# Changing /std:c++14 to /std:c++20

## How Hard Could It Be?

KEITH STOCKDALE

Cppcon
The C++ Conference

2025
September 13 - 19

## About me!

- Senior Software Engineer at Rare Ltd
- Started as an intern in 2015
- Work on the rendering and engine teams
- Website -> https://kstocky.github.io/

## About Rare

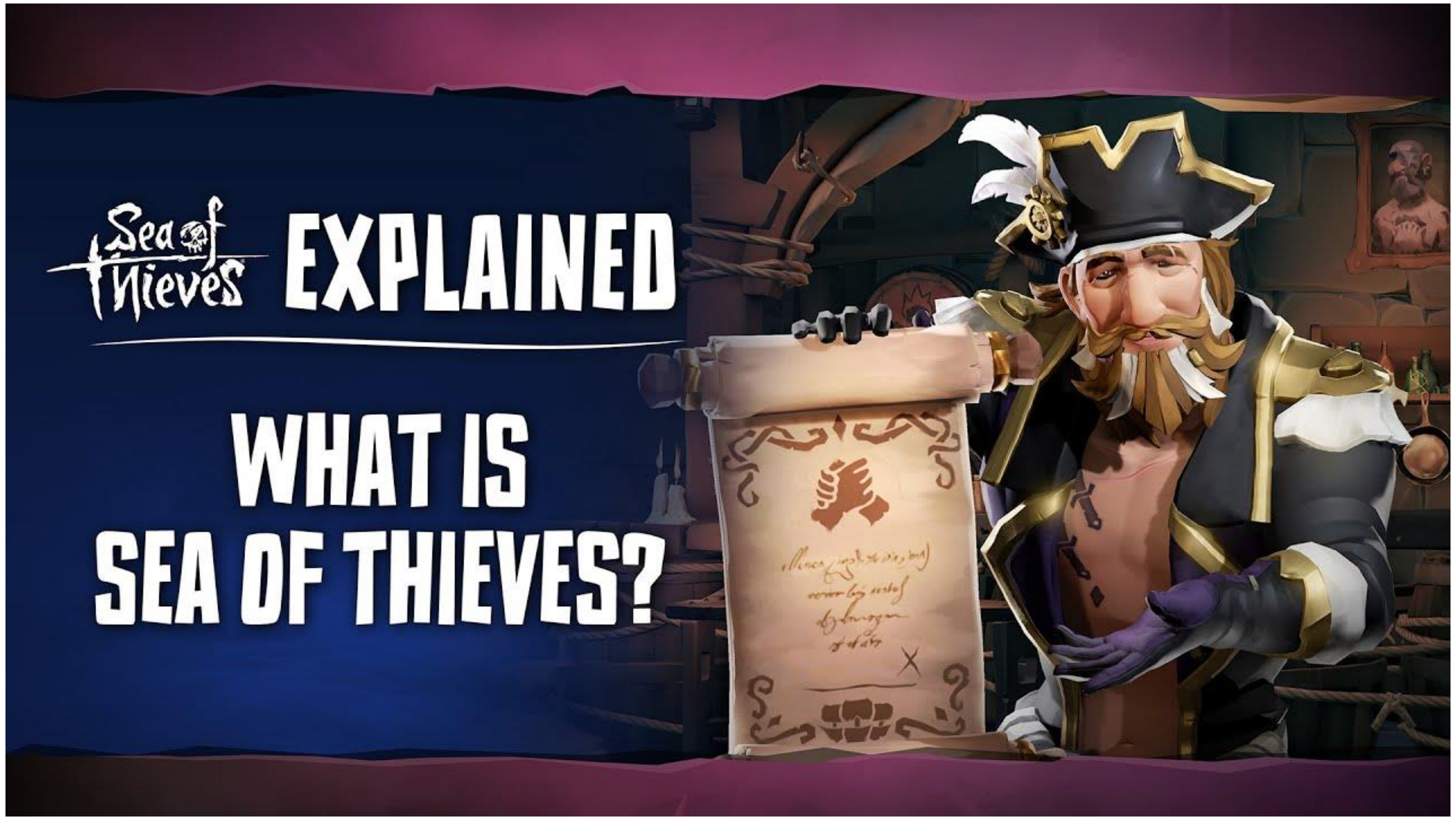
Sea of Thieves

## Agenda

- Talk about Sea of Thieves
- Backstory of the upgrade
- Summarize the motivations
- Planning
- The Fun™ Part
- Conclusions and future work

## What is Sea of Thieves?

## What is Sea of Thieves



## The beginning

Visual Studio 2013

UNREAL ENGINE

4.6

↓

4.10

## Why stop taking new releases?

- Many engine mods
  - About 37,500
- Each release took more time to integrate than the last
- We forked
- Cherry-pick features when it makes sense

## Development continues…

- We release in March 2018!
- And we continue to release new stuff!

## Unreal Engine development continues…

- UE 5.0 Released  April 2022
- Big features like:
  - Nanite
  - Lumen
  - C++17
- UE 5.3 Released in September 2023
- C++20!

Upgrade to C++20

## Improved developer experience

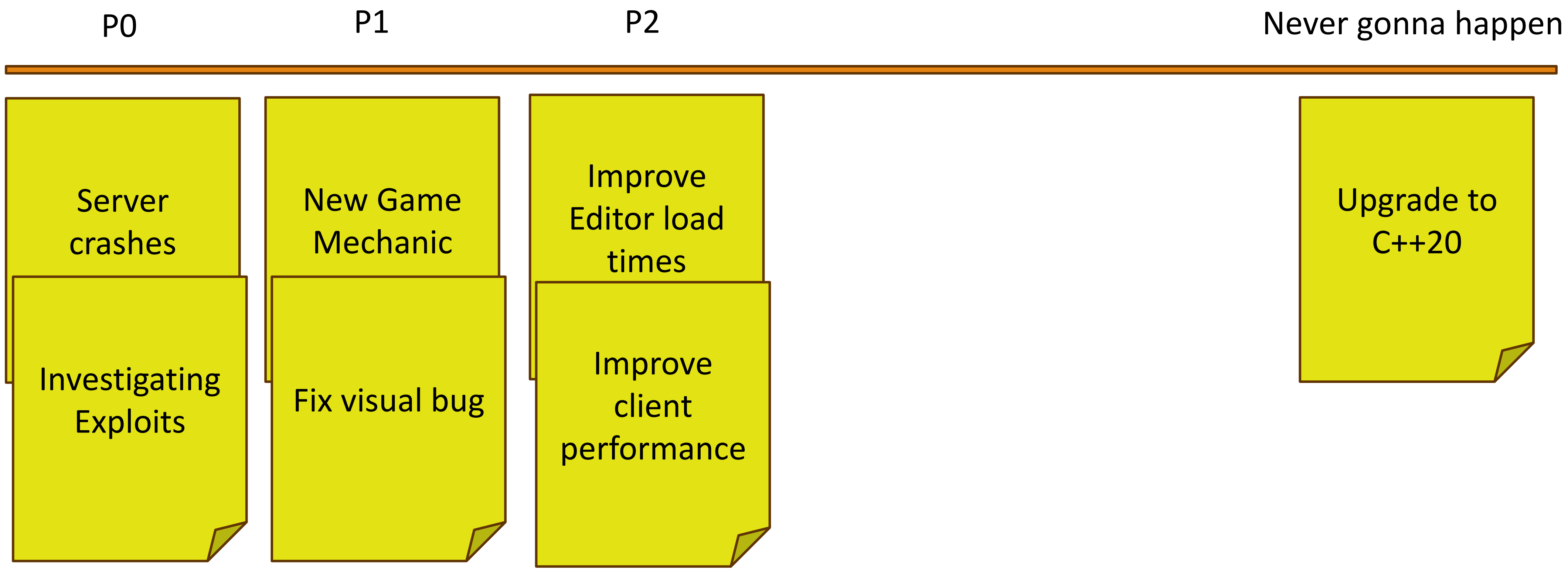
```
for (const auto& [Key, Value] : MyMap)
{

}
```

```
TArray Values{ 1, 2, 3, 4 };
```

```
template<typename T>
concept THasToString = requires(T In)
{
    { In.ToString() } ->
        std::convertible_to<FString>;
};
```

```
MyStruct MyObj
{
    .Name{ TEXT("Foo") },
    .Position{ 2.0, 3.14, 42.0 }
};
```

