{\partial v^{n+1}}{\partial x}(\phi(X, t^n))F(X, t^n)$
  - 接著將其寫成對時間離散化的形式: $\frac{\partial}{\partial t}F(X_p, t^{n+1}) \approx \frac{F_p^{n+1} - F_p^n}{\Delta t}$
  - 經過整理，即可得到deformation gradient更新的方式: $F_p^{n+1} = F_p^n + \Delta t \frac{\partial v^{n+1}}{\partial x}(x_p^n)F_p^n = (I + \Delta t \frac{\partial v^{n+1}}{\partial x}(x_p^n))F_p^n$
  - 而其中的$\frac{\partial v^{n+1}}{\partial x}$，也稱為速度導數，由於$v^{n+1}(x) = \sum_i v_i^{n+1}N_i(x)$，因此可以寫成$\frac{\partial v^{n+1}}{\partial x}(x) = \sum_i v_i^{n+1}(\frac{\partial N_i}{\partial x}(x))^T$
  - 所以更新deformation gradient的方式為$F_p^{n+1} = (I + \Delta t \sum_i v_i^{n+1}(\frac{\partial N_i}{\partial x}(x_p^n))^T)F_p^n$
  - 透過以上方式，只要知道下一個時刻每個節點的速度v_i^{n+1}，就可以更新下一個時刻每個粒子的deformation gradient
- 要更新速度，可以從彈性能量的觀點去看。
  - 假設在下一個時刻，每個粒子移動至$\hat{x}$，則整體的彈性能量為$e(\hat{x}) = \sum_p \psi(F_p(\hat{x}))V_p^0$，將其對節點位置偏微分，即可得到每個節點在下一個時刻的受力，也就是$-f_i(\hat{x}) = \frac{\partial e}{\partial \hat{x}_i}(\hat{x}) = \sum_p V_p^0 \frac{\partial \psi}{\partial F}(\hat{F}_p(\hat{x}))(F_p^n)^T \nabla w_{ip}^n$，每一個維度的分量$-f_{i\alpha}(\hat{x}) = \frac{\partial e}{\partial \hat{x}_{i\alpha}}(\hat{x}) = \sum_p P_{\alpha\gamma}(F_p(\hat{x}))F_{p_{\tau\gamma}}^n N_{i,\tau}(x_p^n)V_p^0$，其實就是動量方程式中的等式右側，可以使用這個受力去更新節點的速度。
  - 也就是說，更新動量的方式為$(mv)_{i\alpha}^{n+1} = (mv)_{i\alpha}^n - \Delta t \frac{\partial e}{\partial \hat{x}_{i\alpha}}(\hat{x})$，透過受力更新動量
  - 以上使用的是下一個時刻的粒子位置造成節點的受力，是一種implicit的更新方式，要計算$P(\hat{F}_p(\hat{x}))$會需要計算Heissan matrix，因此實務上為了效率以及簡化，會使用explicit的方式更新，也就是直接使用這個時刻的粒子位置x去做計算，$f_i^n = f_i(x_i^n) = -\sum_p V_p^0(\frac{\partial \psi_p}{\partial F}(F_p^n))(F_p^n)^T \nabla w_{ip'}^n$
- 總之，每個時刻必須去計算每個節點的受力並更新節點的動量，並透過新的節點速度更新deformation gradient。

