HOSTED BY /* ACADEMY SOFTWARE FOUNDATION

OpenEXR

2021 Project Update

OpenEXR: Agenda

Open Source Days 2021

- Year in Review
- Technical Overview of OpenEXR/Imath v3.1 (Kimball Thurston)
- Spectral Image Data in EXR (Alban Fichet)

OpenEXR: Project Mission

The goal of the OpenEXR project is to keep the EXR format reliable and modern and to maintain its place as the preferred image format for entertainment content creation.

- **Robustness**, **reliability**, **security**
- Backwards **compatibility**, data longevity
- **Performance** - read/write/compression/decompression time
- **Simplicity**, ease of use, maintainability
- Wide **adoption**, multi-platform support - Linux, Windows, macOS, and others

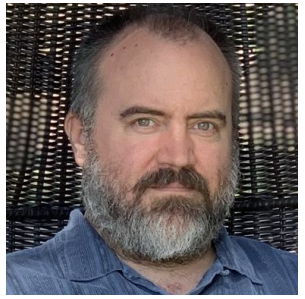
OpenEXR: Technical Steering Committee

Open Source Days 2021

Cary Phillips
Industrial Light & Magic

Christina Tempelaar-Lietz
Epic Games

Joseph Goldstone
ARRI, Inc



Kimball Thurston
Weta Digital, Ltd

Larry Gritz
Sony Pictures Imageworks

Nick Porcino
Pixar Animation Studios

Peter Hillman
Weta Digital, Ltd

Rod Bogart
Epic Games

OpenEXR: Contact Us

Open Source Days 2021

Email: **openexr-dev@lists.aswf.io**

Message archive: https://lists.aswf.io/g/openexr-dev/topics

Slack: #openexr

Repo: https://github.com/AcademySoftwareFoundation/openexr/issues

Calendar: https://lists.aswf.io/g/openexr-dev/calendar
Every other Thursday, 1:30pm Pacific Time

OpenEXR: Year in Review

Open Source Days 2021

- Security patches to v2.4 and v2.5
- Release of v3.0

Theme...

- *Security (buffer overrun, int overflow, etc)*
- *Build/CMake Improvements*
- *Reorganize/modernize*

OpenEXR: Standard Optional Attributes

Open Source Days 2021

2020 Proposal:

Update OpenEXR default metadata to be consistent with ACES Container spec.

VES Tech Committee project:

- What data is most important?
- Names, definitions, units?
- Who is the keeper of the standard across stakeholders?
- OpenEXR vs other containers (e.g. OTOI)?
- ves-tech-camera-metadata@googlegroups.com

Imath 3.1

Optimized half-to-float/float-to-half conversion:

- F16C intrinsics if available (gcc/clang -mf16c)
- Lookup-table-free alternative
- C-language API

