# Bundled Sample Applications

CodeXL contains several bundled sample applications to demonstrate CodeXL usage scenarios and features.

The bundled sample applications are:

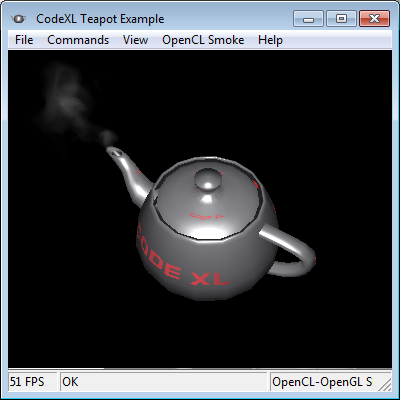
* **Teapot**

* **Matrix** **Multiply**
* [**D3DMultiThreading**](#_D3D12Multithreading)

## Teapot

The Teapot sample application draws a spinning teapot with steam coming out of its spout. The steam vapors movement is simulated using OpenCL. The rendering of the teapot and steam is implemented using OpenGL. In the Linux version of Teapot, the window, menu bar and UI widgets are implemented using the [FLTK](http://www.fltk.org/) library.

The Teapot application can be used to demonstrate the features of CodeXL’s GPU Debugger, GPU Profiler and Static Analyzer.



**Building the Teapot sample application**

### Building the Teapot sample application

CodeXL includes a pre-built binary of the Teapot application. Follow the steps below if you wish to build it yourself.

On Windows

Follow these steps to build AMDTTeapot on Windows from the default install location:

* Launch Visual Studio with administrator privileges.
* From the CodeXL menu, select the ‘Open Teapot Sample Project’ command.
* Wait for the project to open and press F7.
* **Note:** Teapot is preconfigured to use Windows 10 SDK release 10.0.10586.0. You can build Teapot with other Windows SDK releases by setting the Visual Studio project settings: Project > right-click > properties > General > target platform version.

On Linux

Follow these steps to build AMDTTeapot on Linux:

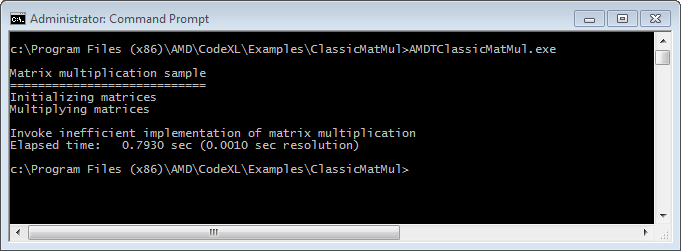
* Install the FLTK library
  + The bundled binaries of AMD Teapot were built with 64-bit binaries of FLTK 1.1.10.
  + Download FLTK sources from <http://www.fltk.org>
  + Extract and build the fltk libraries from the source files, then install the created fltk libraries
* Modifying The AMDTTeaPot Makefile
  + There are two makefiles in the Teapot sample folder. One is the /examples/Teapot/AMDTTeaPot/Makefile which you’ll need to edit, and the other is the /examples/Teapot/AMDTTeaPotLib/Makefile that does not need to be changed.
  + Open /examples/Teapot/AMDTTeaPot/Makefile in a text editor.
  + Replace the -L"*Replace with path to your local FLTK lib folder*" with the path of your local FLTK libraries, for example –L/usr/lib64 in case you placed the libraries in the system folder and you are running a 64bit system
  + Replace the -I"*Replace with path to your local FLTK headers folder*" with the path of your local FLTK include files. Please note that the FLTK headers are placed in a folder named “FL” and the path you provide should point to the parent of the “FL” folder. Example –I/user/include if you placed them in the system include and not –I/user/include/FL
* Building the Teapot
  + Teapot consists of a library and an application so first the library needs to be built.
  + cd to the /examples/Teapot/AMDTTeaPotLib folder
  + make all
  + if everything went well you’ll see libAMDTTeaPot.a in that folder
  + Now the main application can be built.
  + cd to /examples/Teapot/AMDTTeaPot
  + make all
  + If successful the output should be in /examples/Teapot/release. Look for a file named AMDTTeaPot-bin.

