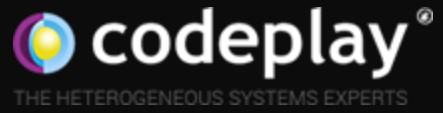


GPU Programming in Modern C++

Gordon Brown & Michael Wong with Steffen Larsen

CppCon 2020 – Sep 2020

Legal disclaimer

This work represents the view of the author and does not necessarily represent the view of Codeplay

Other company, product, and service names may be trademarks or service marks of others

Acknowledgement disclaimer

Numerous people internal and external to the original C++/Khronos group, in industry and academia, have made contributions, influenced ideas, written part of this presentations, and offered feedback to form part of this talk.

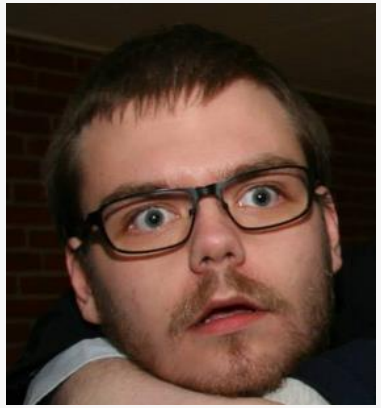
We even lifted this acknowledgement and disclaimer from some of them.

But we claim all credit for errors, and stupid mistakes. **These are ours, all ours!**

Instructors and helpers

Michael Wong

Gordon Brown



Steffen Larsen

Products

Acoran

Integrates all the industry standard technologies needed to support a very wide range of AI and HPC

ComputeCpp

C++ platform via the SYCL™ open standard, enabling vision & machine learning e.g. TensorFlow™

ComputeAorta

The heart of Codeplay's compute technology enabling OpenCL™, SPIR-V™, HSA™ and Vulkan™

Markets

High Performance Compute (HPC)
Automotive ADAS, IoT, Cloud Compute
Smartphones & Tablets
Medical & Industrial

Technologies: Artificial Intelligence
Vision Processing
Machine Learning
Big Data Compute

Company

Leaders in enabling high-performance software solutions for new AI processing systems

Enabling the toughest processors with tools and middleware based on open standards

Established 2002 in Scotland with ~80 employees

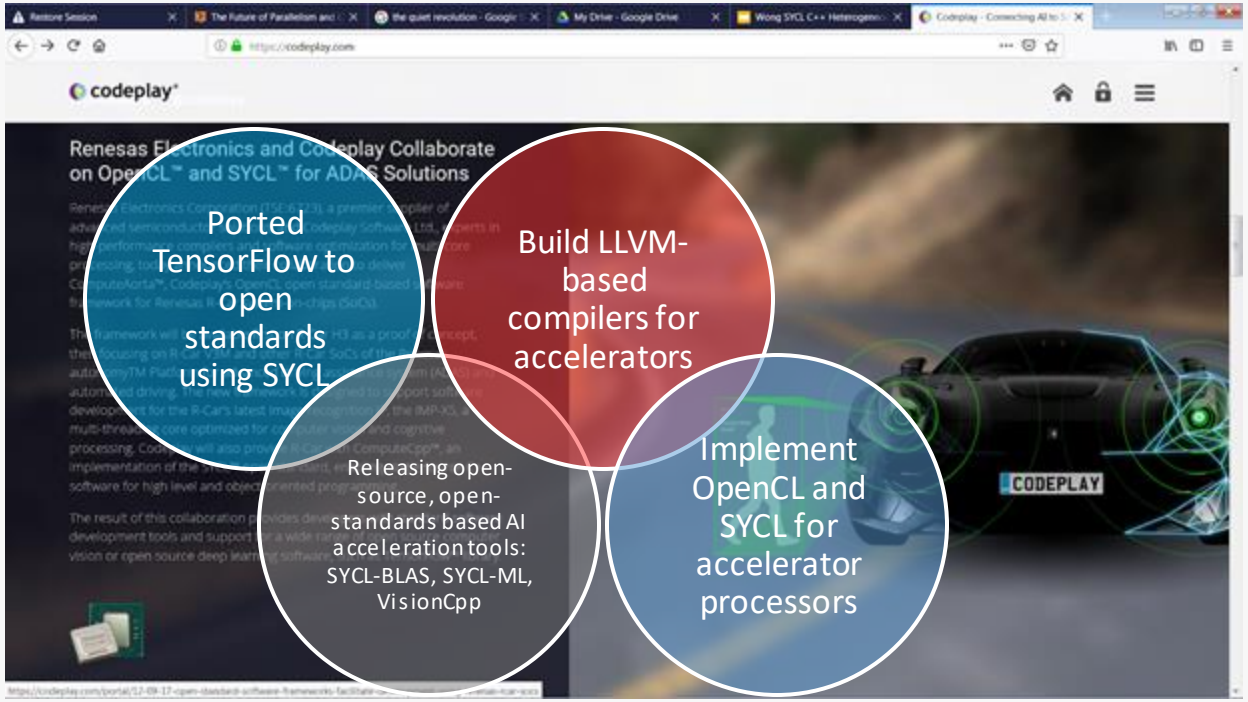
Customers

arm
BROADCOM
CEVA
Imagination
intel
RENESAS
SYNOPSYS
And many more!