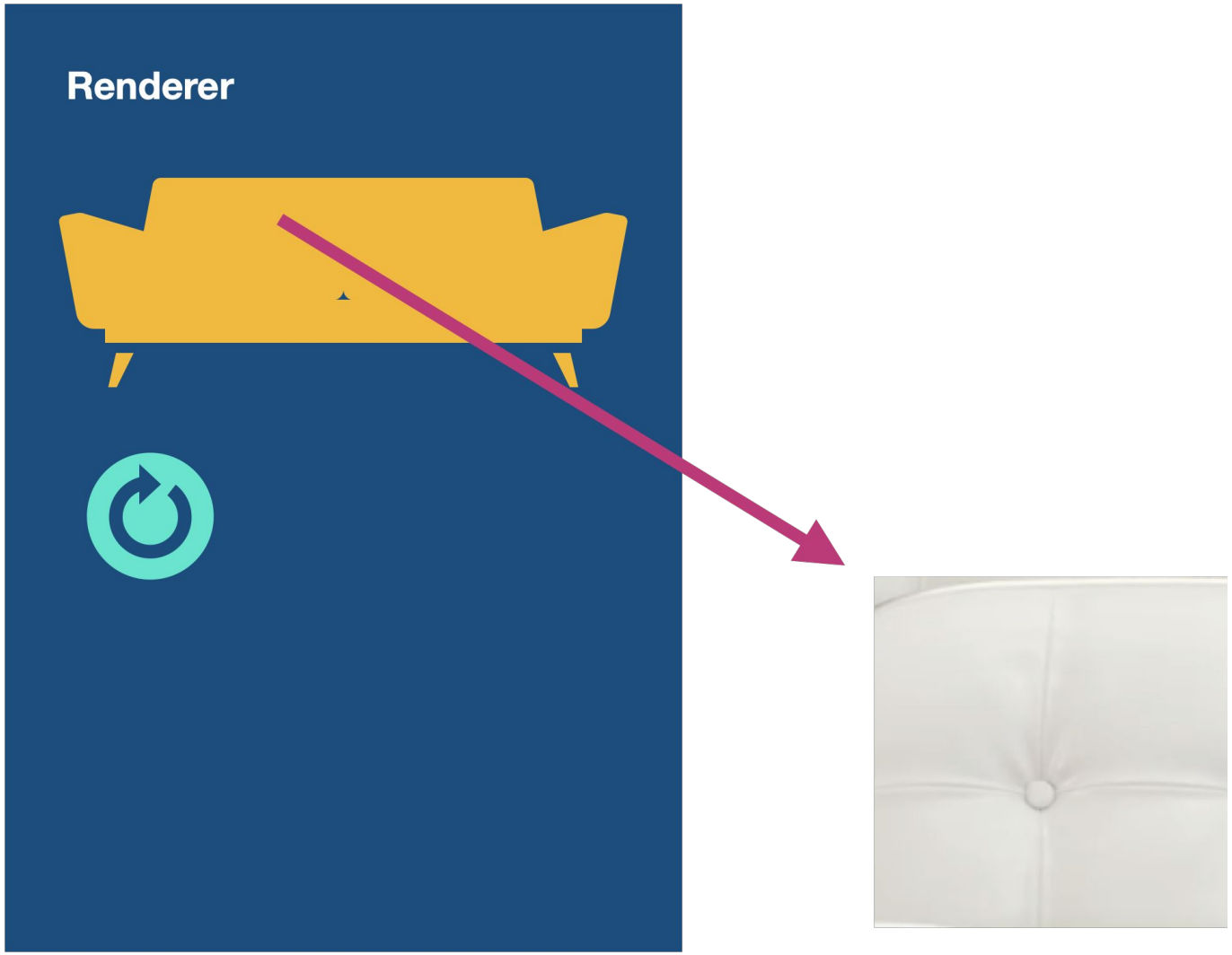
OpenEXR 3.1:

Open Source Days 2021

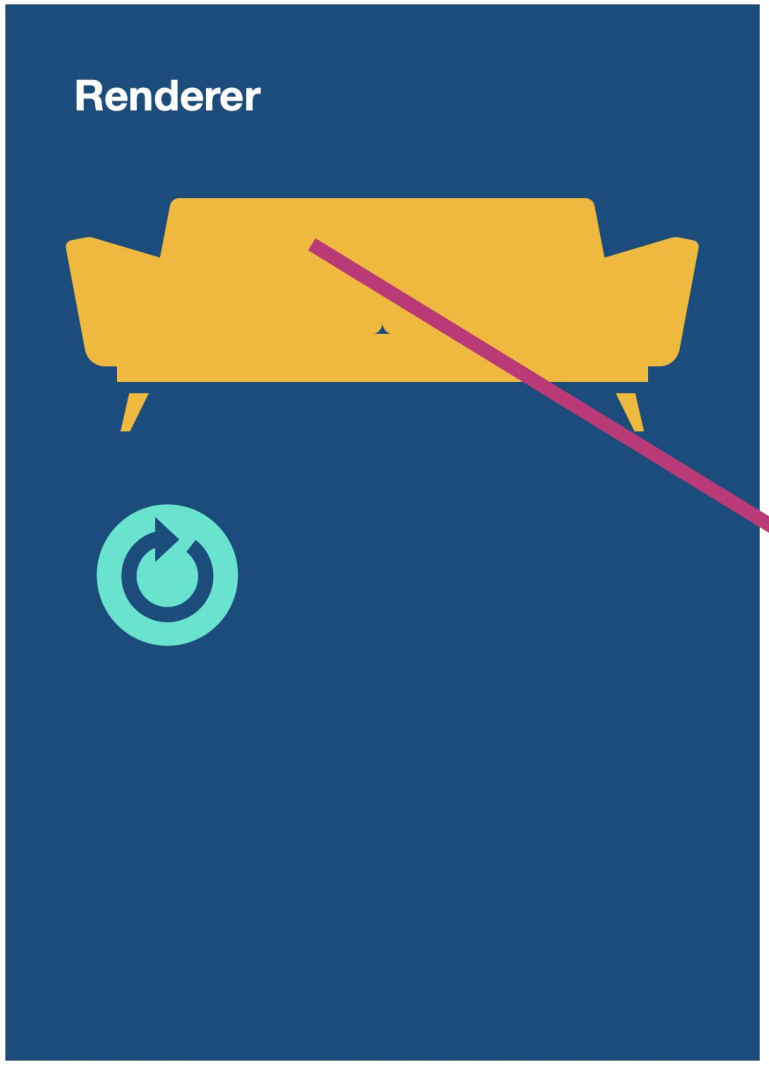
libOpenEXRCore:

