Position Based Fluids

- ## Basic Field Theory
  - Operator
    - **Hamiltonian**
      - **Scalar form:** $\nabla_{\mathbf{v}} f = \frac{\partial f}{\partial \mathbf{v}} = \{\frac{\partial f}{\partial v_x}, \frac{\partial f}{\partial v_y}, \frac{\partial f}{\partial v_z}\}$
      - **Vector form:** $\nabla_{\mathbf{v}} \boldsymbol{f} = \mathbf{J}(\boldsymbol{f}(\mathbf{v})) = \frac{\partial \boldsymbol{f}}{\partial \mathbf{v}}$, refer to Chapter 2 Math Background: Vector, Matrix and Tensor Calculus
    - **Laplacian:** $\Delta f = \nabla^2 f = \nabla \cdot \nabla f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} + \frac{\partial^2 f}{\partial z^2}$
  - Gradient
    - **Link:** 梯度 - 维基百科，自由的百科全书 (wikipedia.org)
  - Divergence
    - **Link:** 散度 - 维基百科，自由的百科全书 (wikipedia.org)
  - Curl
    - **Link:** 旋度 - 维基百科，自由的百科全书 (wikipedia.org)
- ## Position Based Fluids

  Simulation

  - Theory
    - **Incompressibility**
      - **Explanation:** Fluids have a tendency to maintain the rest density
      - **Constraints:** $C_i(\mathbf{p}) = \frac{\rho_i}{\rho_0} - 1 = -1 + \frac{1}{\rho_0}\sum_{j=1}^{n}\frac{1}{V}m_j W(\mathbf{p}_i - \mathbf{p}_j, h), i = 1, 2, ..., n$, if we assume all particles have the same mass, then we can simplify this equation to $C_i(\mathbf{p}) = -1 + \frac{1}{\rho_0}\sum_{j=1}^{n} W(\mathbf{p}_i - \mathbf{p}_j, h), i = 1, 2, ..., n$

        > Note that we assume that $\mathbf{p}_i, \mathbf{p}_j$ belong to existing particles
      - **Goal:** $C_i(\mathbf{p} + \Delta\mathbf{p}) = 0, i = 1, 2, ..., n$
      - **Transformation**
        - **Step 1:** Use Taylor Expansion, $C_i(\mathbf{p} + \Delta\mathbf{p}) \approx C_i(\mathbf{p}) + \frac{\partial C_i(\mathbf{p})}{\partial \mathbf{p}}\Delta\mathbf{p} = C_i(\mathbf{p}) + (\nabla_{\mathbf{p}} C_i(\mathbf{p}))^T \Delta\mathbf{p}$
        - **Step 2:** We assume that each $\Delta\mathbf{p}_i$ in $\Delta\mathbf{p}$ is along with the direction of $\nabla_{\mathbf{p}_i} C_i(\mathbf{p})$ in $\nabla_{\mathbf{p}} C_i(\mathbf{p})$, and then we get its mathematic form $\Delta\mathbf{p} \approx \nabla_{\mathbf{p}} C_i(\mathbf{p})\lambda_i$

- **Step 3:** $C_i(\mathbf{p}+\Delta\mathbf{p}) \approx C_i(\mathbf{p}) + (\nabla_{\mathbf{p}}C_i(\mathbf{p}))^T\nabla_{\mathbf{p}}C_i(\mathbf{p})\lambda_i = 0$, note that we use approximate twice during the transformation
- **Step 4:** $\lambda_i = -\frac{C_i(\mathbf{p})}{(\nabla_{\mathbf{p}}C_i(\mathbf{p}))^T\cdot\nabla_{\mathbf{p}}C_i(\mathbf{p})} = -\frac{C_i(\mathbf{p})}{\sum\limits_{j=1}^{n}(\nabla_{\mathbf{p}_j}C_i(\mathbf{p}))^T\cdot\nabla_{\mathbf{p}_j}C_i(\mathbf{p})} = -\frac{C_i(\mathbf{p})}{\sum\limits_{j=1}^{n}||\nabla_{\mathbf{p}_j}C_i(\mathbf{p})||^2}$. In case of the denominator becoming zero, we can transform the formula into $-\frac{C_i(\mathbf{p})}{\sum\limits_{j=1}^{n}||\nabla_{\mathbf{p}_j}C_i(\mathbf{p})||^2+\epsilon}$. We compute λ_i for all particles in the same way

  > **Note:** ϵ is the relaxation parameter in case of the denominator becoming zero

