XML formats

CoppeliaSim supports two different XML formats, where each follow a different goal:

- an **exhaustive format**: the exhaustive format is a lossless format, which means that all scene or model information will be same between a save and load operation. The drawback is however that the file should not be manually modified, since the risk of corrupting it is high. This format is ideal for version control for example.
- a **simple format**: the simple format is a lossy format, which means that not all information contained in a scene or model will be saved, only a subset of it. The advantage is however that it can be manually created and/or modified. This format is ideal for creating initial models, or for creating CoppeliaSim exporters in different applications.

Both formats have two variables (defined in file *system/usrset.txt*) allowing to control the output XML file:

- ***xmlExportSplitSize***: represents the subset data size threshold where data will be referenced and written to a separate file. Set to 0 if you wish to create a single file.
- ***xmlExportKnownFormats***: if *true*, and the export generates several files, then the *png* format for images, and the *dae* format for meshs are used. The *dae* format is however only supported with the simple xml format.

The exhaustive format will not be discussed further here. Refer to the generated files for additional information.

Simple XML format

The simple format, however its name, can contain quite complex information. Each piece of information is wrapped as *element* inside of a start- and end-*tag*. Most tags are optional (if not specified, then default values will be used). The simplest to get a good understanding of such a file's organization and which tags are not optional is to first export a scene, then examine/modify the generated file.

Following gives a brief overview, based on simple examples. First, a scene containing a camera attached to a dummy. Required items are indicated in red:

```
<CoppeliaSim>
  <filetype>simpleScene</filetype> (Should be simpleScene or simpleModel)
  <xmlSerializationNb>1</xmlSerializationNb>
  <environment> (ignored when loading a model)
    ... (refer to generated files for details)
  </environment>
  <settings> (ignored when loading a model)
    ... (refer to generated files for details)
  </settings>
  <dynamics> (ignored when loading a model)
    ... (refer to generated files for details)
  </dynamics>
  <simulation> (ignored when loading a model)
    ... (refer to generated files for details)
  </simulation>
  <dummy>
    ... (refer to generated files for details)
    <camera>
      ... (refer to generated files for details)
    </camera>
  </dummy>
</CoppeliaSim>
```

It is a good idea to specify a few more tags for a scene object, such as the object alias, its position/orientation relative to its parent object:

```
...
<camera>
  <common>
    <alias>DefaultCamera</alias>
    <localFrame>
      <position>1.120530 -1.899800 1.079718</position>
      <euler>-110.932999 -28.703005 169.590027</euler>
    </localFrame>
    ... (refer to generated files for details)
  </common>
  ... (refer to generated files for details)
</camera>
...
```

Shapes require a bit more explanations. Here a **primitive cuboid** shape:

```
<shape>
  <common>
    ... (refer to generated files for details)
  </common>
  <primitive> (should be compound, primitive, heightfield or mesh)
    <type>cuboid</type> (should be cuboid, sphere, cylinder, cone, plane or disc)
    <size>0.100000 0.100000 0.100000</size>
    <localFrame> (this is in addition to the local frame you may specify in <common>)
      <position>0.000000 0.000000 0.000000</position>
      <euler>-0.000000 0.000000 -0.000000</euler>
    </localFrame>
    ... (refer to generated files for details)
  </primitive>
  <dynamics>
    ... (see further below for details)
  </dynamics>
</shape>
```

Here a **mesh**, referencing an external file:

```
<shape>
  <common>
    ... (refer to generated files for details)
  </common>
  <mesh> (should be compound, primitive, heightfield or mesh)
    <fileName>test.simscene_mesh_Cuboid0.dae</fileName> (or vertices and indices)
```

```
    <localFrame> (this is in addition to the local frame you may specify in <common>)
      <position>0.000000 0.000000 0.000000</position>
      <euler>-0.000000 0.000000 -0.000000</euler>
    </localFrame>
    ... (refer to generated files for details)
  </mesh>
  <dynamics>
    ... (see further below for details)
  </dynamics>
</shape>
```

Here a **mesh**, with inlined mesh data:

```
<shape>
  <common>
    ... (refer to generated files for details)
  </common>
  <mesh> (should be compound, primitive, heightfield or mesh)
    <vertices>0.05 -0.05 -0.05 -0.05 -0.05 -0.05 0.05 0.05 -0.05 -0.05 ...</vertices>
    <indices>0 1 2 2 1 3 1 0 4 1 4 5 2 3 6 6 3 7 6 7 4 4 7 5 3 1 7 7 1 5 0 2 6 0 6 4</indices>
    <localFrame> (this is in addition to the local frame you may specify in <common>)
      <position>0.000000 0.000000 0.000000</position>
      <euler>-0.000000 0.000000 -0.000000</euler>
    </localFrame>
    ... (refer to generated files for details)
  </mesh>
  <dynamics>
    ... (see further below for details)
  </dynamics>
</shape>
```

