Project 6 Particle Simulation 技術文件

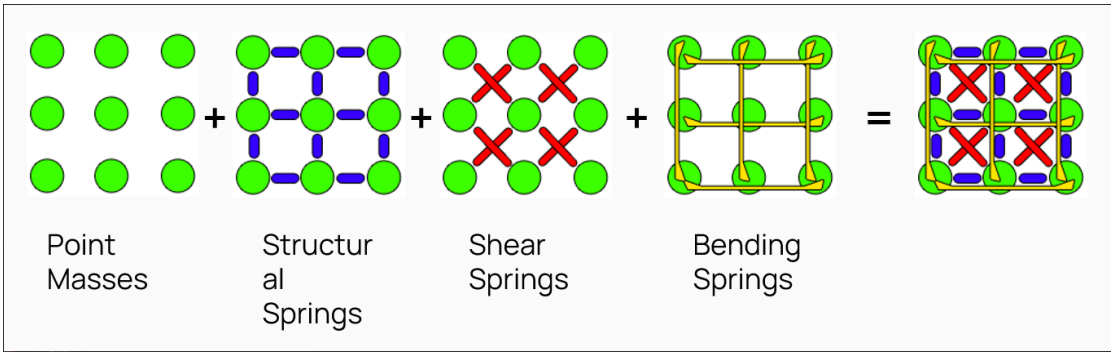
(一)、 布料

每塊布料用一個 ClothSystem.cs 來管理。

```
public List<GameObject> Particles = new List<GameObject>();
public List<ParticleCollider> Colliders = new List<ParticleCollider>();
public List<Vector3> Vertexes = new List<Vector3>();
public List<Vector2> UVs = new List<Vector2>();
public List<int> TrianglesIndexes = new List<int>();
```

```
private List<SpringSystem> springArray = new List<SpringSystem>();
```

Particles 就是所有粒子，Collders 粒子的碰撞組建，Vertexes 是要計算粒子的位置，UVs 與 TriangleIndexes 是為了使用材質。而布料可以視為好幾個粒子用彈簧連接起來，所以還有一個 springArray 來管理所有彈簧。



彈簧使用上圖的 Mass-spring model 架構。

```
int NextIndex;
// Structural Springs
// 向上
NextIndex = index + 1;
// 確保在同一行
if (NextIndex / SideCount == index / SideCount)
{
    springArray.Add(new SpringSystem(index, NextIndex, UnitDistance));
    NewLineRenderer(parent, structSpringMat);
}
// 向右
NextIndex = index + SideCount;
if (NextIndex / SideCount < SideCount)
{
    springArray.Add(new SpringSystem(index, NextIndex, UnitDistance));
    NewLineRenderer(parent, structSpringMat);
}
```

Structural Spring，連接右方和上方。

```
// Shear Springs
// 右上
NextIndex = index + SideCount + 1;
// 避免超出邊界且要在隔壁
if (NextIndex / SideCount < SideCount && NextIndex / SideCount == index / SideCount + 1)
{
    springArray.Add(new SpringSystem(index, NextIndex, UnitDistance * Mathf.Sqrt(2)));
    NewLineRenderer(parent, shearSpringMat);
}
// 左上
NextIndex = index - SideCount + 1;
if (NextIndex > 0 && NextIndex / SideCount < SideCount && NextIndex / SideCount == index / SideCount - 1)
{
    springArray.Add(new SpringSystem(index, NextIndex, UnitDistance * Mathf.Sqrt(2)));
    NewLineRenderer(parent, shearSpringMat);
}
```

Shear Spring 連接右上和左上。

```
// Bending Springs
// 向上
NextIndex = index + 2;
if (NextIndex < SideCount)
{
    springArray.Add(new SpringSystem(index, NextIndex, UnitDistance * 2));
    NewLineRenderer(parent, bendSpringMat);
}
// 向右
NextIndex = index + SideCount * 2;
if (NextIndex / SideCount < SideCount)
{
    springArray.Add(new SpringSystem(index, NextIndex, UnitDistance * 2));
    NewLineRenderer(parent, bendSpringMat);
}
```

Bend Spring 連接上上和右右。

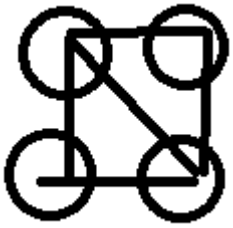
```
float u = (float)i / (SideCount - 1);
float v = (float)j / (SideCount - 1);
UVs.Add(new Vector2(u, v));
```

因為布料是正方形，uv 左下角 00 右上角 11，所以直接索引值除以邊長。

```
for (int i = 0; i < SideCount - 1; i++)
{
    for (int j = 0; j < SideCount - 1; j++)
    {
        // Index 資訊
        int index = i * SideCount + j;
        TrianglesIndexes.Add(index);
        TrianglesIndexes.Add(index + 1);
        TrianglesIndexes.Add(index + SideCount);

        TrianglesIndexes.Add(index + 1);
        TrianglesIndexes.Add(index + 1 + SideCount);
        TrianglesIndexes.Add(index + SideCount);
    }
}
```

