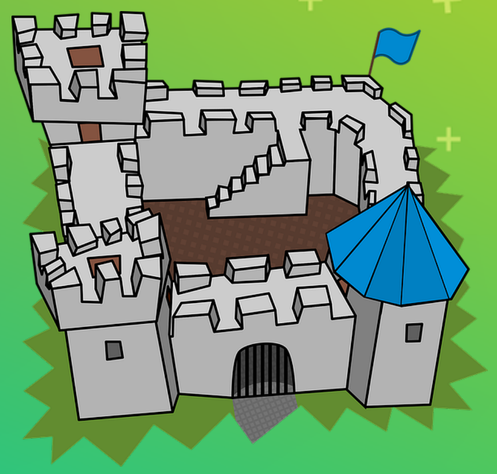
A C++ Ride Through the Gamedev Universe



Anastasia Kazakova

@anastasiak2512

anastasia.kazakova@jetbrains.com

Talents In Games 2020

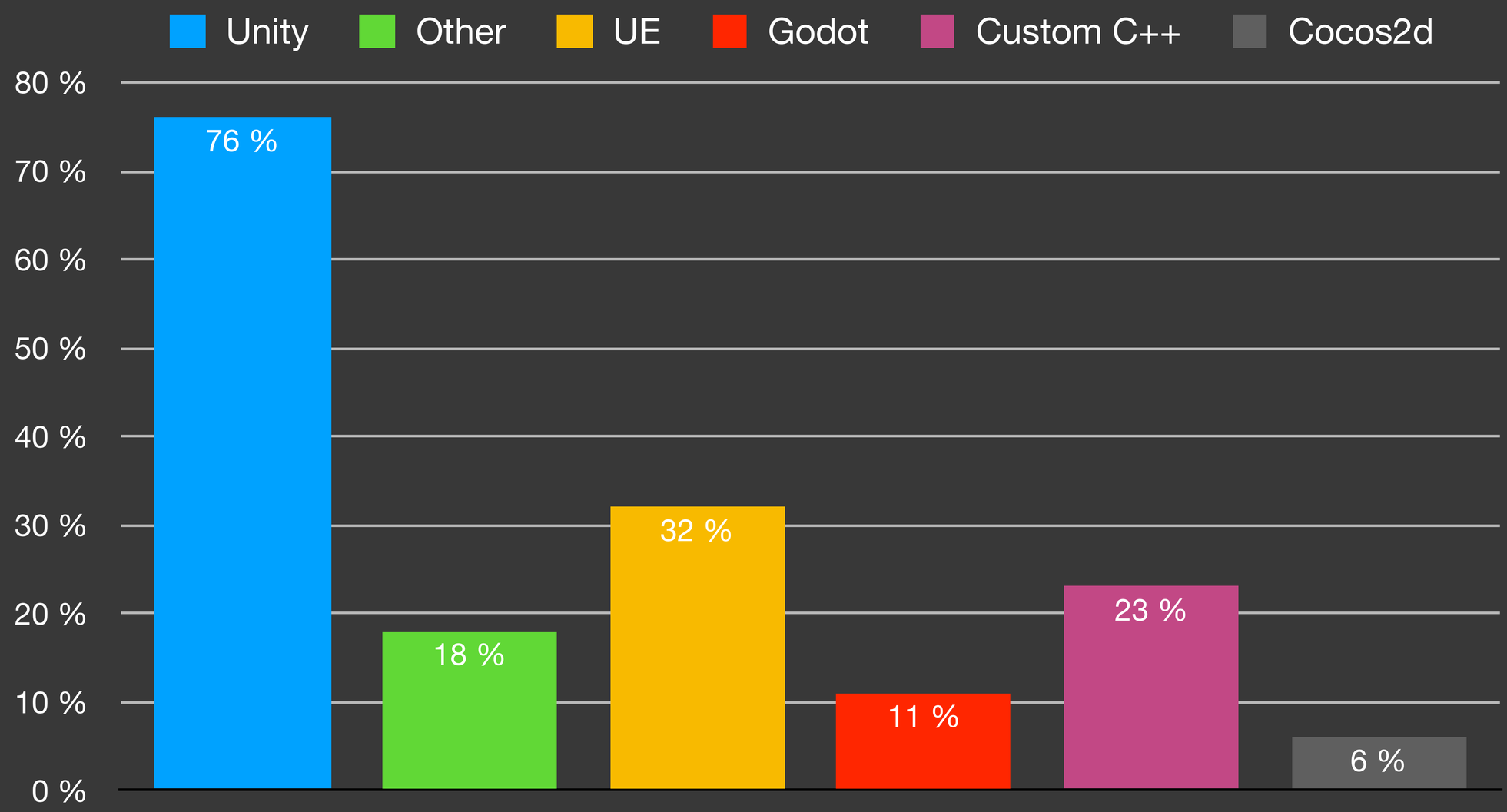
Intro

- From C++ developer to C++ & .NET tools marketing lead
- OKTET Labs, Microsoft Research, Yota / RooX, JetBrains
- St. Petersburg C++ User Group
  - https://www.meetup.com/St-Petersburg-CPP-User-Group/
- CLion, Rider & Rider for Unreal Engine

Top C++ area

- Finances / Banking / Trading (HFT)
- Embedded systems
- Game Dev
- Machine Learning, Networking, AI, ...

GameDev & C++


Unity
Other
UE
Godot
Custom C++
Cocos2d
80 %
70 %
60 %
50 %
40 %
30 %
20 %
10 %
0 %
76 %
18 %
32 %
11 %
23 %
6 %

C++ in General

- Release every 3 year
- Current C++20: Concepts, Coroutines, Modules, and much more
- SG14 (the GameDev & low latency ISO C++ working group)
- For C++23: Reflection

C++ in Games

- Windows platform.
- AAA games are mostly C++, custom engines.
- Rendering is in C.
- SDK for consoles.
- Reflection mechanisms.
- Specific allocators (InplaceArray<ubi32, 8>).
- Specific STL (EASTL, etc.).

Powerful and challenging C++

```
#define X(a) myVal_##a,
enum myShinyEnum {
#include "xmacro.txt"
};
#undef X

void handle_value(myShinyEnum en) {
    switch (en) {
        case myVal_a:break;
        case myVal_b:break;
        case myVal_c:break;
        case myVal_d:break;
    }
}
```

```
//xmacro.txt

X(a)
X(b)
X(c)
X(d)
```

Powerful and challenging C++

```
#define X(a) myVal_##a,
enum myShinyEnum {
#include "xmacro.txt"
};
#undef X

void handle_value(myShinyEnum en) {
    switch (en) {
        case myVal_a:break;
        case myVal_b:break;
        case myVal_c:break;
        case myVal_d:break;
    }
}
```

```
//xmacro.txt

X(a)
X(b)
X(c)
X(d)
```

C++ in Games

```
UCLASS(config=Game)
class APremiumGameProjectile : public AActor
{
  GENERATED_BODY()

  /** Sphere collision component */
  UPROPERTY(VisibleDefaultsOnly, Category=Projectile)
  class USphereComponent* CollisionComp;

public:
  APremiumGameProjectile();

  /** called when projectile hits something */
  UFUNCTION()
  void OnHit(UPrimitiveComponent* HitComp, AActor* OtherActor,
             UPrimitiveComponent* OtherComp, FVector NormalImpulse,
             const FHitResult& Hit);
   //…
}
```

