What’s New in Visual Studio 2015 and Future Directions

Steve Carroll – Group Software Engineering Manager – Visual C++

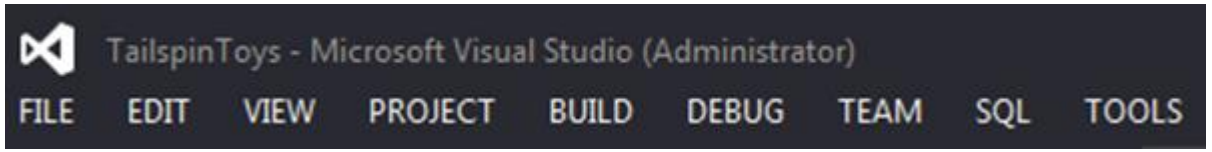
Ayman Shoukry – Group Program Manager – Visual C++

A note on our mission

- Visual Studio will be the best IDE out there for EVERY C++ developer.
  - Not just Windows developers
  - Especially for Open Source developers
- C++ is *the* language for cross platform development
  - Which is why C++ development is actually growing share amongst pro devs
- We can make C++ development more productive
- Finish catching up on conformance, participate in standards bodies to help make the language better. Stay caught up.

Saving the best for first

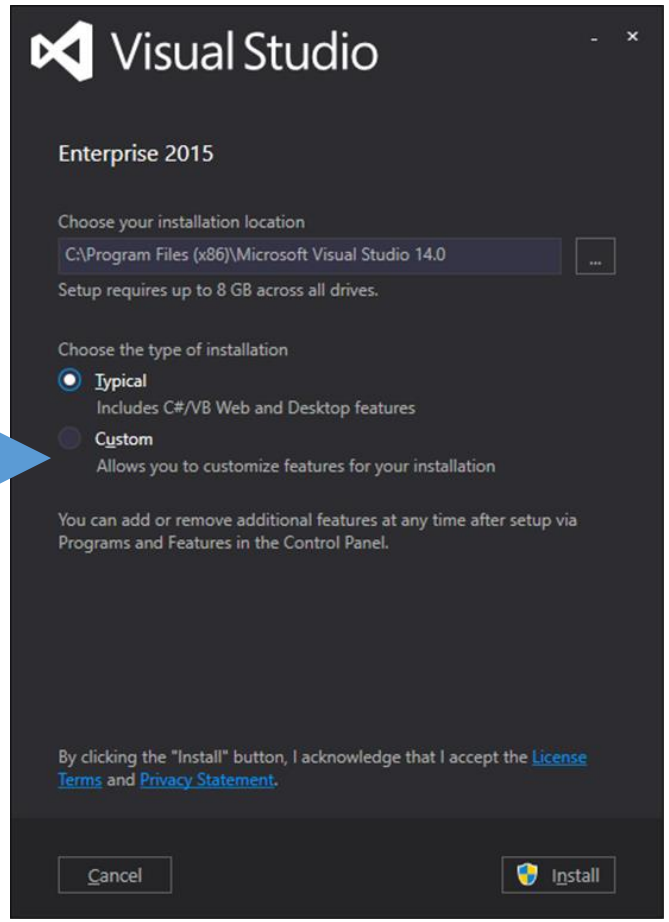
- Before (VS2013):

