Bremen

Massively Parallel Algorithms - Tutorial

Abhishek Srinivas, Gabriel Zachmann

University of Bremen, Germany

cgvr.cs.uni-bremen.de

Our Tasks Today

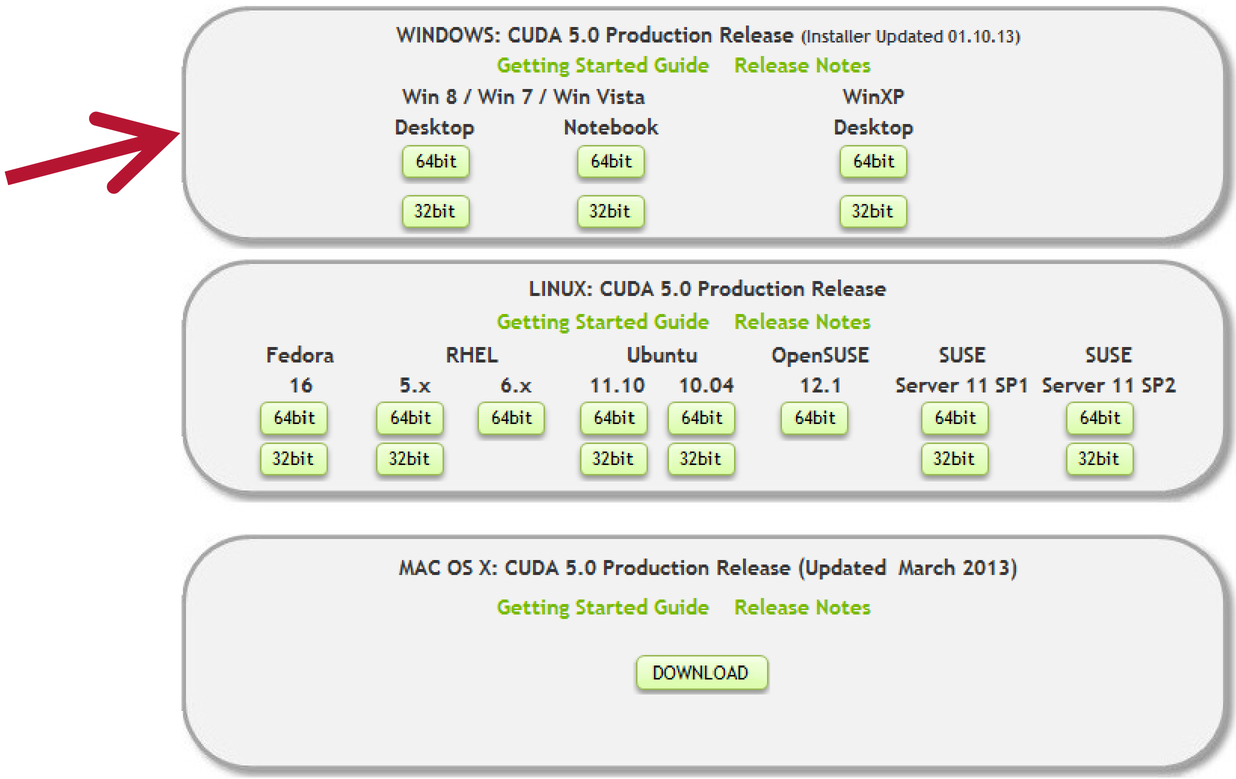
- Get & install Cuda
- Compile & run Cuda programs
- Debugging Cuda code
- Additional tools for some exercises

- Questions
  - Who has a Notebooks with an Nvidia Chip or
  - A computer with an Nvidia graphics adapter at home?
  - Whats the Cuda capability (later how to find out)

- We have PC's with Nvidia graphics adapter in our Lab
  - Just ask me for an account

Preparing Your Computer for Cuda

- Get Cuda from https://developer.nvidia.com/cuda-downloads
- I recommend Windows + Visual Studio 2010 + Nsight (debugging)



