컴퓨터애니메이션 (CA)

#HW3 Mass Spring System

Self-scoring table

	report	create	attach	nail	drag	point damping	damped Spring	unstable	stable	falling	total
Score	1/1	1/1	1/1	1/1	1/1	1/1	1/1	1/1	1/1	1/1	10/10

2018707068 김경환

KwangWoon University

Code Analysis:
이번 과제에서는 복잡한 Mass Spring Systemd을 시뮬레이션하기 위한 2D interface를 구현하고 시뮬레이션한다.

먼저 이번 과제에 적용된 Mass Spring System에 대한 설명을 하겠다. external force fields인 Gravity과 internal force fileds인 Spring force를 부여하였고, 4개의 벽을 구성하여 이에 대한 Collision Handling을 수행하였다. 그리고 이러한 Elastic Spring을 기술하기 위해 Stiffness constant: k와 Initial spring length: l, Current spring length: L을 사용하여 Hooke's law에 따라 Spring force의 크기를 계산하였다. 이때 Spring force는 edge(=spring)에 대하여 양 끝점에서 서로 반대방향의 힘이 작용하므로 한 점에 대해 힘의 방향을 계산한 후 다른 한 점은 부호를 바꿔 적용해주면 된다. 또한, spring constant(=Stiffness constant)는 방향키로 조절이 가능한 Global spring constant를 주고 길이 에 반비례하게 각 edge에 대해 계산하여 부여하였다.

이러한 rendering을 update하기 위해서는 먼저 Total force(internal force, external force)를 계산하고 시간 t에 대한 position과 velocity를 계산해야하는데, 이는 상미분방정식을 사용하여 적분을 수행해 구해낸다. 해당 system의 force는 2차로 표현되기 때문에 두 개가 결합된 1차 상미분방정식으로 표현하여 계산한다. External force로 작용하는 gravity는 total force에 더해주는 방식으로 적용하고, internal force인 spring force는 edge의 current length와 edge의 end point에서의 힘의 방향을 계산하여 위에서 설명한대로 Hooke's law에 따라 값을 계산하여 두 end point에서 서로 방향이 반대가 되도록 한 뒤 더해줘서 부여한다. 여기서 spring damping도 이와 같은 방법으로 하나의 end point에서 상대적인 힘의 작용 방향과 current length를 계산하고 damping constant를 곱해서 빼주는 방법으로 부여한다.

여기서 적분을 수행하는 방법으로는 두 가지를 사용하였는데, 먼저 가장 간단한 Euler's method는 현재의 position과 velocity를 사용하여 time step 다음의 position, velocity를 계산해낸다. 그리고 Modified Euler's method는 먼저 현재의 velocity를 사용하여 다음 time step의 velocity를 구하고, 이를 반영하여 다음 time step의 position을 계산한다. Euler's method는 Sub steps의 개수에 의존적이므로 Sub steps의 개수가 적을 때는 만족스럽지 않은 모습 을 보인다. 하지만 Modified Euler's method 는 그에 비해 Sub steps의 개수에 덜 영향을 받아 Sub steps의 개수가 적 어도 상대적으로 만족스러운 모습을 보인다.

벽에 대한 collisionHandling은 직전 과제의 보고서에서 상세하게 설명했으므로 여기서는 넘어가겠다를 통해 curve segment를 그리는 방법에 대해 설명하겠다. Uniform Cubic B-spline curves는 controlPoints 중에 인접한 Control Point 4개를 순차적으로 지정하고, 해당 Control Point 4개와 0에서 1까지 균일하게 나눠진 t값을 사용하여 curve 위의 point 계산한다. 더 자세한 B-Spline curve 위의 point를 계산하는 방법은 t값을 Bell-shaped basis function에 대입하고 Control Point 4개와 곱하여 값을 얻어낸다. 이렇게 얻어낸 Curve Segment를 그릴 때는 B-Spline 위의 Point값을 GL_LINE_STRIP으로 그려낸다. 또한 Draw Control Polygon을 수행할 때는 B-Spline 위의 Point를 계산할 때 사용한 Control Point 4개를 GL_LINE_STIPPLE과 GL_LINE_STRIP을 사용하여 그린다.

벽에 대한 collisionHandling은 xy평면에 대해 벽이 이루어지므로 noraml vector는 이에 맞게 설정한 뒤 collisionHandling을 수행할 때 바 뀌는 window의 aspect에 따라 벽의 position을 설정해준다. 벽에 대

한 collision은 벽의 position에서 particle의 position으로 향하는 vector와 벽의 normal vector를 dot product하여 distance를 얻어 판별한다. 만약 이렇게 구한 distance가 particle의 radius와 임의로 준 epsilon을 합한 값보다 작다면 particle이 벽을 통과한 상태이므로 particle의 position을 벽의 normal 방향으로 radius와 epsilon을 더한 값에서 distance 를 뺀만큼 이동시켜준다. 그리고 벽의 normal vector와 particle의 속도 vector를 dot product한 값의 부호를 따져 벽 안으로 들어오 는 것에 대해서만 collision reponse를 수행한다. 여기서 collision reponse을 계산할 때는 velocity를 tangent 성분과 normal 성분으로 나눠 normal 성분에 restitution 계수를 곱해 계산을 진행한다. 마지막으로 매 frame 마다 해당 physical law을 반영하여 update를 수행할 때는 상미분방정식을 풀어야 한다. 이러한 상미분방정식을 풀 때는 1개의 frame 당 Sub step의 개수만큼을 계산해서 오차를 줄여 rendering한다.

이제 2D curve editing interface의 구현에 대한 설명을 하겠다.
첫 번째로 Create는 마우스 좌클릭을 눌렀을 때 Particle의 개수를 표현하는 변수 nParticle을 1 증가시키고, Particle의 좌표를 저장하는 C++ STL의 2차원 vector에 1차원 vector를 추가한다. 그리고 마우스 커서의 좌표를 얻어와서 nParticle번째 vector에 커서의 x, y, 0을 순서대로 추가한다. 여기서 Curve는 2D에 그려지므로 z좌표는 0을 대입한다. 그리고 마우스 커서의 좌표는 screen 좌표에서 얻어지므로 이를 world 좌표로 바꿔주는 과정을 수행한다. 이러한 마우스 커서의 좌표를 얻기 위해 좌표를 변환하는 과정은 4가지 기능 전부에서 이루어진다.

