WTF IS BUILD.ZIG

Ed Yu

AGENDA

Introduction

Why (Not) Zig

What is Zig

Zig as a Compiler

What is Build.Zig

Zig Package Manager

Future of Zig

“I WILL BE SUBJECTIVELY OBJECTIVE AND OBJECTIVELY OPINIONATED”

- ED

PRE-INTRODUCTION

Disclaimers

- I don't assume you know Zig, so we start from beginning.
- But this is not an introductory talk about Zig.
- This is a very candid "what it is" not "what it should be" or "how it will be."

Who Am I?

- I'm not a C++ expert nor a Makefile expert.
- I like programming languages, and my favorites so far are Haskell, Clojure, and Zig.
- I'm using Zig to build a decentralized youtube + reddit.

WHY WTF

- Zig Package Manager -- WTF is Zon
- Zig Union(Enum) -- WTF is switch(union(enum))
- Zig If -- WTF is !?bool

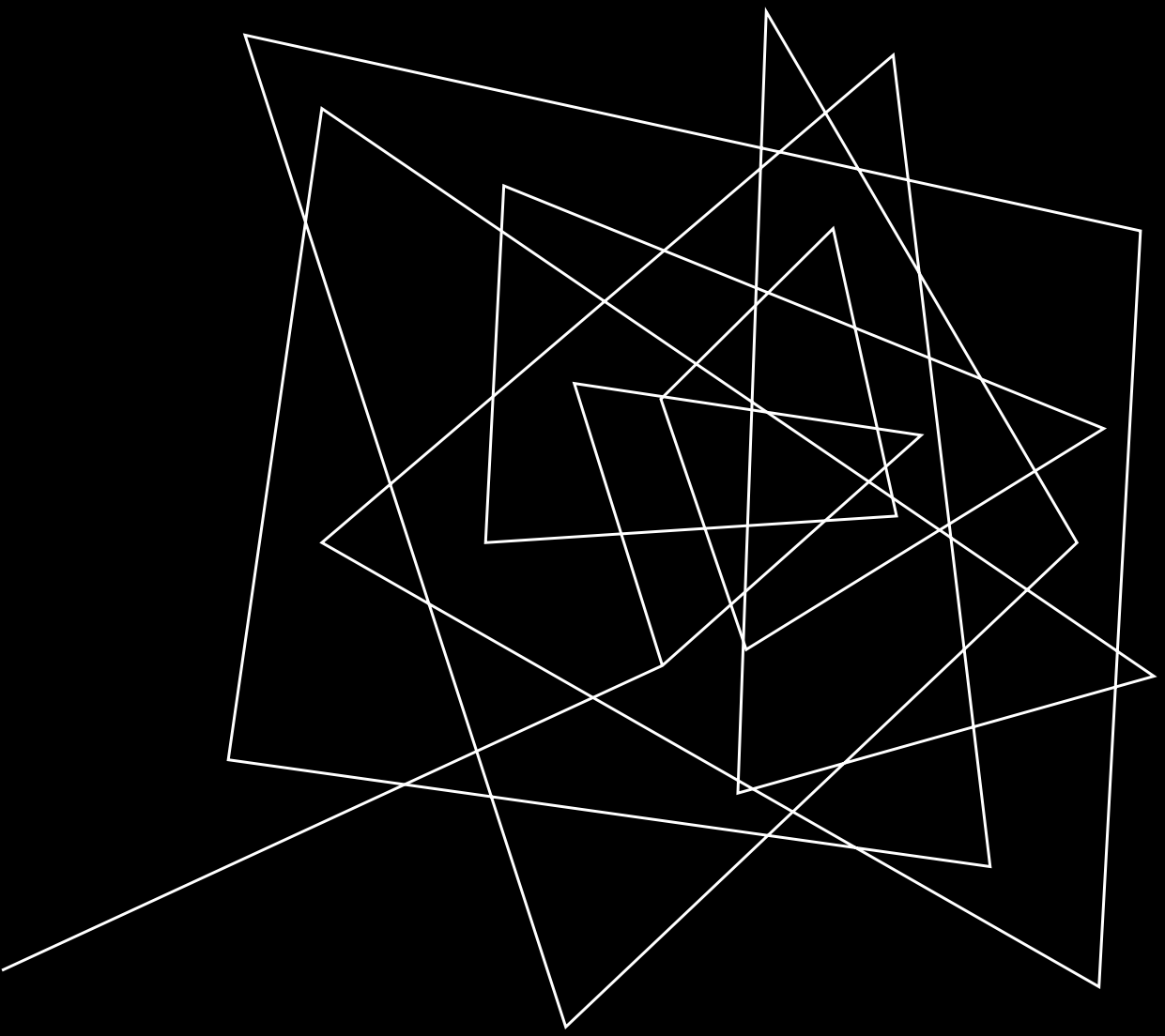
Posted on zig.news or Medium

INTRODUCTION

Zig is a general-purpose programming language and **toolchain** for maintaining robust, optimal and reusable software. – *http://ziglang.org*

The conference is called “Software You Can Love.”

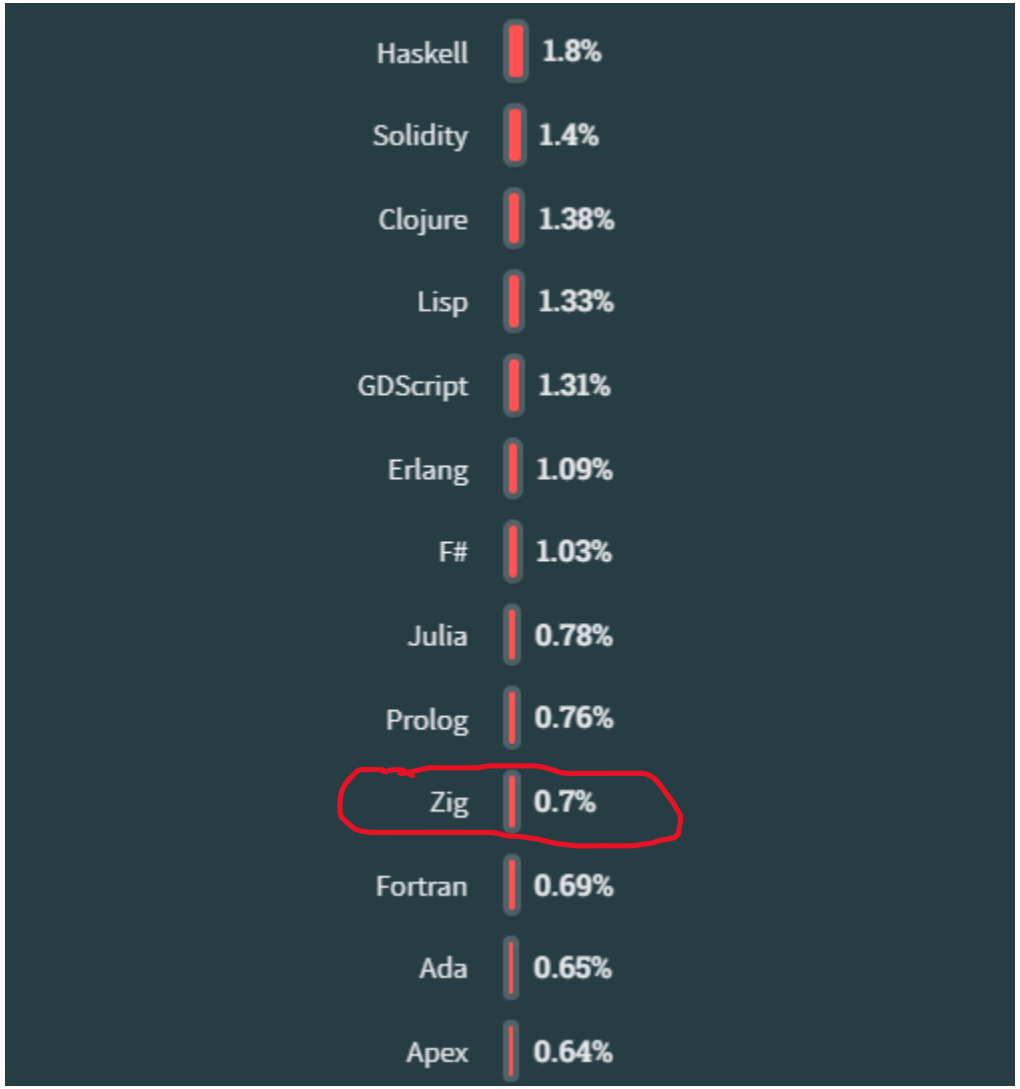
“Better C” and “Systems Programming”