Documentation

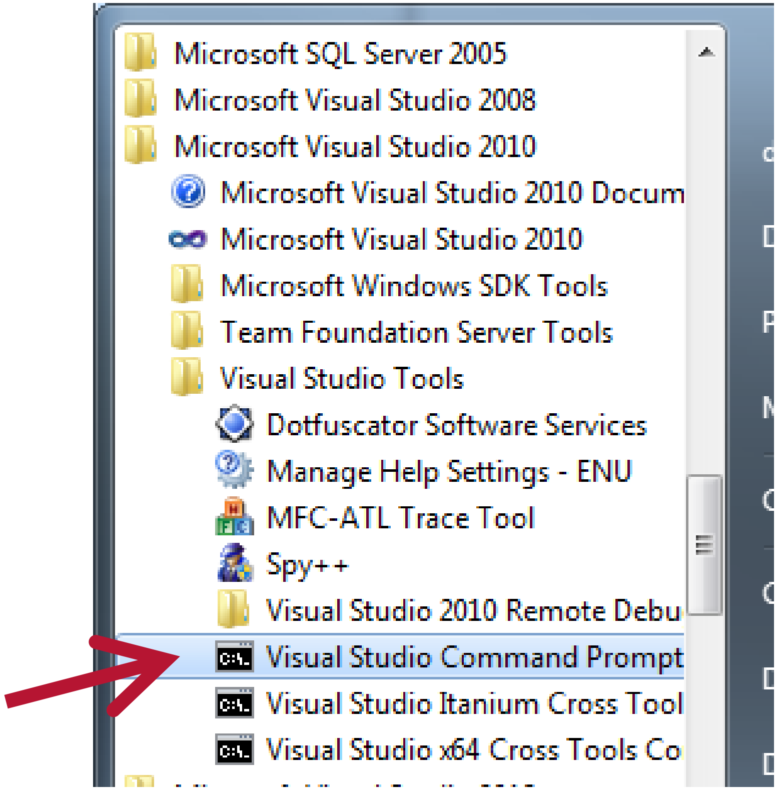
- `%NVCUDASAMPLES_ROOT%\doc`
  - print path: `echo %NVCUDASAMPLES_ROOT%`
- Important files
  - CUDA_C_Programming_Guide.pdf
  - CUDA_Toolkit_Reference_Manual.pdf
  - nvcc.pdf

- Our first CUDA program: „Hello World“ from the lecture
- Create the file `helloworld.cu`

```
#include <stdio.h>

__global__
void printFromGPU( void )
{
	printf( "hello world!\n" );
}

int main( void )
{
	printFromGPU<<<1,2>>>(); // kernel launch
	cudaDeviceSynchronize(); // important
	return 0;
}
```

Windows + Visual Studio Users

- Check your system variable PATH for the CUDA Toolkit binary folder NVIDIA_GPU_Computing_Toolkit\v5.0\bin\
- You need a Visual Studio Command Prompt (to ensure cl.exe is available)



