게임엔진

제10강 캐릭터 컨트롤러

학습 안내

■ 학습 목표

□ 씬노드의 구성 및 회전 방법을 응용하여, 구면 카메라 및 캐릭터 컨트롤을 구현해본다.

■ 학습 내용

□ 구면 카메라 구현을 위한 씬노드 구성 및 회전

□ 캐릭터 컨트롤을 위한 씬노구 구성 및 회전

카메라 및 캐릭터 컨트롤 구현 목표

- **카메라 컨트롤**
  - 게임 카메라 컨트롤(구면 카메라)
    - 마우스를 이용한 좌우 패닝, 상하 피칭.
    - 휠 스크롤을 이용한 줌인 및 줌아웃.
    - 캐릭터를 중심으로 회전됨.
  - 카메라를 회전시킬 때, 캐릭터는 자신의 상태 유지.

- **캐릭터 컨트롤**
    - WASD 를 이용한 캐릭터의 전후진이동, 좌우이동, 대각선 이동

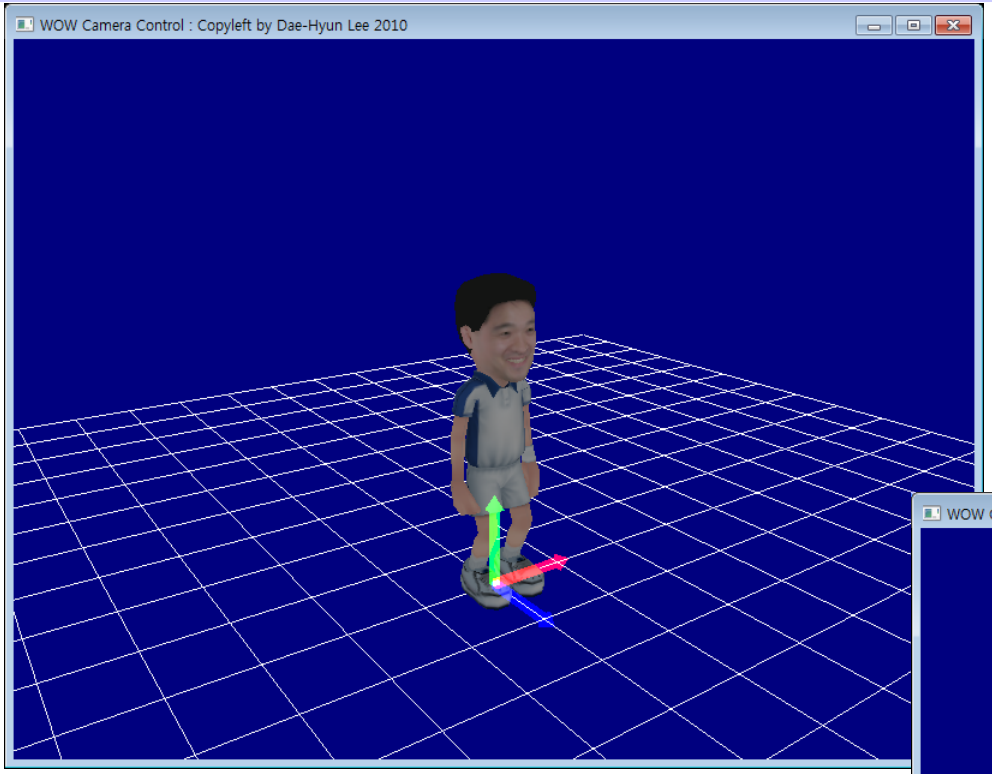
SphericalCamera
구면 카메라 구현

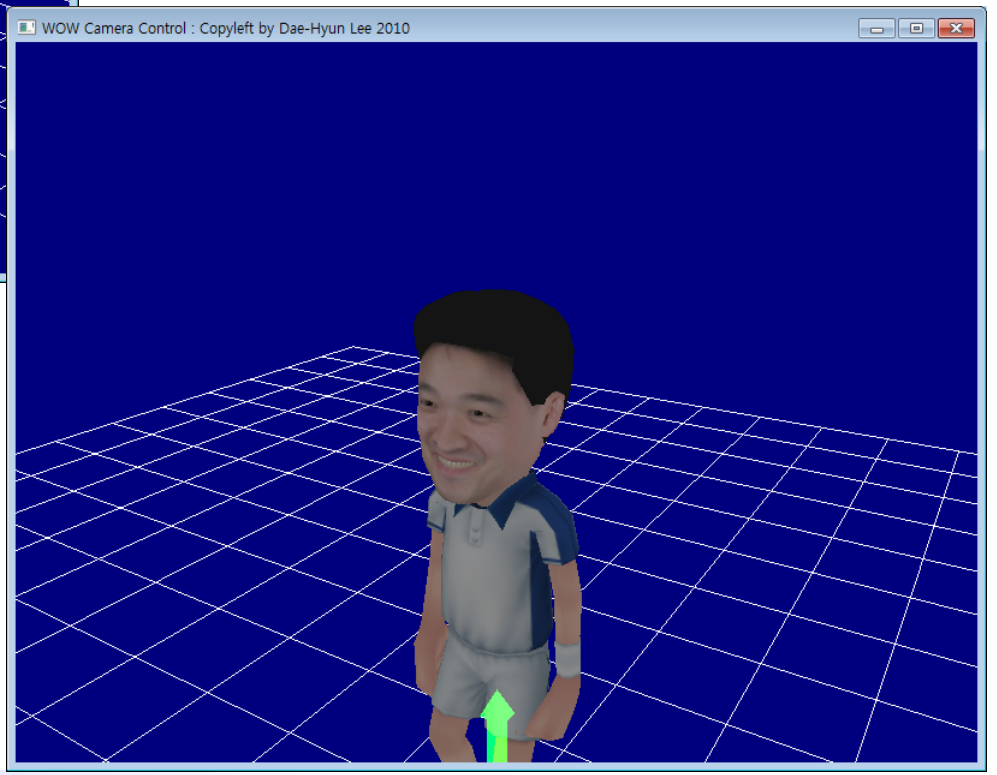

```
bool mouseMoved( const OIS::MouseEvent &evt )
{
  mCameraYaw->yaw(Degree(-evt.state.X.rel));
  mCameraPitch->pitch(Degree(-evt.state.Y.rel));

  mCameraHolder->translate(Ogre::Vector3(0, 0, -evt.state.Z.rel * 0.1f));

  return true;
}
```

class LectureApp


```
void go(void)
{
  ... 중략 ...
  SceneNode* cameraYaw =
    professorRoot->createChildSceneNode("CameraYaw",Vector3(0.0f,120.0f,0.0f));

  SceneNode* cameraPitch =
    cameraYaw->createChildSceneNode("CameraPitch");

  SceneNode* cameraHolder =
    cameraPitch->createChildSceneNode("CameraHolder",Vector3(0.0f,80.0f,500.0f));
  ... 중략 ...
}
```

