OPENCL ECOSYSTEM UPDATES

Developer UX at the forefront

TABLE OF CONTENTS

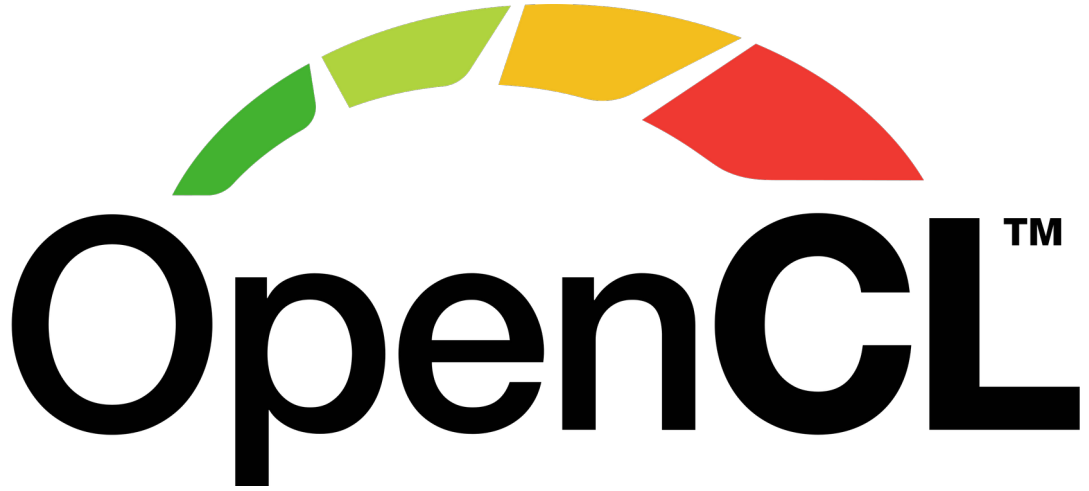
STREAM
High Performance Computing

1ST PARTY OPENCL-SDK

Where it all begins

OPENCL 3.0

STREAM
High Performance Computing



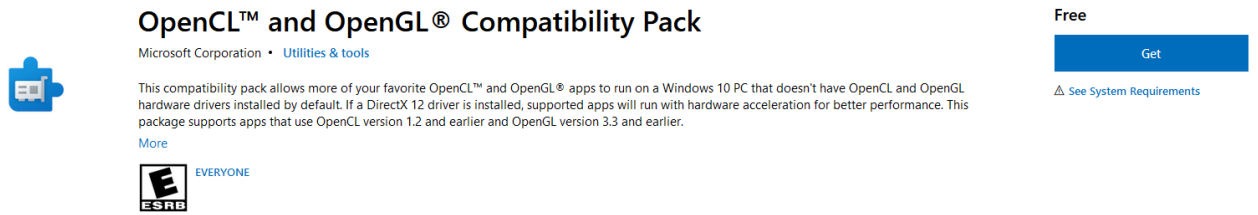
- Re-imagination of the API
- Builds on top of the last widely adopted version, 1.2
- Adopts the Vulkan-like „slim core API with everything else extensions” approach
- Lowers the bar for adoption
  - Newcomers
  - Solves issues for existing adopters

1ST PARTY OPENCL-SDK

OPENCL 3.0 ECOSYSTEM REBOOT

STREA
High Performance Computing

- New ICD Layers capability allows augmenting/modifying runtime behavior
  - Add missing features, validate correctness, trace, profile, etc
- OpenCL SDK serves as a „one-stop shop” for devs
  - Initial batch of native samples & utilities awarded to Stream
    - Coming to a GitHub near you
- Defines the semantics of layered implementations to fill in the gaps of platform support
  - Most notably OpenCLOn12, enabling OpenCL across the entire Windows ecosystem, regardless of vendor support.



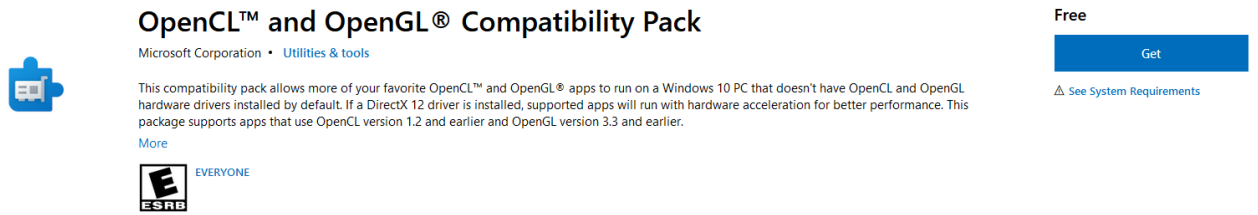
https://www.microsoft.com/en-us/p/opencl-and-opengl-compatibility-pack/9nqpsl29bfff

1ST PARTY OPENCL-SDK

OPENCL 3.0 ECOSYSTEM REBOOT

STREAM High Performance Computing

- New ICD Layers capability allows augmenting/modifying runtime behavior
  - Add missing features, validate correctness, trace, profile, etc
- OpenCL SDK serves as a „one-stop shop” for devs
  - Initial batch of native samples & utilities awarded to Stream
    - Coming to a GitHub near you
- Defines the semantics of layered implementations to fill in the gaps of platform support
  - Most notably OpenCLOn12, enabling OpenCL across the entire Windows ecosystem, regardless of vendor support.



https://www.microsoft.com/en-us/p/opencl-and-opengl-compatibility-pack/9nqpsl29bfff

1ST PARTY OPENCL-SDK

WHAT'S INSIDE THE BOX?

STREAM
High Performance Computing

- OpenCL-Headers
- OpenCL-ICD-Loader
- OpenCL-CLHPP
- Utility libraries
- Sample codes
- Documentation
  - OpenCL-Guide

1ST PARTY OPENCL-SDK

WHAT'S INSIDE THE BOX?

STREA
High Performance Computing

- **OpenCL-Headers**
- OpenCL-ICD-Loader
- OpenCL-CLHPP
- Utility libraries
- Sample codes
- Documentation
  - OpenCL-Guide

- The definitive C bindings to the OpenCL API
- Relatively new helper headers for
  - Creating your own ICD
  - Creating your own layer

1ST PARTY OPENCL-SDK

WHAT'S INSIDE THE BOX?

STREAM High Performance Computing

- OpenCL-Headers
- **OpenCL-ICD-Loader**
- OpenCL-CLHPP
- Utility libraries
- Sample codes
- Documentation
  - OpenCL-Guide

1ST PARTY OPENCL-SDK

- The canonical Installable Client Driver loader
  - libOpenCL.so/OpenCL.dll
- Responsible for loading vendor runtimes

```
clGetPlatformIDs(numPlatforms, platforms,
NULL);
```

- Exposed to
  - Let users tap into latest features
  - Help implementers debug

WHAT'S INSIDE THE BOX?

