OpenEXR Project Ideas

Potential projects for the Google Summer of Code or beyond.

Explore Other Compression Schemes.

- OpenEXR currently supports lossless compression through zlib, but other more modern compression schemes exist. We would like to explore adding additional options for compressing with other utilities, such was Zstandard. To do this properly, we also need to assemble a standard set of images to use as benchmarks, and also develop some basic performance metrics in order to compare compression/decompression time across different options.

Performance Metric Suite

- OpenEXR needs a mechanism for quantifying read/write and compress/decompress times.

A Fast Header Read

- Extend the OpenEXR API with a way to read just the header attributes you request, and nothing else, for efficiency when applications need only the header information but no pixel data.

Add a Part Type That is Only Metadata

- An image part that holds only metada with no pixel data could be useful for managing metadata in motion picture pipelines.

Add support for bfloat16

- https://en.wikipedia.org/wiki/Bfloat16_floating-point_format
- Adding a new pixel type might be tricky, and may present backwards compatibility issues.

Add Support for Sorting of Attributes

- Currently, attributes are written and read in alphabetical order in the file, and stored alphabetically internally.
- But sometimes it would be convenient to organize and present them in the API, or GUI's, in a logical, non-alphabetical order.
- The solution needs some investigation, but it might involve changing the internal attribute storage mechanism, or leaving the internal representation alone and adding an "order" attribute to store the preferred order.

Convert boost::python Imath Bindings to pybind11

- Eliminate Imath's dependency on boost/boost_python by re-implementing the bindings using pybind11.

Add a New Spectral Attribute Type

- Follow up on Alban Fichet's spectral image storage presentation by implementing a custom attribute to hold the spectra: https://hal.inria.fr/hal-03252797

Switch C++ API to Use the New C Core

- Retrofit the existing C++ API to use the new thread-safe C API underneath