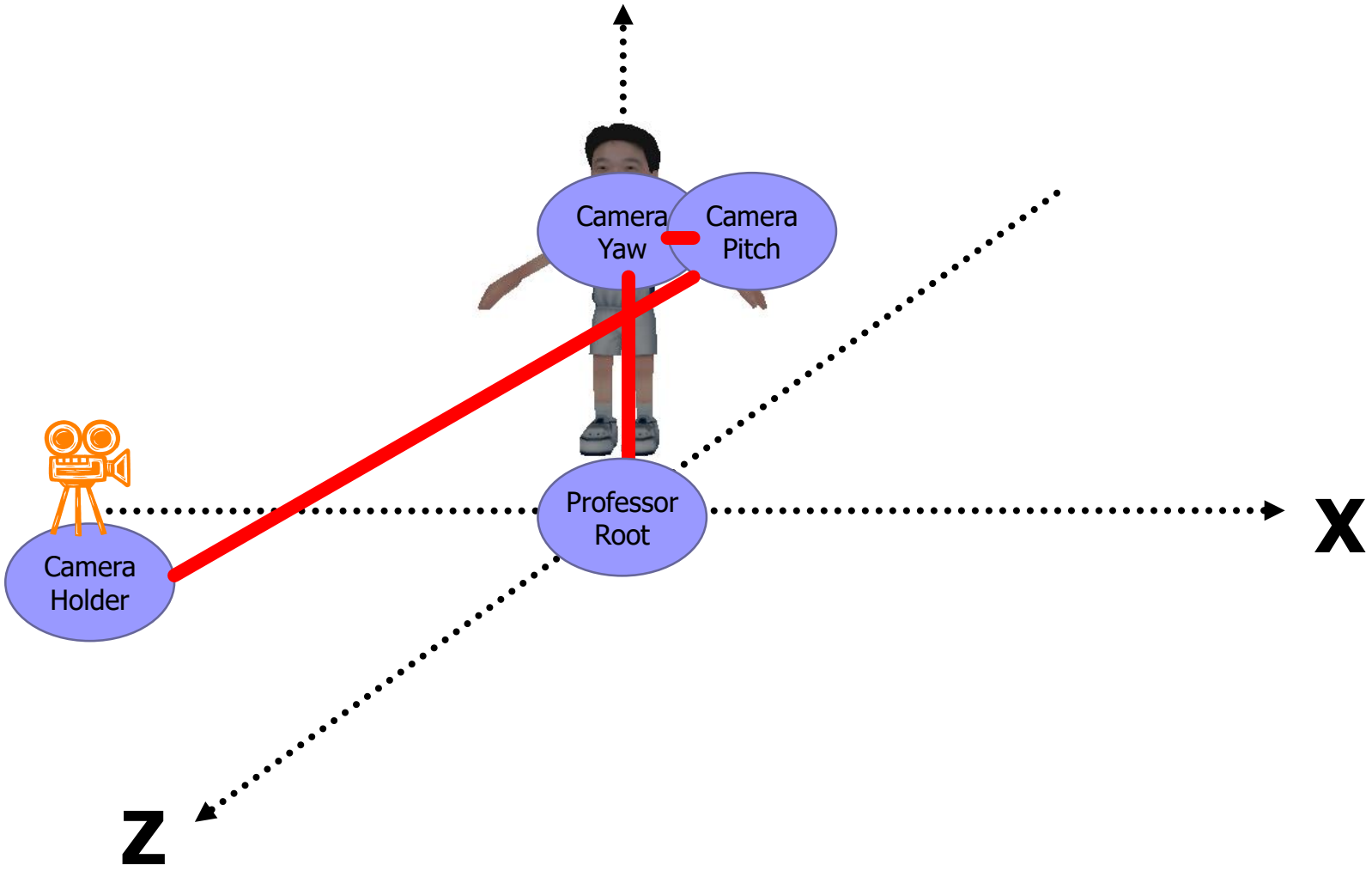
실행 결과 - 마우스를 이용한 구면 카메라 컨트롤



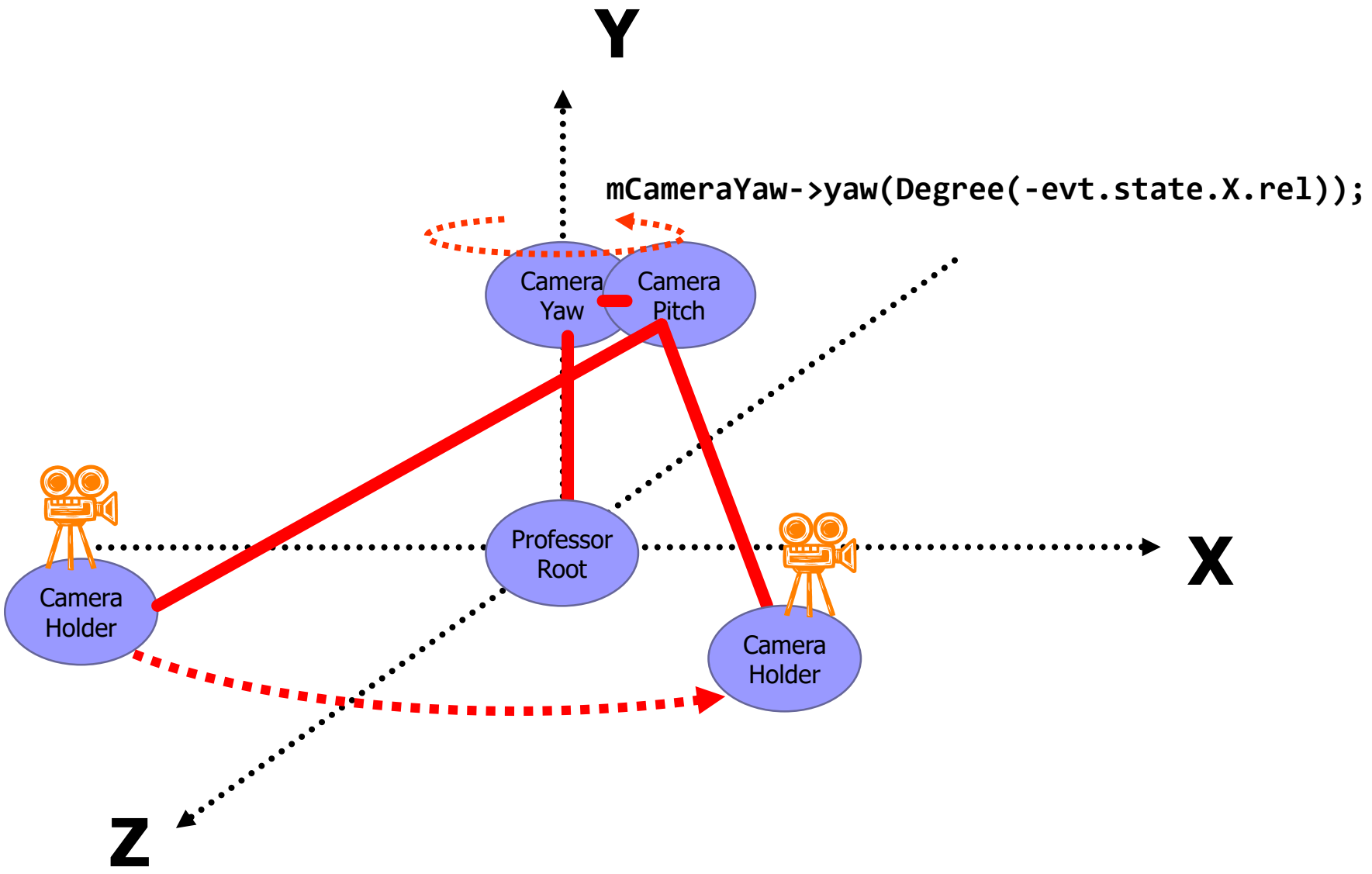


카메라 컨트롤 및 홀더 노드 설정