## But there is a problem...



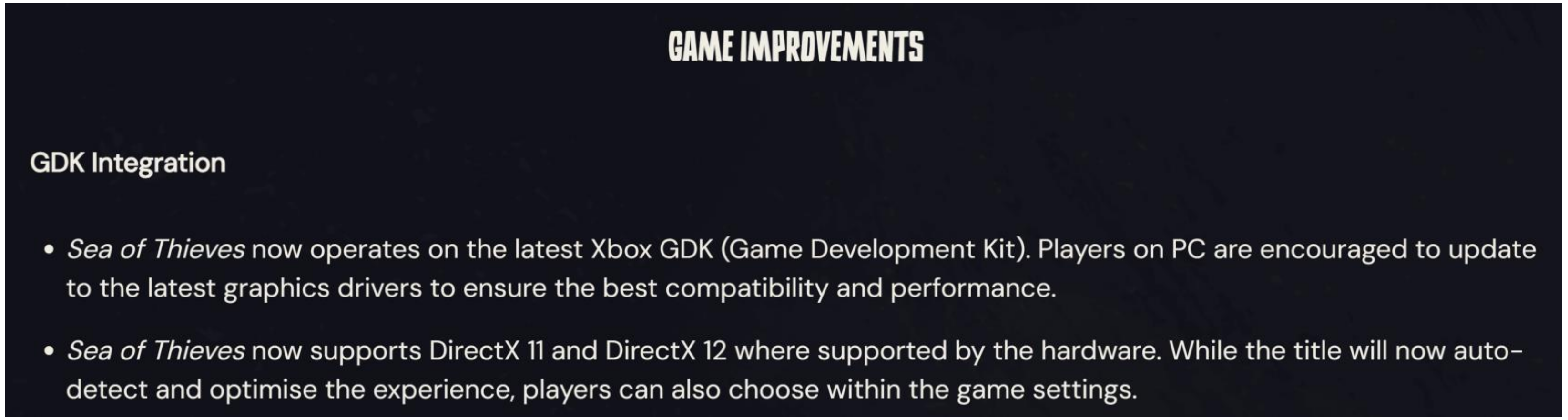
## Hard to prioritize

- C++14 is fine
- Doesn’t directly improve player experience
- Old libraries don’t support C++17/20
- Hard to gauge amount of work required

## Sea of Thieves code base

| | Sea of Thieves project | Engine | Engine Test Project |
|---|---|---|---|
| Number of C++ files | 28,269 | 19,419 | 579 |
| Lines of Code | 2,977,148 | 3,449,028 | 56,057 |

## But then two things happened…


GAME IMPROVEMENTS

GDK Integration

- *Sea of Thieves* now operates on the latest Xbox GDK (Game Development Kit). Players on PC are encouraged to update to the latest graphics drivers to ensure the best compatibility and performance.
- *Sea of Thieves* now supports DirectX 11 and DirectX 12 where supported by the hardware. While the title will now auto-detect and optimise the experience, players can also choose within the game settings.

## But then two things happened…

Sea of Thieves
PREMIUM EDITION
OUT NOW ON PLAYSTATION® 5
XBOX GAME STUDIOS
RARE

## Getting the PS5 port to compile

```
ZeroMemory(&MemoryStats[0], sizeof(MemoryStats));
```

```
#if PLATFORM_PS5
memset((void*)&MemoryStats[0], 0, sizeof(MemoryStats));
#else
ZeroMemory(&MemoryStats[0], sizeof(MemoryStats));
#endif
```

## Why not just fix the code?

### FIX THE CODE

- Pros
  - Minimal Tech Debt
- Cons
  - Could break other platforms
  - Slower

### PREPROCESSOR BRANCHING

- Pros
  - Much less risk in breaking other platforms
  - Faster
- Cons
  - Accrues Tech Debt

# Other kinds of issues arise…

```cpp
template<typename T>
auto DoTheThing()
{
    T::RetType ret = T::Do();
    return ret;
}

struct A
{
    using RetType = int;
    static int Do() { return 42; }
};

int main()
{
    return DoTheThing<A>();
}
```

```cpp
template<typename T>
auto DoTheThing()
{
    T::RetType ret = T::Do();
    return ret;
}

struct A
{
    using RetType = int;
    static int Do() { return 42; }
};

int main()
{
    return DoTheThing<A>();
}
```

## Other kinds of issues arise…

```cpp
template<typename T>
auto DoTheThing()
{
    T::RetType ret = T::Do();
    return ret;
}

struct A
{
    using RetType = int;
    static int Do() { return 42; }
};

int main()
{
    return DoTheThing<A>();
}
```

Dependent name requires `typename`

```cpp
template<typename T>
auto DoTheThing()
{
    typename T::RetType ret = T::Do();
    return ret;
}

struct A
{
    using RetType = int;
    static int Do() { return 42; }
};

int main()
{
    return DoTheThing<A>();
}
```

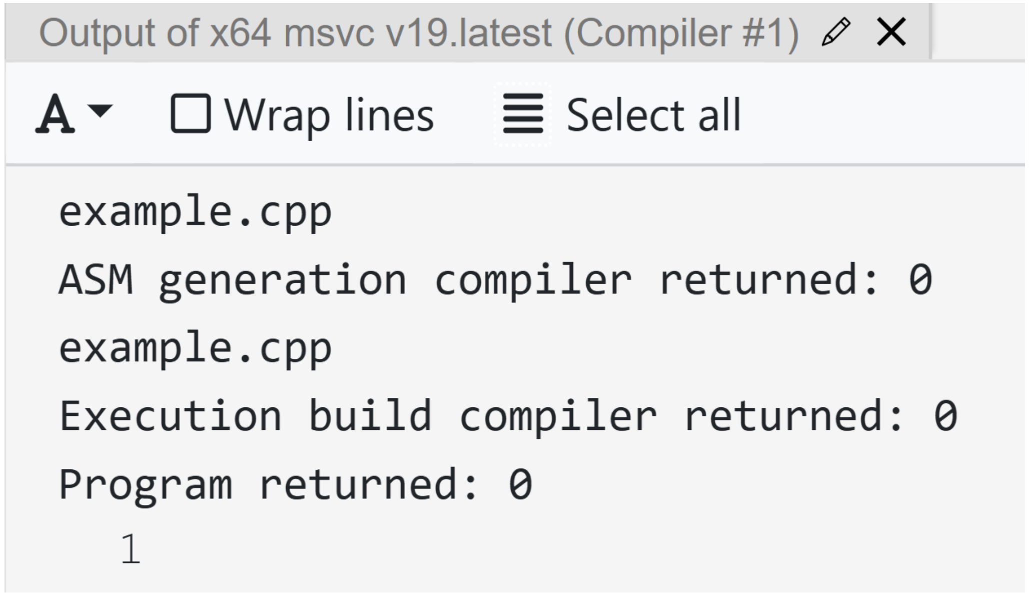
## MSVC “permissive” mode

- Default for C++14 and C++17
- Non-standard conformant C++
- Turned off with /permissive-
- /permissive- is the default in C++20
- https://learn.microsoft.com/en-us/cpp/build/reference/permissive-standards-conformance?view=msvc-170