C++ in Games

```
UCLASS(config=Game)
class APremiumGameProjectile : public AActor
{
  GENERATED_BODY()

  /** Sphere collision component */
  UPROPERTY(VisibleDefaultsOnly, Category=Projectile)
  class USphereComponent* CollisionComp;

public:
  APremiumGameProjectile();

  /** called when projectile hits something */
  UFUNCTION()
  void OnHit(UPrimitiveComponent* HitComp, AActor* OtherActor,
             UPrimitiveComponent* OtherComp, FVector NormalImpulse,
             const FHitResult& Hit);
  //…
}
```

Powerful and challenging C++

```
template <class T> int foo(T & t)   { return 1; }
template <class T> int foo(T && t)  { return 2; }
int foo(signed char t)              { return 3; }
int foo(unsigned char t)            { return 4; }
int foo(int)                        { return 5; }

int main() {
    auto C = 'C';
    std::cout << foo(C) << foo('C')
              << foo('C++') << foo("C++");
}
```

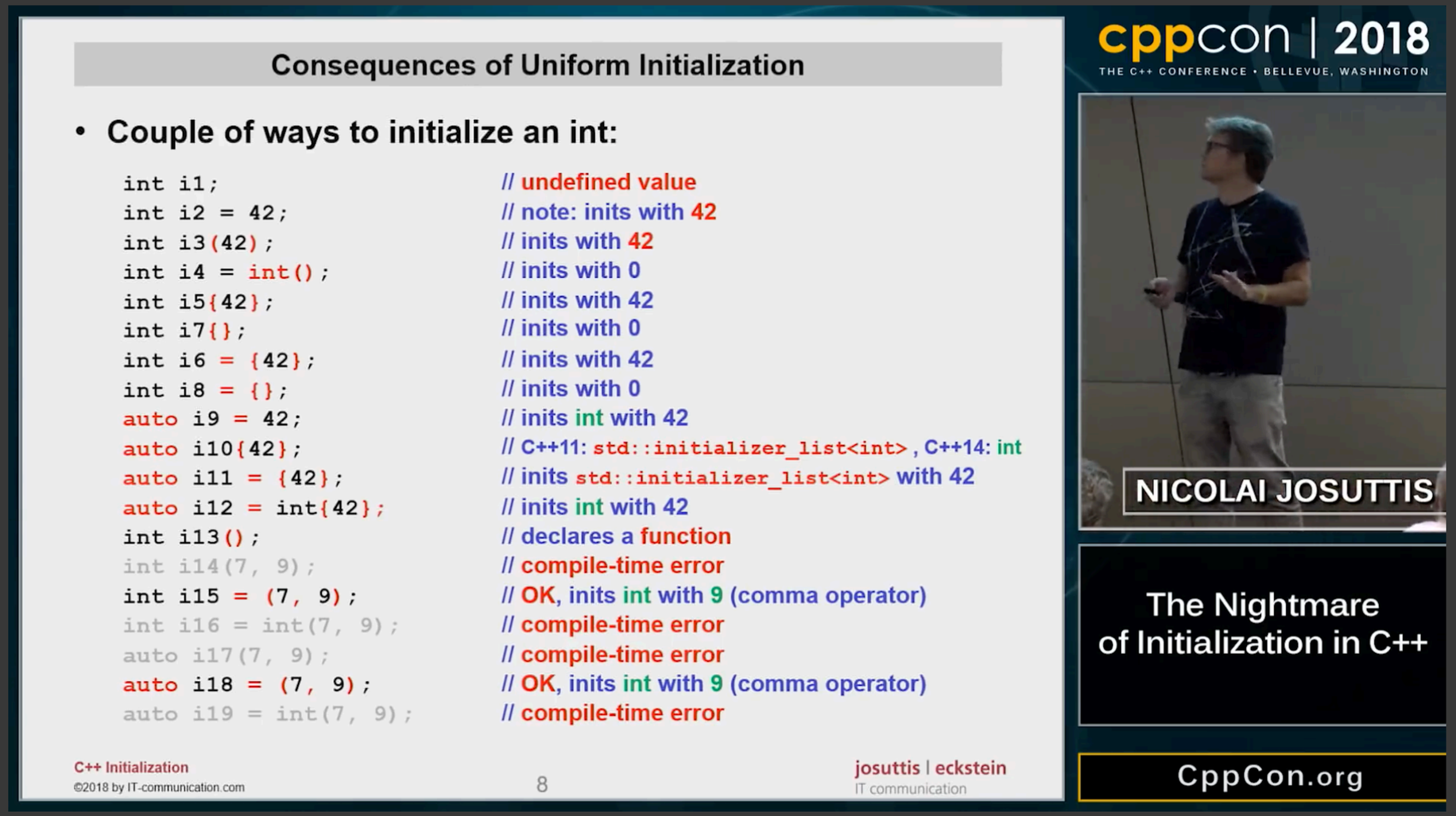
Powerful and challenging C++

```
std::ostream& operator<<(std::ostream& out, const Fraction& f) {
    return out << f.num() << '/' << f.den();
}

void fraction_sample() {
    Fraction f1(3, 8), f2(1, 2);

    std::cout << f1 << " * " << f2 << " = " << f1 * f2 << '\n';
}
```

Powerful and challenging C++



Powerful and challenging C++

“With a sufficient number of uses of an API, it does not matter what you promise in the contract: all observable behaviours of your system will be depended on by somebody.”

(Hyrums Law,

Software Engineering at Google,

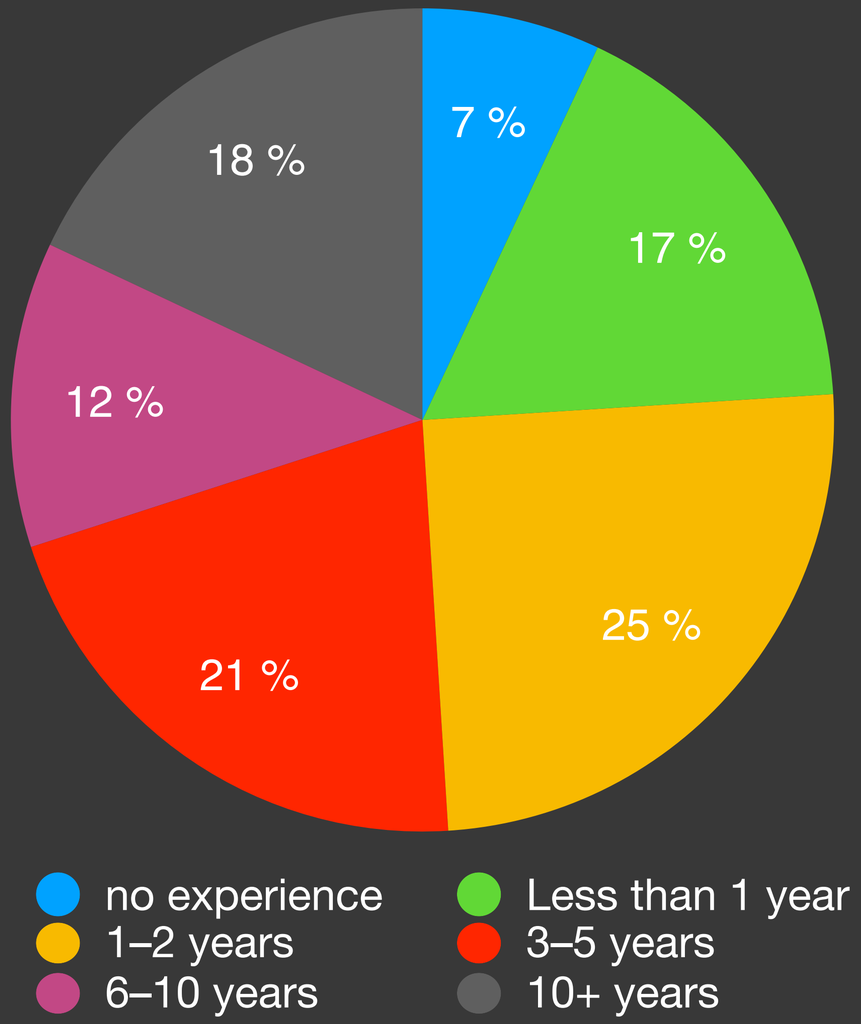
by Titus Winter, Tom Manshrek, Hyrum Wright)