STREAM
High Performance Computing

- OpenCL-Headers
- OpenCL-ICD-Loader
- **OpenCL-CLHPP**
- Utility libraries
- Sample codes
- Documentation
  - OpenCL-Guide

- Canonical C++ bindings to the OpenCL API
  - Reduces verbosity
  - Adds safety guards
- C++11 is the minimum

1ST PARTY OPENCL-SDK

WHAT'S INSIDE THE BOX?

STREAM
High Performance Computing

- OpenCL-Headers
- OpenCL-ICD-Loader
- OpenCL-CLHPP
- **Utility libraries**
- Sample codes
- Documentation
  - OpenCL-Guide

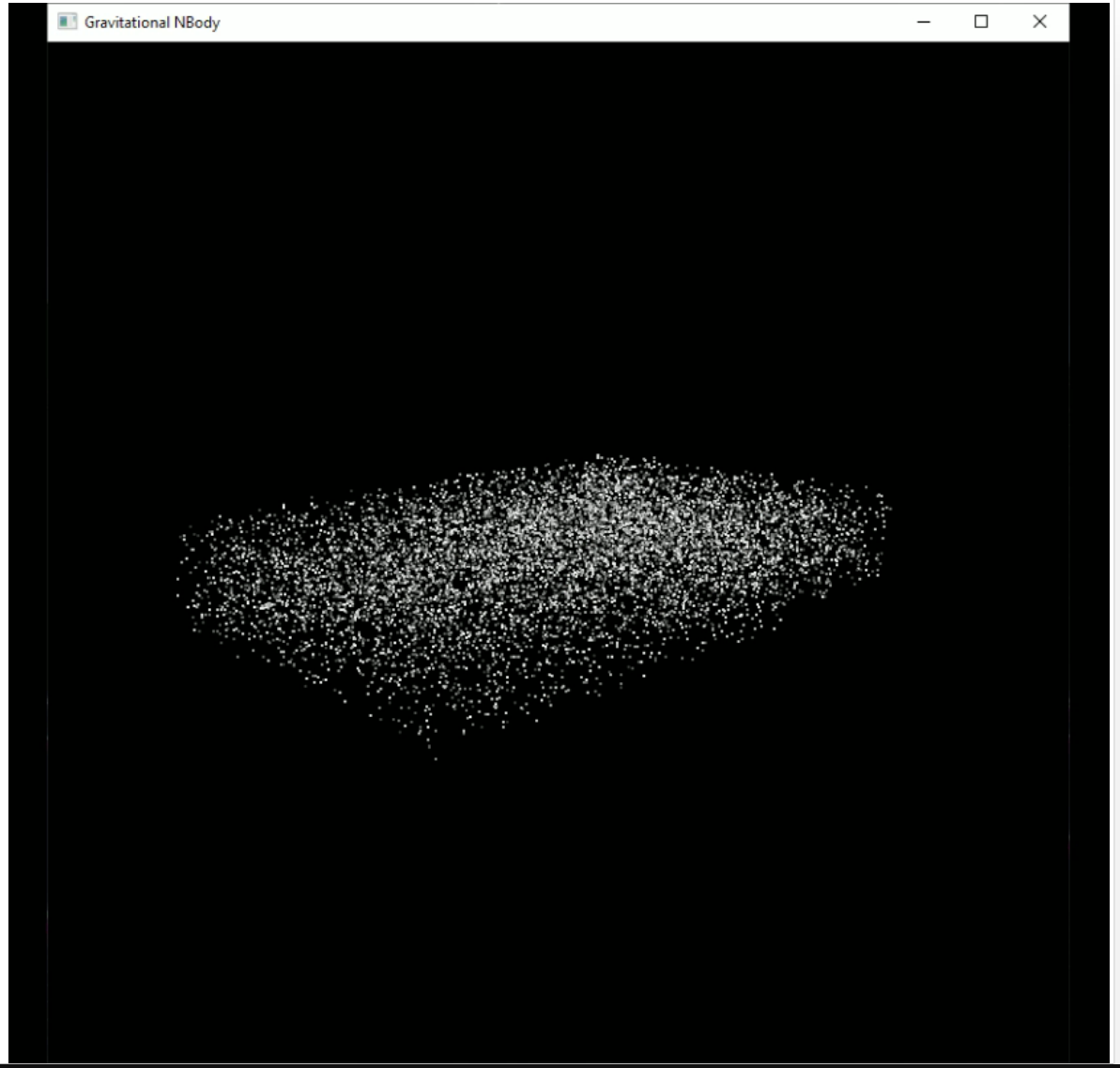
1ST PARTY OPENCL-SDK

- Utility libraries to help developers with common tasks, for eg.
  - Give me a context with device N on platform M
  - Obtain profiling data in std::chrono-friendly format
- Contributions are welcome
  - Convert error code -48 to „CL_INVALID_KERNEL"?

WHAT'S INSIDE THE BOX?

STREAM High Performance Computing

- OpenCL-Headers
- OpenCL-ICD-Loader
- OpenCL-CLHPP
- Utility libraries
- **Sample codes**
- Documentation
  - OpenCL-Guide

