Práctica 04. Query Device

Obtener las propiedades de la tarjeta NVidia del equipo actual.

U.A.Q. Fac. de Informática
Dra. Sandra Luz Canchola Magdaleno
Correo: sandra.canchola@uaq.mx
Dra. Reyna Moreno Beltrán
Correo: reyna.moreno@uaq.mx

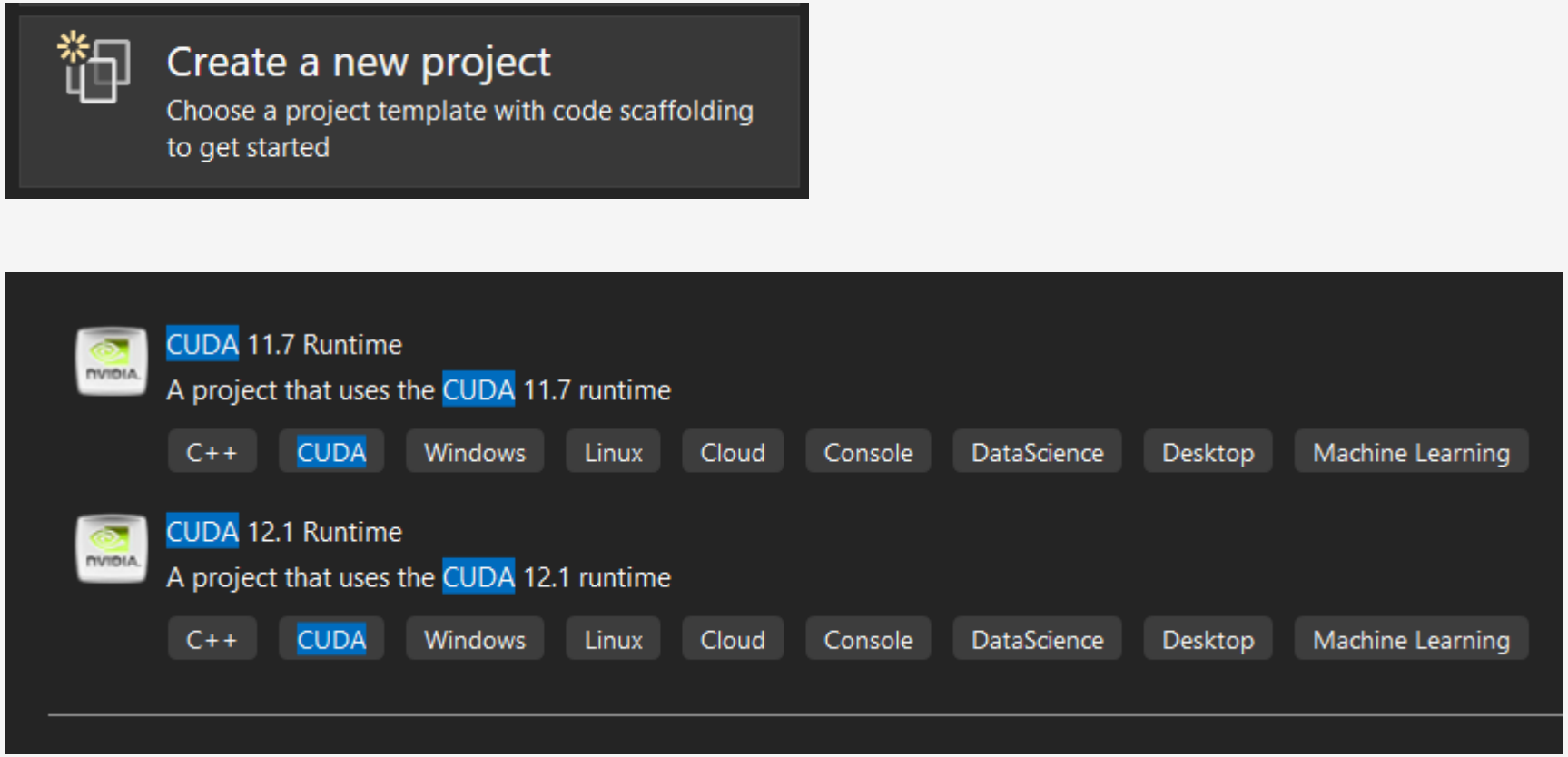
Conocer las capacidades de la tarjeta NVidia