Ecosystem researches

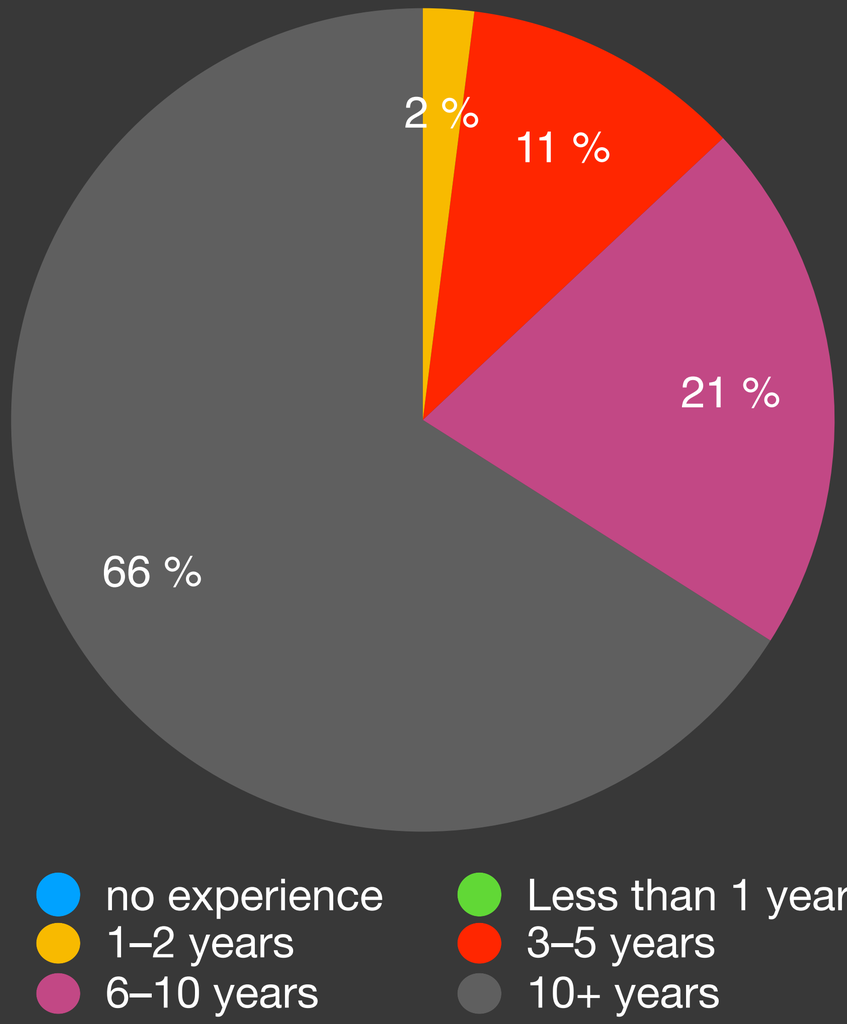
- C++ Foundation “Lite” Survey
  - 1000+
- JetBrains Developers Ecosystem
  - 1800+ / 19000+