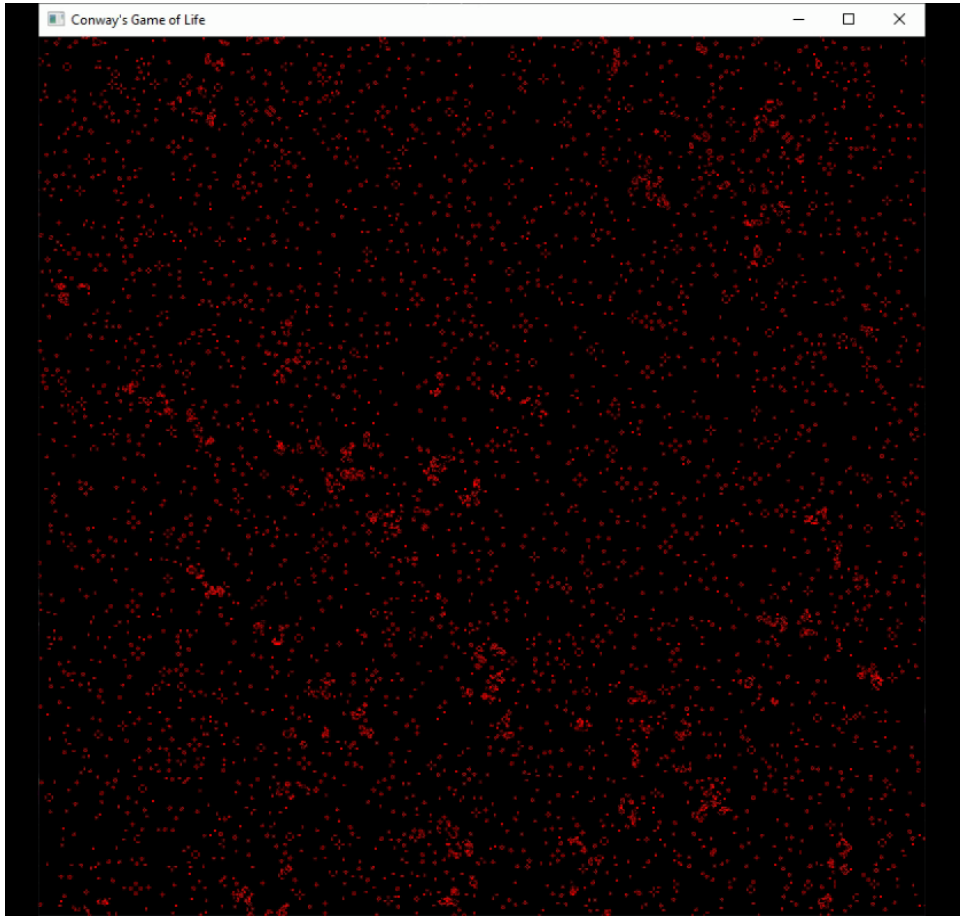


1ST PARTY OPENCL-SDK

WHAT'S INSIDE THE BOX?

STREAM
High Performance Computing

- OpenCL-Headers
- OpenCL-ICD-Loader
- OpenCL-CLHPP
- Utility libraries
- **Sample codes**
- Documentation
  - OpenCL-Guide



1ST PARTY OPENCL-SDK

WHAT'S INSIDE THE BOX?

STREAM
High Performance Computing

- OpenCL-Headers
- OpenCL-ICD-Loader
- OpenCL-CLHPP
- Utility libraries
- Sample codes
- **Documentation**
  - OpenCL-Guide

- All documentation is written in Markdown
- Cross-linking with the OpenCL-Guide
  - Getting started guides
    - Linux
    - Windows
  - CMake support
    - More on this later

1ST PARTY OPENCL-SDK

HIGHER STANDARDS

STREAM
High Performance Computing

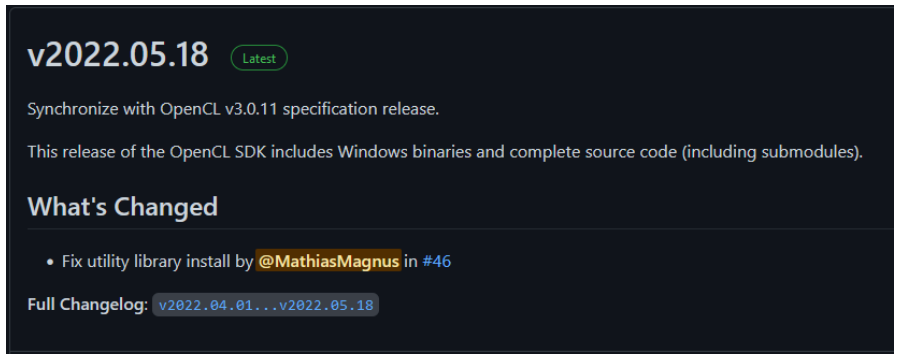
- The entire ecosystem is thoroughly tested
  - Warning free using highest warn levels
    - 3 platforms (Windows, Linux, MacOS)
    - 3 compilers (MSVC, GCC, Clang)
    - Multiple language C/C++ standards
    - With/without compiler extensions
  - Public facing CI scripts

1ST PARTY OPENCL-SDK

HIGHER STANDARDS

STREAM
High Performance Computing

- The entire ecosystem is thoroughly tested
  - Warning free using highest warn levels
    - 3 platforms (Windows, Linux, MacOS)
    - 3 compilers (MSVC, GCC, Clang)
    - Multiple language C/C++ standards
    - With/without compiler extensions
  - Public facing CI scripts
- Continuous Deployment in place
  - Quarterly source & binary releases



1ST PARTY OPENCL-SDK

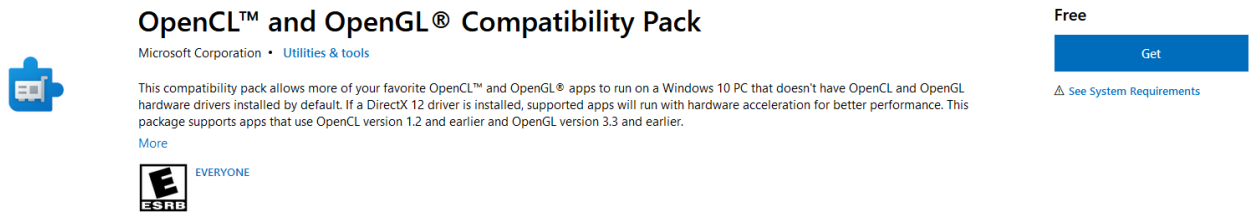
LAYERS ARE KEY

- Ogres are like onions.
- They smell and turn white if you leave them on the sun?
- No Donkey! They got layers!

OPENCL 3.0 ECOSYSTEM REBOOT

STREAM High Performance Computing

- New ICD Layers capability allows augmenting/modifying runtime behavior
  - Add missing features, validate correctness, trace, profile, etc
- OpenCL SDK serves as a „one-stop shop” for devs
  - Initial batch of native samples & utilities awarded to Stream
    - Coming to a GitHub near you
- Defines the semantics of layered implementations to fill in the gaps of platform support
  - Most notably OpenCLOn12, enabling OpenCL across the entire Windows ecosystem, regardless of vendor support.



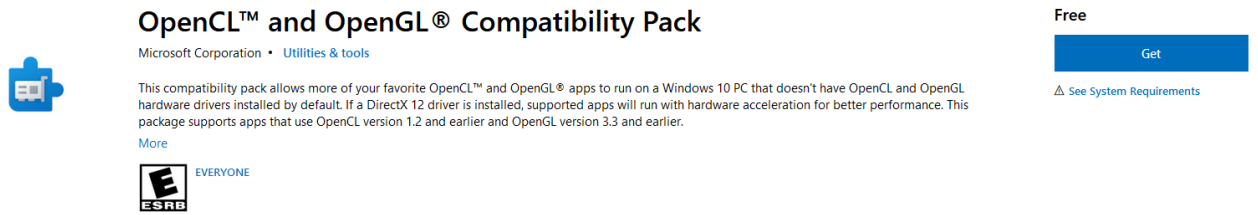
https://www.microsoft.com/en-us/p/opencl-and-opengl-compatibility-pack/9nqpsl29bfff

LAYERS ARE KEY

OPENCL 3.0 ECOSYSTEM REBOOT

STREAM High Performance Computing

- New ICD Layers capability allows augmenting/modifying runtime behavior
  - Add missing features, validate correctness, trace, profile, etc
- OpenCL SDK serves as a „one-stop shop” for devs
  - Initial batch of native samples & utilities awarded to Stream
    - Coming to a GitHub near you
- Defines the semantics of layered implementations to fill in the gaps of platform support
  - Most notably OpenCLOn12, enabling OpenCL across the entire Windows ecosystem, regardless of vendor support.