Note that CodeXL RPM and Debian packages install CodeXL and is bundled sample applications under the /opt folder, which requires elevated privileges to write to on some Linux distributions.

## Matrix Multiply

The matrix multiplication sample application performs multiplication of two matrices using 3 different implementations. All implementations are using ANSI C code.

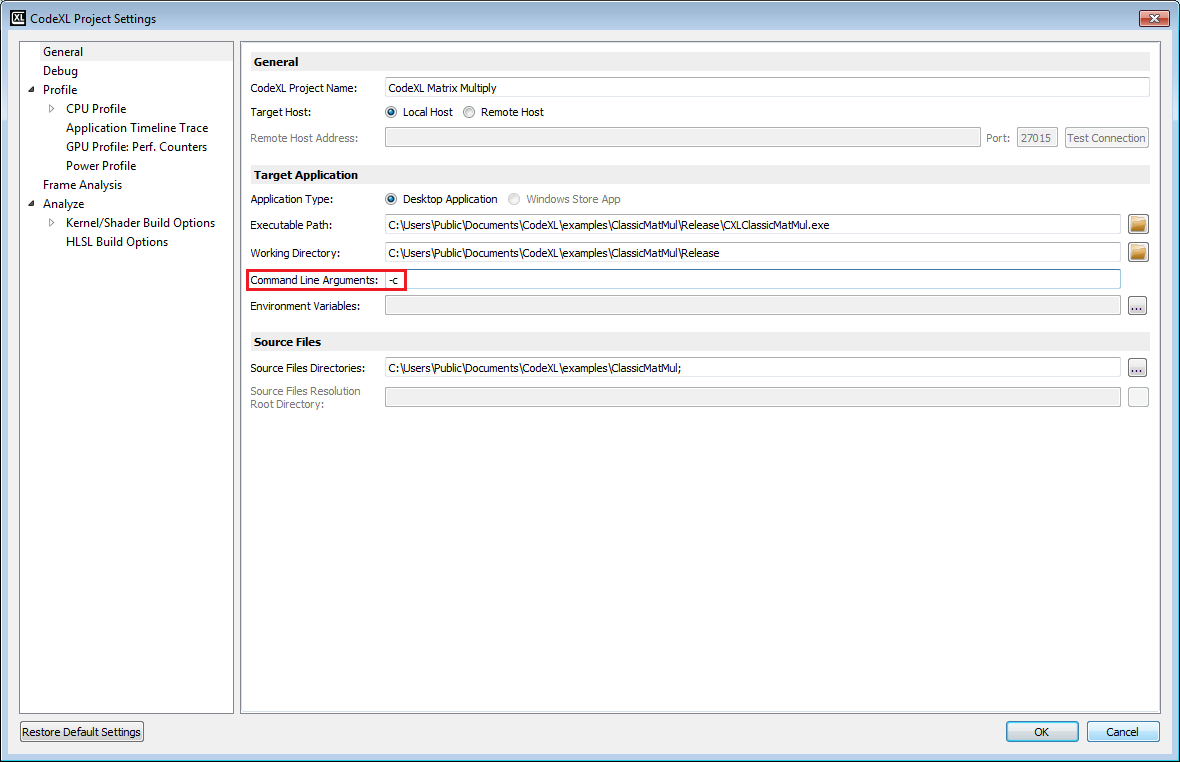
The sample takes a command line argument that defines which of the matrix multiplication implementations will be executed. See the following table for descriptions of the 3 implementations and the command line argument that invokes each one.



This sample demonstrates how the CPU Profiler can be used to detect bottlenecks, identify problematic memory access patterns, and verify improved performance.

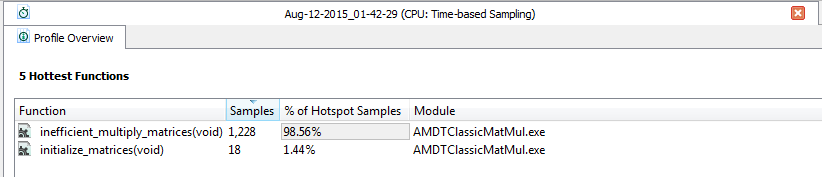
To select which implementation of the matrix multiplication will be performed, open the CodeXL Project Settings dialog, or in Visual Studio open the VS Project Settings dialog.