- Thread-safe, non-blocking
- Custom unpacking of EXR data (on the GPU)
- C API
- Extension (no changes to existing API)

Thread Issue in Review

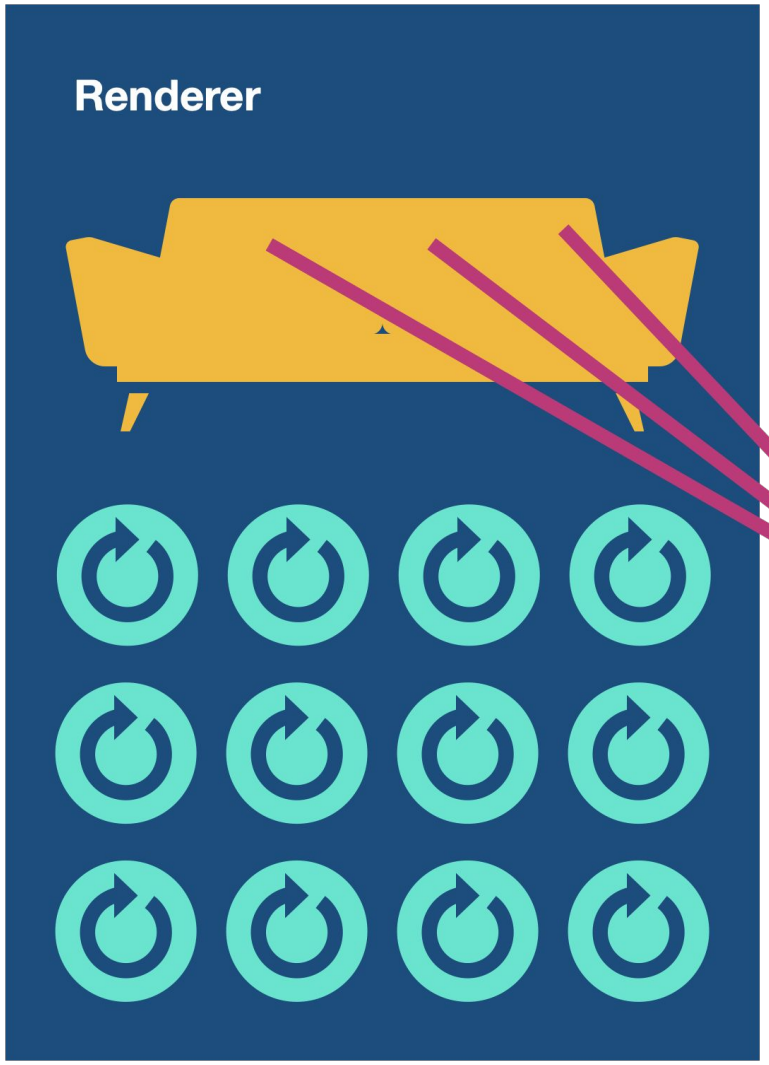


Thread Issue in Review




```
fileHandle->setFrameBuffer( ... );

fileHandle->readTile( tileX, tileY, mipLevel );
```