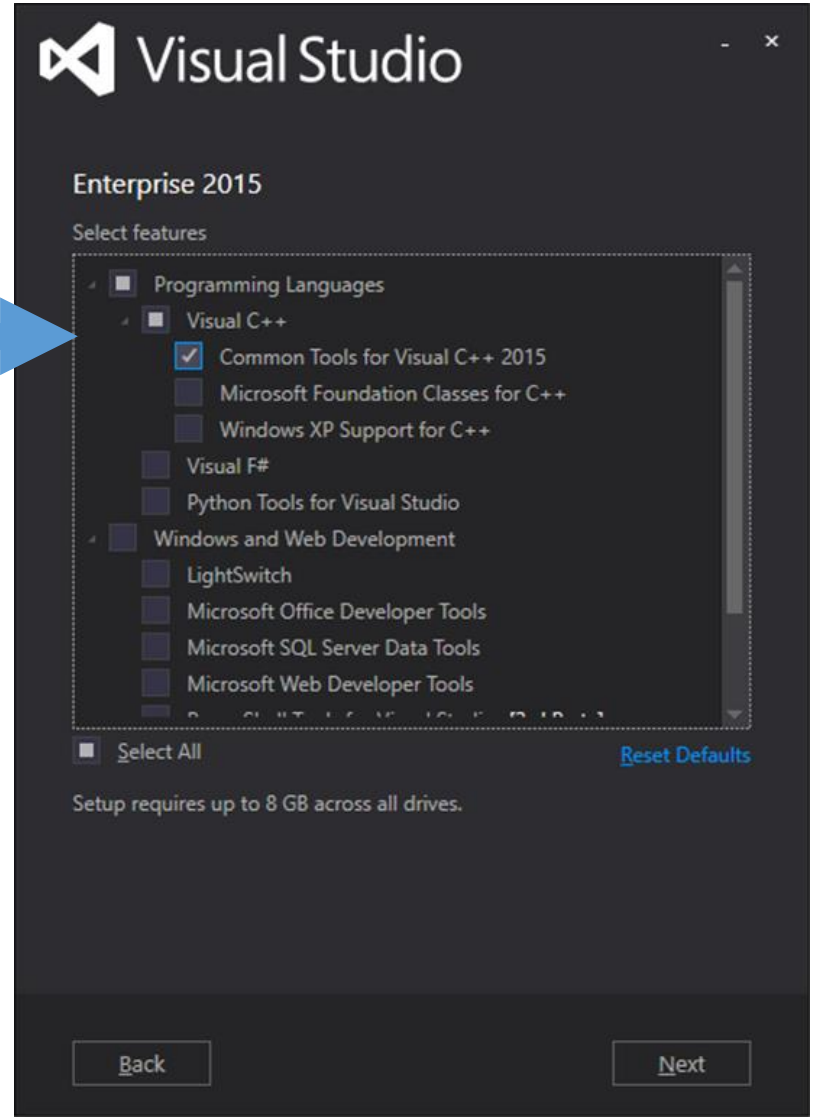


- After (VS2015):

Saving the worst for second

Pick ME!!!





Not new: Common Misconceptions

- You can still target XP in VS2015
- You can still do app-local deployment

Demo: Language Agnostic IDE Goodness

- Custom Window Layouts
- Touch screen enhancements
- Send-a-smile (or frown if you must)
- Find in file append
- Git and Github integration

Demo: C++ IDE Productivity

- Refactors (precision and accuracy)
  - Create Definition <-> Declaration (multiple)
  - Move function definition
  - Implement Pure Virtual
  - Rename
  - Extract Method
  - Convert to Raw String Literal
- Single File Intellisense
    - Quick editing of a makefile project or file->new file and int main, etc