두 번째로 Attach는 마우스 좌클릭을 눌렀을 때 Particles를 전부 탐색하며 마우스 커서의 위치와 가장 가까운 Particle을 찾아내고 커서와 Particle의 거리가 epsilon 보다 작을 때 이를 select한다. 이때 select된 개수가 2개가 되면 2개의 Particle 번호를 각각 e1과 e2에 넣어준다. 그리고 2개의 Particle 사이의 거리를 구해서 l에 넣어주고 이렇게 계산된 l을 사용해서 해당 spring(edge)의 spring constant를 k에 저장한다. 마지막으로 Edge의 개수를 나타내는 nEdge를 1 증가시킨다. 여기서 주의할 점으로 같은 Particle을 두 번 연속 선택하여 Attach를 수행하게 되면 오류가 발생하므로 첫 선택은 그대로 수행하지만, 두 번째 선택은 첫 번째 선택과 다른 Particle을 선택했을 때만 기능을 수행하도록 한다.

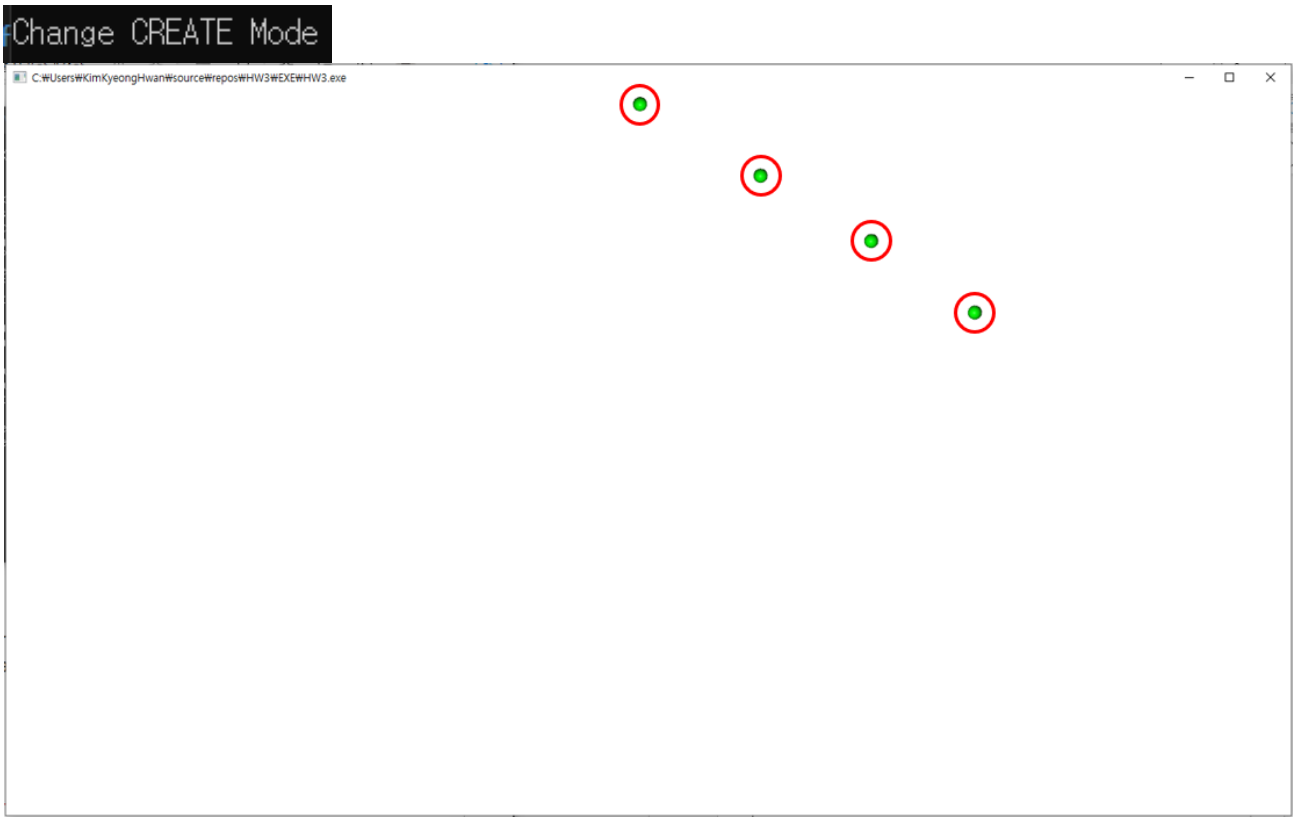
세 번째로 Drag는 먼저 D키를 눌러 Drag Mode로 변경했을 때 해당 Mode가 여러 번의 Drag 이후에도 계속 유지되도록 하기 위해 마우스 왼쪽 버튼을 놓았을 때 pickMode를 COMPLETE로 바꿔준다. 그리고 Drag를 위한 좌클릭을 눌렀을 때 만약 pickMode가 COMPLETE라면 DRAGGING으로 바꿔주고 DRAGGING에 알맞은 일을 수행해주게 된다. 이제 DRAG에 알맞은 일은 먼저 Attach와 유사하게 모든 Particles를 탐색하며 마우스 커서와 가장 가까운 Particle을 찾는다. 이때 커서와 Particle의 거리가 epsilon 보다 작을 경우 해당 커서의 위치에 create를 수행해주고 생성된 Particle과 가장 가까운 Particle 2개를 select한 걸로 해서 Attach를 수행한다. 마지막으로 생성된 Particle은 nail이 된 걸로 수행하고, 마우스를 놓았을 때는 create와 attach, nail에서 수행한 것들을 돌려놓는다.

참고로 두 번째와 세 번째에서 마우스 좌표와 Particle 사이의 최단 거리를 구하는 방법으로는 두 point를 시작과 끝으로 갖는 벡터의 norm을 구함으로 계산할 수 있었다.

네 번째로 nail은 Particle의 개수만큼 nail을 할지말지를 나타낼 bool 타입 배열을 선언한다. 그리고 N키를 눌러 Nail Mode로 변경시키고, 마우스 좌클릭을 눌렀을 때 Particles를 전부 탐색하며 마우스 커

