Chai3D – Unity3D

A very simple example for a network-based solution

Context.

This example is built upon the "06-ODE-exploration" project (module from CHAI3D-ODE). I attach a **very simple** project, which I called **Pick-and-Place** (just cubes visualization tested), to show how I saw and implemented communication between Chai3D and Unity3D. I was not able to test it with haptic devices, so you can first run without them to see how it works and then test and modify if needed to include the haptic devices. It is quite messy, sorry…

The idea behind is to provide a dynamic-link library from your chai3d project (here I made one out of a simplification and modification of the 06-ODE-exploration project that comes with chai3d). This library (Pick-and-Place C++ project) can be used directly from Unity3D (needs writing a plugin to expose C++ public methods to Unity3D in C#), and of course from another C++ project (see Testing project in C++). In my case, I used it from another C++ project which is a networking app (Networking project).

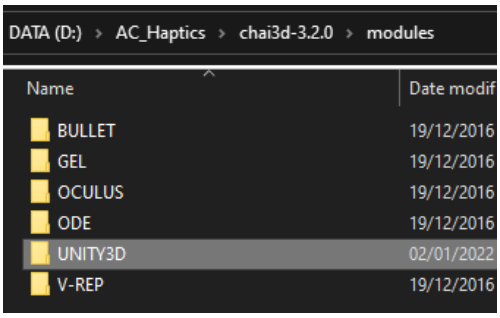
As I mentioned before, this is just a simple project to show how to communicate chai3d and unity3d. My thesis project involved modifications to the chai3d core to include the Arduino (serial comm) and the experimental protocol, but needed a lot of re-writing to be used by another dev ☺. I can send it if needed, but this small project is much more clear to start with.

The idea behind can be applied to a specific project from chai3d, like this Pick-and-Place, but I think that it will be a better approach to work on a general protocol to be able to communicate any Chai3D project to Unity3D (re-think shapes that it will support, devices capabilities and common configuration for force feedback and dynamics).

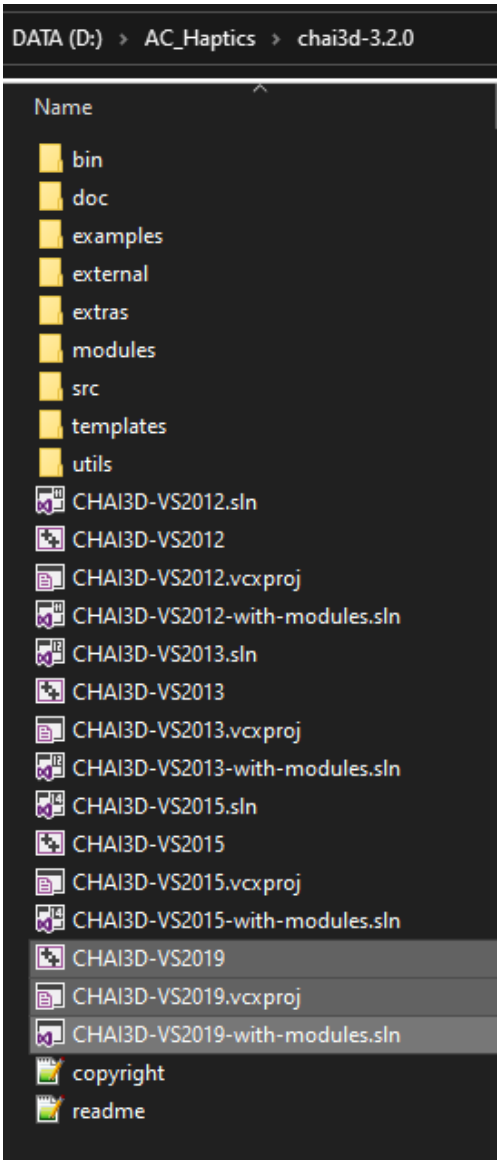
How to test everything…

→ Compilation and execution of the **C++ project**. (If you cannot make it work through this guide tell me and I can send you a link with the whole chai3d folder project).

