R/ E/ P/ O/ R/ T

Self-scoring table :

Score	report	Create	Attach	Nail	Drag	Point damping	Total 10/10
	1/1	1/1	1/1	1/1	1/1	1/1	
	Damped spring	Unstable	Stable	Falling			
	1/1	1/1	1/1	1/1			

제출일: 20. 06. 05

과제명: Programming Assignment #2

과목 : 컴퓨터애니메이션

학과 : 소프트웨어 학부

학번 : 2015726076

이름 : 김현구

1. Create

```
161   void addPoint()
162   {
163       p.push_back(ee_goal);
164       v.push_back(Vector3f(0,0,0));
165
166       //collision
167       contact.push_back(false);
168       contactN.push_back(Vector3f(0, 0, 0));
169       constrained.push_back(false);
170
171       //selected check value
172       s.push_back(false);
173   }
```

A 키를 누르고 파티클을 추가합니다.

P 에 좌표가 저장되고, v, collision, selected 관련 벡터들도 추가해줍니다.

2. Attach

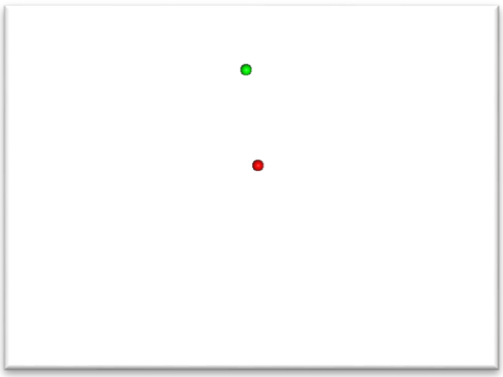
```
195  void addEdge(int idx)
196  {
197      switch(selectedCount)
198      {
199          case 1:
200          {
201              e1.push_back(idx);
202              break;
203          }
204          case 2:
205          {
206              e2.push_back(idx);
207              l.push_back((p[e1[e1.size()-1]] - p[e2[e2.size()-1]]).norm());
208
209              //init
210              selectedCount = 0;
211              s.clear();
212              s.resize(p.size(), false);
213              rebuildSpringK();
214              break;
215          }
```

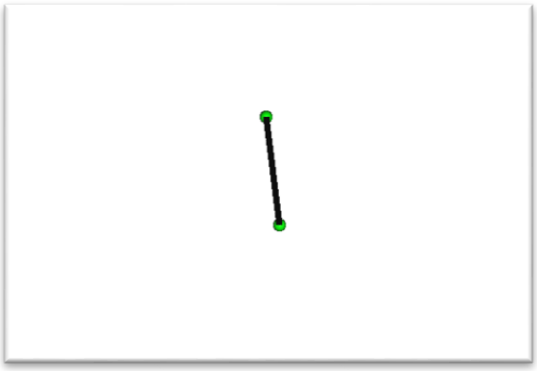
```
else if(s[i])
    drawSphere(radius, Vector3f(1, 0, 0), 20);
```

T 키를 누르고 두 파티클을 연결합니다.

선택할 때마다 selectedCount 가 올라가며, 2 가 될 경우 연결합니다.

Vector<bool> s 를 이용해 색을 빨간색으로 표시합니다.





3. Nail

```
for (int i = 0; i < p.size(); i++)
{
    //constraint
    if (constrained[i]) continue;

    //contact force
    if (contact[i]) f[i] -= contactN[i].dot(f[i]) * contactN[i];

    switch (intMethod)
    {
    case EULER:
        p[i] += h * v[i];
        v[i] += h * f[i] / m;
        break;
    case MODIFIED_EULER:
        v[i] += h * f[i] / m;
        p[i] += h * v[i];
        break;
    }
}
```

N 을 누르고 클릭하면 가까운 점을 선택하고 constrained 벡터에 추가합니다.

추가된 점은 노란색으로 표시되고, 파티클의 속도와 위치를 계산하는 연산에서 제외됩니다.

4. Drag

```
if (action == GLFW_RELEASE)
    isDragged = false;
```

D 키를 누르고 클릭했을 때 glfwSetCursorPosCallback 함수를 이용해서 커서의 위치를 가져오고,

isDragged 상태일 때 클릭한 지점에서 가장 가까운 점을 드래그 합니다.

5. Damping

```
//in force
for (int i = 0; i < e2.size(); i++)
{
    Vector3f v_i = p[e1[i]] - p[e2[i]];
    float L_i = v_i.norm();
    Vector3f f_i = k[i] * (L_i - l[i]) * v_i / L_i;

    f[e2[i]] += f_i;
    f[e1[i]] -= f_i;

    //add damping force
    if(pointDamping)
    {
        f[e2[i]] -= dampValue * v[e2[i]];
        f[e1[i]] -= dampValue * v[e1[i]];
    }
    if(dampedSpring)
    {
        Vector3f n = v_i / L_i; //nij
        Vector3f vel = v[e1[i]] - v[e2[i]]; //vij
        f[e2[i]] += dampValue * n.dot(vel) * n;
        f[e1[i]] -= dampValue * n.dot(vel) * n;
    }
}
```

Inner force 를 계산할 때, 해당 파티클에 반하는 damping force 를 줍니다.

