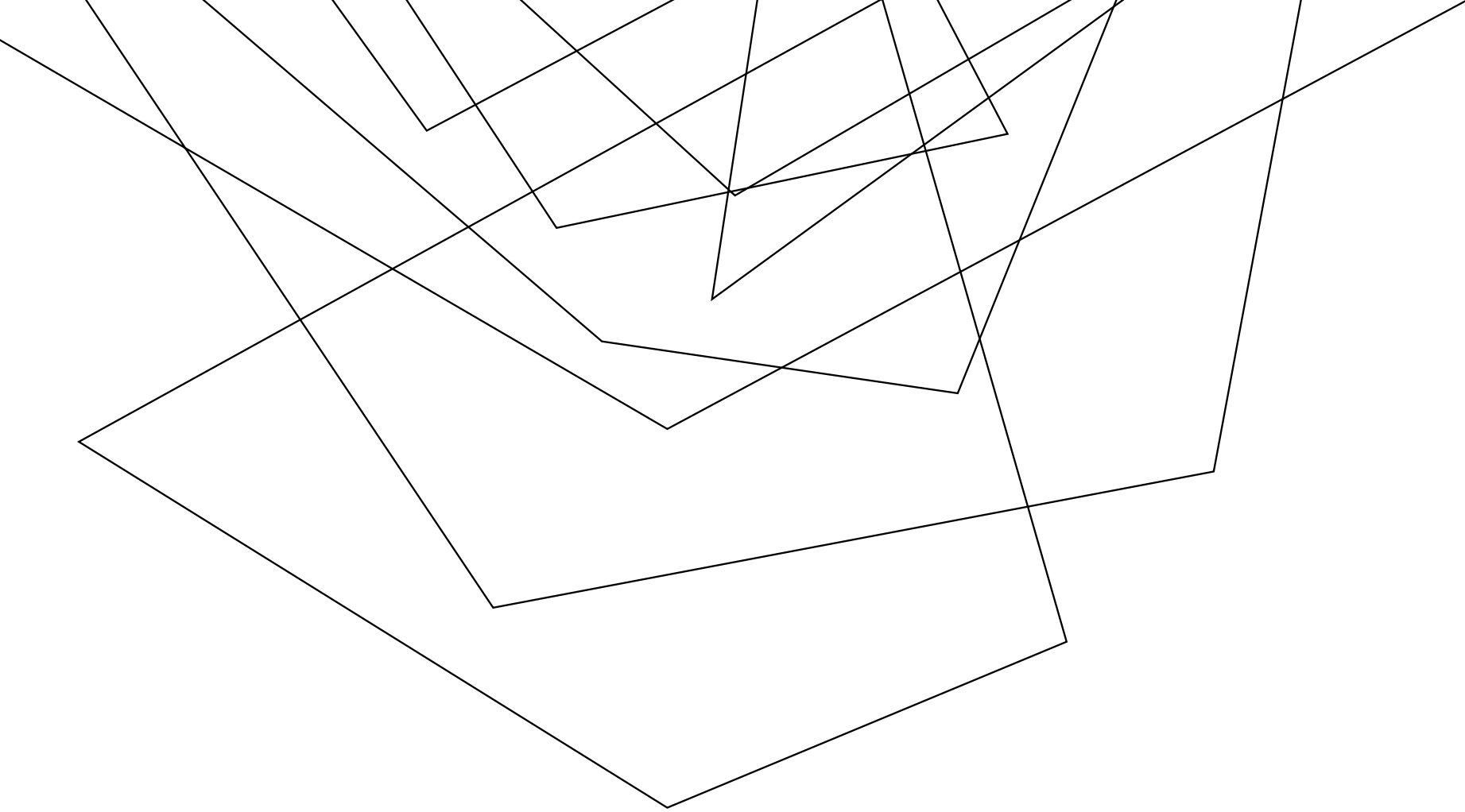


EXERCISE 33

QUIZ 3



**ADMINISTRIVIA
AND
ANNOUNCEMENTS**



LINTING

EECS 677: Software Security Evaluation

Drew Davidson

LAST TIME: SSDLC

