C++ Should Be C++

D3023R1

David Sankel, Adobe

Acknowledgments

Papers

- Direction for ISO C++ (P2000R4)
- Thriving in a Crowded and Changing World
- How can you be so certain? (P1962R0)
- Operating principles for evolving C++ (P0559R0)

Feedback

- Niall Douglas
- Inbal Levi
- Bjarne Stroustrup
- Herb Sutter

Reflector discussion

- Axel Naumann
- Balog Pal
- Corentin Jabot
- Harald Achitz
- Howard Hinnant
- Jarrad Waterloo
- JF Bastien
- Oliver Hunt
- Patrice Roy
- Peter Dimov
- René Ferdinand Rivera Morell
- Ville Voutilainen

It's been a tough couple years.

Questions raised

Questions raised

- What is the point of our work?

Questions raised

- What is the point of our work?
- Why is it worthwhile?

Questions raised

- What is the point of our work?
- Why is it worthwhile?
- What should we be doing?

Questions raised

- What is the point of our work?
- Why is it worthwhile?
- What should we be doing?
- What are our obstacles?

Goal: **start a conversation**

"if we are not careful, C++ can still fail"

"if we are not careful, C++ can still fail"

"[principles are needed] to keep C++ alive, healthy and progressing"

What can C++ loose?

What can C++ loose?

utility

What can C++ loose?

utility

Observation 1: other programming languages cannot "take away" C++'s utility

What can C++ loose?

utility

Observation 1: other programming languages cannot "take away" C++'s utility

Observation 2: legislation cannot reduce C++'s capabilities

So what can threaten C++'s utility?

So what can threaten C++'s utility?

us

So what can threaten C++'s utility?

us

backwards compatibility mitigates this

The surest way to sabotage a standard is to say yes to everything

The surest way to sabotage a standard is to say yes to everything

"[a complex mess of incoherent ideas becomes] insanity to the point of endangering the future of C++"

How do we mitigate this risk and align the committee to a greater good?

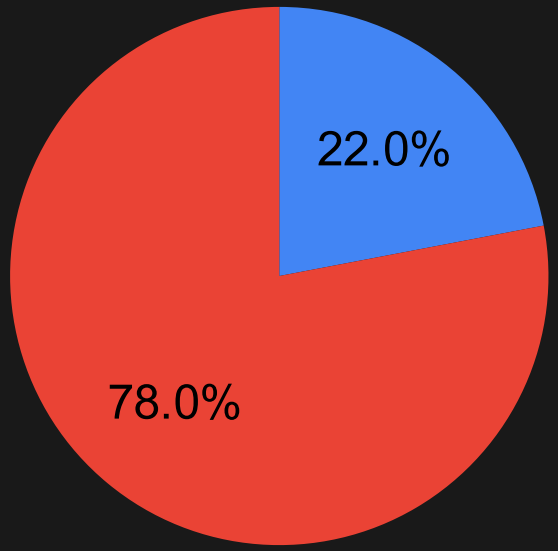
Mission

Some not-so-great missions

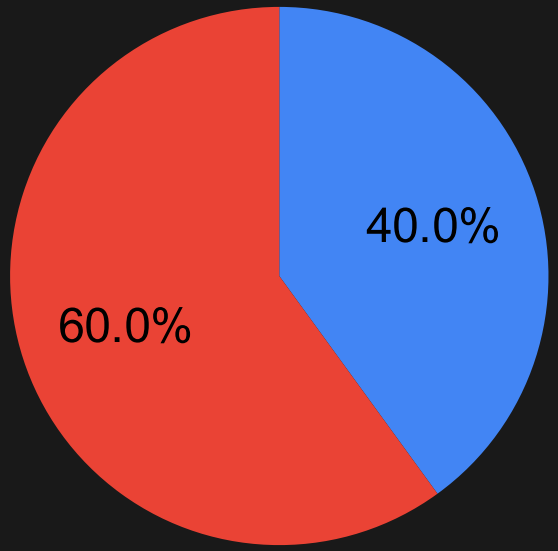
1. Make/keep C++ the best language in the world
2. Make C++ the only language people use
3. Make C++ the most popular language

Consequences of this line of thought

- **Misalignment.** "competing" tools good for users, but bad for mission
- **Ignorance.** Why investigate when we're the best?



