
Supplementary material for *A Companion Converter for Immersive Molecular Dynamics in Virtual Reality to Save Time on Creating Simulation Files*

Polina Pereshivkina, Nadezhda Karandasheva, Maria Mikhailenko and Mikhail Kurushkin*

*Corresponding author e-mail: kurushkin@scamt-itmo.ru

- 5 Chemistry Neuroeducation Laboratory, SCAMT Institute, ITMO University, 9 Lomonosova Str., Saint Petersburg, Russian Federation, 191002

Converting the mol2 files from Narupa Builder into xml files for the purpose of simulation

10 In the xml file, which can be run in NarupaXR, it is necessary to type all coordinates, temperature, types of bonds and other structural information into a new xml file. For simple molecules it is fast to do by hand but with larger structures it takes a lot of time. Although, molecules as large as glucose or caffeine can be assembled and optimized manually, larger molecules are extremely time-consuming to assemble in this way and yet it is not guaranteed that the simulation will eventually work, due to a high possibility of a misspelling. These difficulties prompted the creation of new automatization strategies for the most optimal and user-friendly experience with Narupa Builder and NarupaXR. The results of this research is an open access converter, which sufficiently speeds up the process of transforming the raw mol2 files, obtained from Narupa Builder, into the final xml files that are loaded into the NarupaXR environment. The code was created for the purpose of research, engineering and especially education.

15

20 The converter is publicly available via the following link (https://gitlab.com/teamSCAMT/converter_mol2_to_xml.git).

The converter reported in the present paper is a standalone code (however, it is not an executable); hence, in order to use it, several simple steps need to be taken. Further, these steps will be described in detail.

First of all, one needs to create an account on a GitLab platform (<https://gitlab.com/>). Next, find the account “teamSCAMT” and the project called “converter_mol2_to_xml”. Then click on the “Fork” button (Figure 1).



Figure 1. Clicking on the “Fork” button on the GitLab platform.

To run the code, install the development environment. The recommended environment is the “IntelliJ IDEA” (<https://www.jetbrains.com/ru-ru/idea/>).

Next, click on the converter_mol2_to_xml in a personal repository on the GitLab and copy the link (Figure 2).

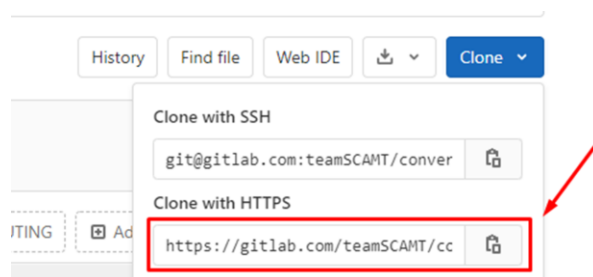


Figure 2. Cloning the HTTPS link.

After installation, open the program and create a new project from version control (Figure 3).



40 Figure 3. Creating a new project.

After that, paste the copied link in the URL line (Figure 4).



Figure 4. Pasting the copied link.

45

In order to run the code with the help of IntelliJ IDEA application, one needs to open all the three classes of the code: "Reader", "Beliviks" and "Writer", this can be done by clicking on them.

As a result, the folder with the converter code will appear on the desktop. Next, add all Narupa Builder mol2 files, which require conversion to the folder with the converter code. Folder hierarchy is

50 usually similar to the one given in Figure 5.

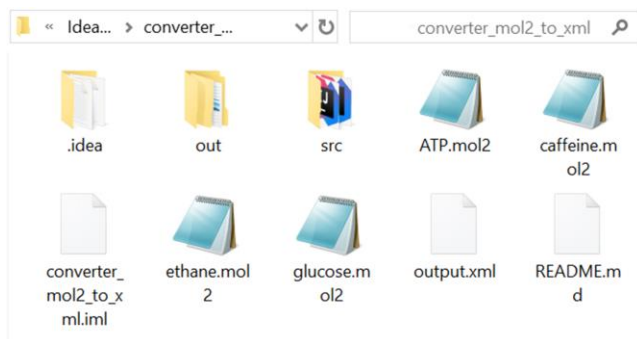


Figure 5. Folder hierarchy.

If everything is done right, the necessary mol2 file will appear in the IntelliJ IDEA between “Writer”
and “README.md” classes (Figure 6).

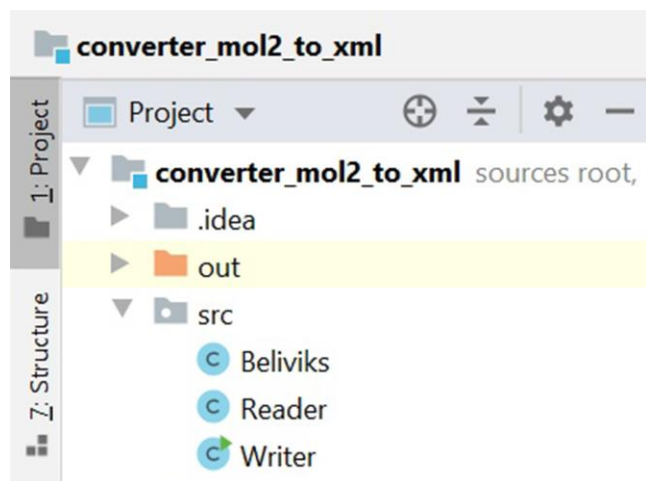


Figure 6. Adding the mol2 file obtained from Narupa Builder into the area where the finalized xml simulations files are opened.