Simplified Template IntelliSense

- Visual Studio 2013:

```
wstring myString(L"");
```

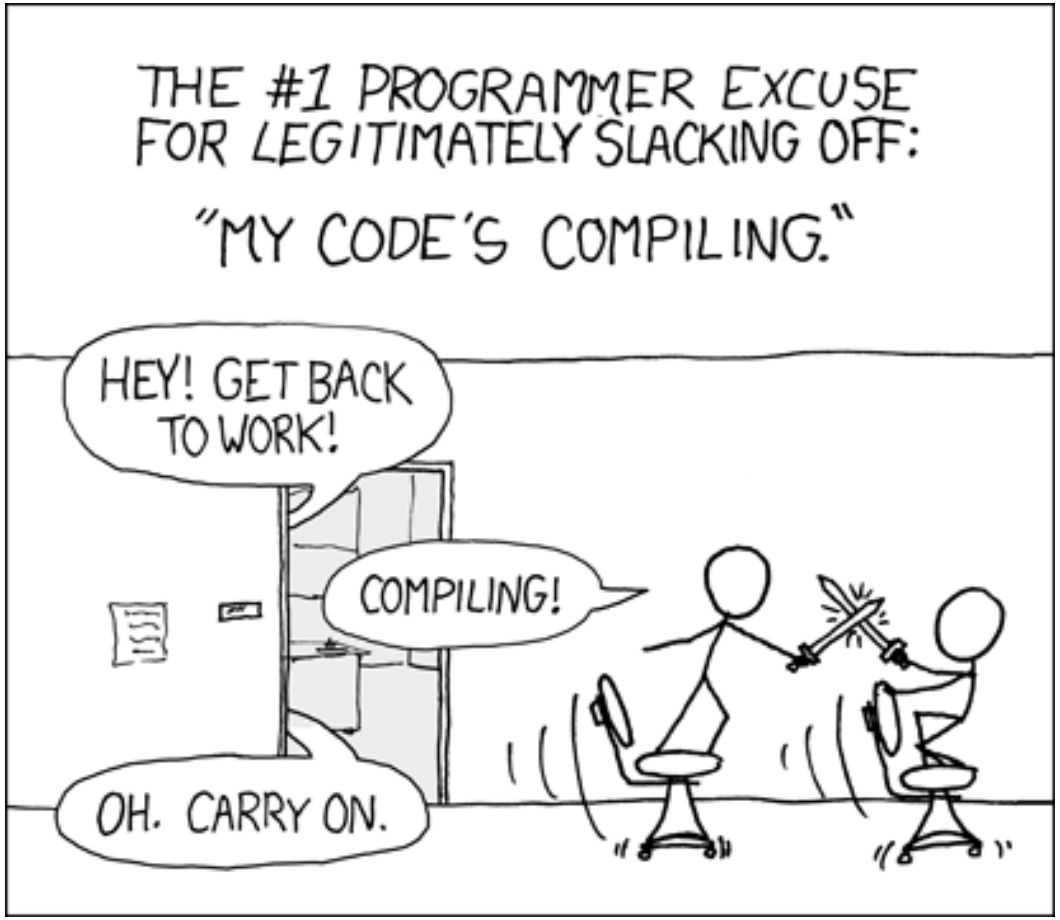
basic_string(**const std::basic_string<wchar_t, std::char_traits<wchar_t>, std::allocator<wchar_t>>::_Myt &_Right**,
std::basic_string<wchar_t, std::char_traits<wchar_t>, std::allocator<wchar_t>>::size_type _Roff,std::basic_string<wchar_t, std::char_traits<wchar_t>,
const std::_String_alloc<false, std::_String_base_types<wchar_t, std::allocator<wchar_t>>>::_Alloc &_Al)

- Visual Studio 2015:

```
wstring myString(L"");
```

wstring(**const std::wstring &_Right**, size_t _Roff, size_t _Count, const std::allocator<wchar_t> &_Al)

