Preparing photorealistic environment using UM Scene module

Tools for creating advanced photorealistic scenes for UM models are considered

2024

Contents

1. Preparing and using photorealistic environment

1.1. Introduction

UM Scene module is aimed for preparing photorealistic environment and includes the collection of 3D objects, textures, sky boxes and tools integrated into **UM Simulation** program.

Collection of 3D objects includes buildings (country houses, apartments, industrial buildings), road sections (straight sections, turns, cross roads), road and street elements (fences, benches, trash bins, road signs, traffic lights, lamps), trees and bushes, static cars, trucks and trailers. Some of the 3D objects mentioned above are presented in the table below.

All 3D models included into UM Scene are licensed for commercial use. One can use those 3D models in any way: create scenes and UM models, take snapshots and record and distribute video files etc.

1.2. Getting started

Step-by-step instructions how to quickly start working with **UM Scene** and make yourself familiar with some sample scenes are considered in this section.

- **Install a collection of standard 3D objects.** Please read Sect. 1.3. "*Structure of 3D object collection*", page 1-5 for more detailed information. Here is the direct link to the collection: http://www.universalmechanism.com/download/umscenecollection.exe.
  A link to the up-to-date version of the collection is published on UM web site in the section "*Download - Universal Mechanism / Installation Packages*", see http://www.universalmechanism.com/en/pages/index.php?id=3.
- **Make sure that you have a license for UM Scene in your UM configuration.** Every module in UM software is licensed separately. To load scenes a UM user needs the **UM Scene** module to be available. To check if you have the **UM Scene** module run **UM Simulation** program then from the main menu select the **Help | About** item and make sure that you have **'+'** sign for **UM Scene**, see Figure 1.1. If you use a trial UM license, your UM configuration already includes **UM Scene**. If you are a registered user you can request for a free trial 3-month license via um@universalmechanism.com.

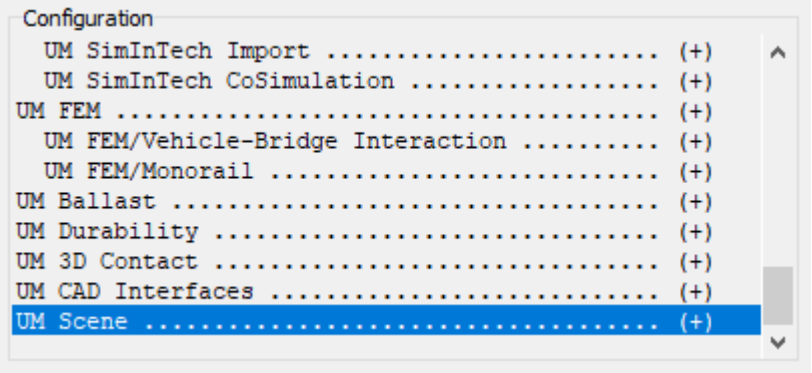


Figure 1.1. UM Scene in the UM configuration

- **Load sample models with preconfigured scenes.** UM installation includes a few sample models with preconfigured scenes.
  - AC4 rail car model includes two configurations with scenes: [**Scene - Railroad through village]** and [**Scene - Railroad through forest]**. To have a look at them run UM Simulation program, load the model from the following folder *{UM Data}\Samples\Rail_Vehicles\ac4*, then using the **File| Load configuration** menu item load configurations listed above, see Figure 1.2.

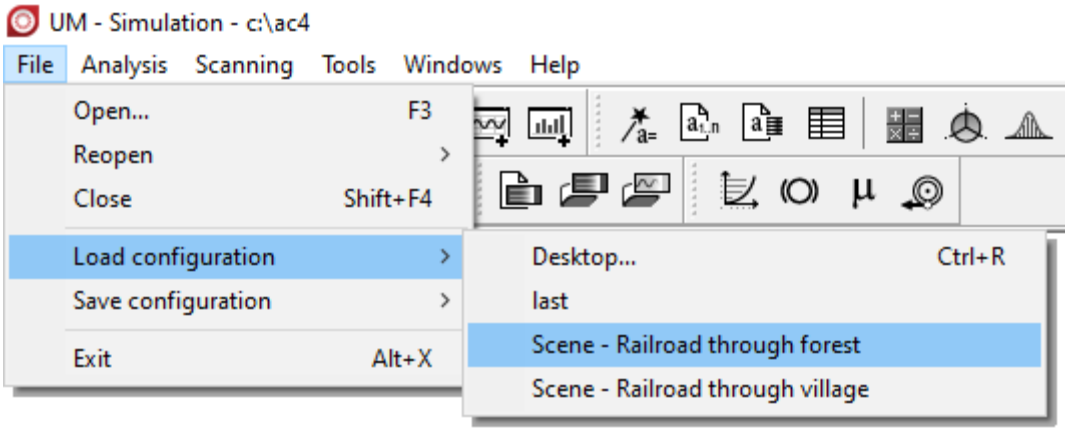


Figure 1.2. Preliminarily prepared configurations for the AC4 model

- Truck and trailer model from the "*{UM Data}\Samples\Automotive\trucktrailer"* folder also includes two configuration with scenes: **[Industrial zone Trajectory 1]** and **[City zone Trajectory 2]**.

1.3. Structure of 3D object collection

3D object collection consists of two parts: 3D objects that are distributed along with Universal Mechanism software (further referred to as "*standard 3D objects*") and user 3D objects. User 3D object collection is empty by default and should be filled in by the user.

Note	Collection of standard 3D objects is a part of Universal Mechanism software but it is not included into the regular UM installation. The collection of the standard 3D objects should be downloaded and installed separately. The last version of the collection is available via the following link http://www.universalmechanism.com/pages/index.php?id=3 in the *Universal Mechanism / Installation Packages* section.

Collection of the standard 3D objects by default is installed to the working directory *{UM Data}\Scene data\Standard objects* and includes a ZIP-archive with 3D models [*StandardObjects.db*] and the XML-file with the object description, their initial position and orientation [*StandardObjects.xml*]. More detailed information about UM working directory is given in the Sect. 1.1.1. *"UM location"*, page 1-3.

In the *{UM data}\Scene data\Scenes* folder some preliminary developed scenes are located.

The reference to the last active scene is saved in the *.icf configuration file in the "*with environment*" structure, field "*SceneFilename*". On the model loading the saved scene is looked for using the direct path specified in the *.icf file, then in the model's folder and finally in the global scene folder *{UM data}\Scene data\Scenes*. Take this into account when you copy your models to other computers.

1.4. Interaction between scene and model elements

Scenes created with the help of the UM Scene module that is considered in this chapter do NOT interact with UM model in mechanical sense – a railway vehicle can run through a house and a car can run through a tree if such scene elements are on their way. One should create *compatible* scenes and models where a car runs on a road and does not run through a building.

If you need to simulate a vehicle dynamics in a particular scene, you should specify its position and orientation so as to have a vehicle in a correct position at the simulation start and also you should specify the vehicle trajectory so as it goes matching the scene objects (buildings and road elements). If you need to move and rotate the whole scene you should use **<SceneCenter>** node of the **<EnvironmentSettings>** node. How to specify a trajectory for a road vehicle is described in Sect. 12.3.1. "*Track macro geometry*". Setting trajectories for railway vehicles is considered in Sect. 8.2. "*Track*".

If you need to simulate a car following some trajectory or a railway vehicle on some specific track then you have to create/modify a scene so as the given a road/track trajectory would be correctly completed by the scene.

Creating a scene and a trajectory are two different tasks. For example, if you developed a city scene with some road UM cannot create any trajectory to correctly run that roads. On the contrary, if you have already prepared trajectory UM cannot automatically create a correspondent road.

Please also note the follows. During the simulation of a road or railway vehicle dynamics in UM software its initial position coincide with the origin of the global frame of reference, X-axis towards forward, Z-axis is vertical. If you need same scene for a few different scenarios with different starting points or directions you have to move and rotate your scene correspondingly to provide correct initial position of the vehicle within the scene. To adjust the scene use nodes **<EnvironmentSettings> / <SceneCenter>** as it was described above in this section.

If you have to specify a mechanical interaction between bodies in your model and scene elements, for example collision between a car and a tree, you should use regular UM approaches. For instance, to simulate such a collision one may use built-in **UM 3D Contact** tool and specify a cone-shape contact manifold for a trunk of the tree (in fact a **Base0** body) and a correspondent contact manifold for a car body and finally describe a contact interaction between two bodies.

