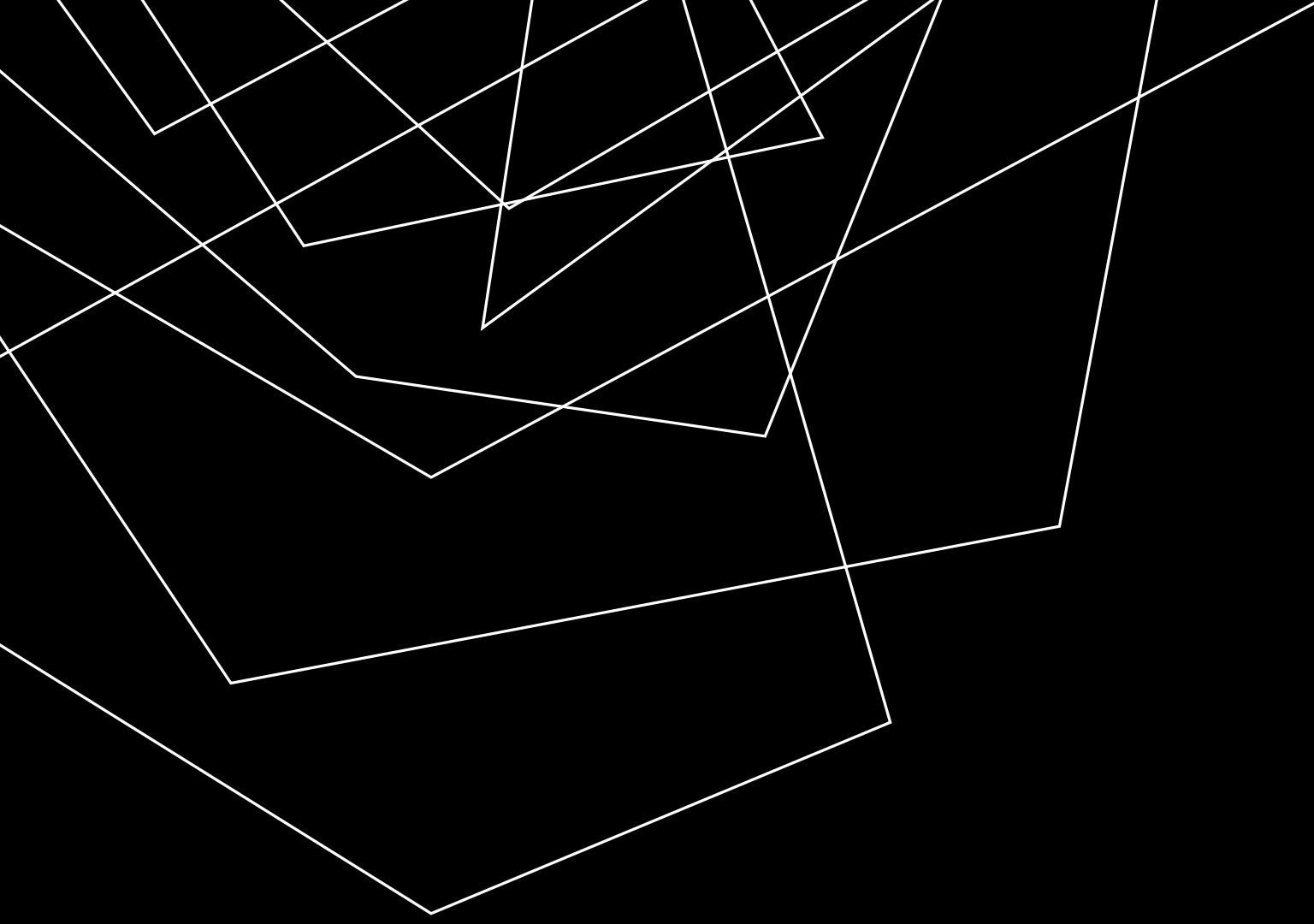




AUTOCHECK
DJORDJE TODOROVIC
VLADIMIR VUKSANOVIC
PETAR JOVANOVIC
HTEC GROUP

ABOUT AUTOCHECK

- Static analysis tool for checking compliance with AUTOSAR/MISRA C++ Guidelines
- Command line tool
- Generate rule violation reports for a project provided
- IDE Integration (VSCode)
- AI Integration (llama.cpp)
- Presented at **2020 LLVM Developers' Meeting**
 - Extending Clang for Checking Compliance With Automotive Coding Standards - Milena Vujosevic Janicic - <https://www.youtube.com/watch?v=-6dL-7xkIV0>