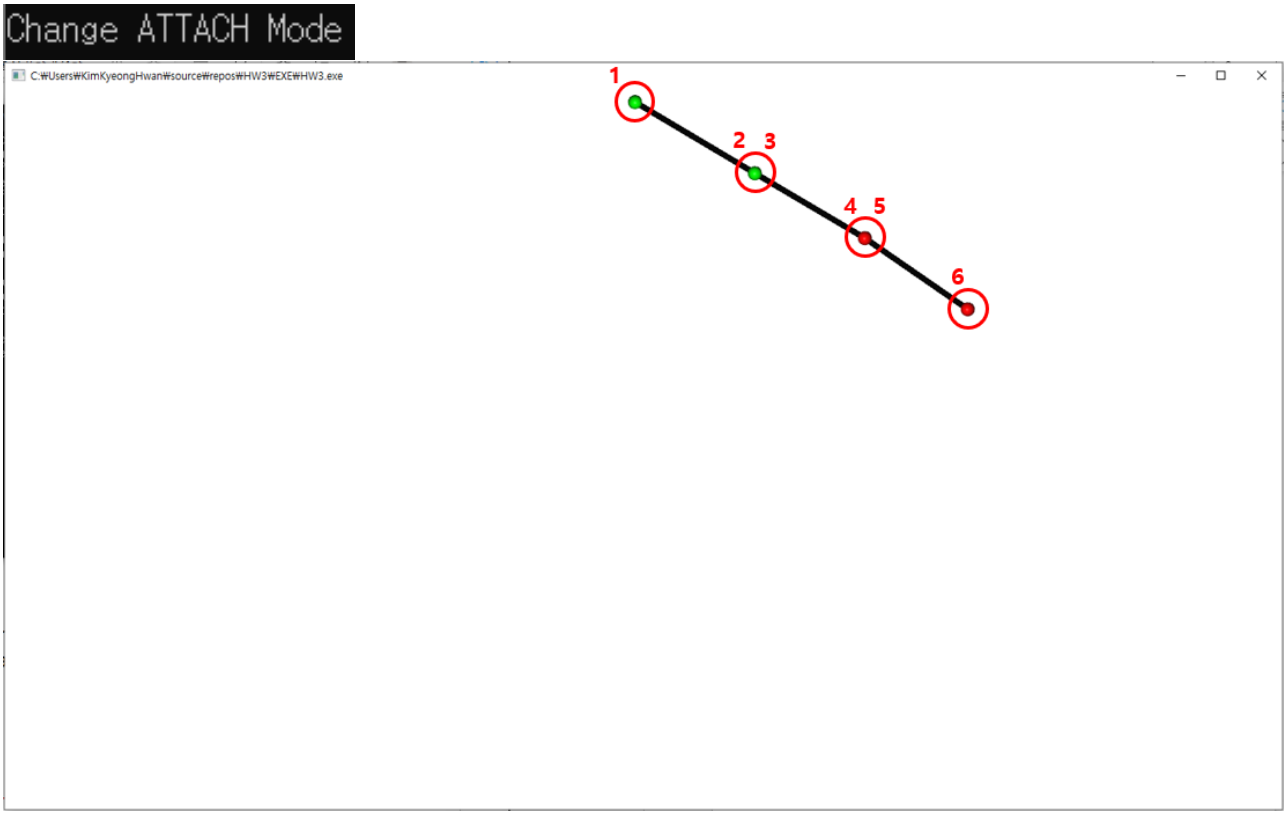
서의 위치와 가장 가까운 Particle을 찾아내고 커서와 Particle의 거리가 epsilon 보다 작을 때 해당 Particle의 번호에 해당하는 bool 타입 배열을 false로 바꿔준다. 이렇게 false로 바뀐 Particle은 position과 velocity를 계산하기 위해 상미분방정식을 풀 때 제외하고 푼다.

1. Select/Create Particles (C key):



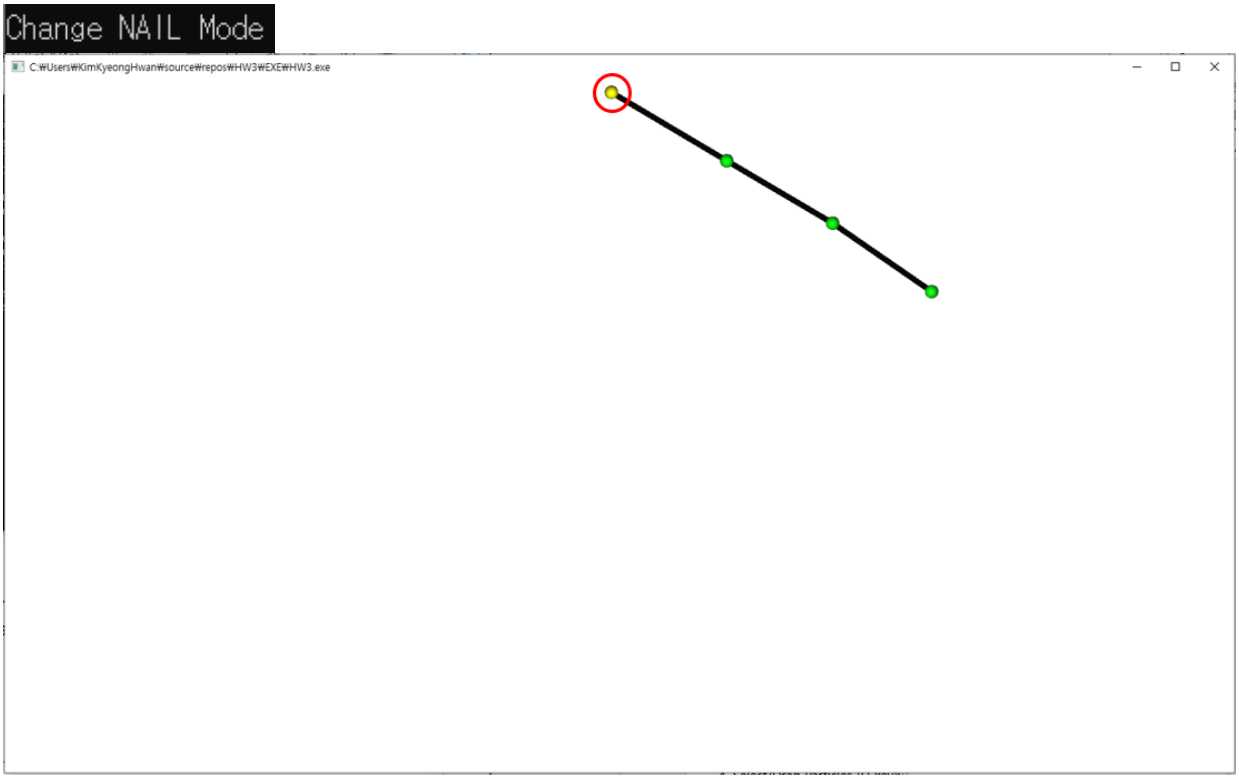
c키를 눌러 pickMode를 CREATE Mode로 바꾸고, Mouse Cusor의 Left Click으로 빨간색 원안에 위치한 4개의 Particle을 추가해 총 4개의 Particle이 그려진 모습이다.