Ecosystem researches

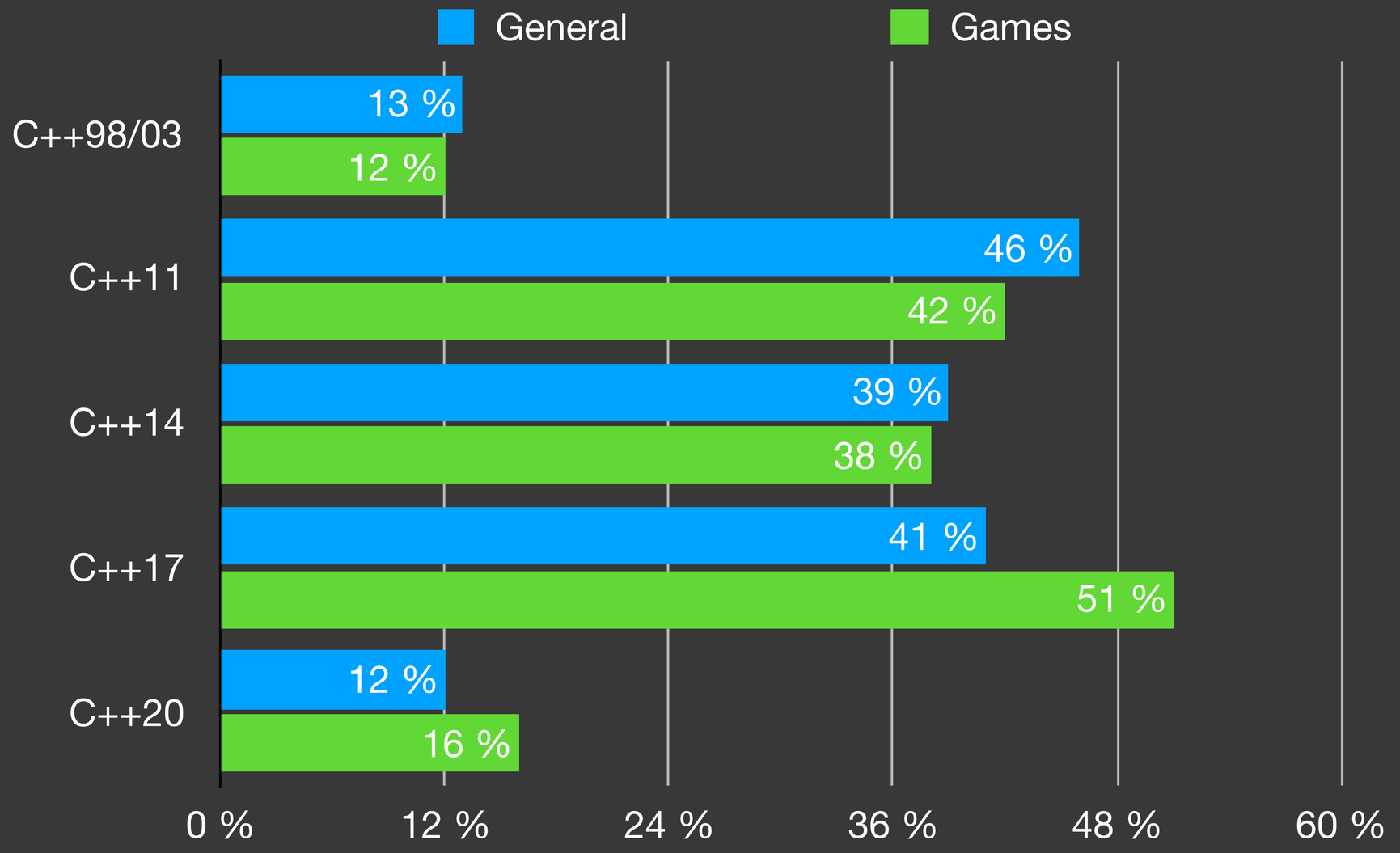
Dev Eco audience 2020



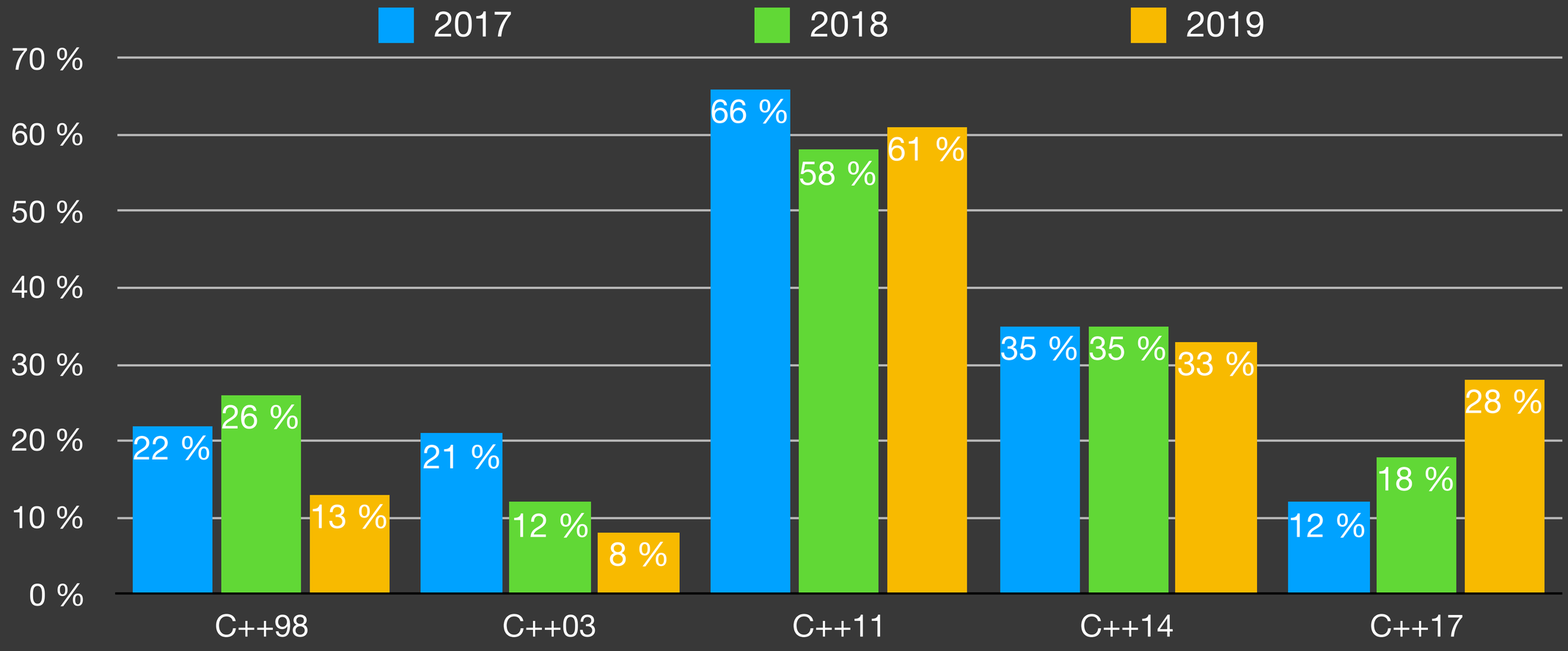
C++ Foundation Lite audience 2020



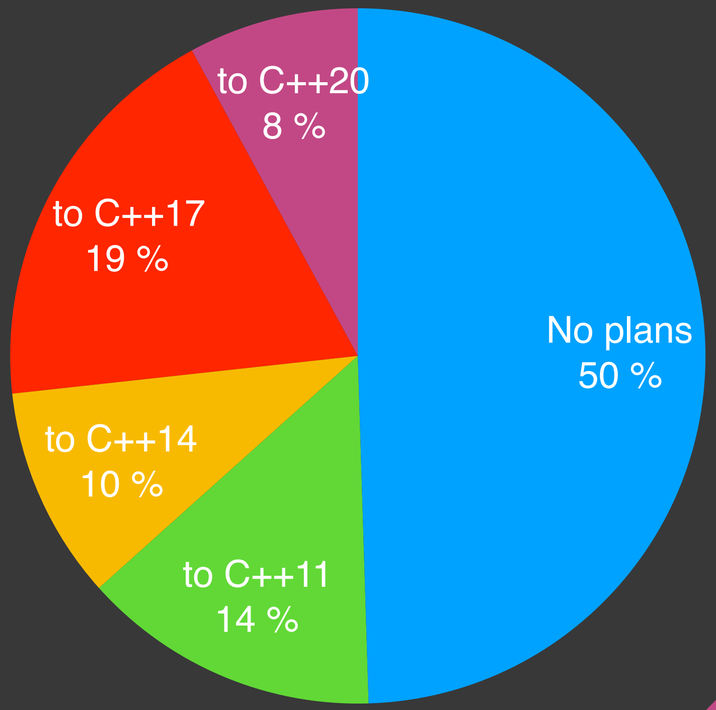
C++ in 2020


General
Games
C++98/03
13 %
12 %
C++11
46 %
42 %
C++14
39 %
38 %
C++17
41 %
51 %
C++20
12 %
16 %
0 %
12 %
24 %
36 %
48 %
60 %

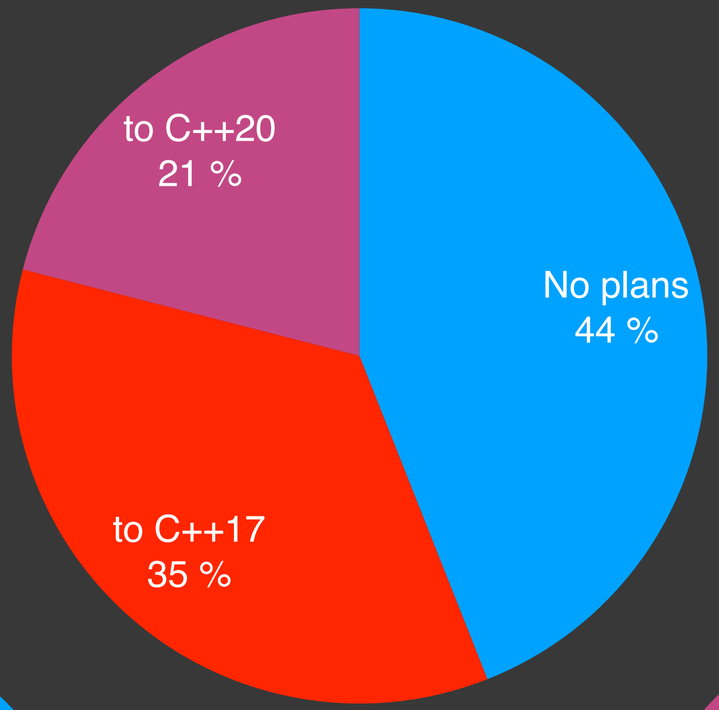
C++ in 2020


2017
2018
2019
70 %
60 %
50 %
40 %
30 %
20 %
10 %
0 %
22 %
26 %
13 %
21 %
12 %
8 %
66 %
58 %
61 %
35 %
35 %
33 %
12 %
18 %
28 %
C++98
C++03
C++11
C++14
C++17