- MPM Scheme
  - 統整以上的推導，得到MPM方法的整體演算法
  - At each time step n
    1. P2G: 將粒子的質量以及動量scatter到結點上
       1. $m_i = \sum_p m_p N_i(x_p)$
       2. $(mv)_i \sum_p m_p v_p N_i(x_p)$
    2. 透過節點的質量以及動量計算節點的速度: $v_i = \frac{(mv)_i}{m_i}$
    3. 決定每個節點的自由度，只處理部分需要處理的節點，也就是質量不為零的節點
    4. 計算節點的力: $f_i^n = f_i(x_i^n) = -\sum_p V_p^0(\frac{\partial \psi_p}{\partial F}(F_p^n))(F_p^n)^T \nabla w_{ip'}^n$
    5. 更新節點的速度: $v_i^{n+1} = v_i^n + \Delta t \frac{f_i(x_i^n)}{m_i}$
       1. 在這一步需要處理碰撞情形
       2. 如果需要維持不可壓縮性，可以在更新完速度後做Chorin style projection將散度投影為0，得到最後正確的速度。
    6. 透過新的節點速度更新粒子的deformation gradient，並限制彈性型變得singular value，轉為塑性型變
       1. $F_p^{n+1} = (I + \Delta t \sum_i v_i^{n+1}(\frac{\partial N_i}{\partial x}(x_p^n))^T)F_p^n$
       2. $F_E^{n+1} = U_E \hat{\Sigma}_E V_E^T$, where $F_E^n = U_E \Sigma_E V_E^T$ and $\hat{\Sigma}_E = clamp(\Sigma_E, [1 - \theta_c, 1 + \theta_s])$
       3. $F_P^{n+1} = (F_E^{n+1})^{-1}F^{n+1} = V_E \hat{\Sigma}_E^{-1} U_E^T F^{n+1}$
    7. G2P: 將節點的速度gather回粒子上

        1. $v_p = \sum_i v_i N_i(x_p)$
    8. 更新粒子的位置: $x_p^{n+1} = x_p^n + \Delta t v_p^{n+1}$
    9. 粒子碰撞處理
  - MPM方法利用Lagrangian視角以及Eulerian視角的優勢，各取其長，Lagrangian負責儲存大部分物理量與平移，Eulerian負責計算力以及做Projection(如果需要的話)。
- APIC(Affine PIC)
  - 在使用PIC進行P2G以及G2P時，由於是從一個粒子轉到多個節點，可能會損失部分動量。
  - 因此Affine PIC對PIC做了修正，在做P2G以及G2P的過程中加上一個Affine矩陣，以更好的維持動量。
  - 具體而言，對於任何一個粒子會定義一個Affine matrix $C_p^n = B_p^n (D_p^n)^{-1}$，其中
    - $D_p^n = \sum_i w_{ip}^n (x_i - x_p^n)(x_i - x_p^n)^T$
    - $B_p^{n+1} = \sum_i w_{ip}^n \hat{v}_i^{n+1} (x_i - x_p^n)^T$
  - 在P2G時，質量scatter方式不變，動量scatter方式則修正為
    - $m_i^n v_i^n = \sum_p w_{ip}^n m_p (v_p^n + B_p^n (D_p^n)^{-1} (x_i - x_p^n))$
  - 在G2P時，更新下一個時刻的Affine matrix
    - $C_p^{n+1} = \frac{4}{\Delta x^2} \sum_i w_{ip} v_i^{n+1} (x_i - x_p^n)^T$
  - 若要完整保留動量，可以使用更複雜的PolyPIC，但通常只會使用APIC

MLS-MPM(Moving least square MPM)

