Graphical implentation of cellsim

**Introduction -**

One of the final challenges in the process of creating cellsim was the graphical interface. It was decided that Python would be a good language to implement the graphical interface. To do this, it was necessary to *extend* Python; that is to say increase the functionality of Python with a statically-compiled language by creating a shared library.

**Section 1.1 – Why Python?**

It is necessary to find some other language for graphics regardless because C++ has no native support for graphical interfaces. Python is a good choice for these types of graphics because it is very powerful as far as interpreted languages go and there are numerous supported graphics libraries to choose from. Cellsim uses Zelle’s graphics library because it is almost perfect for our applications, it has a simple interface and plenty of support for moving shapes with lots of customizability.

**Section 1.2 – What is SWIG?**

SWIG stands for Simplified Wrapper and Interface Generator. It is a tool which can be used to generate wrappers and interfaces from lower-level statically-compiled languages like C and C++ to high-level interpreted languages including Perl, Python, Ruby and Tcl. This is useful for cases like ours where we want to make a lower-level implementation of a system, specifically cellsim makes use of type derivation for calling kernels of proteins easily.

**Section 1.3 – What’s a wrapper?**

The simplest explanation of what a wrapper is, it just makes something accessible in one form accessible to something else in another form; it just “wraps” something up in another packaging. For a more in-depth explanation of a wrapper is, continue reading; else you may skip this section. A wrapper is simply a function which takes one function and makes it usable to another. In cellsim, we see this with again the grid, where all of the member functions of the GridField class which need to be accessed in Python into simple functions, so something akin to:

int GetA() { return a; }

which is a member function of some class into

int GetA\_wrapper(void \*args) {

Class\* arg1;

arg1 = reinterpret\_cast<Class\*>(args);

int result = arg1->a;

return result;

}

which although it appears complicated on the surface it is simply reinterpreting am unknown pointer type (void\*) into our Class object and accessing the data contained in a. These then get put back into another class of the same name in Python when SWIG builds the rest of the Python interface for importing the shared library.