WHY NOT ZIG

WTF is Zig

POPULARITY RANKING



0.10.0 — Current Release

0.11.0 — Next Release

EOY 2023 — 99.99% 1.0 will NOT be ready

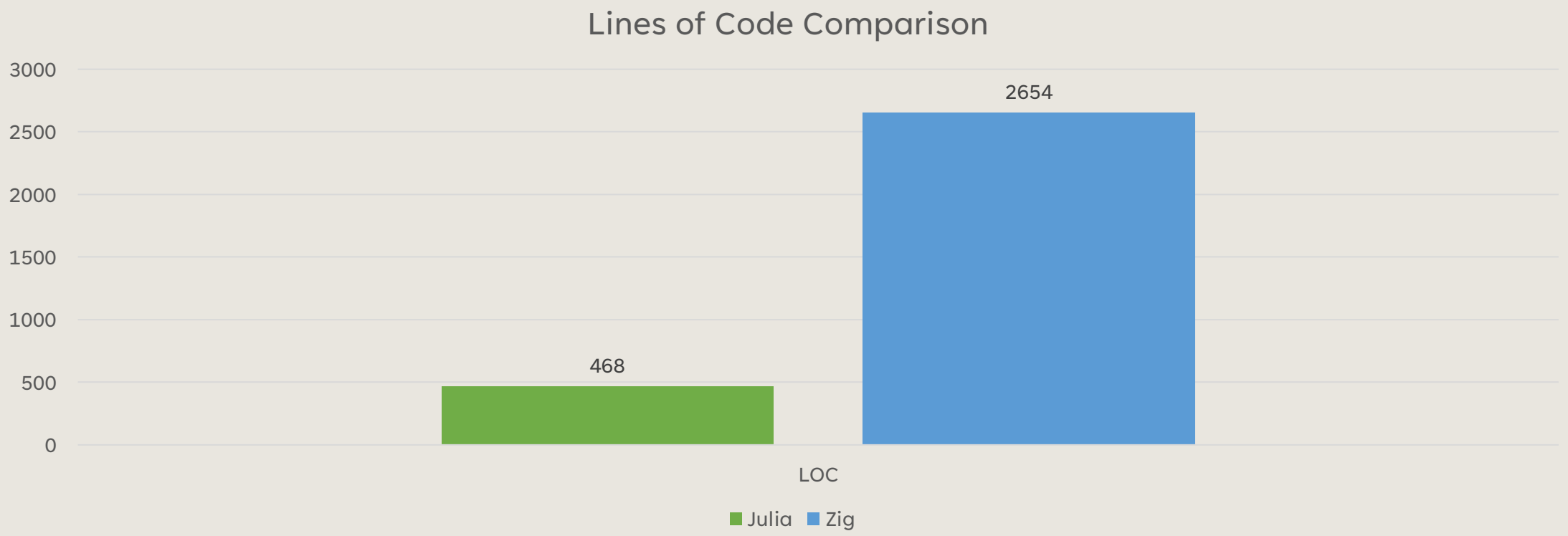
1.0 — Ready when it’s ready

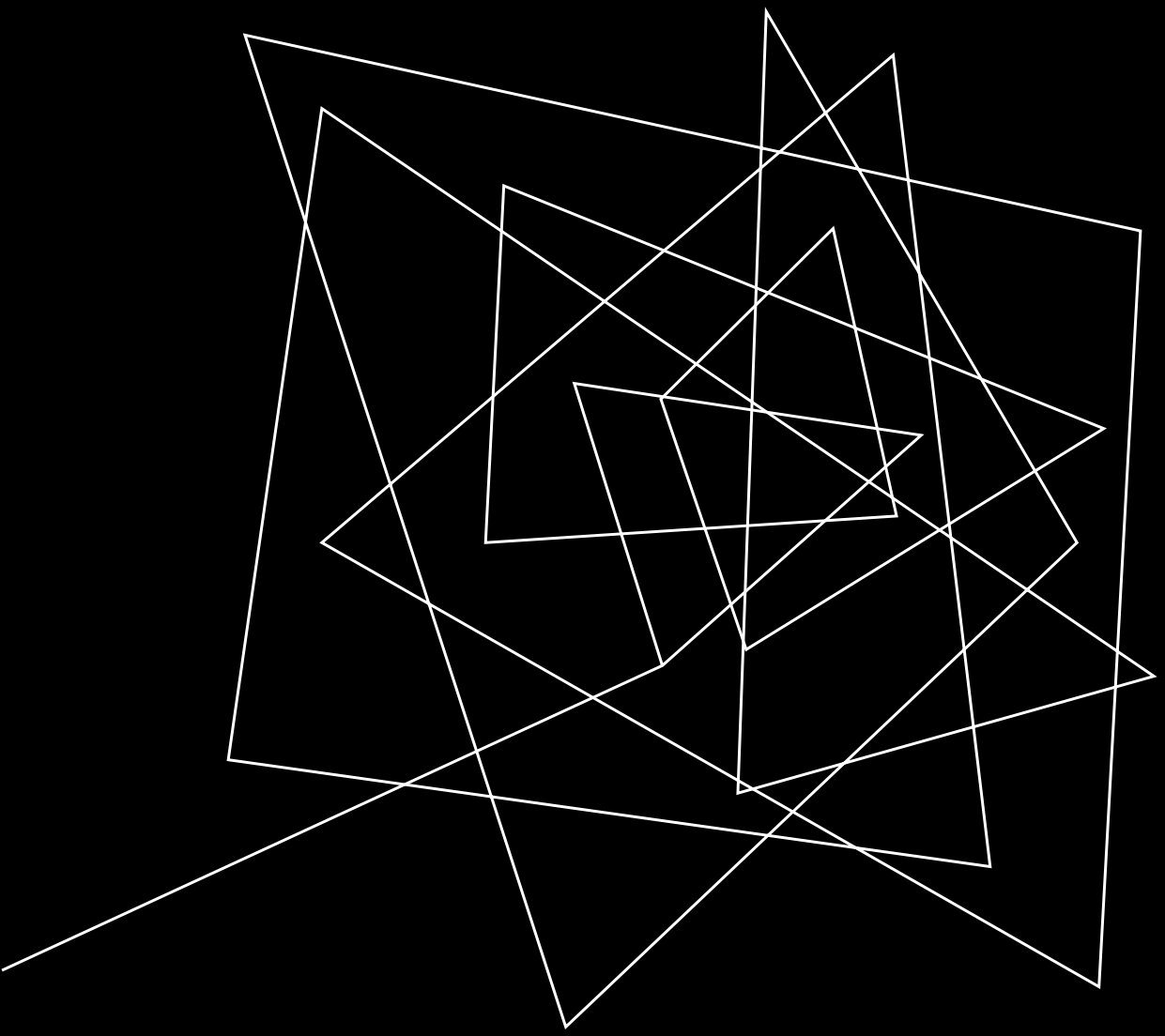
TIMELINE

WHY NOT ZIG

1. “Don’t use it in production” - Andrew
2. No corporate support except maybe 2 projects
3. Very young idealistic core team

