Tutorial 1 - Basic Multibody Simulation

Abstract

In this practical lecture you should gather (first) experience with a multibody simulation. For an easy start, we use Bullet Physics with its natural binding PyBullet. Recently, PyBullet has been used in different scientific applications, e.g. deep-learning for mimic movement, simulation-reality gap learning and others.

The Bullet Physics Library is still under development and incorporates many usefull features like inverse kinematics, control, ray tracing.

In this tutorial you should: - Get to know some basic simulation parameters - See the influence of geometry on the simulation - Create a simple multibody via scripting

A word of Advice

- If you feel stuck, feel free to ask. However, the goal is that you play with the simulation and get a feeling for the influence of different parameters.
- Pybullet has a decent API documentation, the Pybullet Quickstart Guide, where almost every command is explained.
- Futhermore, the Examples Folder within the Github Repository is full of . . . examples.

We will use Jupyter Lab for this tutorial, which enables an interactive use of python - and other languages. You can start via:

```
# Use these commands in a terminal

# Activate the simulation environment of conda
source activate crrn_sim

# Navigate to an appropriate folder of your choice
cd my/simulation/folder

# Start a jupyter lab server
jupyter lab
```

A browser opens up, where you will find a user interface. Select `File -> New -> Notebook` and Python 3 as a kernel. Rename your notebook accordingly.

Part 1 - Simple Physics - Simple models

We will start by investigating the classical examples which lead to the model of gravity as given by Isaac Newton : A falling object. Oppose to using an apple, a sphere will suffice for our investigation.

You need the following commands:

```
import pybullet
import pybullet_data

# Start a server with a graphical user interface
pybullet.connect(*args)

# Get some objects to import
pybullet_data.getDataPath()

# Load a urdf
pybullet.loadURDF( filename, *args)

# Set the gravity
pybullet.setGravity(*args)

# And some way to step the simulation
pybullet.setRealTimeSimulation(*args)
pybullet.stepSimulation(*args)
```

Tasklist

- Start a simulation server
- Get the data path of pybullet_data
- Add the objects 'plane.urdf' and 'sphere2.urdf' and load them into the simulation
- Set the gravity accordingly
- Simulate for a sufficient time period, e.g. 10 seconds

What do you see?

Hints

- Ideally, the sphere has to be placed above the plane.
- Look up all commands in the Pybullet Quickstart Guide to get the necessary `*args`

Part 2 - Simple Physics - Extended

Next, we want to enhance the simulation. We clearly need to change some settings to get a real behaviour.

First, remember that the momentum of two colliding objects is (ideally) conserved

$p_1(t_0) + p_2(t_0) = p_1(t_1) + p_2(t_1)$

However, due to thermodynamics, we can assume that the overall energy of the mechanical system is constant or decreasing.

$E(t_0) = \sum_i \frac{1}{2m_i} p_i^2(t_0) \geq E(t_1)$

Which is modeled via the restitution of a collision

$p_i(t_1) = m_i v_i(t_1) = e\ m_i v_i(t_0)$

We can manipulate these parameters in Pybullet as well. Try the following commands:

```
# Get available information on a body
pybullet.getDynamicsInfo(*args)

# Change that information
pybullet.changeDynamics(*args)

# Can you make a debug parameter to change the dynamics on the fly
pybullet.addUserDebugParameter(*args)
pybullet.readUserDebugParameter(*args)
```

Hints

- If you use a simulation loop, you can apply the `pybullet.readUserDebugParameter` and `pybullet.changeDynamics(*args)` each loop
- You can use this tactic with the real time simulation as well.

Part 3 - Simple Physics - Hyperparameter

Still, some influences remain which we have not touched yet. To explore the hyperparameter of the simulation, we can use the following commands:

```
# Get the available parameter of the engine
pybullet.getPhysicsEngineParameter(*args)

# Set them
pybullet.setPhysicsEngineParameter(*args)
```

Where parameter like step size can be varied. Add user debug parameter for `fixedTimeStep`, `numSubSteps`, `erp`, `contactERP` and `frictionERP` and see whats happening.

Part 4 - Simple Physics - Create a Body

Next, we will create another sphere with the API.

Copy the following code snippet into a jupyter cell and execute:

```
pybullet.resetSimulation()

pybullet.loadURDF(pybullet_data.getDataPath() + '/plane.urdf', useFixedBase = True)

collision = pybullet.createCollisionShape(
    shapeType = pybullet.GEOM_BOX,
    halfExtents = [0.5, 0.5, 0.5]
)

visual = pybullet.createVisualShape(
    shapeType = pybullet.GEOM_SPHERE,
    radius = 0.5
)

new_sphere = pybullet.createMultiBody(
    baseMass = 1.0,
    baseCollisionShapeIndex = collision,
    baseVisualShapeIndex = visual,
    basePosition = [1, 1, 0.5]
)
```

A new sphere is created at position $r = (1, 1, 1)$.

Create another sphere `new_sphere_2` at location $r = (1.25, 0.75, 1.8)$ and look what happens. Remember to set the gravity and start the simulation.