Principle Engineer

Gordon Brown

- Background in C++ programming models for heterogeneous systems
- Principle Engineer with Codeplay Software
- Worked on ComputeCpp (SYCL) since its inception
- Contributor to the Khronos SYCL standard for 6 years
- Contributor to C++ executors and heterogeneity or 2 years

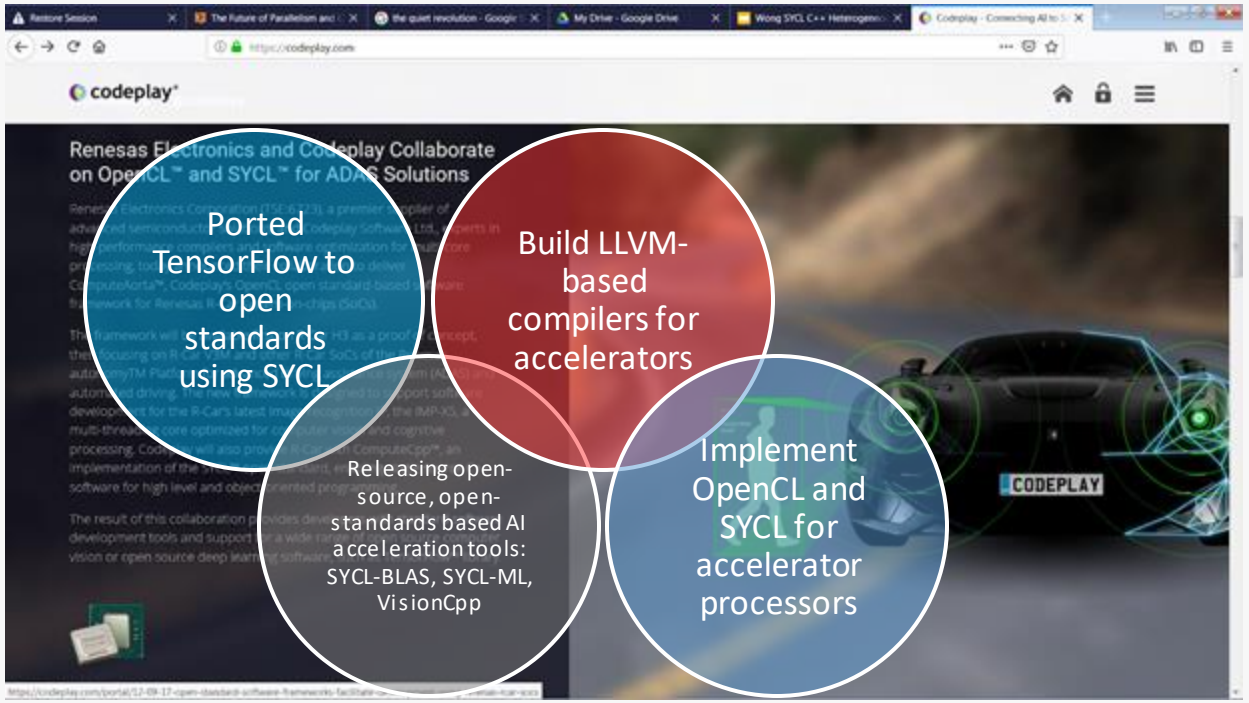


We build GPU compilers for semiconductor companies
Now working to make AI/ML heterogeneous acceleration safe for autonomous vehicle

Steffen Larsen

Staff Engineer

- Background in parallel computing platforms (CUDA, Futhark, etc.)
- Worked on ComputeCpp and the DPC++ CUDA backend
- Contributor to the Khronos SYCL standard