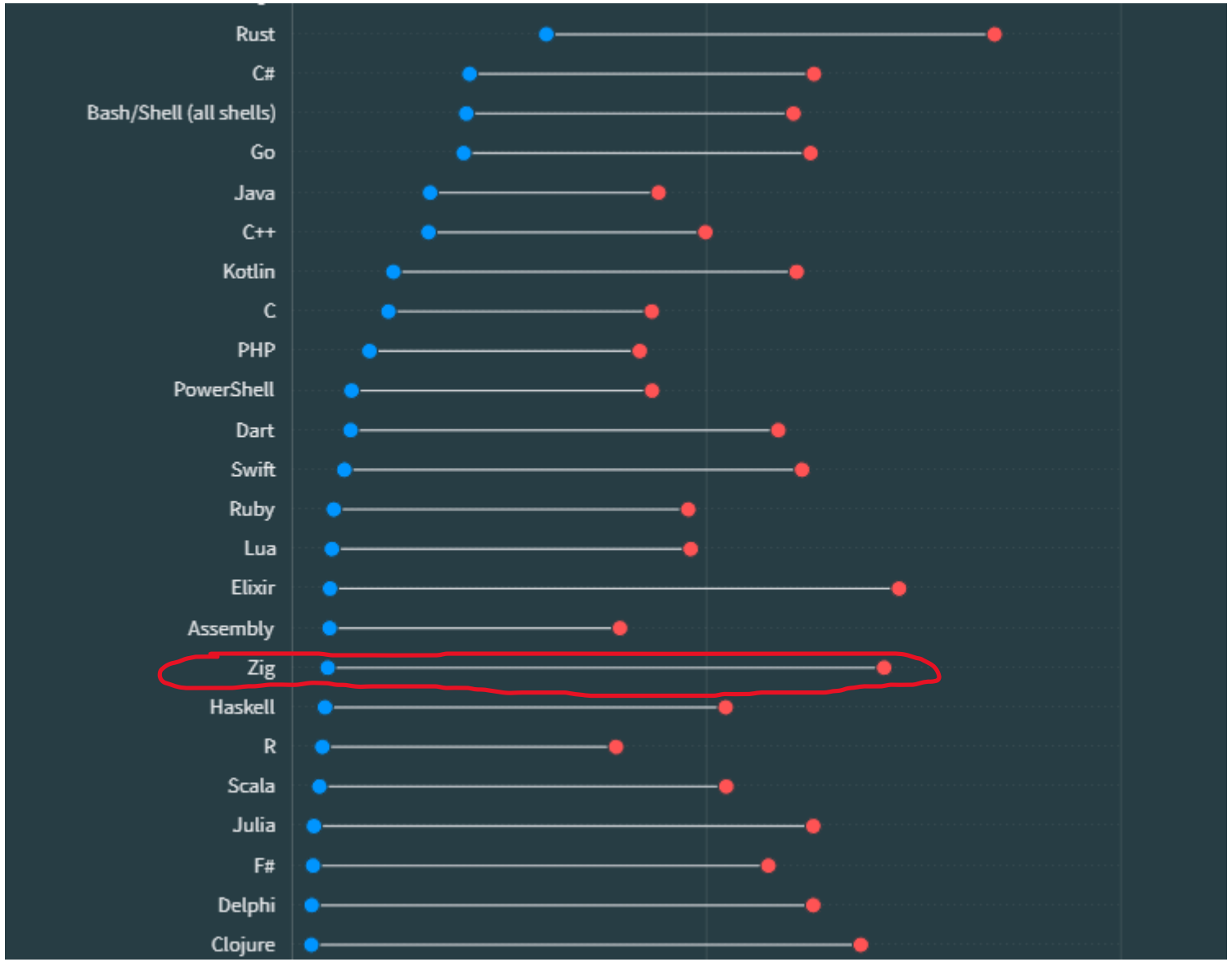
JULIA VS ZIG


Lines of Code Comparison
3000
2500
2000
1500
1000
500
0
468
2654
LOC
Julia
Zig



BUT WHY ZIG

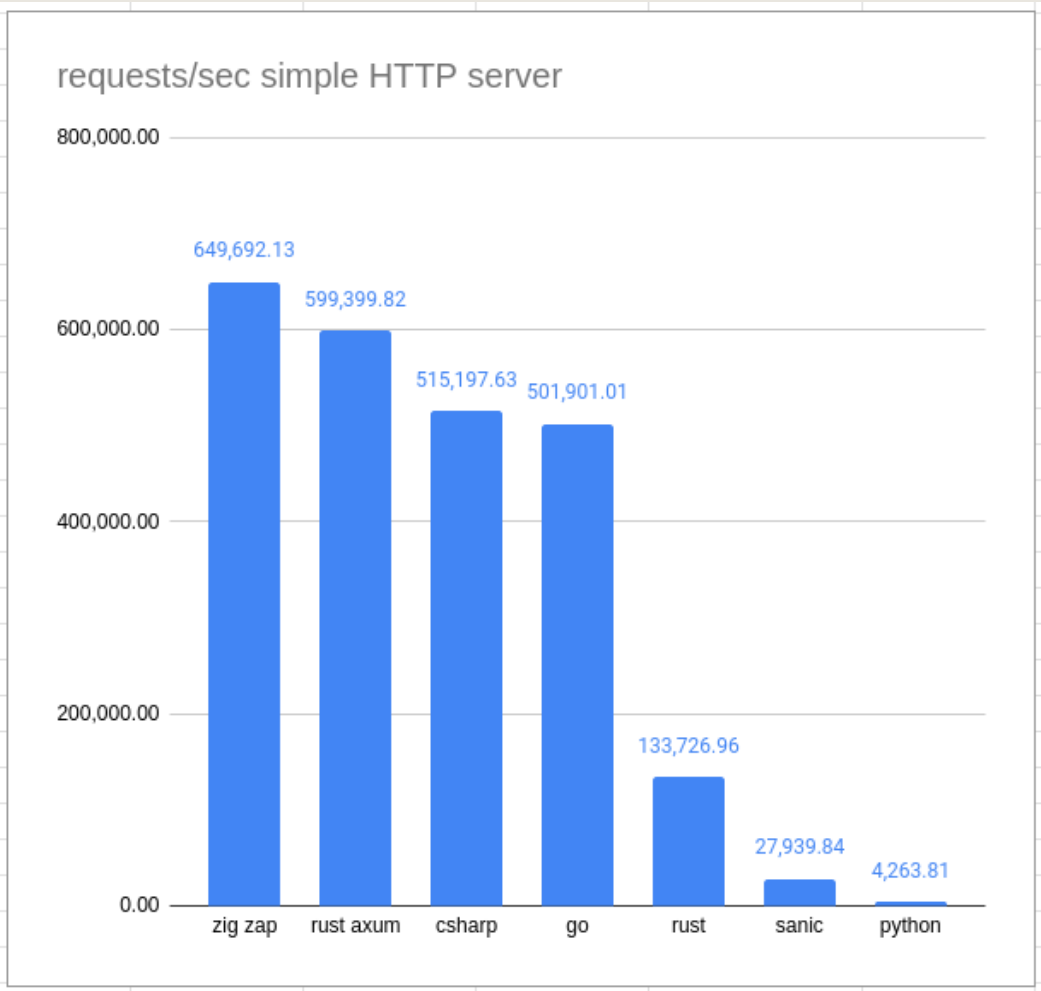
MOST ADMIRED #3

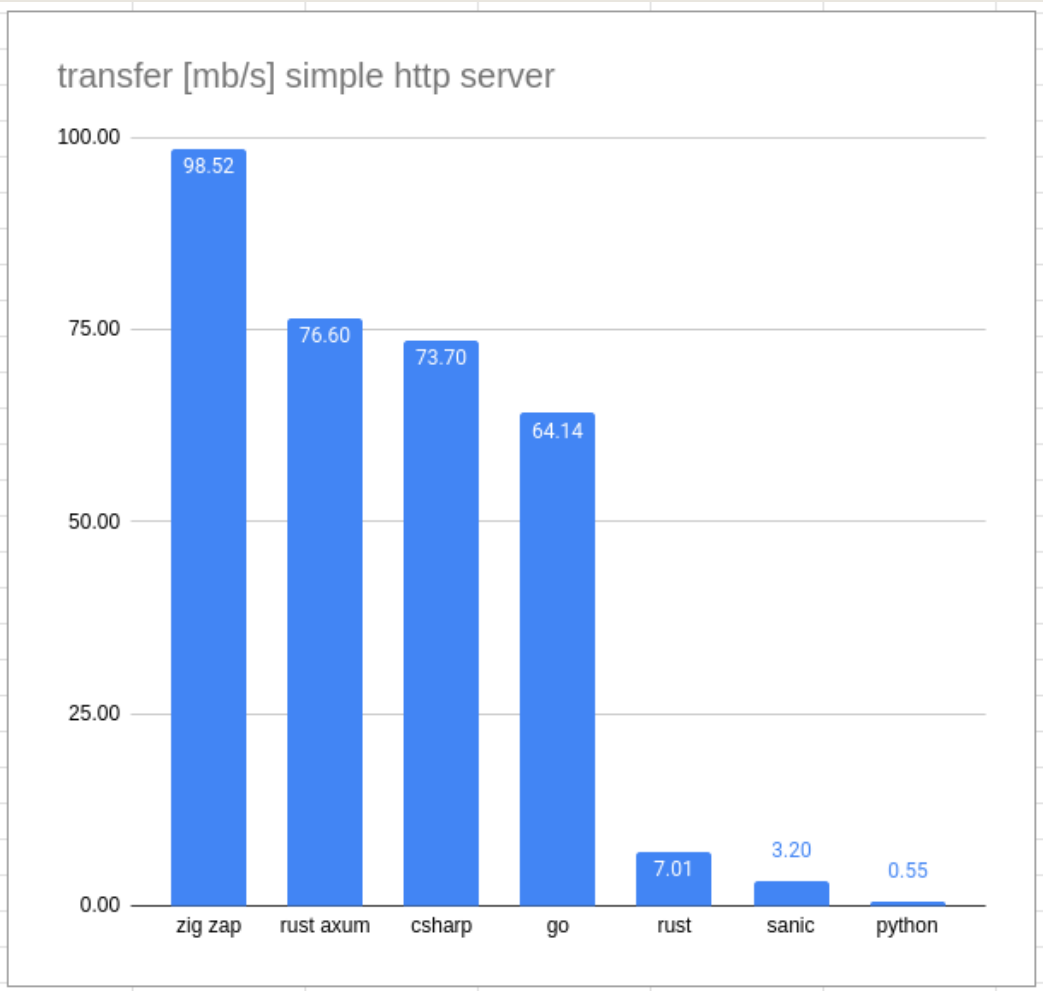


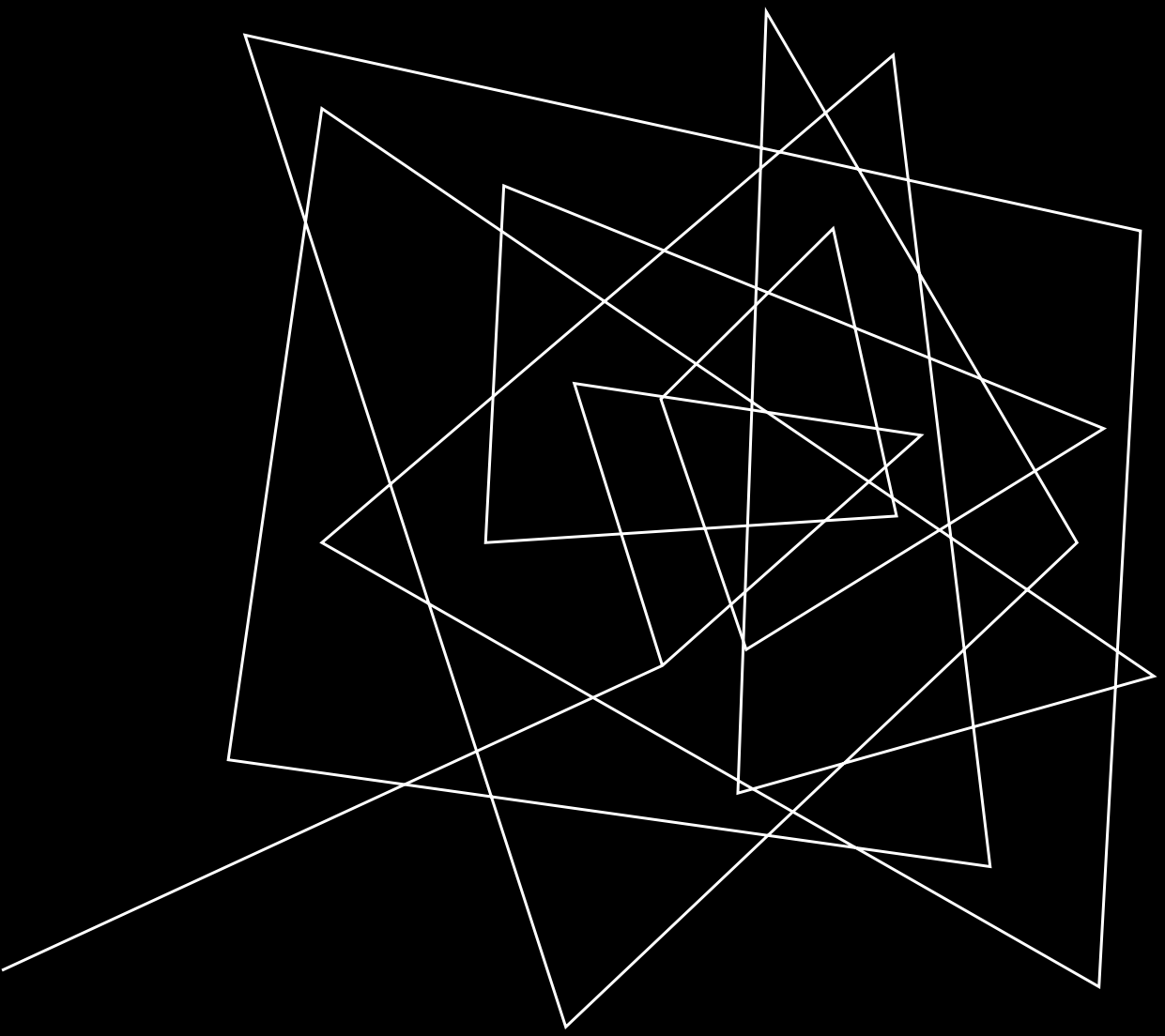
WHY ZIG

1. Smaller than C++ or Rust.
2. It makes sense.
3. Very welcoming community

WEBSERVER PERFORMANCE







WHAT IS ZIG

WHAT IS ZIG

- Somewhat simple language
- Comptime can almost do duck-typing
- C/C++ compiler and a build system for cross compilation

HELLO WORLD

```zig
const std = @import("std");

pub fn main() void {
    std.debug.print("Hello, world!\n", .{});
}
```

SOMEWHAT SIMPLE LANGUAGE

1. No hidden control flow
   - No @property functions
   - No operator overloading
   - No exception handling
2. No hidden memory allocation
   - You need allocator to do any allocation
   - No closures or other FP goodies
   - Zig is not considered a safe language
3. No preprocessor or macros
   - Comptime and builtins take that role

MORE COMPLEX THAN C

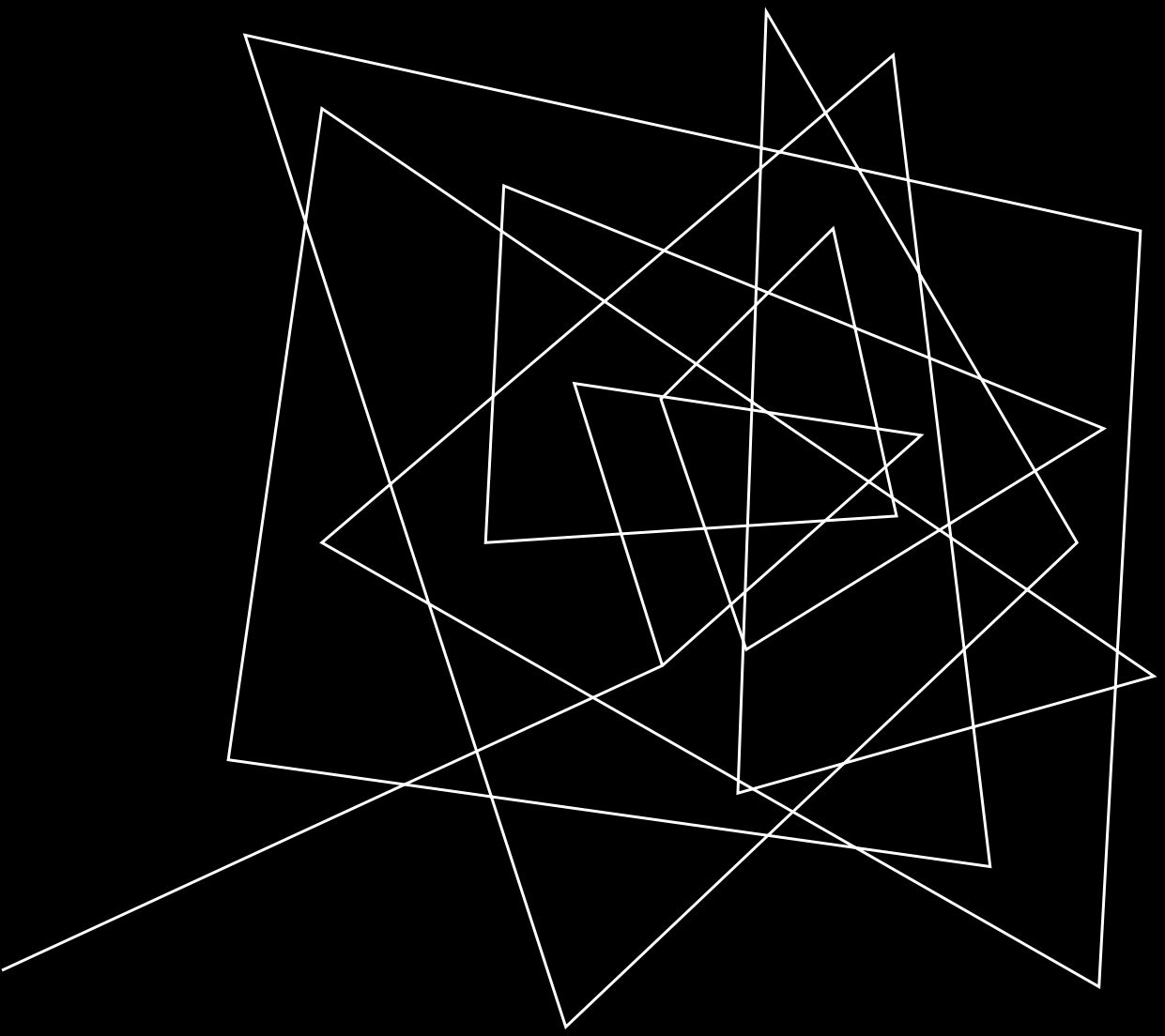
```zig
const std = @import("std");
const parseInt = std.fmt.parseInt;

test "parse integers" {
    const input = "123 67 89,99";
    const allocator = std.testing.allocator;

    var list = std.ArrayList(u32).init(allocator);
    defer list.deinit();

    var it = std.mem.tokenize(u8, input, " , ");
    while (it.next()) |num| {
        const n = try parseInt(u32, num, 10);
        try list.append(n);
    }

    const expected = [_]u32{ 123, 67, 89, 99 };

    for (expected, list.items) |exp, actual| {
        try std.testing.expectEqual(exp, actual);
    }
}
```

