CROSS-PLATFORM PITFALLS AND HOW TO AVOID THEM

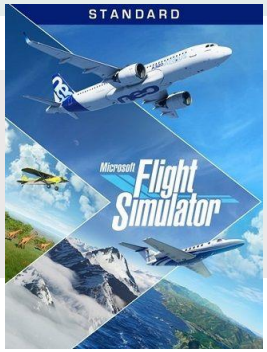
Erika Sweet (she/her)

Microsoft C++ Team

Welcome to CppCon 2020!

Visit our table at the Expo Hall
https://aka.ms/cppcon/expo

- Meet the Microsoft C++ team
- Ask any questions
- Discuss the latest announcements



Take our survey
https://aka.ms/cppcon

Chance to win 1 of 5 copies of Microsoft Flight Simulator

$$^{10}C_1 * {}^{6}C_2 * {}^{8}C_2 * {}^{10}C_1 * {}^{4}C_1 = 168{,}000$$

$$^{10}C_1 * {}^{6}C_2 * {}^{8}C_2 * {}^{10}C_1 * {}^{4}C_1 = 168{,}000$$

$$^{10}C_1 * {}^{6}C_2 * {}^{8}C_2 * {}^{10}C_1 * {}^{4}C_1 = 168{,}000$$

$$^{10}C_1 * {}^{6}C_2 * {}^{8}C_2 * {}^{10}C_1 * {}^{4}C_1 = 168{,}000$$

$$^{10}C_1 * {}^{6}C_2 * {}^{8}C_2 * {}^{10}C_1 * {}^{4}C_1 = 168{,}000$$

$$ {}^{10}C_1 * {}^{6}C_2 * {}^{8}C_2 * {}^{10}C_1 * {}^{4}C_1 = 168{,}000 $$

$$^{10}C_1 * {}^{6}C_2 * {}^{8}C_2 * {}^{10}C_1 * {}^{4}C_1 = 168{,}000$$

So what?

Agenda

Build systems and build system generators

Dependency management

Debugging

Testing

Agenda

- Build systems and build system generators
- Dependency management
- Demo
- Debugging
- Testing
- Demo

BUILD SYSTEMS AND BUILD SYSTEM GENERATORS

The solution space

Build systems vs. build system generators

Build system: a tool or set of tools used to compile and link source code

- GNU Make, NMake, XCode, MSBuild

Build systems vs. build system generators

Build system: a tool or set of tools used to compile and link source code

- GNU Make, NMake, XCode, MSBuild

Build system generator: a tool or set of tools used to generate project files for the specified build system(s)

- CMake, Meson, qmake, Premake

Build systems vs. build system generators

Build system: a tool or set of tools used to compile and link source code

- GNU Make, NMake, XCode, MSBuild

Build system generator: a tool or set of tools used to generate project files for the specified build system(s)

- CMake, Meson, qmake, Premake

CMake network effect

CMAKE

The problems

CMake: the problems

Learning the language of the build system

- Syntax, best practices

CMake: the problems

- Learning the language of the build system
  - Syntax, best practices
- Abdicating responsibility to CMake
  - Reimplementing functionality that CMake has built-in to abstract platform and compiler specifics

Use built-in CMake command line tools….

```
execute_process(COMMAND ${CMAKE_COMMAND} -E create_symlink
${filepath} ${sympath})
```

…instead of calling system commands directly

```
execute_process(COMMAND mklink ${filepath} ${sympath}) # Windows
execute_process(COMMAND ln -s ${filepath} ${sympath}) # Unix
```

Use built-in project commands whenever possible….

```
target_precompile_headers(myTarget PUBLIC my_project.h PRIVATE
<unordered_map>)
```

…instead of reimplementing functionality with custom functions and compiler specific logic

Set compile features (CMake 3.1 or later)….

```
target_compile_features(myTarget PRIVATE cxx_nullptr)
```

….or meta-features (CMake 3.8 or later)…

```
target_compile_features(myTarget PUBLIC cxx_std_11)
```

