**Class Assignment 3**

**2016025687 컴퓨터소프트웨어 학부 이경훈**

**Skeleton code )**

한 변의 길이가 1인 사각형을 가로 60개 세로 60개로 xz 평면을 구현하였고, ClassAssignemnt 1에 있는 카메라 회전을 그대로 사용하였다.

**Drop callback)**

파일을 끌어다 놓으면 dropcallback이 실행되어, 해당 파일을 파싱하는데, 한 개의 joint마다 구조체처럼 사용하는 part라는 list에 정보가 들어가게 된다. 이때, [[[‘ROOT’][offset1,offset2,offset3]][‘XPOSITION’,,,,,,’ZROTATION’]],[[‘JOINT’][offset1,offset2,offset3]] [‘XPOSITION’,,,,,,’ZROTATION’]]………] 와 같이 JOINT별로 element를 묶은 형태로 파싱이 된다. 또한 괄호 { } 에 대한 list, MOTION에 대한 list를 만든다. 해당정보들은 나중에 rendering을 할 때 사용된다.

파싱을 할 때, 파일이름만 추출을 하는데, 추가과제때 적용되는 샘플 bvh 파일이름을 인식하기 위해 **import os**를하여, **os.path.basename**이라는 함수를 사용한다. 리눅스와 윈도우 둘다 파일 경로가 형식이 다르기 때문에 해당 함수를 사용했다.

**파일정보 출력)**

해당 함수가 실행되면 file name, 프레임 수, FPS, Joint 개, Joint 이름을 출력하며, Joint의 개수와 이름에는 End joint들은 포함시키지 않고, ROOT의 경우는 포함을 시켜 출력한다.

**T pos)**

Bvh 파일을 끌어다 놓으면 KeyToggle이라는 변수가 기본적으로 0으로 설정되기 때문에, 바로 재생되지 않고, render()에서 darw\_skeleton()이라는 함수가 실행된다. 해당함수는 drop call back에서 파싱을 할 때, 괄호 정보를 저장해놨는데 괄호 정보로 루프를 돌며 line을 그린다. ‘{‘가 나오면 push matrix를 하고 원점에서 part list에 저장된 offset만큼 떨어진 line을 그리게 되고 이후 offset만큼 glTranslatef를 해준다. ‘}’가 나오면 pop matrix를 해주어 stack처럼 동작하게 구현했다.

**Animation)**

render에서 keyToggle의 값이 1이면 draw\_animation() 함수를 동작하며, main문의 while루프가 돌 때 마다 frame을 가르키는 index가 1이 증가하여, frame\_index에 맞게 motion 값들을 들고 와서, part에 저장되어 있는 CHANNEL 순서에 맞게, glTranslatefv, glRotatefv를 해준다. frame index가 파일의 frame 수를 넘어가면 다시 0으로 만들어 처음부터 재생이 된다.

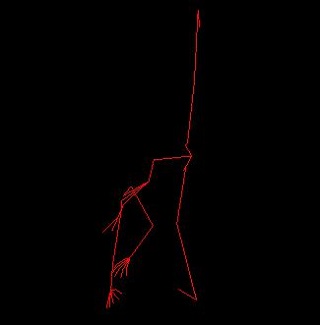
**추가과제)**

Cube로 rendering 되어 동작을 하며, 흰색 빛으로 동작하는 light 광원 1개와, cube에 대한 flat shading으로 구현하였다. 해당 기능은 **sample-walk.bvh** 파일에 대해 동작하는데, render에서 gCube의 값을 확인하여 1이면, light 광원을 넣어주고, draw\_skeleton2, draw\_animation2중에 하나를 선택해서 cube를 그려준다. 길이가 1인 큐브에 대해, T pos기준으로 joint가 어디 축 방향으로 떨어져 있는 확인후 offset의 length만큼 scaling을 해주었다. 해당 방향이 아닌 다른 축의 scaling의 값은 0.05를 주었고, 다소 offset 값들이 작으면, 발의 모양처럼, 뚜렷하게 안보일 수는 있으나, joint에 맞게 잘 동작한다.

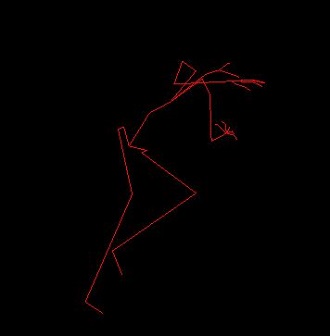
**동작 사진)**



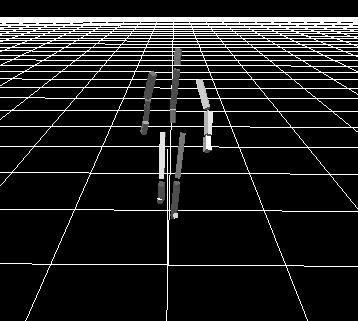
댄스 2



댄스 1



사다리 내려오기



추가 과제