Add GPU support to the Vc vectorization library

Jan Stephan

Background and experience

- Computer Science at TU Dresden
- 6th semester
- student assistant since May 2012
- 3D reconstruction system for X-ray imaging
- implementation in C++ and OpenCL
- practical courses in the field of operating systems
- ISTQB® Certified Tester, Foundation Level

The Vc vectorization library

- free software library
- eases explicit vectorization of C++ code
- What does this mean?

The idea of SIMD

- Single Instruction, Multiple Data
- operate on a “vector” of data with a single instruction
- example: x86 SSE registers
- 128bit-registers that can hold four (32bit) floats
- a single instruction to modify all of them
- Vc abstracts away the platform specific parts

Code examples - sqrt

Naive implementation:

```
float a[vectorSize], b[vectorSize];

// initialize here

for(size_t i = 0; i < vectorSize; ++i)
    b[i] = std::sqrt(a[i]);
```

x86 SSE implementation:

```
float a[vectorSize], b[vectorSize];
__m128 rA, rB;

// initialize here

// declare pointers
float *pA = a;
float *pB = b;

for(size_t i = 0; i < vectorSize; i += 4)
{
    rA = _mm_load_ps(pA);
    rB = _mm_sqrt_ps(rA);
    _mm_store_ps(pB, rB);
    // adjust pointers
    pA += 4;
    pB += 4;
}
```

Code examples - sqrt

Vc implementation:

```
Vc::Memory<Vc::float_v, vectorSize> a;
Vc::Memory<Vc::float_v, vectorSize> b;

// initialize here

for(size_t i = 0; i < a.vectorsCount(); ++i)
    b.vector(i) = Vc::sqrt(a.vector(i));
```

Goals and challenges

- project goal: port the library to NVIDIA's CUDA platform
- challenges:
  - be performant (copying to GPU will introduce additional overhead)
  - understand all the template magic in the sources

Questions?

Sources

M. Kretz, *Extending C++ for Explicit Data-Parallel Programming via SIMD Vector Types*, Frankfurt 2015