TU Berlin Robotics WS2018/2019

Version: 15.10.2018, 5:00pm

Lab Assignment #0

Please upload your solution by **Sunday, October 21, 2018 at 23:59 pm**, using your ISIS account. Remember that this is a hard deadline, extensions impossible!

By submitting your solution you announce your intent to take part in the following assignments. The groups will be created from all assignment #0 submissions.

Please also fill out the Course Registration form on the ISIS website.

Preliminary

This assignment is a preparatory coding assignment which should help you to refresh your C++ coding skills.

You can compile and run the given code as follows:

```
cd folder_with_code
g++ -std=c++11 SpringMass.cpp SpringDamperMass.cpp main.cpp -o main
./main
```

After successful completion of the assignment, the output should be the following:

```
All tests passed (7 assertions in 2 test cases)
```

On unix systems, you can use the GNU debugger (GDB) for debugging using the following command:

```
gdb ./main
```

Note: We will test your solution with our own test cases and if your code fails to pass all test cases, you will not be allowed to continue the course. There will be no exceptions to this rule!

Implementations

In this simple exercise you will simulate the trajectory of a 1-D spring mass system. The equations of motion for the system are given as follow:

$$\begin{aligned} \dot{x}(t+1) &= \dot{x}(t) - \frac{k}{m}(x(t) - x_o) \\ x(t+1) &= x(t) + \dot{x}(t+1) \end{aligned}$$

where m is the mass of the object, x is the position of the object, x_o is the position of the object when the spring is unstretched, $\dot{x}$ is the velocity of the object and k is the spring constant.

The following step-by-step procedure will guide you in implementing the simulation. When implementing the methods, please also read carefully the doctype comments above each method, class and member declaration.

1. Implement the `struct Vec2d` (located in SpringMass.h) which should represent a two-dimensional vector (x and y). You might want to add a constructor for your convenience.

2. Implement the constructor of class `SpringMass`. Define the necessary member variables for storing the initial position and velocity of the object and the position and velocity of the object when the spring is unstretched and in equilibrium. Don't forget to create a variable for the current time.

3. Implement the method `SpringMass::step()`. Use the constants defined in the `SpringMass` class. It should perform one simulation step of the system according to the equations of motion given above. It should return the last simulated timestep.

4. Implement the method `SpringMass::getCurrentSimulationTime() const.`

5. Implement the method `SpringMass::getConfiguration(int t, Vec2d& state) const.` Given a time t, it should return the state (position, velocity) of the object at the time. Only times which have already been simulated should be allowed as input (return false if t is invalid, true otherwise).

6. Generate a trajectory with initial position 200, initial velocity 0 and x_0=161 for the spring mass system for t going from 0 to 500. Use your favorite plotting tool to visualize the generated data (position and velocity).

7. Implement the class `SpringMassDamper`. We now add a damper to our system. The equations of motion change to:

$$\begin{aligned} \dot{x}(t+1) &= \dot{x}(t) - \frac{b}{m}\dot{x}(t) - \frac{k}{m}(x(t) - x_o) \\ x(t+1) &= x(t) + \dot{x}(t+1) \end{aligned}$$

   where b is the damping coefficient.
   Implement the new class such that it follows the altered equations of motion. It should be a subclass of the class SpringMass.

8. Generate a trajectory with initial position 200, initial velocity 0, x_0=161 and b=1 for the spring mass damper system for t going from 0 to 500. Use your favorite plotting tool to visualize the generated data (position and velocity).

Deliverables

Your solution must contain the following files in a single zipped file with the name `yourName_Assignment0.zip`.

- `main.cpp SpringMass.cpp SpringMass.h SpringDamperMass.h SpringDamperMass.cpp catch.h`
- The **plot** containing the trajectory of the object **without damper**: `mass.png`
- The **plot** containing the trajectory of the object **with damper**: `mass_damper.png`

In addition to your submission, please also fill out the Course Registration form on the ISIS website.