Exercise 10: 3D Visualization

The goal of this exercise is to create a 3D animation of the trolley (movable body with pendulum). Three Python files are given for this purpose:

model.py

Contains the differential equation system (function `rhs` ("right hand side")) a function to calculate a single integration step (`calcStep` by using the solver from `scipy.integrate`).

functions.py

Contains the function `calcPositions` to calculate the position and orientation of the bodies from the simulation result and the function `setPokeMatrix` which passes the position information to the vtk geometry objects.

visualization.py

This is the main script. Here the geometry primitives and the scene object are created. For this purpose the function `updateScene` is given. To get an animation, it must be called repeatedly. Within `updateScene` the following things happen:

- a simulation step is calculated (`calcStep`),
- from those simulation result the positions and orientations of the bodies are calculated (`calcPositions`)
- these are passed to the geometry objects (actors) ((`setPokeMatrix`)) and
- finally the image is re-rendered (`renWin.Render`)

Tasks

1. Complete the function `calcPositions(y)` in the file `functions.py`. Calculate the position and orientation of the trolley and the load from the result y of one simulation step. Since the trolley can only move translationally, its orientation corresponds to the 3x3 unit matrix (`np.eye(3)`). The position of the load is calculated according to the formula:

$$r = [x + l\sin(\varphi), 0, -l\cos(\varphi)] \tag{1}$$

   For the calculation of the orientation matrix R_y of the load, see slide 12 of the corresponding lecture slides.

   Note: Use the given return value of the function for the variable name.

2. Complete the function `setPokeMatrix(actor, r, R)` in the same file. Here, fill the `poke` matrix elementwise with the values from r and R (see also slide 12).

3. Create a cuboid for the trolley in the `visualization.py` file (size: $x = 0.3, y = 0.1, z = 0.1$). Add a mapper and an actor for the cuboid

4. Repeat taks 3 for the load. But this time create a *cube* with edge length 0.1.

5. Add both actors to the renderer (`ren.AddActor(...)`)

6. Also in `visualization.py` : Import the needed functions to update the scene (see hints above).

7. Complete the `updateScene` function. Pay attention to the return values of each function and the order of arguments.

8. Test the animation. The command `iren.CreateRepeatingTimer(20)` specifies the update interval in ms, i.e. the speed of the animation.

Additional Tasks

9. Create a cylinder representing the rope (r=0.01, h=0.5). The cylinder gets the same orientation as the load, but still needs to be moved according to the pendulum length (calculate position r seperately).

10. Replace the visualization of the trolley with the `stl` file in the `/data` directory. Also add the hook as an stl file. To do this, use the `vtkSTLReader` (see slide 14). The hook will get the same position and orientation as the trolley.