Estructura cudaDeviceProp obtiene las propiedades de la tarjeta i-ésima, la sintaxis es:

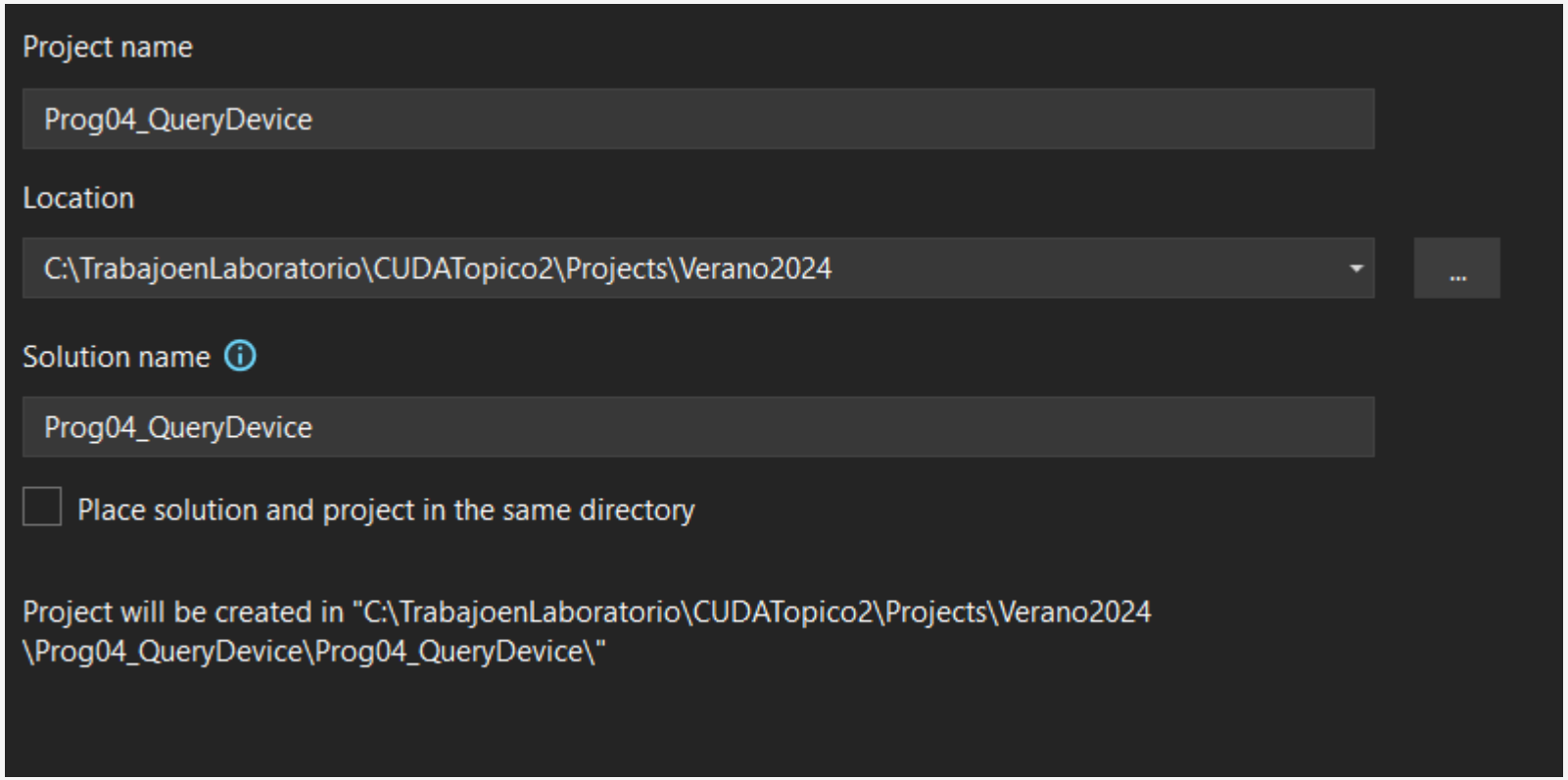
```
cudaDeviceProp devProp;

cudaGetDeviceProperties(&devProp, i);
```