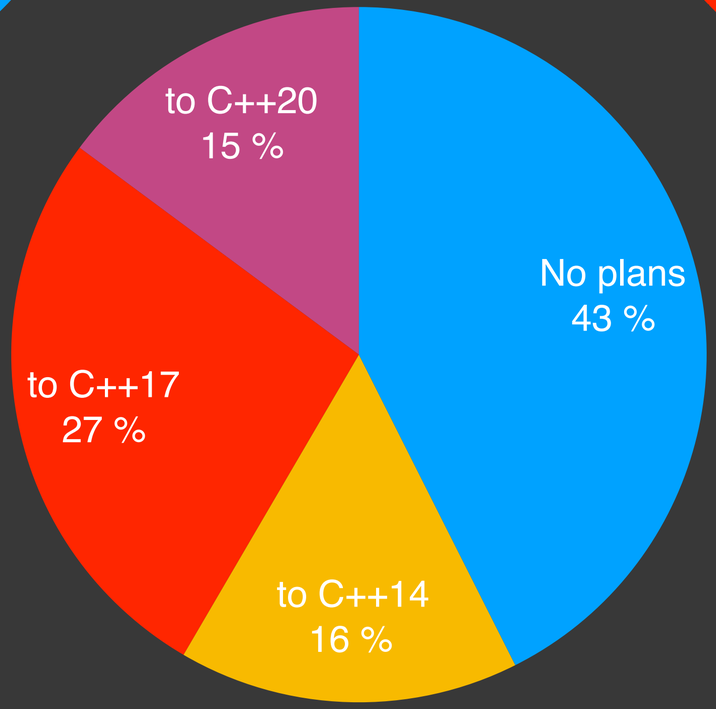
From C++98/03



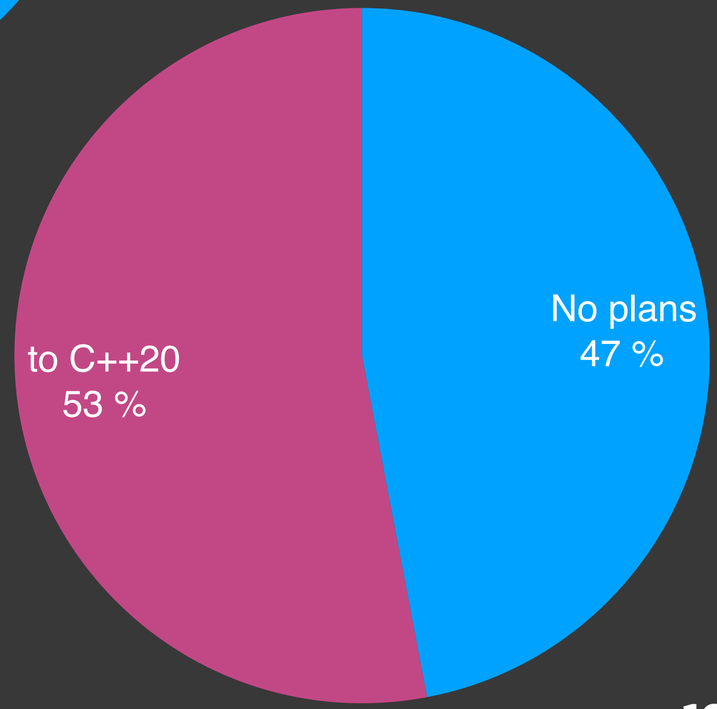
From C++14



From C++11



From C++17



C++ in 2020

“Moving existing code to the latest language standard”
– major pain point for 7%
(C++ Foundation “Lite” 2020)

C++ in 2020

1. “Managing libraries my application depends on”

– major pain point for 47%

2. “Build times”

– major pain point for 42%

3. “Setting up a continuous integration pipeline from scratch (automated builds, tests, …) ”

