NAME

nvidia::ml − Perl bindings to NVML, the NVIDIA Management Library

SYNOPSIS

```
use nvidia::ml qw(:all);

nvmlInit();

($ret, $version) = nvmlSystemGetDriverVersion();
die nvmlErrorString($ret) unless $ret == $nvidia::ml::bindings::NVML_SUCCESS;
print "Driver version: " . $version . "\n";

($ret, $count) = nvmlDeviceGetCount();
die nvmlErrorString($ret) unless $ret == $nvidia::ml::bindings::NVML_SUCCESS;

for ($i=0; $i<$count; $i++)
{
    ($ret, $handle) = nvmlDeviceGetHandleByIndex($i);
    next if $ret != $nvidia::ml::bindings::NVML_SUCCESS;

    ($ret, $speed) = nvmlDeviceGetFanSpeed($handle);
    next if $ret != $nvidia::ml::bindings::NVML_SUCCESS;
    print "Device " . $i . " fan speed: " . $speed . "%\n";

    ($ret, $info) = nvmlDeviceGetMemoryInfo($handle);
    next if $ret != $nvidia::ml::bindings::NVML_SUCCESS;
    $total = ($info->{"total"} / 1024 / 1024);
    print "Device " . $i . " total memory: " . $total . " MB\n";
}

nvmlShutdown();
```

DESCRIPTION

Provides a Perl interface to GPU management and monitoring functions.

This is a wrapper around the NVML library. For information about the NVML library, see the NVML documentation.

REQUIRES

Exporter

EXPORTS

This module has no exports. To add functions and constants to your namespace use: use nvidia::ml qw(:all);

METHODS

See EXPORTS and NVML documentation. Perl methods wrap NVML functions, implemented in a C shared library. The functions use is the same with the following exceptions:

Perl methods accept the input arguments of the C function it wraps only. All C function output parameters are returned after the return code, left to right

```
C:
nvmlReturn_t nvmlDeviceGetEccMode(nvmlDevice_t device,
                                  nvmlEnableState_t *current,
                                  nvmlEnableState_t *pending);
Perl:
($ret, $current, $pending) = nvmlDeviceGetEccMode($device);
```

Perl handles string buffer creation

```
C:
nvmlReturn_t nvmlSystemGetDriverVersion(char* version,
                                        unsigned int length);
Perl:
($ret, $version) = nvmlSystemGetDriverVersion();
```

C structs are converted to Perl hashes, nested as needed

```
C:
nvmlReturn_t DECLDIR nvmlDeviceGetMemoryInfo(nvmlDevice_t device,
                                             nvmlMemory_t *memory);
typedef struct nvmlMemory_st {
    unsigned long long total;
    unsigned long long free;
    unsigned long long used;
} nvmlMemory_t;
Perl:
($ret, $memory) = nvmlDeviceGetMemoryInfo($device);
print "Total memory " . $memory->{"total"} . "\n";
print "Used memory " . $memory->{"used"} . "\n";
print "Free memory " . $memory->{"free"} . "\n";
```

VARIABLES

See EXPORTS and NVML documentation. All NVML constants and enums are exposed.

COPYRIGHT

Copyright (c) 2011−2012, NVIDIA Corporation. All rights reserved.

LICENSE

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of the NVIDIA Corporation nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS ‘‘AS IS’’ AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.