AUTOSAR / MISRA C++ GUIDELINES

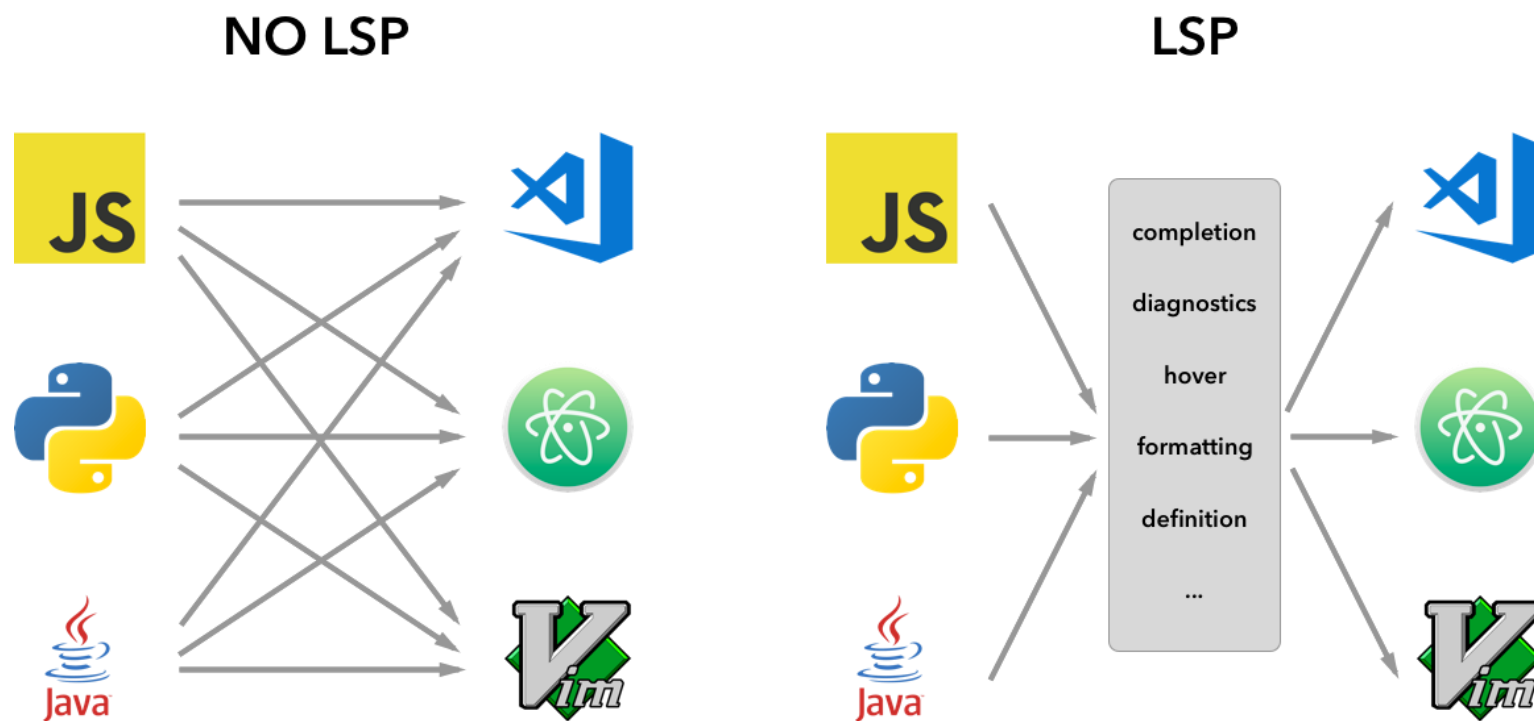
- Coding guidelines for using C++ in safety-related and critical systems
- Started as MISRA (Motor Industry Software Reliability Association) C++ in 2008
- Updated for newer C++ standards AUTOSAR (AUTOMOTIVE OPEN SYSTEM ARCHITECTURE) C++14
- Updated again as MISRA C++: 2023 for C++ 23

