Modern & Idiomatic C++14

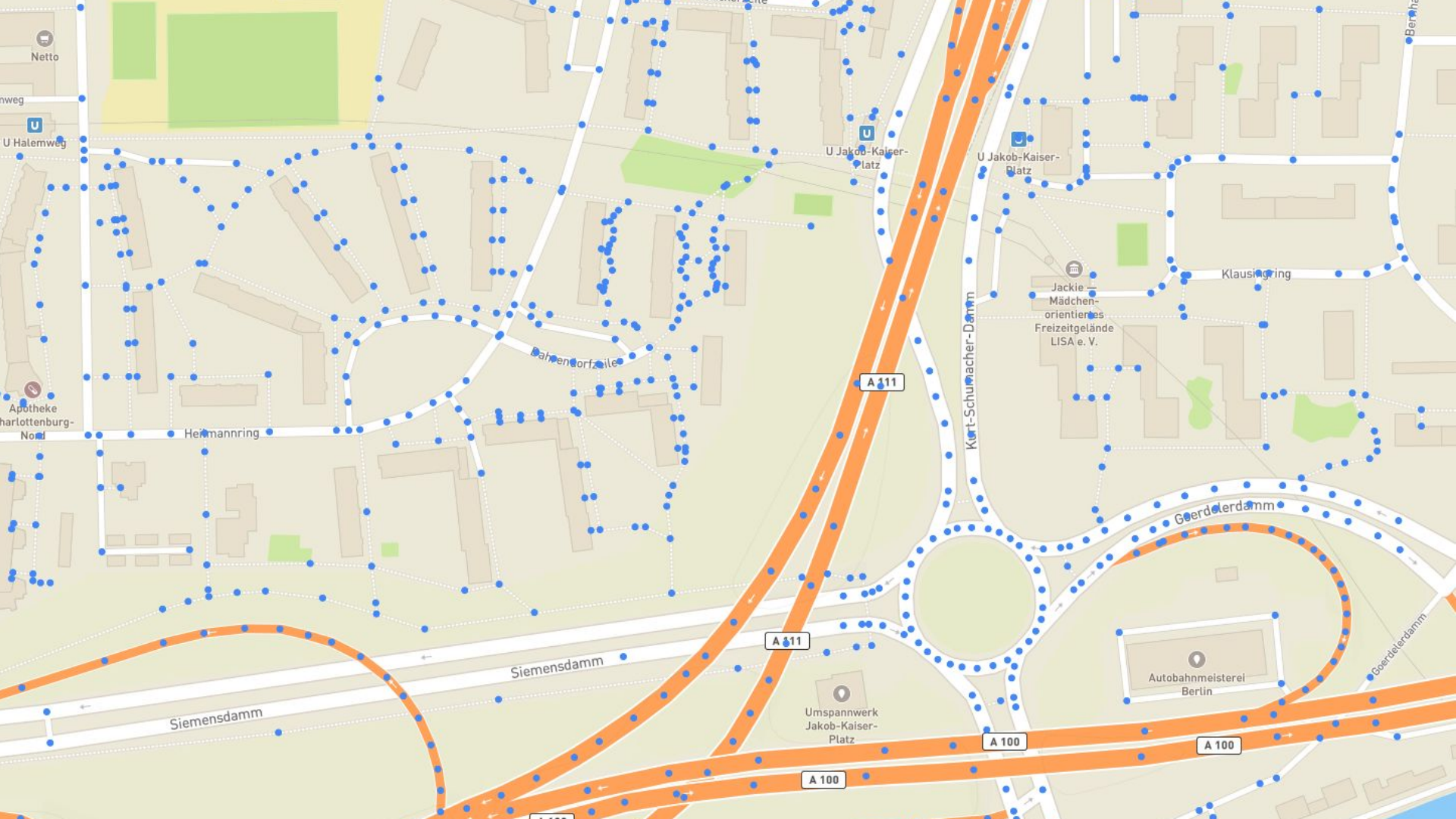
GridKa School 2016

Organizational

Daniel J H, with Mapbox (Directions / Drive)

https://wiki.scc.kit.edu/gridkaschool/index.php

https://github.com/daniel-j-h/gridka2016 (git.io/gridka2016cpp)


Netto
U Halemweg
U Jakob-Kaiser-Platz
U Jakob-Kaiser-Platz
Klausingring
Jackie – Mädchen-orientiertes Freizeitgelände LISA e. V.
Kurt-Schumacher-Damm
Bahndorfzeile
Apotheke Charlottenburg-Nord
Heimannring
A 111
A 111
Siemensdamm
Siemensdamm
Goerdelerdamm
Goerdelerdamm
Autobahnmeisterei Berlin
Umspannwerk Jakob-Kaiser-Platz
A 100
A 100
A 100


San Pablo Bay
Wildlife Area
Richmond Heights
Tiburon
Berkeley
Walnut Creek
Clayton
Brentwood
Mount Diablo State
Park
Moraga
Blackhawk
San Francisco
Pacifica
Hillsborough
San Mateo
Moss Beach

Agenda

Interactive workshop, not many slides, dive into code and issues

Warmup Tooling Ownership

Ranges Optional Variant

Concepts Performance

What You'll Need

Clang/GCC/CMake/Boost

coliru.stacked-crooked.com

http://gcc.godbolt.org

https://github.com/daniel-j-h/gridka2016 (git.io/gridka2016cpp)

VMs

```
ssh \
  -o PreferredAuthentications=password \
  -o PubkeyAuthentication=no \
  gks@host
```

Warmup

Clone the repository

Make sure your setup works

Let’s have a look at some code!