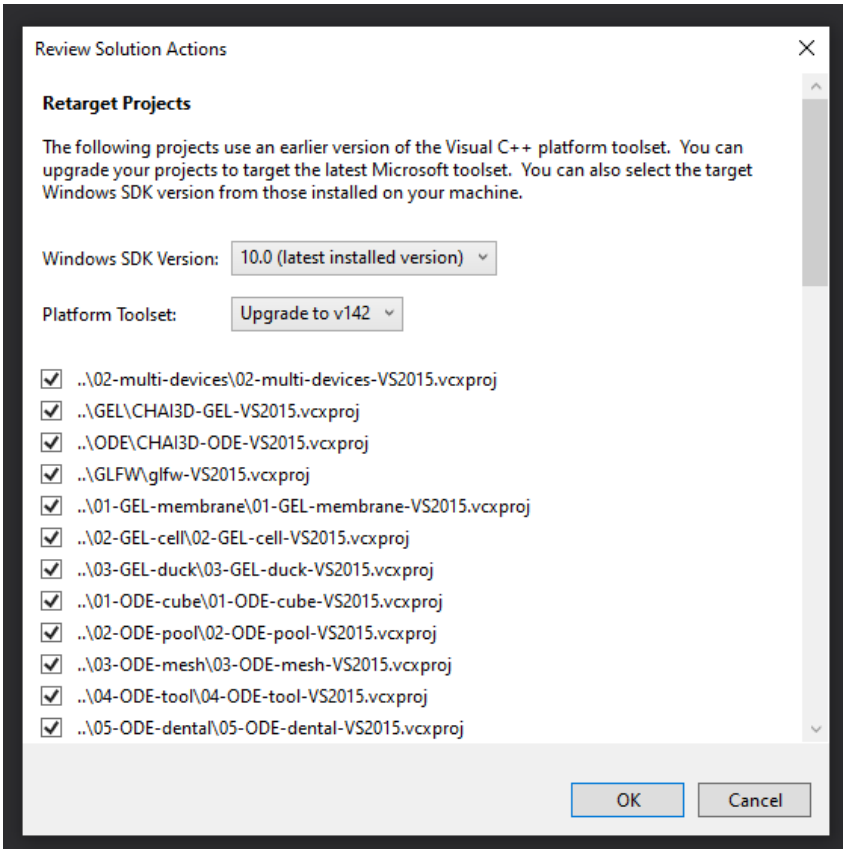
1. Download and extract chai3d for Windows (v3.2.0 for VisualStudio) from the official website: https://www.chai3d.org/download/chai3d-3.2.0-VisualStudio.zip
2. Extract and copy the UNITY3D folder I sent you under chai3d-3.2.0/modules:



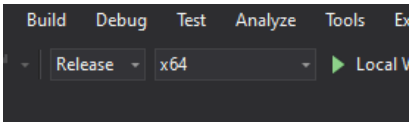
3. Copy the VS2019 project and solution files for Visual Studio (3 files) in the main chai3d-3.2.0 folder:



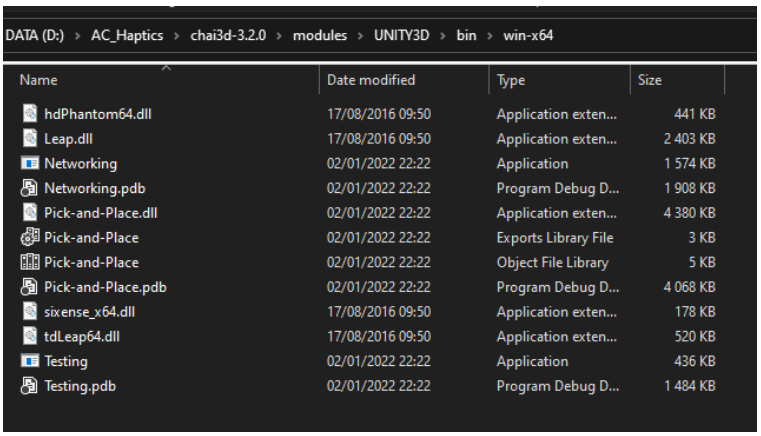
4. Open the *CHAI3D-VS2019-with-modules.sln* with **VisualStudio 2019** (chai3d download comes ready for vs2015, I migrated to vs2019, but you can use vs2015…)
5. If you use vs2019 you will be asked to retarget the projects to the more recent platform tool and sdk: press OK