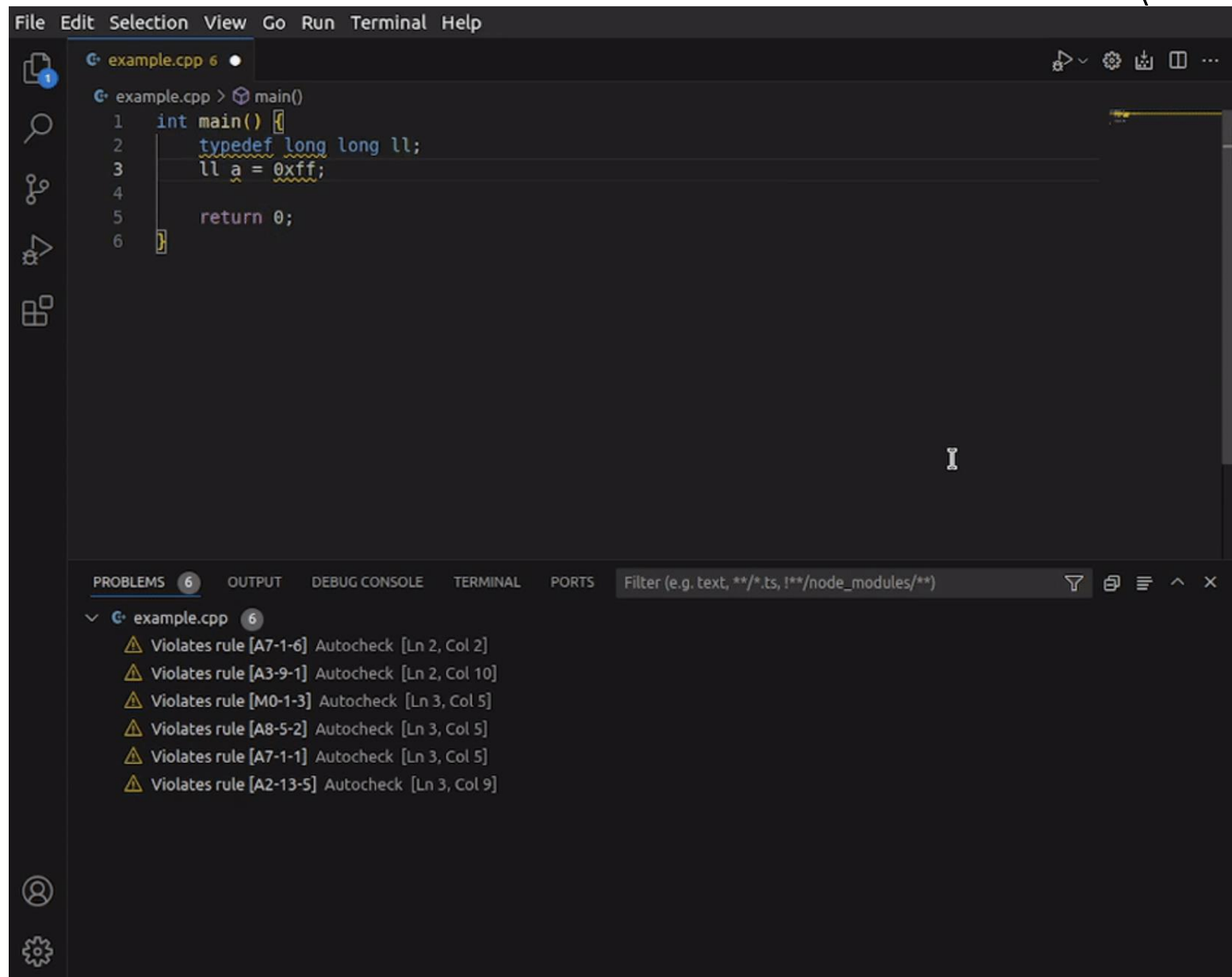
IMPLEMENTATION

- Based on the LLVM Compiler Infrastructure, like clang-tidy
 - libClang
- Works on multiple compilation stages:
 - Preprocessor
 - Lexer
 - Abstract syntax tree (AST)
 - AST Visitors
 - AST Matchers
 - Static Analyzer
 - LLVM Intermediate Representation (IR)

IDE INTEGRATION

- Language Server Protocol
- Supported by Visual Studio, VSCode, IntelliJ, vim, emacs, Qt Creator...