COMPTIME

```zig
const std = @import("std");

fn List(comptime T: type) type {
    return struct {
        items: []T,
        len: usize,
    };
}

pub fn main() void {
    var buffer: [10]i32 = undefined;
    var list = List(i32){
        .items = &buffer,
        .len = 0,
    };

    std.debug.print("{d}\n", .{list.items.len});
}
```

CURRENT IMPLEMENTATION

- LLVM as backend
- Link time optimizations and advanced CPU features are enabled by default
- Same toolchain for all supported targets and self-contained



ZIG AS C/C++ COMPILER

ZIG AS C COMPILER

```c
#include <stdio.h>
int main() {
   // printf() displays the string inside quotation
   printf("Hello, World!");
   return 0;
}
```

```
~/w/z/test ❯ zig cc hello.c -o "hello-c"
~/w/z/test ❯ ./hello-c
Hello world!
```

ZIG AS C++ COMPILER

```cpp
// Your First C++ Program

#include <iostream>

int main() {
    std::cout << "Hello World!";
    return 0;
}
```

```
~/w/z/test ❯ zig c++ hello.cpp -o "hello-cpp"
~/w/z/test 1s ❯ ./hello-cpp
Hello world!
```

ZIG AS C CROSS-COMPILER

```
I ~/w/z/test ❯ zig cc hello.c -o "hello-c.exe" -target x86_64-windows
```

```
I ~/w/z/test ❯ cp hello-c.exe /mnt/c/Users/edlyu/Downloads/
```

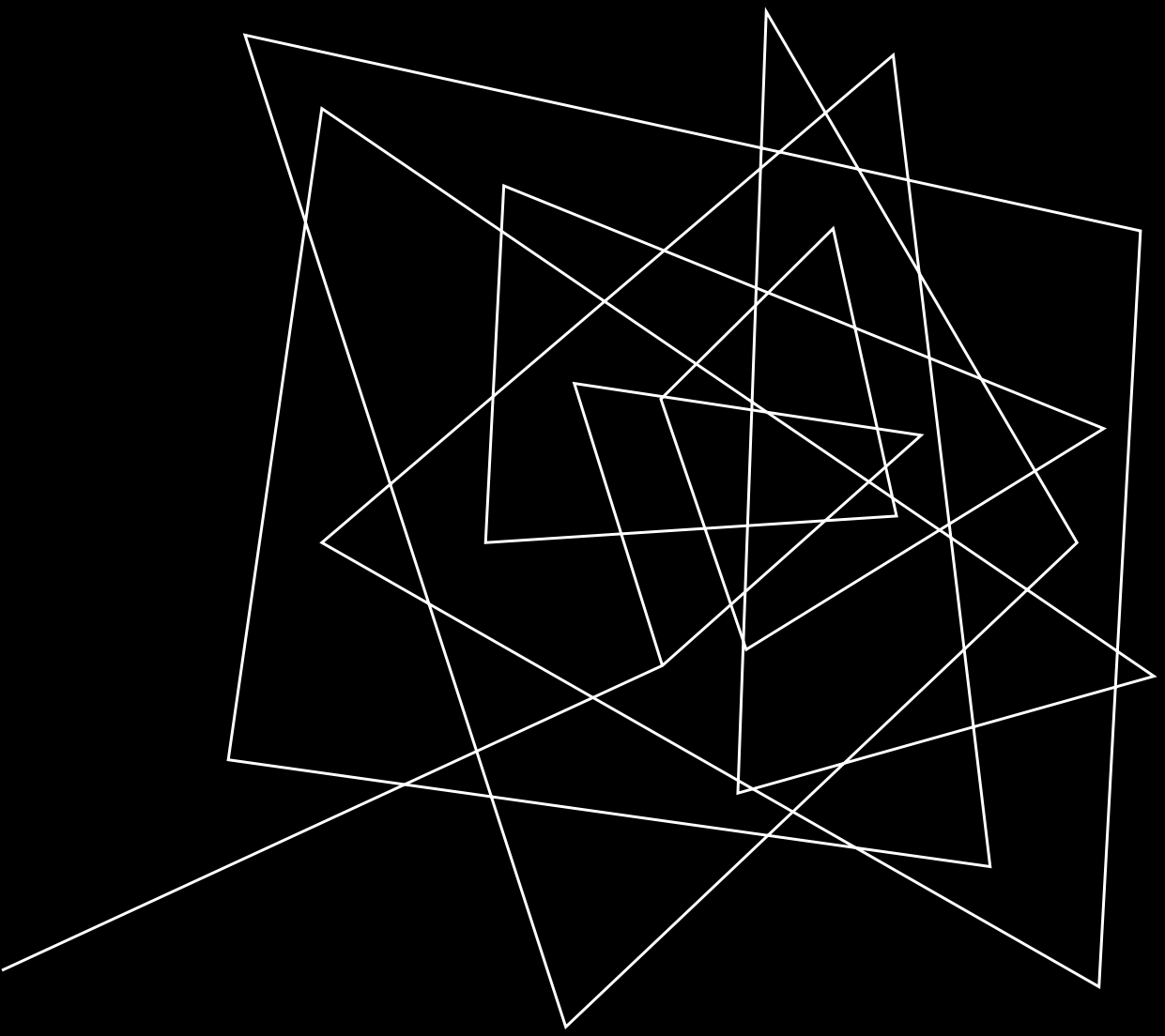
```
PS 2: C:\..\edlyu\Downloads> .\hello-c.exe
Hello world!
```

ZIG AS C++ CROSS-COMPILER

```
I ~/w/z/test ❯ zig c++ hello.cpp -o "hello-cpp.exe" -target x86_64-windows
```

```
I ~/w/z/test 12.4s ❯ cp hello-cpp.exe /mnt/c/Users/edlyu/Downloads/
```

```
PS 3: C:\..\edlyu\Downloads> .\hello-cpp.exe
Hello world!
```



ZIG IN UBER

[Bootstrapping Uber’s Infrastructure on arm64 with Zig | Uber Blog]

“REDUCE UBER’S COMPUTE COSTS, INCREASE CAPACITY DIVERSITY, AND MODERNIZE OUR PLATFORM BY DEPLOYING SOME PRODUCTION APPLICATIONS ON ARM64.”

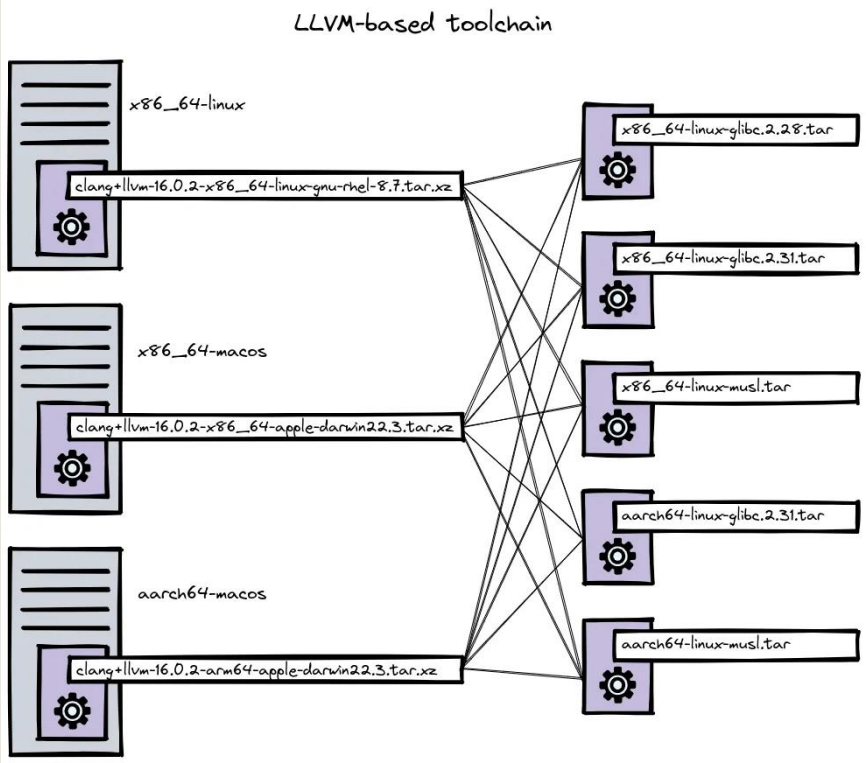
Bootstrapping Uber’s Infrastructure on arm64 with Zig | Uber Blog

WHY ZIG

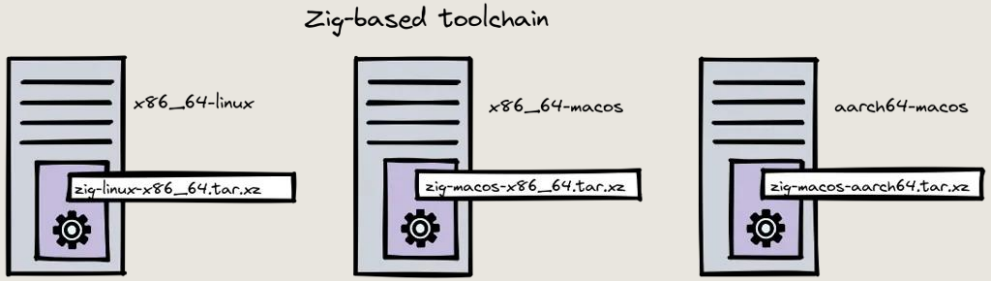
- Go Monorepo
- Goal is to be hermetic (system-independent)
- Nov 2021 – June 2023
- C++ toolchain for x86_64 & arm64
  - Zig cc
- Based on bazel-zig-cc (https://github.com/ajbouh/bazel-zig-cc/)