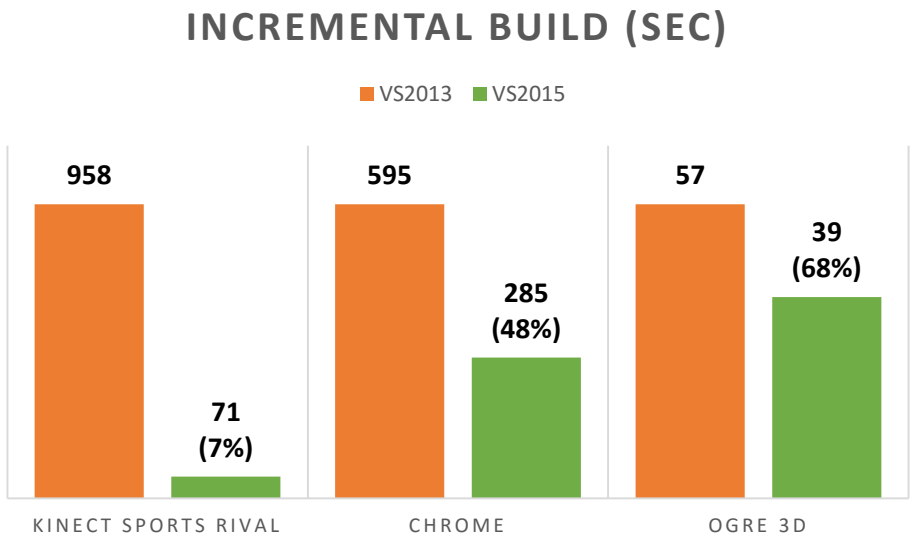
Productivity pt. 2: Build Throughput

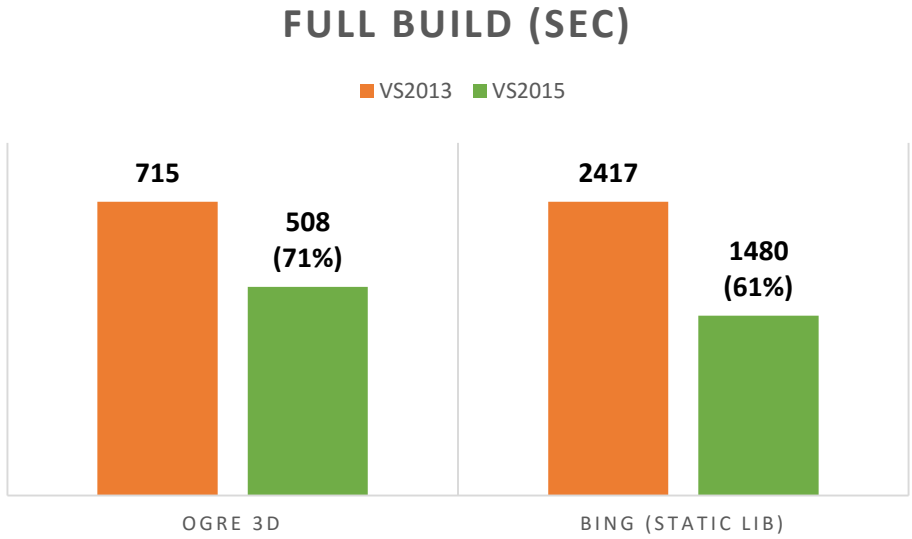


credit: This is my favorite xkcd.com strip of all time

Build Throughput Improvements

- Compiler
  - Faster template processing
- PDB
  - Reduced PDB generation time
  - /Debug:fastlink
- Linker
  - Incremental linking for static libraries
  - Algorithmic improvements
- LTCG & PGO
  - Fast LTCG
  - PGO instrumented builds has lower overhead
- General algorithmic improvements across tools





Productivity #3: Diagnostics Demos

- Exception settings
- Perf Tips
- Diagnostics Tools Window
- Memory tools
- NatViz improvements
- Completely new and improved Edit and Continue

The Silo Approach

Windows

C#, C++/CX

iOS

ObjC, Swift

Android

Java

Benefits

- Full native experience
- Total access to the device as provided by the SDK

Negatives

- Minimal code reuse
- Higher development cost
- One platform becomes the dominant platform

Trivia!

Android

Java

How many of the **top 100** applications on the **Android Playstore (U.S.)** leverage C++ code?

- None
- 15%
- 40%
- 75%

Trivia!

Android

Java

C++

How many of the **top 100** applications on the **Android Playstore (U.S.)** leverage C++ code?

- None
- 15%
- 40%
- **75%**

Top 100 Android Playstore applications (U.S.)

