### Installation of Project Chrono API

###### Download and install necessary softwares

**Visual Studio** - programming: <https://visualstudio.microsoft.com/downloads/>

·It seems that there would be some issue using VS2022. Maybe try VS2019?

**CMake** - Program project generation: <https://cmake.org/download/>

**SourceTree** - Package cloning: <https://www.sourcetreeapp.com/>

**CUDA** - Parallel computing: <https://developer.nvidia.com/cuda-downloads>

###### Download and clone (with SourceTree) necessary external packages

**Irrlicht -** Graphic engine: <https://downloads.sourceforge.net/irrlicht/irrlicht-1.8.5.zip>

**Eigen -** Linear algebra: <https://gitlab.com/libeigen/eigen/-/archive/3.4.0/eigen-3.4.0.zip>

**GLM -** OpenGL library: <https://github.com/g-truc/glm/archive/0.9.9.8.zip>

**GLFW** - OpenGL library: <https://github.com/glfw/glfw/releases/download/3.3.7/glfw-3.3.7.zip>

**GLEW** - OpenGL library: <https://sourceforge.net/projects/glew/files/glew/2.1.0/glew-2.1.0-win32.zip/download>

**Blaze** - Math library (clone): <https://bitbucket.org/blaze-lib/blaze/src/master/>

###### **Package Preparation**

1. Unzip all packages (preferably into one folder).
2. Compile GLFW package
3. Use CMake to create a program project for GLFW.

·Enter the root directory of GLFW into “Where to build the source code” and “Where to build the binaries”.

·Then press “Configure”, and choose the right version of Visual Studio in “Specify the generator for this project”, then press “Finish”.

·Wait for some values to come up, then select “BUILD\_SHARED\_LIBS”. Press “Configure” again.

·After the configuration is done, press “Generate”.

1. Compile GLFW with Visual Studio

·Change “Debug” to “Release” in the toolbar.

·Press “Build” → “Build the solution”.

###### **Get Project Chrono API**

1. Download the Project Chrono API from CyBox (directory: Hantao\Project Chrono for Geotechnical Simulation\Tutorials\Class #1 - Installation of Project Chrono API\Chrono-API.zip), and unzip it.
2. Compile Project Chrono API
3. Use CMake to create a program project for Project Chrono API

·Enter the root directory of Project Chrono into “Where to build the source code”, and enter a different directory into “Where to build the binaries” (preferably a “build” folder under the root directory of Project Chrono).

·Then press “Configure”, and choose the right version of Visual Studio in “Specify the generator for this project”, then press “Finish”.

·Wait for some values to come up. It is normal that some error information coming up.

·Enter the root directory of Eigen package into “EIGEN3\_INCLUDE\_DIR”, then press “Configure” again.

·Tick “ENABLE\_MODULE\_IRRLICHT”, “ENABLE\_MODULE\_OPENGL”, “ENABLE\_MODULE\_PARALLEL”, “ENABLE\_MODULE\_POSTPROCESS”, then press “Configure”.

·Enter the root directory of Blaze package into “BLAZE\_DIR”, enter “\lib\win64-visualstudio\irrlicht.lib” under the root directory of Irrlicht package into “IRRLICHT\_LIBRARY”, enter the root directory of Irrlicht package into “IRRLICHT\_ROOT”, tick “USE\_PARALLEL\_CUDA”, then press “Configure”.

·Enter “\bin\release\x64\glew32.dll” under the root directory of GLEW package into “GLEW\_DLL”, enter “\include” under the root directory of GLEW package into “GLEW\_INCLUDE”, enter “\lib\release\x64\glew32.lib” under the root directory of GLEW package into “GLEW\_LIBRARY”, enter “\src\release\glfw3.dll” under the root directory of GLFW into “GLFW\_DLL”, enter “\include\GLFW” under the root directory of GLFW into “GLFW\_INCLUDE\_DIR”, enter “\src\release\glfw3dll.lib” under the root directory of GLFW into “GLFW\_LIBRARY”, enter the root directory of GLM into “GLM\_INCLUDE\_DIR”, then press “Configure”.

·Press “Generate”.

1. Compile Project Chrono with Visual Studio

·Same way as the compilation of GLFW

###### **Test**

Run “\build\bin\release\demo\_PAR\_ballsSMC.exe”, if there is an OpenGL interface, then the program is successfully compiled.

###### **Reference**

·Official website of Project Chrono

·Tutorial video in the same directory of “Chrono\_API.zip”

### Direct Sheat Test (DST) on Irregular Particles

###### **Get example program code**

1. Download the Project Chrono API from CyBox (directory: Hantao\Project Chrono for Geotechnical Simulation\Tutorials\Class #3 - Direct shear test with irregular particles\code.zip), and unzip it.
2. Compile DST code
3. Use CMake to create a program project for DST code

·Enter the root directory of DST code into “Where to build the source code”, and enter a different directory into “Where to build the binaries” (preferably a “build” folder under the root directory of DST code).

·Then press “Configure”, and choose the right version of Visual Studio in “Specify the generator for this project”, then press “Finish”.

·Enter the “\cmake” directory under the root directory of Project Chrono API into “Chrono\_DIR”, then press “Configure” again.

·Press “Generate”.

1. Compile Project Chrono with Visual Studio

·Same way as the compilation of GLFW and Project Chrono API.