REVIEW: LAST LECTURE

CORRESPONDING SECURITY TASKS FOR THE SOFTWARE DEVELOPMENT LIFECYCLE

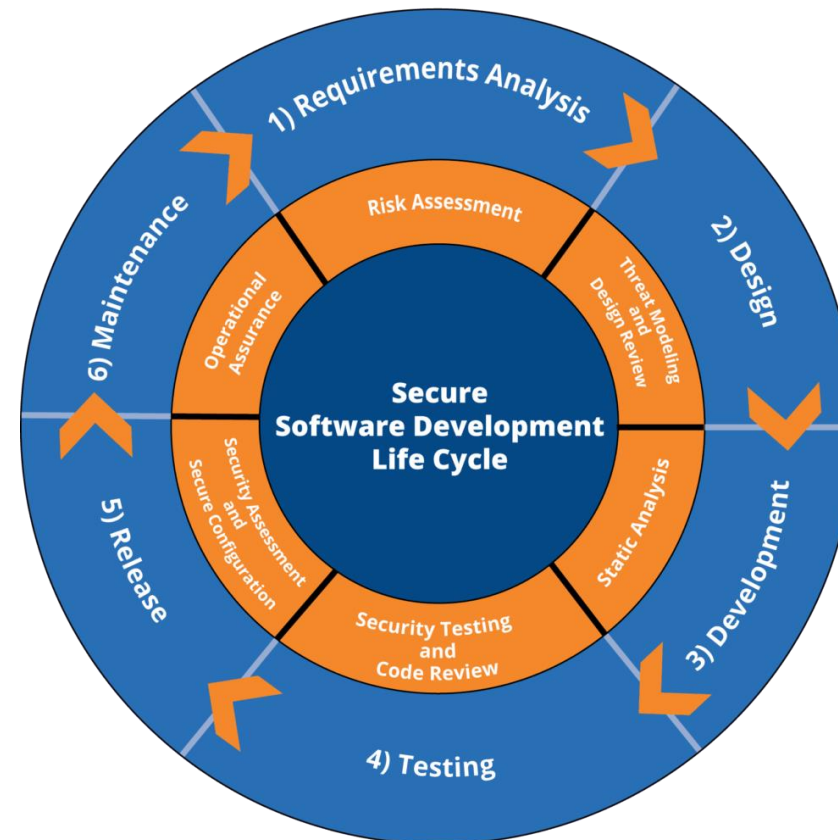
Requirement Analysis – Risk Assessment and Threat models

