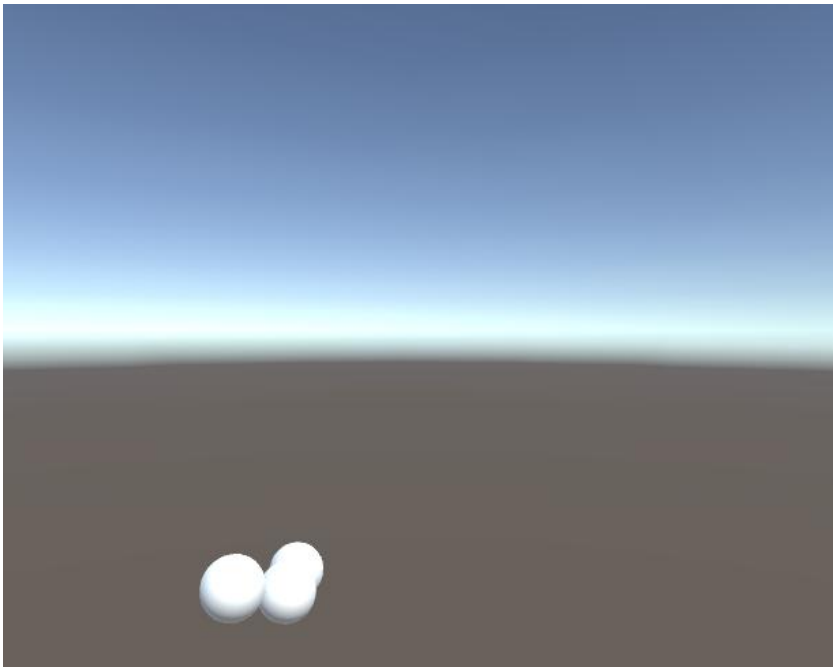
一、单摆模拟

代码部分：

```
//Explicit Euler
k1 = currentOmega;
l1 = -(gravity / len) * Mathf.Sin(currentAngle);
//Midpoint
k2 = currentOmega + delta * l1 / 2;
l2 = -(gravity / len) * Mathf.Sin(currentAngle + delta * k1 / 2);
//Trapezoid
k3 = currentOmega + delta * l1 / 2;
l3 = (-(gravity / len) * Mathf.Sin(currentAngle + delta * k1) + l1) / 2;

if (movementType == MovementType.Explicit_Euler)
{
    currentAngle += delta * k1;
    currentOmega += delta * l1;
}
else if (movementType == MovementType.Midpoint)
{
    currentAngle += delta * k2;
    currentOmega += delta * l2;
}
else if (movementType == MovementType.Trapezoid)
{
    currentAngle += delta * k3;
    currentOmega += delta * l3;
}
```

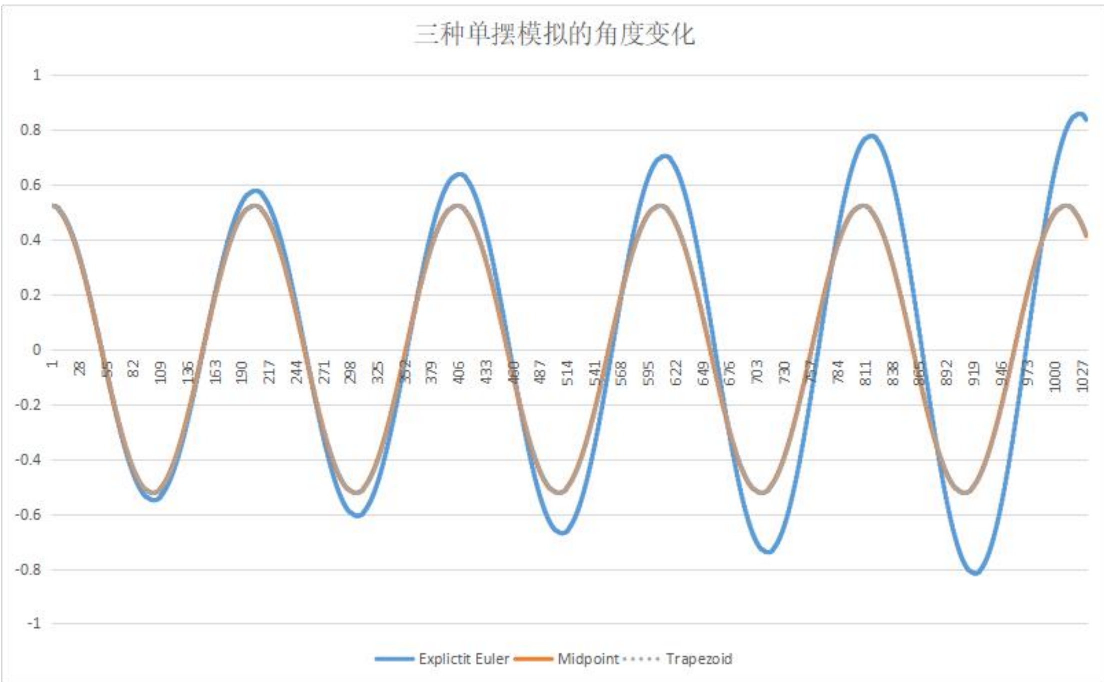
其中 k1 和 l1 是显式欧拉的参数，k2 和 l2 是中点法的参数，k3 和 l3 是四边形法的参数