Bootstrapping Uber's Infrastructure on arm64 with Zig | Uber Blog

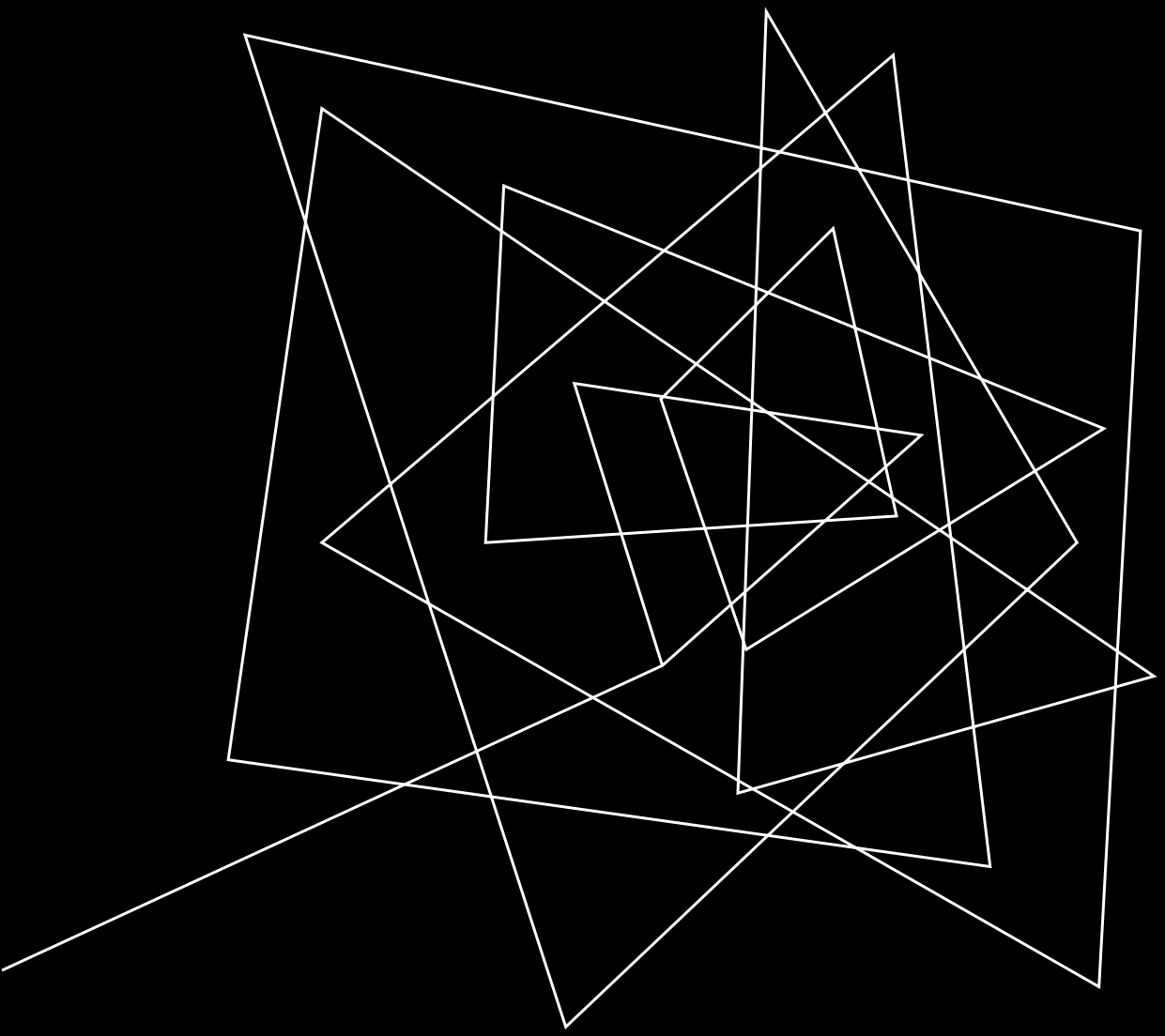
LLVM-Based C/C++ Toolchain



Zig-Based Toolchain



[Bootstrapping Uber's Infrastructure on arm64 with Zig | Uber Blog]



ZIG COMPILER CASE STUDY

BUILDING GRPC USING ZIG

Complex Build

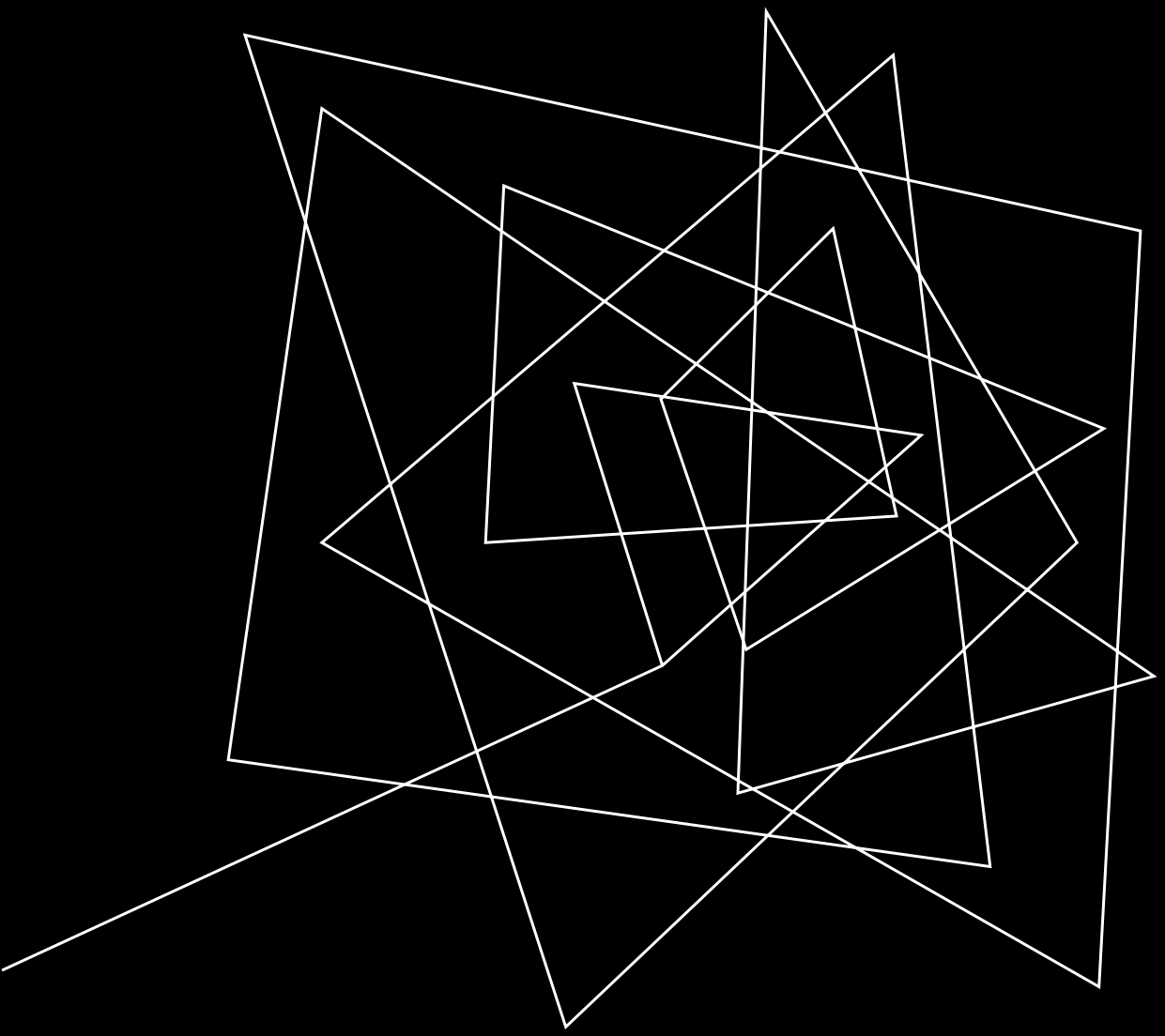
- Uses CMake
- C++
- Many dependencies

24 Dependencies

```
I  ~/w/z/grpc/third_party v1.56.0• ❯ ls
ABSEIL_MANUAL.md        googleapis/                re2/
BUILD                   googletest/                rules_python.patch
README.md               incremental.BUILD          six.BUILD
abseil-cpp/             libprotobuf_mutator.BUILD  toolchains/
address_sorting/        libuv/                     twisted.BUILD
android/                libuv.BUILD                upb/
benchmark/              objective_c/               utf8_range/
bloaty/                 opencensus-proto/          xds/
boringssl-with-bazel/   opentelemetry/             xxhash/
cares/                  protobuf/                  yaml.BUILD
constantly.BUILD        protobuf.patch             zlib/
cython.BUILD            protobuf2.patch            zlib.BUILD
enum34.BUILD            protoc-gen-validate.patch  zope_interface.BUILD
envoy-api/              py/
futures.BUILD           rake-compiler-dock/
I  ~/w/z/grpc/third_party v1.56.0• ❯
```

zig c++

```
CC="zig cc -mcrc32" CXX="zig c++ -mcrc32" cmake \
  -DgRPC_INSTALL=ON \
  -DgRPC_BUILD_TESTS=OFF \
  -DOPENSSL_NO_ASM=ON \
  -DCMAKE_INSTALL_PREFIX=/home/edyu/.env \
  ../..
```



ZIG BUILD SYSTEM

ZIG AS A BUILD SYSTEM

```
I  ~/w/z/t/build ❯ zig init-exe
```

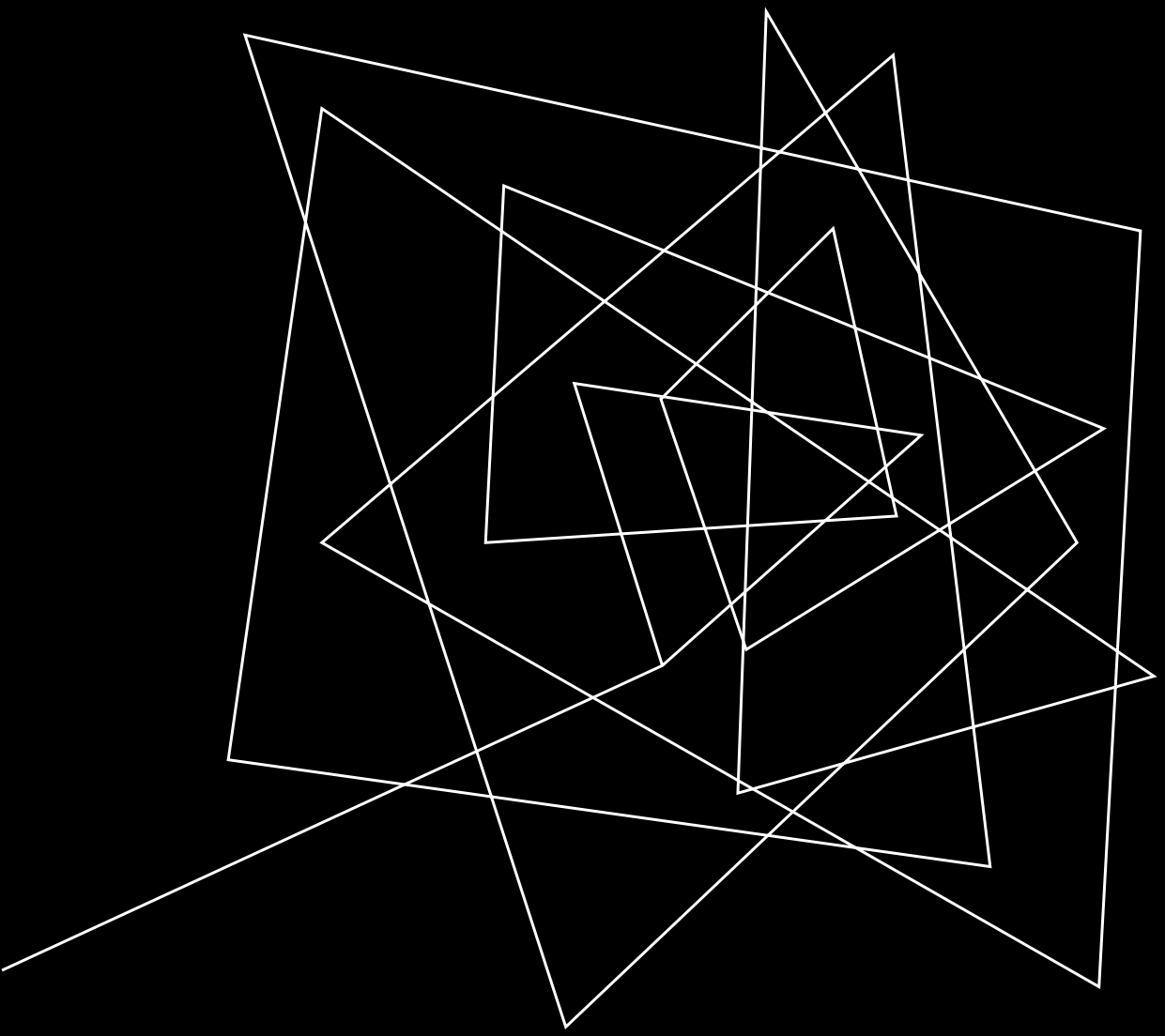
```
I  ~/w/z/t/build ❯ tree
.
├── build.zig
└── src
    └── main.zig

1 directory, 2 files
I  ~/w/z/t/build ❯
```

