Outdoor AR Library Tutorial
Creating Custom Scene Classes
by Tristan Scott

The Human Interface Technology Laboratory New Zealand
http://www.hitlabnz.org/mobileAR

This tutorial shows how to implement custom scene classes which will allow developers to have their own data structure and methods in scene objects used in the OutdoorAR library.

A scene is an element that can be displayed on AR, Map or List Component activities. A simple scene would consist of one data model but may contain many. This tutorial illustrates how to create two scenes that each use differing data model types, one displaying an image using a billboard data model and another that uses a .OBJ file to display a 3D object. The fundamentals of doing both are essentially the same with the only difference being the model data that is being used.

Creating a Scene

To create a scene your custom class must extend OAScene:

```java
public class CustomScene extends OAScene{
}
```

After doing this give your scene a name. This name is what will be displayed when we implement the onSceneSelected callback in the OAARComponentBase activity that this scene will be added to.

```java
this.setName("Scene name");
```

Next the scene must have a location object created of which it can refer to for its position. Using the location object set the latitude, longitude and altitude of where the scene is to be located and set this scene's location to the object.

```java
SceneLocation location = new SceneLocation();
location.setLatitude(-43.522313);
location.setLongitude(172.583233);
location.setAltitude(0);
this.location = location;
```

Now that the scene has a name and a location we can specify a model (or models) that will be contained in the scene.
ModelData model = new ModelData();
    model.name = "House";
    model.id = "HouseData";

Here we have defined a model named house whose appearance and rendering information can be found in a file called HouseData. Add this model to the list of models that the scene may contain.

    this.models.add(model);

Now that all the necessary objects are defined in order to make the custom scene display, it is possible to truly customize your scene by adding fields and methods to suit the needs required.

    public String houseColour = "green";
    public int numberOfRooms = 7;

Here a house color and number of rooms are added to the house scene.

When the scene modifications are complete it is added to any class extending OAARComponentBase and the class will display the object defined in the scene at the location specified in the scene.

**TutorialCustomScene**

The TutorialCustomScene demonstrates how to implement custom scenes with two different types of data model. The CustomCatScene class describes a 3D cat and retrieves its data from files stored locally to the program. The CustomFlowerScene class describes an image of flowers and uses the billboardModelData class (see Tutorial ARTags) defined in the AR library.

**CustomFlowerScene**

A reference to the resources we are using must be passed into the constructor of the class extending OAScene in order to locate the image to be displayed during runtime. A name and location are created as described in creating a scene, above. The id field of the OAScene superclass does not need to be overridden as the Billboard model data class contains all the rendering information. Further customisation of the scene is provided by adding a public field that describes the type of flowers.

    public String flowerType = "Daisies";

A BillboardModelData object is defined that is given the location of the image to display and the dimensions of which the image is to have.
BillboardModelData flowers = new BillboardModelData(resources.getDrawable(R.drawable.flowers), 5,5);

The BillboardModelData object is added to the scene.

this.models.add(flowers);

The scene is further added to the TutorialCustomScene class inside its setUpScenes method.

dataManager.addScene(new CustomFlowerScene(getResources()));

**CustomCatScene**

Once the class structure extending OAScene is defined, a name and location are created as described in creating a scene, above. To truly customise the cat scene a public field called “noise” is added.

public String noise = "meeeow";

Here a Model object is instantiated and provided with a name and id.

ModelData model = new ModelData();
    model.name = "Cat";
    model.id = "MrMeow";

The id is the name of the local file which holds the rendering data. In this particular instance a transformation object is created which scales the cat and rotates it to face the camera.

Transform modelTransform = new Transform();
    modelTransform.setTranslation(new Coordinate(0,-2,0));
    modelTransform.setScale(new Vector3f(5.0f,5.0f,5.0f));
    modelTransform.setRotation(new Vector3f(0.0f,1.0f,0.0f));
    modelTransform.setRotationAngle(150.0f);
    model.addTransform(modelTransform);

After the transformation is applied, the model is added to the scene.

this.models.add(model);

The scene is further added to the TutorialCustomScene class inside its setUpScenes method.
dataManager.addScene(cat);

You now have all information required to create custom scenes. If you have further problems or questions, visit our website (http://www.hitlabnz.org/mobileAR) and post your problem on the Mobile AR Framework Support Forum.