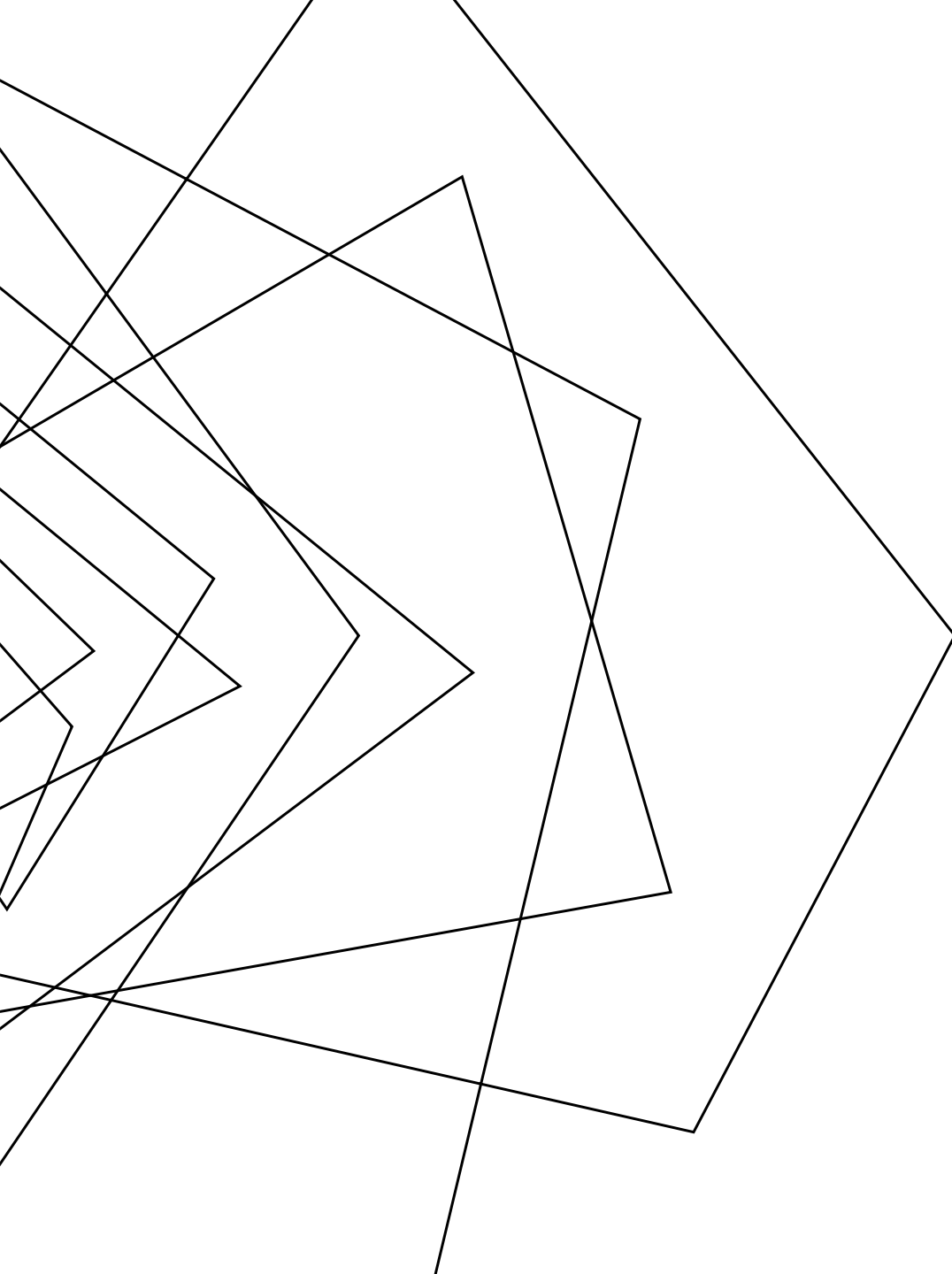
Design – Security Design Review

Development – Automated Code Analysis

Testing – Security Testing and Code Review

Maintenance and Evolution – Security Assessment and Configuration





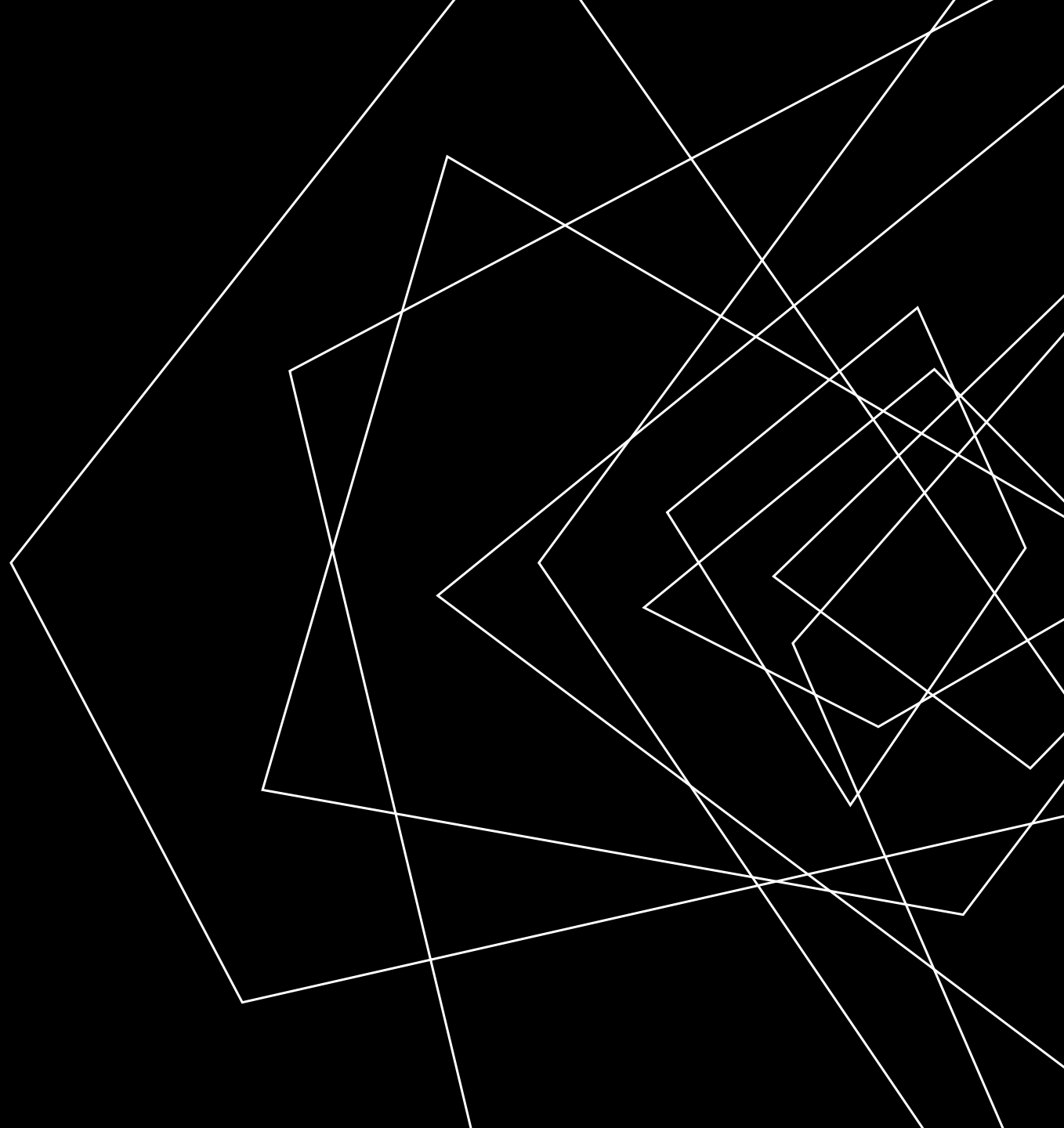
CLASS PROGRESS

HANDLING THE “SOFTER SIDE” OF SECURITY EVALUATION

We've described some of the high-level best-practices, let's talk about tool support

LECTURE OUTLINE

- Background / Context
- Linting
- Anti-Patterns
- Splint



SAD FACT: IT'S EASY TO WRITE INSECURE CODE

LINTING: BACKGROUND/CONTEXT

MANY PROGRAMMING LANGUAGES
HAVE EXPLOITABLE CONSTRUCTS

Programming constructs that do not operate as
intended under unforeseen circumstances



Artistic depiction of C programming

RECALL: SECURITY V USABILITY

LINTING OVERVIEW

MAINSTREAM PL PHILOSOPHY
PRIORITIZES SPEED AND SIMPLICITY

C *could* do more checking, but it doesn't

- Bounds checking
- Type safety



RECALL: SECURITY V USABILITY

LINTING OVERVIEW

EXPECTATIONS OF EFFICIENCY AND PERFORMANCE ARE HARD TO QUIT!

Disallowing unsafe behavior means going back on what's already been accomplished

- Rewrite legacy code
- Give up on some performance



CASE STUDY: MELTDOWN AND SPECTRE

LINTING OVERVIEW

THE PROBLEM: BRANCH PREDICTORS AND SPECULATIVE EXECUTION

Impact: leaking secrets

THE SOLUTION: MEDIATE SPECULATIVE EXECUTION

Early Fix performance:

OS Bench:

Intel Xeon 84~87%

AMD EPYC 91~94%.