2. Select/Attach Particles (A key):

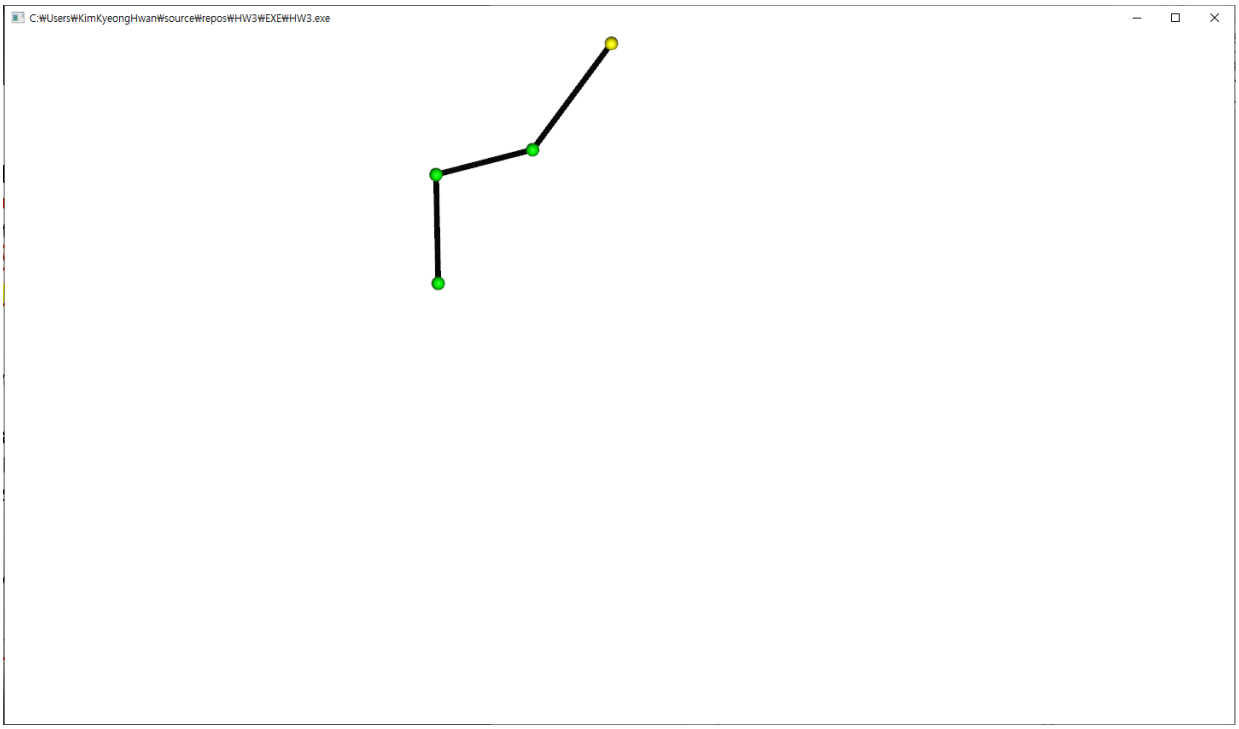


a키를 눌러 pickMode를 ATTACH Mode로 바꾸고, Mouse Cusor의 Left Click으로 빨간색 원에 해당하는 4개의 Particle을 빨간색 숫자에 순서에 따라 눌른 결과이다. 선택된 Particle은 빨간색으로 rendering된다. 연달아 누른 6개의 Particle 사이에 3개의 spring(edge)가 생성된 것을 볼 수 있다.

3. Select/Nail particles (N key):

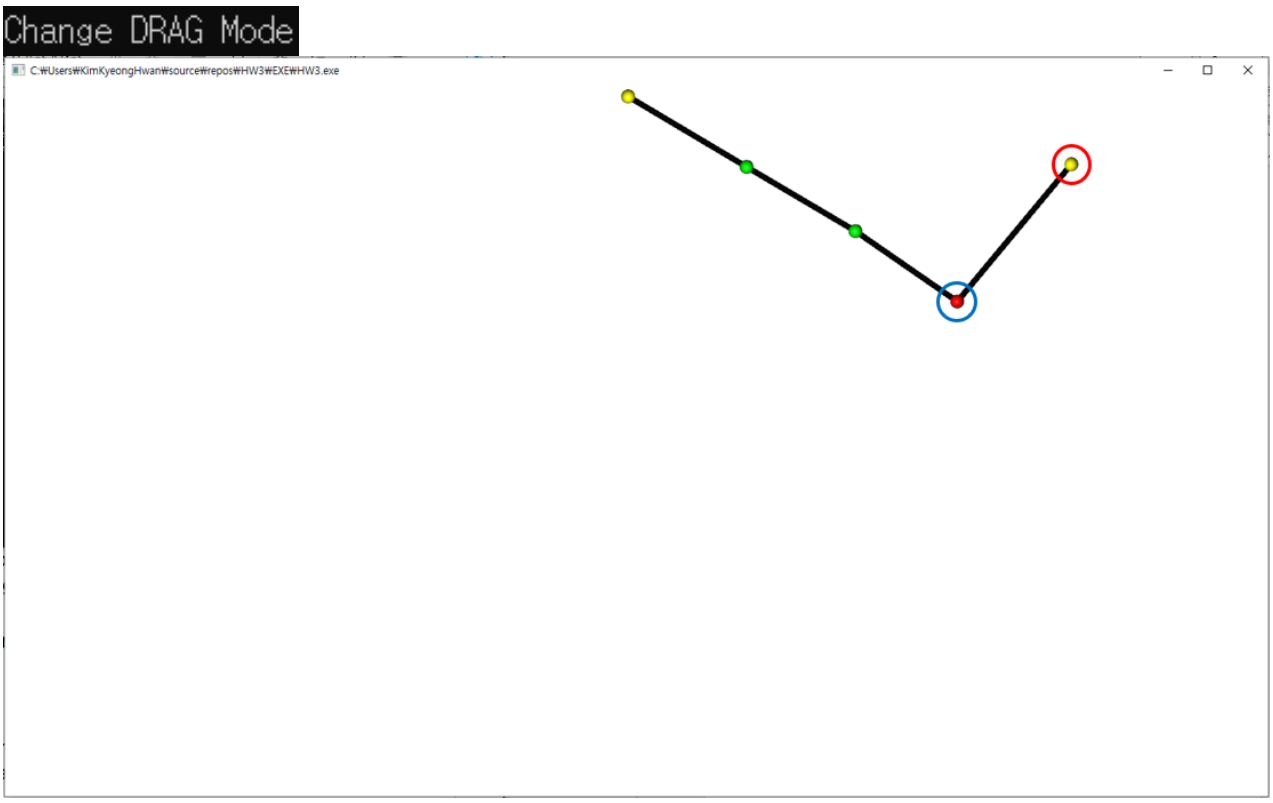


n키를 눌러 pickMode를 Nail Mode로 바꾸고, Mouse Cusor의 Left Click으로 빨간색 원안에 표시된 1개의 Particle을 선택해서 Nail state로 바꿔 노란색으로 rendering하게 했다.
이렇게 선택된 1개의 Particle은 position과 velocity를 계산하기 위한 상미분 방정식을 풀 때 제외된다.