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Multiplication Implementation** | **Command line argument** | **Description** |
|  | Inefficient |  | Inefficient implementation that performs redundant loop iterations. |
|  | Classic | -c | Classic textbook implementation that uses naïve nested loops. The loops perform a sub-optimal memory access pattern. |
|  | Improved | -i | Improved implementation that uses nested loops with continuous memory access pattern. |

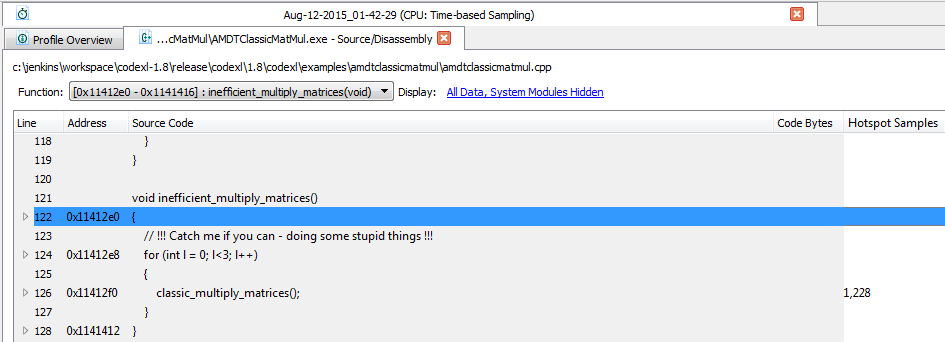


Project Settings for launching the classic implementation of matrix multiplication using the –c command line argument

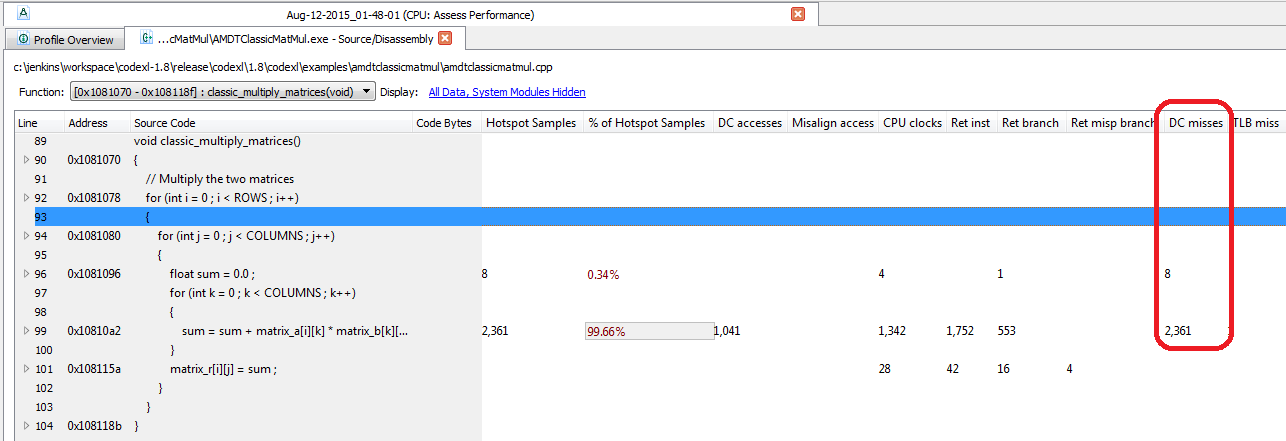
Use Time-Based Profiling to detect the redundant loop iterations in the inefficient implementation (launched by not supplying any command line argument).