…instead of manually setting flags via CMAKE_CXX_FLAGS

Keep your paths platform independent….

```
target_include_directories(myTarget
    PUBLIC
        $<INSTALL_INTERFACE:include/myTarget>
        $<BUILD_INTERFACE:${CMAKE_CURRENT_SOURCE_DIR}/include/myTarget
    PRIVATE
        ${CMAKE_CURRENT_SOURCE_DIR}/src)
```

…and not platform specific

Use one toolchain file per target platform….

```
set(CMAKE_SYSTEM_NAME Linux)
set(CMAKE_SYSTEM_PROCESSOR arm)

set(CMAKE_C_COMPILER arm-linux-gnueabihf-gcc)
set(CMAKE_CXX_COMPILER arm-linux-gnueabihf-g++)
```

…instead of adding logic to your toolchain file

CMake: other resources

Daniel Pfeifer: Effective CMake

Henry Schreiner & other contributors: Modern CMake

DEPENDENCY MANAGEMENT

The problems

Dependency management: the problems

	Major paint point	Minor paint point	Not a significant issue for me	Total
Managing libraries my application depends on	46.54% 478	38.56% 396	14.90% 153	1,027

Source: ISO C++ 2020 Developer Survey

Dependency management: the problems

	Major paint point	Minor paint point	Not a significant issue for me	Total
Managing libraries my application depends on	46.54% 478	38.56% 396	14.90% 153	1,027

Source: ISO C++ 2020 Developer Survey

Automatic and reproducible dependency installation

Single source of truth

Consistency across platforms

DEPENDENCY MANAGEMENT

The solution space

Dependency management: the solution space

System package managers (e.g. apt)

Dependency management: the solution space

System package managers (e.g. apt)

Build system specific package managers (e.g. NuGet)

Dependency management: the solution space

System package managers (e.g. apt)

Build system specific package managers (e.g. NuGet)

Language specific package managers (e.g. vcpkg, Conan)

- Our recommendation for C++ cross-platform development

C++ package managers: Conan, vcpkg

Bring down and build libraries from source on Windows, Linux, and macOS

C++ package managers: Conan, vcpkg

Bring down and build libraries from source on Windows, Linux, and macOS

Transitive dependencies fetched and resolved automatically

C++ package managers: Conan, vcpkg

Bring down and build libraries from source on Windows, Linux, and macOS

Transitive dependencies fetched and resolved automatically

Acquire prebuilt binaries or cache binaries

C++ package managers: Conan, vcpkg

- Bring down and build libraries from source on Windows, Linux, and macOS
- Transitive dependencies fetched and resolved automatically
- Acquire prebuilt binaries or cache binaries
- Declarative manifest file that can be checked-in to source control

C++ package managers: Conan, vcpkg

- Bring down and build libraries from source on Windows, Linux, and macOS
- Transitive dependencies fetched and resolved automatically
- Acquire prebuilt binaries or cache binaries
- Declarative manifest file that can be checked-in to source control
- Install from multiple sources