# Permissive vs Standard: Nested lambdas

```
struct A
{
    int Do(){
        return [this]()
        {
            return [MyThis = this]()
            {
                return sizeof(*MyThis);
            }();
        }();
    }
    bool MyBool;
};

int main(){
    A thing;
    std::cout << thing.Do();
}
```

https://godbolt.org/z/xWhP1vGr7

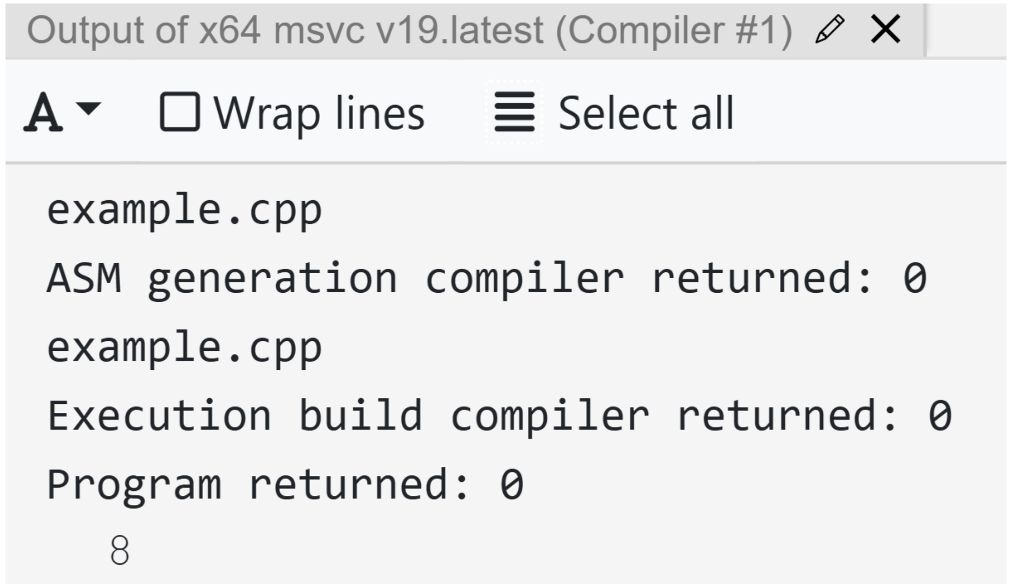


```
example.cpp
ASM generation compiler returned: 0
example.cpp
Execution build compiler returned: 0
Program returned: 0
   1
```

# Permissive vs Standard: Nested lambdas

```
struct A
{
    int Do(){
        return [this]()
        {
            return [MyThis = this]()
            {
                return sizeof(*MyThis);
            }();
        }();
    }
    bool MyBool;
};

int main(){
    A thing;
    std::cout << thing.Do();
}
```

https://godbolt.org/z/qKT55Ke74



```
example.cpp
ASM generation compiler returned: 0
example.cpp
Execution build compiler returned: 0
Program returned: 0
   8
```

## Permissive vs Standard: Copy elision

```
static int Num = 0;
struct Inc
{
    Inc() { ++Num; }
    Inc(const Inc&) { ++Num; }
    Inc(Inc&&) { ++Num; }
    Inc& operator=(const Inc&) { ++Num; return *this; }
    Inc& operator=(Inc&&) { ++Num; return *this; }
    ~Inc() { ++Num; }
};
```

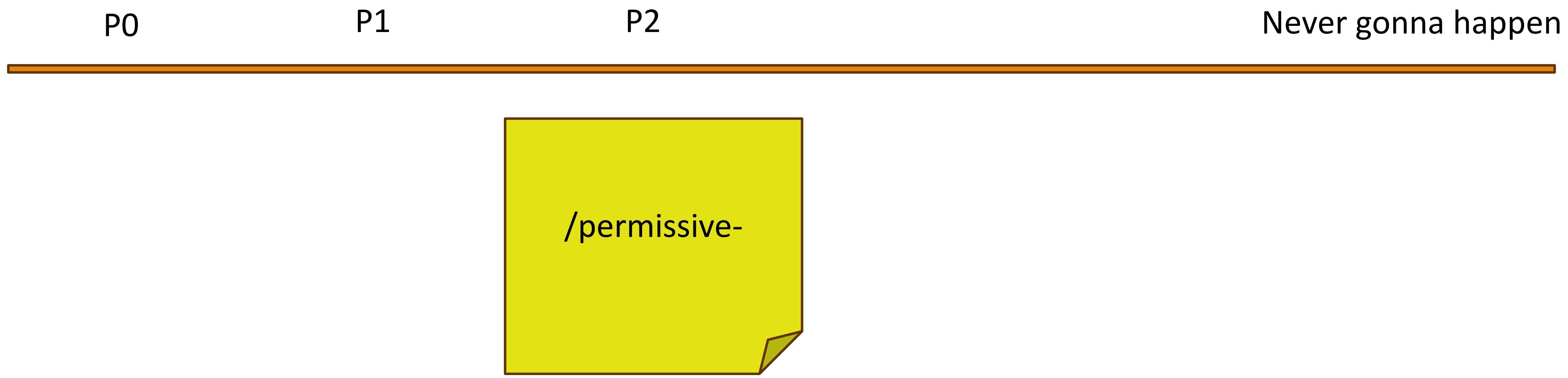
```
int main()
{
    auto Make = []()
    {
        Inc ret;
        return ret;
    };
    auto thing = Make();
    std::cout << Num;
}
```

```
Execution build compiler returned: 0
Program returned: 0
 With Standard C++: 1
```

```
Execution build compiler returned: 0
Program returned: 0
 With Permissive C++: 3
```

# /permissive- is important

- "Permissive C++" is detrimental to our developer experience
- Compilers should interpret code in the same way
- More copy elision is great



## Reprioritizing C++20

- /permissive- should help towards C++20
- Largely the same kind of work
- Flick a switch, then
  - Hit compile
  - Fix compilation errors
  - Repeat until no compilation errors.

## Plan of Action

## Background work

- Sea of Thieves is a game
- New features and perf work take priority
- No deadline

## No “fixing” things

## Small changes

- Divide fixes into small changelists
- Easier to review
- Easier to undo if something breaks
- Enabling compiler switch should be a one-line change

## Step 1: Standards conformance

## Step 2: C++17

- Structured Bindings
- CTAD
- If Constexpr

```
for (const auto& [Key, Value] : MyMap)
{

}

TArray Values{ 1, 2, 3, 4 };

template<typename T>
auto GetThing(T t)
{
    if constexpr (std::is_pointer_v<T>)
        return *t;
    else
        return t;
}
```

## Step 3: C++20

- Concepts

```
template<typename T>
concept THasToString = requires(T In)
{
    { In.ToString() } ->
        std::convertible_to<FString>;
};
```

- Designated Initializers

```
MyStruct MyObj
{
    .Name = TEXT("Foo"),
    .Position = FVector(2.0, 3.14, 42.0)
};
```

- Spaceships!

```
auto operator<=>(const A&, const A&) = default;
```

## Let the Fun™ Begin!

## Two-phase name lookup

- Rules for name resolution in templates
- “Permissive C++” does not follow these rules
- Template bodies only evaluated at instantiation time
- https://devblogs.microsoft.com/cppblog/two-phase-name-lookup-support-comes-to-msvc/

```
template<typename T>
void DoThing(T)
{
    This is not real code!
}

int main()
{
}
```

# Accessing templated base class members

```cpp
template<typename T>
struct Base
{
    void DoTheThing(){}
};

template<typename T>
struct Derived : Base<T>
{
    void Do()
    {
        DoTheThing();
    }
};
```

```cpp
template<typename T>
struct Base
{
    void DoTheThing();
};

template<typename T>
struct Derived : Base<T>
{
    void Do()
    {
        DoTheThing();
    }
};
```

# Accessing templated base class members

```
template<typename T>
struct Base
{
    void DoTheThing(){}
};

template<typename T>
struct Derived : Base<T>
{
    void Do()
    {
        DoTheThing();
    }
};
```

```
template<typename T>
struct Base
{
    void DoTheThing();
};

template<typename T>
struct Derived : Base<T>
{
    void Do()
    {
        this->DoTheThing();
    }
};
```

## What’s a template?

```
template < typename T >
struct Other
{
    template < typename U >
    static void DoTheThing()
    {
    }
};

template<typename T>
void Do()
{
    Other<T>::DoTheThing<T>();
}
```

## What’s a **template**?

```
<source>(13): error C2760: syntax error: ')' was unexpected here; expected 'expression'
<source>(13): error C2760: syntax error: ')' was unexpected here; expected ';'
<source>(13): error C3878: syntax error: unexpected token ')' following 'expression_statement'
<source>(13): note: error recovery skipped: ')'
<source>(13): error C2760: syntax error: '>' was unexpected here; expected 'declaration'
<source>(13): note: error recovery skipped: ')'
```

## What’s a **template**?

```
template < typename T >
struct Outer
{
    template < typename U >
    static void DoTheThing()
    {
    }
};

template<typename T>
void Do()
{
    Outer<T>::DoTheThing<T>();
}
```