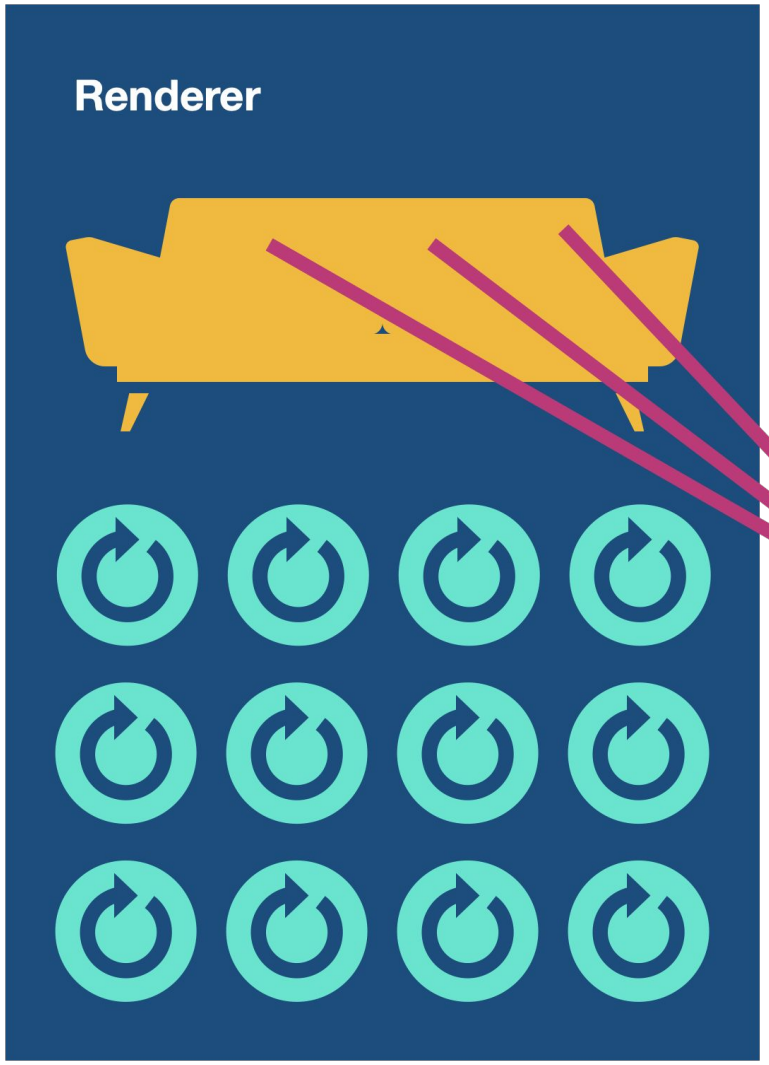
Thread Issue in Review



fileHandle->setFrameBuffer(...);

fileHandle->readTile(tileX, tileY, mipLevel);

Thread Issue in Review



fileHandle->setFrameBuffer(...);

fileHandle->readTile(tileX, tileY, mipLevel);

Thread Issue in Review (Performance)


```
fileHandle->readTile( tileX, tileY, mipLevel );
```

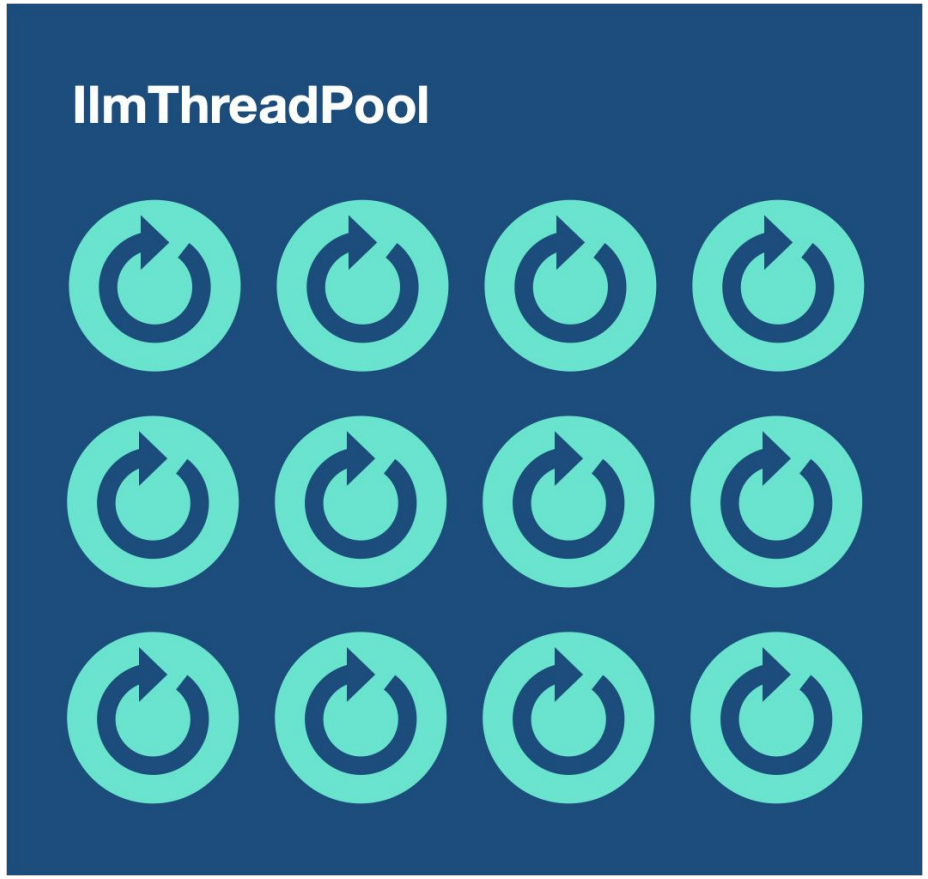
Thread Issue in Review (Performance)


```
fileHandle->readTile( tileX, tileY, mipLevel );
```

Read

(Decompress)

Unpack



Thread Issue in Review (Performance)