– major pain point for 32%

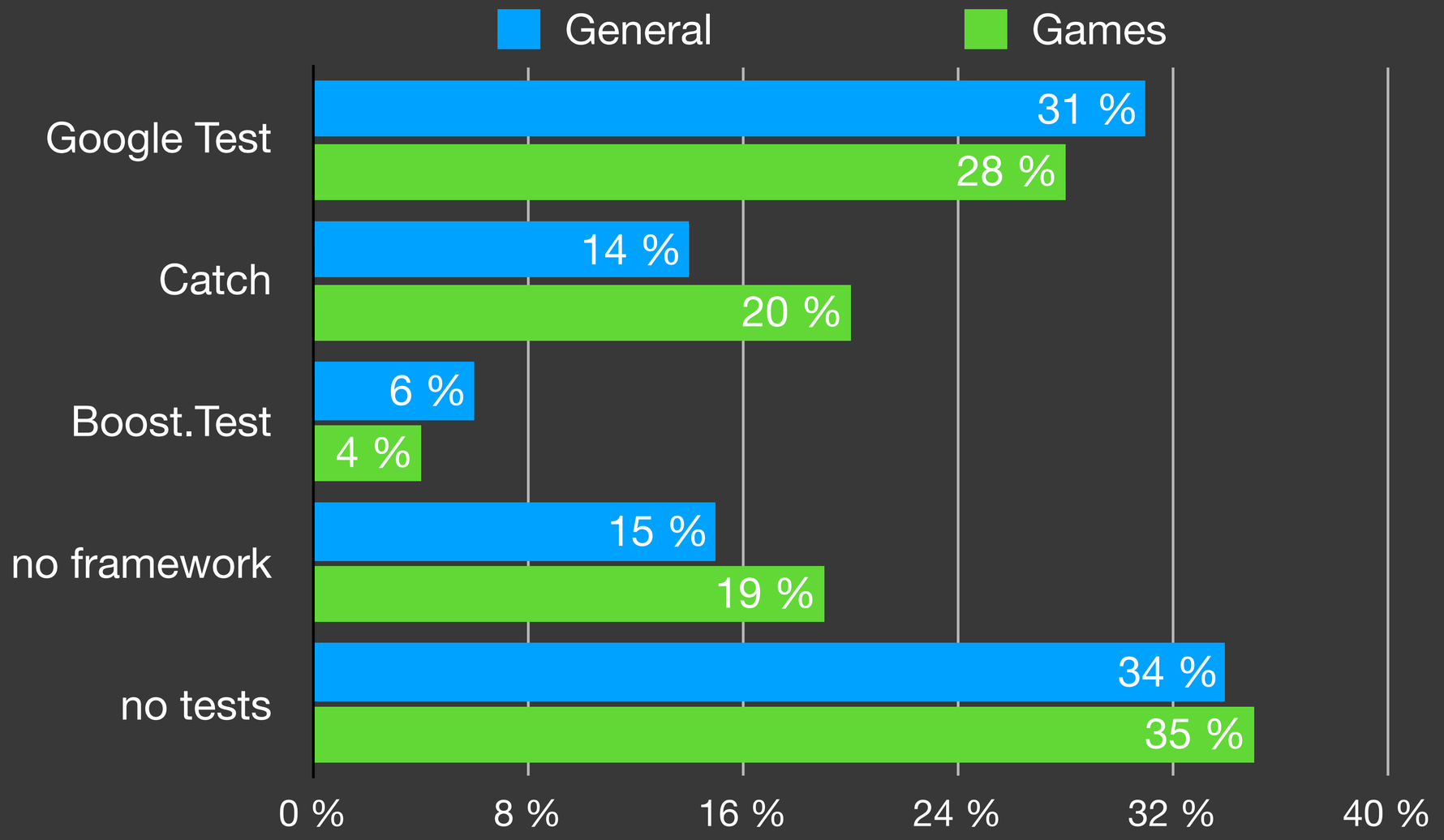
C++ in 2020: Unit Testing

- 101 “reasons” not to write unit tests
  - This is GUI code
  - This depends on disk / network / etc.
  - I already know the code is correct.
  - I need to ship this thing.

[CppCon 2016] David Sankel

“Building Software Capital: How to write the highest quality code and why"

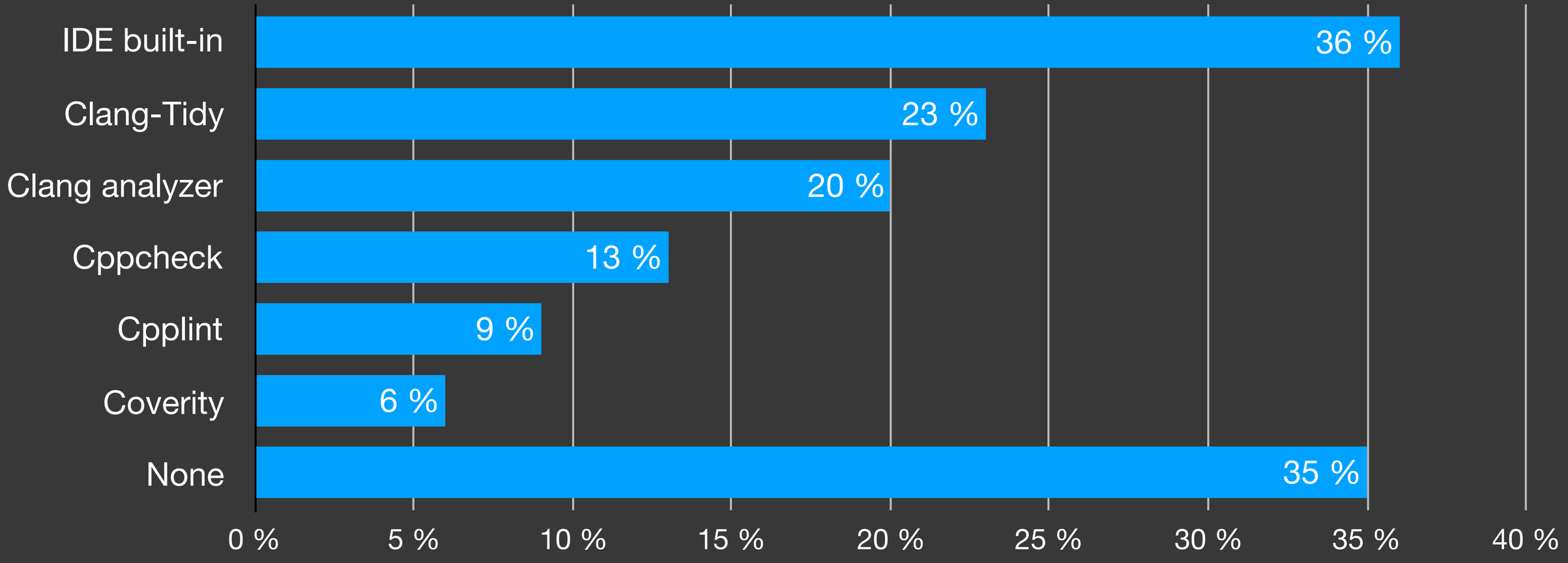
C++ in 2020: Unit Testing



C++ in 2020: Static Analysis

- Adoption static analysis into your workflow:
  - I don’t run it
  - I run it manually from time to time
  - I run it weekly (Friday evenings) on the CI server
  - I keep it always-on in my IDE