Android

Java

C++

- Messenger
- Facebook
- Pandora Radio
- Instagram
- Minecraft
- Snapchat
- Spotify Music
- Du Speed Booster
- Twitter
- The Game of Life
- Super Bright LED FlashLight
- Soda Saga
- Skype – Free
- Whatsapp Messenger
- Clean Master
- Netflix
- Kik
- Crossy Road
- Clash of Clans
- Amazon Shopping
- Candy Crush e IM and Video Calls
- 8 Ball Pool
- Glass Tower
- Subway Surfers
- Pinterest
- Cooking Fever
- Zedge Ringtones and Wallpaper
- Word Academy
- Poshmark - Buy and Sell
- Candy Crush Saga
- Dragon Blaze
- Marvel Future Fight
- Emoji Keyboard
- DU Battery saver
- SoundCloud - Music and Radio
- Monopoly
- Twitter
- CM Security Antivirus
- Slots - Journey of Magic
- Yahoo Mail - Free Email App
- iHeart Radio - Radio and Music
- Temple Run 2
- Boom Beach
- Despicable me
- ebay
- Wish - shopping made fun
- Trivia Check
- Juice Jam
- Game of War - Fire Age
- TouchPal Keyboard
- Geometry Dash Lite
- Flow Free
- Bird Climb
- Coin Dozer
- Uber
- Google Earth
- Flow Free
- Bird Climb
- Coin Dozer
- Uber
- Google Earth
- Archery Master 3d
- Go Keyboard - Emoji
- ooVoo video call
- Inbox by Gmail
- Samsung Smart Switch Mobile
- Tango - Free video call and chat
- Earn to Die 2
- Fruit Ninja Free
- Farm Heroes Saga
- Wallapop
- Capital One Wallet
- Truck Driver 3d: offroad
- Solitare
- Plants vs Zombies
- Hidden Object - Marrinotes
- Tinder
- DropBox
- Hulu
- Extreme Car driving simulator
- The Sims 3
- Word Search
- Hidden Object - Marrinotes
- Tinder
- Hulu
- Extreme Car driving simulator
- Need for Speed Most Wanted
- Angry Birds
- Shazam
- MyRadar Weather Radar
- Vine
- Line: Free calls and messages
- Waze Social GPS Maps
- Google Translate
- Don't tap the white tile
- Panda Pop
- EA Sports UFC
- Flipagram
- Hill Climb Racing
- Tasty Tale - The Cooking Game
- Yelp
- Offer Up - Buy, sell
- CM Launcher - Boost, Secure
- Temple Run
- Empire and Allies
- Google Docs
- Tetris
- Battery Doctor
- Beats Music
- Walmart
- Surgery Doctor
- EA FrostBite

C++ the common denominator

Windows

C#, CX

C++

iOS

ObjC, Swift

C++

Android

Java

C++

Benefits

- Full native experience
- Total access to the device as provided by the SDK
- Code Reuse
- Performance
- Security

C++ the common denominator


.appx


.apk


.ipa

XAML		XML	Cocoa Touch
C#	Cx	Java	
Pinvoke C++ Wrapper		Java/C++ JNI Wrappers	ObjC Wrapper

Shared C++ Backend

Shared C++ backend is compiled as:

.appx	.apk	.ipa
C#, C++/Cx	Java Dex / ART	ObjC Swift
Dynamic Link Library (.dll) Static Library (.lib)	Dynamic shared library (.so) Static library (.a)	Static library (.a)

DropBox

Cross-platform C++ demo

Android Development Features

- Supported versions
  - 4.4 “KitKat”, API Level 19
  - 5.0 “Lollipop”, API Level 21
- Emulator features include
  - OpenGL ES 2.0
  - Multi-touch gestures
  - Advanced camera simulation
  - WiFi network simulation

- Debug directly from Visual Studio
  - Integrated LogCat viewer
  - Natvis debugger visualizations
  - Externally built native activity apps
  - Can attach to running APKs
  - “Stripped” debugging to reduce deployment size
  - Deploy, Debug directly from VStudio. Emulator or devices.