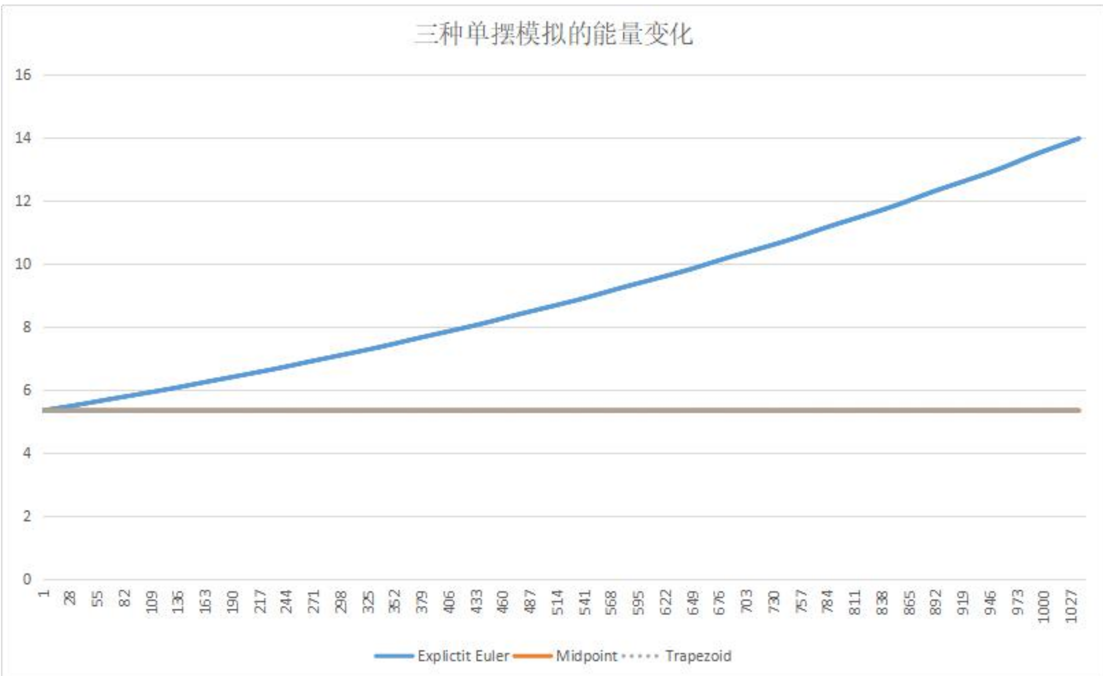


从前往后分部是显式欧拉，中点法，四边形法的单摆模拟，可以发现显示欧拉法的摆动幅度与另外两种明显有差异

```
//float E = currentOmega * currentOmega * len * len / 2 + gravity * len * (1 - Mathf.Cos(currentAngle));
//WriteMessage(E.ToString());
```