Next, open class “Writer” and specify the name of the file, which requires conversion (an example is underlined in red in Figure 7).

```
public static void main(String[] args) throws ParserConfigurationException,
    DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
    DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
    Document doc = dBuilder.newDocument();
    Reader read = new Reader(Paths.get( first: "ATP.mol2"));
```

Figure 7. Opening the “Writer” class and indicating the name of the desired mol2 file.

Finally, run the code and the resulting xml code will appear in the output file. The converted into final xml file code is presented below in Figure 8:

```

1  <?xml version="1.0" encoding="utf-8"?>
2  <Simulation Name="c2h6">
3    <SystemProperties>
4      <SimulationBoundary SimulationBox="2, 2, 2" MinimumBoxVolume="2" />
5      <Thermostat Type="BerendsenThermostat" EquilibriumTemperature="200" MaximumTemperature="10000" BerendsenCoupling="0.003" />
6      <Integrator Type="VelocityVerlet" TimeStep="0.001" />
7    </SystemProperties>
8    <Topology>
9      <Templates>
10       <Residue Name="c2h6"> <!-- Created by Nadezhda Karandasheva March 16 2020-->
11         <Atoms>
12           <Atom Element="carbon" Position="-0.6165,0.0925,-0.6713" />
13           <Atom Element="carbon" Position="-0.5555,0.0741,-0.5798" />
14           <Atom Element="Hydrogen" Position="-0.7070,0.0516,-0.6599" />
15           <Atom Element="Hydrogen" Position="-0.5711,0.0516,-0.7504" />
16           <Atom Element="Hydrogen" Position="-0.6261,0.1910,-0.6857" />
17           <Atom Element="Hydrogen" Position="-0.4737,0.0221,-0.6042" />
18           <Atom Element="Hydrogen" Position="-0.6096,0.0221,-0.5137" />
19           <Atom Element="Hydrogen" Position="-0.5287,0.1615,-0.5394" />
20         </Atoms>
21         <Bonds>
22           <Bond A="0" B="1" />
23           <Bond A="1" B="7" />
24           <Bond A="1" B="6" />
25           <Bond A="1" B="5" />
26           <Bond A="0" B="4" />
27           <Bond A="0" B="3" />
28           <Bond A="0" B="2" />
29         </Bonds>
30         <ForceFields>
31           <InteractiveGaussianForceField GradientScaleFactor="2000" />
32           <MM3ForceField>
33             <MM3AtomMappings>
34               <MM3AtomMapping AtomPath="0" Type="1" />
35               <MM3AtomMapping AtomPath="1" Type="1" />
36               <MM3AtomMapping AtomPath="2" Type="5" />
37               <MM3AtomMapping AtomPath="3" Type="5" />
38               <MM3AtomMapping AtomPath="4" Type="5" />
39               <MM3AtomMapping AtomPath="5" Type="5" />
40               <MM3AtomMapping AtomPath="6" Type="5" />
41               <MM3AtomMapping AtomPath="7" Type="5" />
42             </MM3AtomMappings>
43           </MM3ForceField>
44           <LennardJonesForceField>
45             <LennardJonesAtomMappings>
46               <LennardJonesAtomMapping AtomPath="0" MM3Type="1" />
47               <LennardJonesAtomMapping AtomPath="1" MM3Type="1" />
48               <LennardJonesAtomMapping AtomPath="2" MM3Type="5" />
49               <LennardJonesAtomMapping AtomPath="3" MM3Type="5" />
50               <LennardJonesAtomMapping AtomPath="4" MM3Type="5" />
51               <LennardJonesAtomMapping AtomPath="5" MM3Type="5" />
52               <LennardJonesAtomMapping AtomPath="6" MM3Type="5" />
53               <LennardJonesAtomMapping AtomPath="7" MM3Type="5" />
54             </LennardJonesAtomMappings>
55           </LennardJonesForceField>
56         </ForceFields>
57       </Residue>
58     </Templates>
59     <Spawners>
60       <Spawner Name="c2h6" Template="c2h6" Count="1" />
61     </Spawners>
62   </Topology>
63 </Simulation>

```

Figure 8. The example of the code converted into xml format.

70 In the final xml file code, there are parts, which the user needs to fill in manually. These are responsible for the assignment of bond type and are given in **bold** as can be seen in an example below. In order to assign the bond type, one needs a glossary provided in the mm3.xml file in the \server\Assets\Data\ folder.

75 <MM3AtomMapping AtomPath="0" **Type=" "**>
<MM3AtomMapping AtomPath="1" **Type=" "**>
...
<MM3AtomMapping AtomPath="7" **Type=" "**>
...
80 <LennardJonesAtomMapping AtomPath="0" **MM3Type=" "**>
<LennardJonesAtomMapping AtomPath="1" **MM3Type=" "**>
...
<LennardJonesAtomMapping AtomPath="7" **MM3Type=" "**>