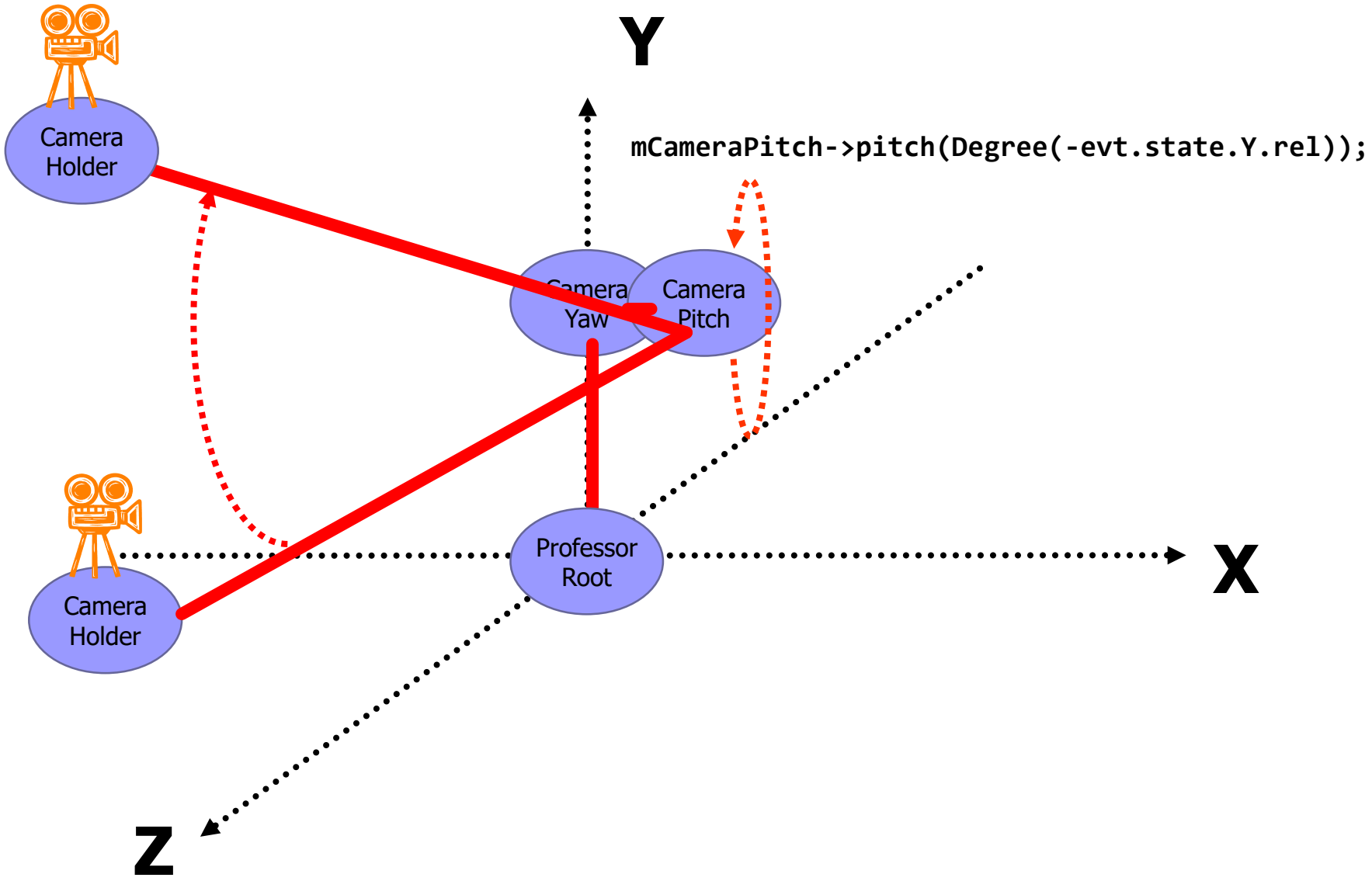
```
SceneNode* cameraYaw = professorRoot->createChildSceneNode("CameraYaw",Vector3(0.0f,120.0f,0.0f));
SceneNode* cameraPitch = cameraYaw->createChildSceneNode("CameraPitch");
SceneNode* cameraHolder = cameraPitch->createChildSceneNode("CameraHolder",Vector3(0.0f,80.0f,500.0f));
```



