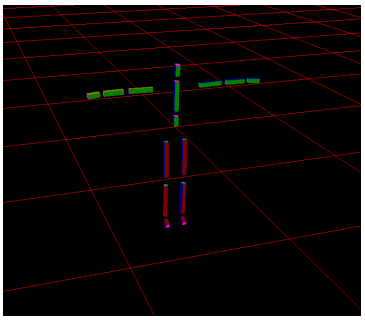
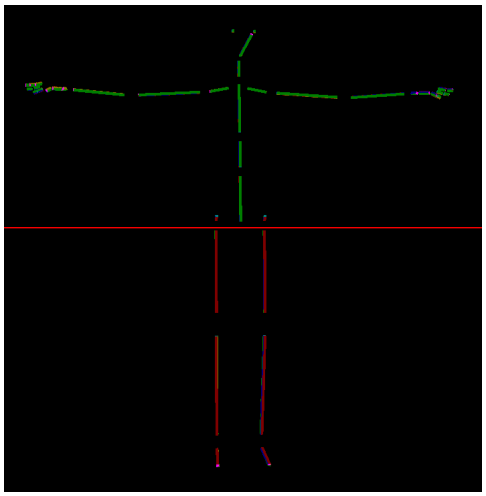
Computer Graphics Project 3: Bvh viewer

2023008413 장원

2025.06.08.

1. 드래그 앤 드롭을 이용한 bvh 파일 불러오기





```
bvh loaded
File name : jump.bvh
Number of frames : 811
FPS : 120.00004800000192
Number of joints : 43
List of all joint names :
hip
  abdomen
    chest
      neck
        head
          leftEye
          rightEye
      rCollar
        rShldr
          rForeArm
            rHand
              rThumb1
                rThumb2
              rIndex1
                rIndex2
              rMid1
                rMid2
              rRing1
                rRing2
              rPinky1
                rPinky2
      lCollar
        lShldr
          lForeArm
            lHand
              lThumb1
                lThumb2
              lIndex1
                lIndex2
              lMid1
                lMid2
              lRing1
                lRing2
              lPinky1
                lPinky2
  rButtock
    rThigh
      rShin
        rFoot
  lButtock
    lThigh
      lShin
        lFoot
```

```python
def parse_bvh(path):
    global g_motion_data, g_frame_time,g_num_frame
    with open(path, 'r') as f:
        lines = iter(f.readlines())

    stack = []
    rootnode = None
    channel_index = 0
    jointnum = 0
    joints = []

    for line in lines:
        words = line.strip().split()
        if not words:
            continue
        if words[0] in ('ROOT', 'JOINT'):
            jointnum += 1
            joint_name = words[1]
            joint = Joint(joint_name)
            if stack:
                stack[-1].add_child(joint)
            stack.append(joint)
            if rootnode is None:
                rootnode = joint
            joints.append(joint)
        elif words[0] == 'End':
            joint = Joint('End Site')
            stack[-1].add_child(joint)
            stack.append(joint)
        elif words[0] == '{':
            continue
        elif words[0] == '}':
            stack.pop()
        elif words[0] == 'OFFSET':
            stack[-1].offset = [float(w) for w in words[1:]]
        elif words[0] == 'CHANNELS':
            channels = words[2:]
            stack[-1].channels = channels
            stack[-1].channel_indices = list(range(channel_index,channel_index+len(channels)))
            channel_index += len(channels)
        elif words[0] == 'Frames:':
            g_num_frame = int(words[1])
        elif words[0] == 'Frame' and words[1] == 'Time:':
            g_frame_time = float(words[2])
            break

    g_motion_data = [list(map(float, next(lines).strip().split())) for _ in range(g_num_frame)]
    setup_all_bone_transforms(rootnode)
    return rootnode,jointnum
```

```python
def bvhdrop_callback(window, paths):
    global  g_bvhroot, g_motion_data, g_current_frame, g_motion_active, g_ispreloaded
    bvhfile = paths[0]
    g_bvhroot,rootnum = parse_bvh(bvhfile)

    print("File name : " + bvhfile)
    print("Number of frames : " + str(g_num_frame))
    print("FPS : "+str(1/g_frame_time))
    print("Number of joints : " + str(rootnum))
    print("List of all joint names : ")
    g_bvhroot.printall()
    g_current_frame = 0
    g_motion_active = False
    g_ispreloaded = False
    print("bvh loaded")
```

```python
class Joint:
    def __init__(self, name):
        self.name = name
        self.offset = np.array((0,0,0), dtype=float)
        self.channels = []
        self.children = []
        self.parent = None
        self.channel_indices = []
        self.bone_local_transform = None
        self.obj_local_transform = None
        self.obj = None

    def add_child(self, joint):
        joint.parent = self
        self.children.append(joint)

    def printall(self, level=0):
        print("  " * level + self.name)
        for child in self.children:
            if child.name != "End Site":
                child.printall(level + 1)
```

```python
def set_bone_local_transform(joint):
    if joint.parent is not None:
        start = np.array([0,0,0], dtype=np.float32)
        end = joint.offset
        direction = end - start
        length = np.linalg.norm(direction)
        if length < 1e-6:
            joint.bone_local_transform = glm.mat4(1.0)
            return
        center = (start + end) / 2

        y = np.array([0, 1, 0], dtype=np.float32)
        direction_n = direction / length
        axis = np.cross(y, direction_n)
        angle = np.arccos(np.clip(np.dot(y, direction_n), -1, 1))
        R = glm.mat4(1.0)
        if np.linalg.norm(axis) > 1e-6 and angle > 1e-6:
            R = glm.rotate(glm.mat4(1.0), angle, glm.vec3(*axis))
        S = glm.scale(glm.mat4(1.0), glm.vec3(g_bonesize, length, g_bonesize))
        T = glm.translate(glm.mat4(1.0), glm.vec3(*center))
        joint.bone_local_transform = T*R*S
    else:
        joint.bone_local_transform = glm.mat4(1.0)
```