Interpreted as an actual “Less Than” symbol

```
template < typename T >
struct Outer
{
    template < typename U >
    static void DoTheThing()
    {
    }
};

template<typename T>
void Do()
{
    Outer<T>::DoTheThing<T>();
}
```

## What’s a **template**?

```
template < typename T >
struct Outer
{
    template < typename U >
    static void DoTheThing()
    {
    }
};

template<typename T>
void Do()
{
    Outer<T>::DoTheThing<T>();
}
```

Interpreted as an actual “Less Than” symbol

```
template < typename T >
struct Outer
{
    template < typename U >
    static void DoTheThing()
    {
    }
};

template<typename T>
void Do()
{
    Outer<T>::template DoTheThing<T>();
}
```

## static_assert(false, “”)

```
template<typename T>
struct OnlyInt
{
    static_assert(false,
        “T must be an int!");
};

template<>
struct OnlyInt<int>
{

};
```

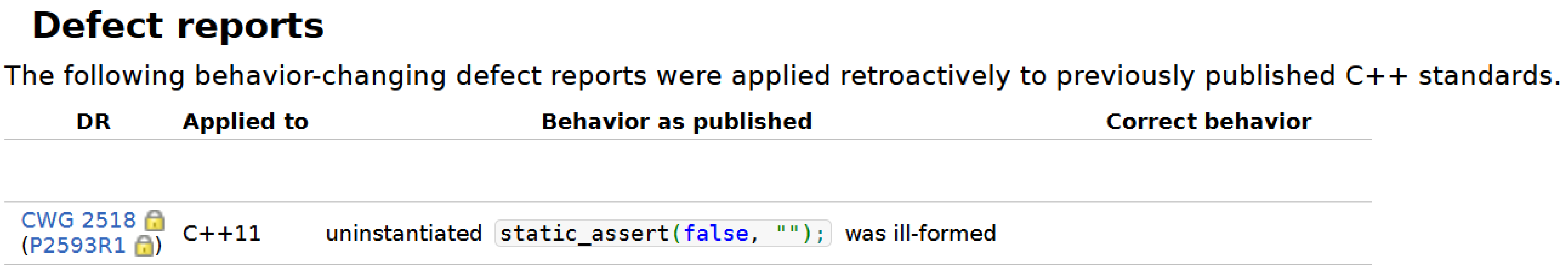
```
template<typename T>
struct OnlyInt
{
    static_assert(false,
        “T must be an int!");
};

template<>
struct OnlyInt<int>
{

};
```

## static_assert(false, “”)

```
template<typename T>
struct OnlyInt
{
    static_assert(false,
        “T must be an int!");
};

template<>
struct OnlyInt<int>
{

};
```

```
template<typename T>
struct OnlyInt
{
    static_assert(TIsAlwaysFalse<T>,
        “T must be an int!");
};

template<>
struct OnlyInt<int>
{

};
```

# Enter P2593/ CWG2518



### Defect reports

The following behavior-changing defect reports were applied retroactively to previously published C++ standards.

| DR | Applied to | Behavior as published | Correct behavior |
|---|---|---|---|
| | | | |
| CWG 2518 (P2593R1) | C++11 | uninstantiated `static_assert(false, "");` was ill-formed | |

- GCC 13
- Clang 17
- MSVC 19.40 / VS 2022 17.10

## Implicit conversions

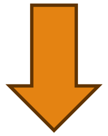
```
enum class IndexEnum
{
    First,
    Second,
    Third,
    Num
};
```

```
int main()
{
    int* MyIntPtr = false;
    int MyInts[IndexEnum::Num] = { 4, 5, 6 };

    int Choice = static_cast<int>(IndexEnum::Second);
    switch (Choice)
    {
        case IndexEnum::First:
            return MyInts[0];
        case IndexEnum::Second:
            return MyInts[1];
        default:
            return MyInts[2];
    }
}
```

# Direct vs Copy Initialization

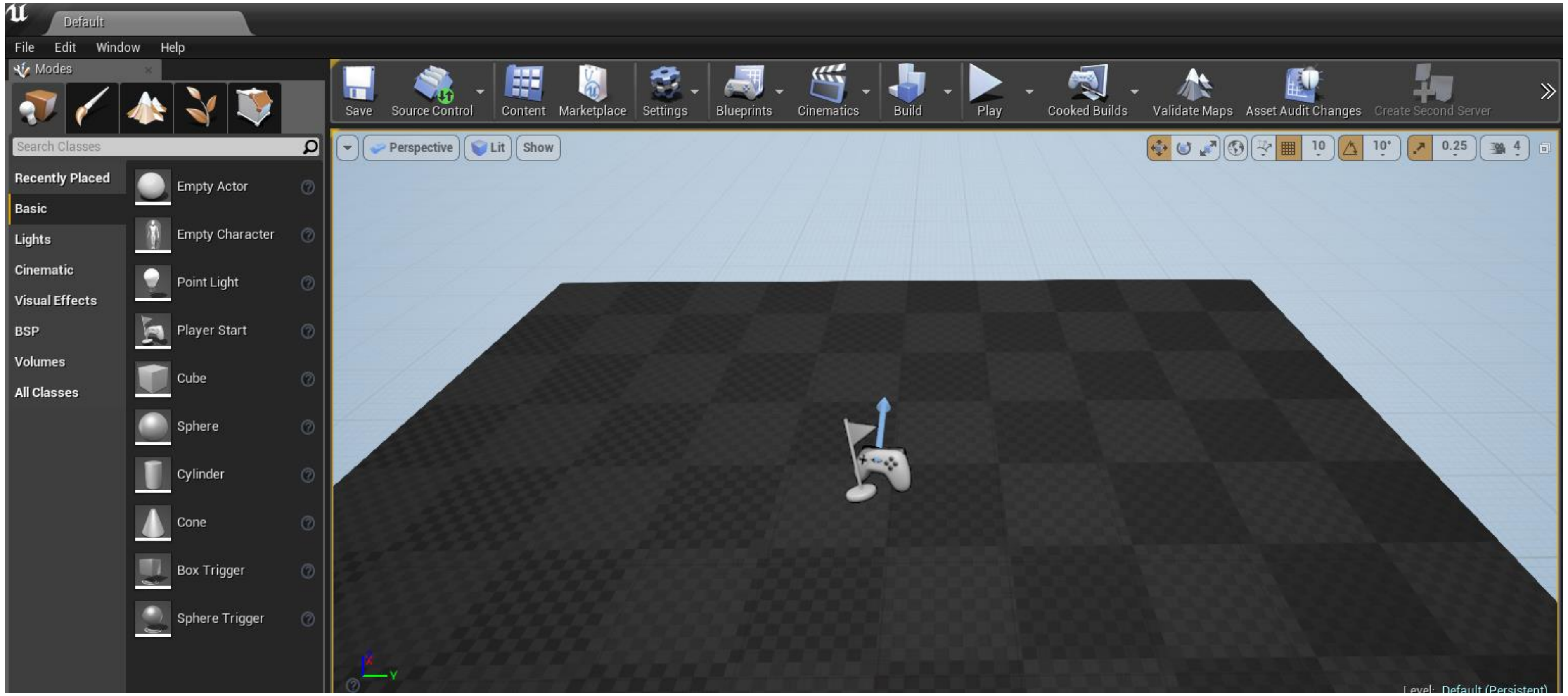
```
struct A {};

struct B
{
    B(A) {}
};

struct C
{
    C(B) {}
};
```

```
int main()
{
    C c = A{};
}
```



```
int main()
{
    C c{ A{} };
}
```

## It compiles!


Default
File
Edit
Window
Help
Modes
Search Classes
Recently Placed
Basic
Lights
Cinematic
Visual Effects
BSP
Volumes
All Classes
Empty Actor
Empty Character
Point Light
Player Start
Cube
Sphere
Cylinder
Cone
Box Trigger
Sphere Trigger
Save
Source Control
Content
Marketplace
Settings
Blueprints
Cinematics
Build
Play
Cooked Builds
Validate Maps
Asset Audit Changes
Create Second Server
Perspective
Lit
Show
10
10°
0.25
4
Level: Default (Persistent)