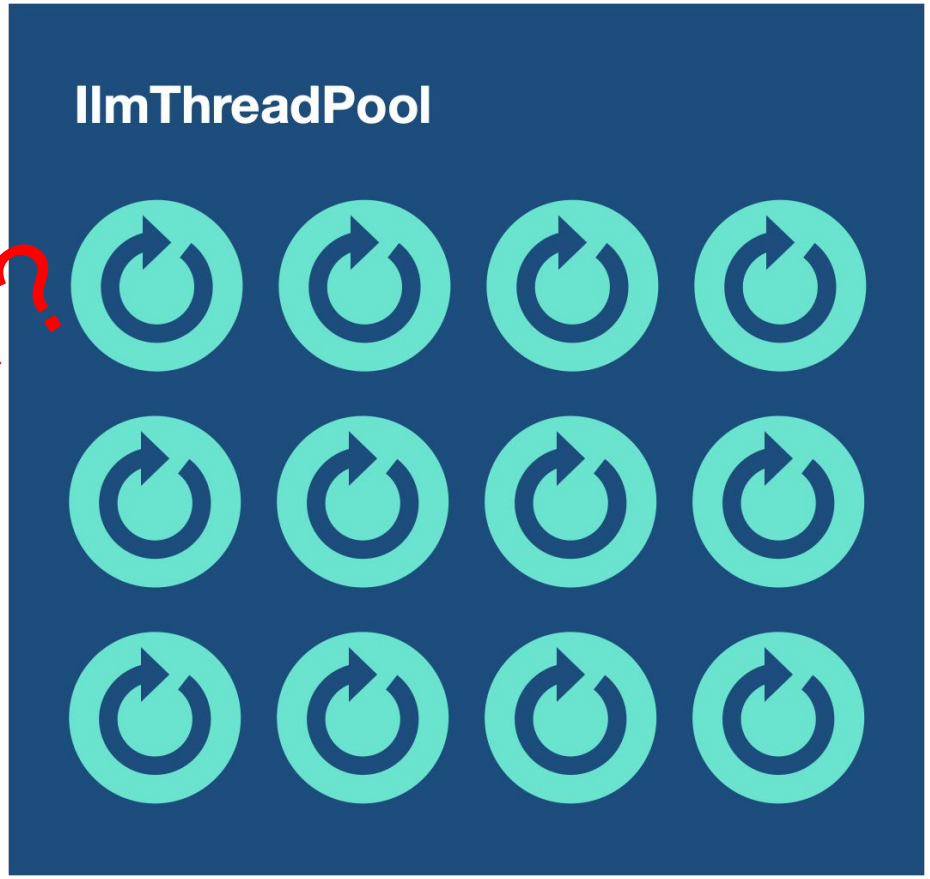

```
fileHandle->readTile( tileX, tileY, mipLevel );
```

Read

(Decompress)

Unpack

FAKE NEWS???



Performance

Open Source Days 2021

Current:

```
readChunkOffsetTable( n, ... )
{
    for ( int i = 0; i < n; ++i )
        Xdr::read( is, offsets[i] );
}
```

New:

```
readChunkOffsetTable( n, ... )
{
    pread( fd, offsets, n * sizeof(int64_t) );
    swap_if_necessary( ... );
}
```

Performance: You tell us

Open Source Days 2021

Existing Library:

Stream based

Avoids seeking at all cost, extra ifs even

But Core: Know the file position, be the file position

Posix compliant systems: pread

Win32: ReadFile(handle, ..., &overlapped);

Reading

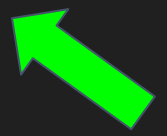
Open Source Days 2021

```
exr_context_t fcontext;
exr_context_initializer_t cinit = EXR_DEFAULT_CONTEXT_INITIALIZER;

cinit.error_handler_fn = &myerrorhandler;
// allocation?
cinit.alloc_fn = &mymalloc;
cinit.user_data = &somedata;
cinit.read_fn = &mycustomread;

exr_result_t rv = exr_start_read( &fcontext, "somefile.exr", &cinit );
if (rv != EXR_ERR_SUCCESS)
  goto fail;
```

Reading

Open Source Days 2021

```
exr_context_t fcontext;
exr_context_initializer_t cinit = EXR_DEFAULT_CONTEXT_INITIALIZER;

cinit.error_handler_fn = &myerrorhandler;
// allocation?
cinit.alloc_fn = &mymalloc;
cinit.user_data = &somedata;
cinit.read_fn = &mycustomread;

exr_result_t rv = exr_start_read( &fcontext, "somefile.exr", &cinit );
if (rv != EXR_ERR_SUCCESS)
  goto fail;
```



~55µsec vs. ~400µsec

Attributes

In Core, attribute order is preserved.

Future optimization?

