CS425 GAME PROGRAMMING

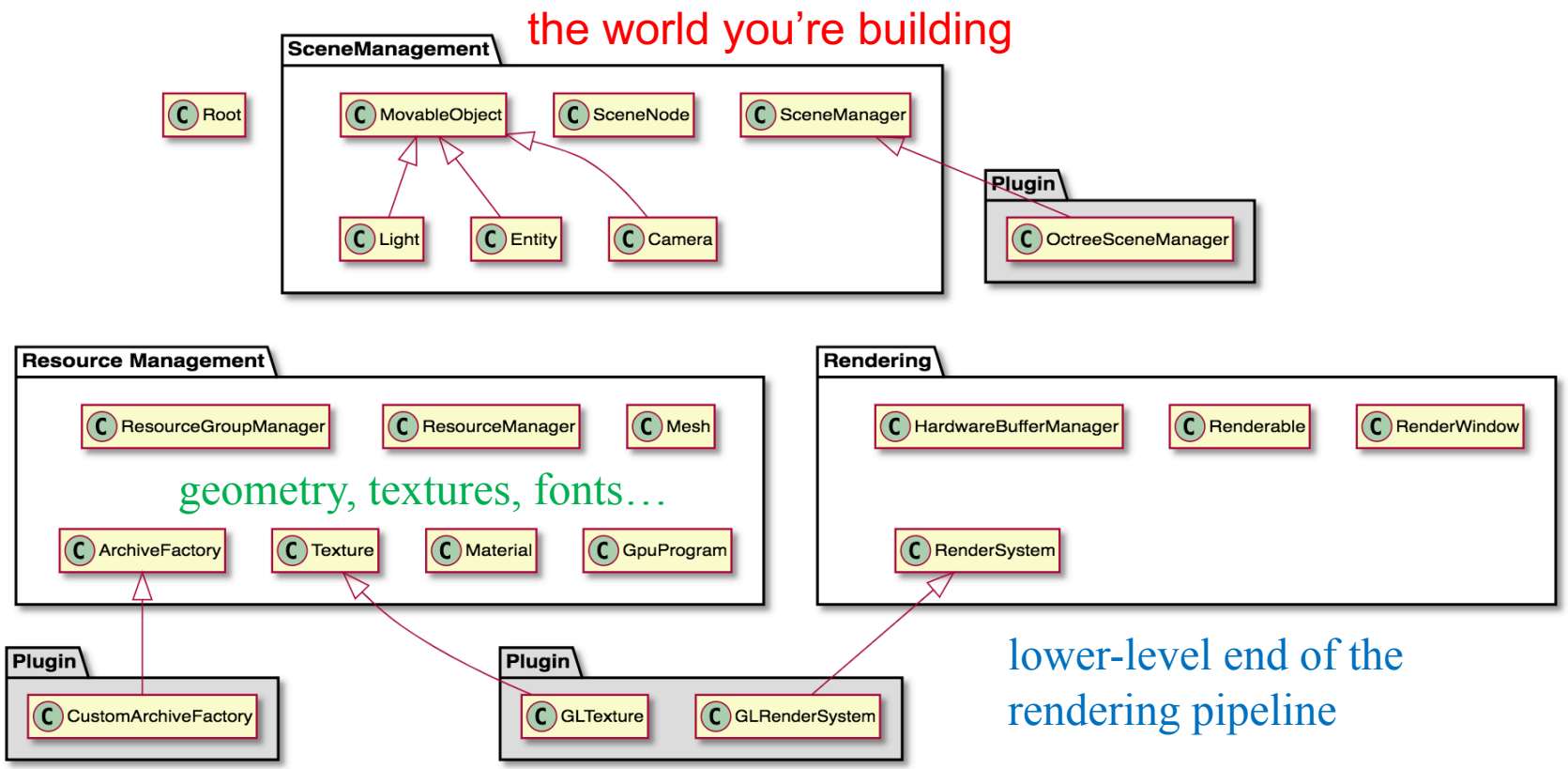
Lecture 02 – OGRE (**O**bject-Oriented **G**raphics **R**endering **E**ngine)