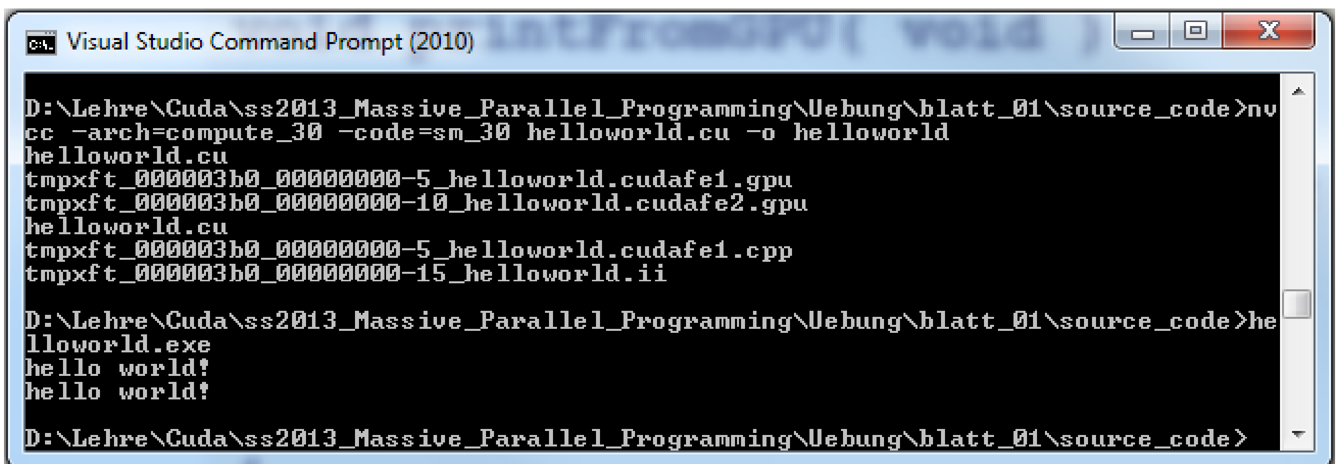
- Compilation

```
% nvcc -arch=sm_30 helloworld.cu -o helloworld
```

- Running

```
% ./helloworld
```

- You should see something like this:



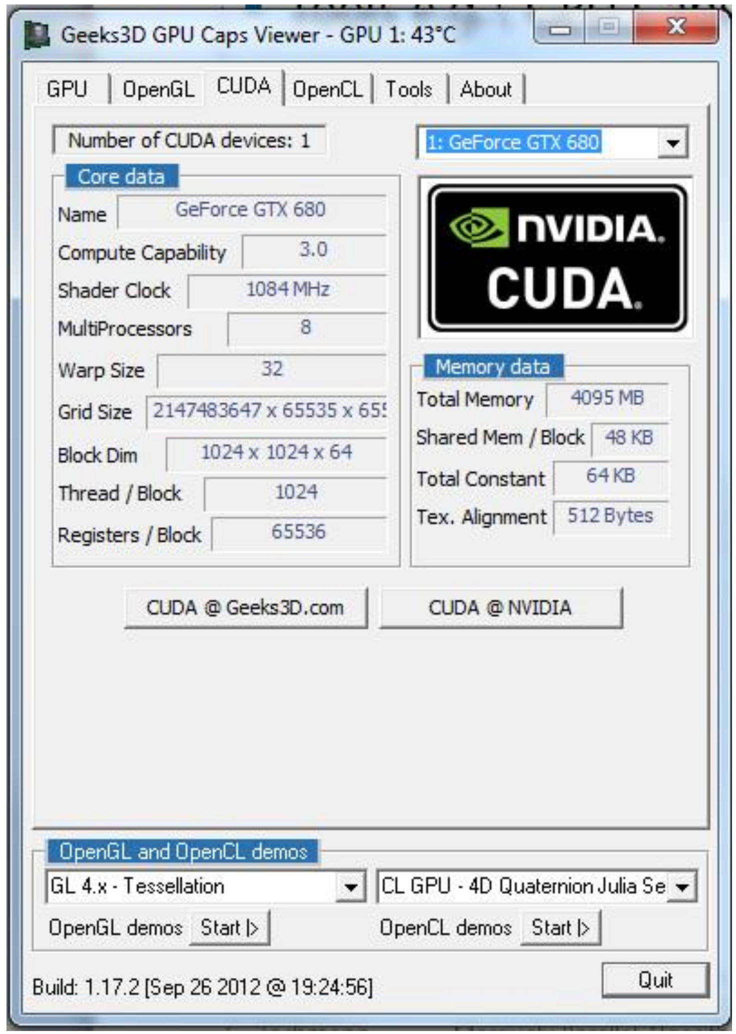
What Architecture I have to compile for?

