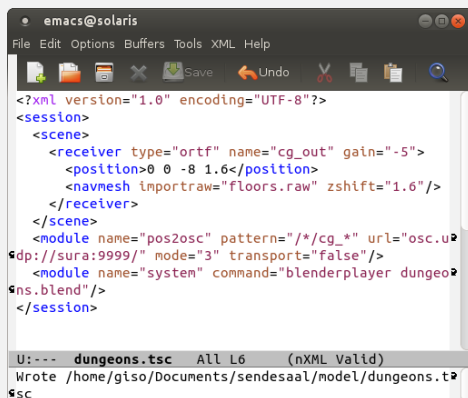**Tutorial 5:**
Audio-visual interaction

Summary

Create the game "Dungeons of screams" to design an interactive audio-visual detection experiment with TASCARpro and the blender game engine.

What will I learn?

- How to send position and orientation of objects to a game engine

- How to control TASCARpro properties from a game engine

- How to use navigation meshes and game controllers in TASCARpro

What can I use it for?

- Audio-visual environments for hearing aid research

- Game programming & having fun

The Story

Good old OLSA was caught and locked into the dungeon of screams. Find the OLSA and free it! Stop the other voices!

How to do it

The game is split into two processes: TASCAR will control the audio as well as the motion of objects. The blender game engine (http://www.blender.org/) will render the visual components and its Python interface can be used for the game logic.

Open your dungeon graphical environment with blender:

```
blender dungeon.blend &
```

Switch between "Default" view, "Game logic" view and "Scripting" view. We use blender logic bricks only for running our main Python script main.py – in that script we do all the logic, e.g., receiving and sending of OSC messages.

Look at the TASCAR file tutorial5.tsc. It looks like a mess – the first scene defines the main acoustic environment. Add some real screams to the "scream" object. You can record them in the gesture lab (ask for help if needed).

Out target is the ghost of OLSA. It is following a track defined in blender. See below how to change the path. We also have a door source, which will be triggered when we approach it. The walls of the room are also defined in blender. Our receiver is controlled by a game controller, but should always be linked to a navigation mesh. A second receiver (omni-directional) is used to generate the diffuse reverberation.

Most important part for the interaction between TASCAR and the blender game engine are the modules defined at the end of the session file:

```
51   <module name="pos2osc" pattern="/scene/*" url="osc.udp://sura:9000/" mode="3"
       transport="false"/>
52   <!-- control the door: -->
53   <module name="nearsensor" url="osc.udp://localhost:9877/" pattern="/scene/out"
       parent="/scene/door" radius="3" mode="0">
54     <msgapp path="/door/go"><f v="0"/><f v="2"/></msgapp>
55     <msgapp path="/scene/door/0/ap0/sndfile/loop"><i v="1"/></msgapp>
56     <msgdep path="/door/go"><f v="2"/><f v="4"/></msgdep>
57     <msgdep path="/scene/door/0/ap1/sndfile/loop"><i v="1"/></msgdep>
58   </module>
59   <!-- door animation: -->
60   <module name="motionpath" active="true" id="door" actor="/scene/door*">
61     <orientation>0 0 0 0
62     2 -130 0 0
63     4 0 0 0</orientation>
64   </module>
65   <!-- game controller -->
66   <module name="joystick" actor="/*/out"
67     x_ax="1" x_scale="-1" x_threshold="0.2" x_max="4" x_min="-4"
68     y_ax="0" y_scale="-1" y_threshold="0.2" y_max="4" y_min="-4"
69     r_ax="3" r_scale="-1" r_threshold="0.3" r_max="4" r_min="-4"
```

```
70    tilt_ax="4" tilt_scale="-0.7" tilt_threshold="0.3"
71    dump_events="false" maxnorm="0" url="osc.udp://sura:9000/"/>
```

The module `pos2osc` can track TASCAR objects and send their positions to the game engine.

The `nearsensor` can detect when we are close to objects, and can trigger user defined OSC messages.

Animations in TASCAR are created using the `motionpath` module. These animations can be triggered via OSC (see line 54).

Game controllers are connected with the `joystick` module. Axis events are interpreted to control the receivers, and all events can be sent also via OSC, to trigger actions in the game engine.

Create/update paths and meshes for use in TASCAR scenes

Motion paths (in form of a CSV file) and face groups (text file with vertex positions) can be exported from blender. If you would like to export something, put it into the "tascar" scene of your blend file (switch back to "game" scene before saving the blend file, otherwise the game will not work).

Modify your paths, reflector and navigation mesh to your needs. Keep it simple. To re-export to TASCAR, type

```
blender -b dungeon.blend -P export_mesh.py
```

The main action: Free OLSA

To free the OLSA, create a near sensor, and trigger an action in the game engine. For example, a button will be active only if you are near the target. If you trigger the button near a scream, you might kill it. Figure out yourself! See the end of `main.py` for an example.