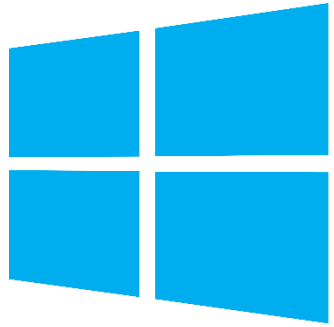
iOS Development Features

- Supports versions 6, 7, and 8
- Debug directly from Visual Studio
- Need a Mac connected deploy and debug on devices only (no emulator)

Universal Windows Platform Development

- **Develop one app for all Windows 10 devices**
- Apps automatically switch between desktop, phone, tablet, Xbox and Surface Hub configurations to fit screen size and input types
- Unified app store

- Design your UI in XAML with WYSIWYG editor
- You can use C++, C#, VB, or HTML/JavaScript
- Easily port Android and iOS apps



New C++11 / C++14 features in VS2015

- Generic lambdas (C++14)
- Init-captures (C++14)
- Auto and decltype(auto) return types (C++14)
- Inheriting constructors
- Magic statics
- C++11 constexpr (bugfixes in Update 1)
- Noexcept
- Sized-deallocation (C++14)
- Attributes
- Alignment
- User-defined literals
- char16_t and char32_t
- Unrestricted unions
- Ref-qualifiers
- Inline namespaces
- Universal character names in literals
- Unicode string literals
- Binary literals (C++14)
- Digit separators (C++14)
- Await (update 1) (C++17?)

What’s still missing?

- Expression SFINAE to complete C++11
  - Major refactor needed -> parse trees
  - Significant progress in Update 1
- Generalized Constexpr (C++14)
  - LARGE
- NSDMIs for aggregates (C++14)
  - small
- Variable templates (C++14)
  - Small
- Two-phase lookup
- C99 Preprocessor

STL conformance for VS2015

- Everything but:

Status	Std	Paper	Title
missing	C++14	N3462	SFINAE-Friendly result_of
missing	C++17	N4387	Improving pair And tuple
missing	C++17	N4508	shared_mutex (Untimed)

Library updates

- Universal CRT
  - From Windows 10 and up, CRT is an OS component
  - Still supported down to XP
  - No more breaking changes
- MFC updates:
  - Dynamic Layout / Auto-resize for dialogs
- New libraries
  - Parallel STL
- New version of Casablanca C++ REST SDK
  - OSS and takes contributions
  - WebSockets
  - oAuth 1.0 and 2.0
  - OSX/iOS support, Android, WP8.1, Windows 10 Universal