Query the list in either original or sorted order

Attributes

Open Source Days 2021

```
exr_attr_box2i_t       dw;

exr_get_data_window (fcontext, 0, &dw);

int foo;

exr_attr_get_int (fcontext, 0, "myspeciallattr", &foo );
```

Attributes


```
exr_attr_box2i_t      dw;

exr_get_data_window (fcontext, 0, &dw);

int foo;

exr_attr_get_int (fcontext, 0, "myspeciallattr", &foo );
```

Attributes


```
exr_attr_box2i_t      dw;

exr_get_data_window (fcontext, 0, &dw);

int foo;

exr_attr_get_int (fcontext, 0, "myspeciallattr", &foo );
```

Attributes (Generic)

```
EXR_EXPORT exr_result_t exr_get_attribute_count (exr_const_context_t ctxt,
int part_index, int32_t* count);

exr_get_attribute_by_index (...);
exr_get_attribute_by_name (...);
exr_get_attribute_list (...);


exr_register_attr_type_handler (...);
```

Getting at the Data: Pipelines

Separate Context Data Structure from File (GPU)

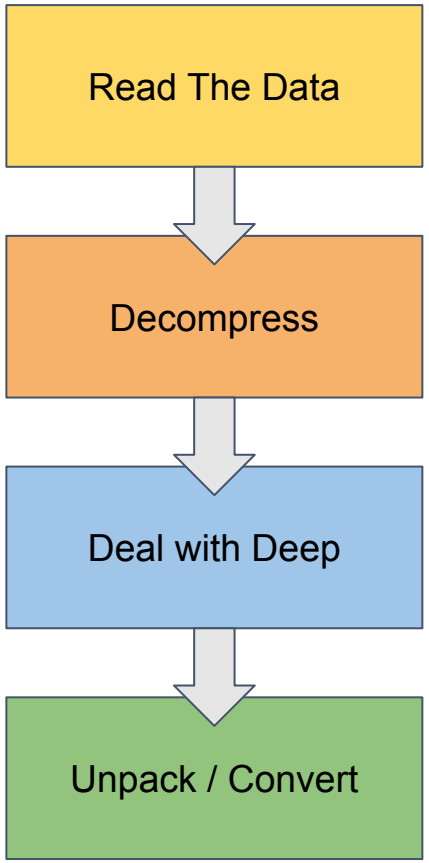
- Custom allocators

Provide pointers to populate (NO CACHE)