먼저 bvhdrop_callback을 사용해 입력받은 bvh를 parse_bvh로 파싱해준다. stack을 사용해 부모와 자식관계를 등록해주고, 마지막에 FrameTime까지 기록한 후엔 프레임별 변환 정보를 g_motion_data에 저장한다. class joint들을 parse_bvh에서 생성하고, 마지막에는 각 joint들의 정보를 이용해 처음 T pose에서의 bone의 길이, 방향을 미리 저장해둔다.

동작을 재생하는 도중에 계속 bone의 길이, 방향을 새로 계산하니 짐벌락때문에 bone이 원치 않게 회전하게 되었고, 제공된 예시 및 사용하고자 하는 bvh에서 frame별로 root를 제외하면 position 변환 정보가 없었기에 미리 계산한 값을 사용하였다.

마지막으로 출력해야하는 bvh의 정보를 모두 출력해준다.

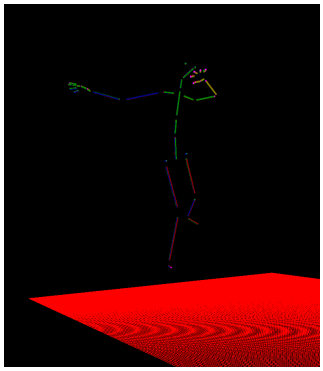
bone의 박스의 두께는 렌더링 전에 x, c로 각각 늘이고 줄일 수 있다.
새로운 파일을 드래그 앤 드롭하면 해당 파일로 교체되어 새롭게 랜더링을 시작한다.