AI INTEGRATION

- Used for automatic code rewriting
- Local LLM instance using llama.cpp
- Llama.cpp enables inference of different LLMs (not just llama) locally either on a CPU, or GPU (CUDA, HIP)
- Used for rules where compiler technology cannot be applied – e.g. adding comments to functions that are missing it

CURRENT STATUS

- Private repo
- Willing to open source it
 - <https://www.phoronix.com/news/Autocheck-LLVM-Safety-Critical>
 - <https://discourse.llvm.org/t/rfc-autocheck-a-source-code-analysis-tool-based-on-clang-llvm-for-automotive/76333>
- Problem with Licensees
 - MISRA/AUTOSAR
 - You have an advice?
 - Drop us a message at
 - djordje.todorovic@htecgroup.com
 - petar.jovanovic@htecgroup.com

```
#include <iostream>

int main() {
    int a = 0xff;
    std::cout << a << std::endl;

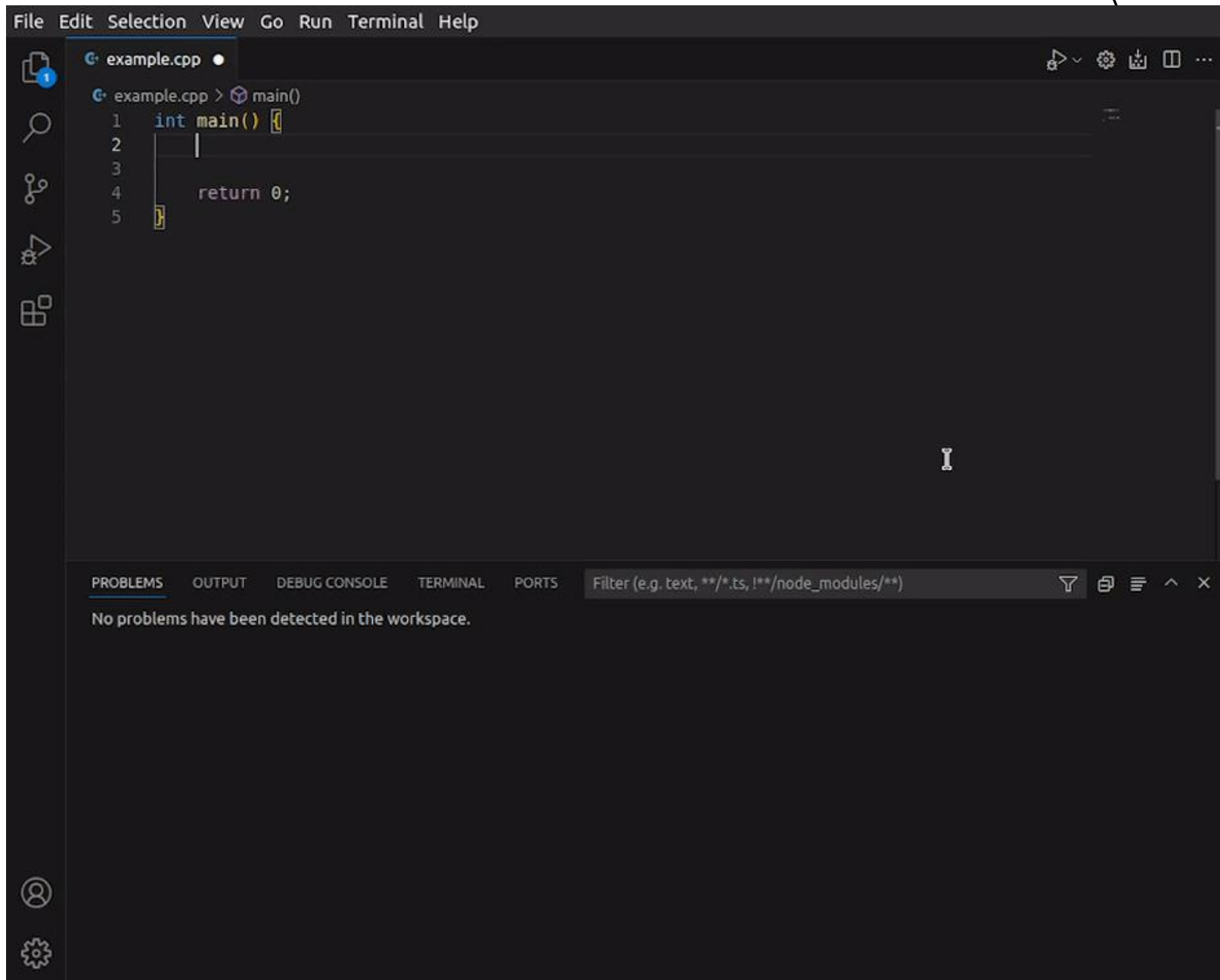
    return 0;
}
```