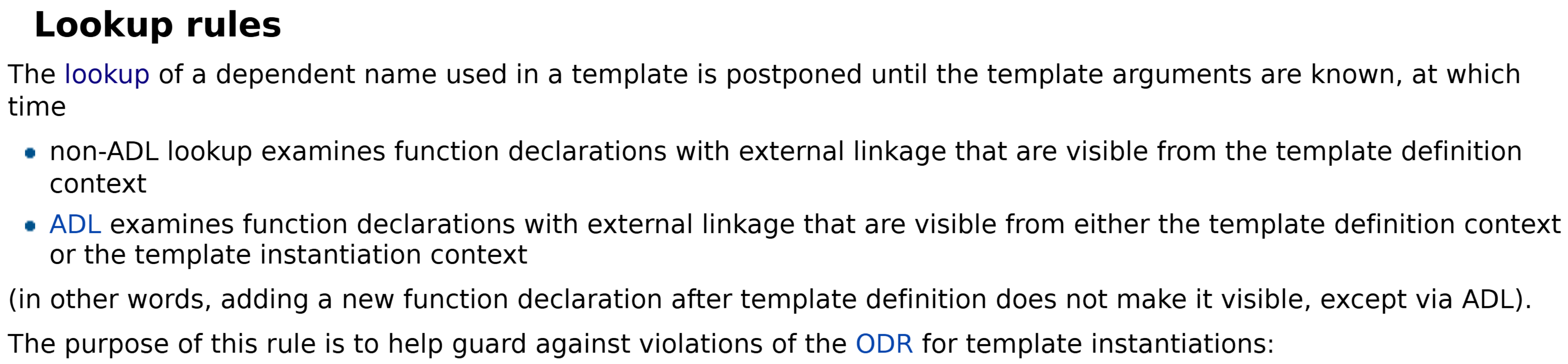
## What about tests?

## Template name lookup

```cpp
template<typename T, typename U>
int GetValueImpl(T, U) {
    return 0;
}

template<typename T>
int GetValue(T In) {
    return GetValueImpl(In, 3.14f);
}

template<typename T>
int GetValueImpl(int, T) {
    return 1;
}
```

```cpp
int main()
{
    return GetValue(42);
}
```

## What’s happening here?



### Lookup rules

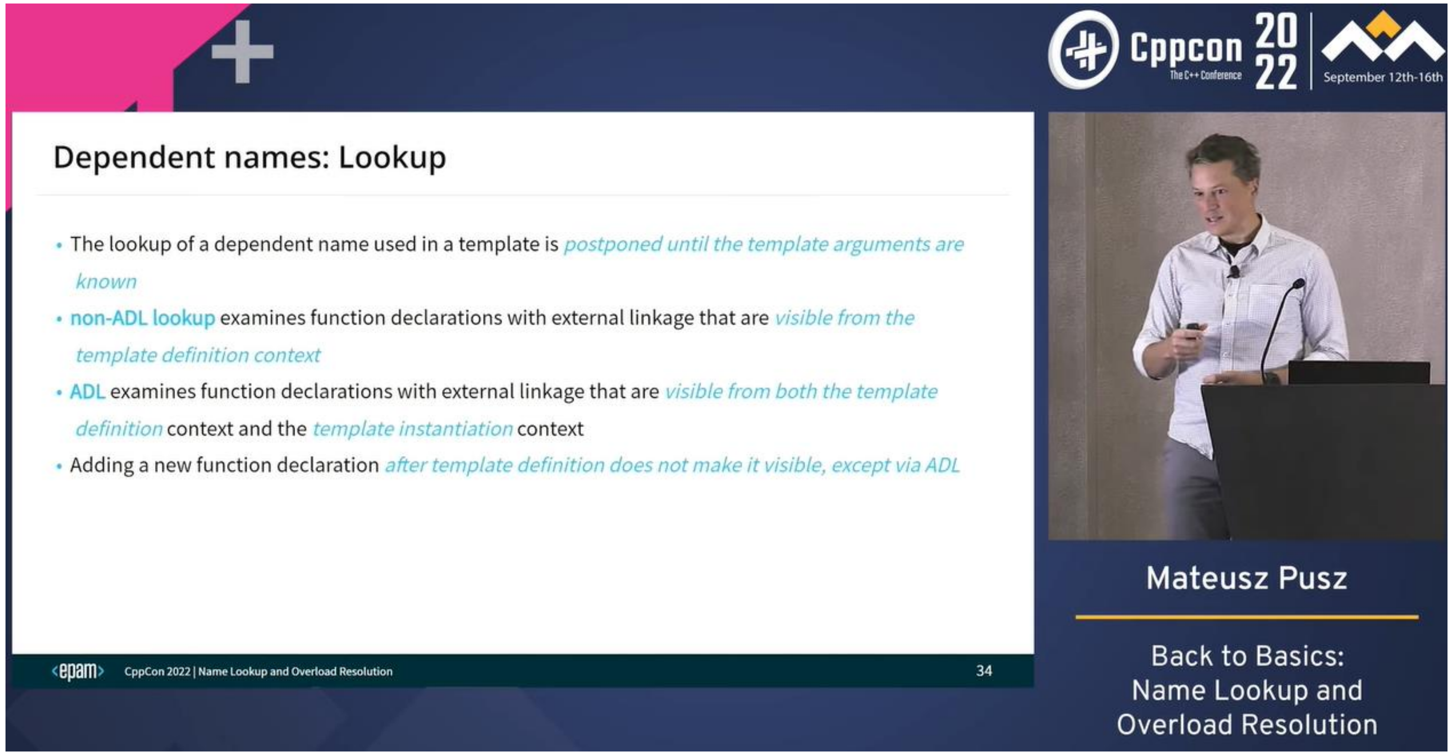
The lookup of a dependent name used in a template is postponed until the template arguments are known, at which time

- non-ADL lookup examines function declarations with external linkage that are visible from the template definition context
- ADL examines function declarations with external linkage that are visible from either the template definition context or the template instantiation context

(in other words, adding a new function declaration after template definition does not make it visible, except via ADL).

The purpose of this rule is to help guard against violations of the ODR for template instantiations:

## Sounds familiar…



Dependent names: Lookup

- The lookup of a dependent name used in a template is *postponed until the template arguments are known*
- non-ADL lookup examines function declarations with external linkage that are *visible from the template definition context*
- ADL examines function declarations with external linkage that are *visible from both the template definition* context and the *template instantiation* context
- Adding a new function declaration *after template definition does not make it visible, except via ADL*

<epam> CppCon 2022 | Name Lookup and Overload Resolution
34
Cppcon The C++ Conference 2022
September 12th-16th
Mateusz Pusz
Back to Basics: Name Lookup and Overload Resolution

## How about now?

## C++17

## std::auto_ptr

```cpp
#include <iostream>
#include <memory>

struct LogAll{
    LogAll(){
        std::cout << "Hello CPPCon\n";
    }

    ~LogAll(){
        std::cout << "Bye CPPCon";
    }
};

int main(){
    std::auto_ptr<LogAll> MyPtr{ new LogAll() };
}
```