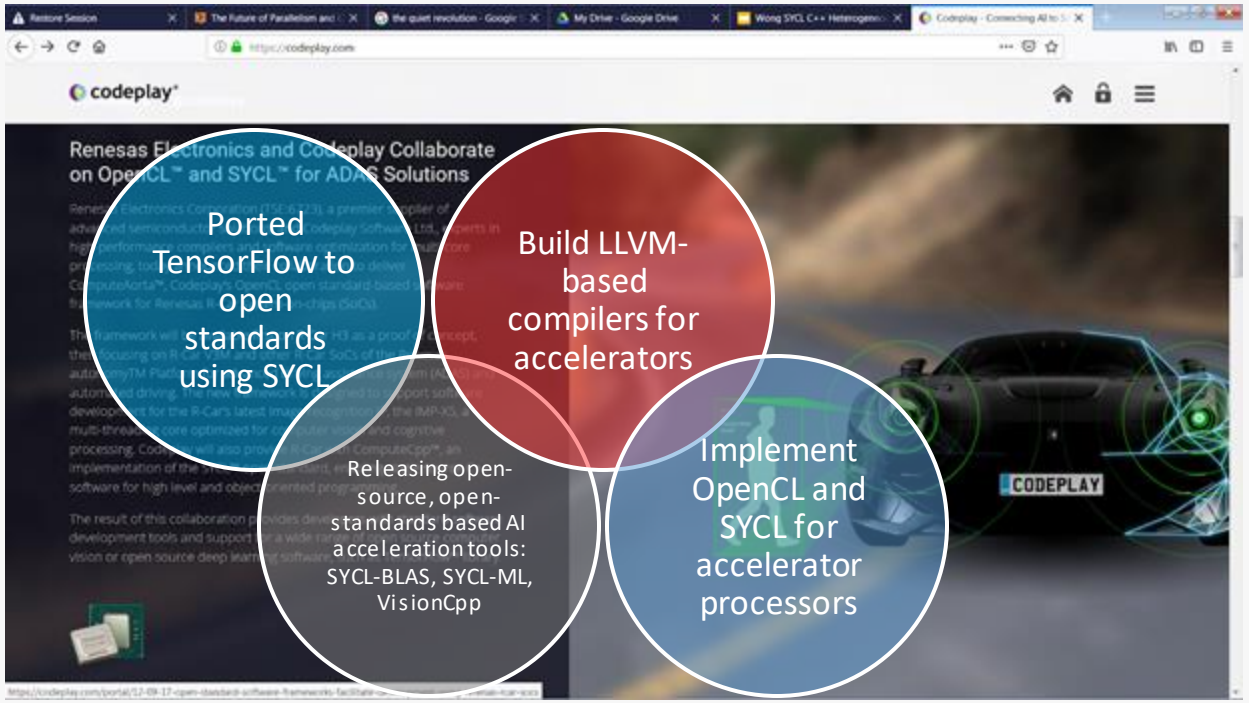


We build GPU compilers for semiconductor companies
Now working to make AI/ML heterogeneous acceleration safe for autonomous vehicle

Distinguished Engineer

Michael Wong

- Chair of SYCL Heterogeneous Programming Language
- C++ Directions Group
- ISOCPP.org Director, VP
  http://isocpp.org/wiki/faq/wg21#michael-wong
- michael@codeplay.com
- fraggamuffin@gmail.com
- Head of Delegation for C++ Standard for Canada
- Chair of Programming Languages for Standards Council of Canada
  Chair of WG21 SG19 Machine Learning
  Chair of WG21 SG14 Games Dev/Low Latency/Financial Trading/Embedded
- Editor: C++ SG5 Transactional Memory Technical Specification
- Editor: C++ SG1 Concurrency Technical Specification
- MISRA C++ and AUTOSAR
- Chair of Standards Council Canada TC22/SC32 Electrical and electronic components (SOTIF)
- Chair of UL4600 Object Tracking
- http://wongmichael.com/about
- C++11 book in Chinese:
  https://www.amazon.cn/dp/B00ETOV2OQ



We build GPU compilers for semiconductor companies
Now working to make AI/ML heterogeneous acceleration safe for autonomous vehicle