22% of C++ users are also using Rust



40% of C++ users want to use Rust

Ignorance of Rust is ignorance of our users.

Ignorance of Rust is ignorance of our users.

We need a more helpful mission

A mission: *improve people's lives*

A mission: *improve people's lives*

```
for(const auto & i: vec) { // "Ah, that's nice!" x 5,000,000
   f(i);
}
```


Habitat
for Humanity®


Peace
Corps

Social obstacles to overcome

C++ as a personal and group identity

What language do you program in?

C++ as a personal and group identity

What language do you program in?

- Clouds reason

C++ as a personal and group identity

What language do you program in?

- Clouds reason
- Deep seated fears

Counterproductive rhetoric

fatal, fail, dead, and death

Counterproductive rhetoric

fatal, fail, dead, and death

- Living things have finite resources (users), competition (other languages), and death (obsolescence)

Counterproductive rhetoric

fatal, fail, dead, and death

- Living things have finite resources (users), competition (other languages), and death (obsolescence)
- C++ is merely a tool that is sometimes useful!

Counterproductive rhetoric

fatal, fail, dead, and death

- Living things have finite resources (users), competition (other languages), and death (obsolescence)
- C++ is merely a tool that is sometimes useful!
- Idea of "enemy" frustrates cooperation

"I can think of few things more depressing than people still using C++ in a million years" - Lisa Lippincott

"I can think of few things more depressing than people still using C++ in a million years" - Lisa Lippincott

- -5,000,000: first homininans

"I can think of few things more depressing than people still using C++ in a million years" - Lisa Lippincott

- -5,000,000: first homininans
- -300,000: first homo sapians

"I can think of few things more depressing than people still using C++ in a million years" - Lisa Lippincott

- -5,000,000: first homininans
- -300,000: first homo sapians
- -40,000: last non-sapian hominan died out

"I can think of few things more depressing than people still using C++ in a million years" - Lisa Lippincott

- -5,000,000: first homininans
- -300,000: first homo sapians
- -40,000: last non-sapian hominan died out
- -12,000: agriculture

"I can think of few things more depressing than people still using C++ in a million years" - Lisa Lippincott

- -5,000,000: first homininans
- -300,000: first homo sapians
- -40,000: last non-sapian hominan died out
- -12,000: agriculture
- -5,000: writing

"I can think of few things more depressing than people still using C++ in a million years" - Lisa Lippincott

- -5,000,000: first homininans
- -300,000: first homo sapians
- -40,000: last non-sapian hominan died out
- -12,000: agriculture
- -5,000: writing
- -500: modern English

"I can think of few things more depressing than people still using C++ in a million years" - Lisa Lippincott

- -5,000,000: first homininans
- -300,000: first homo sapians
- -40,000: last non-sapian hominan died out
- -12,000: agriculture
- -5,000: writing
- -500: modern English
- -78: first computer

"I can think of few things more depressing than people still using C++ in a million years" - Lisa Lippincott

- -5,000,000: first homininans
- -300,000: first homo sapians
- -40,000: last non-sapian hominan died out
- -12,000: agriculture
- -5,000: writing
- -500: modern English
- -78: first computer
- -38: first C++ release

Personal opportunity vs. stewardship

Gain C++ expertise, mingle with celebrities, land a proposal!