在运行时将数据导出进行分析，角度和能量的分析如下图所示





可以发现，中点法与四边形法的的图像基本重合，误差较小，而显示欧拉法的误差较大

二、头发模拟

一）实现基于 Verlet 方法的质点模拟

主要分为三部分

```
void Simulate()
{
    for (int i = 0; i < 20; i++)
    {
        //Verlet积分
        Vector3 p2 = Verlet(strand.nodes[i].p0, strand.nodes[i].p1, damping, a, Time.deltaTime);
        //替换
        Node tmpNode;
        tmpNode.p0 = strand.nodes[i].p1;
        tmpNode.p1 = p2;
        strand.nodes[i] = tmpNode;
    }

    for (int i = 0; i < 19; i++)
    {
        Node currentNode = strand.nodes[i];
        Node nextNode = strand.nodes[i + 1];
        //长度约束
        Node result = LengthConstraint(currentNode.p1, nextNode.p1, nodeLength);
        currentNode.p1 = result.p0;
        nextNode.p1 = result.p1;
        //碰撞约束
        currentNode.p1 = CollideConstraint(head, currentNode.p1);

        strand.nodes[i] = currentNode;
        strand.nodes[i + 1] = nextNode;
    }
    //固定发根
    Node rootNode;
    rootNode.p0 = head.TransformPoint(strand.rootP);
    rootNode.p1 = head.TransformPoint(strand.rootP);
    strand.nodes[0] = rootNode;
}
```

1）实现 Verlet 积分

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

2）进行长度约束和碰撞约束

```
Node LengthConstraint(Vector3 x1, Vector3 x2, float maxLength) //长度约束
{
    float realLen = Vector3.Magnitude(x1 - x2);
    Vector3 x3 = x1 + (x2 - x1) * (realLen - maxLength) / (2 * realLen);
    Vector3 x4 = x2 - (x2 - x1) * (realLen - maxLength) / (2 * realLen);
    Node tmp = new Node();
    tmp.p0 = x3;
    tmp.p1 = x4;
    return tmp;
}
```

```
Vector3 CollideConstraint(Transform head, Vector3 x) //碰撞约束
{
    Vector3 center = head.position;
    float R = head.localScale.x / 2 + headExpend;

    float distance = Vector3.Magnitude(center - x);
    if(distance < R)
    {
        return center + (x - center) * R / distance;
    }

    return x;
}
```

3）固定发根

```
//固定发根
Node rootNode;
rootNode.p0 = head.TransformPoint(strand.rootP);
rootNode.p1 = head.TransformPoint(strand.rootP);
strand.nodes[0] = rootNode;
```

二）通过 LineRender 渲染头发

```
private void Update()
{
    Simulate();

    Vector3[] postions = new Vector3[20];
    for (int i = 0; i < 20; i++)
    {
        postions[i] = strand.nodes[i].p1;
    }
    lineRen.SetPositions(postions);
}
```

三）通过 HairController 控制头发和风向

将 Hair 拖进 Project 视图，制成 Prefab，然后编写 HairController 控制头发

可控制参数：

```
public GameObject prefab;                   //头发预制体
public Transform head;                      //头部
public float nodeLength = 0.07f;            //节点定长
public float damping = 0.95f;               //阻尼系数
public int num = 30;                        //头发密度
private Vector3 a = new Vector3(0, -9.8f, 0);   //加速度
private float headExpend = 0.1f;            //头部扩张
```

在头部半球根据密度固定发根：

```
void Start()
{
    float R = head.localScale.x / 2 + headExpend;
    Vector3 center = head.position;

    for (int i = 0; i <= num; i++)
    {
        for(int j = 0; j <= num / 2; j++)
        {
            Vector3 root = new Vector3(0, 0, 0);
            root.x = 2 * R / num * i - R;
            root.z = 2 * R / num * j;
            float length = Vector3.Magnitude(center - root);
            if(length <= R)
            {
                root.y = Mathf.Sqrt(R * R - length * length);
                GameObject go = Instantiate(prefab) as GameObject;
                go.GetComponent<HairRender>().head = head;
                go.GetComponent<HairRender>().hairRoot = root;
                go.GetComponent<HairRender>().nodeLength = nodeLength;
                go.GetComponent<HairRender>().damping = damping;
                hairs.Add(go);
            }
        }
    }
}
```

控制风向：

```
if (Input.GetKey("z"))
    a.z += 0.3f;
if (Input.GetKey("c"))
    a.z -= 0.3f;

foreach (GameObject e in hairs)
{
    e.GetComponent<HairRender>().a = a;
}
```

四）通过 HeadController 控制头部运动

```
//WASD控制上下左右
Vector3 pos = transform.position;
if (Input.GetKey("w"))
    pos.z += speed;
if (Input.GetKey("s"))
    pos.z -= speed;
if (Input.GetKey("a"))
    pos.x -= speed;
if (Input.GetKey("d"))
    pos.x += speed;
transform.position = pos;
```

