Write new potentials for ReaDDy

Schöneberg

November 16, 2011

There are 3 Steps that need to be done in order to write new Potentials for ReaDDy

1 Create the potential class itself

1. Create a new potential class in readdy.impl.sim.core.pot.potentials
2. Here the logic of the potential is given, namely how the potential gradient is computed. This is the most important part of the implementation and where your logic comes into play.
3. Please stick to the name convention of naming single particle potentials P1_<newName> and pair potentials P2_<newName>.
4. Let the new class implement either the IPotential1 or IPotential2 interface. Depending on the degree of your potential (single or pair).
5. The most easy way would be co copy an existing potential and to change it according to your needs. e.g. Take P2_Harmonic. Assume that you want to add one additional parameter "additionalParameter" to the potential. We create in this tutorial the new potential **P2_Harmonic_AdditionalParameter**
6. Specify the essential input parameter keys that you want to use in the potential in the array essentialParameterKeys. e.g. Change the existing one

   ```
   private String[] essentialParameterKeys =
   new String[]{"id", "name", "type", "subtype", "forceConst"};
   ```

   to

   ```
   private String[] essentialParameterKeys =
   new String[]{"id", "name", "type", "subtype", "forceConst",
   "additionalParameter"
   };
   ```

7. Make the set_parameters() consistent with your specification of the essentialParameterKeys array e.g. add to the existing lines

   ```
   this.id = Integer.parseInt(parameters.get(essentialParameterKeys[0]));
   this.name = parameters.get(essentialParameterKeys[1]);
   this.type = parameters.get(essentialParameterKeys[2]);
   this.subtype = parameters.get(essentialParameterKeys[3]);
   this.k = Double.parseDouble(parameters.get(essentialParameterKeys[4]));
   ```

   the following line

   ```
   this.additionalParameter =
       Double.parseDouble(parameters.get(essentialParameterKeys[5]));
   ```

8. Now you are almost done. You only have to register the potential within ReaDDy.

2 Create a potential template entry in the param_potentialTemplates.xml input file

1. Create a new potential entry via copy and paset of one of the existing entries e.g. take the existing entry for the P2_Harmonic potential:

```
<potential>
    <potentialType>HARMONIC</potentialType>
    <potentialOrder>2</potentialOrder>
    <parameters>
        <param name="name" type="String" default="HARMONIC"/>
        <!-- subtype choices: ["spring", "attractive", "repulsive"] -->
        <param name="subtype" type="String" default="spring" />
        <param name="forceConst" type="double" default="1" />
        <param name="affectedParticleTypeIdPairs" type="int[][]" default="null" />
        <param name="affectedParticleIdPairs" type="int[][]" default="null" />
    </parameters>
</potential>
```

2. Change the <potentialType> to your favorite name of the new potential e.g.

```
    <potentialType>HARMONIC_AdditionalParameter</potentialType>
```

3. Make the parameters after the <parameters> tag consistent with the essentialParameterKeys that you specified in the java potential class itself. In our example add the following additional parameter line:

```
    <param name="additionalParameter" type="double" default="10" />
```

3 Enable the PotentialFactory to create your potential

1. Find the PotentialFactory in readdy.impl.assembly
2. Add a new entry in one of the functions createPotential1(...) and createPotential2(...), depending of the order of your potential (single or pair) e.g. take the existing lines of the harmonic potential

```
if (!matched && template.get_typeName().contentEquals("HARMONIC")) {
    matched = true;
    IPotential2 potential = new P2_Harmonic();
    potential.set_parameters(potentialParameters);
    return potential;
}
```

   and register the new potential by changing the copied lines to the following ones:

```
if (!matched &&
  template.get_typeName().contentEquals("HARMONIC_AdditionalParameter")) {
    matched = true;
    IPotential2 potential = new P2_Harmonic_AdditionalParameter();
    potential.set_parameters(potentialParameters);
    return potential;
}
```

3. Make sure that the type name specified in the template is consistent with the string in the check in the if statement.

4 Using the new potential

If you now want to use the potential, you have to add the following lines to the potential topology file tplgy_potentials.xml:

```
<pot
name="harmonic_particleRepulsion"
type="HARMONIC"
subtype="repulsive"
forceConst="0.25"
additionalParameter="10"
affectedParticleTypeIdPairs="all"
affectedParticleIdPairs="" />
```

The particle types or ids that are affected by the potential, as well as the parameters are up to you and only set in this example to the values above.