- **Step 5:** Plug λ_i into the formula in Form 2, we note that $\Delta\mathbf{p}_i$ appears each time we compute $\lambda_k, k=1,2,...,n$, and thus we get $\Delta\mathbf{p}_i = \sum\limits_{j=1}^{n}\lambda_j\nabla_{\mathbf{p}_i}C_j(\mathbf{p})$
- **Step 6:** According to the formula showed in the following picture, we can transform Form 5 to $\Delta\mathbf{p}_i = \lambda_i\frac{1}{\rho_0}\sum\limits_{k=1}^{n}\nabla_{\mathbf{p}_i}W(\mathbf{p}_i-\mathbf{p}_k,h) + \frac{1}{\rho_0}\sum\limits_{j\neq i}\lambda_j\nabla_{\mathbf{p}_i}W(\mathbf{p}_i-\mathbf{p}_j,h) = \frac{\lambda_i}{\rho_0}\nabla_{\mathbf{p}_i}W(\mathbf{0},h) + \frac{1}{\rho_0}\sum\limits_{j\neq i}(\lambda_i+\lambda_j)\nabla_{\mathbf{p}_i}W(\mathbf{p}_i-\mathbf{p}_j,h) = \frac{1}{\rho_0}\sum\limits_{j\neq i}(\lambda_i+\lambda_j)\nabla_{\mathbf{p}_i}W(\mathbf{p}_i-\mathbf{p}_j,h)$

  > $$\nabla_{\mathbf{p}_i}C_j(\mathbf{p}) = \begin{cases} \frac{1}{\rho_0}\sum\limits_{k=1}^{n}\nabla_{\mathbf{p}_i}W(\mathbf{p}_i-\mathbf{p}_k,h), j=i \\ \frac{1}{\rho_0}\sum\limits_{k=1}^{n}\nabla_{\mathbf{p}_i}W(\mathbf{p}_j-\mathbf{p}_k,h) = \frac{1}{\rho_0}\nabla_{\mathbf{p}_i}W(\mathbf{p}_j-\mathbf{p}_i,h) = -\frac{1}{\rho_0}\nabla_{\mathbf{p}_i}W(\mathbf{p}_i-\mathbf{p}_j,h), j\neq i \end{cases}$$

- **Conservation of Momentum:** Considering the final formula presented in Step 6, we notice $\sum\limits_i\Delta\mathbf{p}_i = \frac{1}{\rho_0}\sum\limits_i\sum\limits_{j\neq i}(\lambda_i+\lambda_j)\nabla_{\mathbf{p}_i}W(\mathbf{p}_i-\mathbf{p}_j,h) = \frac{1}{\rho_0}\sum\limits_{i,j,i\neq j}(\lambda_i+\lambda_j)\nabla_{\mathbf{p}_i}W(\mathbf{p}_i-\mathbf{p}_j,h)$. It's obvious that for pair (i,j), there always exist pair (j,i), which makes $(\lambda_i+\lambda_j)\nabla_{\mathbf{p}_i}W(\mathbf{p}_i-\mathbf{p}_j,h) + (\lambda_j+\lambda_i)\nabla_{\mathbf{p}_j}W(\mathbf{p}_j-\mathbf{p}_i,h) = \mathbf{0}$, and thus $\sum\limits_i\Delta\mathbf{p}_i = \mathbf{0}$, which ensures the conservation of momentum. In other words, regardless of the external forces, the total energy of the system will never grow.

Correction Term

- **Problem:** In Step 5 in **Transformation**, we find that formula doesn't consider the particle close to the boundaries, thus causing the density getting much smaller. So we need to adjust the formula, and that means we need to add correction term to it.
- **Analysis 1:** We can let the boundaries give particles force to push them back, and the closer particles are to the boundaries, the stronger the force is. Also, the force must satisfy the following condition: once a particle is surrounding closely by either particles or boundaries, the resultant force it receives should be zero.