- MLS-MPM對MPM中進行P2G以及G2P的權重進行了修正，將其替換為Moving least square的權重，使的更新deformation gradient以及計算節點受力可以更有效率。
- Moving least square
  - 對於給定的一組資料點$x_0, x_1 \cdots x_n$，以及一段定義於x上之函數$u(x)$，$u_i = u(x_i)$，想要透過一組多項式的線性組合去近似函數上的任意點z，也就是想要使$u(z) = P^T(z)c(x)$，其中
    - $P(z) = [p_0(z), p_1(z), \cdots, p_l(z)]^T$為指定的多項式
    - $c(x) = [c_0(x), c_1(x), \cdots, c_l(x)]^T$為線性組合的係數
  - 可以將多項式平移至以固定點x做為中心，使$u(z) = P^T(z - x)c(x)$
  - 而為了使線性組合接近函數u，定義兩者的差距函數$J_x(c) = \sum_{i \in B_x} \xi_i(x)(P^T(x_i - x)c(x) - u_i)^2$，其中$\xi_i(x)$為以$x_i$為中心的權重函數，$B_x$代表使$\xi_i(x) \neq 0$的i的集合。
  - 透過推導，可以知道使$J_x(c)$最小的係數$c(x) = M^{-1}(x)b(x)$，其中
    - $b(x) = \sum_{i \in B_x} \xi_i(x) P(x_i - x) u_i$
    - $M(x) = \sum_{i \in B_x} \xi_i(x) P(x_i - x) P^T(x_i - x)$
  - 所以函式u就可以寫成$u(z) = \sum_{i \in B_x} P^T(z - x) M^{-1}(x) P(x_i - x) u_i$，令$\phi_i(z) = \xi_i(x) P^T(z - x) M^{-1}(x) P(x_i - x)$為權重函數，則原函數可以寫成$u(z) = \sum_{i \in B_x} \phi_i(z) u_i$，每個資料點的函數值的線性組合
- MLS-MPM對MPM的修正
  - 而MLS-MPM替換了原本MPM中使用的插值函數，使
    - $v_\alpha^n(x) = \sum_j \phi_j(x) v_{j\alpha}^n$
    - $q_\alpha(x, t^n) = \sum_i \phi_i(x) q_{i\alpha}^n$
  - 經過推導可以得到下面三個結果
    - 每個節點的受力可以寫成$-w_{ip} \frac{4\Delta t}{\Delta x^2} \sum_p V_p^0 P(F_p^{n+1})(F_p^{n+1})^T (x_i - x_p^n)$
    - Affine matrix $C_p = \frac{4}{\Delta x^2} \sum_i w_{ip} v_i (x_i - x_p)^T$

- 可以使用Affine matrix近似速度梯度$\nabla v = \frac{\partial v^{n+1}}{\partial x}$，所以在更新deformation gradient時可以直接使用Affine matrix去更新deformation gradient: $F_p^{n+1} = (I + \Delta t C_p^n) F_p^n$
- MLS-MPM Scheme
  - 所以MLS-MPM整體的演算法如下
  - At each time step n
    1. 更新Deformation gradient: $F_p^{n+1} = (I + \Delta t C_p^n) F_p^n$
       1. 並轉移彈性型變至塑性型變
    2. P2G
       1. scatter質量: $m_i^{n+1} = \sum_p m_p w_{ip}$
       2. scatter動量: $(mv)_i^{n+1} = \sum_p w_{ip}\{m_p v_p^n + [m_p C_p^n - \frac{4\Delta t}{\Delta x^2} \sum_p V_p^0 P(F_p^{n+1})(F_p^{n+1})^T](x_i - x_p^n)\}$
          1. 注意: 這裡的$m_p v_p^n + m_p C_p^n (x_i - x_p^n)$為原本的APIC轉換，剩下的部分是節點受力，直接在P2G的時後就直接對節點施加彈性受力，更新動量。
    3. 得到節點速度: $v_i^{n+1} = \frac{(mv)_i^{n+1}}{m_i^{n+1}}$，並做碰撞處理以及Projection。
    4. G2P
       1. gather速度: $v_i^{n+1} = \sum_i w_{ip} v_i^{n+1}$
       2. 更新Affine matrix: $C_p^{n+1} = \frac{4}{\Delta x^2} \sum_i w_{ip} v_i^{n+1} (x_i - x_p^n)^T$
    5. 更新粒子位置: $x_p^{n+1} = x_p^n + \Delta t v_p^{n+1}$
    6. 粒子碰撞處理
- MLS-MPM相比MPM的好處
  1. 使用APIC進行轉換，盡可能保持系統動量。
  2. MLS-MPM中，直接在P2G的scatter動量中去對節點動量進行更新，而不是在P2G之後才去計算受力並更新，可以減少運算量。
  3. MLS-MPM中，直接使用Affine matrix去近似velocity gradient，用來更新deformation gradient，使更新的步驟可以提前至P2G，不需要等待節點計算完新的速度後才去更新，使得兩者可以並行，並減少運算量(步需要計算velocity gradient)