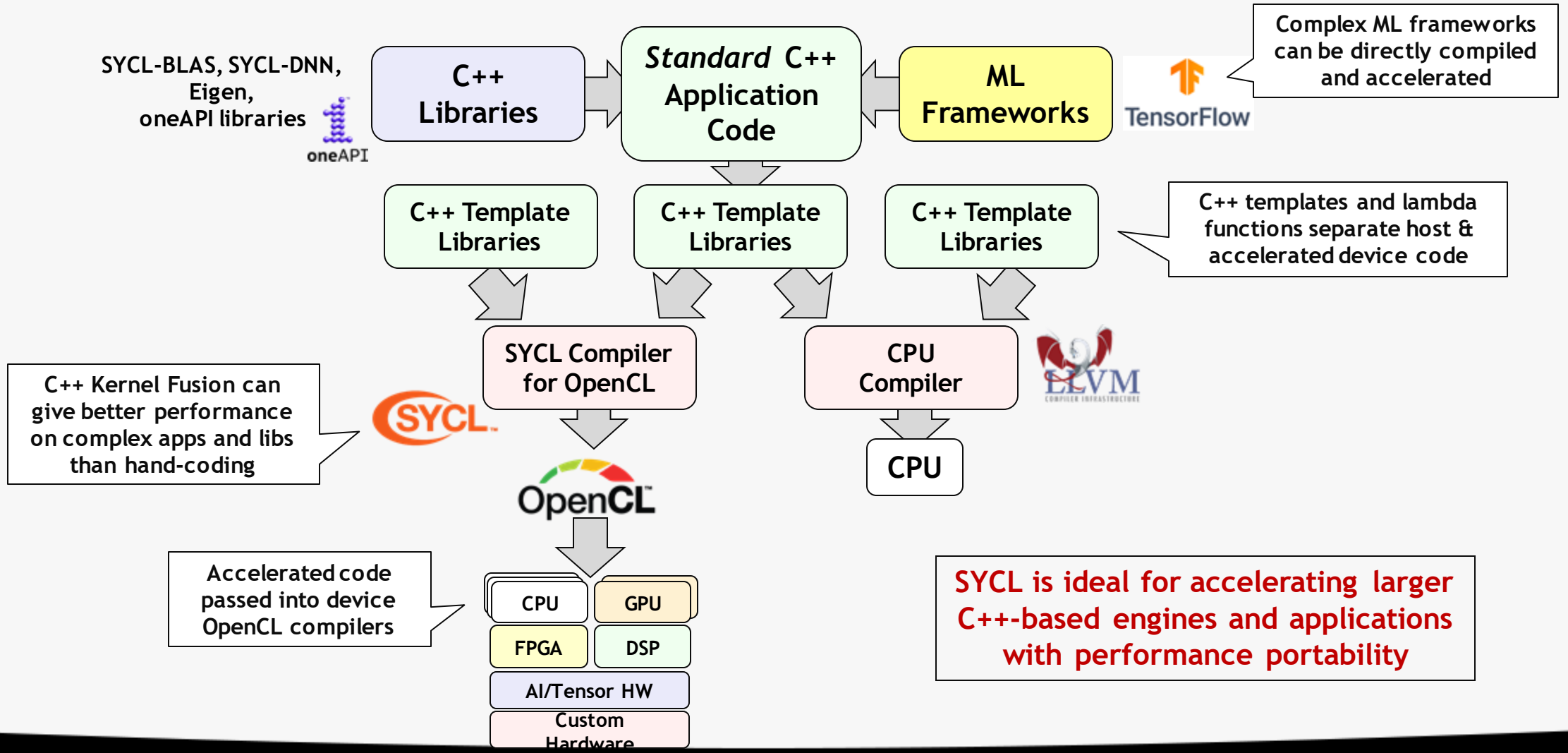
More importantly, who are you?

- Developer
- Manager
- Designer
- Architect
- Teacher
- Hobbyist

What do you want to get out of the class?

Maybe we can tailor future content to what you need.

SYCL Single Source C++ Parallel Programming



SYCL Present and Future Roadmap (May Change)