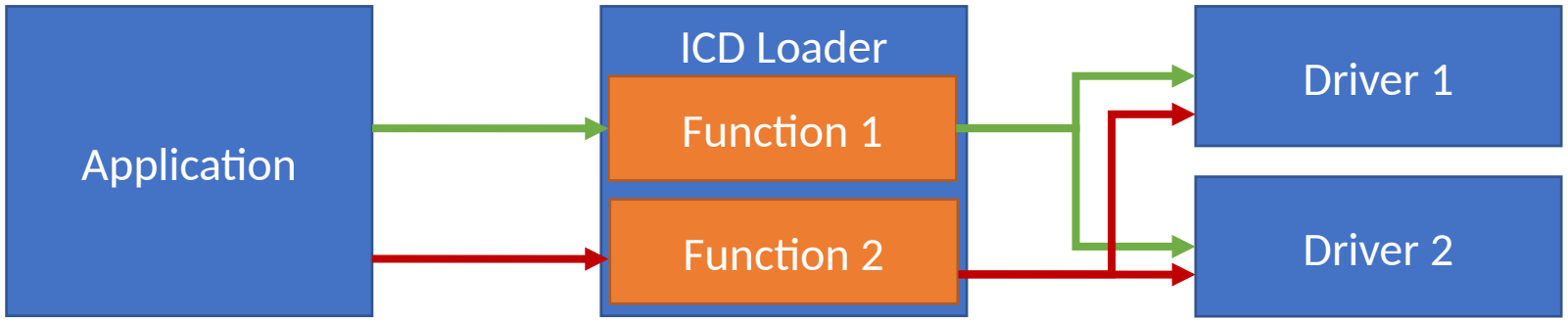


https://www.microsoft.com/en-us/p/opencl-and-opengl-compatibility-pack/9nqpsl29bfff

LAYERS ARE KEY

UNDERSTANDING LAYERS

- Without layers support enabled, the ICD Loader will
  - dispatch the call to the correct vendor driver



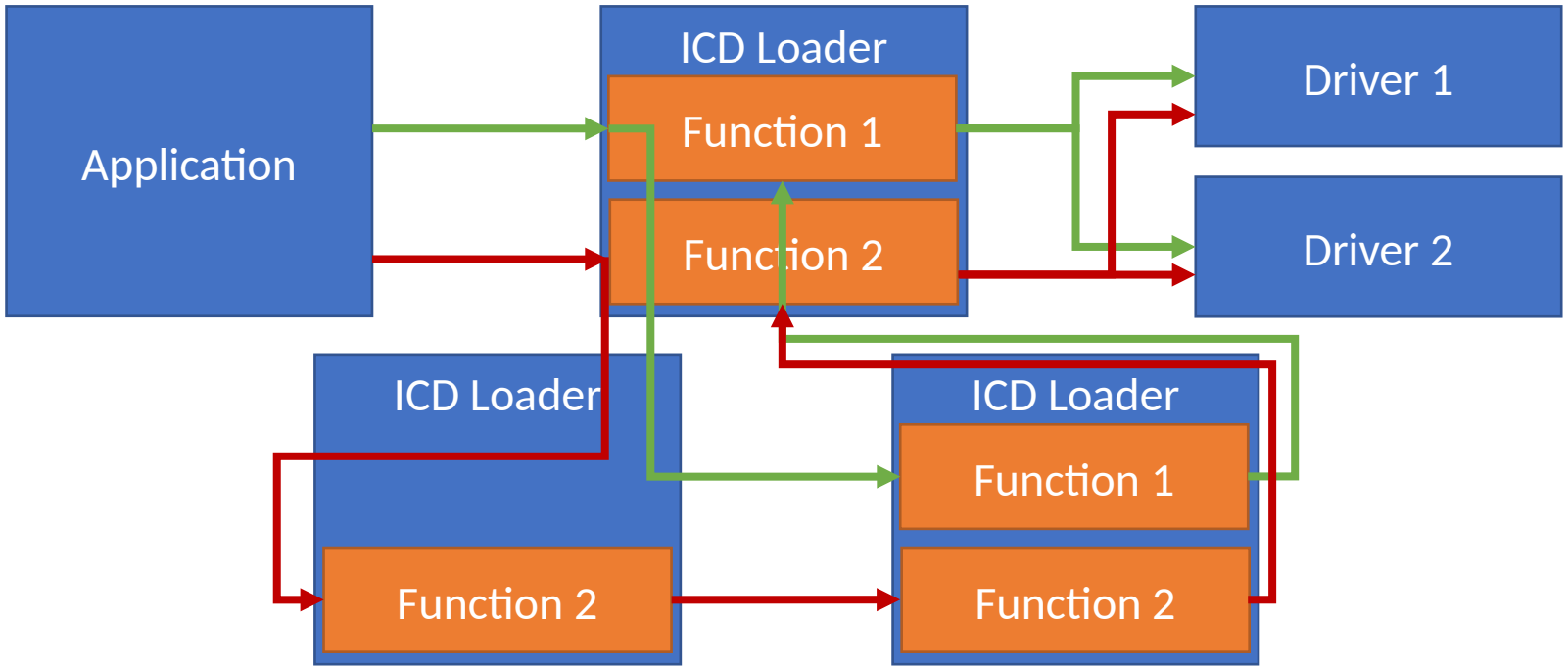
LAYERS ARE KEY

Layers for OpenCL – Brice Videau @ IWOCL 2021
(https://raw.githubusercontent.com/Kerilk/OpenCL-Layers-Tutorial/main/presentation/LayersForOpenCL.pdf)

UNDERSTANDING LAYERS

STREAM High Performance Computing

- With layers support enabled, the ICD Loader will
  - first redirect calls to different active layers
  - then dispatch the call to the correct vendor driver



LAYERS ARE KEY

OBJECT LIFETIME TRACKING

STREAM
High Performance Computing

- Leaking objects

```
clReleaseEvent(evt[0]);
clReleaseEvent(evt[1]);
clReleaseMemObject(bufferSrc);
clReleaseMemObject(bufferDst);
clReleaseMemObject(bufferRes);
clReleaseKernel(kernel);
clReleaseProgram(program);
clReleaseCommandQueue(queue);
clReleaseContext(context);
clReleaseDevice(device);
```