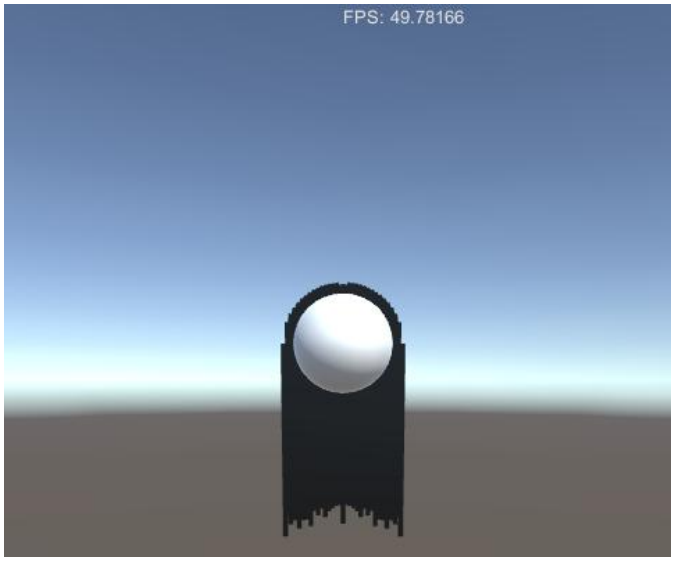
```
//QE控制头部旋转
if (Input.GetKey("q"))
    angle.y -= speed;
else if (Input.GetKey("e"))
    angle.y += speed;
else
    angle = new Vector3(0, 0, 0);
transform.Rotate(angle);
//R归位
if (Input.GetKey("r"))
{
    transform.position = originPosition;
    transform.rotation = originAngle;
}
```

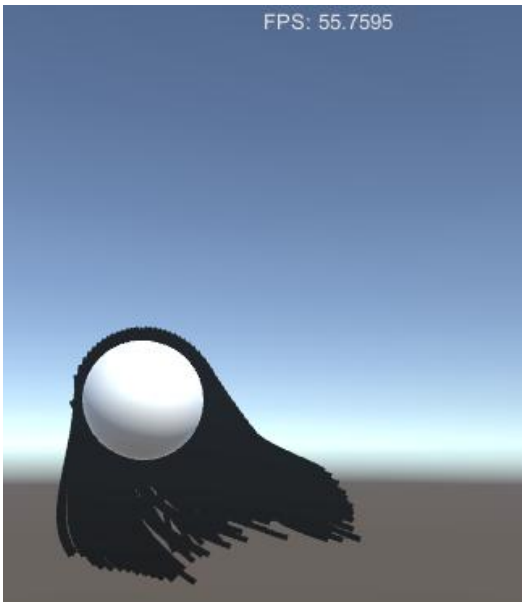
五）显示帧率

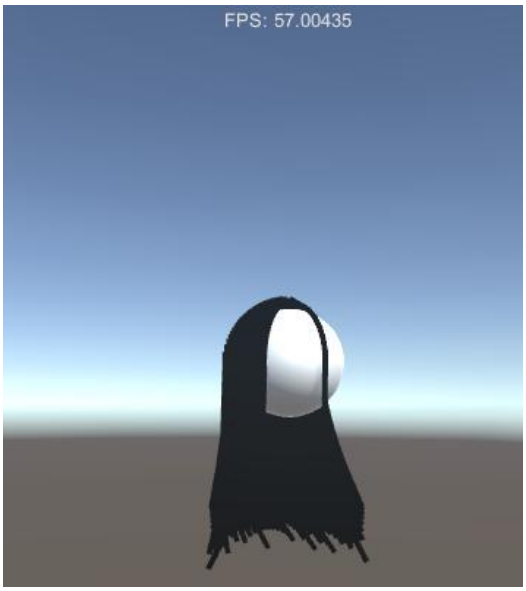
```
void Update()
{
    m_FrameUpdate++;
    if (Time.realtimeSinceStartup - m_LastUpdateShowTime >= m_UpdateShowDeltaTime)
    {
        m_FPS = m_FrameUpdate / (Time.realtimeSinceStartup - m_LastUpdateShowTime);
        m_FrameUpdate = 0;
        m_LastUpdateShowTime = Time.realtimeSinceStartup;
    }
}

void OnGUI()
{
    GUI.Label(new Rect(Screen.width / 2, 0, 100, 100), "FPS: " + m_FPS);
}
```

六）最终效果




FPS: 55.7595


FPS: 57.00435