Updating Applications from Goopax-3 to Goopax-4

With Goopax 4.0, we introduce a few changes. In particular, the syntax of loops is now different. As a result, a few adjustments need to be done to existing code. By following this guide, we hope that applying the necessary changes is relatively straightforward. It is generally safe to just run the compiler and fix compiler errors. Using the old loop syntax with Goopax-4 will always induce compiler errors and never result in wrong code.

What has changed

gpu_for loops:

- In all types of gpu_for loops, the loop body is now provided as a lambda function.
- The comparison operator is now specified as template argument. It can be one of `std::less<>`, `std::less_equal<>`, `std::greater<>`, `std::greater_equal<>`, `std::equal_to<>`, `std::not_equal_to<>`. The default is `std::less<>`.

for_each

The `std::for_each` loop has been overloaded to support GPU types.

C++ standard

The minimum required C++ standard has been increased to C++-14.

CPU/GPU code intermingling:

One of the main rationale behind the new loop syntax is the ability to mix CPU code and GPU code. It is now much easier to write functions that can be called both from CPU code and from GPU code. Consider the following function:

```
template <typename T>
void fill(T& v, typename T::value_type value)
{
    gpu_for_global(v.begin(), v.end(), [&](typename T::iterator p)
    {
        *p = value;
    });
}
```

This function can be called from CPU code, with `v` being a `std::vector`, or from GPU code, with `v` being a `goopax::resource`.

The type of the loop variable is deduced from the type of the lambda function that is passed to the loop. If the loop variable is of CPU type, the loop is treated as an ordinary C++ loop and can be called from a normal C++ function. If the loop variable is of GPU type, the loop is considered a GPU loop and may only be called from within a GPU kernel. Starting from version 4.0.1, gpu_if clauses can also be used in CPU code, if the conditional variable can be converted to bool.

Update instructions from Goopax-3 to Goopax-4:

Updating existing Goopax-3 code to Goopax-4, `gpu_for` loops should be changed as follows:

```
// Goopax-3                                  // Goopax-4

// Index-based loops:
gpu_for(gpu_int k=0, 10)                     gpu_for(0, 10, [&](gpu_int k)
{                                            {
    ...                                          ...
}                                            });

// Loops parallelized over threads in workgroup:
gpu_for_local(gpu_int k=0, 10)               gpu_for_local(0, 10, [&](gpu_int k)
{                                            {
    ...                                          ...
}                                            });

// Loops parallelized over all threads:
gpu_for_global(gpu_int k=0, 10)              gpu_for_global(0, 10, [&](gpu_int k)
{                                            {
    ...                                          ...
}                                            });

// Loops parallelized over workgroups:
gpu_for_group(gpu_int k=0, 10)               gpu_for_group(0, 10, [&](gpu_int k)
{                                            {
    ...                                          ...
}                                            });

// pointer-based loops:
gpu_for(gpu_type<float*> p=a.begin(),        gpu_for(a.begin(), a.end(),
                           a.end())                  [&](gpu_type<float*> p)
{                                            {
    *p=17;                                       *p=17;
}                                            });

// Loops with step size
gpu_for(gpu_int k=0, 10, 2)                  gpu_for(0, 10, 2, [&](gpu_int k)
{                                            {
    ...                                          ...
}                                            });

// Loops with non-default comparison operator
gpu_for(gpu_int k=0, 10, "<=")               gpu_for<std::less_equal<>>(0, 10, [&](gpu_int k)
{                                            {
    ...                                          ...
}                                            });
```