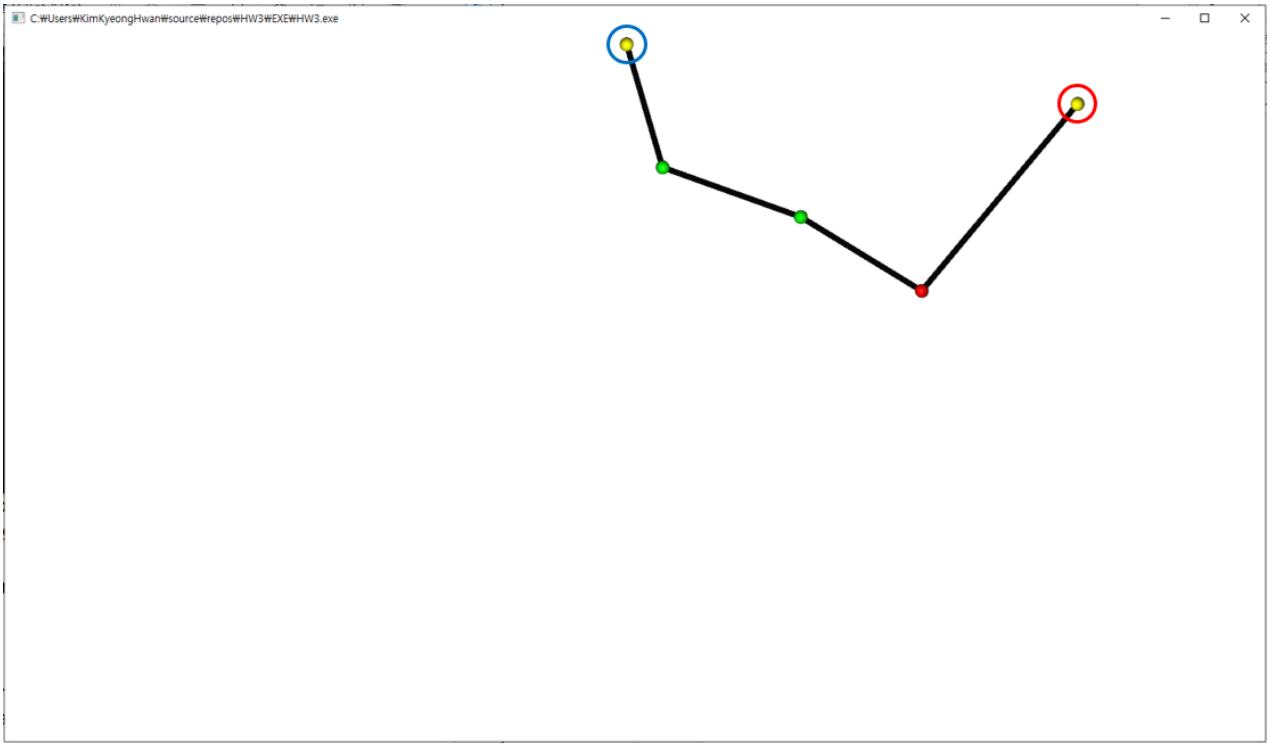


1개의 Particle을 Nail하고 시뮬레이션한 모습이다. 노란색의 Particle만 고정되고, 나머지 초록색의 3개 Particle은 spring에 의해 위아래로 튕기는 모습과 좌우로 진자운동하는 모습을 볼 수 있다.

4. Select/Drag Particles (D key):

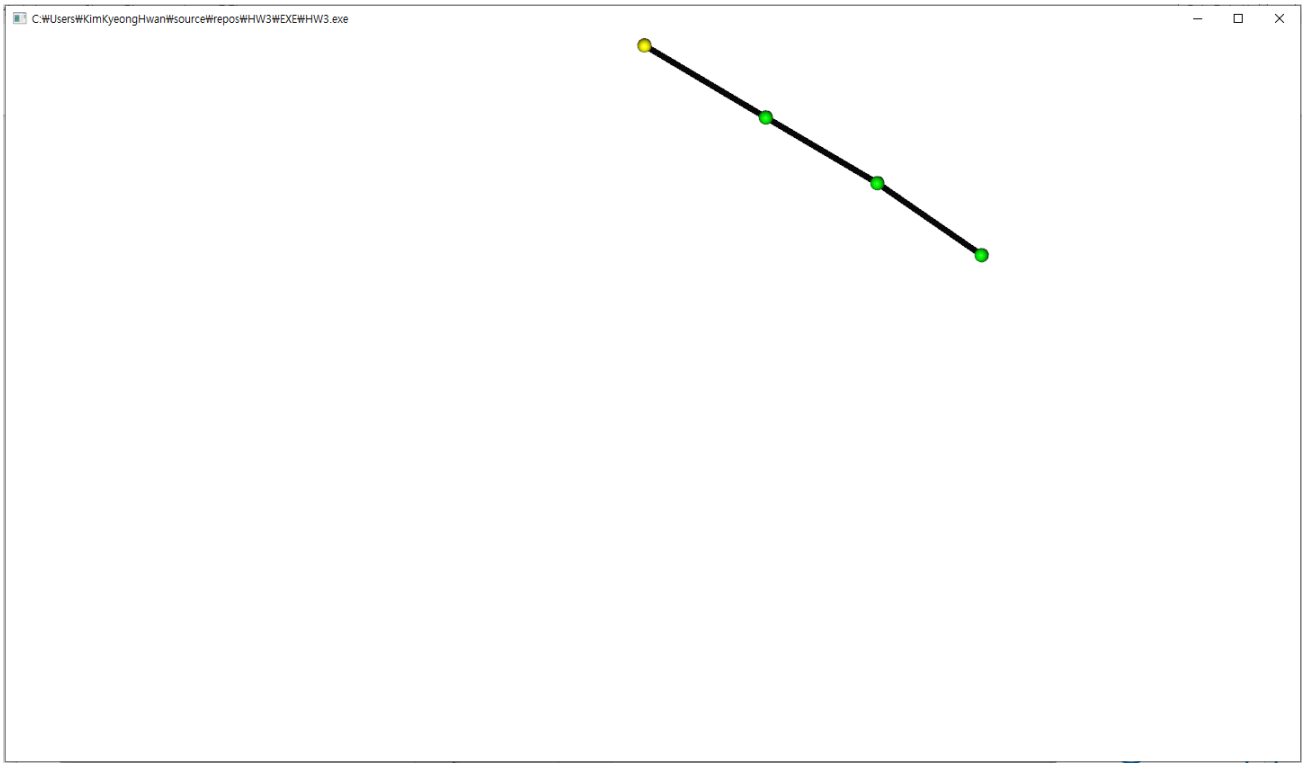


d키를 눌러 pickMode를 Drag Mode로 바꾸고, Mouse Cusor의 Left Click으로 파란색 원안에 표시된 1개의 Particle을 선택해서 노란색 원의 위치까지 Drag한 모습이다. Drag Mode일 때는 Particle이 선택되면 해당 위치에 새로운 Particle을 생성하고, 생성한 Particle을 마우스 커서의 움직임에 따라 이동시킨다. 그리고 이동된 Particle과 선택된 Particle 사이에는 spring(edge)이 생긴다.