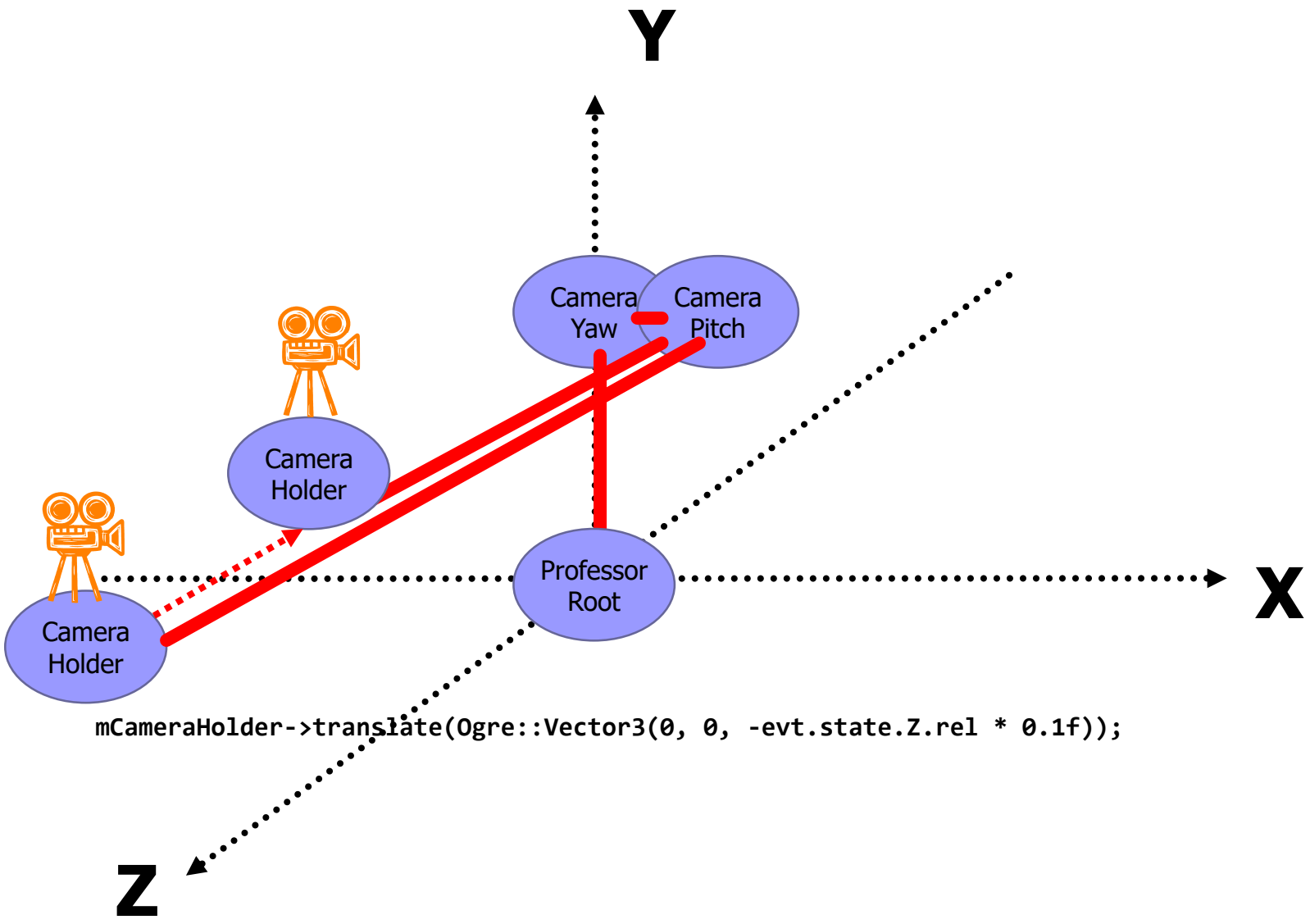
Yaw Control



Pitch Control



Zoom Control




```
mCameraHolder->translate(Ogre::Vector3(0, 0, -evt.state.Z.rel * 0.1f));
```