6. Set the build options to Release x64 and build the whole solution (64 projects should successfully build)



7. In that which concerns my projects, you will get everything under the UNITY3D/bin/win-x64 folder :



8. What do you have and what you can do:
    a. The Pick-and-Place dll and lib if you want to either code a wrapper (plugin) from Unity3D (use it directly without client-server stuff), or generate another C++ project and use it from there.

i. The Testing.exe C++ app (see the UNITY3D/Testing/main.cpp code to see what it does) is a short (null) example to use the pick-and-place from another C++ project. I left it so the students can test modifications on the library without going through the networking app, if needed.

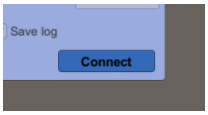
b. The Networking.exe is the chai3d "server" (TCP server and UDP sender). This is mainly what you asked me for. It allows you to establish the communication with UNITY3D (what I chose to use for my thesis basically).

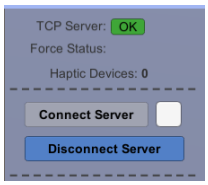
9. Launch the **Networking.exe**

→ Execution of the **Unity3D C# project** (I reused some of the UI from my last thesis project, but only the buttons I mention here are implemented in this example).

For the explanations below I assumed you are using only one computer (set to the correct ip and port if it is not the case).

10. Open the Unity3D project (I used Unity3D LTS version 2020.3.24f1, but there is no magic stuff in what I did, so any version will be ok). There is only one scene: *Pick-and-Place.*

a. Click on Connect 

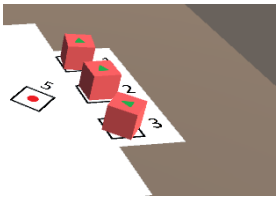


i. If everything is ok you will see TCP server: OK (you can also verify in the Networking.exe terminal)

b. Click on Start Application (launches the chai3d scene) 

c. Click on Reset Positions (this button here will make all cubes follow chai3d actual positions) 

d. You can use the *Position* and *Dynamics* (disabled on start) individual cubes buttons to test. Since I did not have haptic devices there is not much happening in the app, but if you enable dynamics for some of the cubes you will see it drop and follow the movement in chai3d (the surface material of the plane where cubes drop allows for them to slide –no friction– so you will see some small movements after some seconds).



e. When finish you can press on Stop Application, it will close the chai3d app, you can disconnect or close after.

That is all! I hope that when you connect the haptic devices you can do and see more stuff.

I am not quite sure about the world/local coordinates of the haptic positions… You can quickly check and change things on the VirtsaManager.cs file on the Unity3D side (*UpdateTransform* is a callback function to digest the message received through UDP), and the CDemo1.cpp file on the Chai3D side (*GetInteractObjects* is a function to pack all the info to send to Unity3D).

To better understand…

Pick-and-Place C++ project

You have here your actual chai3d project. Pick-and-Place.h contains export headers for the library. This can me “imitated” for the project you want, or better: rethink a generalization to wrap any project. The modularization is due to my experimental protocol, basically you build all the app and launch your threads to keep it alive.

Important functions:

- GetInteractObjects:
  This function is called from the UDP server to communicate positions, rotations, or what you need…

Networking C++ project

This is the server setup (I kept the basic stuff with the protocol used in my experiment). You have the main TCP server and a UDP sender that broadcast the info once app is running.

Important functions:

- ProcessMessage
  Here twitch accordingly to what you need from Unity3D.
  Protocol: COMMAND NUMBER_PARAMETERS <PAR1> … <PARn>
- AppVirtsa.cpp file
  Here you can do the calls you need to the library you developed (Pick-and-Place C++ project)

Unity3D project

I left in this project a simple TCP and UDP to communicate with the C++ networking app (TCPClient.cs UDPListener.cs)

Since I reused things from previous project, there are some scripts to run the UI and the whole thing: MenuSession, MenuExperimentation and SimulationManager, and others to update gameobjects: ForcepsManager, CubeManager,…

Important scripts:

- VirtsaManager.cs
  Here is everything you need to understand the backend of the app: just the sending TCP message and waiting for answer, and callback function for the UDP message reception.