- Query architecture version using Cuda
  - you can compile with `sm_10`

```
#include <stdio.h>

int main( void )
{
    int devID;
    cudaGetDevice( &devID );
    cudaDeviceProp props;
    cudaGetDeviceProperties( &props, devID );
    printf("CUDA device %d: \"%s\" with compute
          capability %d.%d\n", devID, props.name,
          props.major, props.minor );
    return 0;
}
```

Much More Useful Stats

- Tools e.g. „GPU Caps Viewer“



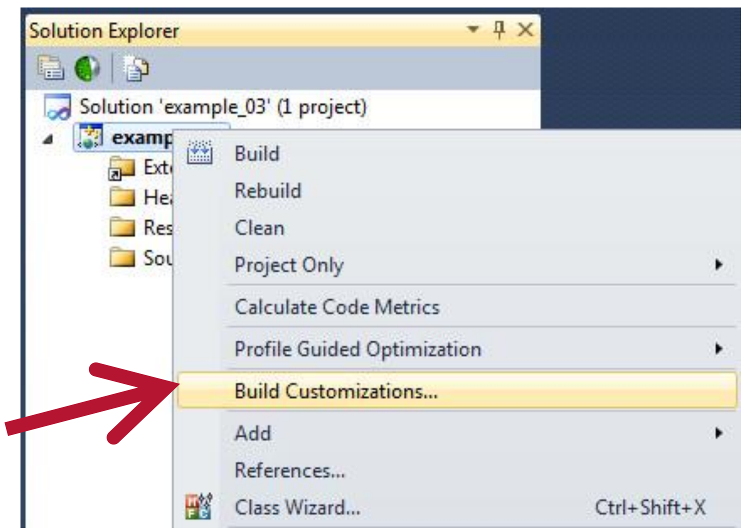
Overview of Architectures

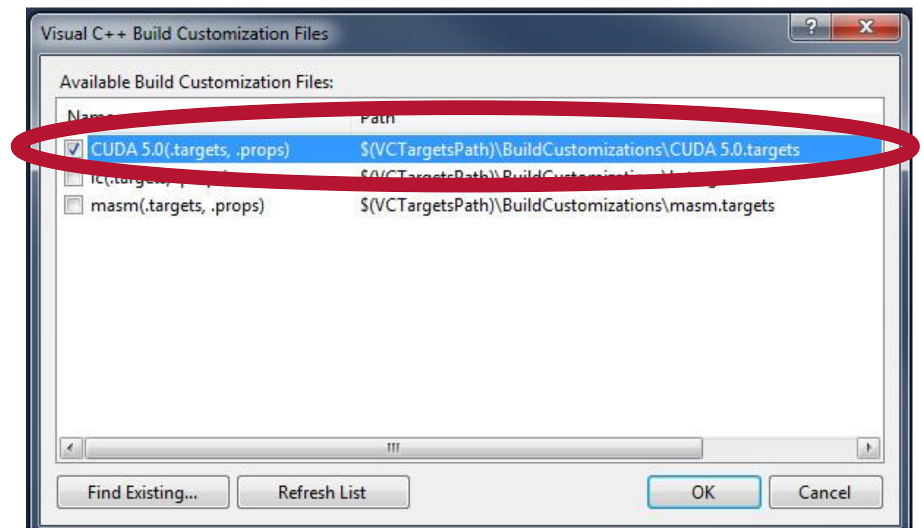
Graphics Adapter	Multiprocessors	Cuda cores	Compute Capability
Tesla K20	13	2496	3.5
GeForce GTX 680	8	1536	3.0
GeForce GTX 590	2 x 16	2 x 512	2.0
GeForce GTX 580	16	512	2.0
GeForce GTX 570	15	480	2.0
GeForce GTX 560 Ti	8	384	2.1
GeForce GTX 560	7	336	2.1
GeForce GT 440	2	96	2.1
GeForce GTX 480	15	480	2.0
GeForce GTX 470	14	448	2.0
GeForce GTX 465	11	352	2.0
GeForce GTX 295	2×30	2×240	1.3
GeForce GTX 280/GTX 285	30	240	1.3
GeForce GTX 260	24	192	1.3
GeForce 210	2	16	1.2
GeForce GT 240	12	96	1.2
GeForce GT 220	6	48	1.2
GeForce GT 130	12	96	1.1
GeForce GT 120	4	32	1.1
GeForce GTS 250	16	128	1.1
GeForce 9800 GX2	2×16	2×128	1.1
GeForce 9800 GTX	16	128	1.1
GeForce 9800 GT	14	112	1.1
GeForce 9600 GSO	12	96	1.1
GeForce 9600 GT	8	64	1.1
GeForce 8800 GTX/Ultra	16	128	1.0
GeForce 8800 GT	14	112	1.1
GeForce 8800 GTS	12	96	1.0
GeForce 8600 GT/GTS	4	32	1.1
GeForce 8400 GS/GT	2	16	1.1