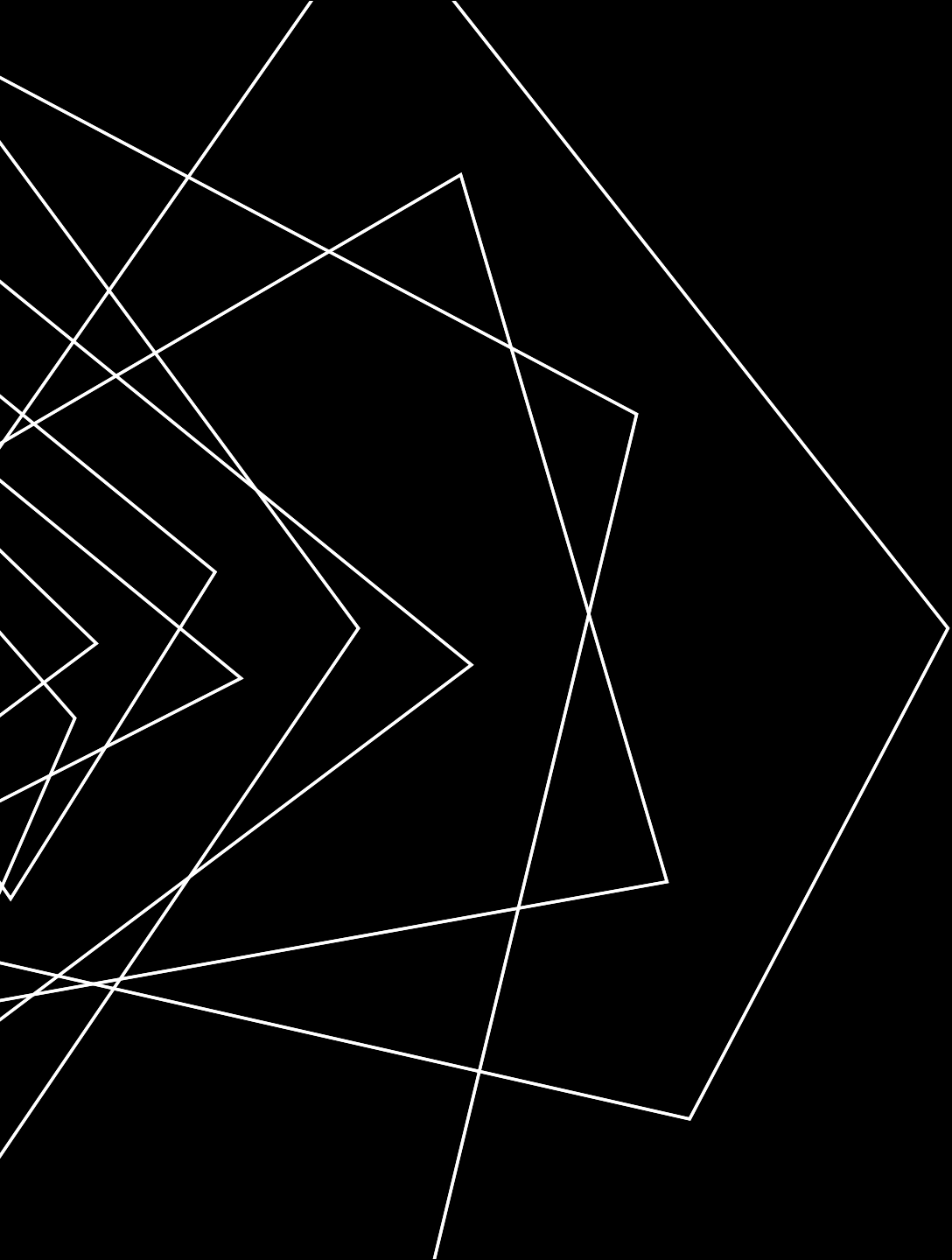
Run

✓

16s

example.cpp 1,1 All





THANK YOU