<Compute Graphics Project #3>

2019030991 홍정범

---

0. Parsing

기본적으로 모든 기능 구현에 앞서서 bvh파일에서 내가 원하는 정보를 읽어와야 vao와 vbo에 정보를 넘겨주어 랜더링을 할 수 있기 때문에 bvh파일의 정보를 parsing하는 함수를 구현했다. 항상 drop을 했을 때는 가장 마지막으로 drop된 파일만 랜더링을 해줘야 하므로 bvh_file의 정보를 담고있는 list들을 초기화 시킨다. Parsing 함수는 terminal에서 출력해주어야 하는 정보를 parsing과정에서 저장한다. 가장 중요한 파싱의 경우에는 hierarchy 부분을 level 1에서, motion 부분의 경우에는 level 2에서 입력을 받았다. Level 1은 stack 구조를 활용해서 nodes 배열에 [joint name, x, y, z, parent index in nodes list]를 저장하게 하였다. Parent가 무조건 nodes 배열 기준 전 칸에 들어오는 점을 이용했다. 최상단 노드의 경우 부모를 -1로 설정해 주었다. Nodes 배열의 저장되는 순서는 입력이 들어오는 그대로 저장이 되기 때문에 channels에 저장하는 것과 nodes에 저장되는 순서가 같아 나중에 channels에 일처리를 해줄 때 편리하게 했다.

---

1. treeing

nodes에 저장했던 정보들을 이용하여 node를 만들어 tree 구조를 구성해주었다. 그리고 만들어진 node들을 tree list에 담아주었다.

---

2. drop_callback

file 입력을 받아서 parsing과 treeing함수를 각각 실행시켜준다. 그 후 과제 명세에 맞게 출력 해준다.

---

3. Main

실행 영상 링크 : https://youtube.com/shorts/O3tvvyEu9Kc?feature=share

우선 프레임 별로 출력을 해주기 위하여 glfwgettme을 이용하여 누적시간을 연산하고 frame 시간만큼 지나면 translate을 해주게 하였다. Drop만 하게되면 기본적으론 line 형태로 렌더링하게 되며 그 후에는 1과 2 중 어떤 것을 누르냐에 따라서 라인과 박스 랜더링을 변환할 수 있게 된다. 이후 스페이스를 누르면 motion에 들어있던 값들을 transform 연산하여 움직임을 구현하기 시작한다.

<motion data 처리>

739번줄부터 motion data를 처리하는 부분이다. Channel과 nodes에 담겨있는 순서가 동일하기 때문에 channel idx와 nodes의 idx를 적절히 이용하여 x_r, y_r, z_r, x_p, y_p, z_p 값을 초기화해준다. 하지만 bvh파일마다 연산해야하는 순서가 다를 수 있으니 들어올 때마다 연산을 진행하여 bvh파일에 상관없이 연산이 될 수 있게 하였다. Site의 경우 channel 값이 없기 때문에 parisng 할 때 block이라는 값을 저장하여 넘어가게 해주었고, 최상단 node의 경우와 다른 node의 경우를 나누어 연산하게 하였다.

