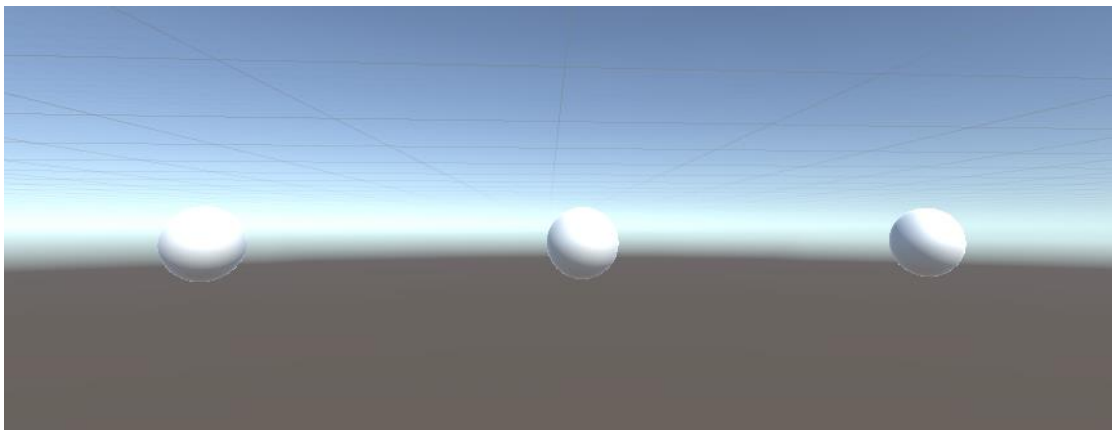
GP2019-hw5

516030910435

1. 单摆模拟(在场景 SampleSence 中)