RECALL: SECURITY V USABILITY

LINTING OVERVIEW

WAITING FOR BETTER TOOLS

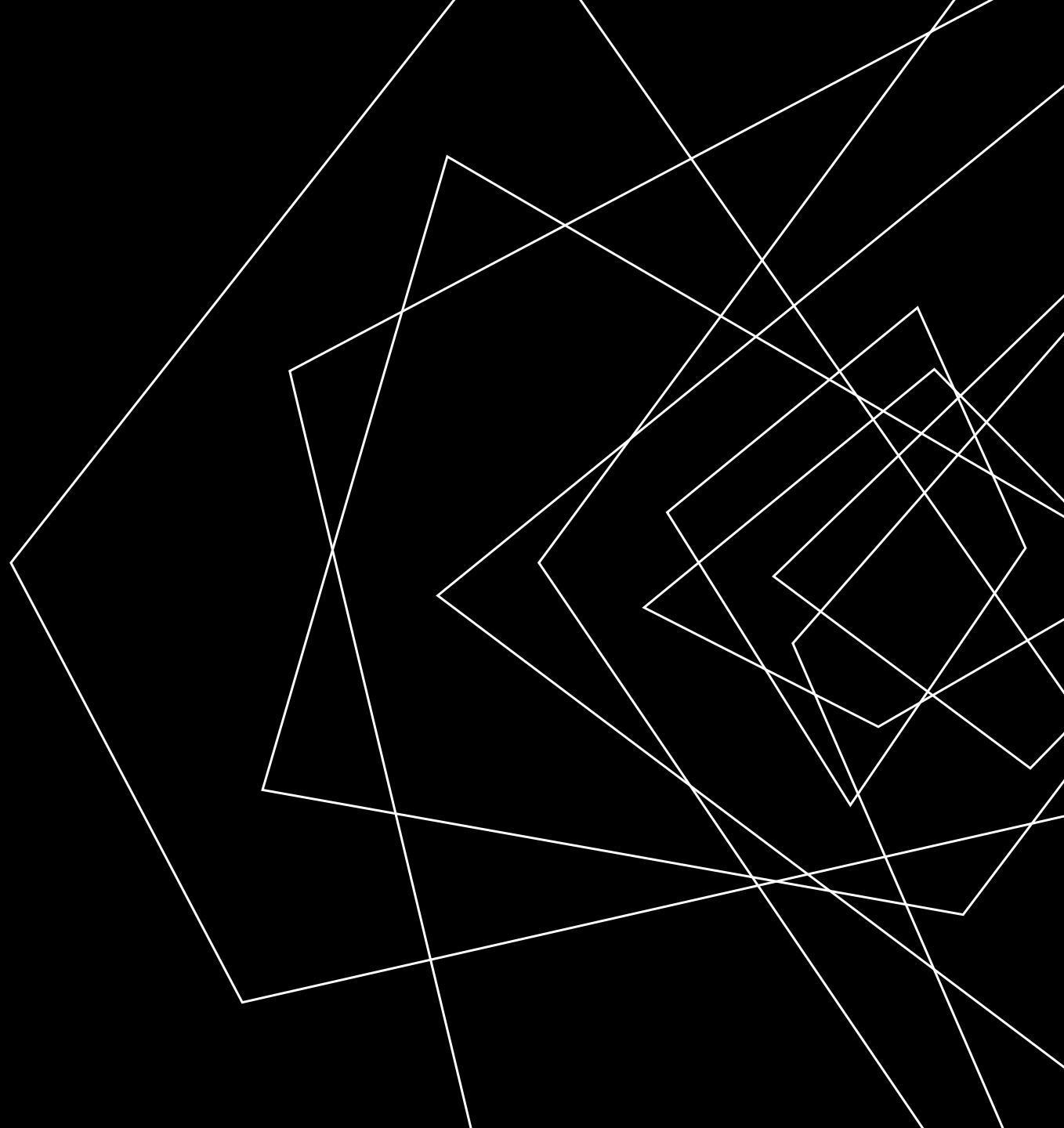
Some feel that the whole of imperative programming is inherently unsafe





LECTURE OUTLINE

- Background / Context
- Linting
- Anti-Patterns
- Splint



HEURISTIC TOOLS FOR AN IMPERFECT WORLD

LINTING: OVERVIEW

TRY NOT TO SHOOT YOURSELF IN THE FOOT

Highlight the stuff you probably shouldn't be doing in the first place



CATCH “ANTI-PATTERNS”

HUMAN FACTORS OF SECURITY

COMMON LANGUAGE-LEGAL PAIN-POINTS

Code that is highly situational, or simply shouldn't be legal in hindsight



HISTORY: JOHNSON, 1978

HUMAN FACTORS OF SECURITY

CREATED A PROGRAM CALLED “LINT”