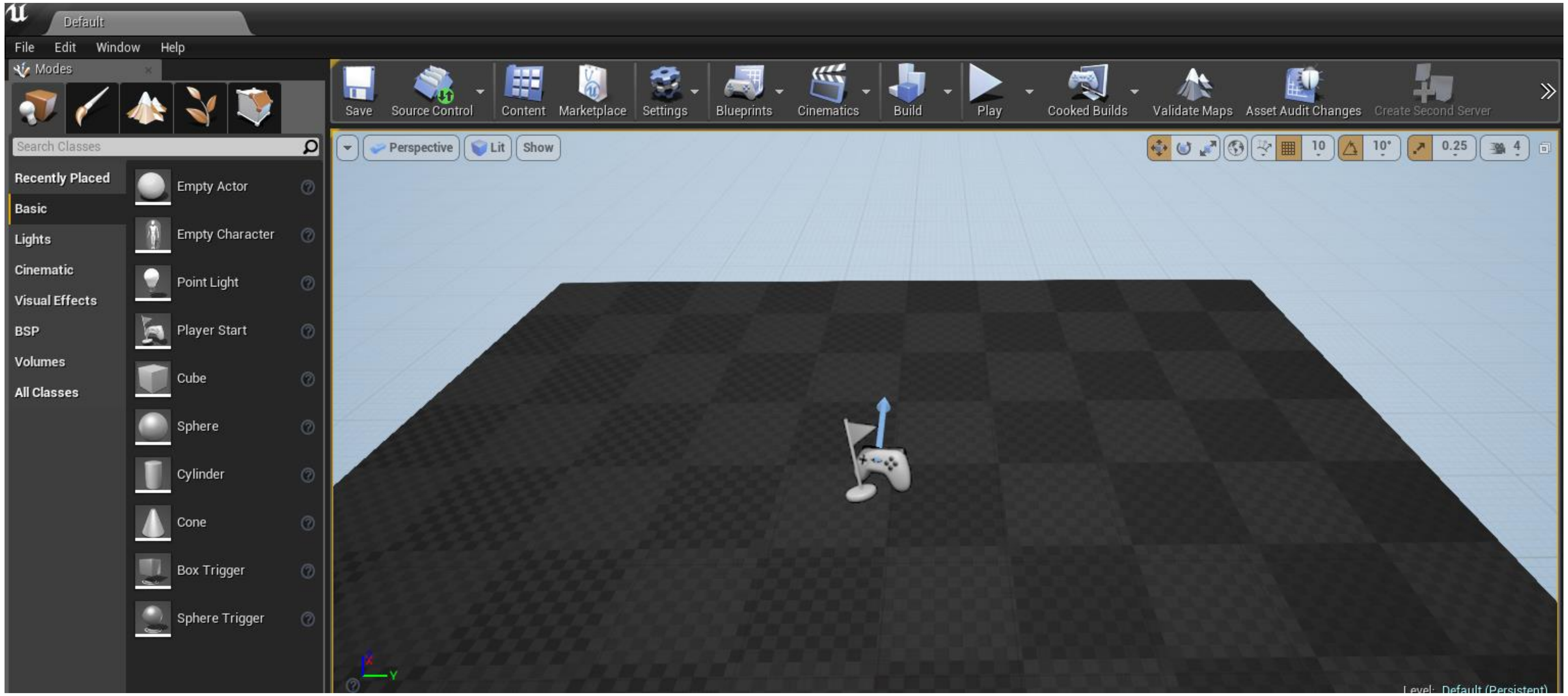
```
Hello CPPCon
Bye CPPCon
```

## Incrementing bools

```
int main()
{
    bool IsTrue = false;
    ++IsTrue;
}
```

```
int main()
{
    bool IsTrue = false;
    IsTrue = true;
}
```

# It compiles!


Default
File
Edit
Window
Help
Modes
Search Classes
Recently Placed
Basic
Lights
Cinematic
Visual Effects
BSP
Volumes
All Classes
Empty Actor
Empty Character
Point Light
Player Start
Cube
Sphere
Cylinder
Cone
Box Trigger
Sphere Trigger
Save
Source Control
Content
Marketplace
Settings
Blueprints
Cinematics
Build
Play
Cooked Builds
Validate Maps
Asset Audit Changes
Create Second Server
Perspective
Lit
Show
10
10°
0.25
4
Level: Default (Persistent)

## What about tests?

❌ ❌

## Expression order fun!

```cpp
#include <iostream>
#include <map>
int main()
{
    std::map<int, int> m;
    m[0] = m.size();
    std::cout << m[0];
}
```

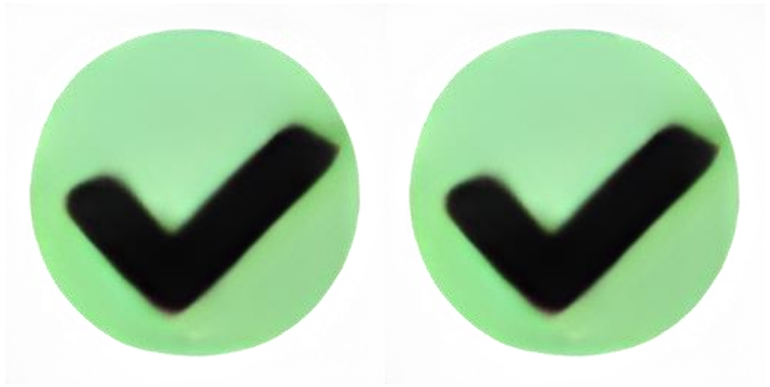
```cpp
#include <iostream>
#include <map>
int main()
{
    std::map<int, int> m;
    m[0] = m.size();
    std::cout << m[0];
}
```

## Expression order fun!

```
#include <iostream>
#include <map>
int main()
{
    std::map<int, int> m;
    m[0] = m.size();
    std::cout << m[0];
}
```

```
#include <iostream>
#include <map>
int main()
{
    std::map<int, int> m;
    auto& element = m[0];
    element = m.size();
    std::cout << m[0];
}
```

## How about now?



# C++20

## Aggregate initialization

```
struct MyStruct
{
    MyStruct() = default;
    int a = 42;
    float b = 3.14f;
};

int main()
{
    MyStruct MyObj{ 42, 3.14f };
}
```

```
struct MyStruct
{
    MyStruct() = default;
    int a = 42;
    float b = 3.14f;
};

int main()
{
    MyStruct MyObj{ 42, 3.14f };
}
```

## Aggregate initialization

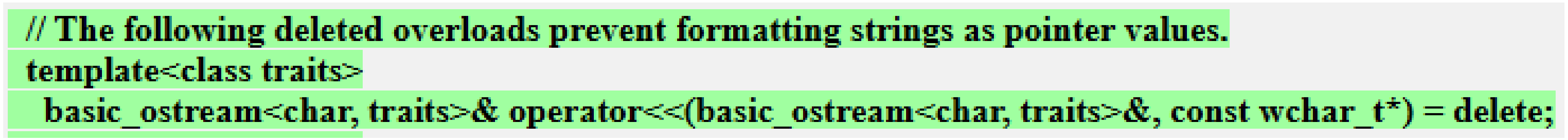
```cpp
struct MyStruct
{
    MyStruct() = default;
    int a = 42;
    float b = 3.14f;
};

int main()
{
    MyStruct MyObj{ 42, 3.14f };
}
```

```cpp
struct MyStruct
{
    int a = 42;
    float b = 3.14f;
};

int main()
{
    MyStruct MyObj{ 42, 3.14f };
}
```

## basic_ostream<char> vs wchar_t*

```cpp
#include <iostream>

int main()
{
    std::cout << L"Hello CPPCon";
}
```

```
Execution build compiler returned: 0
Program returned: 0
 00007FF66CC0D3E0
```



```
// The following deleted overloads prevent formatting strings as pointer values.
template<class traits>
  basic_ostream<char, traits>& operator<<(basic_ostream<char, traits>&, const wchar_t*) = delete;
```

## std::result_of -> std::invoke_result

```
#include <type_traits>

struct MyCallable
{
    bool operator()(int, float) { return true; }
};

static_assert(
    std::is_same_v<std::result_of_t<MyCallable(int, float)>, bool>
);
```

```
static_assert(
    std::is_same_v<std::result_of_t<MyCallable, int, float>, bool>
);
```

## std::result_of -> std::invoke_result

```
#include <type_traits>

struct MyCallable
{
    bool operator()(int, float) { return true; }
};

static_assert(
    std::is_same_v<std::result_of_t<MyCallable(int, float)>, bool>
);

static_assert(
    std::is_same_v<std::invoke_result_t<MyCallable, int, float>, bool>
);
```

## Non-const with comparison operators

```cpp
struct MyStruct
{
    int a = 42;

    bool operator==(const MyStruct& InOther)
    {
        return a == InOther.a;
    }
};
```

# What’s the error?

```
<source>(17): error C2666: 'MyStruct::operator ==': overloaded functions have similar conversions
<source>(6): note: could be 'bool MyStruct::operator ==(const MyStruct &)'
<source>(6): note: or 'bool MyStruct::operator ==(const MyStruct &)' [synthesized expression 'y == x']
<source>(17): note: while trying to match the argument list '(MyStruct, MyStruct)'
```

```
<source>:17:11: warning: C++20 says that these are ambiguous, even though the second is reversed:
17 | a == b;
| ^
<source>:6:10: note: candidate 1: 'bool MyStruct::operator==(const MyStruct&)'
6 | bool operator==(const MyStruct& InOther)
| ^~~~~~~~
<source>:6:10: note: candidate 2: 'bool MyStruct::operator==(const MyStruct&)' (reversed)
<source>:6:10: note: try making the operator a 'const' member function
```