BUILD.ZIG

```zig
const std = @import("std");

pub fn build(b: *std.Build) void {
    const target = b.standardTargetOptions(.{});

    const optimize = b.standardOptimizeOption(.{});

    const exe = b.addExecutable(.{
        .name = "myexe",
        .root_source_file = .{ .path = "src/main.zig" },
        .target = target,
        .optimize = optimize,
    });

    b.installArtifact(exe);

    const run_cmd = b.addRunArtifact(exe);

    run_cmd.step.dependOn(b.getInstallStep());

    if (b.args) |args| {
        run_cmd.addArgs(args);
    }

    const run_step = b.step("run", "Run the app");
    run_step.dependOn(&run_cmd.step);
}
```

```zig
    const libduckdb = b.addSharedLibrary(.{
        .name = "duckdb",
        .target = target,
        .optimize = optimize,
    });
    libduckdb.addCSourceFiles(duckdb_sources.items, &.{});
    libduckdb.addIncludePath("extension/httpfs/include");
    libduckdb.addIncludePath("extension/icu/include");
    libduckdb.addIncludePath("extension/icu/third_party/icu/common");
    libduckdb.addIncludePath("extension/icu/third_party/icu/i18n");
    libduckdb.addIncludePath("extension/parquet/include");
    libduckdb.addIncludePath("third_party/httplib");
    libduckdb.addIncludePath("third_party/libpg_query/include");
    libduckdb.addIncludePath("/opt/homebrew/opt/openssl@3/");
    libduckdb.defineCMacro("BUILD_HTTPFS_EXTENSION", "TRUE");
    libduckdb.defineCMacro("BUILD_ICU_EXTENSION", "TRUE");
    libduckdb.defineCMacro("BUILD_PARQUET_EXTENSION", "TRUE");
    libduckdb.defineCMacro("duckdb_EXPORTS",null);
    libduckdb.defineCMacro("DUCKDB_MAIN_LIBRARY",null);
    libduckdb.defineCMacro("DUCKDB",null);

    if (target.isWindows() or builtin.os.tag == .windows){
        libduckdb.addIncludePath("third_party/openssl/include");
        libduckdb.addObjectFile("third_party/openssl/lib/libcrypto.lib");
        libduckdb.addObjectFile("third_party/openssl/lib/libssl.lib");
        libduckdb.addObjectFile("third_party/win64/ws2_32.lib");
        libduckdb.addObjectFile("third_party/win64/crypt32.lib");
        libduckdb.addObjectFile("third_party/win64/cryptui.lib");
        libduckdb.step.dependOn(
            &b.addInstallFileWithDir(
                .{.path = "third_party/openssl/lib/libssl-3-x64.dll"},
                .bin,
                "libssl-3-x64.dll",
            ).step
        );
        libduckdb.step.dependOn(
            &b.addInstallFileWithDir(
                .{.path = "third_party/openssl/lib/libcrypto-3-x64.dll"},
                .bin,
                "libcrypto-3-x64.dll",
            ).step
        );
        libduckdb.step.dependOn(
            &b.addInstallFileWithDir(
                .{.path = "third_party/openssl/lib/libcrypto-3-x64.dll"},
                .bin,
                "libcrypto-3-x64.dll",
            ).step
        );
    }

    if (target.isLinux() or builtin.os.tag == .linux){
        libduckdb.addIncludePath("third_party/openssl/include");
        libduckdb.defineCMacro("BUILD_JEMALLOC_EXTENSION", "TRUE");
        libduckdb.addIncludePath("extension/jemalloc/include");
        libduckdb.addIncludePath("extension/jemalloc/jemalloc/include");
        libduckdb.linkLibrary(jemalloc_extension);
        libduckdb.linkSystemLibrary("ssl");
        libduckdb.linkSystemLibrary("crypto");
    }

    if (target.isDarwin() or builtin.os.tag == .macos){
        libduckdb.addIncludePath("/opt/homebrew/opt/openssl@3/");
        libduckdb.addLibraryPath("/opt/homebrew/opt/openssl@3/lib");
        libduckdb.linkSystemLibrary("ssl");
        libduckdb.linkSystemLibrary("crypto");
    }

    libduckdb.linkLibrary(fastpforlib);
    libduckdb.linkLibrary(fmt);
    libduckdb.linkLibrary(fsst);
    libduckdb.linkLibrary(hyperloglog);
    libduckdb.linkLibrary(mbedtls);
    libduckdb.linkLibrary(miniz);
    libduckdb.linkLibrary(pg_query);
    libduckdb.linkLibrary(re2);
    libduckdb.linkLibrary(utf8proc);
    libduckdb.linkLibrary(parquet_extension);
    libduckdb.linkLibrary(httpfs_extension);
    libduckdb.linkLibrary(icu_extension);
    _ = try basicSetup(b,libduckdb);
    libduckdb.linkLibC();
}
```

duckdb-zig-build/build.zig at master · kimmolinna/duckdb-zig-build (github.com)



IMPORT C++ LIBRARY

Based on https://stackoverflow.com/questions/73467232/how-to-incorporate-the-c-standard-library-into-a-zig-program

HELLO.CPP

hello.cpp

```
#include <iostream>

extern "C" void helloWorld(void) {
  std::cout << "Hello world!";
}
```

HELLO.H

hello.h

```
1 void helloWorld(void);
```

MAIN.ZIG

main.zig

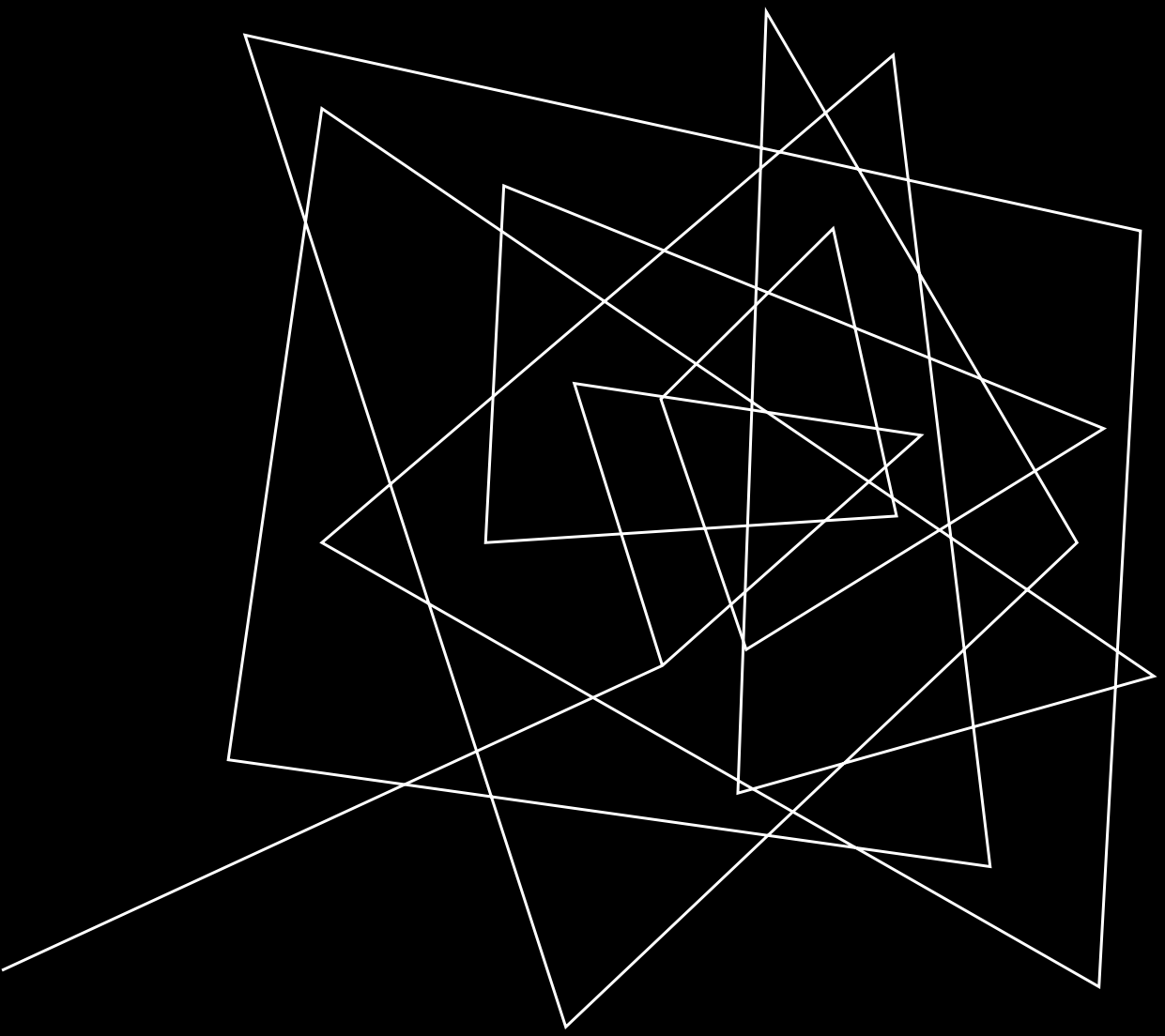
```zig
const std = @import("std");
const cpp = @cImport({
    @cInclude("hello.h");
});

pub fn main() !void {
    cpp.helloWorld();
}
```

BUILD.ZIG

```zig
const std = @import("std");

pub fn build(b: *std.Build) void {
    const target = b.standardTargetOptions(.{});

    const optimize = b.standardOptimizeOption(.{});

    const exe = b.addExecutable(.{
        .name = "helloworld",
        .root_source_file = .{ .path = "src/main.zig" },
        .target = target,
        .optimize = optimize,
    });

    exe.linkLibCpp();
    exe.addIncludePath("src");
    exe.addCSourceFile("src/hello.cpp", &.{});

    b.installArtifact(exe);
}
```