More on https://developer.nvidia.com/cuda-gpus

Integrated Development Environment

- We don‘t want to do commandline developing
- I recommend: Windows 7 + Visual Studio 2010
- How to create a Cuda Project in VS
  1. Create an empty Win32 console application
  2. Add cuda build support

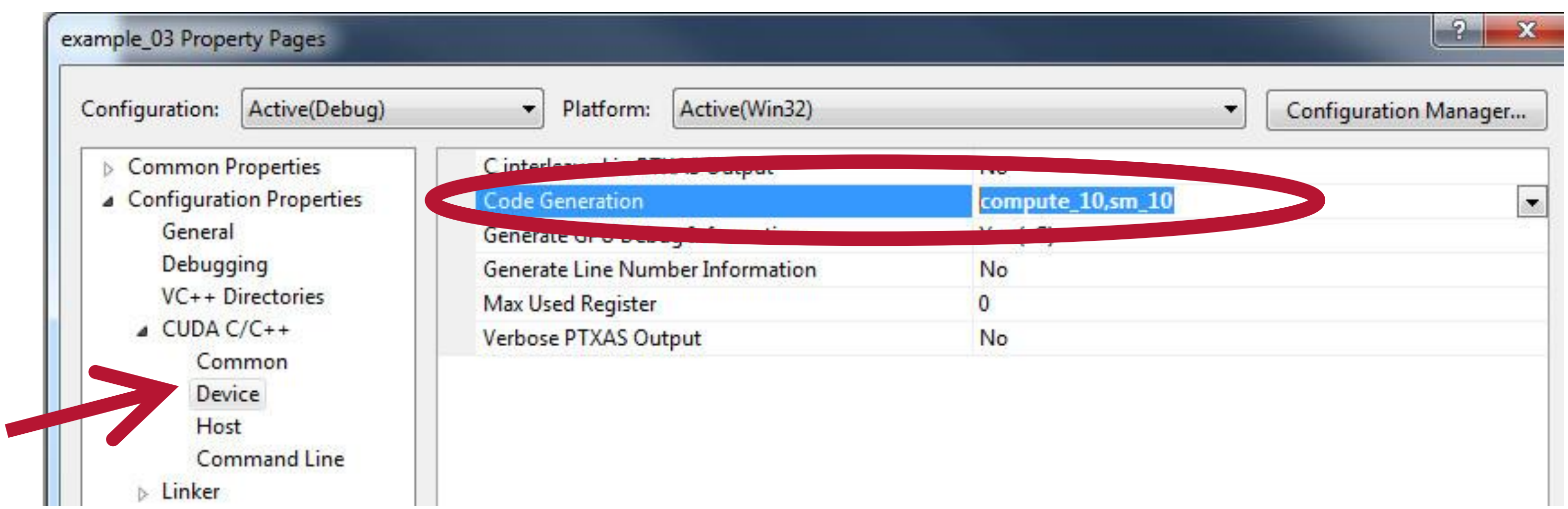