LAYERS ARE KEY

OBJECT LIFETIME TRACKING

STREA
High Performance Computing

- Leaking objects

```
cl_ulong total_time = 0;
for (int i = 0 ; i < steps ; ++i)
{
  cl_event event;
  clEqneueNDRandeKernel(
    queue, kernel, 1, NULL,
    &global_work_size, &local_work_size,
    0, NULL, &event);
  clGetEventProfilingInfo(
    event, CL_PROFILING_COMMAND_START,
    sizeof(cl_ulong), &start, NULL);
  clGetEventProfilingInfo(
    event, CL_PROFILING_COMMAND_END,
    sizeof(cl_ulong), &end, NULL);
  total_time += end – start;

}
```

LAYERS ARE KEY

OBJECT LIFETIME TRACKING

STREA
High Performance Computing

- Leaking objects

```
cl_ulong total_time = 0;
for (int i = 0 ; i < steps ; ++i)
{
  cl_event event;
  clEqneueNDRandeKernel(
    queue, kernel, 1, NULL,
    &global_work_size, &local_work_size,
    0, NULL, &event);
  clGetEventProfilingInfo(
    event, CL_PROFILING_COMMAND_START,
    sizeof(cl_ulong), &start, NULL);
  clGetEventProfilingInfo(
    event, CL_PROFILING_COMMAND_END,
    sizeof(cl_ulong), &end, NULL);
  total_time += end - start;
  clReleaseEvent(event)
}
```

LAYERS ARE KEY

OBJECT LIFETIME TRACKING

STREAM
High Performance Computing

- Leaking objects
- Double free
  - Shortcircuit an option

```
cl_context ctx_a = clCreateContext(...);
clReleaseContext(ctx_a);
// Do a whole bunch of other stuff
/*
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
*/
clReleaseContext(ctx_a);
```

LAYERS ARE KEY

OBJECT LIFETIME TRACKING

STREAM
High Performance Computing

- Leaking objects
- Double free
  - Shortcircuit an option
- Use after free

```
cl_context ctx_a = clCreateContext(...);
clReleaseContext(ctx_a);
// Do a whole bunch of other stuff
/*
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
*/
clGetContextInfo(ctx_a,
CL_CONTEXT_DEVICES,
  0, NULL, &devices_size);
```

LAYERS ARE KEY

OBJECT LIFETIME TRACKING

STREAM
High Performance Computing

- Leaking objects
- Double free
  - Shortcircuit an option
- Use after free
  - Use of dangling handle

```
cl_context ctx_a = clCreateContext(...);
clReleaseContext(ctx_a);

cl_context ctx_b = clCreateContext(...);
clReleaseContext(ctx_a);
```

LAYERS ARE KEY

OBJECT LIFETIME TRACKING

STREAM
High Performance Computing

- Leaking objects
- Double free
  - Shortcircuit an option
- Use after free
  - Use of dangling handle
    - Needs a dictionary between ICD and user code
    - Translate whenever the user uses a handle

```
cl_context ctx_a = clCreateContext(...);
clReleaseContext(ctx_a);

cl_context ctx_b = clCreateContext(...);
clReleaseContext(ctx_a);
```

LAYERS ARE KEY

INPUT ARGUMENT VALIDATION

STREAM
High Performance Computing

- API entry points have various pre/post-conditions users ought not violate
  - If conditions are violated, an error is raised
  - Worst case scenario, the app crashes
- There are (core + extension) functions
  - Writing checks for each is *a lot* of work
  - That work has to be done for every language projection
    - C, C++, C# (OpenCL.Net), Python (PyOpenCL), etc
  - Can we do better?

LAYERS ARE KEY

INPUT ARGUMENT VALIDATION

STREAM
High Performance Computing

```
cl_int clEnqueueWriteBuffer(cl_command_queue command_queue, cl_mem
buffer,
  cl_bool blocking_write, size_t offset, size_t size, const void* ptr,
  cl_uint num_events_in_wait_list, const cl_event* event_wait_list,
  cl_event* event);
```

- command_queue is a valid host command-queue in which the read / write command will be queued. command_queue and buffer must be created with the same OpenCL context.
- buffer refers to a valid buffer object.
- blocking_read and blocking_write indicate if the read and write operations are blocking or non-blocking (see below).
- offset is the offset in bytes in the buffer object to read from or write to.
- size is the size in bytes of data being read or written.
- ptr is the pointer to buffer in host memory where data is to be read into or to be written from.
- event_wait_list and num_events_in_wait_list specify events that need to complete before this particular command can be executed. If event_wait_list is NULL, then this particular command does not wait on any event to complete. If event_wait_list is NULL, num_events_in_wait_list must be 0. If event_wait_list is not NULL, the list of events pointed to by event_wait_list must be valid and num_events_in_wait_list must be greater than 0. The events specified in event_wait_list act as synchronization points. The context associated with events in event_wait_list and command_queue must be the same. The memory associated with event_wait_list can be reused or freed after the function returns.
- event returns an event object that identifies this read / write command and can be used to query or queue a wait for this command to complete. If event is NULL or the enqueue is unsuccessful, no event will be created and therefore it will not be possible to query the status of this command or to wait for this command to complete. If event_wait_list and event are not NULL, event must not refer to an element of the event_wait_list array.

LAYERS ARE KEY

INPUT ARGUMENT VALIDATION

STREAM
High Performance Computing

```
cl_int clEnqueueWriteBuffer(cl_command_queue command_queue, cl_mem
buffer,
  cl_bool blocking_write, size_t offset, size_t size, const void* ptr,
  cl_uint num_events_in_wait_list, const cl_event* event_wait_list,
  cl_event* event);
```

- CL_INVALID_COMMAND_QUEUE if command_queue is not a valid host command-queue.
- CL_INVALID_CONTEXT if the context associated with command_queue and buffer are not the same or if the context associated with command_queue and events in event_wait_list are not the same.
- CL_INVALID_MEM_OBJECT if buffer is not a valid buffer object.
- CL_INVALID_VALUE if the region being read or written specified by (offset, size) is out of bounds or if ptr is a NULL value.
- CL_INVALID_EVENT_WAIT_LIST if event_wait_list is NULL and num_events_in_wait_list > 0, or event_wait_list is not NULL and num_events_in_wait_list is 0, or if event objects in event_wait_list are not valid events.
- CL_MISALIGNED_SUB_BUFFER_OFFSET if buffer is a sub-buffer object and offset specified when the sub-buffer object is created is not aligned to CL_DEVICE_MEM_BASE_ADDR_ALIGN value for device associated with queue. This error code is missing before version 1.1.
- CL_EXEC_STATUS_ERROR_FOR_EVENTS_IN_WAIT_LIST if the read and write operations are blocking and the execution status of any of the events in event_wait_list is a negative integer value. This error code is missing before version 1.1.
- CL_MEM_OBJECT_ALLOCATION_FAILURE if there is a failure to allocate memory for data store associated with buffer.
- CL_INVALID_OPERATION if clEnqueueReadBuffer is called on buffer which has been created with CL_MEM_HOST_WRITE_ONLY or CL_MEM_HOST_NO_ACCESS.
- CL_INVALID_OPERATION if clEnqueueWriteBuffer is called on buffer which has been created with CL_MEM_HOST_READ_ONLY or CL_MEM_HOST_NO_ACCESS.
- CL_OUT_OF_RESOURCES if there is a failure to allocate resources required by the OpenCL implementation on the device.
- CL_OUT_OF_HOST_MEMORY if there is a failure to allocate resources required by the OpenCL implementation on the host.