Proyecto CUDA



Proyecto CUDA



Código

```
#include <iostream>
#include <string>
#include <cuda_runtime.h>
using namespace std;

string getDeviceArchitecture( cudaDeviceProp devProp ){
            string sign = "";
            switch( devProp.major )
            {
                        case 2:
                                    sign = "Fermi";
                                    break;
                        case 3:
                                    sign = "Kepler";
                                    break;
                        case 5:
                                    sign = "Maxwell";
                                    break;
                        case 6:
                                    sign = "Pascal";
                                    break;
                        case 7:
                                    sign = "Volta or Turing";
                                    break;
                        case 8:
                                    sign = "Ampere";
                                    break;
                        default:
                                    sign = "Unknown device type";
                                    break;
            }
            return sign;
}
```

Código

```
int getSPcores( cudaDeviceProp devProp ){
            int cores = 0;
            int mp = devProp.multiProcessorCount;
            switch( devProp.major )
            {
                        case 2:
                                    if( devProp.minor == 1 ) cores = mp * 48;
                                    else cores = mp * 32;
                                    break;
                        case 3:
                                    cores = mp * 192;
                                    break;
                        case 5:
                                    cores = mp * 128;
                                    break;
                        case 6:
                                    if( ( devProp.minor == 1 ) || ( devProp.minor == 2 ) ) cores = mp * 128;
                                    else if( devProp.minor == 0 ) cores = mp * 64;
                                    else cout << "Unknown device type\n";
                                    break;
                        case 7:
                                    if( ( devProp.minor == 0 ) || ( devProp.minor == 5 ) ) cores = mp * 64;
                                    else cout << "Unknown device type\n";
                                    break;
                        case 8:
                                    if( devProp.minor == 0 ) cores = mp * 64;
                                    else if( devProp.minor == 6 ) cores = mp * 128;
                                    else cout << "Unknown device type\n";
                                    break;
                        default:
                                    cout << "Unknown device type\n";
                                    break;
            }
            return cores;
}
```

Código

```
void printDevProp( int i )
{
            cudaDeviceProp devProp;
            cudaGetDeviceProperties( &devProp, i );
            cout << " - ASCII string identifying device: " << devProp.name << "\n";
            cout << " - Device architecture name: " << getDeviceArchitecture( devProp ) << "\n";
            cout << " - Major compute capability: " << devProp.major << "\n";
            cout << " - Minor compute capability: " << devProp.minor << "\n";
            cout << " - Number of multiprocessors on device: " << devProp.multiProcessorCount << "\n";
            cout << " - Number of CUDA cores: " << getSPcores( devProp ) << "\n";
            cout << " - Global memory available on device in bytes: " << devProp.totalGlobalMem << "\n";
            cout << " - Shared memory available per block in bytes: " << devProp.sharedMemPerBlock << "\n";
            cout << " - 32-bit registers available per block: " << devProp.regsPerBlock << "\n";
            cout << " - Warp size in threads: " << devProp.warpSize << "\n";
            cout << " - Maximum pitch in bytes allowed by memory copies: " << devProp.memPitch << "\n";
            cout << " - Maximum number of threads per block: " << devProp.maxThreadsPerBlock << "\n";
            for( int i = 0 ; i < 3 ; ++i )
                        cout << " - Maximum dimension " << i << " of the grid: " << devProp.maxGridSize[i] << "\n";
            for ( int i = 0 ; i < 3 ; ++i )
                        cout << " - Maximum dimension " << i << " of the block: " << devProp.maxThreadsDim[i] << "\n";
            cout << " - Clock frequency in kilohertz: " << devProp.clockRate << "\n";
            cout << " - Constant memory available on device in bytes: " << devProp.totalConstMem << "\n";
            cout << " - Number of asynchronous engines: " << devProp.asyncEngineCount << "\n";
            cout << " - Specified whether there is a run time limit on kernels: " << devProp.kernelExecTimeoutEnabled << "\n";
            cout << " - Alignment requirement for textures: " << devProp.textureAlignment << "\n";
}
```