  3. Add your source file(s) to the project

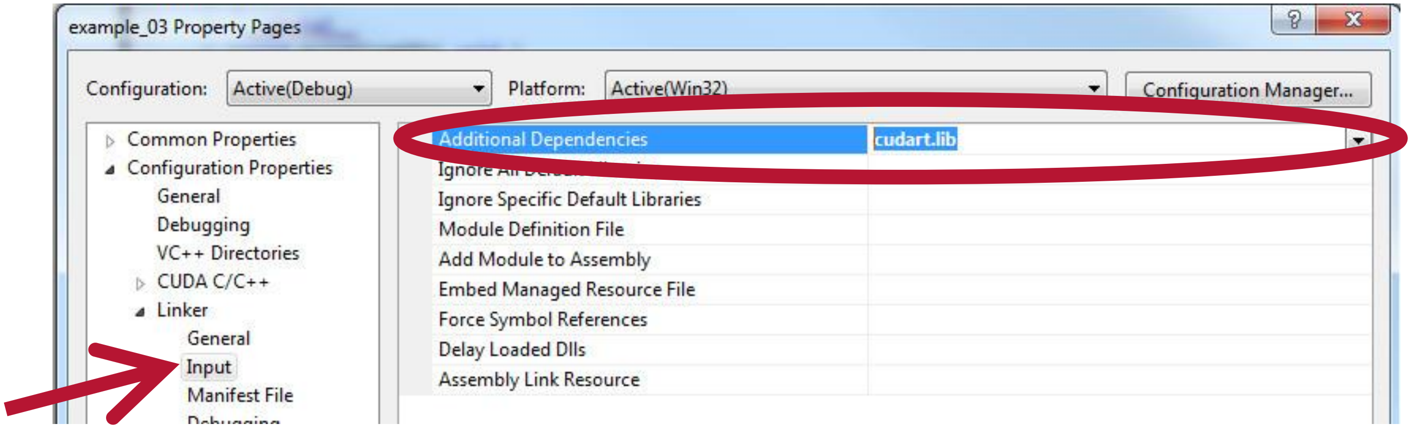
Compilation

- Try to compile your helloworld.cu
  - `error : calling a __host__ function("printf") from a __global__ function("printFromGPU") is not allowed`
- Whats wrong?
- Device printf needs Compute Capability ≥ 2.0
- Set in VS–Device Properties



