# GET STARTED

### Installing OpenNI SDK

### Installing OpenNI SDK on Windows

Double-click the provided msi file to install the SDK on your Windows machine.
The installation performs the following:

- Copies the SDK to the target directory (default is C:\Program Files\OpenNI2 or C:\Program Files (x86)\OpenNI2)
- Installs a USB driver to be used with OpenNI-compliant devices
- Defines environment variables to be used when developing OpenNI applications (see Visual Studio)

### Installing OpenNI SDK on Linux

- Extract the tarball to a directory of your choice
- Go into this directory and run the install script: ./install.sh

The installation creates udev rules which will allow usage of OpenNI-compliant USB devices without root priviledges.

### Samples

OpenNI SDK arrives with pre-compiled samples that can be run immediately after installation. Under the installation directory, go to the Samples/Bin directory and run any of the samples there. Note that some samples have a graphical interface and may require a more powerfull graphic accelerator.

### Creating new project that uses OpenNI

Visual Studio

- Open a new project or an existing one with which you want to use OpenNI
- In the Visual Studio menu, open the Project menu and choose Project Properties.
- In the C/C++ section, under the ”General” node, select ”Additional Include Directories” and add ”$(OPENNI2_INCLUDE)” (if you use the 32-bit version) or ”$(OPENNI2_INCLUDE64)” (if you use the 64-bit version). These are environment variables that point to the location of the OpenNI Include directory. (The defaults are C:\Program Files\OpenNI2\Include or C:\Program Files (x86)\OpenNI2\Include)
- In the Linker section, under the ”General” node, select ”Additional Library Directories” and add ”$(OPENNI2_LIB)” (if you use the 32-bit version) or ”$(OPENNI2_LIB64)” (if you use the 64-bit version). These are environment variables that point to the location of the OpenNI Lib directory. (The defaults are C:\Program Files\OpenNI2\Lib or C:\Program Files (x86)\OpenNI2\Lib)
- In the Linker section, under the input node, select ”Additional Dependencies” and add OpenNI2.lib or OpenNI2.lib
- Ensure that you add the Additional Include and Library directories to both your Release and Debug configurations.
- Copy all the files from OpenNI’s redist directory (see environment variable ”$(OPENNI2_REDIST)” or ”$(OPENNI2_REDIST64)”) to your working directory. (The defaults are C:\Program Files\OpenNI2\Redist or C:\Program Files (x86)\OpenNI2\Redist). Be aware that when you run from command line, the working directory is the directory where the executable can be found, and where you run from Visual Studio the default directory is where the project file (.vcproj, .vcxproj) can be found.

Note:

You may ask Visual Studio to change working directory when debugging to the directory where the executable is by chaning ”Project Properties” -> ”Debugging” -> ”Working Directory” to ”$(TargetDir)”. Note that this setting is not kept as part of the project settings, but on a per-user, per-configuration basis.

**GCC / GNU Make**

In the following section, refers to the directory to where OpenNI SDK was extracted. Note that the installation does not define such an environment variable. Either define it yourself or use the full path.

Add the SDK Include directory, $OPENNI_DIR/Include, to your include path (-I)
Copy the files from the Redist directory, $OPENNI_DIR/Redist, to your execution directory
Add the execution directory to your lib path (-L)
Add libOpenNI2 to your library list (-l)
It is highly suggested to also add the “-Wl,-rpath ./” to your linkage command. Otherwise, the runtime linker will not find the libOpenNI.so file when you run your application. (default Linux behavior is to look for shared objects only in /lib and /usr/lib).

**Writing an Application**

Your code should include OpenNI.h header file.
The entire C++ API is available under the openni namespace.
Be sure to call openni::OpenNI::initialize(), to make sure all drivers are loaded If no drivers are found, this function will fail. If it does, you can get some basic log by callingopenni::OpenNI::getExtendedError() (which returns a string) Note that usually this method fails because OpenNI redist files weren’t copied to the working directory.
When closing your application, call openni::OpenNI::shutdown(), to allow OpenNI to close properly (unload drivers and such).
Open a device using its URI. You can get a list of available devices using openni::OpenNI::enumerateDevices(). Enumeration returns an array of openni::DeviceInfo objects, which include (among other things) the device URI. If you don’t care which device to use, you can specify openni::ANY_DEVICE as the URI. (to work with .oni files, use the path to the file as its URI)
Create a video stream by specifying the device and the sensor.
Be sure to destroy the stream and close the device when you’re done.