실습

CharacterController
방향키와 마우스를 이용한 캐릭터 이동 구현

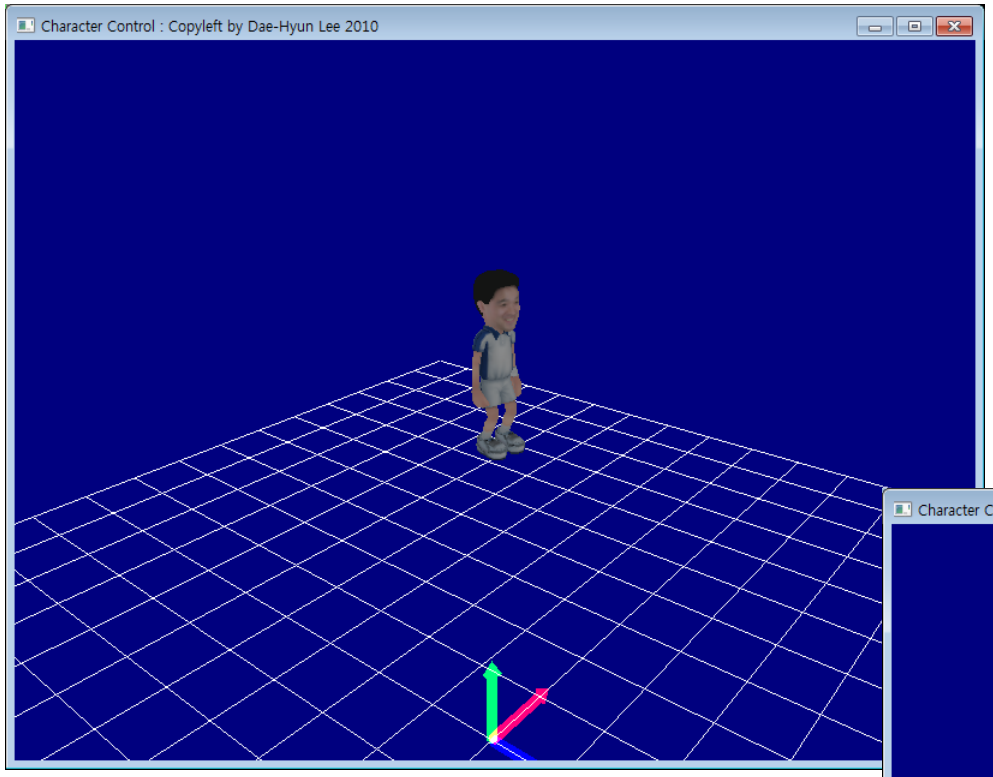

```
bool frameStarted(const FrameEvent &evt)
{
      ... 중략 ...
     if (mCharacterDirection != Vector3::ZERO)
     {
       mCharacterRoot->setOrientation(mCameraYaw->getOrientation());
       Quaternion quat = Vector3(Vector3::UNIT_Z).getRotationTo(mCharacterDirection);
       mCharacterYaw->setOrientation(quat);
       mCharacterRoot->translate(mCharacterDirection.normalisedCopy() * 111 * evt.timeSinceLastFrame
         , Node::TransformSpace::TS_LOCAL);

       if (!mWalkState->getEnabled())
       {
         mWalkState->setEnabled(true);
         mIdleState->setEnabled(false);
       }
       mWalkState->addTime(evt.timeSinceLastFrame);
     }
     else
     {
       if (!mIdleState->getEnabled())
       {
         mIdleState->setEnabled(true);
         mWalkState->setEnabled(false);
       }
       mIdleState->addTime(evt.timeSinceLastFrame);
     }
     ... 중략 ...
}
```