Today

- Introduction to OGRE
- The main goal is not to teach you about OGRE so that you know OGRE.
- The goal is to:
  - Provide you with enough knowledge to build on top of OGRE.
  - Give you a look at how an engine is set up (they have similarities and differences)
  - Refresh your memory on C++

Overview from 10,000 feet

https://ogrecave.github.io/ogre/api/latest/_the-_core-_objects.html



Scene Manager

- SceneManager keeps track of the locations and other attributes of the objects
- Also manages cameras
- mRoot->createSceneManager(ST_GENERIC);
  - DefaultSceneManager
  - OctreeSceneManager
  - BspSceneManager
  - PCZSceneManager
  - …

SceneNode and Entity

- SceneNodes carry information that is used for all of the objects that are attached to it
  - Transforms
  - Part of a scene graph
  - Contain one or multiple entities
- Entity
  - 3D Mesh
  - Light, Particles, and Cameras are not entities
  - Must be attached to a SceneNode before rendering

OGRE Program Flow

- mRoot = new Ogre::Root(Ogre::StringUtil::BLANK);
- Choose Scene Manager
- Create Camera
- Create Viewports
- Load Resources
- Create Scene //Your main task
- Create Frame Listener
- mRoot→startRendering();
- Destroy Scene

Choose Scene Manager

- // Get the SceneManager, in this case a generic one
- mSceneMgr = mRoot → createSceneManager(Ogre::ST_GENERIC);

- //overlay is for rendering heads-up-displays, menus or other layers
- //on top of the contents of the scene
- mOverlaySystem = new Ogre::OverlaySystem();

- mSceneMgr → addRenderQueueListener(mOverlaySystem);

Create Camera

- // Create the camera
- mCamera = mSceneMgr → createCamera("PlayerCam");

- // Position it at 500 in Z direction
- mCamera → setPosition(Ogre::Vector3(0,0,80));

- // Look back along -Z
- mCamera → lookAt(Ogre::Vector3(0,0,-300));
- mCamera → setNearClipDistance(5);

- // create a default camera controller
- mCameraMan = new OgreBites::SdkCameraMan(mCamera);

Create Viewports

- // Create one viewport, entire window
- Ogre::Viewport* vp = mWindow → addViewport(mCamera);
- vp → setBackgroundColour(Ogre::ColourValue(0,0,0));

- // Alter the camera aspect ratio to match the viewport
- mCamera → setAspectRatio(Ogre::Real(vp → getActualWidth()) / Ogre::Real(vp → getActualHeight()));

Load Resources

```
// Load resource paths from config file
    Ogre::ConfigFile cf;
    cf.load(mResourcesCfg);

// Go through all sections & settings in the file
    Ogre::ConfigFile::SectionIterator seci = cf.getSectionIterator();

    Ogre::String secName, typeName, archName;
    while (seci.hasMoreElements())
    {
        secName = seci.peekNextKey();
        Ogre::ConfigFile::SettingsMultiMap *settings = seci.getNext();
        Ogre::ConfigFile::SettingsMultiMap::iterator i;
        for (i = settings → begin(); i != settings → end(); ++i)
        {
            typeName = i → first;
            archName = i → second;
            Ogre::ResourceGroupManager::getSingleton().addResourceLocation(
                archName, typeName, secName);
        }
    }
```

Window/Event Management

- class BaseApplication : public Ogre::FrameListener, public Ogre::WindowEventListener, public OIS::KeyListener, public OIS::MouseListener, OgreBites::SdkTrayListener

- // Ogre::FrameListener
- virtual bool frameRenderingQueued(const Ogre::FrameEvent& evt);

- // OIS::KeyListener
- virtual bool keyPressed(const OIS::KeyEvent &arg);
- virtual bool keyReleased(const OIS::KeyEvent &arg);

- // OIS::MouseListener
- virtual bool mouseMoved(const OIS::MouseEvent &arg);
- virtual bool mousePressed(const OIS::MouseEvent &arg, OIS::MouseButtonID id);
- virtual bool mouseReleased(const OIS::MouseEvent &arg, OIS::MouseButtonID id);