5-1. point damping

Damp coefficient 와 파티클의 속도를 곱해 계산합니다.

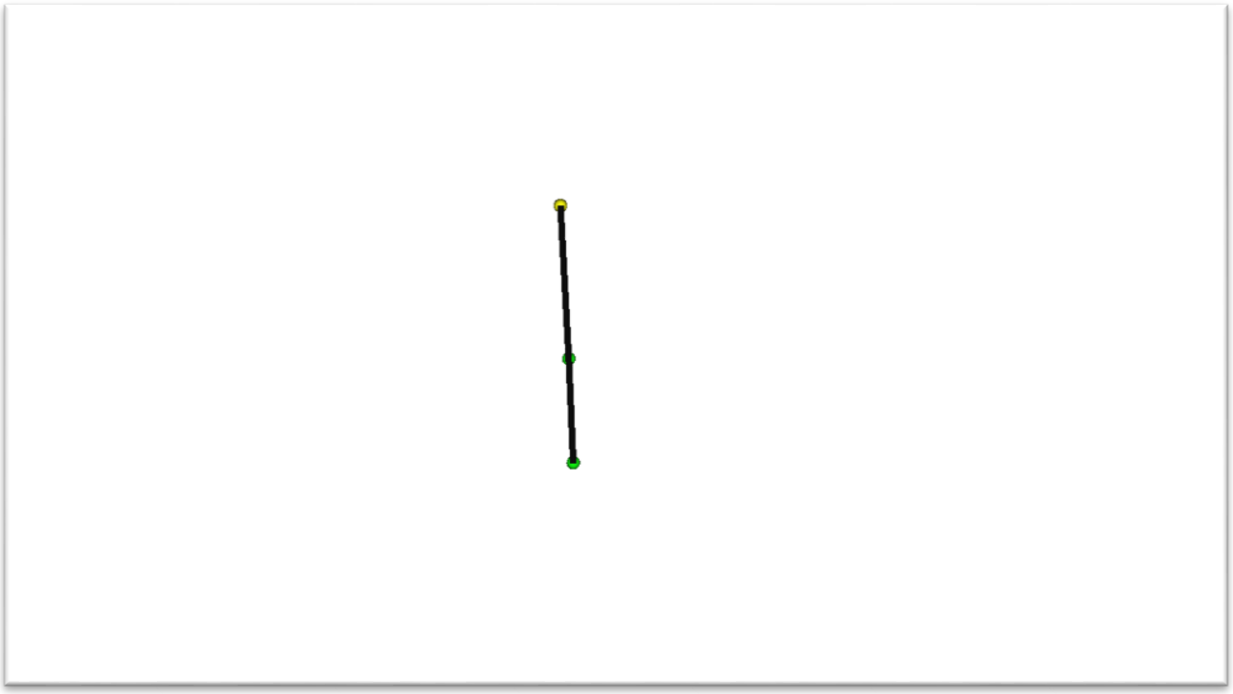
Point damping 은 스프링의 방향(v_i)에 영향을 받지 않고 파티클의 속도에만 영향을 받습니다.

5-2. damped spring

Spring 의 힘의 방향의 반대 방향으로 저항을 줍니다.

현재 파티클의 속도(vel)와 스프링의 방향(n)을 내적하여 크기를 구하고, 스프링의 방향의 힘만 가져와서 연결된 양쪽 파티클에 줍니다.

코드에서는 e1 을 기준으로 damped spring force 를 구해 e2 에 해당하는 파티클에는 부호를 반대로 주었습니다.



6. Stable/Unstable

```
    //Spring constants
case GLFW_KEY_UP: k0 = min(k0 + 0.1, 10.0); rebuildSpringK(); cout << "k : " << k0 << endl; break;
case GLFW_KEY_DOWN: k0 = max(k0 - 0.1, 0.1); rebuildSpringK(); cout << "k : " << k0 << endl; break;
case GLFW_KEY_RIGHT: N_SUBSTEPS = min(N_SUBSTEPS + 5.0, 1000.0); h = 1.0 / 60.0 / N_SUBSTEPS; cout << "h : " << h << endl; break;
case GLFW_KEY_LEFT: N_SUBSTEPS = max(N_SUBSTEPS - 5.0, 1.0); h = 1.0 / 60.0 / N_SUBSTEPS; cout << "h : " << h << endl; break;
```

방향키를 통해 위, 아래키로 spring 상수 K 를,

좌, 우로 SUBSTEP 을 변화시킬 수 있습니다.

```
switch (intMethod)
{
case EULER:
    p[i] += h * v[i];
    v[i] += h * f[i] / m;
    break;
case MODIFIED_EULER:
    v[i] += h * f[i] / m;
    p[i] += h * v[i];
    break;
}
```

euler 를 이용할 때, 불안정성이 커져 K 가 높다면 발산하게 됩니다.

H 를 작게 함으로 이를 안정시킬 수 있습니다.

7. falling

```
    //constraints remove
case GLFW_KEY_R: constrained.clear(); constrained.resize(p.size(), false); cout << "constrait removed" << endl; break;
```

R 키를 눌르면 constrained 벡터를 초기화하여 nail 된 파티클들을 모두 해제할 수 있습니다.