Next Try

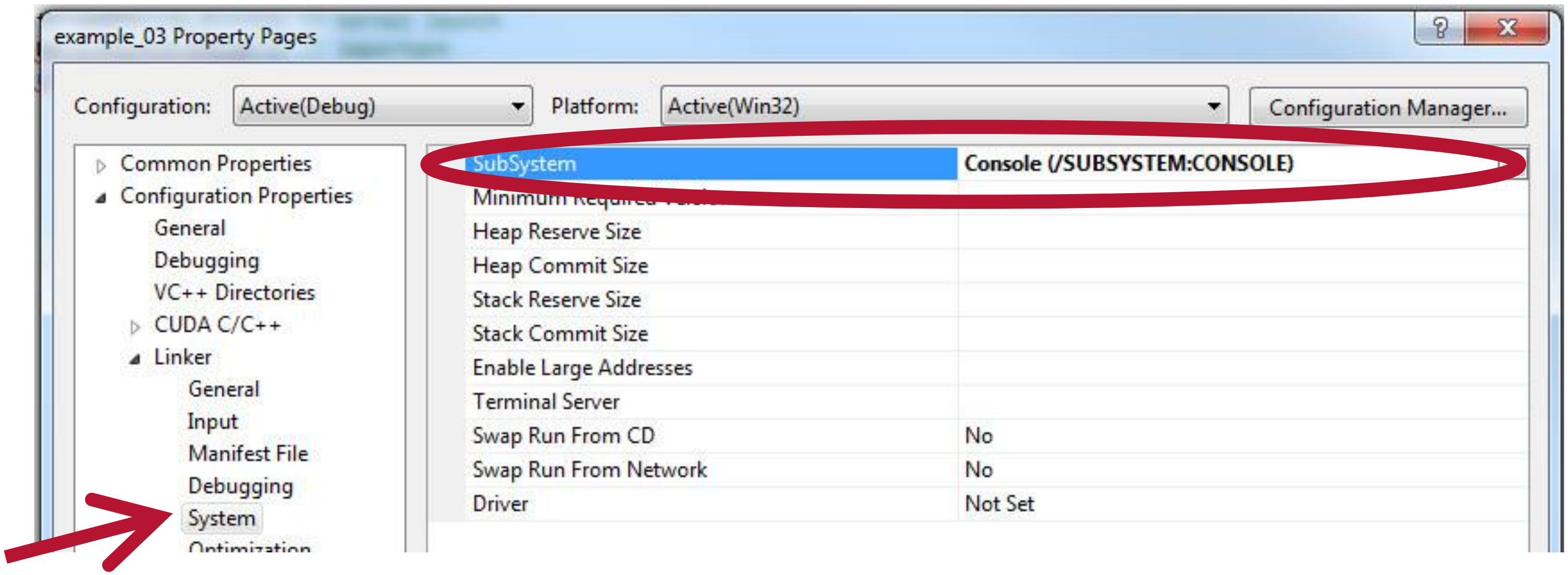
- `helloworld.cu.obj : error LNK2019: unresolved external symbol _cudaDeviceSynchronize@0 referenced in function _main`
- We need the Cuda runtime library `cudart.lib`
  - Project Properties



- Now recompile and run it!
  - Start application with CTRL+F5 to keep command prompt open

VS – Command Prompt

- I have an existing VS Cuda-Project and I don't see a command prompt at all
- Project Properties | Linker | System | SubSystem
  - change the value to CONSOLE



Syntax Highlighting

- Currently our code looks like this

```
__global__
void printFromGPU( void )
{
    printf( "hello world!\n" );
}
```

- How to change this:
  - See `%NVCUDASAMPLES_ROOT%\doc\syntax_highlighting\visual_studio_8\readme.txt`

```
__global__
void printFromGPU( void )
{
    printf( "hello world!\n" );
}
```

- VS still doesn‘t know the keywords and functions

VS – IntelliSense for CUDA

- We need the header files the declarations are found in

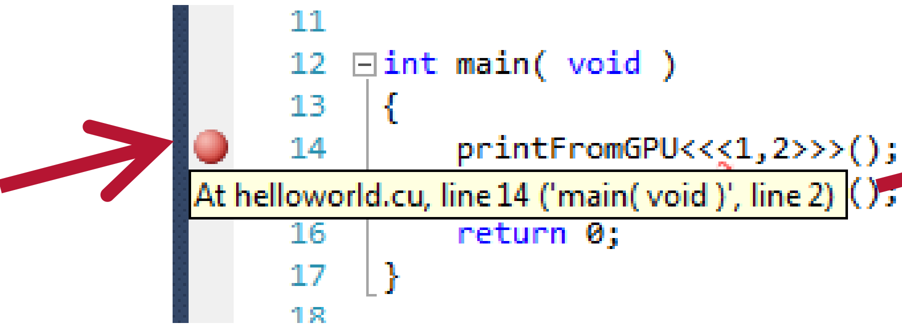
```
#include <device_launch_parameters.h> // cuda keywords
#include <cuda_runtime.h> // cuda runtime functions

...
//  your code
```