Double clicking the inefficient\_multiply\_matrices(void) opens the source code view which demonstrates why this function is not efficient:

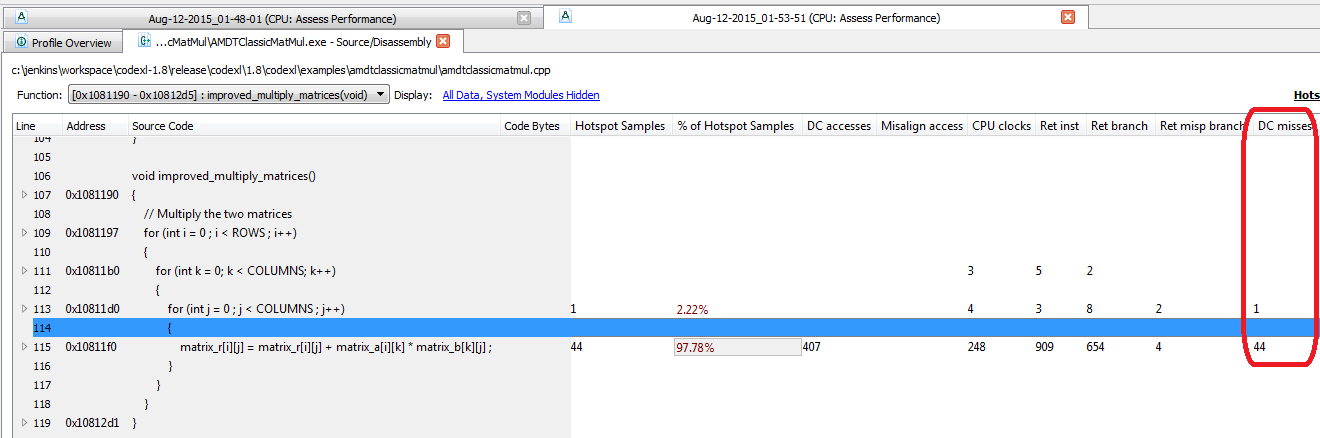


Use Event-Based Profiling with the Assess Performance session type to diagnose the problems in the classic implementation (launched by supplying “-c” as the command line argument):



Large number of cache misses in the internal nested loop non-sequential access to arrays

Finally, executing the improved implementation (launched by supplying “-i” as the command line argument) shows the number of cache misses is significantly reduced:



**Building the Matrix Multiply sample application**

### Building the Matrix Multiply sample application

CodeXL includes a pre-built binary of the Matrix Multiply application. Follow the steps below if you wish to build it yourself.

On Windows

Follow these steps to build Matrix Multiply on Windows from the default install location:

* Launch Visual Studio with administrator privileges.
* From the CodeXL menu, select the ‘Open Matrix Multiply Sample Project’ command.
* Wait for the project to open and press F7.

On Linux

Follow these steps to build Matrix Multiply on Linux:

* Navigate to *codexl\_folder\_path*/examples/ClassicMatMul/src
  + *codexl\_folder\_path* is the folder in which you installed or unzipped CodeXL
* Type ‘make’ and hit Enter.

Note that CodeXL RPM and Debian packages install CodeXL and is bundled sample applications under the /opt folder, which requires elevated privileges to write to on some Linux distributions.

## D3D12Multithreading

The D3D12Multithreading is a sample from the Microsoft D3D SDK, which is bundled with CodeXL.

The sample demonstrates the use of multiple thread with Direct3D 12.

The D3D12Multithreading sample can be used to get to know the Frame Analysis feature in CodeXL.



### Building the D3D12Multithreading sample application

CodeXL includes a pre-built binary of the D3D12Multithreading application. Follow the steps below if you wish to build it yourself.

The D3D12Multithreading sample is a windows only sample, and can only be built in Visual Studio 2015, with the Windows 10 SDK installed.

On Windows

Follow these steps to build D3D12Multithreading on Windows from the default install location:

* Make sure that you have a Visual Studio 2015 IDE installed.
* Make sure that the Windows 10 SDK is installed on your system.
* Launch Visual Studio 2015
* From the CodeXL menu, select the ‘Open D3D12Multithreading Sample Project’ command.
* Wait for the project to open and press F7.