UM Scene module does not have any influence on the mechanical part. It is intended for creating advanced scenes and landscapes. The only way how elements of **UM Scene** are involved into the model is that they are detected by the ray sensors. It is described in the 33rd chapter of UM user's manual.

1.5. Position and orientation

Position of objects is introduced by the **<Position>** node of the XML markup and should be expressed in meters, see the example below.

```
<Position X="25" Y="0" Z="0"/>
```

Orientation of objects is introduced by **<Rotation>** node and might be given in three different ways. Rotation angles are set in degrees.

```
<Rotation Type="rtAngle" Z="90"/>
<Rotation Type="rtAngles">
  <Angle Axis="Z" Value="90"/>
</Rotation>
<Rotation Type="rtQuaternion" X="0" Y="0" Z="0.70710678" W="0.70710678"/>
```

Let us consider three ways to specify the object orientation below.

- **Type="rtAngle"**. It specifies the rotation around one of the axis Z, X or Y. The only axis must be specified. If more than one axis is specified, it does not matter in which order the axes are listed; they will be applied with the following priority: Z, X, Y. This way to specify the object orientation is the most widely used. The thing is that the standard 3D objects from the collection are already prepared in the UM-compatible frame of reference where Z-axis is directed upwards, X and Y-axis form the horizontal plane. Thus for creating the most scenes it is enough to specify the only rotation around Z-axis with the help of this type of rotation specification.
- **Type="rtAngles"**. This type of orientation uses the sequence of rotation angles around axes. This type of rotation is identical to one that is used for description of graphic elements within graphic objects and allows specifying any arbitrary relative orientation.
- **Type="rtQuaternion"**. This type uses a quaternion[1] for description of the orientation. This is the default orientation type and it might be missed in the scene file.

XML-markup nodes **<Position>** and **<Rotation>** might be omitted in a scene file. Then no extra position or orientation will be applied.

1.6. Decimal separator and case sensitivity

You should use point "." as a decimal separator in scene XML files as it is used in other Universal Mechanism related files.

Please note that node and attribute names are case sensitive. Use notation as it is given in the sample scenes. For example the correct attribute is **Name**; not **name** or **NAME**. Using incorrect names will cause errors on loading files.

1.7. Creating scenes

Scenes are saved into **.scene* that are XML-files internally. That files have typical XML-markup. We will not consider syntax of XML-format within this manual. There are many of sources when you can find detailed data about XML-files format, for example, via the following link: https://en.wikipedia.org/wiki/XML.

The current UM version does not include a full functional scene editor that could automatize creating UM scenes. It will be published in one of the future releases. So far UM user has to create/modify scene files manually with the help of any text editor.

[1] More detailed information about quaternions you can find, for example, here: https://en.wikipedia.org/wiki/Quaternion

1.7.1. Scene file format

Let us consider *.*scene* files, its fields and attributes. The general structure of the scene file is given in Figure 1.3.

The root **<Scene>** node includes **<EnvironmentSettings>** and **<Objects>** nodes. The **<EnvironmentSettings>** node includes the skybox[2] description (**<SkyBox>** node) and position of orientation of the scene frame of reference in the global UM frame of reference. Changing scene position and/or orientation will move/rotate the whole scene with all added 3D objects.

The **<Objects>** node contains an arbitrary number of **<Object>** nodes that describe standard and user 3D objects with its position and orientation in the scene and its scale.

The typical scene includes a skybox, one or several underlays and many objects from standard or user 3D object collections. As it was already said above a number of sample scenes are distributed along with Universal Mechanism software and available in the *{UM Data}\Scene data\Scenes* folder.

```
<?xml version="1.0" encoding="utf-8"?>
<Scene Description="Sample scene">
<EnvironmentSettings>
  <SkyBox Enabled="true" Material="SkyBox/CloudyNoon"/>
  <SceneCenter>
    <Position X="10" Y="0" Z="0"/>
    <Rotation X="0" Y="0" Z="0" W="1"/>
  </SceneCenter>
</EnvironmentSettings>
<Objects>
  <Object Name="underlay1" ID="1100005">
    <Position X="0" Y="0" Z="-0.1"/>
    <Rotation X="0" Y="0" Z="0" W="1"/>
    <Scale Value="1"/>
    <Dimensions Width="100" Height="100"/>
  </Object>
  <Object Name="suburbanhouse_3" ID="1200004">
    <Position X="50" Y="0" Z="0"/>
    <Rotation Type="rtAngle" Z="90"/>
    <Scale Value="1"/>
  </Object>
  <Object Name="skoda_yeti_1" ID="2100080">
    <Position X="0" Y="0" Z="0"/>
    <Rotation X="0" Y="0" Z="0" W="1"/>
    <Scale Value="1"/>
  </Object>
</Objects>
</Scene>
```

Figure 1.3. General structure of scene file

The **<Object>** node describes every single 3D object in the scene. Let us consider the structure of that node.

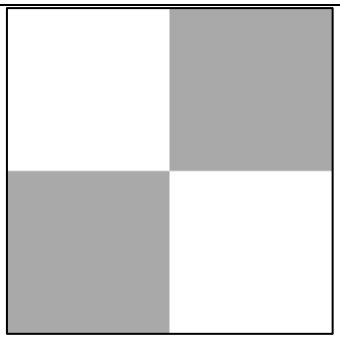
[2] Skybox is distant mountains, distant buildings, and other unreachable objects are projected onto the cube's faces, see https://en.wikipedia.org/wiki/Skybox_(video_games)

- **ID** is the unique identifier of a 3D object from the standard or user collections. It corresponds to the **ID** field from *StandardObjects.xml* or *UserObjects.xml*.
- **Name** is the arbitrary object name. It MUST be unique within the scene file context. It is a strong requirement. Do not forget about that.
- **Position** is the position of the object in the scene's frame of reference given in meters.
- **Rotation** not specifies the orientation of the object in the scene's frame of reference. Please check Sect. 1.5 "*Position and orientation*", page 1-6 for more detailed description of different rotation types.
- **Scale/Value** is the scale factor for the object. Use the scale factor bigger than 1 to make the object bigger, smaller than 1 to make the object smaller than it is in the collection. To make the object bigger in 10 times use the factor 10. To make the object smaller in 10 times you should use factor 1/10.

Note. The **Name** value must be unique within the scene context. If it is not unique, a user will be given error messages.

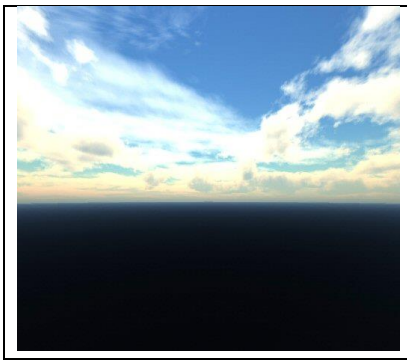
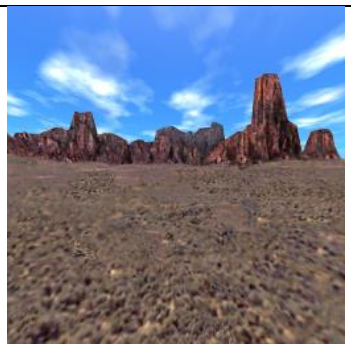
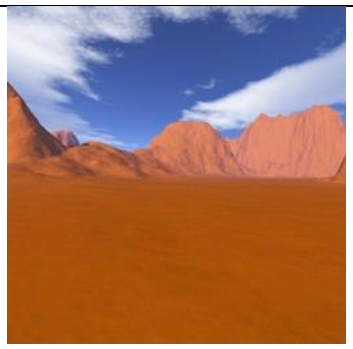
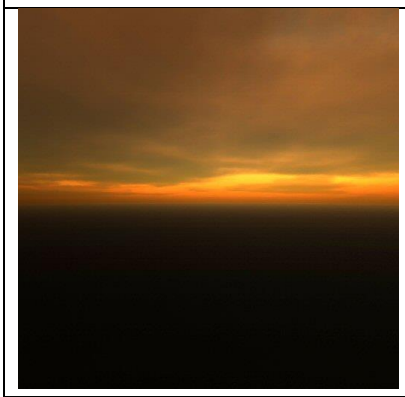
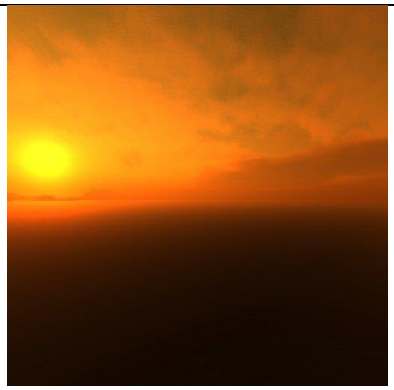
1.7.2. Underlays

The collection of standard object includes a number of underlays given in the table below. Use the ID numbers listed below to select the underlay for your scene.

