Porting a Java-based Brain Simulation Software to C++

Lukas Johannes Breitwieser

https://at.linkedin.com/in/lukasbreitwieser
Simulation of a Self-organizing Neural Network Using Axonal Growth Rules

Roman Bauer
The decision to port Cx3D to C++ was driven by our goal to simulate deeper & richer structures.
ITERATIVE PORTING WORKFLOW

Step 1: Pick a Java class with few dependencies

Step 2: Translate the Java code into C++

Step 3: Write code to enable communication between Java and C++

Step 4: Refactor remaining Java application

Step 5: Run automated tests
Communication between Java and C++

widget.h

```cpp
class Widget{
    ...
    virtual void foo(int i);
};
```

SWIG

moduleJNI.java

```java
public class moduleJNI {
    ...
    public final static native long Widget_foo(
        long jarg1, Widget jarg1_, int jarg2);
    ...
}
```

SWIG customizations

```java
public class Widget{
    ...
    public void foo(int i){
        ...
    }
}
```

moduleJAVA_wrap.cxx

```c
SWIGEXPORT jlong JNIFCALL
Java_package_moduleJNI_Widget_foo(JNIEnv *jenv,
        jclass jcls, jlong jarg1, jobject jarg1_,
        int jarg2) {
    ...
    return jresult;
}
```
- Rules for type conversions and type modifications
  e.g. function with parameter `const std::array<std::shared_ptr<Rational>, 3>&`
  that should translate into `Rational[]` on the Java side.

- Two-way-communication
  e.g. Java defined callback that is passed on to the native implementation and invoked from there.

Working with SWIG is sometimes hard, but it is better than writing all the boilerplate code oneself.
MANY THANKS TO MY SUPERVISORS / SUPPORTERS

Alberto Di Meglio  
Fons Rademakers  
Marco Manca  
Roman Bauer