- **Analysis 2:** The second condition in Analysis 1 is hard to achieve, but it can be achieved to some extent by adjusting parameters. And considering the force generated by the boundaries, we plan to assume out of the boundaries are particles arranged closely.
- **Analysis 3:** According to the latter part in Analysis 2, we can compute rest density using the similar method.
- **Step 1:** We should change the formula from the very beginning. $C_i(\mathbf{p}) = -1 + \frac{\rho_i}{\rho_0} = -1 + \frac{1}{\rho_0}(\sum_{j=1}^{n} W(\mathbf{p}_i - \mathbf{p}_j, h) + \sum_{j=1}^{m} W(\mathbf{p}_i - \mathbf{q}_j, h))$, $\mathbf{q}$ means the positions of virtual particles outside the boundaries, and all of these positions are constants.
- **Step 2:** Now we focus on the simplification of $\Delta\mathbf{p}_i$. $\Delta\mathbf{p}_i = \lambda_i \frac{1}{\rho_0}\sum_{k=1}^{n} \nabla_{\mathbf{p}_i} W(\mathbf{p}_i - \mathbf{p}_k, h) + \lambda_i \frac{1}{\rho_0}\sum_{k=1}^{m} \nabla_{\mathbf{p}_i} W(\mathbf{p}_i - \mathbf{q}_k, h) + \frac{1}{\rho_0}\sum_{j=1, j\neq i}^{n} \lambda_j \nabla_{\mathbf{p}_i} W(\mathbf{p}_i - \mathbf{p}_j, h) = \frac{1}{\rho_0}\sum_{j\neq i}(\lambda_i + \lambda_j)\nabla_{\mathbf{p}_i} W(\mathbf{p}_i - \mathbf{p}_j, h) + \frac{1}{\rho_0}\sum_{k=1}^{m} \lambda_i \nabla_{\mathbf{p}_i} W(\mathbf{p}_i - \mathbf{q}_k, h)$
- **Explanation:** You may notice the new formula doesn't ensure conservation of momentum, this is because the boundaries are designed to counter the external forces applied on particles and the repulsive forces among particles.

- **Final Equation:** $\Delta\mathbf{p}_i = \frac{1}{\rho_0}\sum_{j\neq i}(\lambda_i + \lambda_j)\nabla_{\mathbf{p}_i} W(\mathbf{p}_i - \mathbf{p}_j, h) + \frac{1}{\rho_0}\sum_{k=1}^{m} \lambda_i \nabla_{\mathbf{p}_i} W(\mathbf{p}_i - \mathbf{q}_k, h)$

- **Tensile Instability**
  - **Function:** Reduce the unexpected particles cohesion caused by the lack of neighboring particles(the following formula is **empirical**)
  - **Term:** $s_{corr} = -k(\frac{W(\mathbf{p}_i - \mathbf{p}_j, h)}{W(\Delta\mathbf{q}, h)})^n$

    Note: $|\Delta\mathbf{q}| = 0.1h ... 0.3h, k = 0.1, n = 4$ may work well
  - **Explanation:** k is positive in order to keep particles away from each other to some extent. Note that s_{corr} and Spiky_gradient both have negative sign
  - **Update Final Equation:** $\Delta\mathbf{p}_i = \frac{1}{\rho_0}\sum_{j\neq i}(\lambda_i + \lambda_j + s_{corr,\mathbf{p}})\nabla_{\mathbf{p}_i} W(\mathbf{p}_i - \mathbf{p}_j, h) + \frac{1}{\rho_0}\sum_{k=1}^{m}(\lambda_i + s_{corr,\mathbf{q}})\nabla_{\mathbf{p}_i} W(\mathbf{p}_i - \mathbf{q}_k, h)$
- **Vorticity Confinement**
  - **Function:** Introduce vorticity confinement to counter undesirable damping during Position Based methods
  - **Transformation**

- **Step 1:** $\boldsymbol{w}_i = \nabla \times \mathbf{v}$
- **Step 2:** $$\boldsymbol{w}_i = \nabla \times \sum_{j=1}^{n} \mathbf{v}_{ij} W(\mathbf{p}_i - \mathbf{p}_j, h) = \sum_{j=1}^{n} \mathbf{v}_{ij} \times \nabla W(\mathbf{p}_i - \mathbf{p}_j, h)$$
- **Step 3:** $\boldsymbol{\eta} = \nabla ||\boldsymbol{w}_i||, \mathbf{N} = \frac{\boldsymbol{\eta}}{||\boldsymbol{\eta}||}$
- **Step 4:** $\mathbf{f}_i^{\text{vorticity}} = \epsilon(\mathbf{N} \times \boldsymbol{w}_i)$