BUILD AND RUN THE PROGRAM

```
I  ~/w/z/t/libhello 5.3s ❯ zig build
I  ~/w/z/t/libhello 4.2s ❯ ./zig-out/bin/helloworld
Hello world!
I  ~/w/z/t/libhello ❯
```



ZIG PACKAGE MANAGER

1 List your dependencies

2 Add to your build

3 Import the library

4 Call a function on the package

HOW TO USE A PACKAGE

1. LIST YOUR DEPENDENCY IN BUILD.ZIG.ZON

```
.{
    .name = "myproject",
    .version = "0.0.1",

    .dependencies = .{
        // zap v0.1.8-pre
        .zap = .{
            .url = "https://github.com/zigzap/zap/archive/refs/tags/v0.1.8-pre.tar.gz",
            .hash = "1220e30645c293c943dccc9d4394f7088679cf0969c63a7f4b23f72e375744123729",
        },
    },
}
```

2. ADD TO YOUR BUILD IN BUILD.ZIG

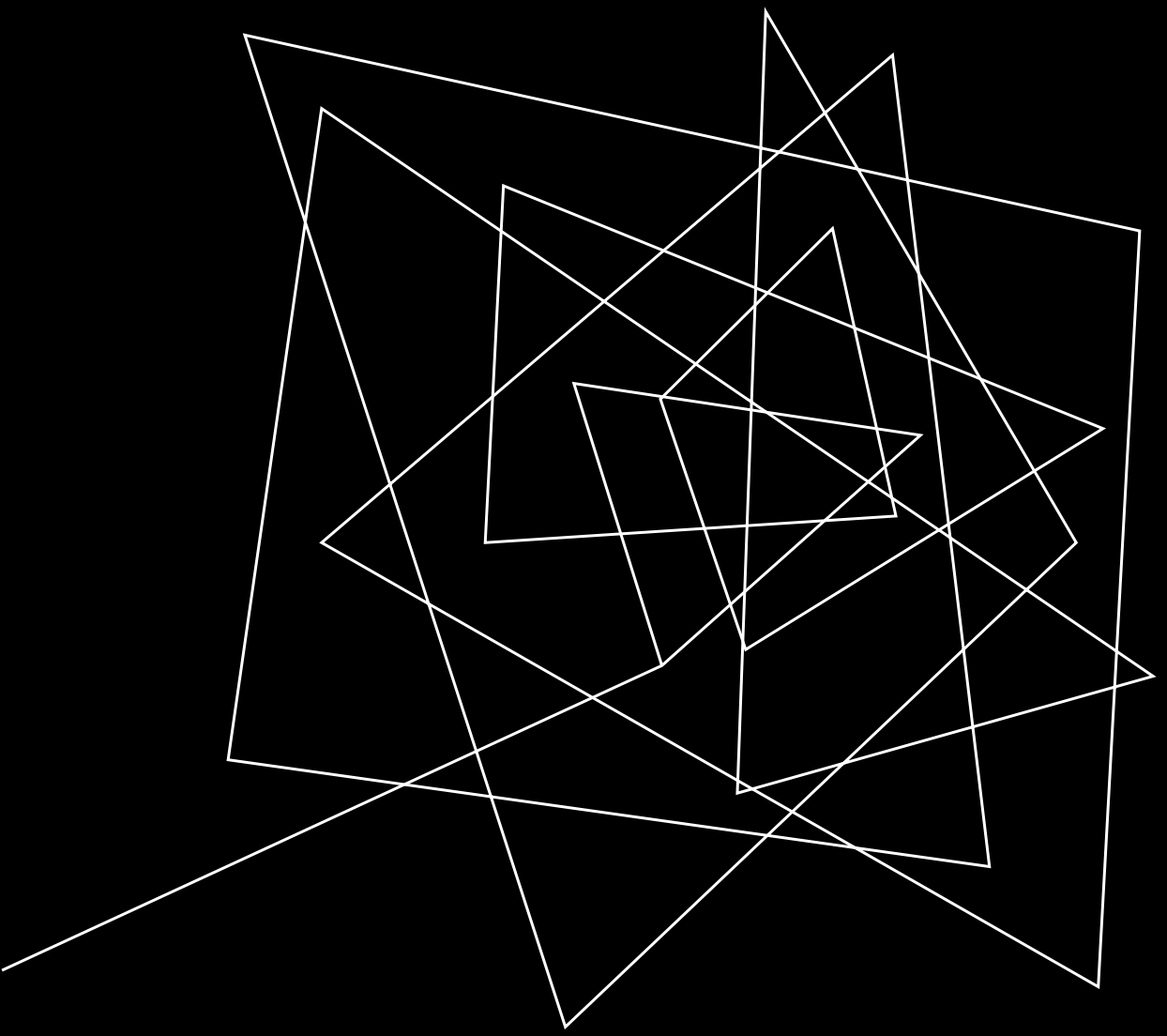
```
26
1    const zap = b.dependency("zap", .{
2        .target = target,
3        .optimize = optimize,
4    });
5    exe.addModule("zap", zap.module("zap"));
6    exe.linkLibrary(zap.artifact("facil.io"));
7
```

3. IMPORT THE LIBRARY IN YOUR SOURCE CODE

```
1 const zap = @import("zap");
```

4. CALL A FUNCTION FROM THE PACKAGE

```zig
const Context = struct {
    user: ?User = null,
    session: ?std.StringHashMap(void) = null,
};

// we create a Handler type based on our Context
const Handler = zap.Middleware.Handler(Context);
```



ZIG PACKAGE MANAGER EXAMPLE

PROBLEM STATEMENT

You have a library

This library is developed by someone else

This library is released as a dynamic library

You want to write a wrapper for the library in another language – zig

You want to use that library in your program

You want to isolate the library in its own package

You want to write a wrapper for the library and place that in its own library

You want to use only the wrapper as a package in your project

EXAMPLE: DuckDB

http://www.animationconnection.com/view/duck-hunting

PACKAGE MANAGER USE CASE

libduckdb

Upstream is written in C++

Upstream released in binary as dynamic library: libduckdb.so, duckdb.h, duckdb.hpp