C++ in 2020: Static Analysis



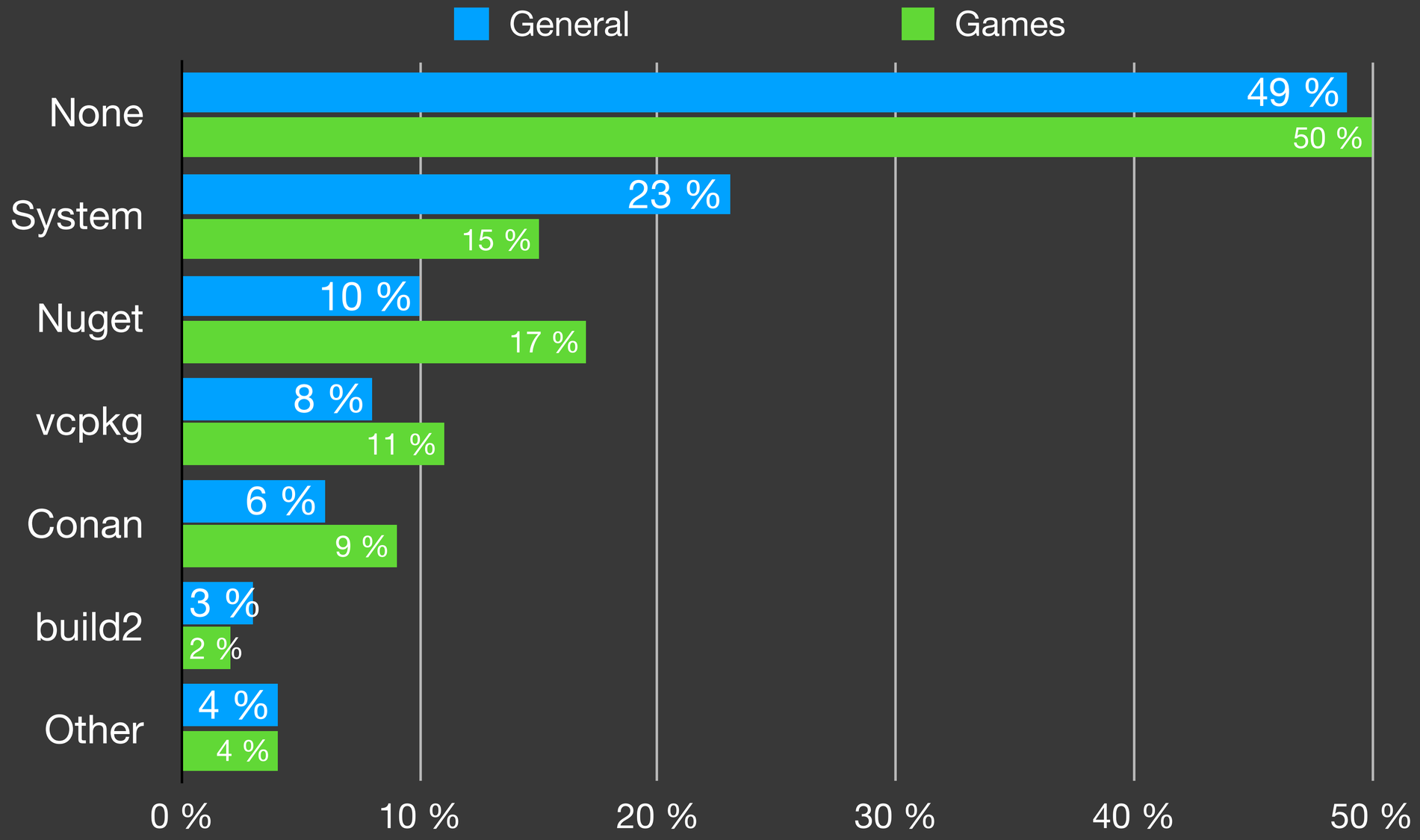
C++ in 2020

1. “Managing libraries my application depends on”
– major pain point for 47%

2. “Build times”
– major pain point for 42%

3. “Setting up a continuous integration pipeline from scratch (automated builds, tests, …) ”
– major pain point for 32%

C++ in 2020: dependency management



C++ in 2020: dependency management

1. “The library source code is part of my build” – 67%

2. “I compile the libraries separately using their instructions” – 58%

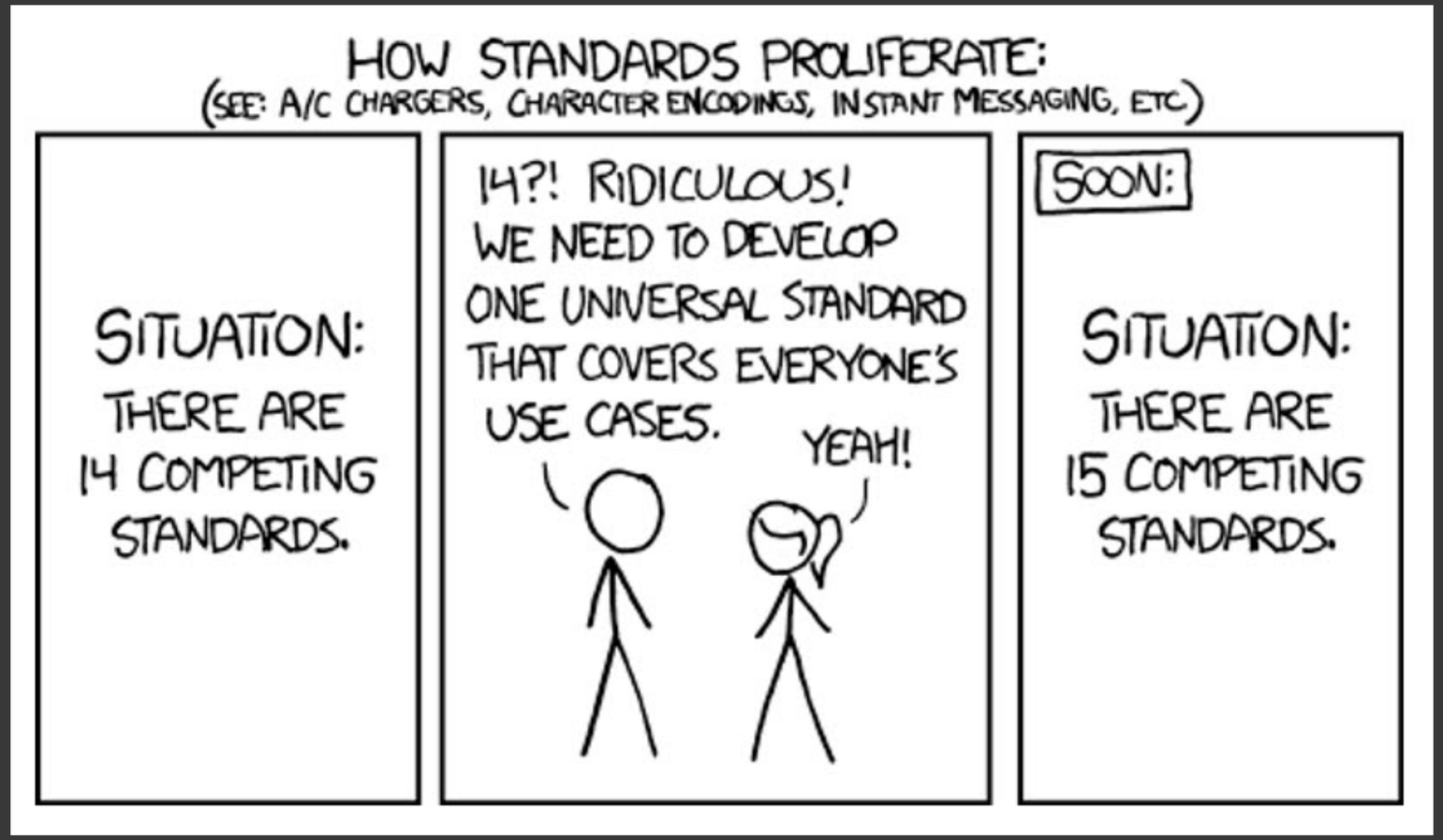
3. “System package managers (e.g., apt, brew, …)” – 39%