- **Detailed explanation of Step 2 in Transformation**
  - **Step 1:** For arbitrary $\mathbf{p}$ in velocity field $\mathbf{v}$, we have $\mathbf{v}_{\mathbf{p}} = \sum_{j=1}^{n} \mathbf{v}_{ij} W(\mathbf{p} - \mathbf{p}_j, h) = \sum_{j=1}^{n} \{(\mathbf{v}_{ij})_x W(\mathbf{p} - \mathbf{p}_j, h),\ (\mathbf{v}_{ij})_y W(\mathbf{p} - \mathbf{p}_j, h),\ (\mathbf{v}_{ij})_z W(\mathbf{p} - \mathbf{p}_j, h)\}$, and thus all the $\mathbf{p}$ form a velocity field

    > **Note:** Constraint of neighbors is unnecessary, cause the contribution coming from particles outside kernel radius will be regarded as zero. $\mathbf{v}_{ij} = \mathbf{v}_j - \mathbf{v}_i$. There is no strict restriction on the choice of $\mathbf{v}_{ij}$ or $\mathbf{v}_j$. Although the mathematically correct curl uses $\mathbf{v}_j$, we choose the one with better performance

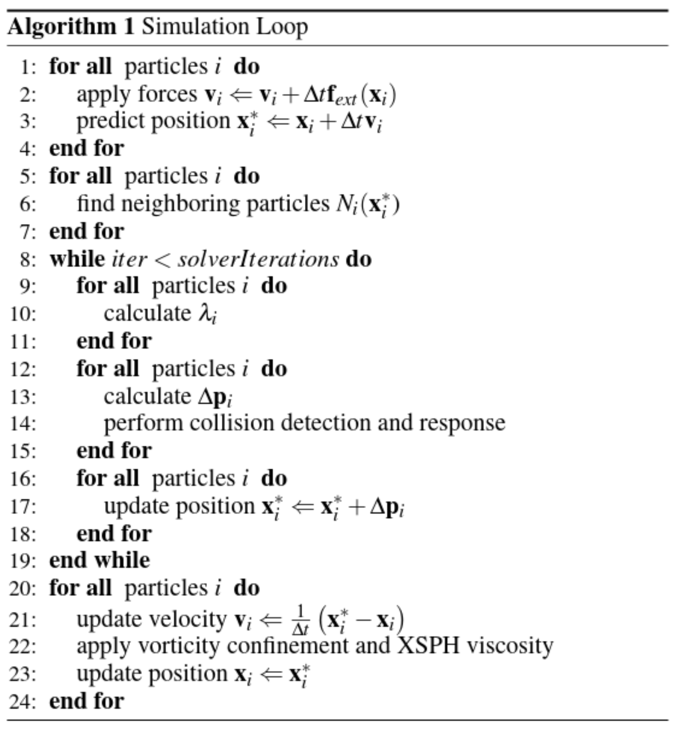
  - **Step 2:** Considering the computation of curl $\nabla \times \mathbf{v} = \nabla \times \{v_x, v_y, v_z\} = \{\frac{\partial v_z}{\partial y} - \frac{\partial v_y}{\partial z}, \frac{\partial v_x}{\partial z} - \frac{\partial v_z}{\partial x}, \frac{\partial v_y}{\partial x} - \frac{\partial v_x}{\partial y}\}$, similarly we get $\nabla \times \mathbf{v} = \sum_{j=1}^{n} \{\frac{\partial (\mathbf{v}_{ij})_z W(\mathbf{p}-\mathbf{p}_j,h)}{\partial y} - \frac{\partial (\mathbf{v}_{ij})_y W(\mathbf{p}-\mathbf{p}_j,h)}{\partial z}, \frac{\partial (\mathbf{v}_{ij})_x W(\mathbf{p}-\mathbf{p}_j,h)}{\partial z} - \frac{\partial (\mathbf{v}_{ij})_z W(\mathbf{p}-\mathbf{p}_j,h)}{\partial x}, \frac{\partial (\mathbf{v}_{ij})_y W(\mathbf{p}-\mathbf{p}_j,h)}{\partial x} - \frac{\partial (\mathbf{v}_{ij})_x W(\mathbf{p}-\mathbf{p}_j,h)}{\partial y}\}$
  - **Step 3:** The equation above can be transformed into $\nabla \times \mathbf{v} = \sum_{j=1}^{n} \mathbf{v}_{ij} \times \nabla W(\mathbf{p}_i - \mathbf{p}_j, h)$, cause $(\mathbf{v}_{ij})_x, (\mathbf{v}_{ij})_y, (\mathbf{v}_{ij})_z$ are constants
  - **Step 4:** Note that we hope the direction of $\boldsymbol{w}_i$ can correctly represent the direction of rotation, and thus the overall equation is $\nabla \times \mathbf{v} = \sum_{j=1}^{n} \mathbf{v}_{ij} \nabla W_{\mathbf{p}_j}(\mathbf{p}_i - \mathbf{p}_j, h)$
- **Detailed explanation of Step 3 in Transformation**
  - **Step 1:** For each particle i, we can compute $\boldsymbol{w}_i$ and $||\boldsymbol{w}_i||$ using the formula in Step 2 in Transformation
  - **Step 2:** And then for arbitrary position $\mathbf{p}$ in space, we can compute its $||\boldsymbol{w}||$ using the estimator as follows: $||\boldsymbol{w}|| = \sum_{j=1}^{n} ||\boldsymbol{w}_j|| W(\mathbf{p} - \mathbf{p}_j, h)$
  - **Step 3:** Further we get $\boldsymbol{\eta} = \nabla ||\boldsymbol{w}|| = \sum_{j=1}^{n} ||\boldsymbol{w}_j|| \nabla W(\mathbf{p} - \mathbf{p}_j, h)$
- **Final Equation:** $\mathbf{f}_i^{\text{vorticity}} = \epsilon(\mathbf{N} \times \boldsymbol{w}_i)$