		
Five kinds of asphalt, ID 1100010-1100014	Six kinds of grass, ID 1100001-1100006	Railway ballast, ID 1100016
		
Three kinds of sand, ID 1100007-1100009	Chess underlay, ID 1100015	

1.7.3. Skyboxes

A few skyboxes are included into the collection of the standard objects.

		
CloudyNoon	Desert1	Desert2
		
EarlyMorning	Evening	IceRiver
		
Yokohama2	Meadow	

Skybox description should be included into the **<EnvironmentSettings>** section in the following way:

```
<SkyBox Enabled="true" Material="SkyBox/CloudyNoon"/>.
```

In the example above you can use any skybox name from the table above instead of the *CloudyNoon*.

Skybox *CloudyNoon* is suitable for most of scenes and is recommended as a default skybox.

1.7.4. Manual creating/editing scene files

Scene editor as a separate tool is not yet available in the present UM release. Scene files should be created and modified via any text editor.

Let us consider creating new scenes using a text editor. Every new object in the scene should be added as one more node under **<Objects>** node in the following way:

```
<Object Name="suburbanhouse_3" ID="1200004">
  <Position X="50" Y="0" Z="0"/>
  <Rotation Type="rtAngle" Z="90"/>
  <Scale Value="1"/>
</Object>
```

Description of fields and format is given in Sect. 1.7.1. "*Scene file format*", page 1-8. Please note that **Name** of every object should be unique within the scene file! You can see object name in the status bar of the animation window when move mouse cursor over the object. **ID** is the unique identifier of the in the standard or user object collections.

How can we determine which **ID** corresponds to the object we would like to place in the scene? To make it easier a number of demo scenes that include all standard objects was created. Demo scenes start with the *<demo>* prefix and available in the *{UM data}\Scene data\Scenes* folder.

Run **UM Simulation**, load any model then open scene wizard and open demo scenes one by one to make yourself familiar with the standard objects.

To determine ID of the object in a demo scene use the following instructions. Open animation window and place mouse onto the object you are interested in. In the status bar of the animation window you can see the object name, see Figure 1.4. Then open that scene in the text editor and find the name of the object there, see Figure 1.5. Finally you will see the ID you need next to the object name:

```
<Object Name="suburbanhouse3_1" ID="1200004">
```

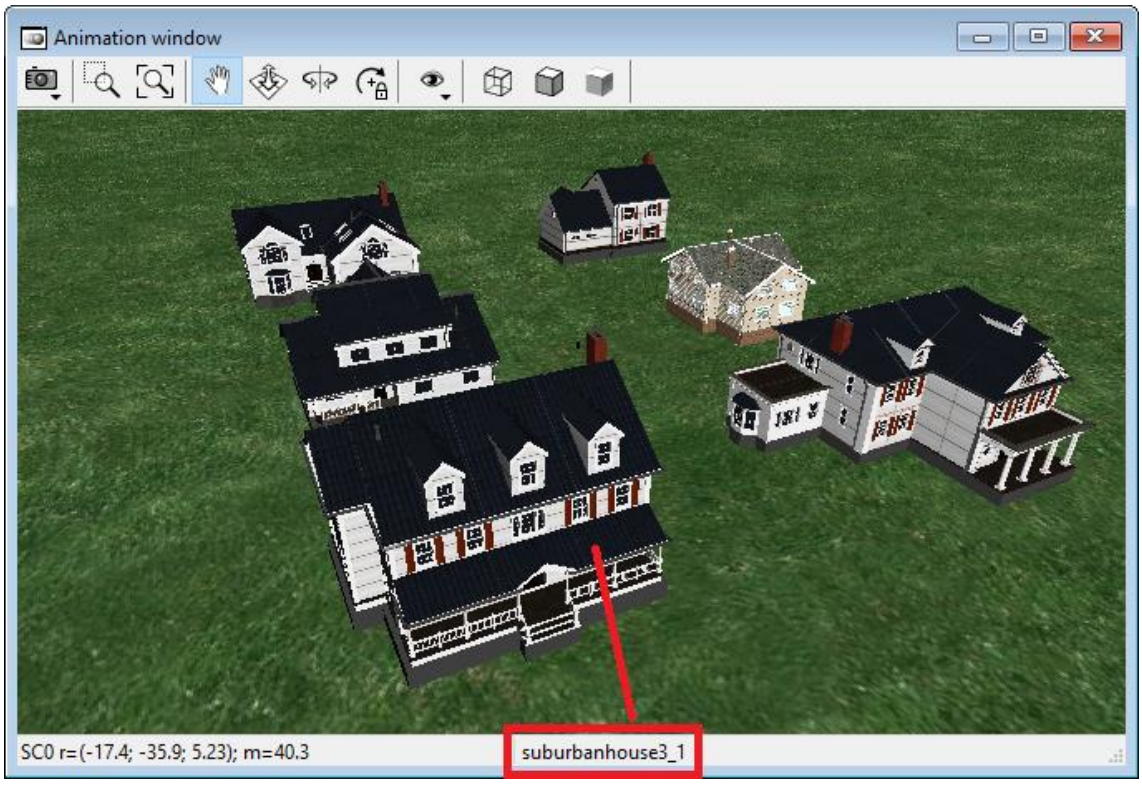


Figure 1.4. Demonstration scene "**demo Suburban house pack.scene**" in the animation window

```
<Object Name="suburbanhouse2_1" ID="1200002">
  <Position X="20" Y="30" Z="0"/>
  <Rotation X="0" Y="0" Z="1" W="0"/>
  <Scale Value="1"/>
</Object>
<Object Name="suburbanhouse3_1" ID="1200003">
  <Position X="-20" Y="-30" Z="0"/>
  <Rotation X="0" Y="0" Z="0" W="1"/>
  <Scale Value="1"/>
</Object>
<Object Name="suburbanhouse4_1" ID="1200004">
  <Position X="-20" Y="30" Z="0"/>
  <Rotation X="0" Y="0" Z="0" W="1"/>
  <Scale Value="1"/>
</Object>
```

Figure 1.5. Demonstration scene "**demo Suburban house pack.scene**" in the text editor

1.8. User collection of 3D objects

Universal Mechanism software uses *Ogre3D* graphical engine starting from v. 8.5, see https://www.ogre3d.org. All objects of the user collection should be prepared in the *Ogre3D* format. Ogre3D, as a quite popular engine with a quite large eco-system, has a number of tools for export of graphical data/objects from popular 3D simulation programs like Autodesk 3ds Max, Blender, Maya, COLLADA, MilkShape 3D, Assimp. List of selected tools is available via the following link: http://wiki.ogre3d.org/OGRE+Exporters.

1.8.1. Exporting object from Autodesk 3ds Max

Note	To be able to export 3D objects from **Autodesk 3ds Max** in *Ogre3D* format you should install Easy Ogre Exporter.

For example let us consider exporting from **Autodesk 3ds Max** and adding to the user 3D object collection a model of **Skoda Yeti**. This model is available in the following folder *{UM Data}\Scene data\Samples\Skoda_Yeti*. Source model is located in the *Autodesk3dsMaxModel* subdirectory. Exported files, created as results of the steps described below, are located in the *ExportedOrge3DModel* subdirectory.

1. Run **Autodesk 3ds Max**. Load the model with the help of **File | Open** menu item for *.max files or use **File | Open | Import | Import** for other file types.

Note	Snapshots given in this manual are taken from **Autodesk 3ds Max 2018**. For other versions of **Autodesk 3ds Max** dialog windows might be different.