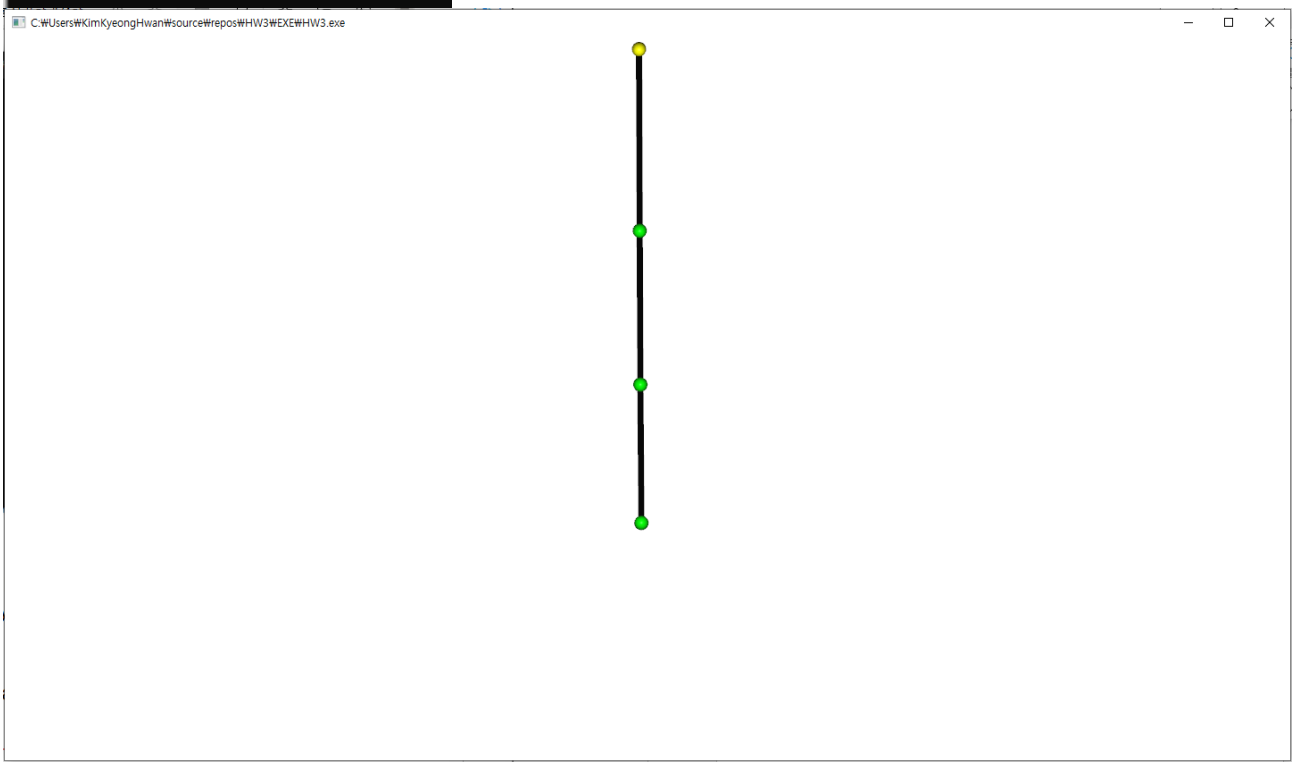
Nail Mode로 Nail 시킨 파란색 원의 Particle과 Drag Mode로 Drag해서 Nail 시킨 빨간색 원의 Particle에 대해서 시뮬레이션을 진행한 모습이다. 두 Particle은 상미분방정식을 풀 때 제외되므로 position과 velocity의 변화가 없고, 나머지 3개의 Particle은 연결된 spring에 대해 internal force인 spring force가 작용하는 모습을 볼 수 있다.

앞으로 5~9)에서 사용될 초기 particle의 state이다.