為了貼材質要三角形，相鄰三個點組成三角形。



(二)、 布料模擬

彈簧之間會互相拉扯，所以根據 Damped Spring 公式計算力。

```
public Vector3 CountForce(Vector3 startSpeed, Vector3 endSpeed, Vector3 startPos, Vector3 endPos)
{
    // Damped spring
    float distance = Vector3.Distance(startPos, endPos);
    return -(Ks * (distance - OriginLength) + Kd * Vector3.Dot(startSpeed - endSpeed, startPos - endPos) / distance)
        * (startPos - endPos) / distance;
}
```

```
speedArray[i] += Vector3.up * Gravity * Time.fixedDeltaTime;
```

重力，v = gt。

接著就是決定用哪種方法，Euler 或 Runge Kutta。

```
private Vector3 EulerMethod(int index, float time)
{
    // x = vt
    return speedArray[index] * time;
}
```

Euler 直接計算，x1 = x0 + hf(x, t)

$$
\begin{aligned}
k_1 &= hf(x_n, y_n) \\
k_2 &= hf(x_n + \frac{h}{2}, y_n + \frac{k_1}{2}) \\
k_3 &= hf(x_n + \frac{h}{2}, y_n + \frac{k_2}{2}) \\
k_4 &= hf(x_n + h, y_n + k_3) \\
y_{n+1} &= y_n + \frac{k_1}{6} + \frac{k_2}{3} + \frac{k_3}{3} + \frac{k_4}{6} + O(h^5)
\end{aligned}
$$

Runge-Kutta 依照上面公式計算。

```
// K2
Vector3 appendSpeedK2 = Vector3.up * Gravity * Time.deltaTime / 2;
for (int i = 0; i < springArray.Count; i++)
{
    if (springArray[i].ConnectIndexStart == index || springArray[i].ConnectIndexEnd == index)
    {
        // 拿 Index
        int StartIndex = springArray[i].ConnectIndexStart;
        int EndIndex = springArray[i].ConnectIndexEnd;

        // 拿資料
        Vector3 StartSpeed = speedArray[StartIndex];
        Vector3 EndSpeed = speedArray[EndIndex];
        Vector3 StartPos = Vertexes[StartIndex] + EulerMethod(StartIndex, time / 2);
        Vector3 EndPos = Vertexes[EndIndex] + EulerMethod(EndIndex, time / 2);

        Vector3 tempForce = springArray[i].CountForce(StartSpeed, EndSpeed, StartPos, EndPos);

        if (index == springArray[i].ConnectIndexStart)
            appendSpeedK2 += tempForce / Mass * Time.fixedDeltaTime;
        else
            appendSpeedK2 -= tempForce / Mass * Time.fixedDeltaTime;
    }
}
Vector3 k2 = EulerMethodWithAppendForce(index, time / 2, appendSpeedK2);
```

```
// K3
Vector3 appendSpeedK3 = Vector3.up * Gravity * Time.deltaTime / 2;
for (int i = 0; i < springArray.Count; i++)
{
    if (springArray[i].ConnectIndexStart == index || springArray[i].ConnectIndexEnd == index)
    {
        // 拿 Index
        int StartIndex = springArray[i].ConnectIndexStart;
        int EndIndex = springArray[i].ConnectIndexEnd;

        // 拿資料
        Vector3 StartSpeed = speedArray[StartIndex];
        Vector3 EndSpeed = speedArray[EndIndex];
        Vector3 StartPos = Vertexes[StartIndex] + EulerMethodWithAppendForce(index, time / 2, appendSpeedK2);
        Vector3 EndPos = Vertexes[EndIndex] + EulerMethodWithAppendForce(index, time / 2, appendSpeedK2);

        Vector3 tempForce = springArray[i].CountForce(StartSpeed, EndSpeed, StartPos, EndPos);

        if (index == springArray[i].ConnectIndexStart)
            appendSpeedK3 += tempForce / Mass * Time.fixedDeltaTime;
        else
            appendSpeedK3 -= tempForce / Mass * Time.fixedDeltaTime;
    }
}
Vector3 k3 = EulerMethodWithAppendForce(index, time / 2, appendSpeedK3);
```

```
// K4
Vector3 appendSpeedK4 = Vector3.up * Gravity * Time.deltaTime / 2;                                  // V =
for (int i = 0; i < springArray.Count; i++)
    if (springArray[i].ConnectIndexStart == index || springArray[i].ConnectIndexEnd == index)
    {
        // 拿 Index
        int StartIndex = springArray[i].ConnectIndexStart;
        int EndIndex = springArray[i].ConnectIndexEnd;

        // 拿資料
        Vector3 StartSpeed = speedArray[StartIndex];
        Vector3 EndSpeed = speedArray[EndIndex];
        Vector3 StartPos = Vertexes[StartIndex] + EulerMethodWithAppendForce(index, time, appendSpeedK3);
        Vector3 EndPos = Vertexes[EndIndex] + EulerMethodWithAppendForce(index, time, appendSpeedK3);

        Vector3 tempForce = springArray[i].CountForce(StartSpeed, EndSpeed, StartPos, EndPos);

        if (index == springArray[i].ConnectIndexStart)
            appendSpeedK4 += tempForce / Mass * Time.fixedDeltaTime;
        else
            appendSpeedK4 -= tempForce / Mass * Time.fixedDeltaTime;
    }

Vector3 k4 = EulerMethodWithAppendForce(index, time, appendSpeedK4);
return k1 / 6.0f + k2 / 3.0f + k3 / 3.0f + k4 / 6.0f;
```

算完之後計算碰撞，對粒子前進方向打一條射線，看看是不是撞到東西，有的話直接速度= 0，會黏在表面上。