Implementations

- Taichi
  - 我們使用Taichi這個程式語言進行物理模擬。Taichi是一個內嵌於Python中的程式語言，針對數值運算進行高效能的優化，並有內建的GUI系統，適合進行物理模擬。
  - 我們將修改其提供的範例程式碼: mpm3d_ggui.py進行我們的物理模擬。
- 我們將透過Taichi生成模擬結果的粒子動畫，並輸出至Houdini中進行渲染，包含表面重建以及光線設定等。

Results and discussion

- 我們測試了MPM以及MLS-MPM模擬的效果，並測試不同參數對模擬效果的影響
- 程式碼位於Snow_simulation資料夾中，為Snow_simulation.py
- 模擬的動畫結果位於這個連結，或是可以使用以下提供的Youtube連結觀看
- 結果中除了mpm_reference_ply以外，都是使用MLS-MPM模擬。
- 以reference_ply的結果作為基準，以下比較不同參數對結果的影響

- 我們調整了四個參數進行比較
    1. Young's modulus E
    2. Hardening coefficient ξ
    3. Critical stretch θ_s
    4. Critical compression θ_c
  - 不同檔名分別對應不同係數的調整
  - Hardening coefficient會影響lame parameter，近一步影響物體特性
    - 如果較大，則雪較脆，可見reference_ply
    - 如果較小，則雪較堅韌，可見lower_hardening_ply
  - Young's modulus會影響材質的硬度
    - 如果較大，則雪較硬，可見reference_ply
    - 如果較小，則雪較軟，可見lower_young_modulus_ply
  - Critical stretch以及Critical compression會影響彈性限度，也就是型變超過此範圍後會產生不可復原的塑性型變。
    - 如果較大，則雪較不容易斷裂，容易成塊，可見reference_ply
    - 如果較小，則雪較容易斷裂，容易變成細粉，可見lower_critical_compression_and_critical_stretch_ply
- 我們也比較了MPM以及MLS-MPM的差異
  - MLS-MPM對於step size的容忍度較大，所以移動較快，且更可以保持動量，可見reference_ply
  - MPM對於step size的容忍度較大，系統容易不穩定，且能量容易耗散，容易卡住之後就不動，可見mpm_reference_ply
- 實際上渲染出的效果與流體相似，較不似雪，我們認為原因有以下幾點
  1. 模擬使用的粒子數量不足，導致模擬效果不接近真實
  2. 使用Houdini重建表面的方式有誤，可能需要使用不同重建表面的方式。

Extra

- 除了研究如何使用MPM以及MLS-MPM進行雪的模擬之外，我們還另外研究了如何模擬相變材料。所謂相變材料，即是在模擬過程中會因為溫度改變相態的物質，例如水的融化以及凝固。
- 為了模擬這種性質，需要去探討溫度的變化以及其對物質特性的影響。Augmented MPM這個方法即對MPM方法做了改善，使其可以模擬相變材料。
- Augmented MPM
  - 整體的概念為在每個時刻更新每個MPM粒子的溫度以及相態，並根據相態具調整物質的參數，以使不同溫度和狀態的粒子表現他們該有的特性。
  - Temperature evolve
    - 為了模擬溫度的變化，必須在每一個time step更新溫度
    - 根據熱力學，有以下三式
      - $\rho\frac{Du}{Dt} = -\nabla\cdot q$，其中$\rho$為密度，$u$為每單位質量擁有的熱能，$q$為熱通量
      - $q = -\kappa\nabla T$，其中κ為熱傳導係數，T為溫度
      - $c = \frac{du}{dT}$，其中c為每單位質量的熱容量
    - 透過簡化，可以得到: $\rho c = \frac{DT}{Dt} = \nabla\cdot(\kappa\nabla T)$
    - 將其對時間做離散化，即可得到: $T^{n+1} - T^n = \frac{\Delta t}{\rho^n c^n}\nabla\cdot(\kappa^n\nabla T^{n+1})$，即可更新下一個時刻的溫度。