Further work will be done mainly using the **Scene Explorer** window, see Figure 1.6. It is usually located in the left part of the main window. If you cannot find it, you can open it using **Scene | Manage Scene Content | Scene Explorer** menu item or press **Alt+Ctrl+O**.

Scene Explorer window shows the objects of the scene. In this case the scene includes 4 objects: **front_left**, **front_right**, **front_types** and **skoda_yeti**, see Figure 1.6. Selecting objects in the **Scene Explorer** leads to highlighting them in the main window.

If you need to export the whole scene as a single object you should unite all the objects in one. This case is considered in Sect. 1.8.1.1. "*Export the whole scene as a single object*", page 1-13. If you want to export every single object separately you should come to Sect. 1.8.1.2. "*Export of objects in Ogre3D format*", page 1-16.

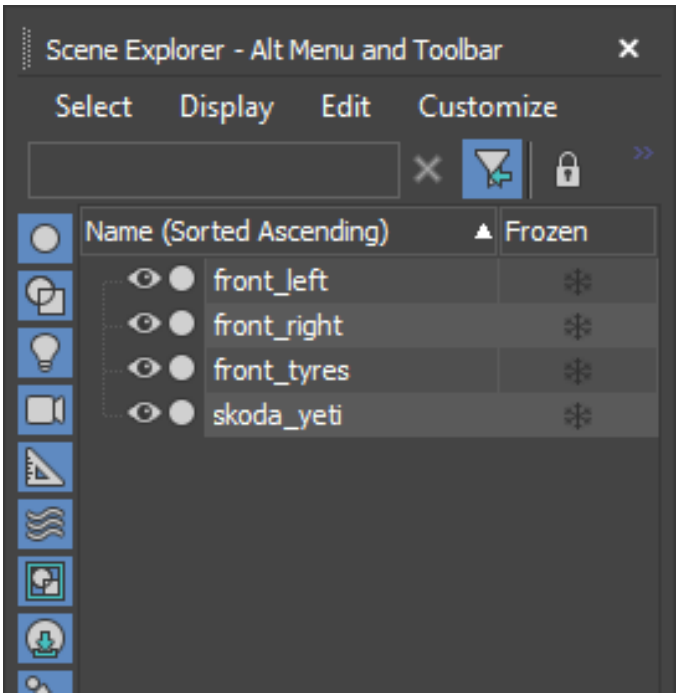


Figure 1.6. Scene Explorer

1.8.1.1. Export the whole scene as a single object

There are a few ways how to unite objects in **3D Studio Max**. One of such ways is described below.

1. In the **Scene Explorer** select the element to which you would like to attach another element. Selected element will be highlighted in the main window, see Figure 1.7.

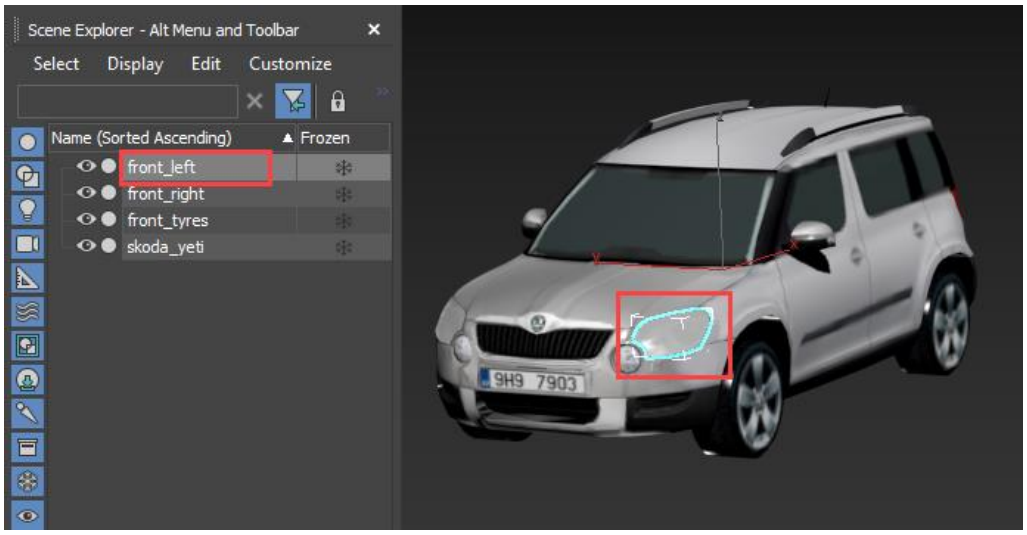


Figure 1.7. Select element

2. After that you should convert it into the editable poly. Click the right mouse button on the selected elements and then in the context menu select **Convert to | Convert to Editable Poly**, see Figure 1.23.

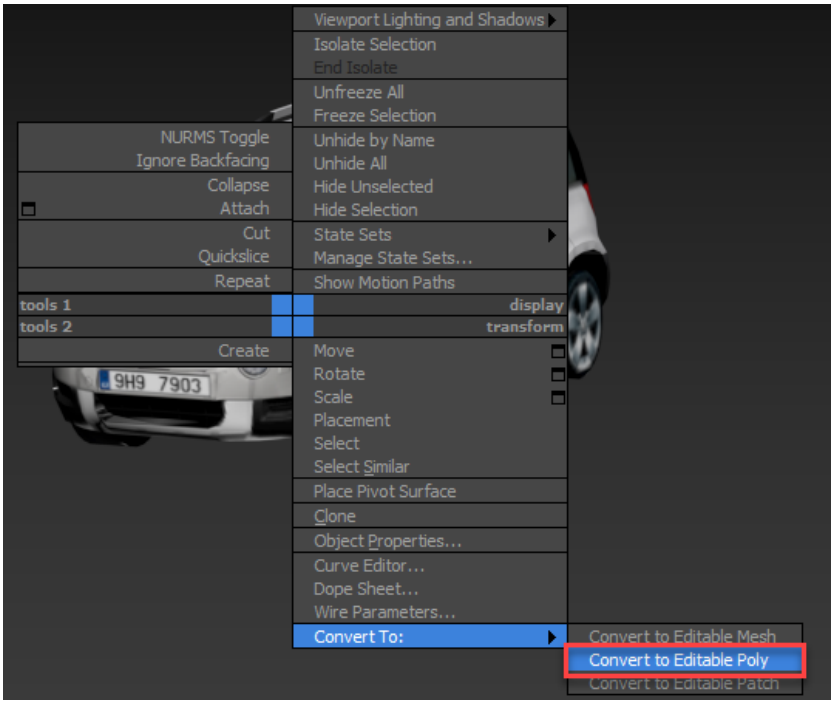


Figure 1.8. Convert to editable poly

3. After that click right mouse button again and select **Attach** in the context menu, see Figure 1.9.

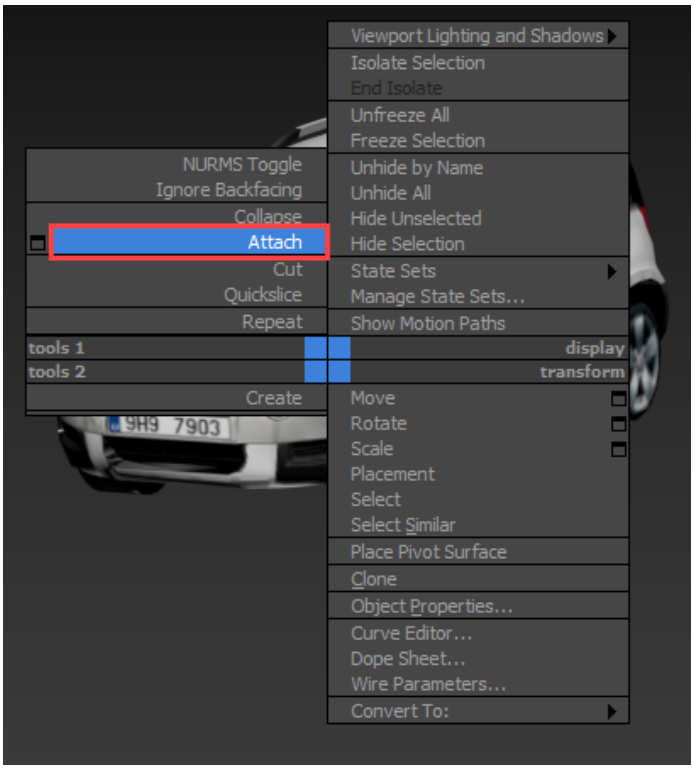


Figure 1.9. **Attach** menu item

4. Then click left mouse button on the object you want to attach in the main window or in the **Scene Explorer**. For our case select, for example, **front_right** object. After that you will see only three objects in the Scene Explorer since **front_right** was attached to the **front_left** object and both headlights will be highlighted in the main window, see Figure 1.10.