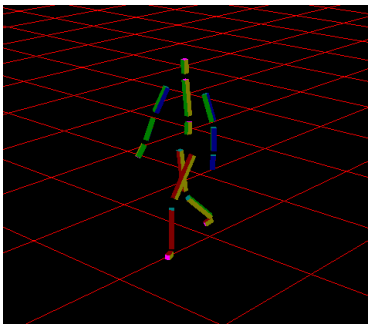
```python
def draw_bone_cube(parent_transform, joint, loc_MVP, cube_vao, MVP):
    M = parent_transform * joint.bone_local_transform
    M = MVP * M
    glUniformMatrix4fv(loc_MVP, 1, GL_FALSE, glm.value_ptr(M))
    glBindVertexArray(cube_vao)
    glDrawArrays(GL_TRIANGLES, 0, 36)

def draw_bvh_cubes(joint, parent_transform, frame_data, loc_MVP, cube_vao, MVP):
    T = glm.translate(glm.mat4(1), glm.vec3(*joint.offset))
    if frame_data is not None and joint.channels:
        for i, j in enumerate(joint.channels):
            value = frame_data[joint.channel_indices[i]]
            if j == "XPOSITION" or j == "Xposition":
                T = glm.translate(T, glm.vec3(value, 0, 0))
            elif j == "YPOSITION" or j == "Yposition":
                T = glm.translate(T, glm.vec3(0, value, 0))
            elif j == "ZPOSITION" or j == "Zposition":
                T = glm.translate(T, glm.vec3(0, 0, value))
            elif j == "XROTATION" or j == "Xrotation":
                T = glm.rotate(T, glm.radians(value), glm.vec3(1, 0, 0))
            elif j == "YROTATION" or j == "Yrotation":
                T = glm.rotate(T, glm.radians(value), glm.vec3(0, 1, 0))
            elif j == "ZROTATION" or j == "Zrotation":
                T = glm.rotate(T, glm.radians(value), glm.vec3(0, 0, 1))
    world_transform = parent_transform * T
    if joint.parent is not None:
        if joint.parent.children[0] == joint:
            draw_bone_cube(parent_transform, joint, loc_MVP, cube_vao, MVP)

    for child in joint.children:
        draw_bvh_cubes(child, world_transform, frame_data, loc_MVP, cube_vao, MVP)
```

```python
    elif g_bvhroot:
        glUniformMatrix4fv(loc_MVP_cubes, 1, GL_FALSE, glm.value_ptr(MVP))
        glUniformMatrix4fv(loc_M_cubes, 1, GL_FALSE, glm.value_ptr(M))
        glUniform3f(loc_view_pos_cubes, corigin.x,corigin.y,corigin.z)
        glUniform3f(loc_lightmov_cubes,lightpos.x,lightpos.y,lightpos.z)
        frame_data = g_motion_data[g_current_frame] if g_motion_active else None
        draw_bvh_cubes(g_bvhroot, glm.mat4(1.0), frame_data, loc_MVP,cube_vao, MVP)
```





글로벌 변수 g_bvhroot에 입력받은 bvh의 root에 해당하는 class joint가 들어있다. main함수에서 draw_bvh_cubes 함수를 호출한다.

joint에 저장된 channels값과 인덱스를 이용해 frame_data값를 탐색해 원하는 변환을 적용한다.

그릴 수 있는 joint라면 draw_bone_cube를 통해 랜더링하고, draw_bvh_cubes로 자기 자식에게 변환된 행렬을 넘겨 다음 위치에서 변환과 랜더링을 시작할 수 있도록 한다.

```python
file = os.path.join('head.obj')
appendmodel(file,'head',g_bvhroot)
file = os.path.join('neck.obj')
appendmodel(file,'neck',g_bvhroot)
file = os.path.join('lCollar.obj')
appendmodel(file,'lCollar',g_bvhroot)
file = os.path.join('rCollar.obj')
appendmodel(file,'rCollar',g_bvhroot)
file = os.path.join('hip.obj')
appendmodel(file,'hip',g_bvhroot)
file = os.path.join('abdomen.obj')
appendmodel(file,'abdomen',g_bvhroot)
file = os.path.join('chest.obj')
appendmodel(file,'chest',g_bvhroot)
file = os.path.join('rFoot.obj')
appendmodel(file,'rFoot',g_bvhroot)
file = os.path.join('lFoot.obj')
appendmodel(file,'lFoot',g_bvhroot)
file = os.path.join('rShin.obj')
appendmodel(file,'rShin',g_bvhroot)
file = os.path.join('lShin.obj')
appendmodel(file,'lShin',g_bvhroot)
file = os.path.join('lThigh.obj')
appendmodel(file,'lThigh',g_bvhroot)
file = os.path.join('rThigh.obj')
appendmodel(file,'rThigh',g_bvhroot)
file = os.path.join('lButtock.obj')
appendmodel(file,'lButtock',g_bvhroot)
file = os.path.join('rButtock.obj')
appendmodel(file,'rButtock',g_bvhroot)
file = os.path.join('lShldr.obj')
appendmodel(file,'lShldr',g_bvhroot)
file = os.path.join('lForeArm.obj')
appendmodel(file,'lForeArm',g_bvhroot)
file = os.path.join('rShldr.obj')
appendmodel(file,'rShldr',g_bvhroot)
file = os.path.join('rForeArm.obj')
appendmodel(file,'rForeArm',g_bvhroot)
file = os.path.join('rHand.obj')
appendmodel(file,'rHand',g_bvhroot)
file = os.path.join('lHand.obj')
appendmodel(file,'lHand',g_bvhroot)
setup_all_obj_transforms(g_bvhroot)
```

```python
def setup_all_obj_transforms(joint):
    set_obj_local_transform(joint)
    for child in joint.children:
        setup_all_obj_transforms(child)
```

```python
def set_obj_local_transform(joint):
    if joint.parent is not None:
        start = np.array([0,0,0], dtype=np.float32)
        end = joint.offset
        direction = end - start
        length = np.linalg.norm(direction)
        if length < 1e-6:
            joint.obj_local_transform = glm.mat4(1.0)
            return
        center = (start + end) / 2

        y = np.array([0, 1, 0], dtype=np.float32)
        direction_n = direction / length
        axis = np.cross(y, direction_n)
        angle = np.arccos(np.clip(np.dot(y, direction_n), -1, 1))
        R = glm.mat4(1.0)
        if np.linalg.norm(axis) > 1e-6 and angle > 1e-6:
            R = glm.rotate(glm.mat4(1.0), angle, glm.vec3(*axis))
        T = glm.translate(glm.mat4(1.0), glm.vec3(*center))
        joint.obj_local_transform=T*R
    else:
        joint.obj_local_transform=glm.mat4(1.0)
```