4. “I download prebuilt libraries from the Internet” – 33%

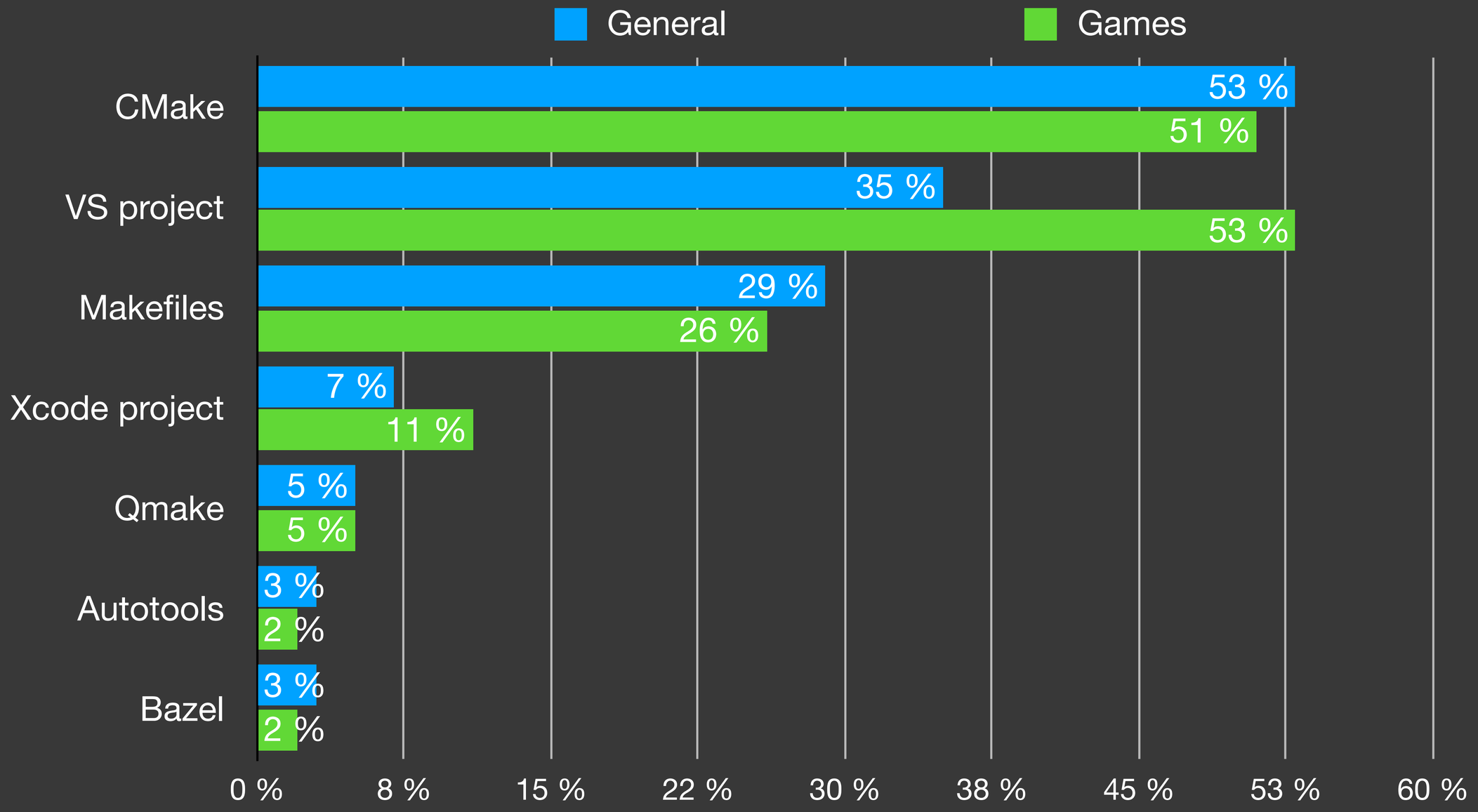
5. Conan, vcpkg – **14%** each

(C++ Foundation “Lite” 2020)

C++ in 2020: project models



C++ in 2020: project models



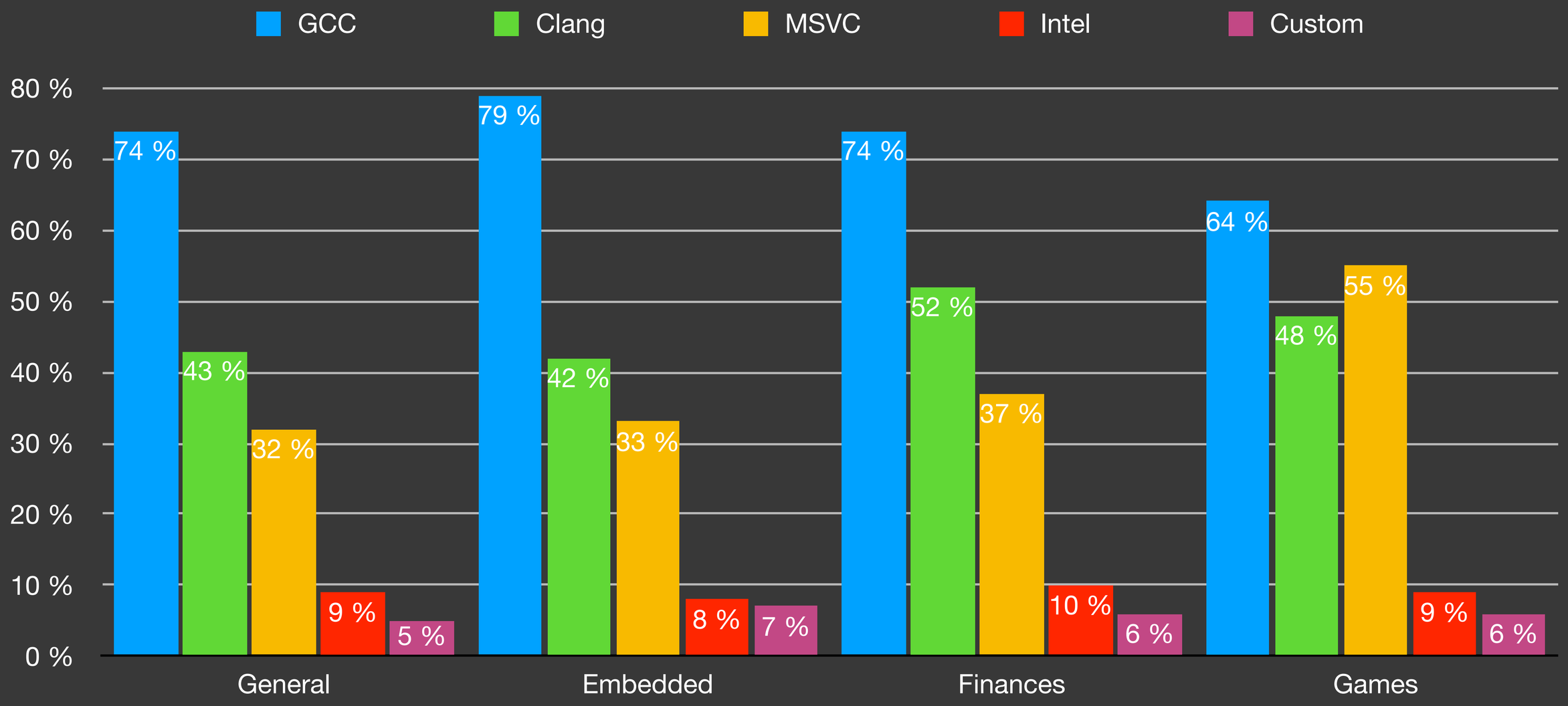
C++ in 2020: project models

“Managing CMake projects”
– major pain point for 27%

“Managing Makefiles ”
– major pain point for 22%

“Managing MSBuild projects”
– major pain point for 19%

C++ in Games


GCC
Clang
MSVC
Intel
Custom
80 %
70 %
60 %
50 %
40 %
30 %
20 %
10 %
0 %
74 %
43 %
32 %
9 %
5 %
79 %
42 %
33 %
8 %
7 %
74 %
52 %
37 %
10 %
6 %
64 %
48 %
55 %
9 %
6 %
General
Embedded
Finances
Games

Questions?

References

- C++ Foundation Developer “Lite” Survey
  - [2020-4] https://isocpp.org/files/papers/CppDevSurvey-2020-04-summary.pdf
- The State of Developer Ecosystem Survey
  - [2020] https://www.jetbrains.com/lp/devecosystem-2020/cpp/
- Herb Sutter “Thoughts on a more powerful and simpler C++ (5 of N)”
  - [CppCon 2018] https://www.youtube.com/watch?v=80BZxujhY38
- Nicolas Fleury "C++ in Huge AAA Games"
  - [CppCon 2014] https://www.youtube.com/watch?v=qYN6eduU06s
- Scott Wardle “Memory and C++ debugging at Electronic Arts”
  - [CppCon 2015] https://www.youtube.com/watch?v=8KIvWJUYbDA
- EASTL - Electronic Arts Standard Template Library
  - [2007] http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2007/n2271.html
  - [GitHub] https://github.com/electronicarts/EASTL