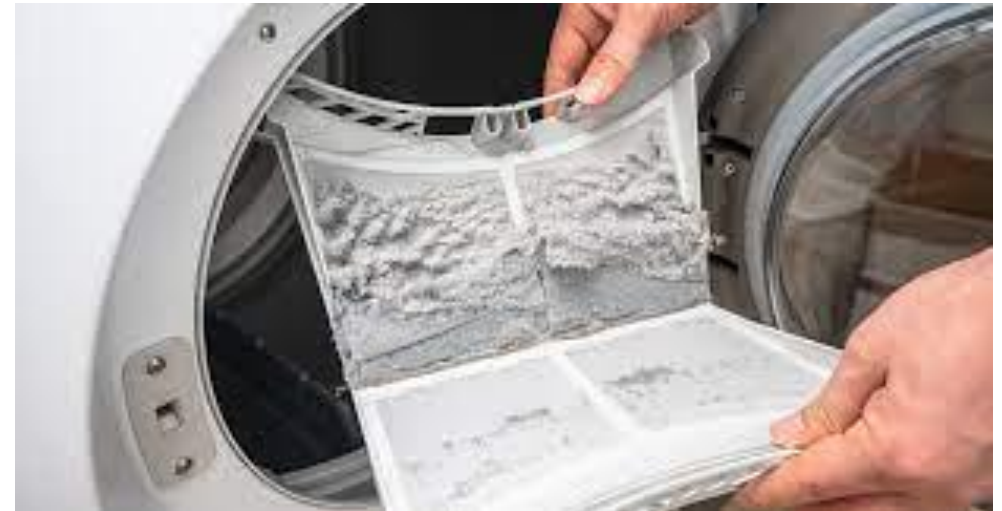
Aided in the development of YACC

Originally internal to Bell Labs, eventually open-sourced

NAME INSPIRED BY DRYER LINT TRAPS

Capture the “loose fibers” that come off the program

Leave the whole of the program intact



PRODUCTION LINTERS

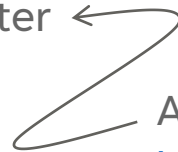
LINTING

MORE MODERN TOOLS

cppcheck – open-source linter

cpplint – Google’s in-house (open-source) linter

flake8 – python linter



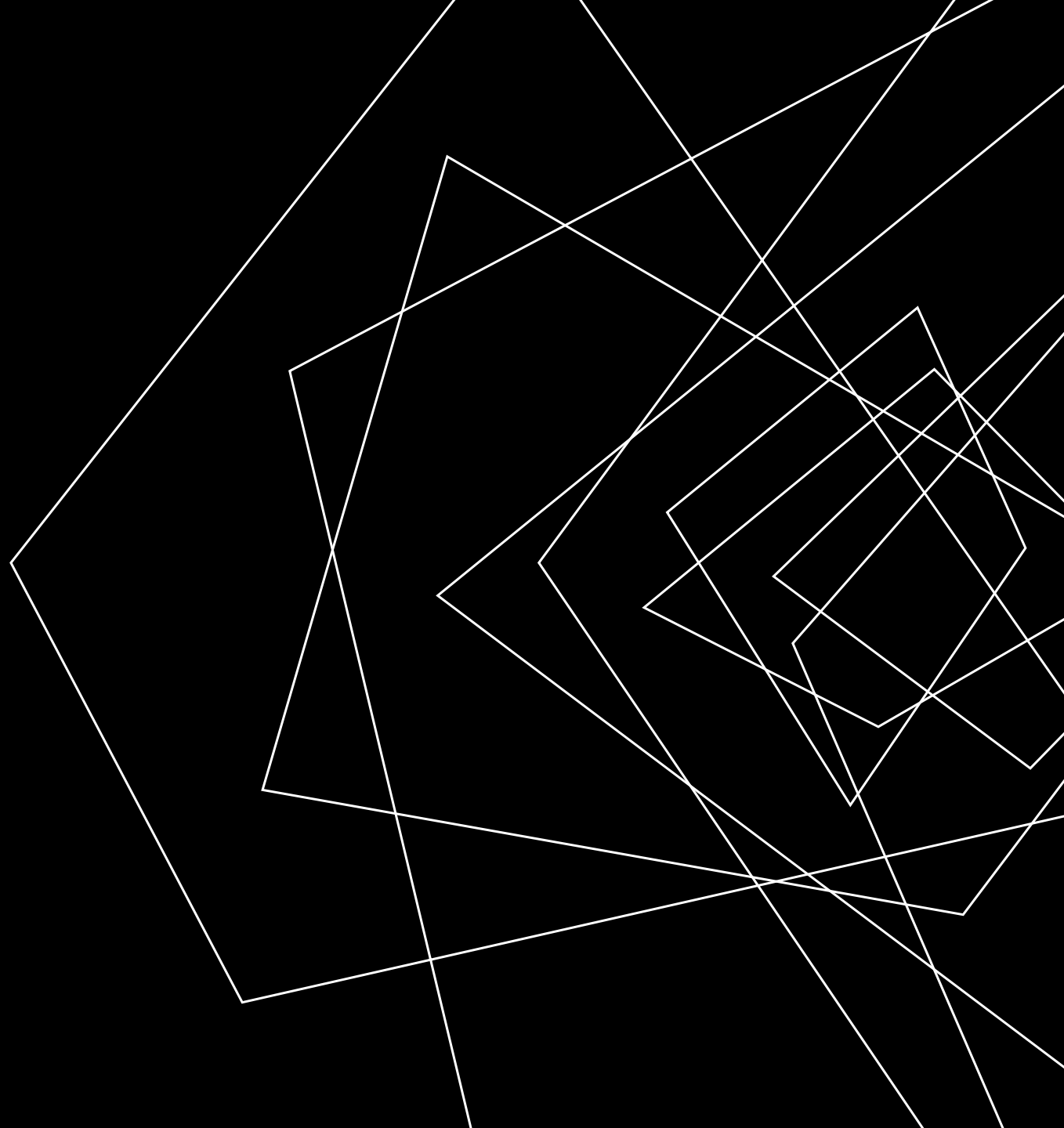
Also ensures adherence to style guide:

<https://google.github.io/styleguide/cppguide.html>

Good reminder that coding is still a human process

LECTURE OUTLINE

- Background / Context
- Linting
- Anti-Patterns
- Splint



ASSIGNMENT IN PREDICATE LINTING

```
int main(int argc, char * argv){  
    if (argc = 1){  
        //Do something  
    } else {  
        //Do something else  
    }  
}
```

MACRO POLLUTION

LINTING

```
#include <algorithm>
#include <Windows.h>

int main()
{
    int k = min(3, 4);
    return 0;
}
```

MACRO POLLUTION

LINTING

```
#include <algorithm>
#include <Windows.h>

int main()
{
    int k = min(3, 4);
    return 0;
}
```

```
#define NOMINMAX
```

~~std::min(3, 4)~~

error C2589: '(' : illegal token on right side of '::'
error C2059: syntax error : '::'

SEPARATING INITIALIZATION FROM USE

LINTING

```
//Clean
int main(){
    int i;
    i = f();
    int jobs = NumJobs();
    //... more code ...
    f(jobs);
}
```

LINE CONTINUATION WEIRDNESS

LINTING

```
void foo(Object * x)
{
    // this comment is continued in the next line \
    if (isUnsafe(x))
        exit(0);
    x->deploy();
}
```

SCOPED INITIALIZATION

LINTING

```
//Good, assignment scoped to construct  
while (const char* p = strchr(str, '/')) {  
    str = p + 1;  
}
```

```
//Inefficient exception  
for (int i = 0; i < 1000000; ++i) {  
    Foo f;  
    f.DoSomething(i);  
}
```

```
{  
    Foo f;  
    for (int i = 0; i < 1000000; ++i) {  
        f.DoSomething(i);  
    }  
}
```


NAMESPACING (GOOD)

LINTING

```
namespace {  
    int foo(){  
        return 2;  
    }  
}  
  
static int bar(){  
    return 3;  
}  
  
int main(){  
    foo() + bar();  
    return 0;  
}
```

HEURISTIC TOOLS FOR AN IMPERFECT WORLD

LINTING: OVERVIEW

TRY NOT TO SHOOT YOURSELF IN THE FOOT

Highlight the stuff you probably shouldn't be doing in the first place



RECALL: SECURITY V USABILITY

LINTING OVERVIEW

MANY PROGRAMMING LANGUAGES
HAVE EXPLOITABLE CONSTRUCTS

Capture the “loose fibers” that come off the
program

Leave the whole of the program intact