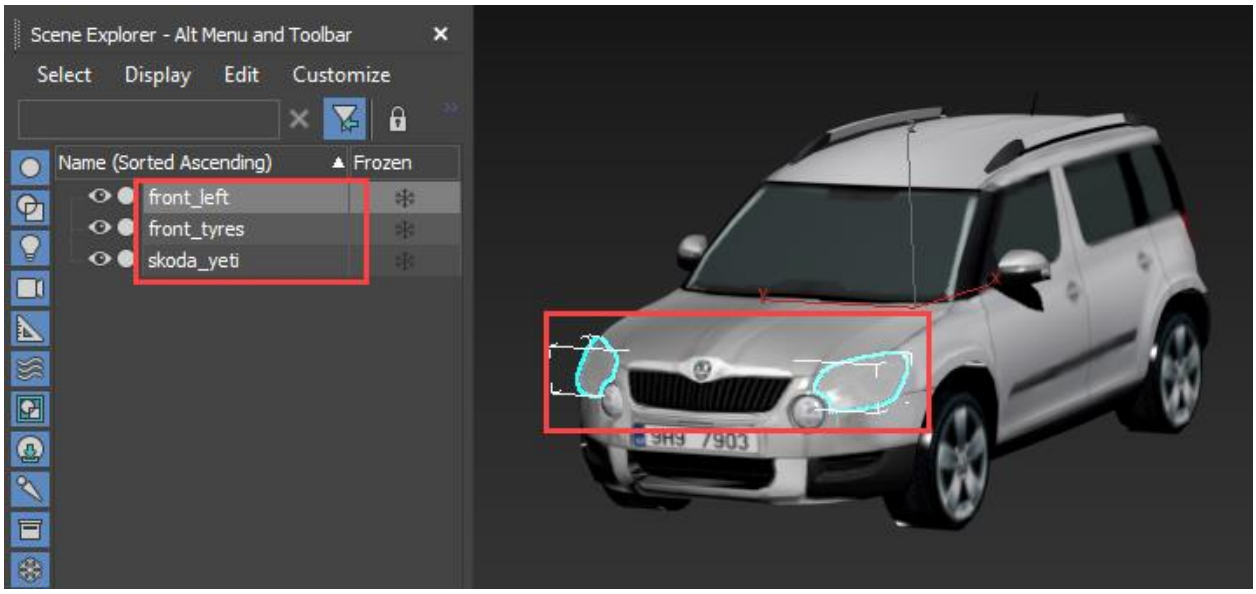


Figure 1.10. United headlights

5. After that select the **front_tyres** element and the **Attach Options** window popped up. It happened because the attached element has the material that differs from the material of the element to which it is being attached. In such a case select the **Match Material IDs to Material** option and turn on the **Condense Material and IDs** check box and click **OK**, see Figure 1.11. Then the element will be automatically attached.

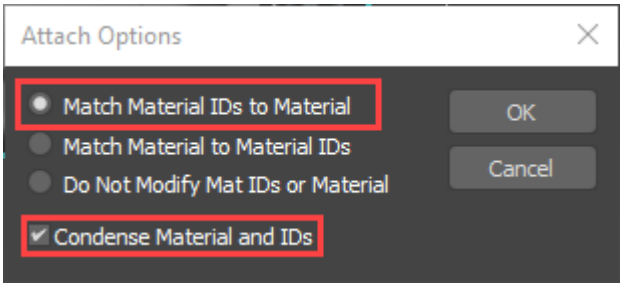


Figure 1.11. **Attach Options** dialog window

6. Finally attach **skoda_yeti** to **front_left** how it is described above in Item 5. After all you will be able to see the only **front_left** element in the **Scene Explorer** window and the whole car will be highlighted in the main window, see Figure 1.12.

Note If you want to join another group of elements simply click in the main window somewhere on the empty space firstly the right mouse button then left one. Then repeat the steps described above.

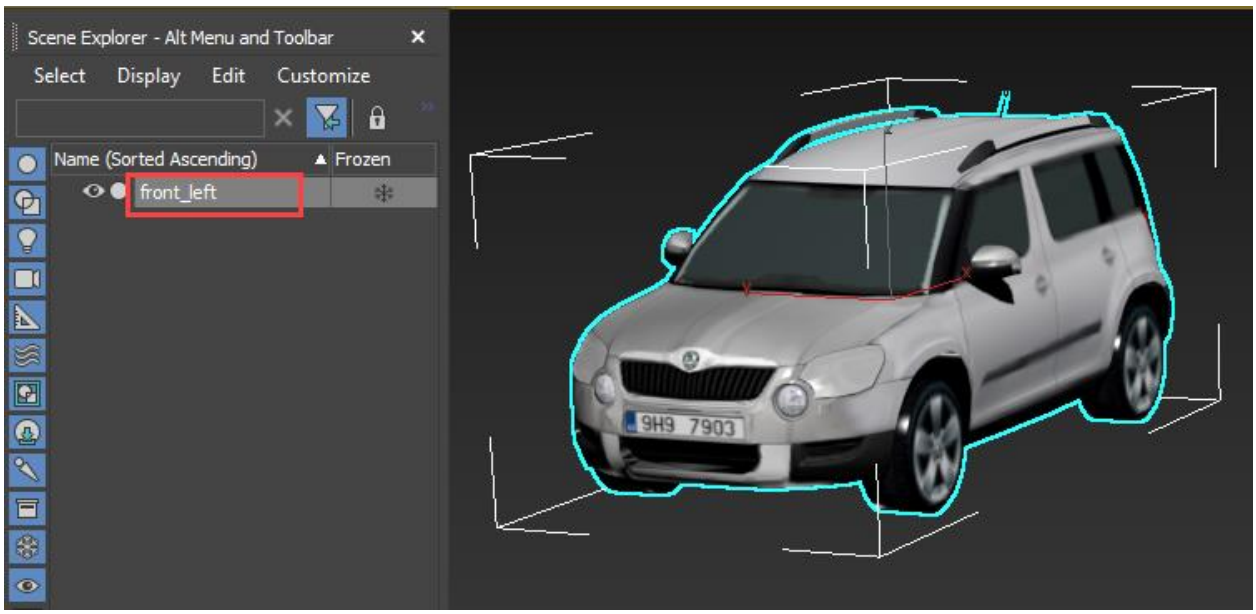


Figure 1.12. The whole car is highlighted

1.8.1.2. Export of objects in Ogre3D format

To export one or several elements in *Ogre3D* format use the following instructions.

If you need to export all scene elements, you should use the **File | Export | Export** menu item, see Figure 1.13. If you need to export just some selected elements, you should select elements you need in the **Scene Explorer** first and then use the **File | Export | Export Selected** menu item.

In the **Select File to Export** create a new directory for the exported element. Folder name should include Latin letters only and no spaces. For our example let us use **Skoda_Yeti** folder, see Figure 1.14. Type **skoda_yeti** for the file name as well, see Figure 1.14.

In the **Save as type** field select the **Ogre Scene (*.SCENE)** and click the **Save button**, see Figure 1.14.

After that the dialog window of **Easy Ogre Exporter** appears, see Figure 1.15. Set **Basic configuration** / **Ogre version** to **Ogre 1.8**. The present version of **UM Scene** supports the **Ogre 1.8** only! You can leave default values in the **Material sub dir**, **Texture sub dir** and **Mesh sub dir** fields as it is shown in Figure 1.15. Materials will be saved as ***.material** file; meshes as *.mesh files. In the **Materials** / **Pixel lighting (CG)** field select **All materials (3 lights)**. All the rest fields set according to Figure 1.15.

If the export procedure finishes successful the **Info** window pops up, see Figure 1.16.