LAYERS ARE KEY

GENERATE CHECKS FROM XML

STREAM High Performance Computing

Before input arg validation

```
<command
suffix="CL_API_SUFFIX__VERSION_1_2
">
  <proto>
    <type>cl_int</type>
    <name>clReleaseDevice</name>
  </proto>
  <param>
    <type>cl_device_id</type>
    <name>device</name>
  </param>
</command>
```

LAYERS ARE KEY

GENERATE CHECKS FROM XML

STREAM High Performance Computing

Before input arg validation

```
<command
suffix="CL_API_SUFFIX__VERSION_1_2
">
  <proto>
    <type>cl_int</type>
    <name>clReleaseDevice</name>
  </proto>
  <param>
    <type>cl_device_id</type>
    <name>device</name>
  </param>
</command>
```

After input arg validation

```
<command
suffix="CL_API_SUFFIX__VERSION_1_2
">
  <proto>
    <type>cl_int</type>
    <name>clReleaseDevice</name>
  </proto>
  <param>
    <type>cl_device_id</type>
    <name>device</name>
  </param>
  <if>
    <object_is_invalid
    name="device"/>
  </if><then>
    <log>device is not a valid
    device
    </log>
    <name>clReleaseDevice</name>
    <value>CL_INVALID_DEVICE</value>
  </then>
</command>
```

LAYERS ARE KEY

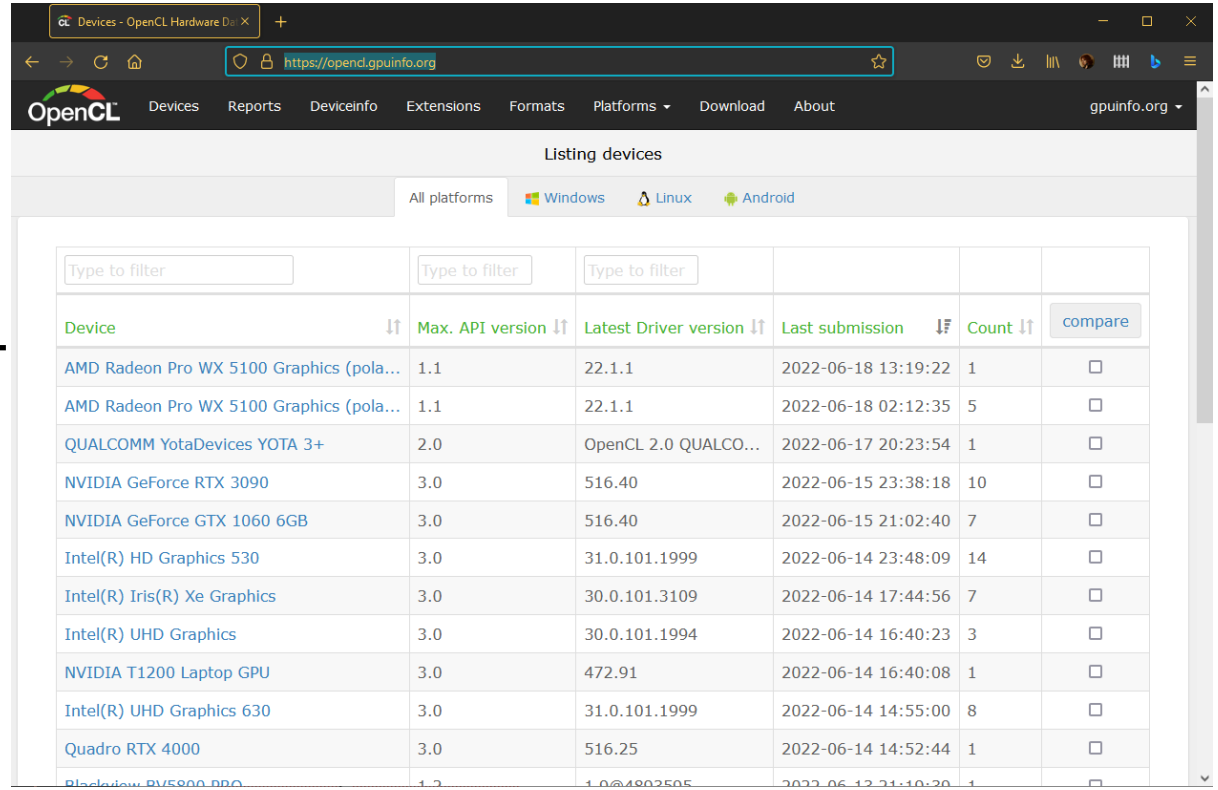
IN ACTUAL NEWS

What happens in OpenCL land?

OPENCL ON GPUINFO

STREA
High Performance Computing

- Popular database of GPU capabilities
- Suports
  - OpenGL, Vulkan, OpenGL ES and now OpenCL!
  - Windows, Linux, Android
- Invaluable if you wish to
  - check the capabilities of a device you don’t own
  - gauge portability when depending on a feature/extension



https://opencl.gpuinfo.org/

IN ACTUAL NEWS

SPIRV2CLC

STREAM High Performance Computing

- Kévin Petit has done some awesome work by releasing a SPIR-V to OpenCL C transpiler
  - Calling it disassembling would be a mild overstatement
  - „If you squint really, you can see the SPIR-V”

IN ACTUAL NEWS

SPIRV2CLC

STREAM
High Performance Computing

- Kévin Petit has done some awesome work by releasing a SPIR-V to OpenCL C transpiler
  - Calling it disassembling would be a mild overstatement
  - „If you squint really, you can see the SPIR-V”

```
kernel void saxpy(
  float a,
  global float* x,
  global float* y)
{
  int gid = get_global_id(0);
  y[gid] = a * x[gid] + y[gid];
}
```

```
kernel void saxpy(float v20, float global* v21, float global* v22){
v23:;
  float __attribute__((aligned(4))) v5_storage; float * v5 =
&v5_storage;
  float global* __attribute__((aligned(8))) v6_storage; float global*
* v6 = &v6_storage;
  float global* __attribute__((aligned(8))) v7_storage; float global*
* v7 = &v7_storage;
  uint __attribute__((aligned(4))) v8_storage; uint * v8 =
&v8_storage;
  *v5 = v20;
  *v6 = v21;
  *v7 = v22;
  ulong v25 = get_global_id(0);
  uint v26 = convert_uint(v25);
  *v8 = v26;
  float v27 = *v5;
  float global* v28 = *v6;
  uint v29 = *v8;
  ulong v30 = convert_long(as_int(v29));
  float global* v31 = &v28[v30];
  float v32 = *v31;
  float global* v33 = *v7;
  uint v34 = *v8;
  ulong v35 = convert_long(as_int(v34));
  float global* v36 = &v33[v35];
  float v37 = *v36;
  float v38 = mad(v27, v32, v37);
  float global* v39 = *v7;
  uint v40 = *v8;
  ulong v41 = convert_long(as_int(v40));
  float global* v42 = &v39[v41];
  *v42 = v38;
  return;
}
```