C++11 | C++14 | C++17 | C++20 | C++23

2011

OpenCL

OpenCL 1.2
OpenCL C Kernel Language

2015

SYCL

SYCL 1.2
C++11 Single source programming

OpenCL

OpenCL 2.1
SPIR-V in Core

SPIR

2017

SYCL

SYCL 1.2.1
C++11 Single source programming

OpenCL

OpenCL 2.2

SPIR

2020

SYCL

SYCL 2020
C++17 Single source programming
Many backend options

OpenCL

OpenCL 3.0

SPIR

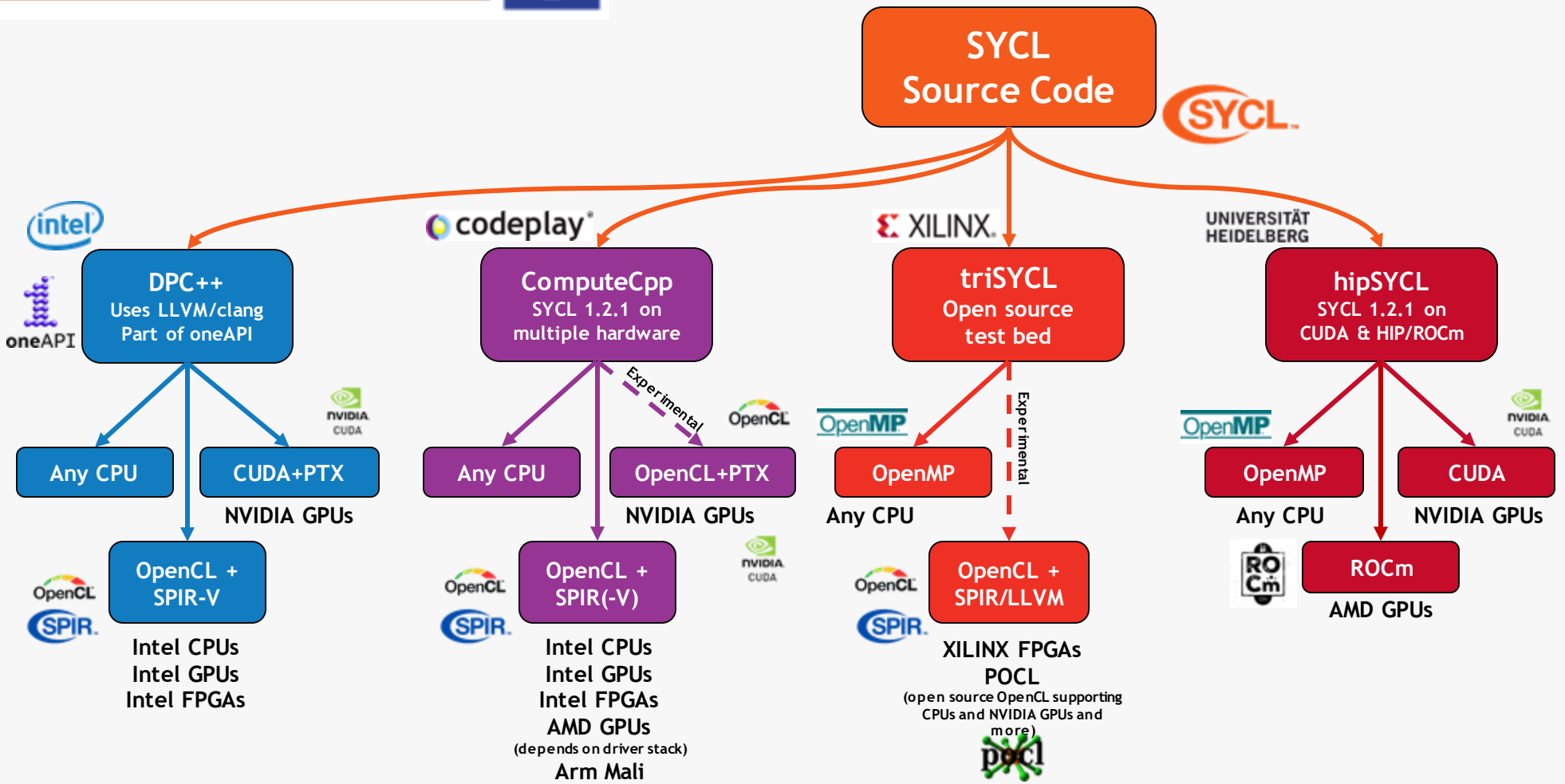
2021-?????

SYCL

SYCL 2021-?
C++20 Single source programming
Many backend options

OpenCL

Multiple members active in both SYCL and ISO C++ committees



SYCL Evolution

SYCL 2020 compared with SYCL 1.2.1

- Easier to integrate with C++17 (CTAD, Deduction Guides,...)
- Less verbose, smaller code size, simplify patterns
- Backend independent
- Multiple object archives aka modules simplify interoperability
- Ease porting C++ applications to SYCL
- Enable capabilities to improve programmability
- Backwards compatible but Minor API break based on user feedback

SYCL 2020 Provisional Features

Backend generalization

Unified shared memory (USM)

API simplifications

Host tasks

In order queues

Nvidia GPU support

Integration of successful Extensions plus new Core functionality

Converge SYCL with ISO C++ and continue to support OpenCL on more devices

CPU
GPU
FPGA
AI processors
Custom Processors

SYCL 2020 Roadmap (WIP, MAY CHANGE)

2017
SYCL 1.2.1

Target 2020
Provisional Q3 then Final Q4

Improving Software Ecosystem

Tool, libraries, GitHub

Expanding Implementation

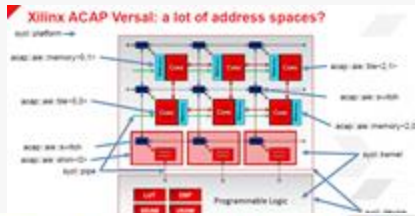
DPC++
ComputeCPP
triSYCL
hipSYCL

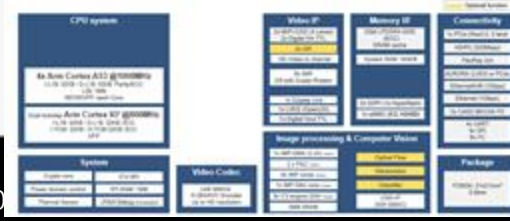
Regular Maintenance Updates

Spec clarifications, formatting and bug fixes
https://www.khronos.org/registry/SYCL/

Selected Extension Pipeline aiming for SYCL 2020 Provisional Q3

Reduction
Subgroups
Accessor simplification
Atomic rework
Extension mechanism
Address spaces