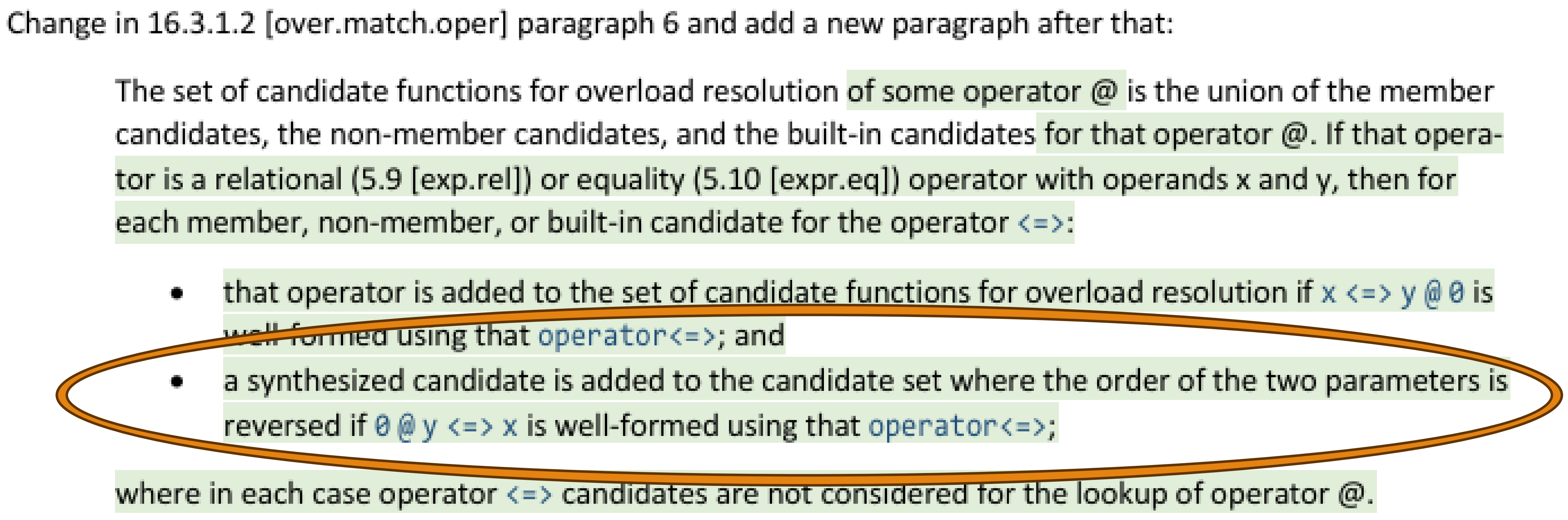
## Non-const comparison operators

```
struct MyStruct
{
    int a = 42;

    bool operator==(const MyStruct& InOther)
    {
        return a == InOther.a;
    }
};
```

# Non-const comparison operators

```
struct MyStruct
{
    int a = 42;

    bool operator==(const MyStruct& InOther) const
    {
        return a == InOther.a;
    }
};
```

# P0515 -> Spaceship operator



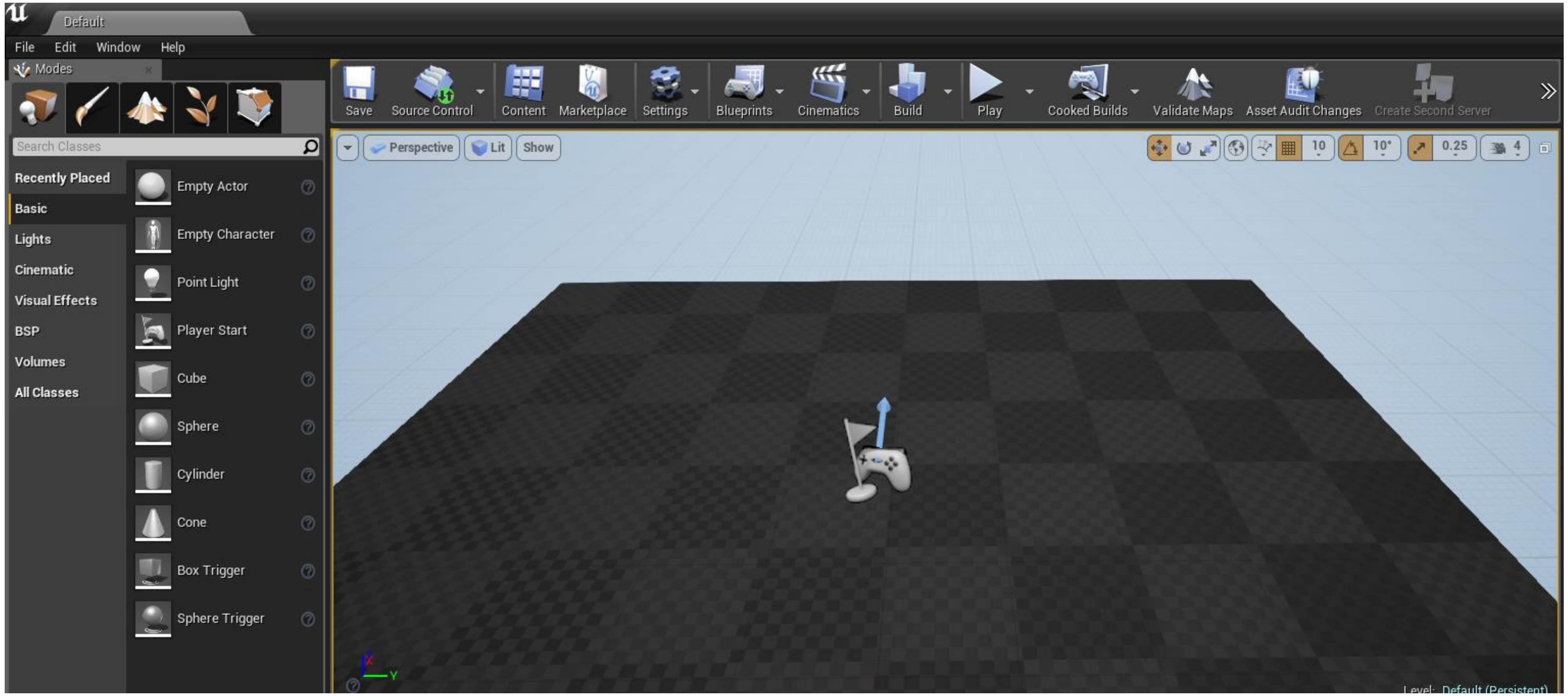
Change in 16.3.1.2 [over.match.oper] paragraph 6 and add a new paragraph after that:

The set of candidate functions for overload resolution of some operator @ is the union of the member candidates, the non-member candidates, and the built-in candidates for that operator @. If that operator is a relational (5.9 [exp.rel]) or equality (5.10 [expr.eq]) operator with operands x and y, then for each member, non-member, or built-in candidate for the operator `<=>`:

- that operator is added to the set of candidate functions for overload resolution if `x <=> y @ 0` is well-formed using that `operator<=>`; and
- a synthesized candidate is added to the candidate set where the order of the two parameters is reversed if `0 @ y <=> x` is well-formed using that `operator<=>`;

where in each case operator `<=>` candidates are not considered for the lookup of operator @.