###### **Run example simulation**

1. Open the “myexe.exe” file under the “/build/release” directory under the DST code root directory.
2. (If the simulation turned off very soon, like several minutes later) Create a new folder named “sphereposition” under “/build/SHEAR\_SMC” directory under the DST code root directory.
3. Wait for the completion of simulation.

###### **Data analysis**

1. Open the “shear\_ratio.dat” file under folder “/build/SHEAR\_SMC” with Microsoft Excel. The first column is the shear length (the horizontal displacement of the bottom part of direct shear box); The second column is the shear-normal force ratio; The third column is the vertical displacement of the top plate of the direct shear box.
2. Under the folder “/build/SHEAR\_SMC/sphereposition”, there are generated CSV files storing positions, linear velocities, and rotation (in the form of quaternion) for each particle.
3. Under the folder “/build/SHEAR\_SMC/contactdata”, there are generated CSV files storing contact information for each pair of contact. Detail information includes: contact normals, positions of particles in contact, and mass of particles in contact.

### Triaxial Test on Irregular Particles

###### **Get example program code**

1. Download the Project Chrono API from CyBox (directory: Hantao\Project Chrono for Geotechnical Simulation\Tutorials\Class #4 - Triaxial test with irregular particles\code.zip), and unzip it.
2. Compile triaxial test code
3. Use CMake to create a program project for triaxial test code

·Enter the root directory of triaxial test code into “Where to build the source code”, and enter a different directory into “Where to build the binaries” (preferably a “build” folder under the root directory of triaxial test code).

·Then press “Configure”, and choose the right version of Visual Studio in “Specify the generator for this project”, then press “Finish”.

·Enter the “\cmake” directory under the root directory of Project Chrono API into “Chrono\_DIR”, then press “Configure” again.

·Press “Generate”.

1. Compile Project Chrono with Visual Studio

·Same way as the compilation of GLFW and Project Chrono API.

###### **Run example simulation**

1. Open the “myexe.exe” file under the “/build/release” directory under the triaxial code root directory.
2. Wait for the completion of simulation.

###### **Data analysis**

1. Open the “specimen\_stress.dat” file under folder “/build/SMC\_P” with Microsoft Excel. Data there include time step, strains in each direction, actual specimen length in each direction (*La*, *Lb*, *Lc*), and stresses in each direction. Noted that subscript “a” means vertical (*z*) direction, “b” means spanwise (*y*) direction, and “c” means streamwise (*x*) direction.
2. Under the folder “/build/SMC\_P/sphereposition”, there are generated CSV files storing positions, linear velocities, rotation (in the form of quaternion) and angular velocity (in the form of quaternion) for each particle.
3. Under the folder “/build/SMC\_P/contactdata”, there are generated CSV files storing contact information for each pair of contact. Detail information includes: contact normals, positions of particles in contact, and mass of particles in contact.

### 4. Build your own simulation

###### **4.1 Particle geometry preparation**

1. Use 3D computer graphics software (such as Blender and 3DS Max) to edit your particle geometry. Set the center of particle as point (0,0,0), and resize the particle geometry to real size (with SI units - 1 in the software should be as 1 meter). Save the particle geometry as an OBJ file. One OBJ file should store only one particle geometry.
2. Output the vertices information in OBJ files as CSV files (That means, every line in these CSV files should mean a 3D coordinate of a vertice in your particle geometry), to let the code read the particles more easily. One way to do it is to use the “readOBJ” Matlab code (https://www.mathworks.com/matlabcentral/fileexchange/18957-readobj) to read the vertice information and use “writematrix” function in Matlab to output them into CSV files.
3. Put the CSV files into the folder “obj” under the root directory of your DST or triaxial test code. To let the code read your particles more easily, it is recommended to name the CSV files by ordered numbers (like, 1.csv, 2.csv, etc).

###### **4.2 Code editing**

**4.2.1 Importing model particle geometries**

Typically, there are around 10,000 particles running in a simulation. If you don’t have one geometry per each particle, then you can use a certain number of model particle geometries, and copy them into the number of particles used in simulation (for example, if you have 40 models, then you can make the 41st particle the same as the 1st, the 42nd the same as 2nd, etc).

Lines 458-474 in example DST code, and lines 657-671 in example triaxial code are for model particle geometry importing. Read the comments and adjust that part of code if needed.

**4.2.2 Initial particle placement**

As shown in Fig. 1, in simulations, the particles are placed in the pattern of *a* (layers) × *b* (columns) × *c* (lines), before they fall and settle down in the container. To make the simulation run, any overlap of particles in the initial placement should be avoided. So, assume the length and width of the container are *l* and *w* respectively, and the largest length and width of any particle in the simulation are *lp* and *wp* respectively, then *b* should be no bigger than *l*/*lp*, and *c* should be no bigger than *w*/*wp*.

Lines 477-509 in example DST code and lines 673-702 in example triaxial code are about initial particle placement.

A picture containing text

Description automatically generated

Fig. 1 Initial particle placement

**4.2.3 Simulation parameters settings**

Lines 216-264 in example DST code, and lines 241-289 are about simulation parameter settings, including simulation time milestones, shear rate (in DST) or vertical strain rate (in triaxial test), normal pressure (in DST) or confining stress (in triaxial test), test container dimensions, particle material properties, and container properties. Read the comments and edit it if needed.

**4.2.4 Other details**

Both codeshave very detailed comments. You can read the comments and try edit them accordingly.