Team Status Update 1#:

Week-1:

Goals:

Determine blender add-on language

* Resolved: The Add-on is built using Python, but we may use tools to wrap and call C++ code from the original if needed

Set-backs:

Who work on what:

* Thomas: Looked for a way to run their code from the command line on Windows. It was designed to use GNU on Linux.
* Addison: Determined blender add-on language, started work on prototype
* Chase: tried to find what files were important to us form the AnisoMPM code
* Alex: Learning Blender interface and options to try to familiarize for add-on

Week-2:

Goals:

What is code can be use from the AnisoMPM:

* Most of their code is not well documented or has many read-me however the majority of the important stuff has a read-me. The most usable part of their code is in their project folder where they show demos that you can run, I overlooked this file and it's read me because it appeared to just be demos that were pretty rendered however after fully reading the read me as well as some of the demo files it appears that the demos are rendered in real-time, making this part of the code the most useful to translate data and objects from blender into a format that the AnisoMPM accept

Develop rudimentary add-on for blender:

* Simple python add-on. Works and confirms it has been enabled/disabled by printing to the console within blender

How the AnisoMPM data is use and where we can input it:

* We found the AnisoMPM has scripts that produce .mesh files. Our blender experience is primarily with .obj files, so we will have to find a way to translate .mesh files or find an extension on blender that can accept them

Set-backs:

Chase: The current setback in regards to use of their code is that my compiler is not accepting their code nor is it compiling. I am working on fixing that however the fix that we originally found might not be working.

Addison: I had a lot of trouble getting tools to wrap/call C++ code from python (swig and other libraries) to work, but now that we’ve found the original code can produce .mesh files via a python script, we likely will not need to call C++ directly and can easily wrap their script in ours for the add-on once we are able to compile and build it

Who work on what:

* Thomas: Found and installed a tool online that helps to use GNU on windows. It appears to have gcc working, though I could not find a way to update the versions of anything.
* Alex: Researched add-on tutorials and methods for crafting add-ons
* Addison: Created first blender add-on script (success!) worked on using swig to run C++ and Cmake code within the script (failure/setback)
* Chase: worked on trying to getting AnisoMPM code to Compile.

Week-3:

Goals:

simple interface for us

* Still being built and tested. Creating a pop-up menu is much more complex than the console commands

Set-backs:

Alex: Getting code to work in the Blender environment is a challenge. Code with dependencies creates an issue, as Blender’s path may not match the expected path of the operating system. More time looking into how Blender add-ons, particularly more complex ones, work is going to be needed.

Chase: Code is still not compiling, going to talk to are visor to get help with getting the AnisoMPM code to compiled

Who work on what:

* Thomas: Tried to get the code running using GNU on Windows.
* Alex: Blender Add-on research and development
* Addison: continued attempts with C++ within script
* Chase: is still trying to get files to compile.

Goals/ Plan for ​Next 3 Weeks:

Week 4:

Chase: work on getting the AnisoMPM domos to Compile

Alex: Finish getting the toolbar add--on installed

Addison: Implement a toolbar or other accessible interface within blender to add-on

Thomas: Try to get the code to compile using GNU on Windows. Chase is also working to compile it, but we’re using different approaches, so we’ll keep in touch if one of us gets it working.

Week 5:

Chase:run the demos in windows

Alex: Start working on a pop-up box that can be called via the button on the previously mentioned toolbar.

Addison: Work on importing .mesh files via the add-on template. If possible, use the meshes directly from working demo

Thomas: Create code to automatically generate header files as input for the anisofracture code. All the examples they made had their own header with all the information on how to run the simulations. We’re going to try to imitate those.

Week 6:

Chase:work on figuring out what in the demos can be changed and altered for use in our project

Alex: Get the pop-up box to accept user input, some text and some radio buttons, to allow us to set settings as needed.

Addison: figure out adjustable variables in the original code (elasticity, etc.) and see if we can modify them without recompiling the entire project

Thomas: Fine tune the code I wrote the previous week. Figure out what user input it needs to customize the header files properly.