Introducing Clang / C2

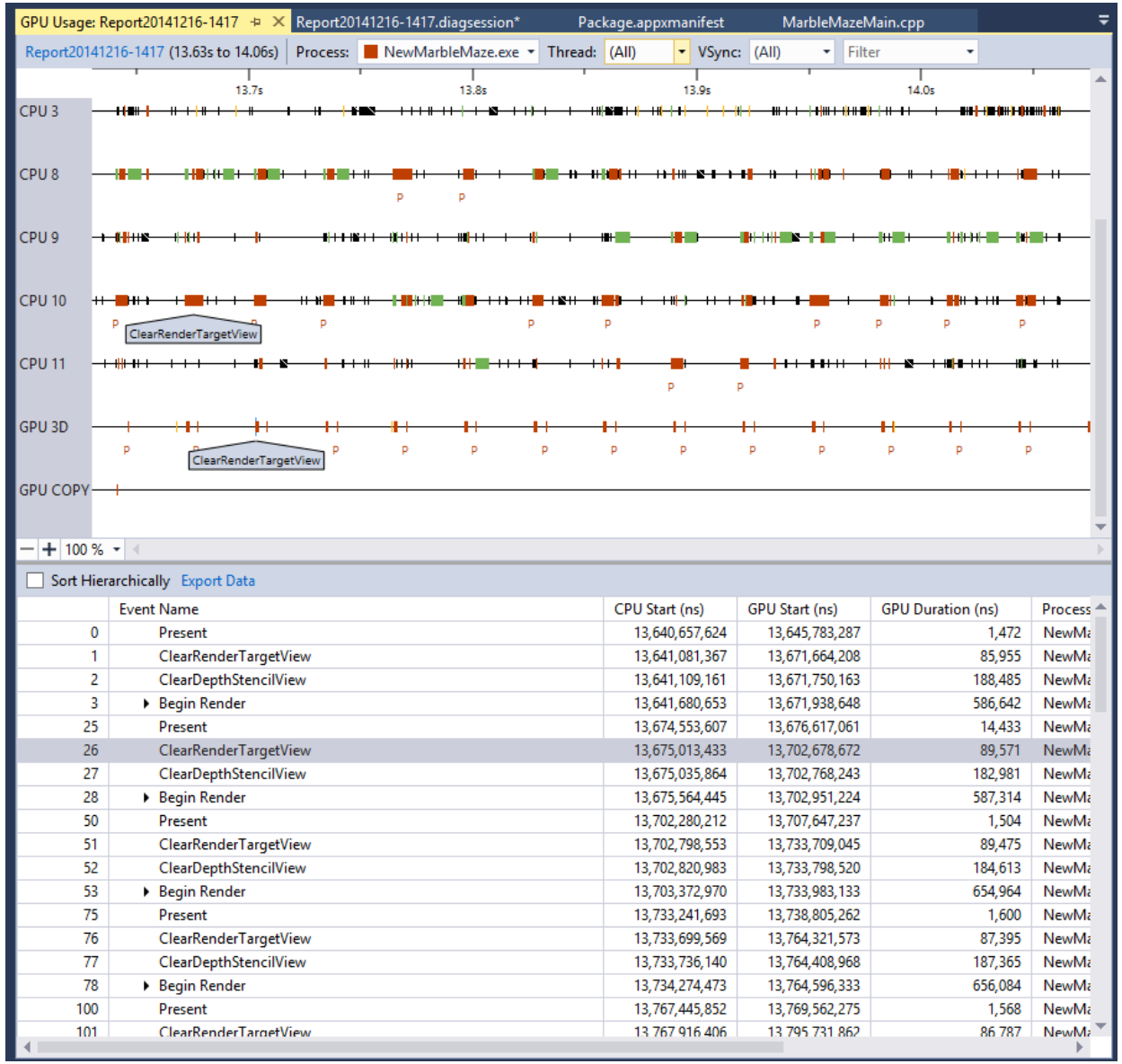
- Clang / C2 hooks up the Clang front-end to the MSVC backend
- Why?
  - use the same frontend to reduce overhead of bringing cross-plat libraries and code to Windows
  - no compromises debugging
  - link compatible with MSVC code (same libraries)
  - Preview coming in Update
- No, we aren't going to stop investing in our existing front-end (c1/c1xx)
  - we respect your investment in your existing code base

A quick note on C99 / C11

- We care about C.
- Our gap isn’t really that large in practice
  - Tgmath.h
  - Variable length arrays (optional in C11)
  - Complex types
  - Some other nits
- We’ll get this via Clang / C2

New for DirectX Devs

- DirectX12 support
- Shader Edit and Apply
- Consecutive Capture
- Programmatic Capture
- Dedicated UI for Graphics Analysis