5. Turn on/off point damping (P key):

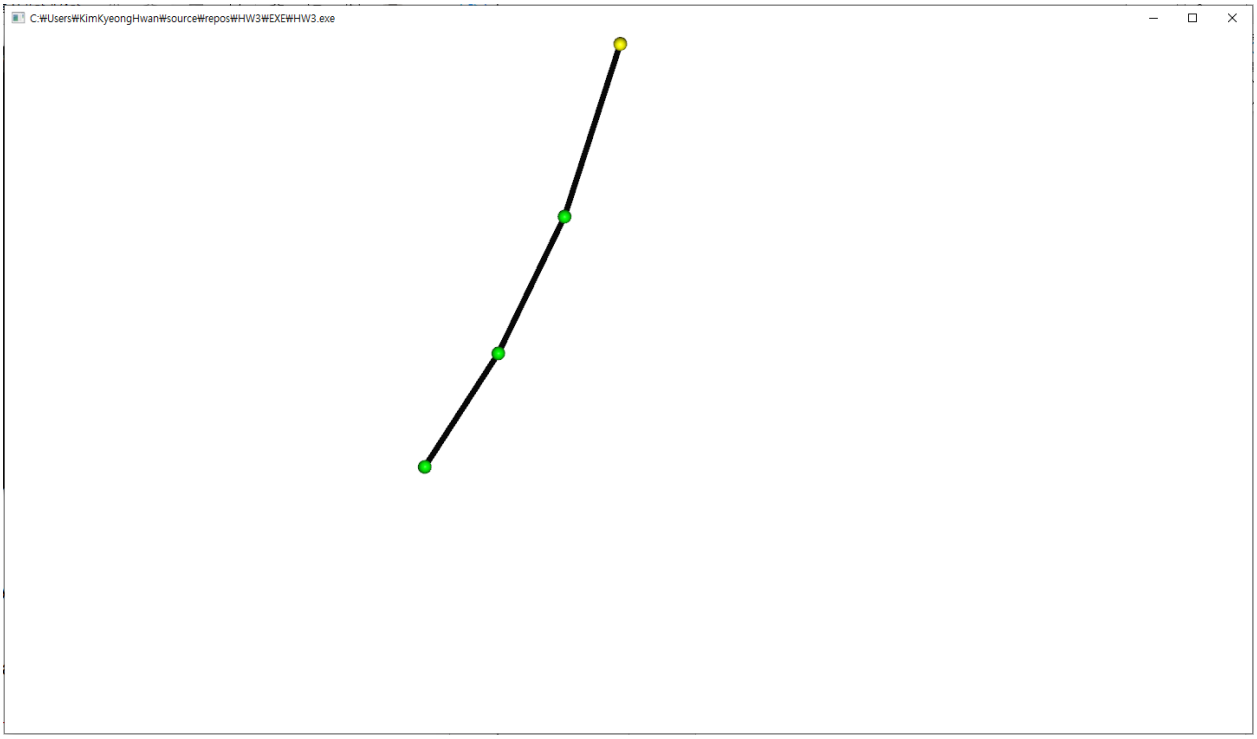
```
usePointDamping is True
```



이전까지 point damping을 off하고 시뮬레이션 했으므로 on하고 시뮬레이션한 결과만 첨부하겠다.
Spring constant로 1.0, Damping coefficient로 0.01을 사용하여 point damping을 on한 모습이다.
Point damping이 좌우로 흔들리는 진자운동과 spring에 의해 위아래로 튕기는 Particle에 저항을 걸어 더 이상 흔들리지 않고 쭉 펴진 걸 볼 수 있다.

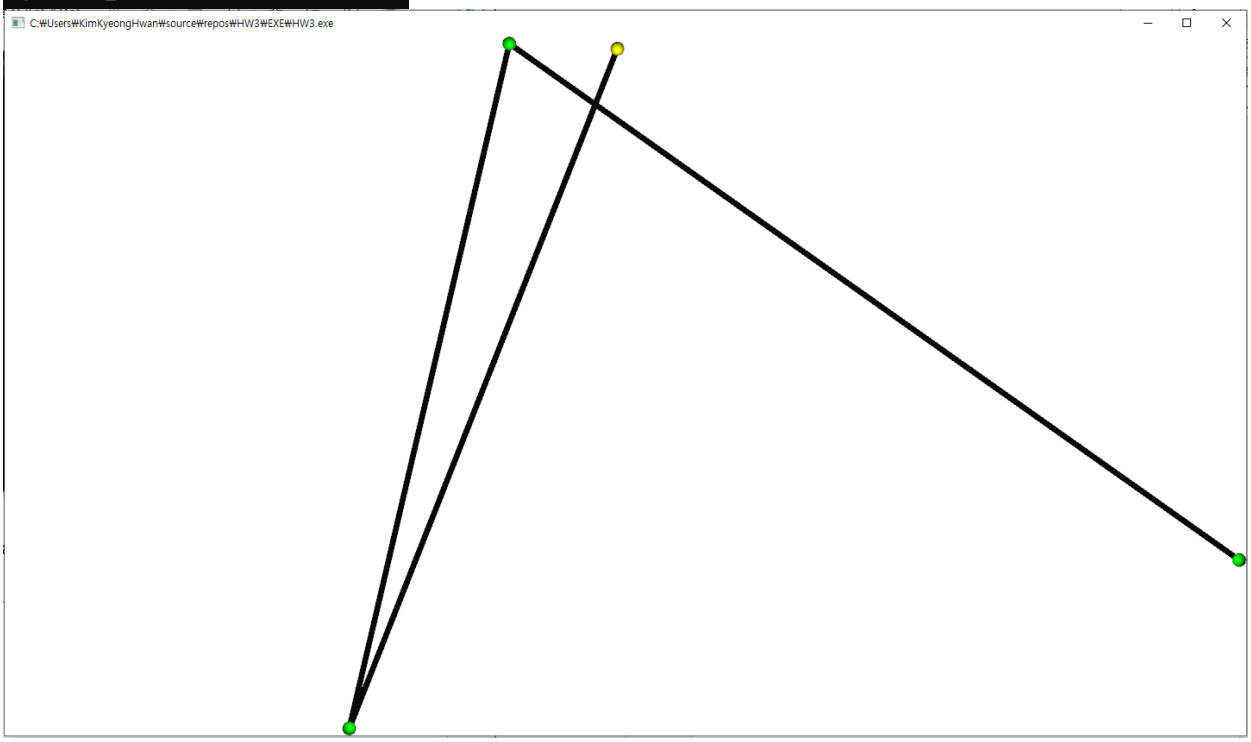
6. Turn on/off damped spring (S key):

```
useSpringDamping is True
```



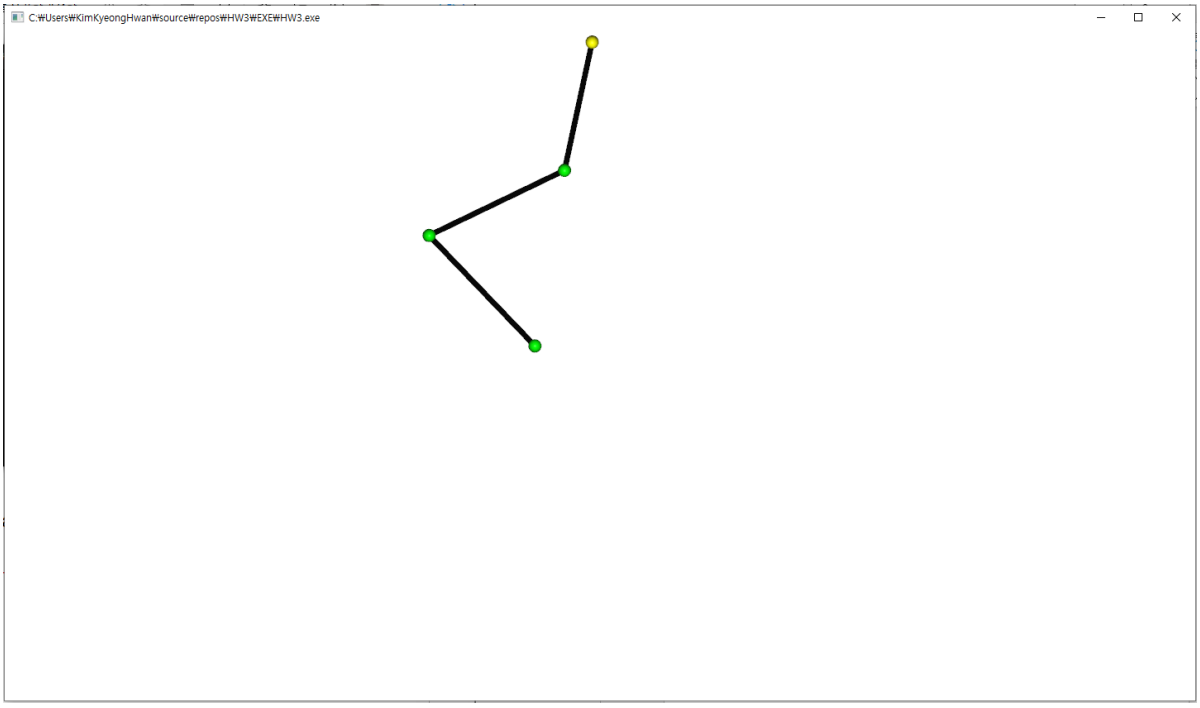
이전까지 spring damping을 off하고 시뮬레이션 했으므로 on하고 시뮬레이션한 결과만 첨부하겠다.
Spring constant로 1.0, Damping coefficient로 0.01을 사용하여 spring damping을 on한 모습이다.
spring damping이 spring에 의해 위아래로 튕기는 Particle에게만 저항을 걸어 위아래로 튕기지 않고 진자운동만을 하는 것을 볼 수 있다.