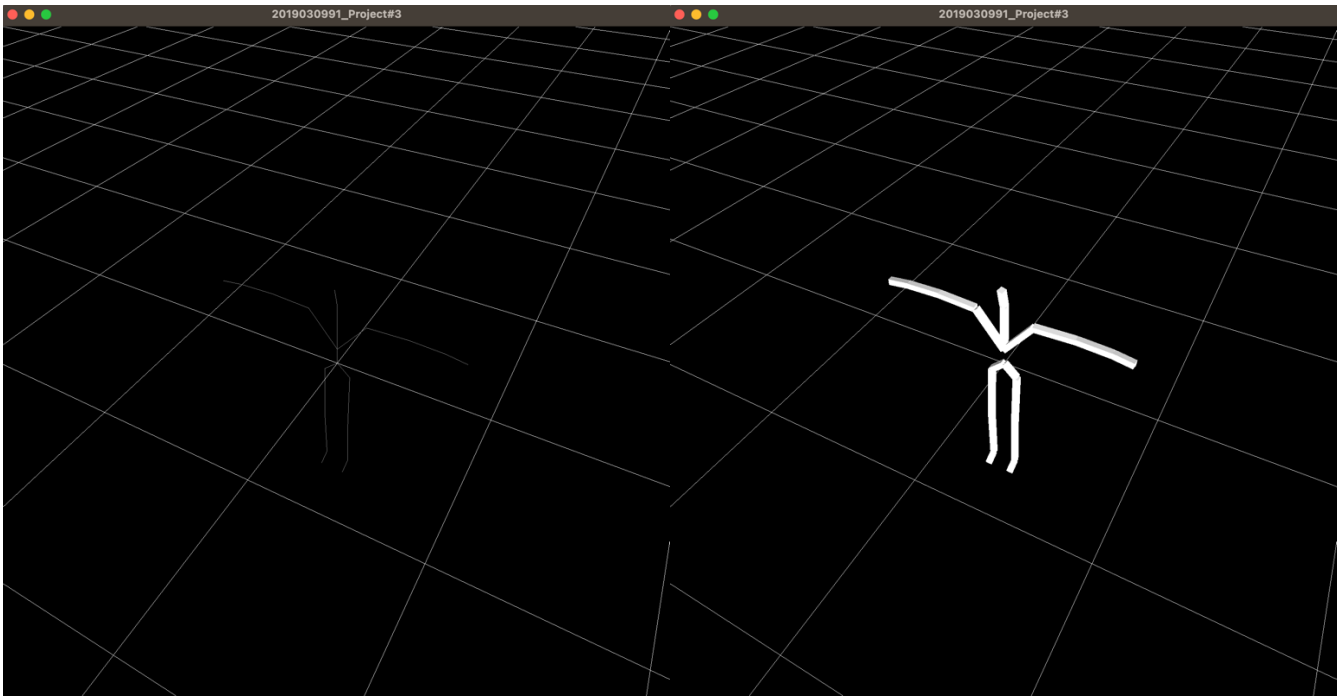
<line rendering>

Line으로 rendering 하는 것의 경우에는 vao를 만들 때 world 상에서 자신의 점과 부모의 점을 transform 값과 offset 값을 기준으로 연산하여 vao를 만들 때 직접 넣어주었다.

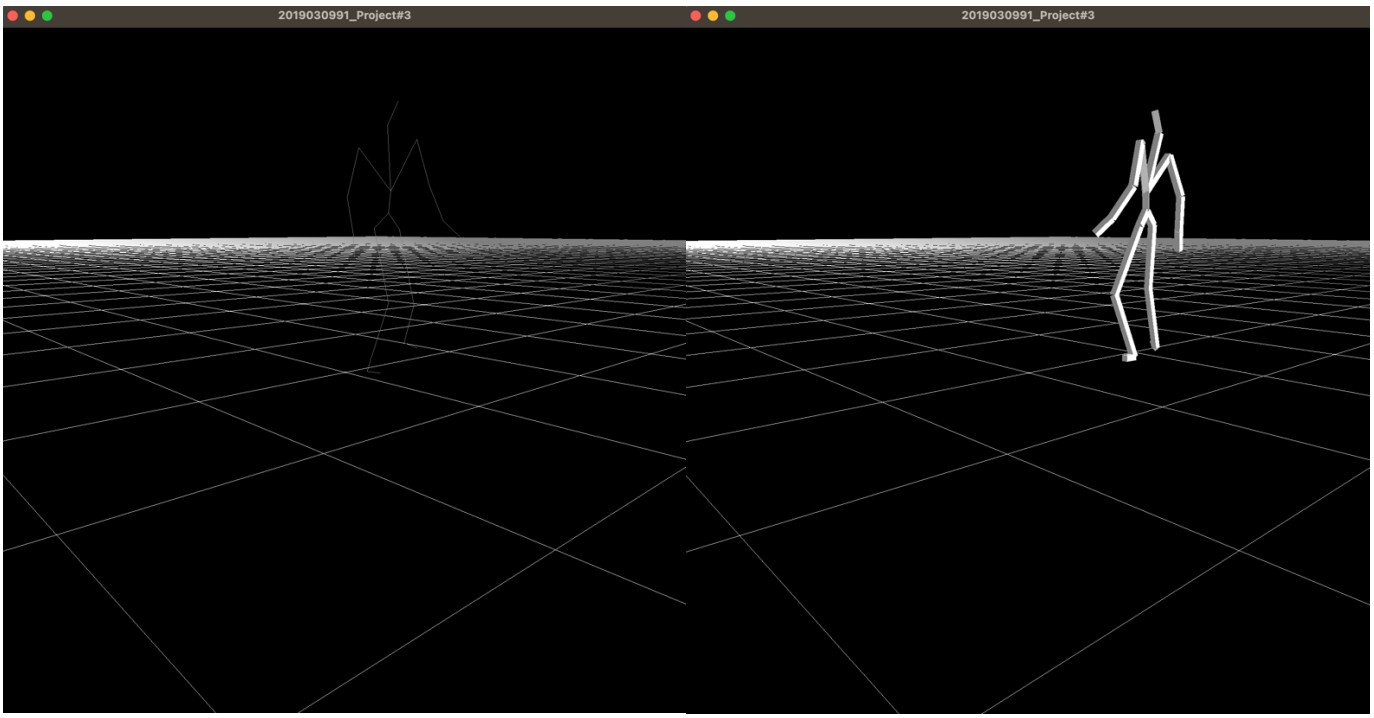
<box rendering>

Box로 rendering 하는 것의 경우에는 vao는 우선 그냥 우리가 일반적으로 그리던 box로 구현을 해주었다. 이를 가져와서 그림을 그릴 때 몇가지 과정을 거쳐서 transform을 해주었다. 첫째로 line rendering과 같이 자신의 점과 부모의 점을 연산해서 박스를 얼만큼 늘일 것인지 결정해주었다. 그리고 내적과 외적을 통해 y축과 연산하여 box들이 얼만큼 공간상에서 회전해야 하는지 연산해주었다. 그리고 박스를 우선 자신의 점과 parent의 점 사이로 옮겨오고, 길이의 절반만큼 scale을 해주면 양 옆으로 늘어나 내가 원하는 길이 만큼 늘일 수 있는 그렇게 적용해주고 회전까지 적용을 해주면 이제 원하는 그림을 그릴 수 있다.

<실행 사진>



Drop 된 직후의 모습



Space bar를 입력했을 때의 모습

Phong shading의 경우 projecet #2와 같이 구현했다.