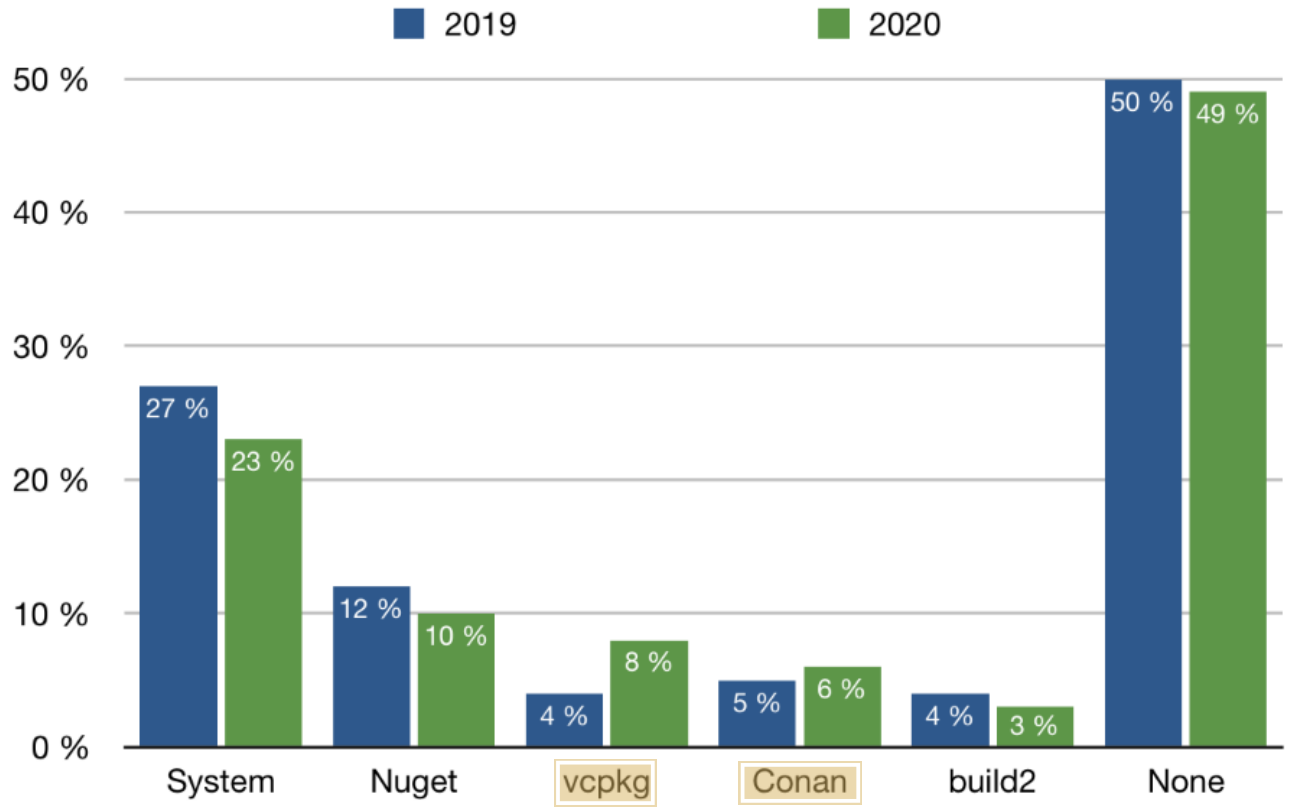
C++ package managers: Conan, vcpkg

- Bring down and build libraries from source on Windows, Linux, and macOS
- Transitive dependencies fetched and resolved automatically
- Acquire prebuilt binaries or cache binaries
- Declarative manifest file that can be checked-in to source control
- Install from multiple sources
- Versioning

C++ package managers: Conan, vcpkg

- Bring down and build libraries from source on Windows, Linux, and macOS
- Transitive dependencies fetched and resolved automatically
- Acquire prebuilt binaries or cache binaries NEW!
- Declarative manifest file that can be checked-in to source control NEW!
- Install from multiple sources COMING SOON
- Versioning COMING SOON

C++ package managers are on the rise



Source: JetBrains "State of the Developer Ecosystem 2020" – C++

DEMO #1

vcpkg & Visual Studio CMake support

Demo #1 recap

vcpkg manifest file

vcpkg binary caching hosted in GitHub packages

CMake language services (go to definition, peek definition, find all references on CMake variables, targets, and functions)

CMake project manipulation support (add, remove, and rename files and targets)

DEBUGGING

The problems

Debugging: the problems

- Unfamiliar with platform specific tools
  - A common theme: teams have a few "Linux devs" who are responsible for all debugging all Linux failures because they are comfortable with GDB

Debugging: the problems

- Unfamiliar with platform specific tools
  - A common theme: teams have a few "Linux devs" who are responsible for all debugging all Linux failures because they are comfortable with GDB
- Graphical interface vs. command driven interface
  - Breakpoints highlighted, watches and locals open simultaneously
  - Learning curve