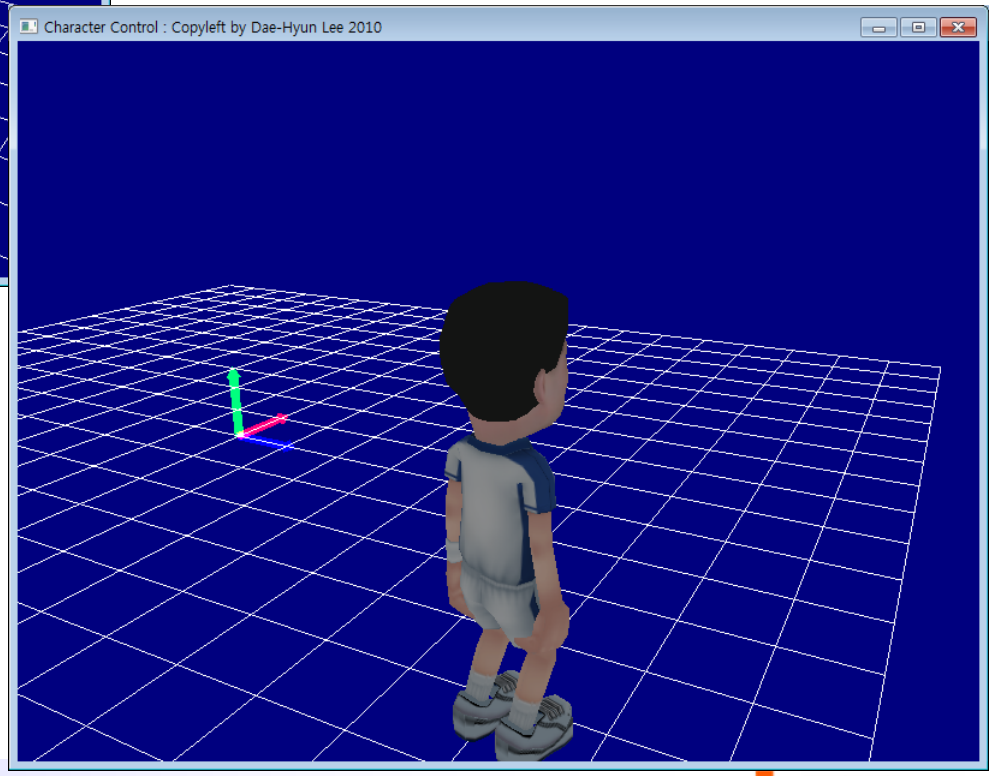

```
bool keyPressed( const OIS::KeyEvent &evt )
{
  switch(evt.key)
  {
  case OIS::KC_W: case OIS::KC_UP: mCharacterDirection.z += -1.0f; break;
  case OIS::KC_S: case OIS::KC_DOWN: mCharacterDirection.z += 1.0f; break;
  case OIS::KC_A: case OIS::KC_LEFT: mCharacterDirection.x += -1.0f; break;
  case OIS::KC_D: case OIS::KC_RIGHT: mCharacterDirection.x += 1.0f; break;
  case OIS::KC_ESCAPE: mContinue = false; break;
  }
  return true;
}

bool keyReleased( const OIS::KeyEvent &evt )
{
  switch(evt.key)
  {
  case OIS::KC_W: case OIS::KC_UP: mCharacterDirection.z -= -1.0f; break;
  case OIS::KC_S: case OIS::KC_DOWN: mCharacterDirection.z -= 1.0f; break;
  case OIS::KC_A: case OIS::KC_LEFT: mCharacterDirection.x -= -1.0f; break;
  case OIS::KC_D: case OIS::KC_RIGHT: mCharacterDirection.x -= 1.0f; break;
  case OIS::KC_ESCAPE: mContinue = false; break;
  }
  return true;
}
```

class LectureApp


```
void go(void)
{
  ... 중략 ...

  SceneNode* professorRoot =
    mSceneMgr->getRootSceneNode()->createChildSceneNode("ProfessorRoot");
  SceneNode* professorYaw =
    professorRoot->createChildSceneNode("ProfessorYaw");

  SceneNode* cameraYaw =
    professorRoot->createChildSceneNode("CameraYaw", Vector3(0.0f, 120.0f, 0.0f));
  SceneNode* cameraPitch =
    cameraYaw->createChildSceneNode("CameraPitch");
  SceneNode* cameraHolder =
    cameraPitch->createChildSceneNode("CameraHolder", Vector3(0.0f, 80.0f, 500.0f));

  cameraYaw->setInheritOrientation(false);

  ... 중략 ...
}
```

