Input/Output infrastructure for Apothesis

Introduction

Apothesis is an open source software for simulating deposition processes via the kinetic Monte Carlo method[1]. As part of its input, it takes an arbitrary list of physical processes such as adsorptions and reactions, making it useful for modeling a wide range of situations. The input format for this list of processes is in JSON format, which has proven to be poorly suited to the structure of the data and difficult to use by researchers. In addition, the output is currently in the form of a log file, which cannot be analyzed without custom code to process it. This makes the software difficult to use. This project aims to create better options for input and output, allowing researchers to easily get the insights they need from the program. To solve this problem, the project will create:

- An input format suitable for describing the input processes, designed for ease of use.
- The necessary code to parse this input format
- A program to analyze the log file and export variables such as deposition rate and roughness in a format that is easily usable by graphing applications.
- The necessary infrastructure to export the 3-dimensional structure of the resulting solid.

Biographical Information

I am Craig White, a 1st year student at Florida Polytechnic University in the Computer Science program. I have had no coursework or professional experience relevant to this project. Instead, my experience comes mainly from researching python, and c++ out of personal interest, and testing my understanding by writing small example programs.

I have also been involved in competitive programming, which demands quickly written, efficient programs that solve problems primarily relating to data structures and algorithms. At first, I used python, but I switched to c++ for most competitions because python's performance wasn't good enough. My codeforces profile contains most of my past programming competition performance. My proudest accomplishment in competitive programming was winning the 2021 HSPT programming competition with my team in high school.

In 2020, I made some contributions to an open source project called TriggerReactor. TriggerReactor is a programming language for Minecraft server operators, for creating custom content without the need for Java plugins. My contributions started with documentation changes, but eventually, I started contributing to the main codebase. Through these contributions, I learned more about source control, software design, and software best practices.

[1] https://github.com/nixeimar/Apothesis

Contact info:

Github handle: gerzytet

Phone number: 352-422-1821

Email address: gerzytet@gmail.com

Project goals

By the end of the project, Apothesis will be much more usable than it was before. Making a specialized input format will enable researchers to rapidly adjust them to try out possibilities more quickly than before. With the new output functionality, they will be spared the work of writing custom code to analyze the log files.

Because there are a wide variety of formats that will be needed in the future for the 3 dimensional structure, this project will focus on creating infrastructure to pave the way for easily allowing output into a variety of formats. I will choose a simple format to export the structure into as a means of testing the infrastructure.

Implementation Plan

To create the input format, I will work with my mentor to design a format that intuitively represents the input data. Next, I will create a parser for the input in c++, modifying the existing input system as necessary to maintain compatibility with the JSON format. Finally, I will create documentation for the input format, translate the existing JSON examples into this format as usage examples, and compliment it with original examples if needed. Because this is an original format, it needs to be more thoroughly tested than the JSON format, so I will also create a variety of valid and invalid input files to test it, designed to trigger bugs and edge cases. Finally, I will reach out to subject matter experts for feedback on usability. The goal of this testing is to make sure the documentation is easy to understand, the format is intuitive, and when invalid files are provided, the error messages clearly point to the problem.

To create the output formats, I will start with existing code that has already been used to analyze the log files.[2] I will start with the logic that is used to determine the height map of the lattice at the end of the process and create the necessary code that would be needed to output the file no matter what format is being used. To test this code, I will create an exporter into the xyz format, a simple format suitable for testing. I will also create a separate program for generating a data for variables such as roughness, deposition rate over time. I will work with my mentor to gain an understanding of how to measure these, then extract them and

2

https://github.com/nixeimar/Apothesis/blob/ce54f38a46721863fb87e171d0210aa9bac05ac5/processing/Untitled.ipynb

choose a format widely compatible with other programs for export. I will make sure the file is at least compatible with excel, plus any other programs my mentor suggests as important for end users. This output can be tested by comparing the results to other competing software simulating a deposition process.

Timeline

May 20 – June 12 Community bonding period

During the community bonding period, I will read about research projects that have used Apothesis or similar software to get a clearer understanding of its use case. In particular, I will note how the researchers analyze the results of the deposition, so I know which data is most valuable to them. I will also be reading the source code of Apothesis itself, to get a feel for how the program works, and what parts need to be modified. I will design the input format during this period. By the end of community bonding period, I aim to have a clear understanding of the inner workings of Apothesis, the research that uses it, and how it needs to change to accommodate the new input/output features, and to have nailed down most details of the input format.

June 13 – June 27

During this period, I will build the parser for the new input format. By the end of this period, the parser should be working with at least simple inputs.

June 28 - July 11

These two weeks are for writing documentation and test files for the parser. By the end of them, me and my mentor should be confident that the parser works even in complex cases, it gives user-friendly error messages, and its documentation is accessible.

July 12 – July 27

In these two weeks, I will start creating the program to analyze the log file and export the 3-D structure of the lattice. The goal by the end is to make sure the previous work on the input format is presentable and ready to submit for phase one evaluations, and to make substantial progress on the 3-D structure export.

July 27 – August 4

In this week, I will finish the 3-D export feature. The code will be able to generate an xyz file as a proof of concept.

August 4 – August 25

During this period, I will create the variable over time output format, and make sure it is easy to use by other programs that researchers are likely to want compatibility with. By the end of this period, the code should be able to generate the output for any log file.

August 26 – September 12

The rest of this time is a buffer for handling unforeseen issues.

After Summer of Code

I will keep in touch with the Apothesis contributors, and maintain the code I wrote in case any bugs are found after the coding period. If the output options are not adequate, I will help researchers using Apothesis get the output in the format they need.