- // Ogre::WindowEventListener and djust mouse clipping area
- virtual void windowResized(Ogre::RenderWindow* rw);

- //Unattach OIS before window shutdown (very important under Linux)
- virtual void windowClosed(Ogre::RenderWindow* rw);

Create Event/Frame Listener

- OIS::InputManager::createInputSystem(…);
- // dose somethings about keyboard and mouse…
- // Register as a Window listener
- Ogre::WindowEventUtilities::addWindowEventListener(window, this);
- mTrayMgr = new OgreBites::SdkTrayManager(”bla", window, context, this);
- mTrayMgr → showFrameStats(OgreBites::TL_BOTTOMLEFT);
- mTrayMgr → hideCursor();
- mRoot → addFrameListener(this);

Demo

- CS425App-01-Tutorial 1

Create Scene

- Load Env
- Setup Env
- Load Objects();
- Load Characters();

Load Env

- //create a floor mesh resource
  - Ogre::MeshManager::getSingleton().createPlane("floor", ResourceGroupManager::DEFAULT_RESOURCE_GROUP_NAME, Plane(Vector3::UNIT_Y, 0), 100, 100, 10, 10, true, 1, 10, 10, Vector3::UNIT_Z);

- //create a floor entity, give it material, and place it at the origin
- Entity* floor = mSceneMgr → createEntity("Floor", "floor");
- floor → setMaterialName("Examples/Rockwall");
- floor → setCastShadows(false);
- mSceneMgr → getRootSceneNode() → attachObject(floor);

Setup Env

- // set shadow properties
- mSceneMgr → setShadowTechnique(…);
- mSceneMgr → setShadowColour(ColourValue(0, 0, 1));
- mSceneMgr → setShadowTextureSize(1024);
- mSceneMgr → setShadowTextureCount(1);
- mCameraMan → setStyle(OgreBites::CS_FREELOOK);

- // use small amount of ambient lighting
- mSceneMgr → setAmbientLight(ColourValue(0.3f, 0.3f, 0.3f));

- // add a bright light above the scene
- Light* light = mSceneMgr → createLight();
- light → setType(Light::LT_POINT);
- light → setPosition(-10, 40, 20);
- light → setSpecularColour(ColourValue::White);
- light → setDiffuseColour(ColourValue::Green);

Load Objects

```
Ogre::Entity *ent;
Ogre::SceneNode *node;

ent = mSceneMgr→createEntity("Knot1", "knot.mesh");
node = mSceneMgr→getRootSceneNode()→createChildSceneNode(”1", Ogre::Vector3(0.0f, 0.0f,  2.5f));
node → attachObject(ent);
node → setScale(0.1f, 0.1f, 0.1f);

ent = mSceneMgr → createEntity("Knot2", "knot.mesh"); // Names must be unique
node = mSceneMgr→getRootSceneNode()→createChildSceneNode(”2",  Ogre::Vector3(5.0f, 0.0f,  5.0f));
node → attachObject(ent);
node → setScale(0.01f, 0.01f, 0.01f);

ent = mSceneMgr→ createEntity("Knot3", "knot.mesh");
node = mSceneMgr→getRootSceneNode()→createChildSceneNode(”3", Ogre::Vector3(-1.0f, 0.0f,-2.0f));
node → attachObject(ent);
node → setScale(1.1f, 1.1f, 1.1f);
node → yaw(Degree(90));
```

Load Characters

- mBodyNode = mSceneMgr → getRootSceneNode() → createChildSceneNode();
- mBodyEntity = mSceneMgr → createEntity("Sinbad", "Sinbad.mesh");
- mBodyNode → attachObject(mBodyEntity);

- // make the Ogre stand on the plane
- mBodyNode → translate(0,5,0);

- //What does the following code do?
- Ogre::Entity* mEnt = mSceneMgr → getEntity("Knot1");
- mEnt → detachFromParent();
- mBodyNode → attachObject(mEnt);

Demo 2

- CS425App-02-Simple Scene