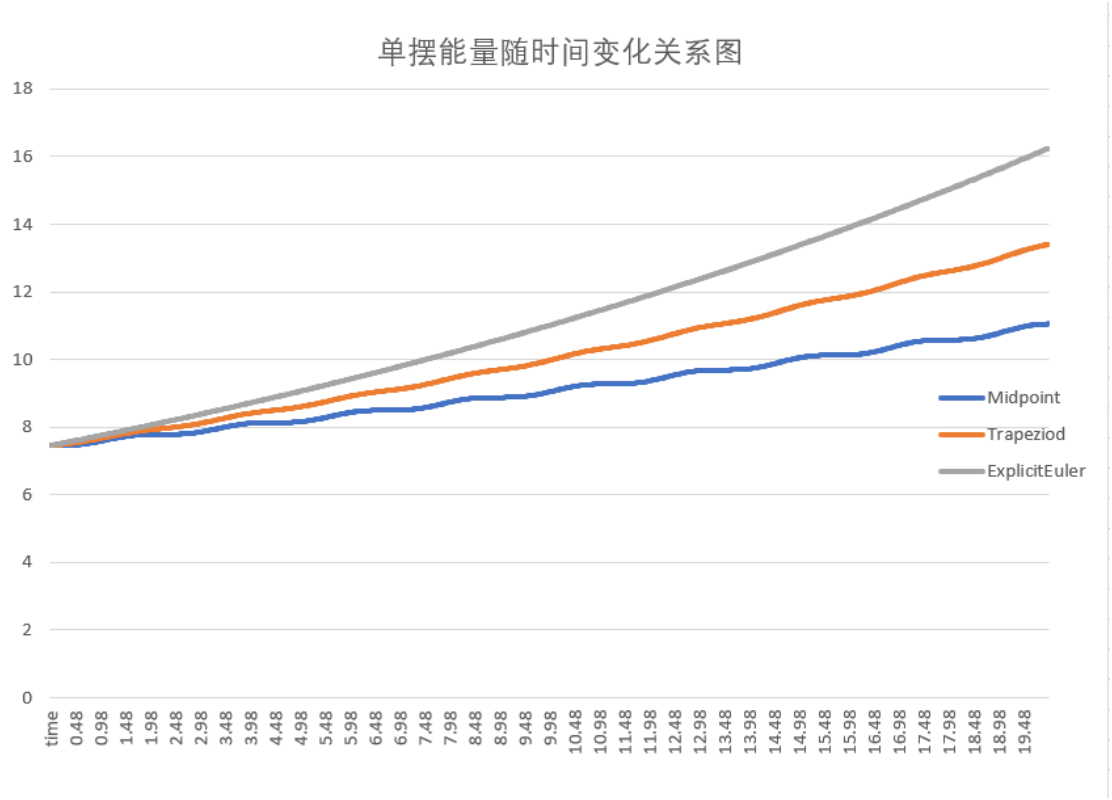


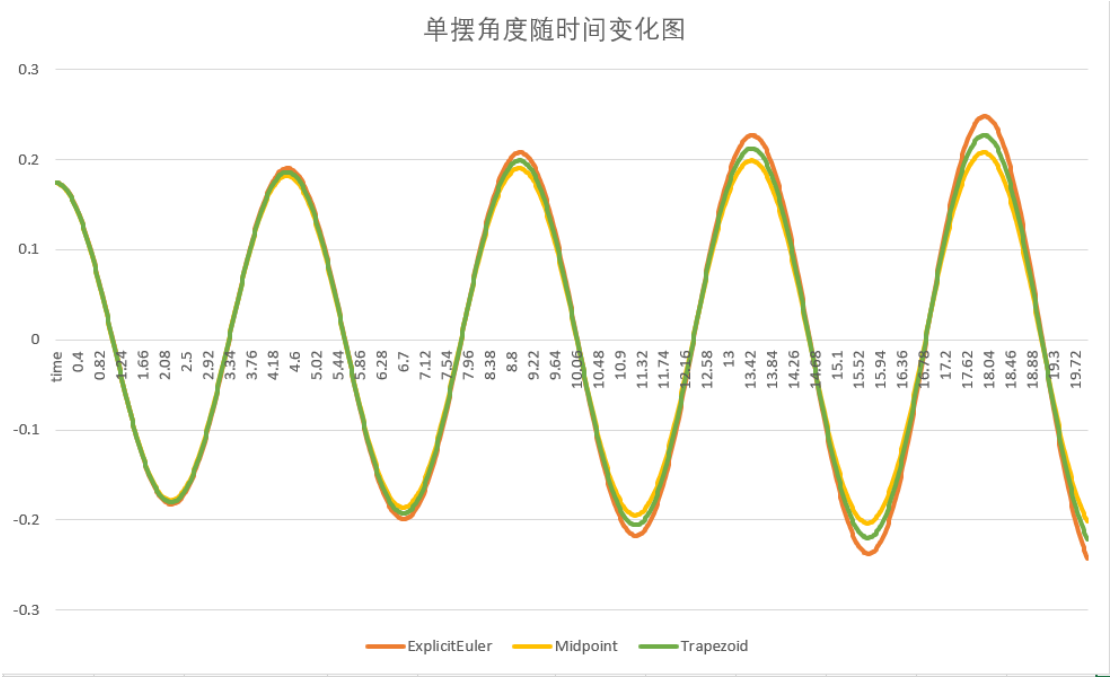
左侧单摆：Trapezoid
中间单摆：Explicit Euler
右侧单摆：Midpoint

单摆模拟：
摆长 ：5
重力加速度：9.81

Explicit Euler：模拟的时候用前一时刻的角速度进行计算
Midpoint：用中间时刻的角速度进行计算
Trapezoid：角速度为 Explicit Euler 与 Midpoint 的平均值

绘图时初始角度选择为 10°，单摆小球质量为 10


单摆能量随时间变化关系图
18
16
14
12
10
8
6
4
2
0
time
0.48
0.98
1.48
1.98
2.48
2.98
3.48
3.98
4.48
4.98
5.48
5.98
6.48
6.98
7.48
7.98
8.48
8.98
9.48
9.98
10.48
10.98
11.48
11.98
12.48
12.98
13.48
13.98
14.48
14.98
15.48
15.98
16.48
16.98
17.48
17.98
18.48
18.98
19.48
Midpoint
Trapeziod
ExplicitEuler


单摆角度随时间变化图
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
time
0.4
0.82
1.24
1.66
2.08
2.5
2.92
3.34
3.76
4.18
4.6
5.02
5.44
5.86
6.28
6.7
7.12
7.54
7.96
8.38
8.8
9.22
9.64
10.06
10.48
10.9
11.32
11.74
12.16
12.58
13
13.42
13.84
14.26
14.68
15.1
15.52
15.94
16.36
16.78
17.2
17.62
18.04
18.46
18.88
19.3
19.72
ExplicitEuler
Midpoint
Trapezoid

2. 头发的模拟（在场景 hair 中）

定义类

```
public class Node
{
        public Vector3 p0, p1; // 前帧/本帧的位置
        public float length;   // 和上一节点的止动长度
};

public class Strand
{
    public int nodeStart, nodeEnd; // 此发束中，节点数组的起始和结束索引
    public Vector3 rootP;             // 发根的局部坐标（相对于头的变换）
};
```