Repeat The Cycle every 1.5-3 years

What you will learn from this class

Why parallelism and heterogeneous computing is important

Fundamental patterns of parallelism

GPU optimization principals and techniques

An introduction to the SYCL 2020 specification

This class is interactive

The class will jump between short lectures and hands-on exercises

Feel free to ask a question in the Slack channel at any time

There will be opportunities after each lecture to answer questions, clarify something about what was presented or have a discussion

Hand-on exercises

There are a number of exercises throughout the class

These are optional, do as much or as little as you want, and feel free to go at your own pace

We encourage you to experiment, try new things to see what happens

The exercises will be available afterwards to continue in your own time if you wish to

The exercises are a part of SYCL Academy

They are currently in a separate branch:

github.com/AerialMantis/syclacademy/tree/cppcon2020

```
git clone --recurse-submodules -b cppcon2020
  git@github.com:AerialMantis/syclacademy.git
```

Configuring

The code exercises can be configured using Cmake

SYCL Academy supports multiple SYCL implementation

For this class we will be limited to ComputeCpp

```
mkdir build
cd build

cmake ../ -G [generator] -A “x64”
  -DSYCL_ACADEMY_USE_COMPUTECPP=ON
  -DSYCL_ACADEMY_INSTALL_ROOT=/path/to/computecpp/install/root
```

Requirements

ComputeCpp CE 2.2.0

Windows 10 or Ubtunu 16.04/18.04

C++17 compiler (MSVC 2019 or GCC 8.0)

CMake (version 3.14or greater)

Your chosen build system (MSVC, Ninja, Make recommended)

Navigating the repo

The lecture slides are in **Lesson_Materials** if you want to review them, they are written in reveal.js, viewable in most browser

The exercises are in **Code_Exercises**, each one has a **doc.txt** file which describes the exercise

Each exercise also has a **source.cpp** file for working on the exercise and a **solution.cpp** file which contains a possible solution

SYCL 2020

SYCL 2020 is a brand new specification (currently provisional)

We wanted to teach the latest version of the standard

But this does mean that implementations are not complete yet

ComputeCpp has support for all features taught in this class

Installing ComputeCpp

Follow the getting started guide in SYCL Academy readme

Requires installing the OpenCL drivers and SDK

This will also be the first exercise, so there will be time to make sure you are set up for the rest of the exercises

Supported devices

For this class we recommend targeting Intel devices (Intel GPU or Intel CPU)

If you are unable to install the OpenCL drivers there is a docker image available with the drivers and ComputeCpp pre-installed

If you do not have an Intel device you can use the host device

Schedule

Day 1	Day 2	Day 3
Welcome		
Importance of Parallelism & heterogeneity What is SYCL	Fundamentals of Parallelism Using USM	GPU Optimization Principals Image Convolution
Break		
Enqueuing a Kernel Managing Data	Asynchronous Execution Data & Dependencies	Coalesced Global Memory Vectorization
Lunch		
Handling Errors Device Discovery	In Order Queues Advanced Data Flow	Local Memory Further Optimization
Break		
Data Parallelism Introduction to USM	Multiple Devices ND Range Kernels	Q & A
Close		

Class Times

Mon/Tue/Wed
September 21/22/23
09:00 – 15:00 MDT (Mountain time, conference time)
08:00 – 14:00 PDT (West Coast)
11:00 – 17:00 EDT (East Coast)
15:00 – 21:00 UTC
17:00 – 23:00 CEST
23:00 – 05:00 Asia/Pacific (Beijing)
https://time.is/MDT