- Latent heat
  - 不只需要改變溫度，當溫度達到熔點或冰點等導致物體相態變化的溫度時，則需要去改變相態
  - 物質在改變相態時，會使用所謂潛熱，也就是若對此材質施加熱能，這些熱能會被拿去改變物質的相態，物質溫度不會變化。
  - 所以，使用一個所謂「潛熱槽」的概念去模擬這種特性。具體而言，每次當溫度改變時，會去計算溫度是否達到熔點或冰點，若已達到則固定其溫度於熔點或冰點，並將多餘的溫度變化計算成熱能累計至潛熱槽中，當潛熱槽超出一定限制時，即表示熱能足夠使此粒子改變相態，因此改變粒子的相態，並使其能繼續改變溫度。
- 物質參數
  - 物質的參數如Lame' parameter、熱容量和熱傳導係數都會隨著溫度以及相態變化，每個time step更新完溫度以及相態後變換更新這些物質參數。
- Constitutive model
  - 由於整個模擬同時牽涉到流體以及固體，因此需要對能量做進一步調整。
  - 首先，能量由彈性型變定義，並使用Fixed Corotated的方式定義能量，所以可以把能量拆分成由μ以及λ負責的部分，即$\psi(F_E) = \psi_\mu(F_E) + \psi_\lambda(J_E)$，其中
    - $\psi_\mu(F_E) = \mu||F_E - R_E||_F^2$
    - $\psi_\lambda(J_E) = \frac{\lambda}{2}(J_E - 1)$
  - 這樣的能量定義適用於固體，但對於流體，由於其僅會根據體積變化而型變，不會因為旋轉而造成型變，因此對於流體通常會將μ設為0，以刪除包含旋轉型變R_E的能量ψ_μ
  - 但由於ψ_μ也包含部分的體積型變，如果將其刪除將無法正確表現特性，因此會對ψ_μ做調整。
    - 對於彈性型變F_E，可以將其拆解為膨脹或收縮的部分$(J_E)^{\frac{1}{d}}I$以及旋轉的部分$(J_E)^{-\frac{1}{d}}F_E$
    - 將ψ_μ修正為$\hat{\psi}_\mu(F_E) = \psi_\mu(J_E^{-\frac{1}{d}}F_E)$，使$\psi_\mu$在整體的能量中僅包含旋轉的部分，如此，使流體狀態的粒子的$\mu = 0$，即可刪除旋轉導致的型變能量。
  - 除此之外，若粒子處於流體狀態，仍需要在每次更新彈性型變F_E之後將其旋轉的部分刪除，以免之後牽涉他的運算會包含旋轉型變。
    - 具體而言，在每個time step最後，若粒子處於流體狀態，會將F_E 更新為$(J_E)^{\frac{1}{d}}I$，只留下體積變化的部分
- Pressure and velocity
  - 壓力變化會造成粒子位置、速度或溫度變化，因此也需要去在每個time step更新壓力。
  - 根據Cauchy stress的定義，$\sigma_\lambda = \frac{1}{J}(\frac{\partial\psi_\lambda}{\partial J_E}\frac{\partial J_E}{\partial F_E})F_E^T = \frac{1}{J}\frac{\partial\psi_\lambda}{\partial J_E}J_E F_E^{-T}F_E^T = -pI$
  - 其中p為壓力，定義為$p = -\frac{1}{J_P}\frac{\partial\psi_\lambda}{\partial J_E} = -\frac{1}{J_P}\lambda(J_E - 1)$
  - 將其對時間離散化
    - $\frac{Dp}{Dt} = -\frac{1}{J_P}\frac{\partial^2\psi_\lambda}{\partial J_E^2}\frac{DJ_E}{Dt}$
    - 由於$J = J_E J_P$且$\frac{D_J}{Dt} = J\nabla\cdot v$，所以$\frac{DJ_E}{Dt} = J_E\nabla\cdot v$
    - 所以$\frac{Dp}{Dt} = -\frac{1}{J_P}\frac{\partial^2\psi_\lambda}{\partial J_E^2}J_E\nabla\cdot v = -\frac{\lambda J_E}{J_P}\nabla\cdot v$
    - 將$\frac{Dp}{Dt}$近似為$\frac{p^{n+1}-p^n}{\Delta t}$，即可得到$\frac{p^{n+1}-p^n}{\Delta t} = -\frac{\lambda^n J_E^n}{J_P^n}\nabla\cdot v^{n+1}$
  - 同時，系統遵循Navier Sroke equation，也將其對時間離散化
    - $\rho\frac{Dv}{dt} = \nabla\cdot\sigma + \rho g = \nabla\cdot\sigma_\mu - \nabla p + \rho g$
    - 將$\frac{Dv}{Dt}$近似為$\frac{v^{n+1}-v^n}{\Delta t}$，即可得到$\frac{v^{n+1}-v^n}{\Delta t} = \frac{1}{\rho^n}\nabla\cdot\sigma_\mu - \frac{1}{\rho^n}\nabla p^{n+1} + g$
    - 將上式做operator splitting，即可得到以下兩式