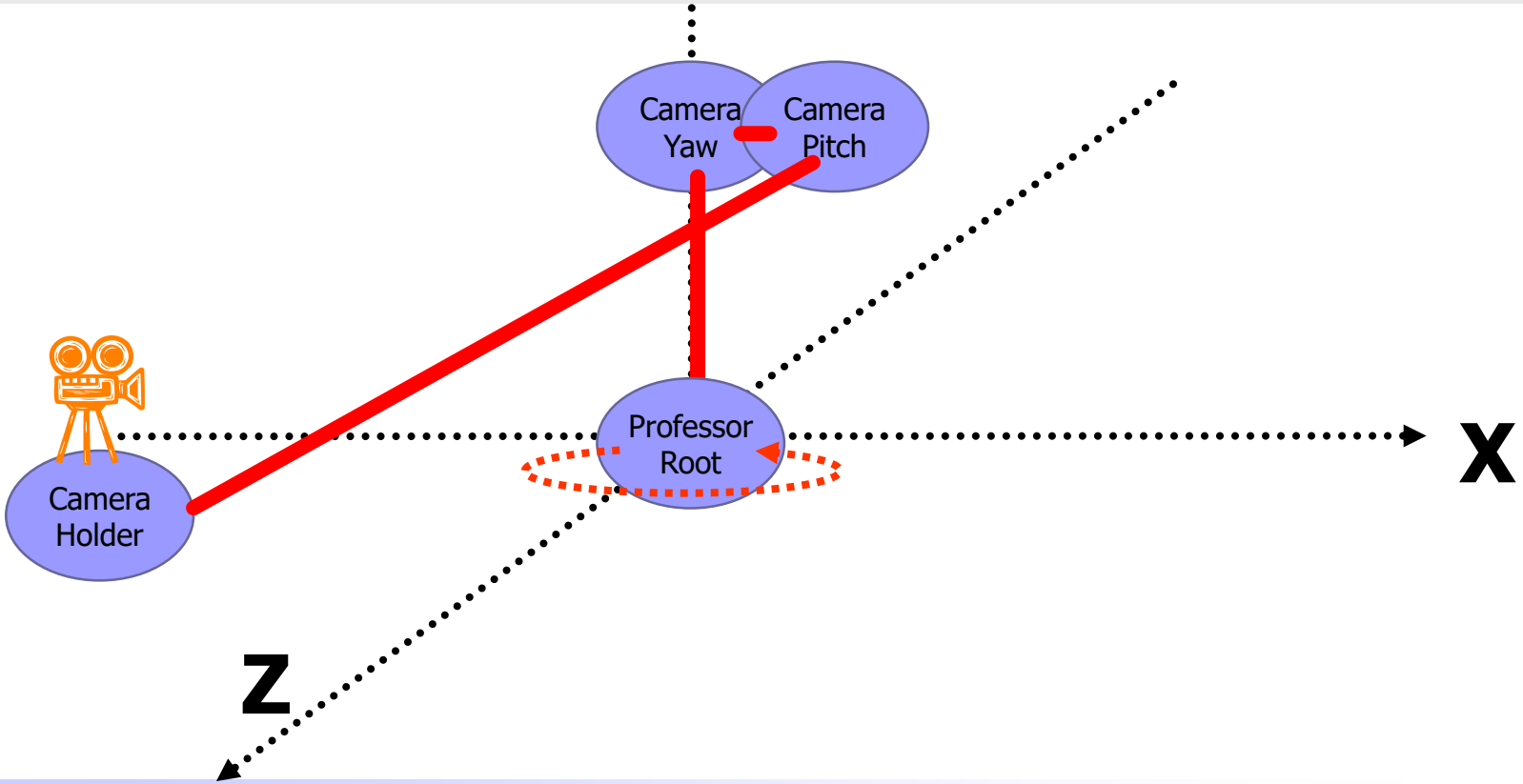
실행 결과 - WASD 캐릭터 이동 + 마우스 카메라 조정





캐릭터와 카메라의 회전 연결 차단

```
SceneNode* professorRoot = mSceneMgr->getRootSceneNode()->createChildSceneNode("ProfessorRoot");
SceneNode* professorYaw = professorRoot->createChildSceneNode("ProfessorYaw");

SceneNode* cameraYaw = professorRoot->createChildSceneNode("CameraYaw", Vector3(0.0f, 120.0f, 0.0f));
SceneNode* cameraPitch = cameraYaw->createChildSceneNode("CameraPitch");
SceneNode* cameraHolder = cameraPitch->createChildSceneNode("CameraHolder", Vector3(0.0f, 80.0f, 500.
    0f));

cameraYaw->setInheritOrientation(false);
```



카메라 방향과 캐릭터 방향의 조정

```
if (mCharacterDirection != Vector3::ZERO)
    {
      // 방향 전환 운동이 시작될 때, 캐릭터의 축을 카메라축과 align
      mCharacterRoot->setOrientation(mCameraYaw->getOrientation());

      // 캐릭터의 바라보는 방향을 전환
      Quaternion quat = Vector3(Vector3::UNIT_Z).getRotationTo(mCharacterDirection);
      mCharacterYaw->setOrientation(quat);

      // 캐릭터 축을 기준으로 하여, 캐릭터 이동(속도: 111cm / sec = 약 4km / sec)
      // Character Root의 local space를 기준으로 이동
      mCharacterRoot->translate(mCharacterDirection.normalisedCopy() * 111 * evt.timeS
   inceLastFrame, Node::TransformSpace::TS_LOCAL);
```

학습 정리

- **구면 카메라 컨트롤**
  - 카메라의 Yaw, Pitch 및 Zoom 을 위한 씬노드를 각각 따로 둠.

- **캐릭터 컨트롤**
  - 캐릭터와 카메라의 회전을 분리함 - 독립적으로 회전 설정.
  - Yaw Control을 이용하여 자체에서 회전.