Import/export

Several type of data and formats can be imported to, or exported from CoppeliaSim:

- mesh data, via various formats
- scene/model data, via CoppeliaSim's XML format
- scene/model data, via the URDF format
- scene/model data, via the SDF format
- animation or scene data, via the GLTF format
- video data, via various formats
- image data, via various formats
- text/binary data
- floor plan data
- LDR / MPD models

New importers/exports can easily be created via an add-on, or via a plugin.

Importing/exporting mesh data

The mesh import/export functionality is handled via the Assimp plugin for CoppeliaSim. More formats can be supported if the Assimp library (and Assimp plugin) are recompiled with the required flags. The plugin's functionality is exposed to CoppeliaSim's GUI and can be reached at [File > Import > Meshes...] or [File > Export > Selected shapes...].

If after the import operation you can't see any shapes in the scene, but the scene hierarchy indicates the presence of newly added shapes, then most probably your shapes are either too big or too small to be seen. You can then proceed to a scaling operation. Additionally, you can subdivide imported meshes via [Edit > Shape grouping/merging > divide].

Make sure that imported meshes do not contain too many triangles (for a robot, typically between 10'000-20'000 triangles in total), otherwise CoppeliaSim could be slowed down. You can decimate an imported mesh via [Modules > Geometry / Mesh > Mesh decimation...]

Heightfields in CoppeliaSim are also meshes, and can be imported via [File > Import > Heightfield...]. Supported formats are image formats (the color or grey tone components represent elevations), or csv or txt formats (comma-separated values (y rows with x values)).

See also the API functions related to mesh import/export.

Importing/exporting URDF files

One can access the URDF importer/exporter functionality via the following two add-ons: [Modules > Importers > URDF importer...] and [Modules > Exporters > URDF exporter...]

See also the API functions related to URDF import/export.

Importing SDF files

One can access the SDF importer functionality via the following add-on: [Modules > Importers > SDF importer...]

See also the API functions related to SDF import.

Exporting a simulation as GLTF

A scene or simulation can be exported via the GLTF format. The result will be a still scene or an animated scene. The functionality is available via two distinct add-ons: [Modules > Exporters > GLTF exporter...] and [Modules > Exporters > GLTF animation exporter...].

See also the API functions related to GLTF export.

Importing/exporting images

Images can be imported/exported with sim.loadImage and sim.saveImage.

See also sim.getScaledImage, sim.transformImage, sim.transformBuffer and the OpenCV plugin API reference.

Importing/exporting text/binary data

Following example illustrates how to log a joint angle to a file, as text data:

Python Lua

```
#python

import math

def sysCall_init():
    sim = require('sim')
    self.jointHandle = sim.getObject('/Joint')
    with open('jointAngles.txt', 'w+') as file:
        file.write('Joint angles for each simulation step:\n\n')
```

```python
def sysCall_sensing():
    v = 180 * sim.getJointPosition(self.jointHandle) / math.pi
    with open('jointAngles.txt', 'a') as file:
        file.write(f'time: {sim.getSimulationTime() + sim.getSimulationTimeStep():.3f} [s]')
        file.write(f', joint angle: {v:.1f} [deg]\n')
```

Following example illustrates how to read a file, line by line:

Python Lua

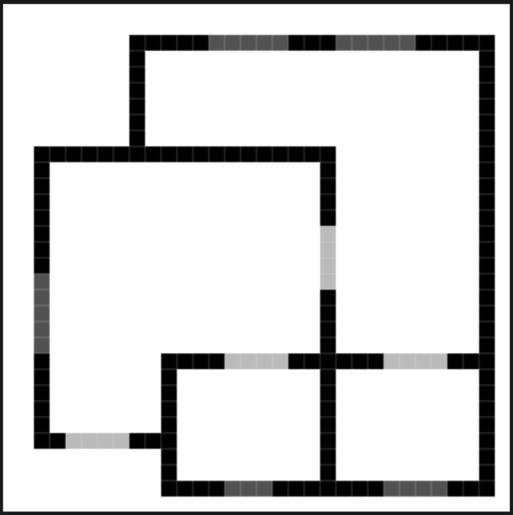
```python
#python

with open('textFile.txt', 'r') as file:
    for line in file:
        print(line.strip())
```

Importing miscellaneous data

The floor plan importer add-on, available via [Modules > Importers > Floor plan importer...], can create shapes from a floor plan image, by mapping each pixel value to either a *wall*, a *door* or a *window*.

A floor-plan image looks like this:



[Example of a floor-plan image]

Note that above image is zoomed in, and actual lines are 1px thick.

Different gray levels are used to indicate different classes. In this example, 0 is used for walls, 82 for window holes, and 187 for door holes. These values can be configured (see below) and anything outside the given ranges will be ignored, so it is possible to use a floor plan that contains more annotations, but only extract wall/door/window lines if given in a specific gray value.

Various other formats derived from or related to images can also be imported. For a complete list of supported formats, use simUI.supportedImageFormats. Following example illustrates the PGM format:

```
P2
18 6
3
0 0 0 0 0 0 0 0 0 0 0 0 2 2 2 2 0 0
0 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 0
0 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 0
0 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 0
0 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 0
0 0 1 1 1 1 0 0 0 0 0 0 2 2 2 2 0 0
```

After selecting the menu entry, an image file must be selected, then a few import options can be specified:

- *pixel size*: the scaling of the image. One pixel in the image will correspond to this length in the real world.
- For every class (walls, windows, doors):
  - *height*: the height of the wall/hole in the real world.
  - *range*: the range of the intensity (grayscale value) in the image.
- *color*: the color of the walls shape.
- *respondable*: wether the walls shape will be respondable.
- *optimize*: if checked, the lines will be dissected to a minimal number of rectangles, so the shape will contain a low count of primitive (cuboid) shapes.
- *invert image values*: if checked, the image values will be inverted before processing.

Importing LDR / MPD models

The LDraw importer add-on is available via [Modules > Importers > LDraw importer...]. Make sure the LDraw parts library is installed. The simLDraw plugin uses the ldrawloader from Christoph Kubisch.

See also the API functions related to LDR / MPD import.