Breaks

First break: 16:15 – 16:45 UTC

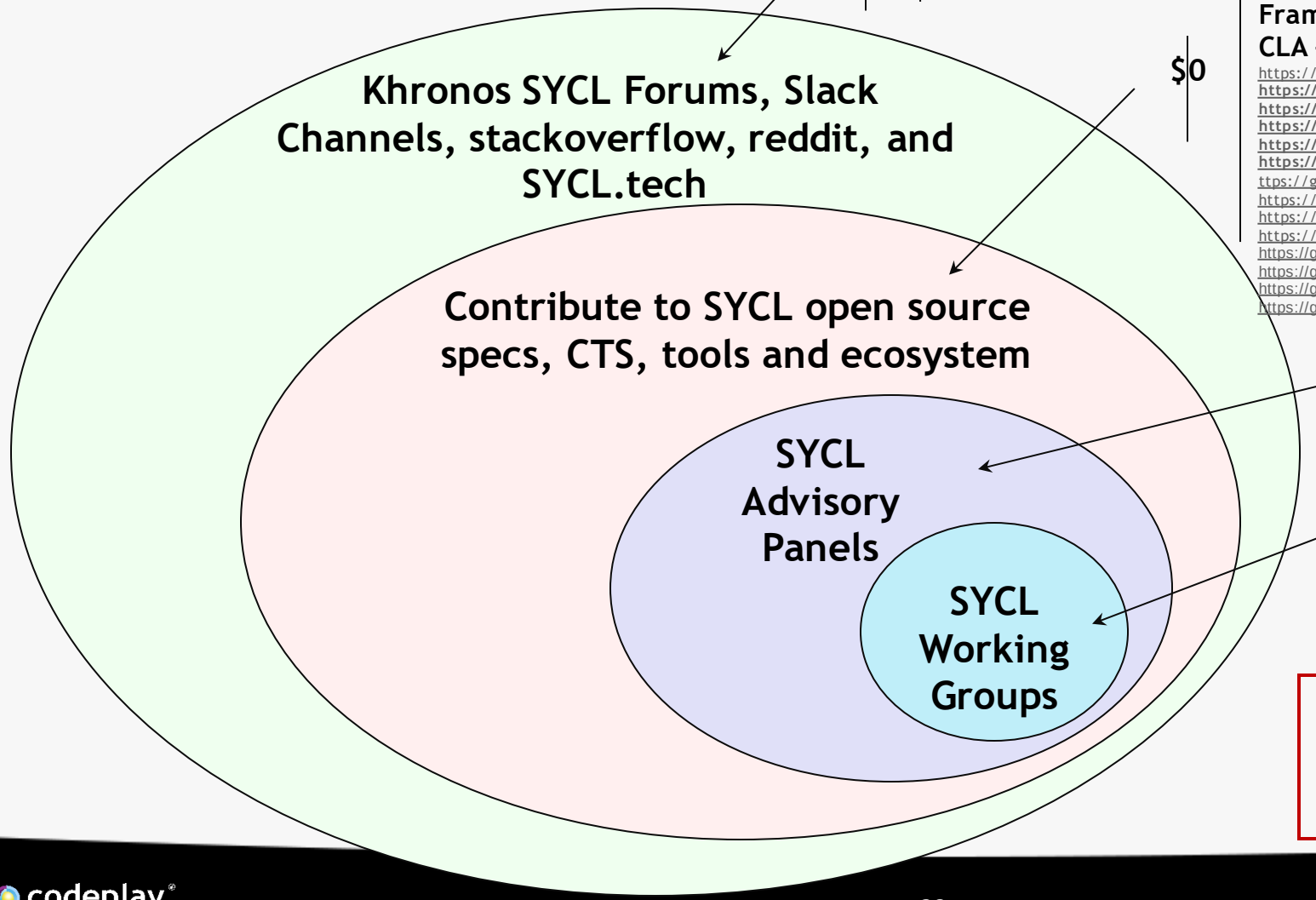
Second break: 17:45 – 18:15 UTC

Third break: 19:15 – 19:45 UTC

SYCL 2020 final is coming

- SYCL 2020 provisional is released
- We need your feedback asap
  - https://khronosdevs.slack.com/archives/CE9UX4CHG
  - https://community.khronos.org/c/sycl/
  - https://sycl.tech/
- What features are you looking for that is not in SYCL 2020?
- What feature would you like to aim for in future SYCL?
- How do you join SYCL?

Engaging with the Khronos SYCL Ecosystem



$0

Open to all!

https://community.khronos.org/www.khr.io/slack
https://khronosdevs.slack.com/archives/CE9UX4CHG
https://community.khronos.org/c/sycl/
https://stackoverflow.com/questions/tagged/sycl
https://www.reddit.com/r/sycl
https://github.com/codeplaysoftware/syclacademy
https://sycl.tech/

$0

Spec fixes and suggestions made under the Khronos IP Framework. Open source contributions under repo's CLA - typically Apache 2.0

https://github.com/KhronosGroup
https://github.com/KhronosGroup/SYCL-CTS
https://github.com/KhronosGroup/SYCL-Docs
https://github.com/KhronosGroup/SYCL-Shared
https://github.com/KhronosGroup/SYCL-Registry
https://github.com/KhronosGroup/SyclParallelSTL
ttps://github.com/codeplaysoftware/SYCL-DNN
https://github.com/codeplaysoftware/tensorflow
https://software.intel.com/en-us/oneapi/onednn
https://github.com/oneapi-src/oneDNN
https://github.com/codeplaysoftware/TensorOpt/tree/master
https://github.com/codeplaysoftware/standards-proposals/blob/master/interop_task/interop_task.md
https://github.com/intel/llvm/tree/sycl/sycl/doc/extensions
https://github.com/triSYCL/triSYCL

$0

Invited Advisors under the Khronos NDA and IP Framework can comment and contribute to requirements and draft specifications
https://www.khronos.org/advisors/

