

1 Overview

1.1 Location \$(AMDAPPSDKSAMPLESROOT)\samples\C++Amp\examples

1.2 How to Run See the *Getting Started* guide for how to build samples. You first must compile the sample.

Use the command line to change to the directory where the executable is located. The default executables are placed in \$(AMDAPPSDKSAMPLESROOT)\samples\C++Amp\bin\x86\ for 32-bit builds, and \$(AMDAPPSDKSAMPLESROOT)\samples\C++Amp\bin\x86_64\ for 64-bit builds.

Type the following command(s).

1. ArrayVsArrayView

This runs the program with the default options: $s = (1024 * 768 * 3)$.

2. ArrayVsArrayView -h

This prints the help file.

1.3 Command Line Options Table 1 lists, and briefly describes, the command line options.

Table 1 Command Line Options

Short Form	Long Form	Description
-h	--help	Show all command options and their respective meaning.
-q	--quiet	Quiet mode. Suppresses text output.
-e	--verify	Verify results against reference implementation.
-t	--timing	Print timing.
-d	--deviceId	Select deviceId to be used (0 to N-1, where N is the ID of the device to be used).
-v	--version	AMD APP SDK version string.
-x	--samples	Number of example input values (multiples of three).

2 Introduction

When choosing between array and array_view in C++ AMP, arrays can be useful:

- Because DX interop APIs expect arrays as parameters.
- As staging buffers to optimize frequent data transfers that take place between the CPU and the GPU.
- To measure the performance of the data transfer to an accelerator.

- To copy to an accelerator asynchronously.

Array_views are useful for future-proofing code.

In this example, we explore staging buffers and how data transfers can be optimized using AMP arrays with them.

3 Implementation Details

As the current C++ AMP is implemented on top of DirectX 11, a copy between host buffer and device buffer consists of the following stages:

1. Create a CPU side staging buffer.
2. Copy from the host buffer to the mapped staging buffer.
3. Unmap the staging buffer, and perform a copy from the staging buffer to the GPU buffer.
4. Release the staging buffer.

Staging arrays are differentiated from normal arrays by their construction. A normal array is constructed by specifying the `accelerator_view` on which the array is to be created. If this parameter is not provided, the array is created on the default accelerator.

A staging array is constructed with a second `accelerator_view` `acclView2` (along with the `accelerator_view` `acclView1`, which specifies the device on which the array is to be created), where:

- The `stagingArray` is physically located on `acclView1`. So the “`accelerator_view`” property of `stagingArray` returns the value of `acclView1`.
- The second `accelerator_view` parameter is called the `associated accelerator_view`. It is a hint to the runtime that a high frequency of data transfers between this array and other arrays on `acclView2` will occur.

The C++ AMP runtime uses this hint so when the `stagingArray` is to be copied to another array on `deviceAcclView`, it avoids an additional copy, buffer creation, and releases; thus saving valuable time. Currently, it only helps in the case where `acclView1` is a CPU `accelerator_view`, and `acclView2` is a GPU `accelerator_view`.

This example shows how C++ AMP staging arrays can be created and used efficiently with a noticeable performance boost over `array_view`. In this example, a simple color conversion routine of RGB to YUV 4:4:4 is used to bring out the uses and advantages of staging arrays.

4 Recommended Input Option Settings

For best performance, enter the following on the command line: `-x 2359296 -q -t -e`

5 References

1. <http://www.danielmoth.com/Blog/array-And-Arrayview-From-Amph.aspx>
2. <http://blogs.msdn.com/b/nativeconcurrency/archive/2012/07/17/choosing-between-array-and-array-view-in-c-amp.aspx>

3. <http://blogs.msdn.com/b/nativeconcurrency/archive/2011/11/10/staging-arrays-in-c-amp.aspx>
4. <http://en.wikipedia.org/wiki/YUV>

Contact

Advanced Micro Devices, Inc.
One AMD Place
P.O. Box 3453
Sunnyvale, CA, 94088-3453
Phone: +1.408.749.4000

For AMD Accelerated Parallel Processing:
URL: developer.amd.com/appsdk
Developing: developer.amd.com/
Support: developer.amd.com/appsdksupport
Forum: developer.amd.com/openclforum



The contents of this document are provided in connection with Advanced Micro Devices, Inc. ("AMD") products. AMD makes no representations or warranties with respect to the accuracy or completeness of the contents of this publication and reserves the right to make changes to specifications and product descriptions at any time without notice. The information contained herein may be of a preliminary or advance nature and is subject to change without notice. No license, whether express, implied, arising by estoppel or otherwise, to any intellectual property rights is granted by this publication. Except as set forth in AMD's Standard Terms and Conditions of Sale, AMD assumes no liability whatsoever, and disclaims any express or implied warranty, relating to its products including, but not limited to, the implied warranty of merchantability, fitness for a particular purpose, or infringement of any intellectual property right.

AMD's products are not designed, intended, authorized or warranted for use as components in systems intended for surgical implant into the body, or in other applications intended to support or sustain life, or in any other application in which the failure of AMD's product could create a situation where personal injury, death, or severe property or environmental damage may occur. AMD reserves the right to discontinue or make changes to its products at any time without notice.

Copyright and Trademarks

© 2012 Advanced Micro Devices, Inc. All rights reserved. AMD, the AMD Arrow logo, ATI, the ATI logo, Radeon, FireStream, and combinations thereof are trademarks of Advanced Micro Devices, Inc. OpenCL and the OpenCL logo are trademarks of Apple Inc. used by permission by Khronos. Other names are for informational purposes only and may be trademarks of their respective owners.