设置参数

```
private LineRenderer lr;
[SerializeField, Range(1, 100)] public int pointcount = 20;     //节点数量
private  int strandcount = 1;                                   //头发数量
[SerializeField, Range(0.01f, 1f)] public float pointdis = 0.3f;//长度约束

private  Vector3 a;      //加速度
private Vector3 g;       //重力加速度
public float m;          //头发质量
public Vector3 Force;    //除重量之外的外力

[SerializeField, Range(0.9f, 1f)] public float damping; //阻尼系数
public Vector3 headToWorld;                             //脑袋坐标

private List<Node> nodes;           //节点集
private List<Strand> strands;       //发束集
```

根据《爱丽丝的头发》中的伪代码进行模拟

```
public void simulater( float damping, float dt)
{
    for(int n = 0; n < nodes.Count; n++)
    {
        a = Force / m + g;
        Vector3 p2 = Verlet(nodes[n].p0, nodes[n].p1, damping, a, dt);
        nodes[n].p0 = nodes[n].p1;
        nodes[n].p1 = p2;
    }

    for(int s = 0; s < strands.Count; s++)
    {
        int start = strands[s].nodeStart;
        int end = strands[s].nodeEnd;
        for (int i = start ; i < end; i++)
        {
            Node na = nodes[i];
            Node nb = nodes[i + 1];

            // 碰撞检测和决议
            //nb.p1 = collideSphere(sphere, nb.p1)
            // 长度约束
            na.p1 = lengthConstraint(na.p1, nb.p1, nb.length)[0];
            nb.p1 = lengthConstraint(na.p1, nb.p1, nb.length)[1];

            nodes[start].p1 = headToWorld + strands[s].rootP;


        }
    }

}
```

其中函数

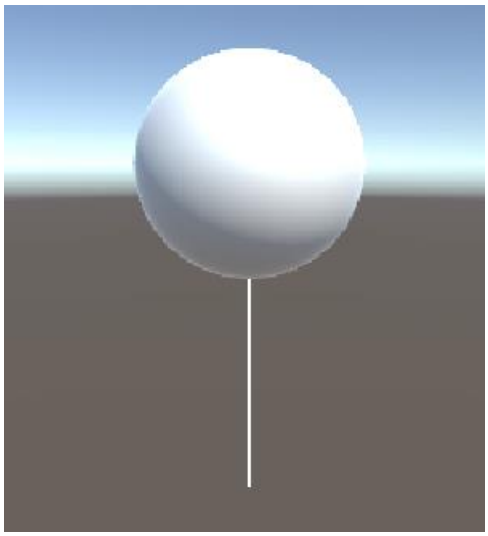
Verlet

```
public Vector3 Verlet(Vector3 p0,Vector3 p1,float damping,Vector3 a,float dt)
{
    Vector3 p2 = p1 + damping * (p1 - p0) + a * dt * dt;
    return p2;
}
```

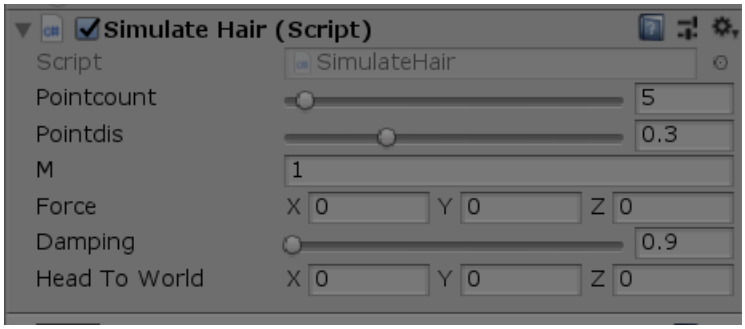
长度约束

```
public List<Vector3> lengthConstraint(Vector3 na,Vector3 nb,float len)
{
    float dis = (na - nb).magnitude;
    Vector3 a = na + (nb - na) * (dis - pointdis) / dis / 2.0f;
    Vector3 b = nb - (nb - na) * (dis - pointdis) / dis / 2.0f;
    List<Vector3> res = new List<Vector3>();
    res.Add(a);
    res.Add(b);
    return res;
}
```

模拟运行（只实现了一根头发）



通过 GUI 界面改变头发参数



Point count 和 pointdis 用于改变头发的长度

M 改变头发质量

Force 施加外力

Damping 改变阻尼系数

Head to world 改变发根位置

施加外力