Tooling in 2016, What's Moving

Clang / GCC, GDB / LLDB, libstdc++ / libc++

CMake

clang-format, clang-tidy, libclang

Sanitizers

Fuzzing

Ownership, Lifetimes

Who is responsible for cleaning up an object and when does it happen

How can we express single / shared ownership and transfer it

https://doc.rust-lang.org/book/ownership.html

https://doc.rust-lang.org/book/references-and-borrowing.html

https://doc.rust-lang.org/book/lifetimes.html

Special Members

Interaction between a type's special member functions

Constructor

Destructor

Copy / Move Constructor

Copy / Move Assignment Operator

https://accu.org/content/conf2014/Howard_Hinnant_Accu_2014.pdf (p 28)

RAII or } as a Garbage Collector

```
handle = open('log.txt')
workOn(handle)


with open('log.txt') as handle:
    workOn(handle)
```

Unique / Shared Ownership Policies

```
struct File {

  File(const string& path)
    : handle{fopen(path.c_str(), “r+”), &fclose()}
  { /* … */ }

  unique_ptr<FILE, decltype(&fclose)> handle;

};
```

https://gist.github.com/daniel-j-h/7ca2bf73047c788f3d9b (Lua example)

Rule of Zero

Rule of Three

Rule of Five

Rule of Zero

https://rmf.io/cxx11/rule-of-zero/

What we aimed for with C++17

Concepts Ranges Modules Coroutines Uniform Call Syntax

Transactional Memory Executors Parallel Algorithms Contracts

Fold Expressions Ctor Type Deduction Future Continuation

Filesystem Networking Any Optional Variant Constexpr If

Structural Bindings If Initializer

What is in C++17

Concepts Ranges Modules Coroutines Uniform Call Syntax

Transactional Memory Executors **Parallel Algorithms** Contracts

Fold Expressions **Ctor Type Deduction** Future Continuation

Filesystem Networking **Any** **Optional** **Variant** **Constexpr If**

Structural Bindings **If Initializer**

Structural Bindings

```
string key; int value;

std::tie(key, value) = make_pair(“ten”s, 10)


const auto [key, value] = make_pair(“ten”s, 10);
```

http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2016/p0217r2.html

If Initializer

```
if (auto it = m.find(key); it != end(m))
  return it->second;
```

```
if (init; condition)
```

http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2016/p0305r0.html

Ranges or std2::

Ranges: `[first, last)` on a higher level

Adaptors: composing computational pipelines

Proposal https://github.com/ericniebler/range-v3

Boost.Range

Expressive Semantics

```
auto duration = [&graph](auto edge) { return graph[edge].duration; };

auto positive = [](auto duration) { return duration > 0; };


auto d = accumulate(edges(graph)
                    | transformed(duration)
                    | filtered(positive), 0);
```

Functions as Ranges

```
copy_n(make_function_input_iterator(readOne, boost::infinite{}), n, out);
```

```
copy(rng, make_function_output_iterator(writeOne));
```

Error Handling: Optional

```
template <typename T>

using Maybe = optional<T>;
```

Sum-Types: Variant

```
template <typename Left, typename Right>
using Either = variant<Left, Right>;
```

```
apply_visitor(visitor, either);
```

http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2016/p0088r2.html

Lambda Visitor or Is this Haskell/Swift/Rust

```
res.match([] (Response) { return "ok"; },
          [] (Error)    { return "error"; });
```

https://gist.github.com/daniel-j-h/04efdc54dae50c1b9bff688ec354e693

https://github.com/mapbox/variant/issues/113

Concepts

Template Duck Typing

http://en.cppreference.com/w/cpp/concept

Concepts Lite

http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2015/n4377.pdf

GCC impl. available: -std=c++1z -fconcepts

Concepts Lite

```
template <typename Container>
requires Sortable<Container>{}
void sort(Container& c);
```

Concepts Lite

```
template <Sortable Container>

void sort(Container& c);
```

Concepts Lite

```
void sort(Sortable& c);
```

Emulate on Traits Building-Blocks

```
template <typename Iter>
struct is_random_access_iterator : std::is_base_of<
  std::random_access_iterator_tag,
  typename std::iterator_traits<Iter>::iterator_category
> {};

static_assert(is_random_access_iterator<Iter>(), “”);
```

Emulate via Introspection

Expression SFINAE + `std::enable_if`

Introspect `.sort` member function, fall back to `std::sort`

Could as well tag-dispatch on `iterator_category` in this scenario

Constexpr If

Generated Assembly, Inliner

http://gcc.godbolt.org/

http://clang.llvm.org/docs/UsersManual.html `-Rpass`

Thanks!

daniel+gridka2016@trvx.org