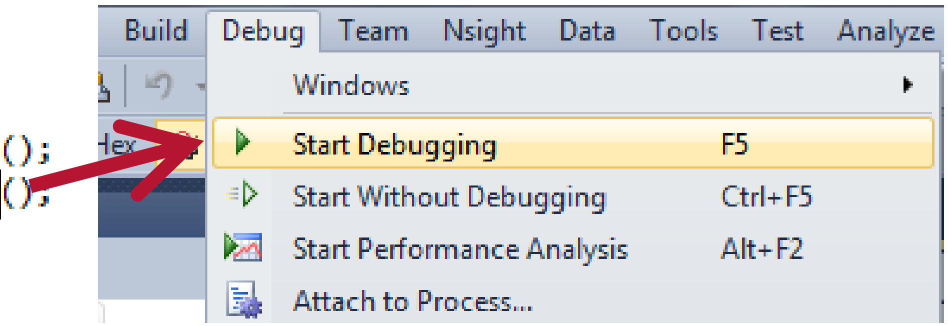
- Now we have a pretty good result

```
__global__
void printFromGPU( void )
{
    printf( "hello world!\n" );
}
```

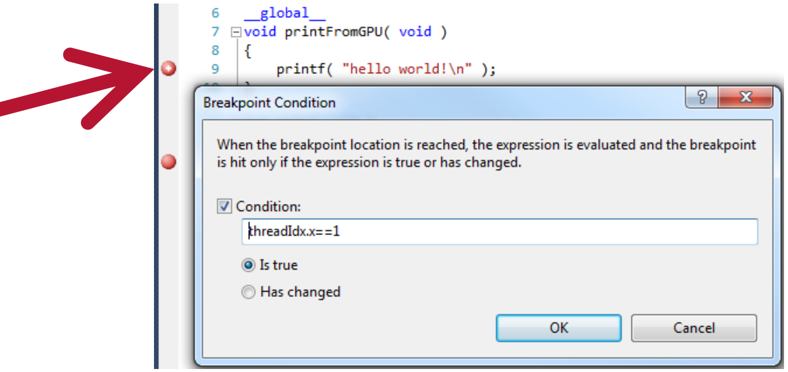
Debugging

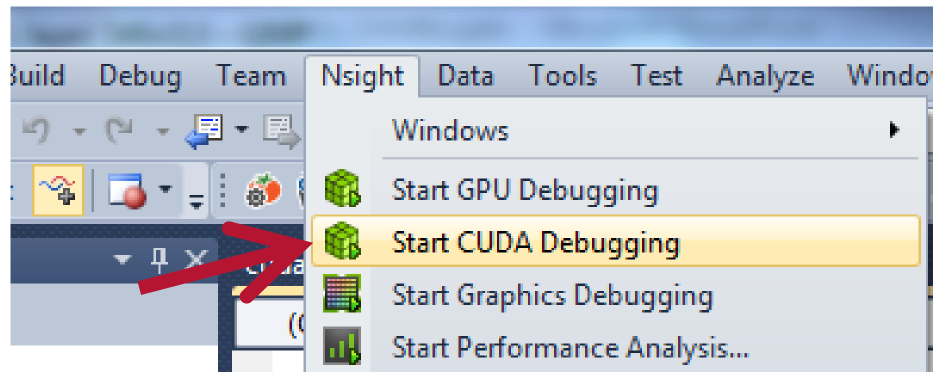
- Host code (C/C++)
  - Set your breakpoint(s) & Start Debugging





- Nsight: you need at least compute capability 3.0
  - Set conditional breakpoint(s) & start CUDA debugging





Exercises

- We provide Frameworks with Visual Studio Project files
- Some of them have graphical output
  - We use the freeglut library
- How to install freeglut
  - Download from http://freeglut.sourceforge.net/
  - Set system variable FREEGLUT_ROOT to *[installation folder] (e.g.* `C:\freeglut`)
  - Add `bin` folder to system variable PATH (e.g. `C:\freeglut\bin`)
  - Add include and library path, and freeglut.lib to your Project