IN ACTUAL NEWS

STANDARD PORTABLE IR

STREAM
High Performance Computing

- It is the PTX of OpenCL
- Was born out of necessity of vendors wanting to obfuscate proprietary kernel code
  - The API at some point needs device code as an unencrypted stream of ASCII char array (easy to hijack)
  - Also soves some vendor compiler non-conformance issues
- SPIR turned out to be possibly the biggest mistake OpenCL has ever made
  - A mistake to be repeated by Microsoft with DXIL
- SPIR-V saves the day („V” stands for Vulkan, not 5)
  - Vulkan needed an IR badly
  - New IR is adopted by both OpenCL and OpenGL

IN ACTUAL NEWS

STANDARD PORTABLE IR

STREAM
High Performance Computing

- It is the PTX of OpenCL

Aspect	SPIR 1.2	SPIR 2.0	SPIR-V 1.X
LLVM Interaction	Uses LLVM 3.2	Uses LLVM 3.4	100% Khronos defined Round-trip lossless conversion
Compute Constructs	Metadata/Intrinsics	Metadata/Intrinsics	Native
Graphics Construct	No	No	Native
Supported Language Feature Supported	OpenCL C 1.2	OpenCL C 1.2 OpenCL C 2.0	OpenCL C 1.2 / 2.X C++ for OpenCL GLSL
OpenCL Ingestion	OpenCL 1.2 Extension	OpenCL 2.0 Extension	OpenCL 2.1/2.2 Core OpenCL 3.0 Extension
Graphics API Ingestion	-	-	Vulkan 1.X OpenGL 4.6 Core

Source: https://www.khronos.org/spir/

IN ACTUAL NEWS

STANDARD PORTABLE IR

STREAM
High Performance Computing

- It is the PTX of OpenCL

Aspect	SPIR 1.2	SPIR 2.0	SPIR-V 1.X
LLVM Interaction	Uses LLVM 3.2	Uses LLVM 3.4	100% Khronos defined Round-trip lossless conversion
Compute Constructs	Metadata/Intrinsics	Metadata/Intrinsics	Native
Graphics Construct	No	No	Native
Supported Language Feature Supported	OpenCL C 1.2	OpenCL C 1.2 OpenCL C 2.0	OpenCL C 1.2 / 2.X C++ for OpenCL GLSL
OpenCL Ingestion	OpenCL 1.2 Extension	OpenCL 2.0 Extension	OpenCL 2.1/2.2 Core OpenCL 3.0 Extension
Graphics API Ingestion	-	-	Vulkan 1.X OpenGL 4.6 Core

Source: https://www.khronos.org/spir/

IN ACTUAL NEWS

STANDARD PORTABLE IR

STREAM
High Performance Computing

SPIR-V FOR GRAPHICS

- 100% Khronos defined
- Bi-directionally translatable to/from LLVM
- Can represent any shading language
- Rich, open-source tooling

SPIR-V FOR COMPUTE

- Everything SPIR-V for graphics does and more
- Unstructured control-flow
- Pointer arithmetic
  - Optimizer looks very different
  - Both extensions can be transformed away
- Lots of optimizer work
- Done in Mesa
  - OpenCLOn12, clvk

IN ACTUAL NEWS

THE STATE OF SPIR-V

STREAM High Performance Computing

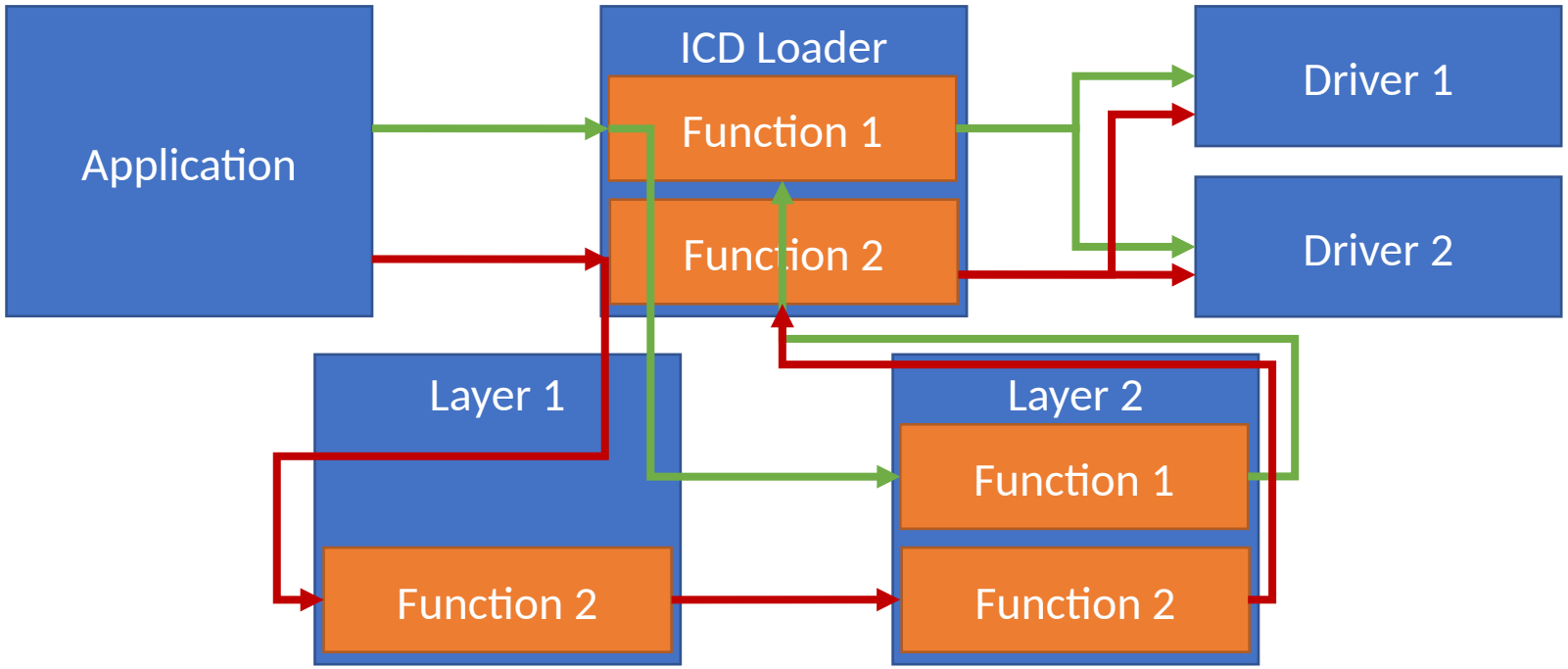
- SPIR-V is a crucial, yet notoriously non-portable feature
  - It’s one of the reasons why Vulkan is such a huge success
  - Tooling around SPIR-V is thriving
  - Greatly improves portability of shaders and tooling
- On desktop, AMD and Nvidia still don’t ship SPIR-V compilers for OpenCL
  - Even OpenCLOn12 supports it
  - cl_khr_il_program is the extension to look for
- Can we teach runtimes that don’t support SPIR-V ingestion to somehow… compile them?
  - I thought you’d never ask