# It compiles!


Default
File
Edit
Window
Help
Modes
Save
Source Control
Content
Marketplace
Settings
Blueprints
Cinematics
Build
Play
Cooked Builds
Validate Maps
Asset Audit Changes
Create Second Server
Search Classes
Perspective
Lit
Show
10
10°
0.25
4
Recently Placed
Basic
Lights
Cinematic
Visual Effects
BSP
Volumes
All Classes
Empty Actor
Empty Character
Point Light
Player Start
Cube
Sphere
Cylinder
Cone
Box Trigger
Sphere Trigger
Level: Default (Persistent)

## What about tests?

## Upgrade done?

## Comparing arrays

```
int main()
{
    int a[] = { 1,2,3 };
    int b[] = { 1,2,3 };

    return a == b ? 1 : 2;
}
```

```
int main()
{
    int a[] = { 1,2,3 };
    int b[] = { 1,2,3 };

    return a == b ? 1 : 2;
}
```

## Comparing arrays

```
int main()
{
    int a[] = { 1,2,3 };
    int b[] = { 1,2,3 };

    return a == b ? 1 : 2;
}
```

```
int main()
{
    int a[] = { 1,2,3 };
    int b[] = { 1,2,3 };

    return &a == &b ? 1 : 2;
}
```

Fix comparison of arrays

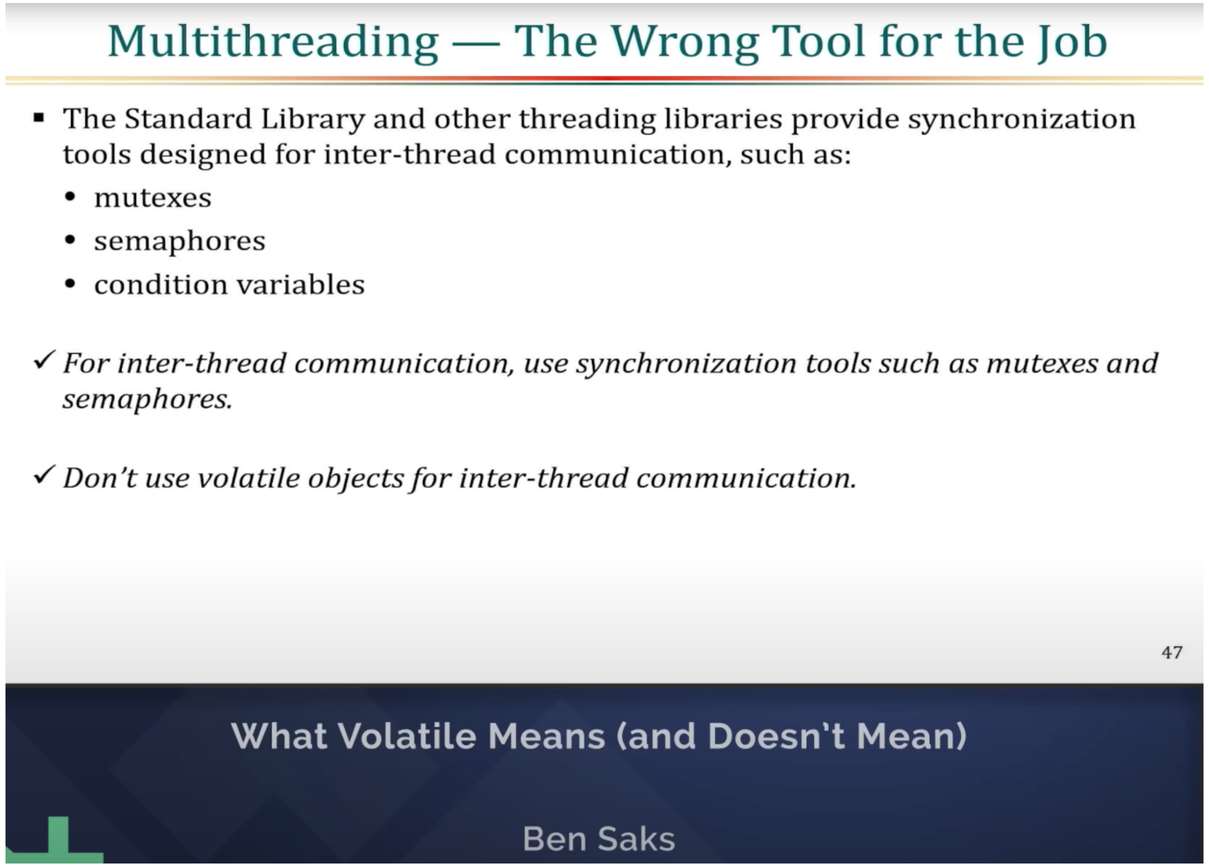
## Volatile = Threadsafe?

```
static volatile int NumThreads = 0;
void MyThreadFunc()
{
    ++NumThreads;
}
```

```
static volatile int NumThreads = 0;
void MyThreadFunc()
{
    NumThreads += 1;
}
```

Investigate volatile usage

# Volatile = threadsafe?



## Multithreading — The Wrong Tool for the Job

- The Standard Library and other threading libraries provide synchronization tools designed for inter-thread communication, such as:
  - mutexes
  - semaphores
  - condition variables

✓ *For inter-thread communication, use synchronization tools such as mutexes and semaphores.*

✓ *Don't use volatile objects for inter-thread communication.*

47

What Volatile Means (and Doesn't Mean)

Ben Saks

## Capturing **this** with **=**

```
struct MyStruct
{
    int ReturnTheThingImpl(int InVal)
    {
        return InVal;
    }

    int ReturnTheThing(int InVal)
    {
        return [=]()
        {
            return ReturnTheThingImpl(InVal);
        }();
    }
};
```

## Capturing **this** with **=**

```
struct MyStruct
{
    int ReturnTheThingImpl(int InVal)
    {
        return InVal;
    }

    int ReturnTheThing(int InVal)
    {
        return [=, this]()
        {
            return ReturnTheThingImpl(InVal);
        }();
    }
};
```

## Capturing `this` with `=`

```
struct MyStruct
{
    int ReturnTheThingImpl(int InVal)
    {
        return InVal;
    }

    int ReturnTheThing(int InVal)
    {
        return [this, InVal]()
        {
            return ReturnTheThingImpl(InVal);
        }();
    }
};
```

It's done.

## Sea of Thieves code base

| | Sea of Thieves project | Engine | Engine Test Project |
|---|---|---|---|
| Number of C++ files | 28,269 | 19,419 | 579 |
| Lines of Code | 2,977,148 | 3,449,028 | 56,057 |

## Upgrade statistics

| | /permissive- | /std:c++17 | /std:c++20 |
|---|---|---|---|
| Number of Files changed/added | 215 | 68 | 111 |
| Number of individual changes | 369 | 31 | 408 |
| Number of Changelists | 30 | 4 | 18 |
| Number of tests to fix | 6 | 2 | 0 |

## So how hard was it?

| | Time to first successful local compilation | Time to submit switch |
|---|---|---|
| Enabling /permissive- | 2 days | ~4 months |
| /std:c++14 -> /std:c++17 | 3/4 hours | 2 weeks |
| /std:c++17 -> /std:c++20 | 1 day | ~3 months |

# Build times

PC Specs:

AMD Ryzen Threadripper PRO 3975WX 32-Cores 3.50GHZ

128GB RAM

Build Config: Development Editor

MSVC: 19.42

| | /std:c++17 (minutes:seconds) | /std:c++17 /permissive- (minutes:seconds) | /std:c++20 (minutes:seconds) |
|---|---|---|---|
| Rebuild 1 | 13:22.1 | 12:53.0 | 12:45.4 |
| Rebuild 2 | 13:01.9 | 12:44.4 | 12:48.9 |
| Rebuild 3 | 12:46.3 | 12:48.4 | 12:45.7 |
| Rebuild 4 | 12:44.6 | 12:39.9 | 12:50.8 |
| Mean | 12:58.7 | 12:46.4 | 12:47.7 |

## What did we achieve?

- Sea of Thieves is even more cross platform
- Faster debug builds
- Fixed test suite error reporting
- Improved error detection
- Got some great new features!

## Devs love the new features

- Sam was our previous Engine team intern

- “I just tried to use designated initializers but then I realized I was on Sea of Thieves. Very much looking forward to getting C++20”

- “Sea of Thieves is now on C++20 so let's upgrade!
I'm already making use of C++20 concepts for unit testing”

## Takeaways

- Upgrading is worth a try
- Newer standards improve code quality
- I learned A LOT
- Compile with multiple compilers
- Automated tests are exceptionally helpful.
- Build times don’t get worse

## Future work

- Removing more #if PLATFORM_PS5
- Enabling warnings as errors
- Re-enable warnings that were disabled
- C++23 and beyond!

## Q&A

## About me!

- Senior Software Engineer at Rare Ltd
- Started as an intern in 2015
- Work on the rendering and engine teams
- Website -> [https://kstocky.github.io/](https://kstocky.github.io/)