Release as a Zig package

duckdb.zig

Simple Zig wrapper for function exposed in libduckdb.so

Dependent on libduckdb zig package

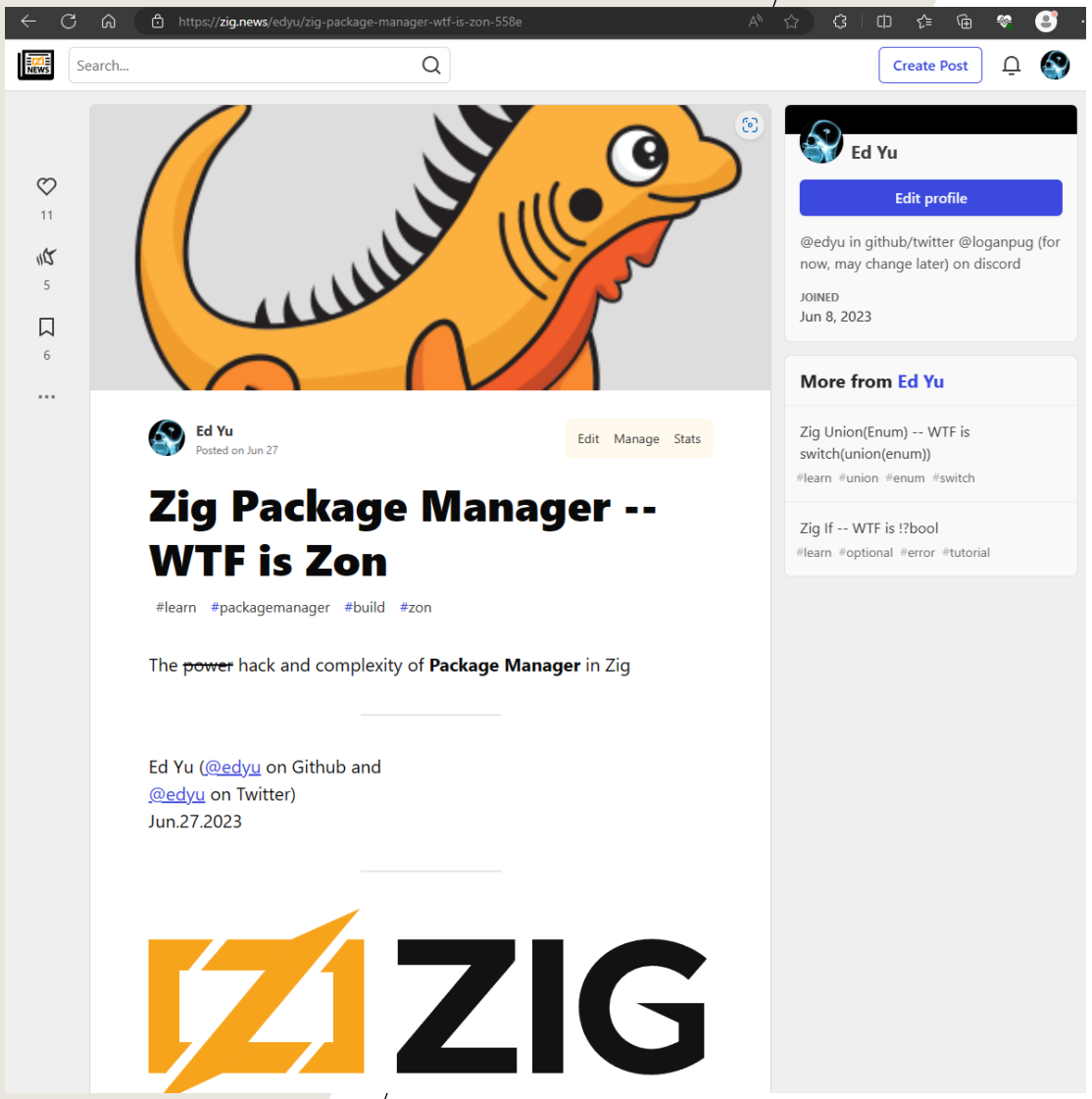
Release as a Zig package

hunter

Uses duckdb.zig package

Written in Zig

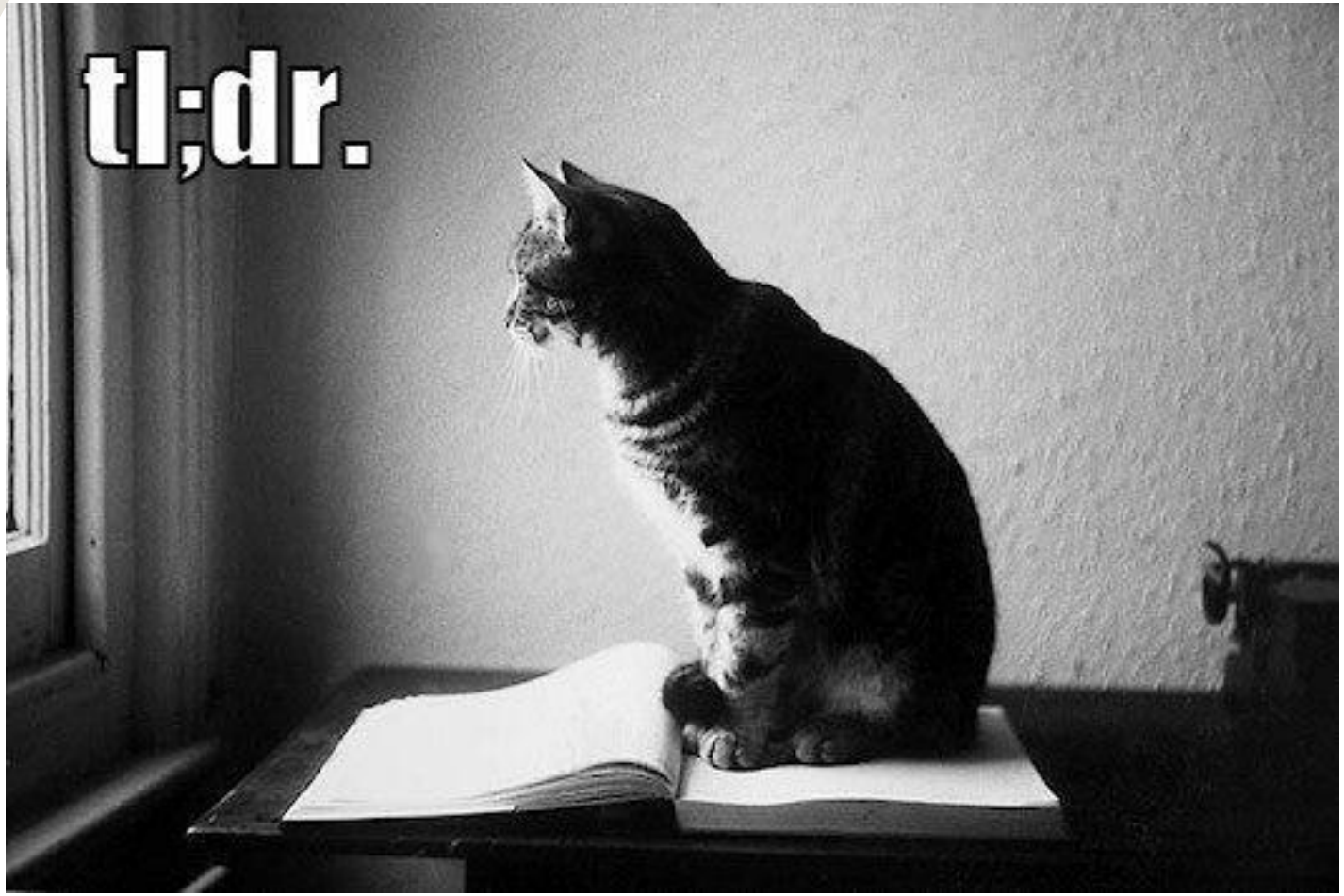
Ideally hide as much language detail from libduckdb as possible both in source code and build



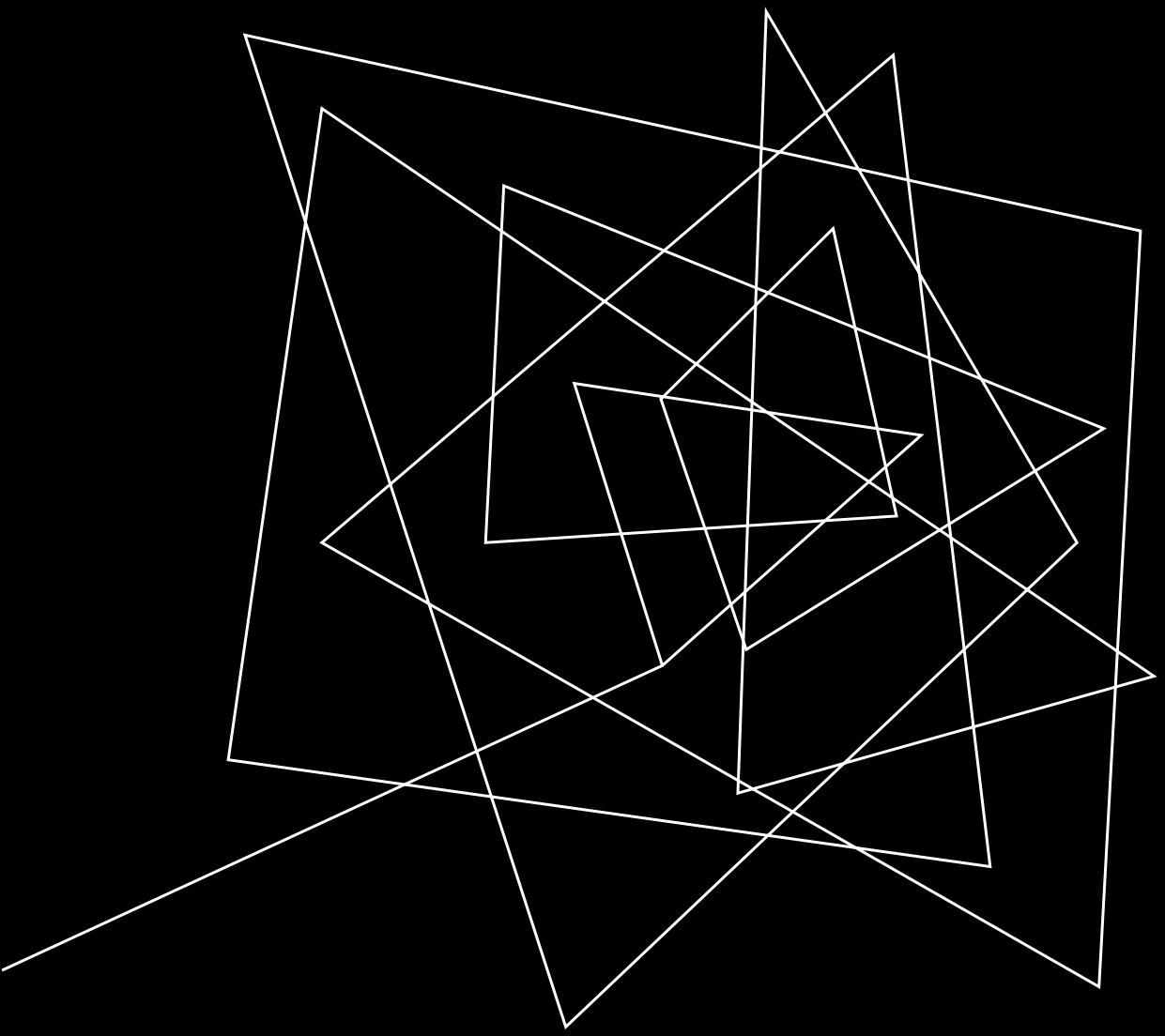
READ IT ON ZIG.NEWS

https://zig.news/edyu/zig-package-manager-wtf-is-zon-558e

IT’S COMPLICATED



https://knowyourmeme.com/memes/tldr

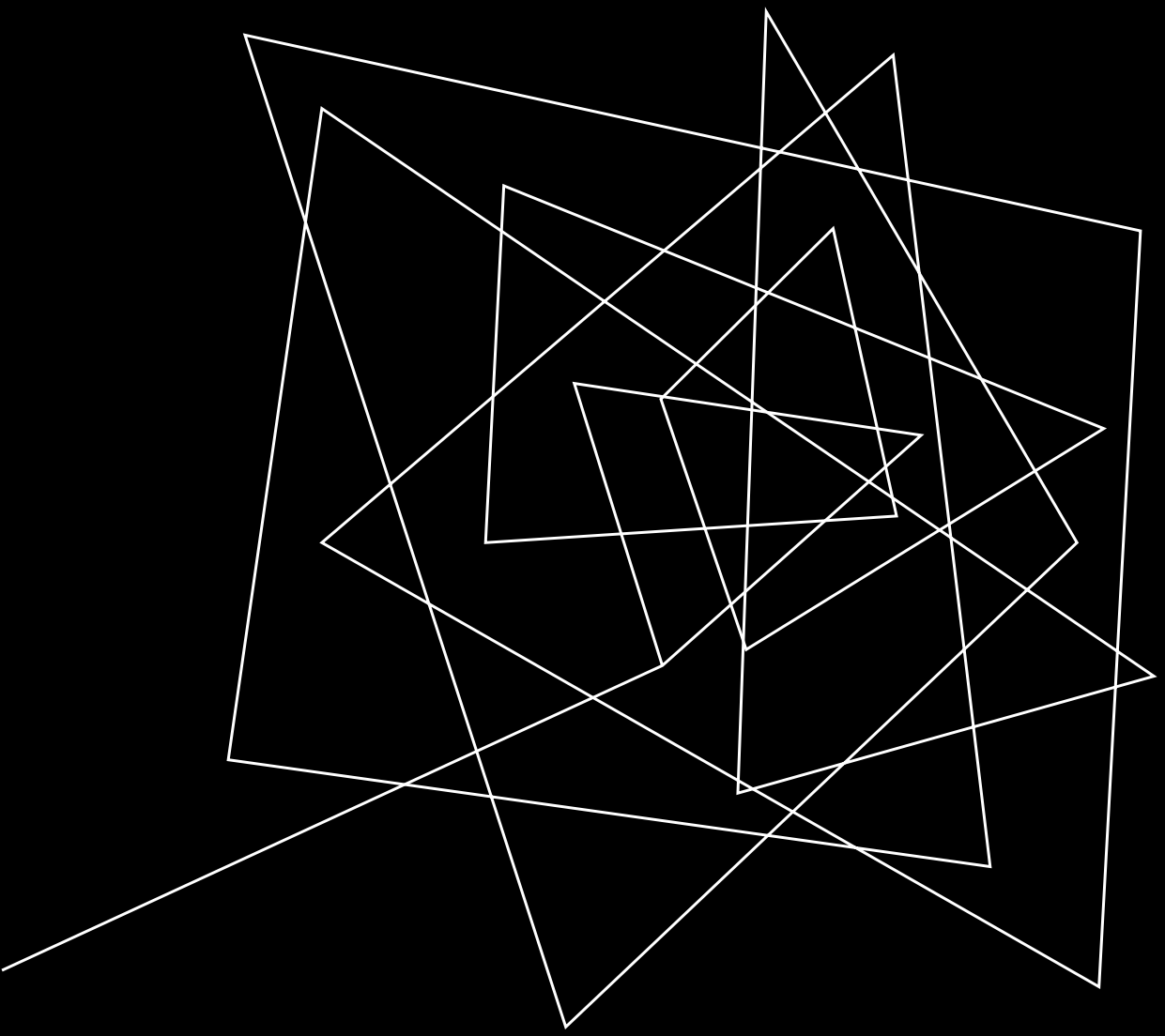


FUTURE OF ZIG

FUTURE OF ZIG FOR C/C++ PROGRAMERS

#16270: Make main zig executable no longer depend on LLVM, LLD, and Clang libraries @andrwrk

zig cc/zig c++ will be an independent package



TRY ZIG

ZIG RESOURCES

- Github: github.com/ziglang/zig
- Discords:
  - Zig - discord.gg/zig
  - SYCL - discord.gg/YpAnJRv4
- Blogs: zig.news
- Forum: ziggit.dev

ZIG ON UBUNTU OR WSL

Try snap:

enable systemd

sudo snap install zig --edge --classic

THANK YOU

Ed Yu (linkedin/in/edlyu)

@edyu (github & twitter)

@ed.yu (discord) [loganpug]

ed@beachglass.io