IN ACTUAL NEWS

SPIR-V FOR THE MASSES

STREAM High Performance Computing

- Are you thinking what I'm thinking? Layers!
- Nothing prevents a layer from calling into a different API function than the one being called.

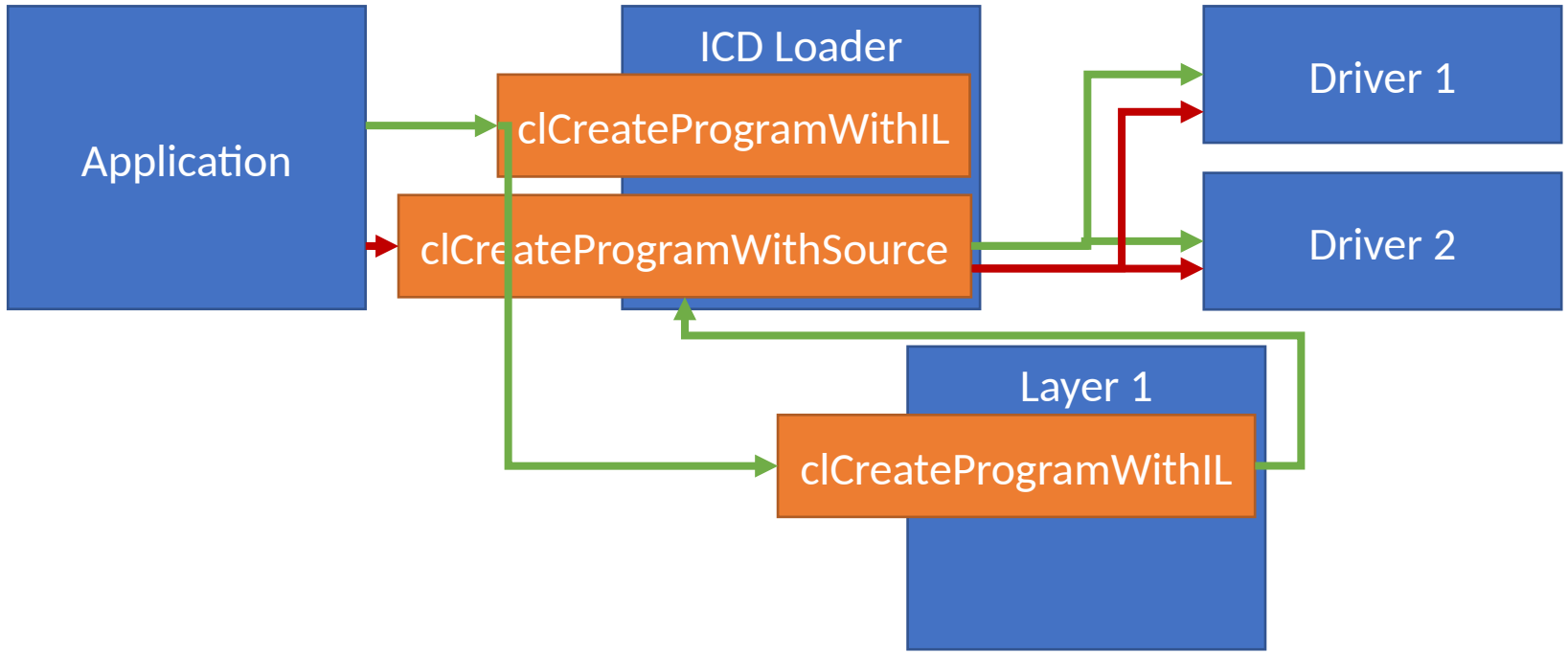


IN ACTUAL NEWS

SPIR-V FOR THE MASSES

STREAM High Performance Computing

- Are you thinking what I'm thinking? Layers!
- Nothing prevents a layer from calling into a different API function than the one being called.



IN ACTUAL NEWS

C++ FOR OPENCL

STREAM High Performance Computing

- Language built on top of OpenCL C 3.0 unified and C++17 enabling most of regular C++ features in OpenCL kernel code.
  - List of restricted C++ features are the same as always
    - Virtual functions, dynamic_cast, refs to func, ptr to member func, RTTI, exceptions, thread_local, non-placement new/delete, STL
- Upstream LLVM 14 with experimental support
  - Need to compile kernels offline

```
clang -cl-std=CLC++2021
kernel.clcpp
```

  - Load IL as binary and feed to clCreateProgramWithIL

```
std::ifstream binary{ location, std::ios::binary };
cl::Program prog{ ctx, std::vector{ std::istreambuf_iterator<char>{ binary
},
                                    std::istreambuf_iterator<char>{} } };
prog.build({ device });
```

- cl_ext_cxx_for_opencl allows online compilation

IN ACTUAL NEWS

C++ FOR OPENCL

STREAM High Performance Computing

- Language built on top of OpenCL C 3.0 unified and C++17 enabling most of regular C++ features in OpenCL kernel code.
  - List of restricted C++ features are the same as always
    - Virtual functions, dynamic_cast, refs to func, ptr to member func, RTTI, exceptions, thread_local, non-placement new/delete, STL
- Upstream LLVM 14 with experimental support
  - Need to compile kernels offline

```
clang -cl-std=CLC++2021
kernel.clcpp
```

  - Load IL as binary and feed to clCreateProgramWithIL

```
std::ifstream binary{ location, std::ios::binary };
cl::Program prog{ ctx, std::vector{ std::istreambuf_iterator<char>{ binary
},
                                    std::istreambuf_iterator<char>{} } };
prog.build({ device });
```

- cl_ext_cxx_for_opencl allows online compilation

IN ACTUAL NEWS

WE’RE HIRING

https://streamhpc.com/jobs/