7. Make the system instable by increasing the spring constant (Up/Down key):



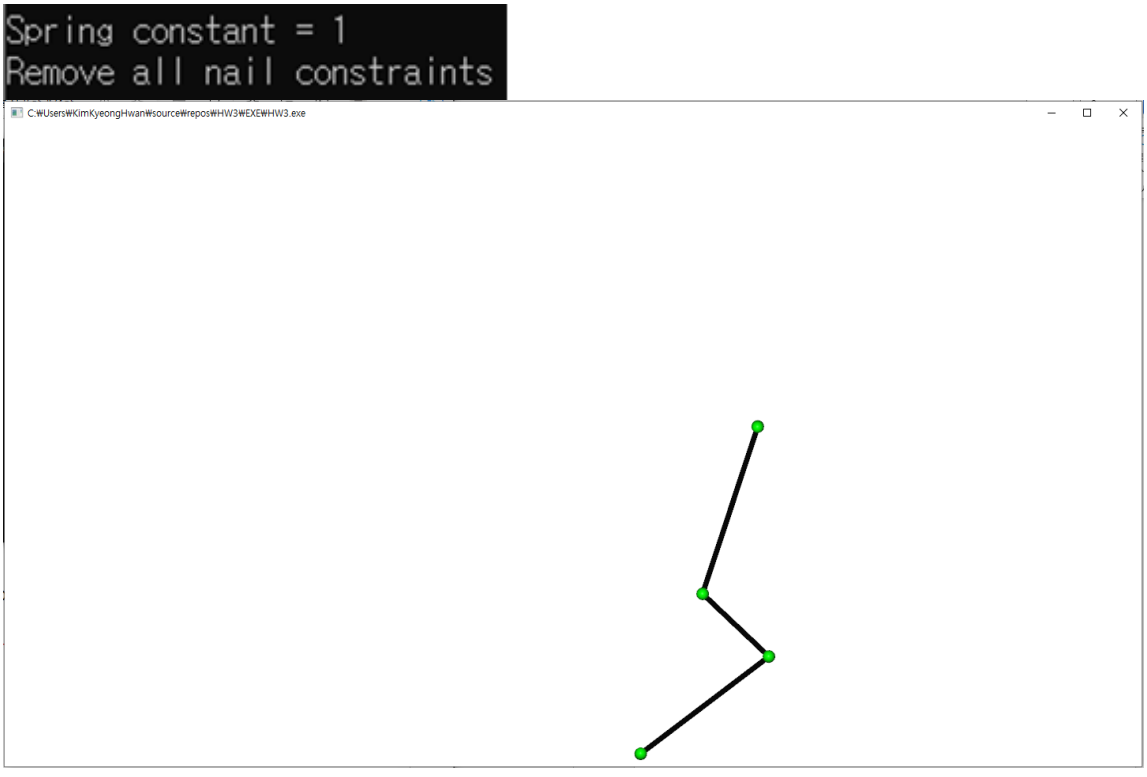
Spring constant를 최대인 10으로 키워 매우 불안정한 system을 만들었다.
Particle과 spring이 엉망으로 움직이고 늘어나 만족스럽지 못한 모습을 보인다.

8. Stabilize the system by decreasing the time step size or sub-time step size:



Sub step의 개수를 9로 키워 불안정적이었던 system을 안정적으로 바꿔놓았다.
이게 가능한 이유는 높은 spring constant로 인해 위아래로 심하게 튕기던 spring이 1 frame당 sub step의 개수인 9번 계산되어 반영되므로 다소 안정적으로 변하였다.
하지만 damping을 모두 꺼 spring의 길이는 유지하지만 아직까지 저항이 없어 심하게 흔들리는 것을 볼 수 있다. 흔들리는 것을 줄이려면 damping을 사용하여 저항을 부여해야한다.

9. Remove all nail constraints so that the particles bounce on the floor and walls:



Spring constant를 다시 1로 돌려놓고 모든 nail constraints를 제거한뒤 시뮬레이션한 결과이다.

External force인 gravity에 의해 바닥으로 떨어지고, wall에 의해 collision이 일어나며 internal force로 spring force가 작용하는 모습이다. Modified Euler method로 상미분방정식을 풀기 때문에 다소 빠르게 모든 particle이 바닥에 붙는 모습을 볼 수 있다.

Appendix:

원활한 시뮬레이션을 위해 위 requirements에서 설명하지 않은 기능들이 있다.

```
Keyboard Input: i for Initialize all Particle
Keyboard Input: r for Remove all nail constraints
Keyboard Input: b for Backup all Particle
Keyboard Input: u for Undo all Particle
```

먼저 i key의 initialize all Particle은 모든 Particle에 대한 state와 control을 초기화 시키는 기능이다.

r key는 모든 nail constraints를 해제하는 기능을 수행한다.

b key는 모든 Particle의 position과 velocity를 저장한다. 여기서 control에 대한 constrained나 edge에 대한 것들은 저장하지 않아도 된다. 이는 position에 따라 바뀌거나 영향이 없는 것들이기 때문이다.

u key는 b key를 사용해 backup해둔 모든 Particle의 position과 velocity를 되돌리는 기능이다. 여기서도 constrained나 edge에 대한 것은 수행하지 않고 position과 velocity만 되돌려놓으면 된다.