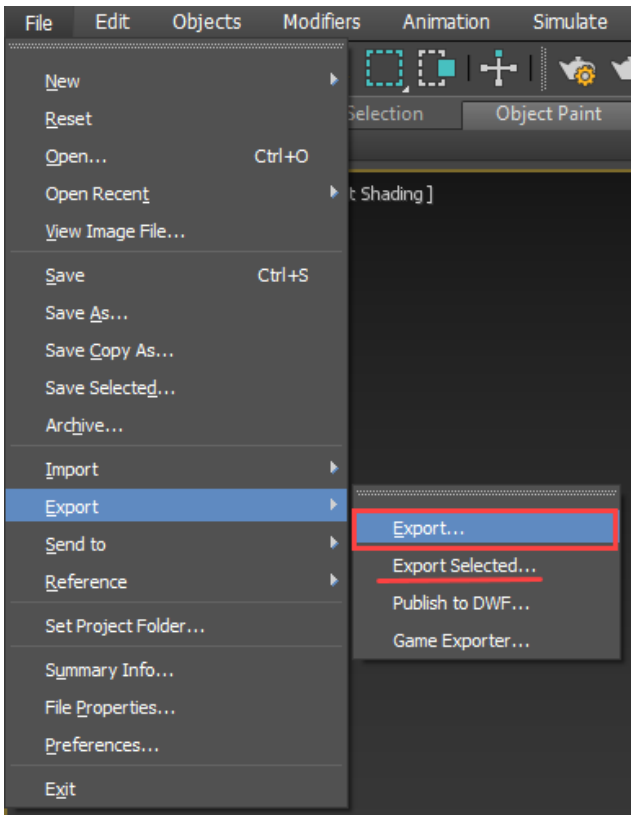


Figure 1.13. Export

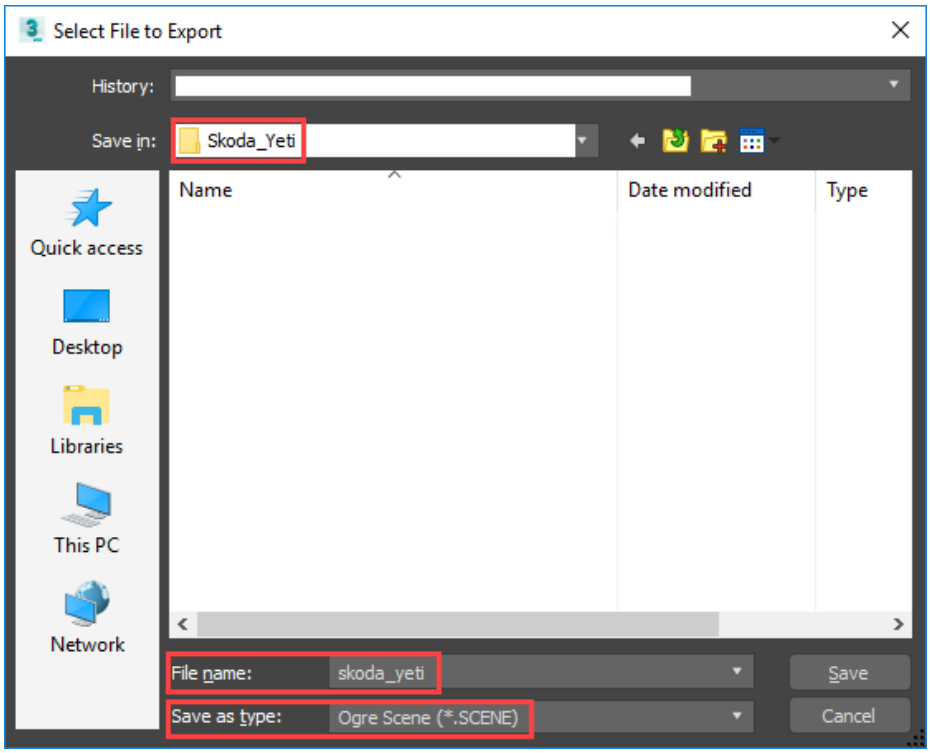


Figure 1.14. Select file to Export

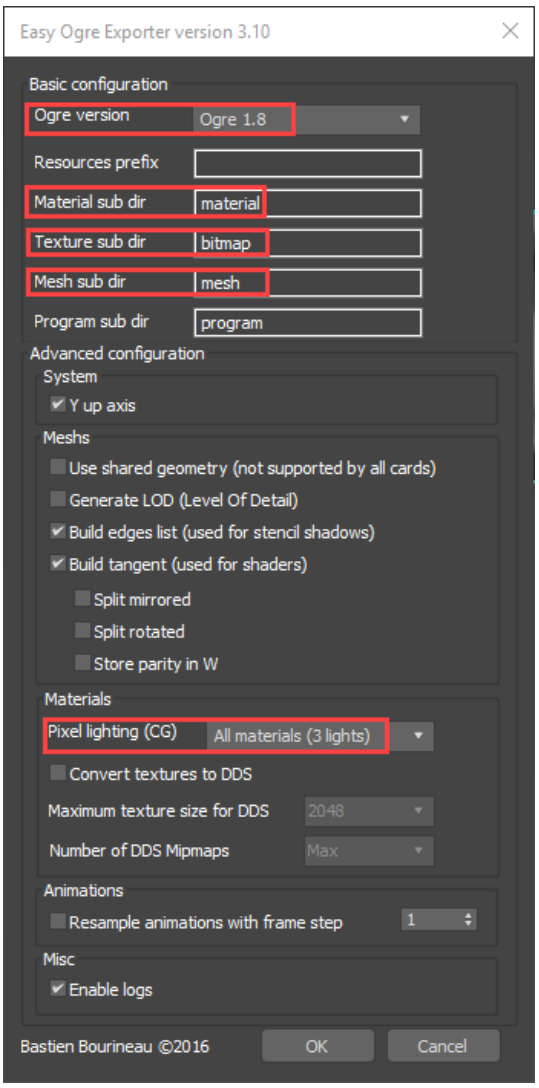


Figure 1.15. **Easy Ogre Exporter** dialog window

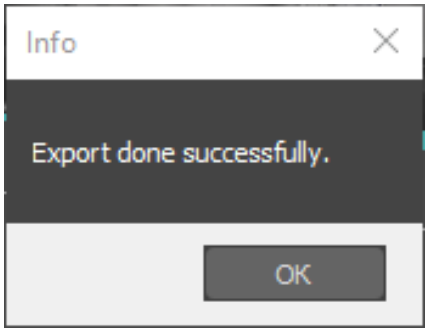


Figure 1.16. **Info** window

1.8.1.3. Typical export problems and ways to fix them

There are a few known problems with the exported files. Let us consider the typical problems and the ways how to fix them below.

- Textures files are not exported sometimes. In such a case you should manually copy them to the target folder. In this example with **Skoda Yeti** the texture files was not exported.

So open the source folder and copy **diffuse_white.jpg** to the **Skoda_Yeti** folder, see Sect. 1.8.2.1. "*Creating ZIP-archive*", page 1-20.

- In the most cases you have to modify ***.material** files. The current version of **UM Scene** does not support some material parameters correctly. So you have to modify description of **ambient**, **specular** and **emissive** colors so as to set the first three numbers to zero, see Figure 1.17.

```
material "Material__32"
{
    technique Material__32_technique
    {
        pass Material__32_standard
        {
            ambient 1 1 1 1
            diffuse 1 1 1 1
            specular 0 0 0 0 25.5
            emissive 0 0 0 1
            vertex_program_ref vsLightGEN0
            {
            }
            fragment_program_ref fpLightGENDIFF0
            {
            }

            texture_unit Material__32_Diffuse#0
            {
                texture diffuse_white.jpg
                tex_coord_set 0
                cclour_op modulate
            }
        }
    }
```

```
material "Material__32"
{
    technique Material__32_technique
    {
        pass Material__32_standard
        {
            ambient 0 0 0 1
            diffuse 1 1 1 1
            specular 0 0 0 1 25.5
            emissive 0 0 0 1
            vertex_program_ref vsLightGEN0
            {
            }
            fragment_program_ref fpLightGENDIFF0
            {
            }

            texture_unit Material__32_Diffuse#0
            {
                texture diffuse_white.jpg
                tex_coord_set 0
                colour_op modulate
            }
        }
    }
```

Figure 1.17. Material properties

- Another typical problem is incorrect reference to the texture file in the ***.material** file, see 'texture diffuse_white.jpg' line in Figure 1.17.
- If you want to make your object transparent, you should add two more lines shown in Figure 1.18 to the material properties and set the forth number for the diffuse color in the range [0..1] where 0 corresponds to the fully transparent material, 1 corresponds to the fully non-transparent material.