Some numbers

- 5,000,000 C++ users
- About the population of Ireland
- >1 C++ programmer for every 2,000 people

Some numbers

- 5,000,000 C++ users
- About the population of Ireland
- >1 C++ programmer for every 2,000 people

"we are writing a standard for millions of programmers to rely on for decades, a bit of humility is in order"

No one is qualified for this

Stewardship responsibilities

- Reject proposals without understandable value proposition
- Resist social pressure when you're against something
- Build an informed opinion (read the paper, test the feature, collaborate)
- Say "yes" only when risk is minimal

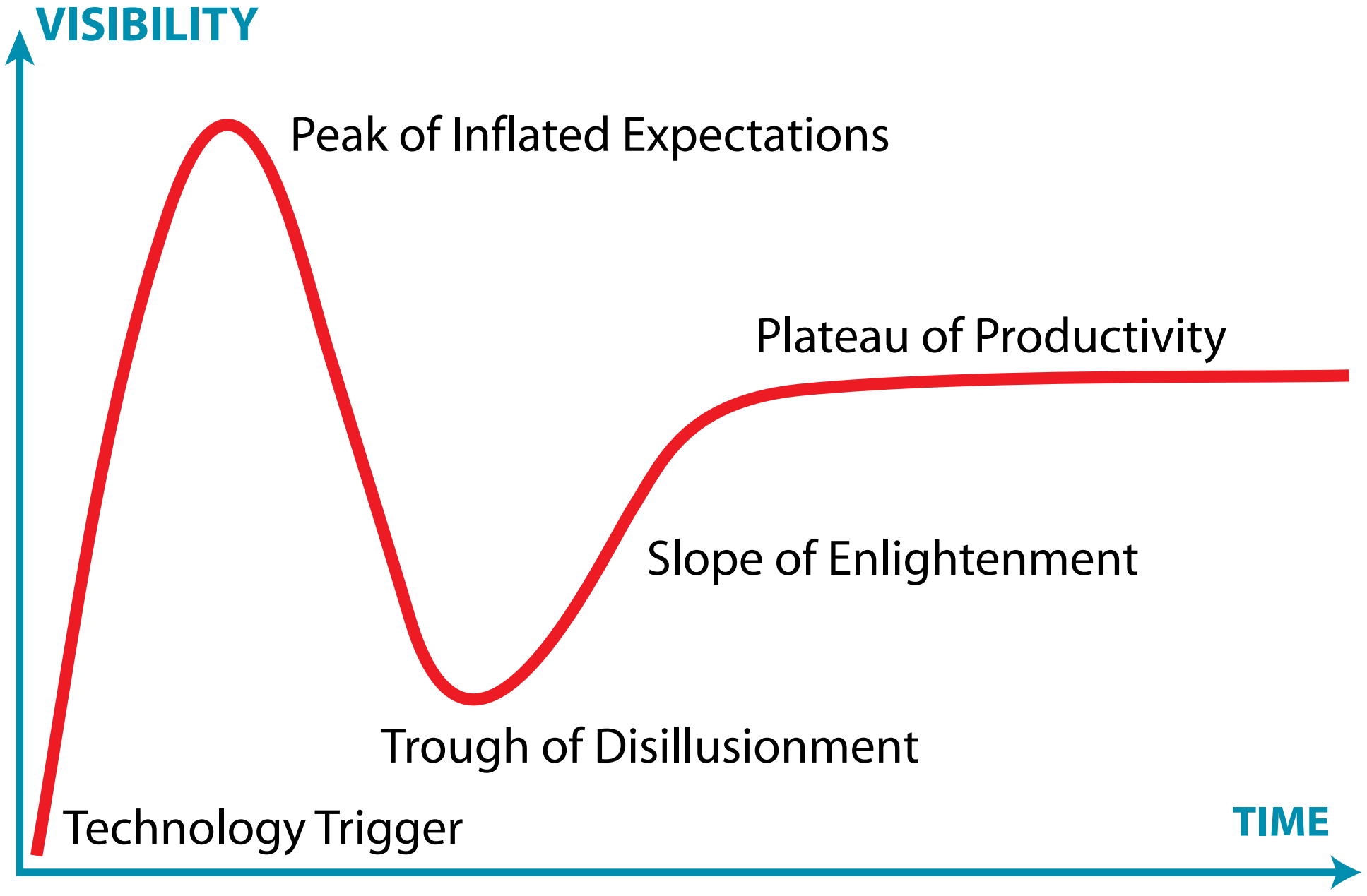
You are a caretaker and guardian of something beyond yourself!