키보드 1을 누르면 미리 저장된 bvh와 obj를 로드하도록 하였다. 토글식으로 다시 누르면 해제된다.

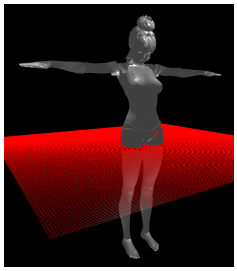
setup_all_obj_transforms로 이전의 bone 큐브와 마찬가지로 초기 상태의 위치와 방향을 미리 joint 안에 저장해둔다.

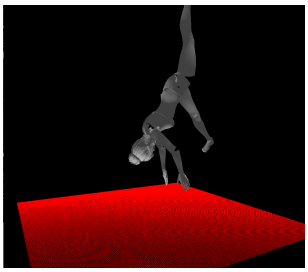
appendmodel로 이전 과제에서 사용한 obj 로더를 이용해 obj를 파싱해와서 해당 joint 안에 저장해둔다.

obj 파일은 하나씩 각 joint 안에 저장되어 있고, bone과 유사하게 draw_bvh_obj에서 draw()를 통해 변환을 마친 후 렌더링된다.

```python
def give_obj(joint,name,obj):
    if joint.name == name:
        joint.obj = obj
        return 0
    for child in joint.children:
        give_obj(child,name,obj)

def appendmodel(file,name,rootnode):
    vertices, normals, indices, normindices = load_obj(file)
    vn = []
    for i in range(len(indices)):
        vn.extend(vertices[3*indices[i]:3*indices[i]+3])
        vn.extend(normals[3*normindices[i]:3*normindices[i]+3])
    vertices = np.array(vn,dtype=np.float32)
    indices = np.array(indices,dtype=np.uint32)
    model = Model(vertices, indices,name)
    give_obj(rootnode,name,model)
```





Phong Illumination과 Phong shading을 사용하였다.

```python
now = glfwGetTime()
elap_time += now-begin_time
begin_time = now
if g_motion_active:
    if elap_time >= g_frame_time:
        g_current_frame = g_current_frame + 1
        elap_time = 0
        if g_num_frame == g_current_frame:
            g_current_frame = 0

glUniform3f(loc_materialcolor,0.5,0.5,0.5)

glUseProgram(cubes_program)
if g_ispreloaded:
    glUseProgram(shader_program)
    frame_data = g_motion_data[g_current_frame] if g_motion_active else None
    draw_bvh_objects(g_bvhroot, glm.mat4(1.0), frame_data, loc_MVP,MVP)
```

처음 T pose에서 스페이스바를 누르면 움직임을 시작한다. 다시 누르면 T pose로 돌아간다. 끝까지 재생되면 처음으로 돌아가 다시 재생된다.

유튜브 영상 링크. 0:55초부터

https://youtu.be/9fhxG8 bm8B4?si=PfehZ61ir4D8 1tBL