Here a **heightfield**:

```
<shape>
  <common>
    ... (refer to generated files for details)
  </common>
  <heightfield> (should be compound, primitive, heightfield or mesh)
    <size>4 3</size>
    <data>0.2 0.2 0.2 0.2 0.1 0.0 0.0 0.1 0.1 0.1 0.0 0.2</data>
    <gridStep>3.333333</gridStep>
    <localFrame> (this is in addition to the local frame you may specify in <common>)
      <position>0.000000 0.000000 0.000000</position>
      <euler>-0.000000 0.000000 -0.000000</euler>
    </localFrame>
    ... (refer to generated files for details)
  </heightfield>
  <dynamics>
    ... (see further below for details)
  </dynamics>
</shape>
```

And here a **compound**:

```
<shape>
  <common>
    ... (refer to generated files for details)
  </common>
  <compound> (should contain at least 2 of any of compound, primitive or mesh)
    <primitive>
      ...
    </primitive>
    <primitive>
      ...
    </primitive>
    <compound>
      ...
    </compound>
    ...
  </compound>
  <dynamics>
    ... (see further below for details)
  </dynamics>
</shape>
```

If a shape has dynamic properties, you should also fill-in the dynamics tags, as in following example:

```
<shape>
  ...
  <dynamics>
    <respondableMask>65535</respondableMask>
    <mass>1.0</mass>
    <localInertiaFrame>
      <position>0.0 0.0 0.0</position>
      <euler>0.0 0.0 0.0</euler>
    </localInertiaFrame>
    <principalMomentOfInertia>0.001667 0.001667 0.001667</principalMomentOfInertia>
    <switches>
      <static>false</static>
      <respondable>true</respondable>
      ... (refer to generated files for details)
    </switches>
    <material>
      <engines>
        <bullet>
          <friction>0.5</friction>
          <oldfriction>1.0</oldfriction>
          ... (refer to generated files for details)
        </bullet>
        <ode>
          <friction>1.0</friction>
            ... (refer to generated files for details)
        </ode>
        <vortex>
          <primlinearaxisfriction>1.0</primlinearaxisfriction>
          <seclinearaxisfriction>1.0</seclinearaxisfriction>
          ... (refer to generated files for details)
        </vortex>
```

```
        <newton>
          <staticfriction>1.0</staticfriction>
          <kineticfriction>1.0</kineticfriction>
          ... (refer to generated files for details)
        </newton>
      </engines>
    </material>
  </dynamics>
</shape>
```

Here a revolute joint, in kinematic mode:

```
<joint>
  <common>
    ... (refer to generated files for details)
  </common>
  <type>revolute</type> (can be revolute, prismatic or spherical)
  <mode>kinematic</mode> (can be kinematic, dependent or dynamic)
  <minPosition>-180.0</minPosition>
  <range>360.0</range>
  <position>0.0</position>
  <switches>
    <cyclic>true</cyclic>
  </switches>
  ... (refer to generated files for details)
</joint>
```

Here a revolute joint, in dynamic mode, motor disabled:

```
<joint>
  <common>
    ... (refer to generated files for details)
  </common>
  <type>revolute</type> (can be revolute, prismatic or spherical)
  <mode>dynamic</mode> (can be kinematic, dependent or dynamic)
  <minPosition>-180.0</minPosition>
  <range>360.0</range>
  <position>0.0</position>
  <switches>
    <cyclic>true</cyclic>
  </switches>
  <dynamics>
    <maxForce>2.5</maxForce>
    <ctrlMode>4</ctrlMode>
    ... (refer to generated files for details)
    <engines>
      ... (refer to generated files for details)
    </engines>
  </dynamics>
</joint>
```

Here a revolute joint, in dynamic mode, position controlled:

```
<joint>
  <common>
    ... (refer to generated files for details)
  </common>
  <type>revolute</type> (can be revolute, prismatic or spherical)
  <mode>dynamic</mode> (can be kinematic, dependent or dynamic)
  <minPosition>-180.0</minPosition>
  <range>360.0</range>
  <position>0.0</position>
  <switches>
    <cyclic>true</cyclic>
  </switches>
  <dynamics>
    <maxForce>2.5</maxForce>
    <ctrlMode>4</ctrlMode>
    <upperVelocityLimit>180</upperVelocityLimit>
    <targetPosition>0.0</targetPosition>
    ... (refer to generated files for details)
    <engines>
      ... (refer to generated files for details)
    </engines>
  </dynamics>
</joint>
```

Other scene objects are not discussed here. For details, export a scene containing such object, then examine the generated file.