Debugging: the problems

- Unfamiliar with platform specific tools
  - A common theme: teams have a few "Linux devs" who are responsible for all debugging all Linux failures because they are comfortable with GDB
- Graphical interface vs. command driven interface
  - Breakpoints highlighted, watches and locals open simultaneously
  - Learning curve
- Switching between platforms

DEBUGGING

The solution space

Debugging: the solution space

Cross-platform IDEs, editors, and tools

- CLion, VS Code, Qt Creator

Debugging: the solution space

- Cross-platform IDEs, editors, and tools
  - CLion, VS Code, Qt Creator
- Remote debugging: debugging a program running on a different system (and likely a different OS) than the one you are working on
  - VS Code remote extensions
  - Visual Studio remote support
  - CLion and Qt Creator

TESTING

The problems

TESTING

The problems

2020: The Year of Sanitizers? – Victor Ciura

Closing the Gap between Rust and C++ Using Principles of Static Analysis – Sunny Chatterjee

Testing: the problems

- Many C++ developers don't write unit tests
  - Unit tests can be especially helpful when used with a CI system to catch runtime errors across multiple platforms

TESTING

The solution space

Unit tests: the solution space

CTest: CMake test driver program

Manage and execute a complete suite of tests

Unit tests: the solution space

- CTest: CMake test driver program
  - Manage and execute a complete suite of tests
- Cross-platform unit testing frameworks
  - Google Test
  - Boost.Test
  - Catch2

Unit tests: the solution space

- CTest: CMake test driver program
  - Manage and execute a complete suite of tests
- Cross-platform unit testing frameworks
  - Google Test
  - Boost.Test
  - Catch2
- Use them together!

DEMO #2

Google Test, remote debugging in Visual Studio & GitHub actions

Demo #2 recap

- CTest and Google Test
- Remote debugging on Linux with Visual Studio
- GitHub Actions pipeline and GitHub packages
  - Acquire CMake and vcpkg, download cached binaries from GitHub packages, configure, build, and run tests on Windows and Linux

Debug Linux core dumps in Visual Studio

Debug Linux core dumps on a remote Linux system or WSL

May be helpful if you run a "Windows shop" but deploy to Linux servers and want to diagnose crashes in a familiar environment

Demo in Marian & Sy's talk: "A New Decade of Visual Studio"

New post on the C++ Team Blog

In summary...

Awareness

In summary...

In summary…

Awareness

Shared solutions for shared problems

In summary…

Awareness

Shared solutions for shared problems

Talk to us!

In summary…

Awareness

Shared solutions for shared problems

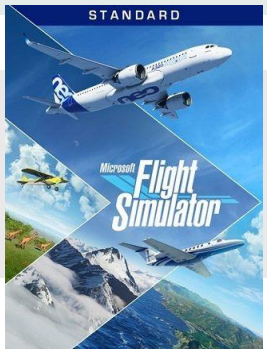
Talk to us!

Codebase: https://github.com/esweet431/box2d-lite

Enjoy the rest **of the conference!**

Visit our table at the Expo Hall
https://aka.ms/cppcon/expo

- Meet the Microsoft C++ team
- Ask any questions
- Discuss the latest announcements



Take our survey
https://aka.ms/cppcon

Chance to win 1 of 5 copies of Microsoft Flight Simulator

Our Sessions

Monday 14th

- A New Decade of Visual Studio: C++20, Open STL, and More – Sy Brand & Marian Luparu
- Collaborative C++ Development with Visual Studio Code – Julia Reid

Tuesday 15th

- Building an Intuition for Composition – Sy Brand
- Closing the gap between Rust and C++ using principles of static analysis – Sunny Chatterjee
- C++20 STL Features: 1 Year of Development on GitHub – Stephan T. Lavavej

Our Sessions

Wednesday 16th

- Dynamic Polymorphism with Metaclasses and Code Injection – Sy Brand
- Cross-Platform Pitfalls and How to Avoid Them – Erika Sweet
- Effective Remote C++ Development with Codespaces – Nick Uhlenhuth

Friday 18th

- Introducing Microsoft's New Open Source Fuzzing Platform – Justin Campbell & Michael Walker