	Event Name	CPU Start (ns)	GPU Start (ns)	GPU Duration (ns)	Process
0	Present	13,640,657,624	13,645,783,287	1,472	NewMa
1	ClearRenderTargetView	13,641,081,367	13,671,664,208	85,955	NewMa
2	ClearDepthStencilView	13,641,109,161	13,671,750,163	188,485	NewMa
3	▸ Begin Render	13,641,680,653	13,671,938,648	586,642	NewMa
25	Present	13,674,553,607	13,676,617,061	14,433	NewMa
26	ClearRenderTargetView	13,675,013,433	13,702,678,672	89,571	NewMa
27	ClearDepthStencilView	13,675,035,864	13,702,768,243	182,981	NewMa
28	▸ Begin Render	13,675,564,445	13,702,951,224	587,314	NewMa
50	Present	13,702,280,212	13,707,647,237	1,504	NewMa
51	ClearRenderTargetView	13,702,798,553	13,733,709,045	89,475	NewMa
52	ClearDepthStencilView	13,702,820,983	13,733,798,520	184,613	NewMa
53	▸ Begin Render	13,703,372,970	13,733,983,133	654,964	NewMa
75	Present	13,733,241,693	13,738,805,262	1,600	NewMa
76	ClearRenderTargetView	13,733,699,569	13,764,321,573	87,395	NewMa
77	ClearDepthStencilView	13,733,736,140	13,764,408,968	187,365	NewMa
78	▸ Begin Render	13,734,274,473	13,764,596,333	656,084	NewMa
100	Present	13,767,445,852	13,769,562,275	1,568	NewMa
101	ClearRenderTargetView	13,767,916,406	13,795,731,862	86,787	NewMa

Vectorization of control flow

- Vectorization of loops with if-then-else

```
void blackscholes(float* input, int *signArray, int n) {
    for (int i = 0; i < n; i++) {
        float InputX = input[i];
        int sign;
        if (InputX < 0.0f) {
            InputX = -InputX;
            sign = 1;
        } else {
            sign = 0;
        }
        input[i] = InputX;
        signArray[i] = sign;
    }
}
```

Optimized to branch-less code

```
mask   = InputX < 0.0f ? 0xFFFFFFFF : 0;
InputX = (mask & -InputX) | (~mask & InputX);
sign   = (mask & 1) | (~mask & 0);
```

300%+ speedup in blackscholes benchmark

Better stall-forward-avoidance

- Writes to large structures followed by reads from substructures can lead to hardware stalls.
- New optimizations targeted at std::complex<float> and std::complex<double>
- Improved code generation of VUNPCKLPD/VUNPCKLPS instructions

- **40% speedup in Eigen quaternion multiplication code**
- **35% speedup in Eigen FFT code**

Bit-test merging

- Optimizations targeting cascading bit tests:

Source code:

```
if ((((data->bitfield >> 3) & 1) != 0) ||
    (((data->bitfield >> 6) & 1) != 0) ||
    (((data->bitfield >> 9) & 1) != 0) ||
    (((data->bitfield >> 4) & 1) != 0) ||
    (((data->bitfield >> 8) & 1) != 0)) {
    ...
}
```

Optimized as if:

```
if ((data->bitfield & 0x1258) != 0) {
    ...
}
```

- Matters in code dealing with bitfields

14% speedup in quantum mechanics benchmark

Loop-if unswitching

- Improved code generation for loop-invariant control flow

Source code:

```
for (int i = 0; i < 100; i++)
    if (some_invariant_condition)
        ...
```

Optimized as if:

```
if (some_invariant_condition)
    for (int i = 0; i < 100; i++)
        ...
```

Hits often, especially when considering inlining

Other code generation improvements

- Better vectorization of STL constructs (range based for-loops, std::vector)
- Vectorization under /O1 (optimize for size)
- Better codegen of std::min/std::max (200% improvement in runtime)
- Better instruction emission for bit-test (more BT, BTC, BTR, BTS)
- Plus much more…

What’s next: Compiler features (conformance) in Updates

- No need to wait for major VS updates
- Features include:
  - Safer C++
  - Modules
  - Await completed in Update 1 (to current proposal)
  - Clang/C2
- No breaking changes by default

What’s next: IDE productivity and build-lab scenario

- More Refactorings
- VC ++ Scalability and Perf
- Build-time improvements
- Standalone Build tools

What’s next: Early thoughts!!

- Come talk to us about your Linux / IoT dev!
  - E.g. Remote Linux debugging

- Come talk to us about what you want from VS Code for C++!

Resources

- VCBlog – best way to follow us
  - http://blogs.msdn.com/b/vcblog/
- Channel 9 – Going Native
  - https://channel9.msdn.com/Shows/C9-GoingNative