$

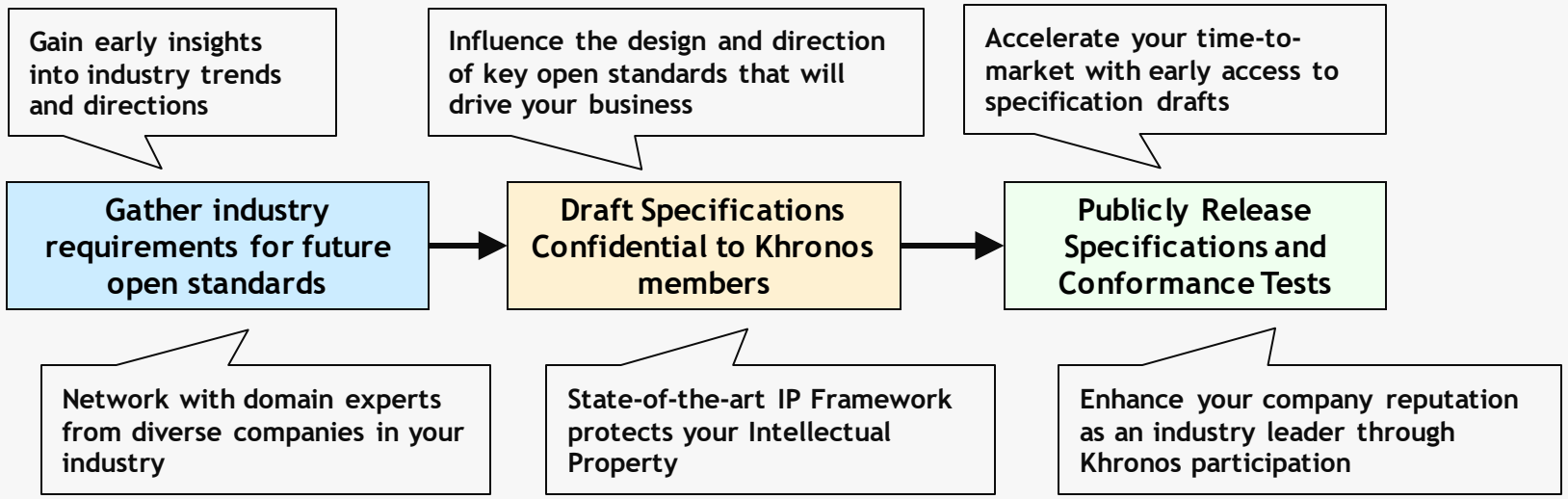
Khronos members under Khronos NDA and IP Framework participate and vote in working group meetings. Starts at $3.5K/yr.
https://www.khronos.org/members/
https://www.khronos.org/registry/SYCL/

Any member *or* non-member can propose a new SYCL feature or fix

Thank You!

- Khronos SYCL is creating cutting-edge royalty-free open standard
  - For C++ Heterogeneous compute, vision, inferencing acceleration
- Information on Khronos SYCL Standards: https://www.khronos.org/sycl/
- Any entity/individual is welcome to join Khronos SYCL: https://www.khronos.org/members/
- Join the SYCLCon Tutorial Monday and Wednesday Live panel : Wednesday Apr 29 15:00-18:00 GMT
  - Have your questions answered live by a group of SYCL experts
- Michael Wong: michael@codeplay.com | wongmichael.com/about

KHRONOS GROUP



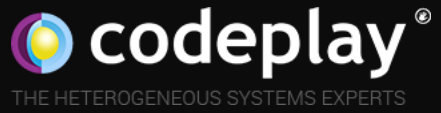
Benefits of Khronos membership

SYCL Resources

- SYCL 2020 Provisional Specification
  - https://www.khronos.org/registry/SYCL/specs/sycl-2020-provisional.pdf
- SYCL 1.2.1 Reference Guide
  - https://www.khronos.org/files/sycl/sycl-121-reference-guide.pdf
- SYCL 2020 Provisional Reference Guide VERY EARLY REVIEW
  - NOT TO BE DISTRIBUTED OUTSIDE OF CLASS
  - this is an early draft WIP. It's known to be incomplet and incorreck, and it has lots of bad formatting.
  - But Feedback is welcome but not everything in the reference guide may be implemented for SYCL 2020 final
  - https://drive.google.com/file/d/1CizY5bfScf0GQXKTATc_L0Pu_Kc96Bs3/view?usp=sharing
  - What examples would you want to see in a Ref Guide?

Questions during tutorial

- Please watch the time here: https://time.is/MDT as class tracks to Mountain daylight time
- Please ask questions during live class by raising your hand (for clarifications) or typing in zoom chat (for longer questions) during the class in Zoom, moderator will call each by turn
  - We may not be able to answer all questions due to time. Any Unanswered questions in zoom chat will be transferred to slack or just ask directly on slack.
- During breaks and at all times, please ask questions on slack channel
  - https://bit.ly/CppConSlack2020Invite1
  - https://cppcon.slack.com/archives/GKFUQDU1G
- Slides will be in the repo
- Please mute and turn off cameras to enable bandwidth



Questions?