Código

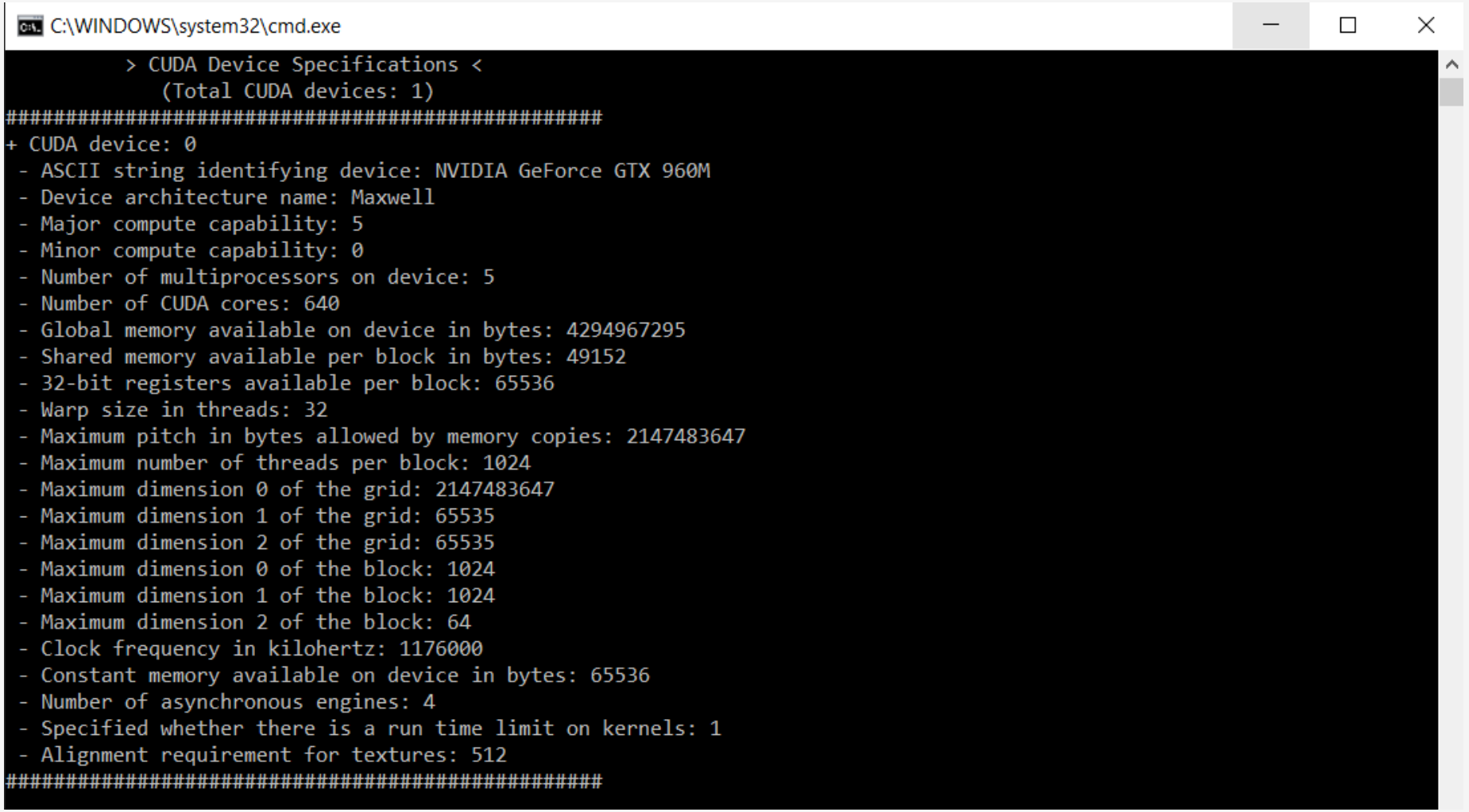
```
int main( int argc, char* argv[] )
{
            int devCount;
            cudaGetDeviceCount( &devCount );

            cout << "####################################################\n";
            cout << "\t  > CUDA Device Specifications <\n";
            cout << "\t     (Total CUDA devices: " << devCount << ")\n";

            for ( int i = 0 ; i < devCount ; ++i )
            {
                        cout << "####################################################\n";
                        cout << "+ CUDA device: " << i << "\n";
                        printDevProp( i );
                        cout << "####################################################\n\n";
            }

            system( "pause" );
            return 0;
}
```

