#include "OGRE\ExampleApplication.h"

using namespace Ogre;

class myFrameListener : public FrameListener{

private:

 Camera\* \_cam;

 SceneNode\* \_node;

 float speed = 50.f;

 Timer \_timer;

 bool bWalked;

 float \_rotation;

 Vector3 dir = Vector3(0, 0, 0);

 OIS::ParamList pl;

 size\_t windowHnd = 0;

 std::stringstream windowHndStr;

 OIS::InputManager\* m\_InputManager;

 OIS::Keyboard\* m\_Keyboard;

 OIS::Mouse\* m\_Mouse;

 Entity\* \_ent;

 AnimationState\* \_animStateBase;

 AnimationState\* \_animStateTop;

 AnimationState\* \_animStateSliceH;

 AnimationState\* \_animStateSliceV;

public:

 myFrameListener(SceneNode\* node, RenderWindow\* m\_window, Camera\* camera, Entity\* ent){

 \_timer.reset();

 \_ent = ent;

 \_cam = camera;

 \_node = node;

 m\_window->getCustomAttribute("WINDOW", &windowHnd);

 windowHndStr << windowHnd;

 pl.insert(std::make\_pair(std::string("WINDOW"), windowHndStr.str()));

 m\_InputManager = OIS::InputManager::createInputSystem(pl);

 m\_Keyboard = static\_cast<OIS::Keyboard\*>(m\_InputManager->createInputObject(OIS::OISKeyboard, false));

 m\_Mouse = static\_cast<OIS::Mouse\*>(m\_InputManager->createInputObject(OIS::OISMouse, false));

 \_animStateBase = \_ent->getAnimationState("RunBase");

 \_animStateBase->setEnabled(false);

 \_animStateBase->setLoop(false);

 \_animStateTop = \_ent->getAnimationState("RunTop");

 \_animStateTop->setEnabled(false);

 \_animStateTop->setLoop(false);

 \_animStateSliceH = \_ent->getAnimationState("SliceHorizontal");

 \_animStateSliceH->setEnabled(false);

 \_animStateSliceH->setLoop(false);

 \_animStateSliceV = \_ent->getAnimationState("SliceVertical");

 \_animStateSliceV->setEnabled(false);

 \_animStateSliceV->setLoop(false);

 }

 ~myFrameListener(){

 if (m\_InputManager){

 m\_InputManager->destroyInputObject(m\_Keyboard);

 m\_InputManager->destroyInputObject(m\_Mouse);

 OIS::InputManager::destroyInputSystem(m\_InputManager);

 }

 }

 bool frameStarted(const FrameEvent &evt){

 m\_Keyboard->capture();

 m\_Mouse->capture();

 if (m\_Keyboard->isKeyDown(OIS::KC\_ESCAPE)){

 return false;

 }

 dir = Vector3(0, 0, 0);

 bWalked = false;

 if (m\_Keyboard->isKeyDown(OIS::KC\_W)){

 dir += Vector3(0, 0, 1);

 bWalked = true;

 \_rotation = 0.0f;

 }

 if (m\_Keyboard->isKeyDown(OIS::KC\_S)){

 dir += Vector3(0, 0, -1);

 bWalked = true;

 \_rotation = 3.14f;

 }

 if (m\_Keyboard->isKeyDown(OIS::KC\_A)){

 dir += Vector3(1, 0, 0);

 bWalked = true;

 \_rotation = 1.57f;

 }

 if (m\_Keyboard->isKeyDown(OIS::KC\_D)){

 dir += Vector3(-1, 0, 0);

 bWalked = true;

 \_rotation = -1.57f;

 }

 if (m\_Keyboard->isKeyDown(OIS::KC\_LSHIFT)){

 if (\_timer.getMilliseconds() > 200){

 speed -= 5;

 if (speed < 0)

 speed = 0;

 \_timer.reset();

 }

 }

 if (m\_Keyboard->isKeyDown(OIS::KC\_RSHIFT)){

 if (\_timer.getMilliseconds() > 200){

 speed += 5;

 if (speed > 1000)

 speed = 1000;

 \_timer.reset();

 }

 }

 if (bWalked){

 \_animStateBase->setEnabled(true);

 \_animStateTop->setEnabled(true);

 if (\_animStateBase->hasEnded()){

 \_animStateBase->setTimePosition(0.0f);

 }

 if (\_animStateTop->hasEnded()){

 \_animStateTop->setTimePosition(0.0f);

 }

 }

 else{

 \_animStateBase->setEnabled(false);

 \_animStateTop->setEnabled(false);

 \_animStateBase->setTimePosition(0.0f);

 \_animStateTop->setTimePosition(0.0f);

 }

 \_animStateBase->addTime(evt.timeSinceLastFrame);