- **XSPH Artificial Viscosity**
  - **Explanation:** Fluids have the tendency to perform coherent motion

- **Equation:** $$\mathbf{v}_i^{new} = \mathbf{v}_i + c\sum_{j=1}^{n} \mathbf{v}_{ij} \cdot W(\mathbf{p}_i - \mathbf{p}_j, h)$$

  > Note that in this equation $\mathbf{v}_{ij} = \mathbf{v}_j - \mathbf{v}_i$

- Algorithm
  - **Illustration**



Algorithm 1 Simulation Loop

```
for all particles $i$ do
    apply forces $\mathbf{v}_i \Leftarrow \mathbf{v}_i + \Delta t \mathbf{f}_{ext}(\mathbf{x}_i)$
    predict position $\mathbf{x}_i^* \Leftarrow \mathbf{x}_i + \Delta t \mathbf{v}_i$
end for
for all particles $i$ do
    find neighboring particles $N_i(\mathbf{x}_i^*)$
end for
while $iter < solverIterations$ do
    for all particles $i$ do
        calculate $\lambda_i$
    end for
    for all particles $i$ do
        calculate $\Delta\mathbf{p}_i$
        perform collision detection and response
    end for
    for all particles $i$ do
        update position $\mathbf{x}_i^* \Leftarrow \mathbf{x}_i^* + \Delta\mathbf{p}_i$
    end for
end while
for all particles $i$ do
    update velocity $\mathbf{v}_i \Leftarrow \frac{1}{\Delta t}(\mathbf{x}_i^* - \mathbf{x}_i)$
    apply vorticity confinement and XSPH viscosity
    update position $\mathbf{x}_i \Leftarrow \mathbf{x}_i^*$
end for
```

- Question
  - WHY WE PLACE NEIGHBORS FINDING OUTSIDE THE NEWTON STEP?
  - WHAT'S THE MEANING OF THE LAST STEP OF VORTICITY?
  - PERFORMANCE OPTIMIZATION

Screen space fluid rendering

rendering

- Theory
  - 增大粒子半径（观感上可以重叠，甚至可以说必须重叠，否则纹理生成时空隙较大，即使使用滤波或模糊都无法弥补）
  - **Get Texture**
    - **Depth Texture:** 将DepthBuffer中的值提取到纹理即可。
    - **Thickness Texture:** 开启Blend模式，关闭DepthTest，Blend模式中粒子亮度的设置影响层次感以及效果，亮度越低层次感越好，效果越差；反之层次感越差，效果越好。还需要注意混合时需要正确计算粒子厚度？（非必要）。面剔除（优化性能）
    - **Normal Texture**
  - **Smooth Texture**

        - **Smoothed Depth**
        - **Smoothed Thickness**
        - **Method:** bilateral filter/gaussian blur
        - **Key:** Kernel类型，Kernel半径
    - **Rendering**
        - 借助之前步骤得到的纹理通过两个三角形直接渲染即可（rasterization传统成像）
        - 自身的颜色由Smoothed Thickness决定，渲染过程使用NormalTexture，SmoothedDepth
- **Note:** gaussian blur与bilateral filter对比：bilateral filter可以更好地保留边缘信息，按需选用二者