The following text should be added to the material properties to make it transparent:

scene_blend alpha_blend
depth_write off

```
material "Material__31_20000001"
{
    technique Material__31_technique
    {
        pass Material__31_standard
        {
            ambient 0 0 0 1
            diffuse 0.533333 0.533333 0.533333 0.3
            specular 0 0 0 1 0
            emissive 0 0 0 1
            scene_blend alpha_blend
            depth_write off
        }
```

Figure 1.18. Extra properties for a transparent material

- One more correction is required for the *.material file. In the **texture** field there must be the full path to the texture starting from the root of the ZIP-archive expressed via a forward slash '/'. As for the considered example for the material named **Material__32_20000001** we should add '**Skoda_Yeti/**' prior to the texture file name, Figure 1.19.

```
material "Material__32_20000001"
{
    technique Material__32_technique
    {
        pass Material__32_standard
        {
            ambient 0 0 0 1
            diffuse 1 1 1 1
            specular 0 0 0 1 25.5
            emissive 0 0 0 1

            texture_unit Material__32_Diffuse#0
            {
                texture Skoda_Yeti/diffuse_white_20000001.jpg
                tex_coord_set 0
                colour_op modulate
            }
        }
    }
}
```

Figure 1.19. Path to the texture file

Note	Names of materials mentioned in *.material files should be globally unique taking into account both standard and user collections. In other words every material name should be used only once in both standard and user collections. In this example the material name **Material__32_20000001** should not be used somewhere else. Also *.material files should have different names.

1.8.2. Adding objects to user collection of 3D objects

To add you newly exported object(s) to the user collection of 3D objects you should create two files: **XML-file** with the object description and **ZIP-archive** with the objects themselves. Let us consider how to prepare every mentioned file below.

Note	Preliminarily prepared ZIP- and XML-files for the user 3D object collection are stored in the [*{UM Data}\Scene data\Samples\User objects*] folder for your convenience.

1.8.2.1. Creating ZIP-archive

Before coming to the nest step make sure that you fixed all typical problems described in Sect. 1.8.1.3. "*Typical export problems and ways to* fix them", page 1-18. After that go to the instructions given below.

- Create a new directory, let it be **SampleUserBase** for our case.
- Then create one more subdirectory for the exported Skoda Yeti model, **SampleUserBase\Skoda_Yeti_Sample**. Please note that the names of both directories must be in Latin letters without spaces.
- Copy exported ***.mesh**, ***.material** and texture files, if any, to this newly created folder **SampleUserBase\Skoda_Yeti_Sample**, see Figure 1.20.
- For a better readability rename **front_left.mesh** as **skoda_yeti.mesh**.
- Then make a ZIP-archive **SampleUserBase.zip** with the **SampleUserBase** folder.
- Finally copy the archive to the standard folder for the user 3D object collection *{UM Data}\Scene data\User object*, see Figure 1.21.

Note	Name of archive must be written in Latin letters without spaces. The file extension and file format must be **.ZIP**. It is recommended to use the highest compression level.

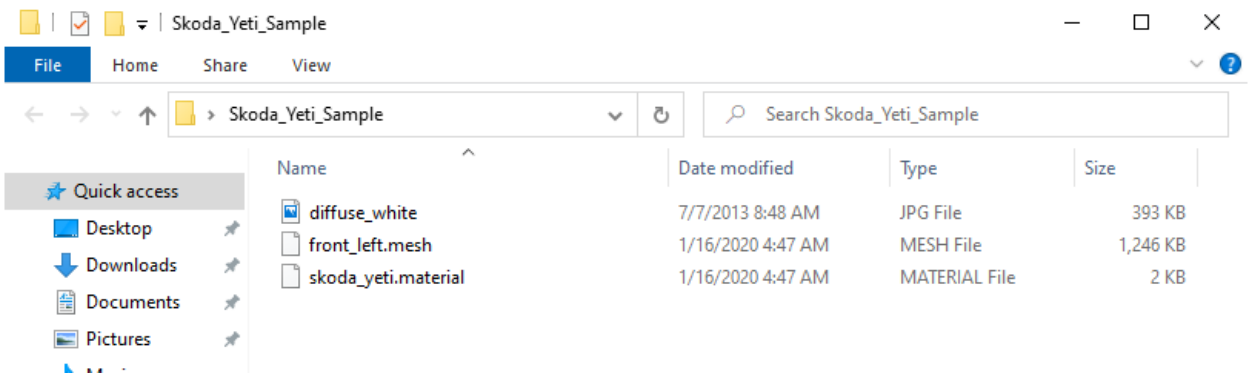


Figure 1.20. Creating the folder for a 3D object

Limitations	Every ***.ZIP** archive MUST not exceed 2 Gb. If your objects all together exceed 2 Gb you had better to divide it into a few ***.ZIP** archives of smaller size. Please also note that every single non-compressed ***.MESH** file MUST not exceed 2 Gb either.

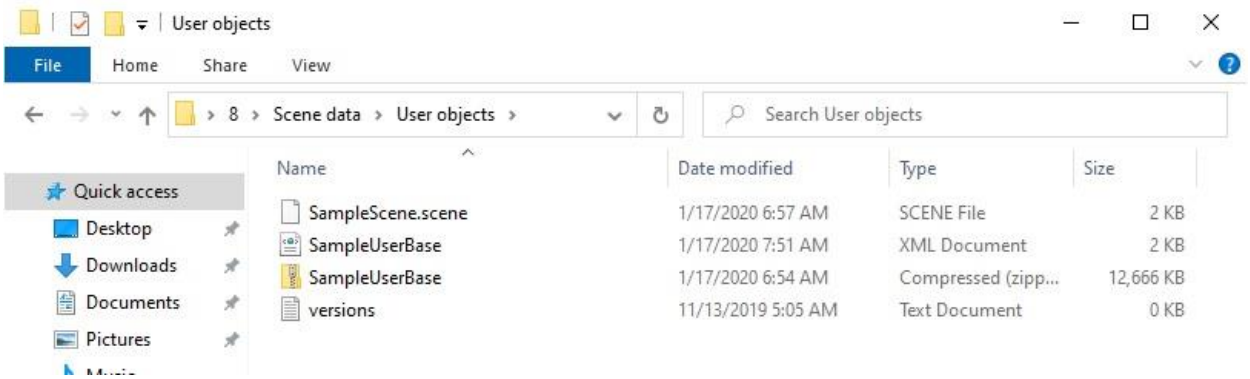


Figure 1.21. *{UM Data}\Scene data\User object* folder

1.8.2.2. Creating XML-file

After creating a *.ZIP file you should create a correspondent *.XML file. Such *.XML file contains information about 3D objects included into the *.ZIP file. Let it be **SampleUserBase.xml** for this test case. The general structure of the *.XML files is described below, see Figure 1.22.

- The root node of the XML-file is <**Base**> node. The only attribute of the <**Base**> node is **FileName**. This attribute should be equal to the correspondent *.ZIP file that is being described by this XML-scheme. In our case it is **"SampleUserBase.zip"**.
- Objects in the archive must be included into one or several groups, for example, *"Cottages"*, *"Tree"* etc. The <**Group**> node is used for that. **ID** attribute of the <**Group**> node gives is the arbitrary integer number that is used for grouping the group items in the scene editor that will be available later. ID range from 0 to 19999999 is reserved for the standard 3D collection. ID of the user 3D object should be 20000000 or higher. The **Name** attribute is any arbitrary group name that is used just for user-friendly readability.

Every <**Object**> node includes description of the *.MESH object and material with its initial position and orientation and scale. Let us consider the structure of the <**Object**> node.