        - $\frac{v^*-v^n}{\Delta t} = \frac{1}{\rho^n}\nabla\cdot\sigma_\mu + g$
        - $\frac{v^{n+1}-v^*}{\Delta t} = -\frac{1}{\rho^n}\nabla p^{n+1}$
      - 對第二式兩側取divergence，整理後可以得到: $\frac{J_P^n}{\lambda^n J_E^n}\frac{p^{n+1}}{\Delta t} - \Delta t\nabla\cdot\left(\frac{1}{\rho^n}\nabla p^{n+1}\right) = \frac{J_P^n}{\lambda^n J_E^n}\frac{p^n}{\Delta t} - \nabla\cdot v^*$
    - 所以，更新速度以及壓力的流程為
      1. 透過$\frac{v^*-v^n}{\Delta t} = \frac{1}{\rho^n}\nabla\cdot\sigma_\mu + g$，使用Cauchy stress以及外力得到$v^*$
      2. 透過$\frac{J_P^n}{\lambda^n J_E^n}\frac{p^{n+1}}{\Delta t} - \Delta t\nabla\cdot\left(\frac{1}{\rho^n}\nabla p^{n+1}\right) = \frac{J_P^n}{\lambda^n J_E^n}\frac{p^n}{\Delta t} - \nabla\cdot v^*$得到下一個時刻的壓力$p^{n+1}$
      3. 透過$\frac{v^{n+1}-v^*}{\Delta t} = -\frac{1}{\rho^n}\nabla p^{n+1}$，得到下一個時刻的速度$v^{n+1}$
- Augmented MPM Scheme
  - 統整上述，以下為Augmented MPM的整體演算法
  - At each time step
    1. 對deformation gradient施加plasticity，也就是限制彈性型變的singular value，轉移至塑性型變
    2. P2G
       1. 注意: 這裡使用的網格節點分為兩種: 中心網格節點以及MAC網格節點，中心網格節點位於網格中心，MAC網格節點位於每條邊上
       2. 粒子的質量會分別scatter至兩種節點
       3. 粒子的速度和熱傳導係數會scatter至MAC網格節點
       4. 粒子的$J, J_E, c, T, \lambda^{-1}$會scatter至中心網格節點
    3. Classify cell
       1. 首先，會先去檢查一個網格的四個邊是否碰撞，並標記。
       2. 接著標記網格種類，在接下來的步驟會用到
          1. 如果網格的四個邊的都碰撞，則標記此網格為碰撞
          2. 若否，則檢查網格的四個邊是否皆有質量，若是則標記此網格為內部網格
          3. 若否，則標記此網格為空網格，
    4. 使用計算每個節點受力並將v更新為v*，可以使用MPM中explciit或是implicit的方法做計算。
    5. 透過方程式: $\frac{J_P^n}{\lambda^n J_E^n}\frac{p^{n+1}}{\Delta t} - \Delta t\nabla\cdot\left(\frac{1}{\rho^n}\nabla p^{n+1}\right) = \frac{J_P^n}{\lambda^n J_E^n}\frac{p^n}{\Delta t} - \nabla\cdot v^*$求得$p^{n+1}$
       1. 這是一個柏松方程，可以使用與Chorin style projection的方式將其離散化後寫為線性系統並求解
       2. 可以使用Conjugate Gradient或是Multigrid Preconditioning Conjugate Gradient對線性系統快速求解。
    6. 透過方程式: $\frac{v^{n+1}-v^*}{\Delta t} = -\frac{1}{\rho^n}\nabla p^{n+1}$求得$v^{n+1}$
       1. 將右側離散化即可更新
    7. 透過方程式: $T^{n+1} - T^n = \frac{\Delta t}{\rho^n c^n}\nabla\cdot\left(\kappa^n\nabla T^{n+1}\right)$更新溫度
       1. 將右側離散化後對線性系統求解即可。
    8. G2P: 將節點速度、熱容量以及溫度gather回粒子上，並更新deformation gradient
       1. 如果粒子處於流體狀態，則要刪除其彈性型變的旋轉部分: $F_{E_P}^{n+1} = \left(J_{E_P}^{n+1}\right)^{\frac{1}{d}} I$
       2. 更新溫度時，如果達到熔點或冰點等，需要依照潛熱的方式處理
    9. 粒子的碰撞處理
- Implementation
  - 雖然我們已經完成整個演算法的實現，但是由於未知的原因，數值系統在經過幾個time step之後便會變的不穩定。因此無法展示模擬結果。