If you're writing a proposal, consider...

- "[C++ is] trying to do too much too fast"
- "[C++ needs] to become more restrained and selective"
- Experience reports on high-impact proposals are more helpful than low-impact proposal papers
- Help is available

Technical obstacles to overcome

Neophilia


VISIBILITY
Peak of Inflated Expectations
Plateau of Productivity
Slope of Enlightenment
Trough of Disillusionment
Technology Trigger
TIME

Neophilia risks

- Failed expectations
- Poor integration
- Non-expert frustration with learning costs

Neophilia risks

- Failed expectations
- Poor integration
- Non-expert frustration with learning costs

"Keeping up with the Jonses is a disservice"

Expert bias

- **Underrepresentation** of average programmers in committee
- **Opportunity cost** of expert features is improving lives at scale
- **Expert friendly** means smaller impact

Know your numbers

Query	# files	%
*	34.6M	100%
cout	9M	26%
int main(	8.3M	24%
std::vector	4.7M	14%
std::unordered_map	692k	2.0%
std::sort	379k	1.1%

Query	# files	%
*	34.6M	100%
#include <thread>	281k	0.81%
std::integral_constant	5.6k	0.016%
std::pmr	1.7k	0.0049%
hazardptr OR hazard_ptr	848	0.0025%
std::launder	278	0.00080%
std::atomic<std::shared_ptr	42	0.000121%

Get average engineer feedback

- Is this something you would use? How often?
- Is this ergonomic?
- How hard is this to learn?
- Is this worth another chapter in the C++ book?

Complexity

Complexity: hiring impact

- Barrier to entry is too high
- Fewer people want to learn C++ and fewer schools want to teach it

“C++ [is] in danger of losing coherency due to proposals based on differing and sometimes mutually contradictory design philosophies and differing stylistic tastes.”

- `std::function` -> `std::copyable_function`, `std::function_ref`, and `std::move_only_function`
- Preference for new tagged/named parameters over simple classes
- Recent proposals contradicting Stephanov's regular concept
- ...

Aim for coherence

- "Is this the common C++ style?" "Is this increasing C++'s barrier to entry?"
- Have study groups get early feedback from evolution groups on feature desirability
- Overcome reluctance to say "I don't think this belongs in C++"

Prioritize well

- Don't deny the greater number of users time spent on proposals that can improve their lives.
- Avoid pet peeves
- Say "no" more often and, if needed, repeatedly

Opinions on moving ahead

Memory safety

- We're near the peak of inflated expectations
- 1.6% of CMake projects mention `fsanitize`
- There are 31 Rust projects for each of these

Memory safety - dangers

- Opportunity cost
- Assuming complexity and incoherence trying to "keep up"
- Missing out on memory-safe language interop

Major C++ overhaul

- So-called successor languages?
- C++ 2.0

C++ 2.0

- New syntax isn't a priority for typical C++ developers
- Users desire coherence in their C++ code bases
- We aren't especially suited to make a C++ successor

Our biggest opportunity to improve people's lives:

Focus on C++ as it is today

Some examples

Command-line processing library

- 3.2% of GitHub C++ source files use a command-line parser library
- 2.0% of GitHub C++ source files *manually* parse the command line
- Manual command-line parsing is as popular as `std::unordered_map`
- This could easily impact a million lives

Simple JSON library

- 2.9% of GitHub C++ source files mention JSON (0.2% for YAML and 3.5% for XML)
- Consider the number of custom formats replaced with JSON

Hash containers

- New techniques produce 2-6x speedups
- Require API changes
- Considerable energy/environmental impact

There are many others...

C++ Should Be C++

- Mission: improve people's lives
- Stewardship
- Technical obstacles: neophilia, expert bias, complexity/coherence
- Focus on broad improvements like range-based for loops

Let's discuss!