Corrida


C:\WINDOWS\system32\cmd.exe
> CUDA Device Specifications <
(Total CUDA devices: 1)
##
+ CUDA device: 0
- ASCII string identifying device: NVIDIA GeForce GTX 960M
- Device architecture name: Maxwell
- Major compute capability: 5
- Minor compute capability: 0
- Number of multiprocessors on device: 5
- Number of CUDA cores: 640
- Global memory available on device in bytes: 4294967295
- Shared memory available per block in bytes: 49152
- 32-bit registers available per block: 65536
- Warp size in threads: 32
- Maximum pitch in bytes allowed by memory copies: 2147483647
- Maximum number of threads per block: 1024
- Maximum dimension 0 of the grid: 2147483647
- Maximum dimension 1 of the grid: 65535
- Maximum dimension 2 of the grid: 65535
- Maximum dimension 0 of the block: 1024
- Maximum dimension 1 of the block: 1024
- Maximum dimension 2 of the block: 64
- Clock frequency in kilohertz: 1176000
- Constant memory available on device in bytes: 65536
- Number of asynchronous engines: 4
- Specified whether there is a run time limit on kernels: 1
- Alignment requirement for textures: 512
##

Bibliografía

- Documentación **CUDA C++ Programming Guide** NVIDIA. 2024 https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html
- Sitio **CUDA Toolkit Documentation** NVIDIA, 2024. https://docs.nvidia.com/cuda/index.html
- Storti, Duane; Yurtoglu, Mete. **CUDA for Engineers:An Introduction to High-Performance Parallel Computing**. Addisson Wesley. 2015.
- Cheng, John; Grossman, Max; McKercher. **Professional CUDA C Programming**. Edit. Wrox. 2014.
- Sanders, Jason; Kandrot, Edward. **CUDA by Example:An Introduction to General-Purpose GPU Programming**. Addisson Wesley. 2011.
- Kirk, David; Hwu, Wen-mei. **Programming Massively Parallel Processors: A Hands-on Approach**. Elsevier. 2010.

Gracias por su atención

DRA.
Sandra Luz
CANCHOLA
MAGDALENO

U.A.Q. Fac. de Informática
Campus Juriquilla

Dra. Sandra Luz Canchola Magdaleno
sandra.canchola@uaq.mx
Cel. 442-1369270

Dra. Reyna Moreno Beltrán
reyna.moreno@uaq.mx