- 程式碼位於Phase_change_material中，主程式為phase_change.py，其餘為輔助程式碼或是線性系統求解器。

Conclusion

- MPM方法結合Lagrangian視角以及Eulerian視角，各取所長，各自負責物理模擬流程中自己擅長的部份，並使用P2G和G2P轉換兩種視角。
- MLS-MPM方法修改MPM方法中的權重，減少運算量，並使用APIC做為P2G以及G2P的方式，維持動量守恆
- Augmented MPM透過溫度以及相態調整物質參數，使物體可以模擬相態變化時的特性。

Member contribution

- 110550059 劉珆睿
    - 題目發想
    - MPM, MLS-MPM, Augmented MPM方法研究
    - 實驗
    - 報告製作
- 110550047 巫廷翰
    - 題目發想
    - 實驗
    - Houdini 渲染
    - 細節調整
    - 報告製作

Reference

- STOMAKHIN, A., SCHROEDER, C., CHAI, L., TERAN, J., AND SELLE, A. 2013. A material point method for snow simulation. ACM Trans. Graph. 32, 4 (July 2013) (paper)
- C. Jiang, C. Schroeder, A. Selle, J. Teran, and A. Stomakhin. 2015. The affine particle-in-cell method. ACM Trans Graph 34, 4 (2015), (paper)
- Hu Y., Fang Y., Ge Z., Qu Z., Zhu Y., Pradhana A., and Jiang C.. 2018. A moving least squares material point method with displacement discontinuity and two-way rigid body coupling. ACM Transactions on Graphics 37, 4, Article 150 (2018) (paper)(github)
- Yuanming Hu, Tzu-Mao Li, Luke Anderson, Jonathan Ragan-Kelley, and Frédo Durand. 2019. Taichi: A Language for High-Performance Computation on Spatially Sparse Data Structures. ACM Trans. Graph., 38, 6 (2019), Article 201, Nov(paper)(github)
- STOMAKHIN, ALEXEY, SCHROEDER, CRAIG, JIANG, CHENFANFU, et al. 2014. “Augmented MPM for Phase-Change and Varied Materials”. ACM Trans. Graph. 33.4 (July 2014) (paper)
- C. Jiang, C. Schroeder, J. Teran, A. Stomakhin, and A. Selle. 2016. The material point method for simulating continuum materials. In SIGGRAPH 2016 Course (pdf)
- GAMES201 online course (github)
- WaterSim: 2D fluid simulation (github)
- FLIP using MGPCG (zhihu)

Files and links

- 程式碼位於Group_34_code.zip中，分別有Snow_simulation以及Phase_change_material兩個資料夾。
- 模擬結果的Youtube連結如下
    - lower_critical_compression_ply
    - lower_young_modulus_ply
    - lower_critical_stretch_ply
    - mpm_reference_ply
    - lower_critical_compression_and_stretch_ply
    - lower_hardening_ply
    - reference_ply