 \_animStateTop->addTime(evt.timeSinceLastFrame);

 \_node->translate(speed \* dir \* evt.timeSinceLastFrame);

 \_node->getChild("SINBAD\_NODE")->resetOrientation();

 \_node->getChild("SINBAD\_NODE")->yaw(Radian(\_rotation));

 float rotX = m\_Mouse->getMouseState().X.rel \*evt.timeSinceLastFrame \* -1;

 float rotY = m\_Mouse->getMouseState().Y.rel \*evt.timeSinceLastFrame \* -1;

 \_cam->yaw(Radian(rotX));

 \_cam->pitch(Radian(rotY));

 if (m\_Mouse->getMouseState().buttonDown(OIS::MB\_Left)){

 \_animStateSliceH->setEnabled(true);

 if (\_animStateSliceH->hasEnded()){

 \_animStateSliceH->setTimePosition(0.0f);

 }

 }

 else{

 \_animStateSliceH->setEnabled(false);

 \_animStateSliceH->setTimePosition(0.0f);

 }

 \_animStateSliceH->addTime(evt.timeSinceLastFrame \*0.75f);

 if (m\_Mouse->getMouseState().buttonDown(OIS::MB\_Right)){

 \_animStateSliceV->setEnabled(true);

 if (\_animStateSliceV->hasEnded()){

 \_animStateSliceV->setTimePosition(0.0f);

 }

 }

 else{

 \_animStateSliceV->setEnabled(false);

 \_animStateSliceV->setTimePosition(0.0f);

 }

 \_animStateSliceV->addTime(evt.timeSinceLastFrame \*0.75f);

 return true;

 }

};

class myInteractiveExampleApplication : public ExampleApplication{

private:

 SceneNode\* \_SinbadNode;

 myFrameListener\* myListener;

 Entity\* Sinbad;

 SceneNode\* node;

public:

 void createScene(){

 Plane groundPlane(Vector3::UNIT\_Y, -10);

 MeshManager::getSingleton().createPlane("PLANE\_01", ResourceGroupManager::DEFAULT\_RESOURCE\_GROUP\_NAME, groundPlane,

 1500, 1500, 200, 200, true, 1, 5, 5, Vector3::UNIT\_Z);

 Entity\* planeEnt = mSceneMgr->createEntity("GROUND\_PLANE\_ENT\_01", "PLANE\_01");

 mSceneMgr->getRootSceneNode()->createChildSceneNode()->attachObject(planeEnt);

 planeEnt->setMaterialName("Examples/BeachStones");

 Light\* light = mSceneMgr->createLight("LIGHT\_01");

 light->setType(Light::LT\_DIRECTIONAL);

 light->setDirection(Vector3(1, -1, 0));

 node = mSceneMgr->createSceneNode("NODE\_01");

 mSceneMgr->getRootSceneNode()->addChild(node);

 Sinbad = mSceneMgr->createEntity("SINBAD\_01", "Sinbad.mesh");

 \_SinbadNode = node->createChildSceneNode("SINBAD\_NODE");

 \_SinbadNode->setScale(5, 5, 5);

 \_SinbadNode->setPosition(0, 15, 0);

 \_SinbadNode->attachObject(Sinbad);

 SceneNode\* camNode = node->createChildSceneNode("TP\_CAM\_NODE");

 camNode->attachObject(mCamera);

 camNode->translate(0, 50, -100);

 mCamera->lookAt(0, 0, 0);

 mCamera->setNearClipDistance(5);

 AnimationStateSet\* set = Sinbad->getAllAnimationStates();

 AnimationStateIterator itr = set->getAnimationStateIterator();

 while (itr.hasMoreElements()){

 std::cout << "Animation: " << itr.getNext()->getAnimationName() << std::endl;

 }

 Entity\* Sword1 = mSceneMgr->createEntity("SWORD\_01", "Sword.mesh");

 Sinbad->attachObjectToBone("Handle.R", Sword1);

 Entity\* Sword2 = mSceneMgr->createEntity("SWORD\_02", "Sword.mesh");

 Sinbad->attachObjectToBone("Handle.L", Sword2);

 }

 void createCamera(){

 mCamera = mSceneMgr->createCamera("MY\_CAMERA\_01");

 }

 void createFrameListener(){

 myListener = new myFrameListener(node, mWindow, mCamera, Sinbad);

 mRoot->addFrameListener(myListener);

 }

 myInteractiveExampleApplication(){

 myListener = NULL;

 }

 ~myInteractiveExampleApplication(){

 if (myListener)

 {

 delete myListener;

 }

 }

};

int main(){

 myInteractiveExampleApplication myGame;

 myGame.go();

 return 0;

}