- **Name** attribute MUST be unique within the range of standard and user 3D objects.
- **Description** attribute is a brief text description of the object.
- **ID** attribute must be unique within the range of standard and user 3D objects integer number. It is recommended to increment **ID** for every next object. It is recommended that for simplicity the **ID** of the first object should be group **ID+1**.
- **Type** attribute describes the type of the object. In the most cases it is **otMesh**. The **otUnderlay** type should be used if the object describes the user's underlay. In this case **smtMaterial** type of the material should be used, see <**Material/Type**> attribute below.
- **MeshFileName** attribute has the path to the *.mesh file. It is required that the full path to the MESH-file should be mentioned started right from the root of the ZIP-file. While specifying path one should use the forward slash '/' as it is shown in Figure 1.22.
- **Position** and **Rotation** nodes set the additional constant position and orientation that are applied for the exported *.MESH file. See Sect. 1.5 "*Position and orientation*", page 1-6.

The <**Material**> node has the **Type** attribute that specifies the way how the material is applied for the object. There are three possible ways for that, see Figure 1.22:

- **smtSubMaterial** is used when the object consists of a few sub-objects with different materials. Description of that materials are in ***.material** file. Every material should be described in the <**SubMaterial**> node. **Index** of the **SubMaterial** sets the index of the corresponding sub-object. The first sub-object has the index **0**.
- **smtMaterial** is used when the only material is used for the object. In the **Name** attribute is the name of the corresponding ***.material** file.
- **smtBitmap** is used when the simple bitmap texture or texture map is used with no correspondent ***.material** file. In this case **TextureFileName** has the full texture file name.

```
<?xml version="1.0" encoding="utf-8"?>
<Base FileName="SampleUserBase.zip">
  <Group ID="2000000" Name="User Group">
    <Object Name="skoda_yeti" Description="Skoda Yeti" ID="2000001" Type="otMesh">
      <MeshFileName Value="Skoda_Yeti/skoda_yeti_2000001.mesh"/>
      <Position X="0" Y="0" Z="0.83"/>
      <Rotation X="0" Y="0.707" Z="0.707" W="0"/>
      <Scale Value="0.027"/>
        <Material Type="smtSubMaterial">
          <SubMaterial Name="Material__32_2000001" Index="0"/>
          <SubMaterial Name="Material__31_2000001" Index="1"/>
        </Material>
    </Object>
    <Object Name="Maple" Description="Maple" ID="2000002" Type="otMesh">
       <MeshFileName Value="Maple/maple_2000002.mesh"/>
       <Position X="0" Y="0" Z="0"/>
       <Rotation X="0.707106781186547" Y="0" Z="0" W="0.707106781186548"/>
       <Scale Value="0.4"/>
       <Material Type="smtMaterial" Name="maple_1_2000002"/>
    </Object>
    <Object Name="Bus_Stop_Sign" Description="Bus Stop Sign" ID="2000003" Type="otMesh">
       <MeshFileName Value="Bus_Stop_Sign/Bus_Stop_Sign_2000003.mesh"/>
       <Position X="0" Y="-0.215" Z="2"/>
       <Rotation X="1" Y="0" Z="0" W="0"/>
       <Scale Value="17"/>
       <Material Type="smtBitmap" TextureFileName="Bus_Stop_Sign/Bus_Stop_Sign_2000003.png"/>
    </Object>
    <Object Name="Grass" Description="Grass" ID="2000004" Type="otUnderlay">
      <Material Type="smtMaterial" Name="Grass_2000004"/>
    </Object>
  </Group>
</Base>
```

Figure 1.22. Structure of the XML-file for user objects

1.8.2.3. Connection between ZIP-archives and XML-files

Let us consider a few comments regarding the general structure of the user 3D object collection and the order of its loading. On starting UM Simulation scans all XML-files in the *{UM Data}\Scene data\User object* folder and loads into the *Ogre3D* environment all models from ZIP-archives mentioned in the **<Base/Filename>** attributes, see Figure 1.22. A XML-file should include the only **<Base>** node. So as every XML-file describes the only ZIP-archive. XML-file names may differ from the correspondent ZIP-file names. However for readability and easier maintenance it is recommended to keep the same names for both XML- and ZIP-files.

Please also note that the only ZIP-archives mentioned in the XML-files are loaded into the *Ogre3D* environment. If there is a ZIP-file that does not any reference to it from the XML-file, such a ZIP-file will not be loaded and 3D objects from it will not be available for user scenes.

1.9. Loading scenes into UM Simulation

Loading scenes is available in the **UM Simulation** only. You can upload any scene or create a new one for any model.

As it was described above, the reference to the last active scene is saved in the *.icf configuration file in the "*with environment*" structure, field "*SceneFilename*". On the model loading the saved scene is looked for using the direct path specified in the *.icf file, then in the model's folder and finally in the global scene folder *{UM data}\Scene data\Scenes*. Take this into account when you copy your models to other computers.

Scene wizard, see Figure 1.23, is available in **UM Simulation** program via the **Tools | Scenes** menu command. To load a scene simply specify a file in the **Active scene** field or select the scene you need in the list of scenes and click **Make the scene active** of the popup menu item. Use **Deactivate scene** button to clear all scene-related objects in animation windows. If you modify the active scene via text editor you can simply click **Reload scene** button to refresh the scene.

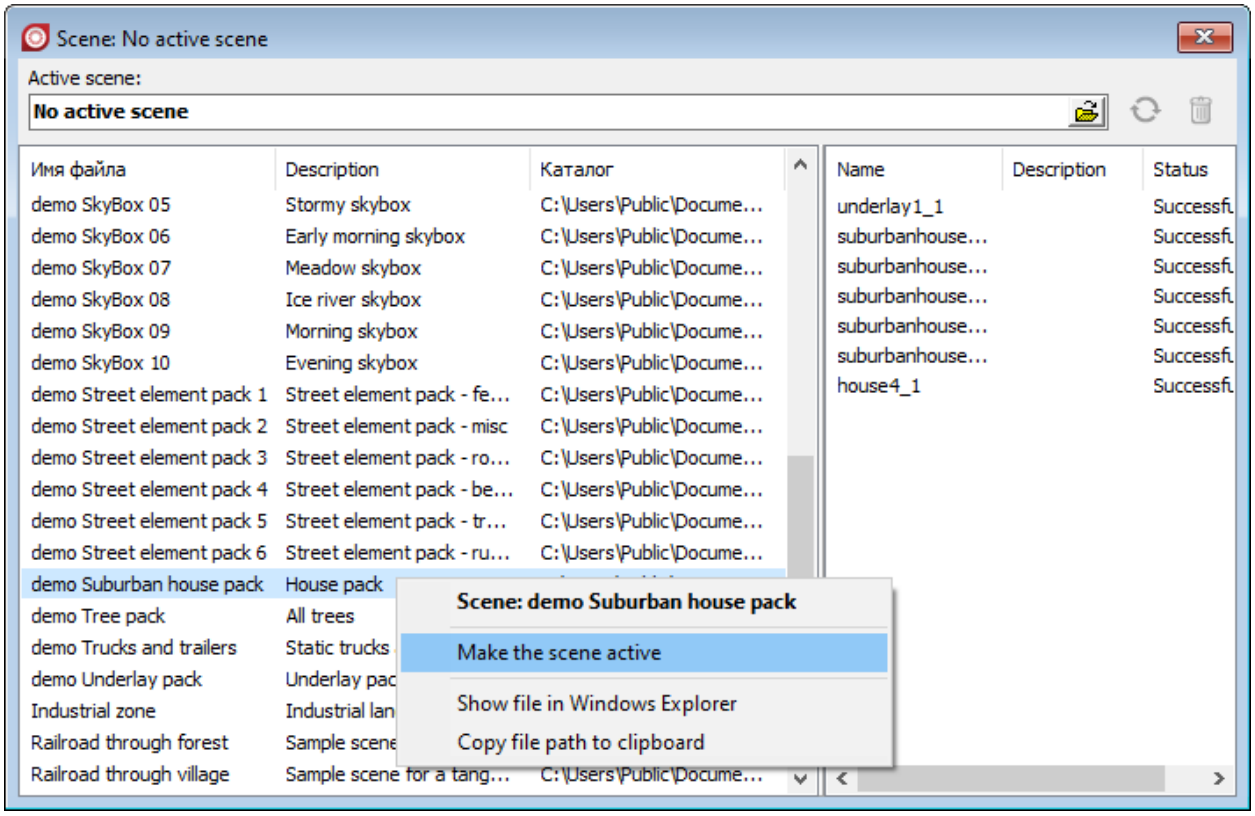


Figure 1.23. Scene wizard