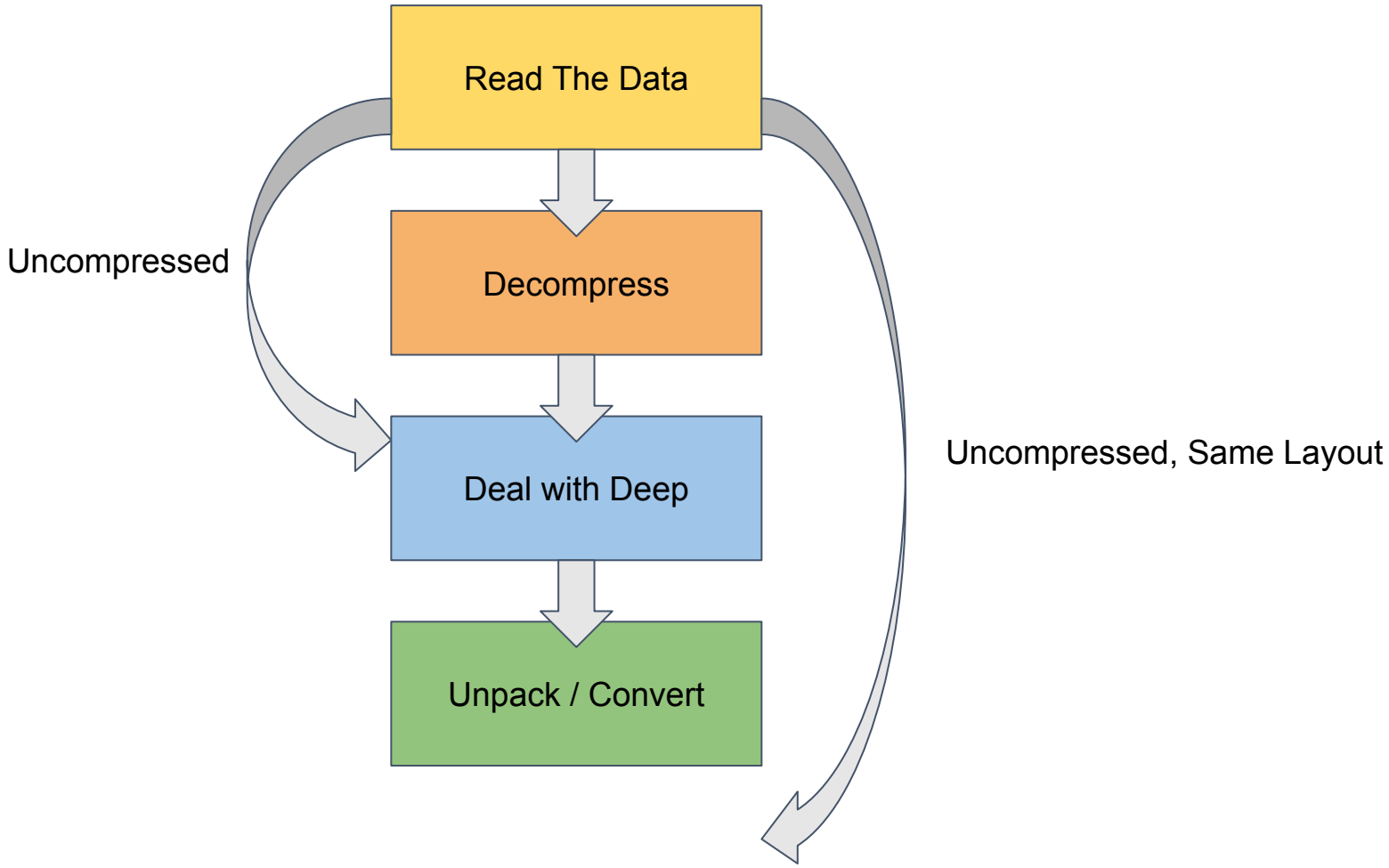
- Zero-copy-ish

Decode Pipeline can be re-used to avoid re-allocs

Getting at the Data: Pipelines



Getting at the Data: Pipelines



Getting at the Data

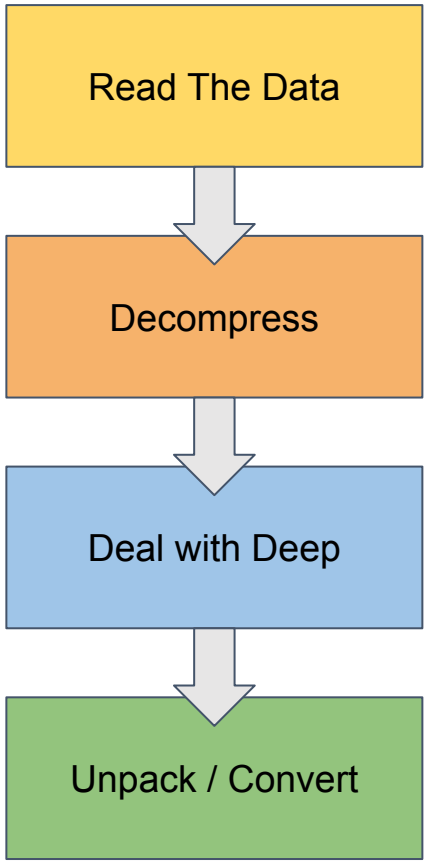
Open Source Days 2021

```
exr_get_scanlines_per_chunk (f, 0, &scansperchunk);
for (int y = dw.min.y; y <= dw.max.y; y += scansperchunk) {
    exr_read_scanline_chunk_info (f, 0, y, &cinfo);
    exr_decoding_initialize (f, 0, &cinfo, &decoder);
    // exr_decoding_update (f, 0, &cinfo, &decoder);
    for (int c = 0; c < decoder.channel_count; ++c) {
        decoder.channels[c].decode_to_ptr      = (uint8_t*) ptr;
        decoder.channels[c].user_pixel_stride = pixelstride;
        decoder.channels[c].user_line_stride  = linestride;
    }
    exr_decoding_choose_default_routines (f, 0, &decoder);
    exr_decoding_run (f, 0, &decoder);
}
```

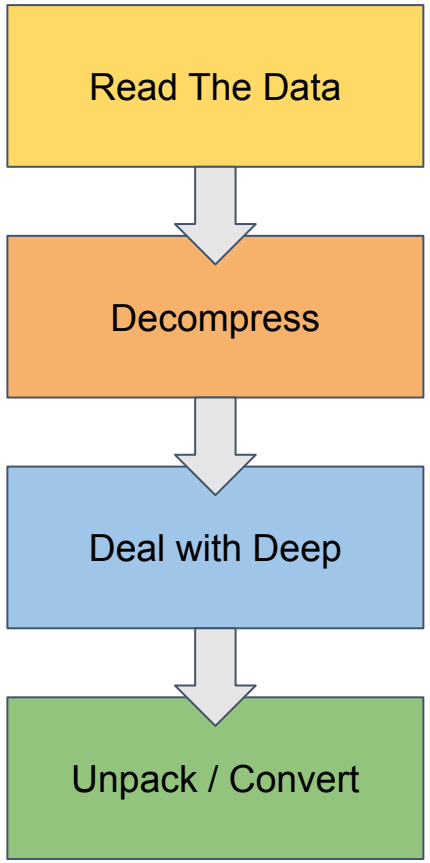
File Cleanup

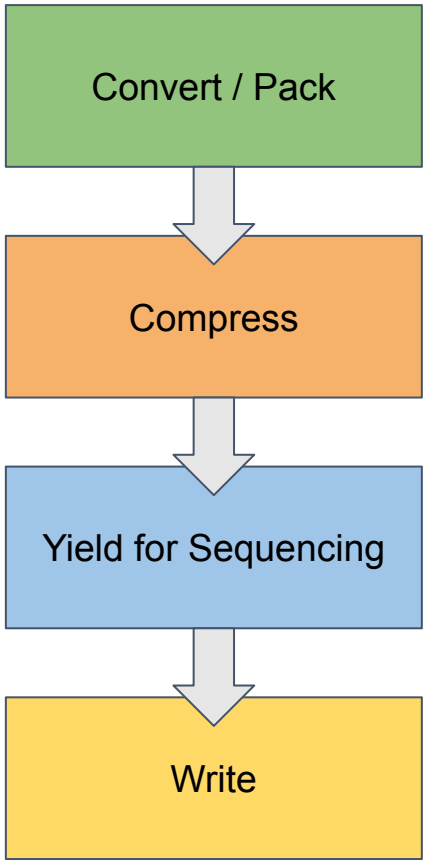

```
exr_decoding_destroy (f, 0, &decoder);

exr_finish( &f );
```

What about Writing?



What about Writing?





Some Notes

Removes threading overhead, makes caller responsible

No hidden data

- No Cache
- No mutexes

Assumes use case for overlapped I/O (cloud storage failure?)

- I/O customizable

Purely additive at this point (just a new C api)

What's Next?

Open Source Days 2021

- Finish some missing bits:
  - DWA* compression
  - Deep packing (writing is fine, but no default pack routine)

- Validate GPU unpacking capabilities (please help!?!?)

- Start replacing internals of C++
  - ABI change, so 4.0
  - Add API (deprecate the SetFrameBuffer / read calls?)

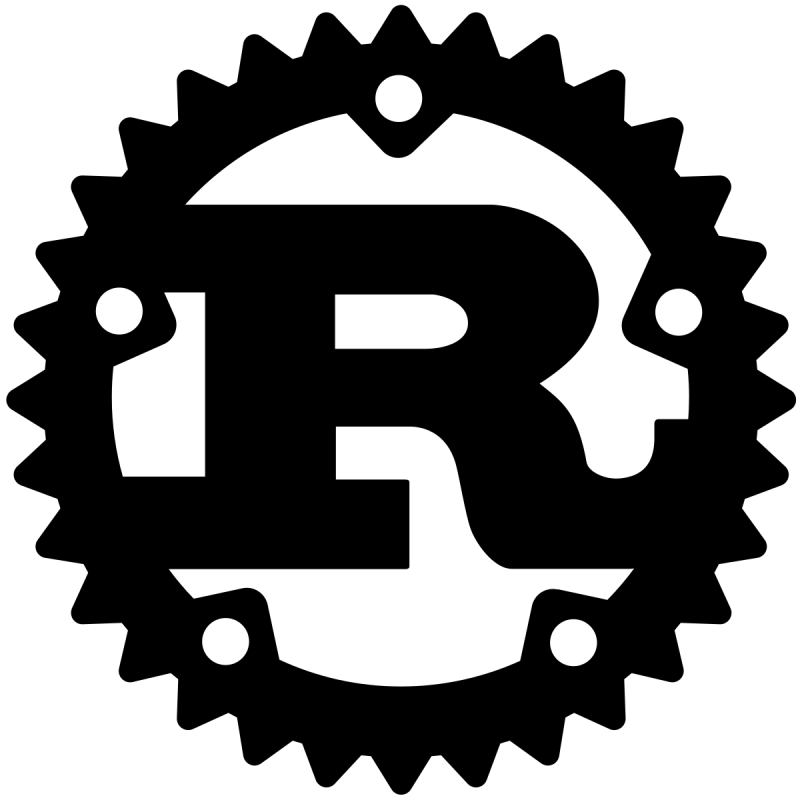
Rust Bindings

Open Source Days 2021

Core API designed to be compatible

- Mutability
- Data lifetimes

But: C++ bindings! And not just for EXR...



Discussion: Spectral Image Data in EXR Files

Alban Fichet, Charles University, Prague

Open Source Days 2021

Open Source Days 2021

Spectral Image Data

Alban Fichet, Charles University, Prague