Microsoft

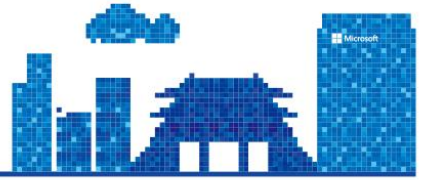
프로그래밍 언어의 F1머신 C++을 타고 Windows 10 UWP앱 개발의 세계로~

유영천 / 메이크어스 모바일

tech·days Korea 2015

2015.10.27(화) 09:00 AM -09:00 PM
세종대학교 컨벤션센터 B2

강연 목표



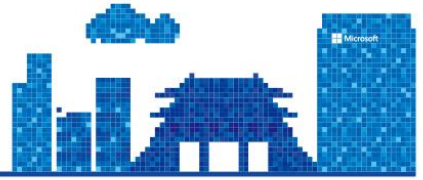
- UWP이 뭔지 알았다.
- UWP을 개발하고 싶다.
- UWP앱을 C++로 개발하고 싶다!

Hello UWP App!

Minecraft

- Windows UWP app
- C++

UWP(Universal Windows Platform)

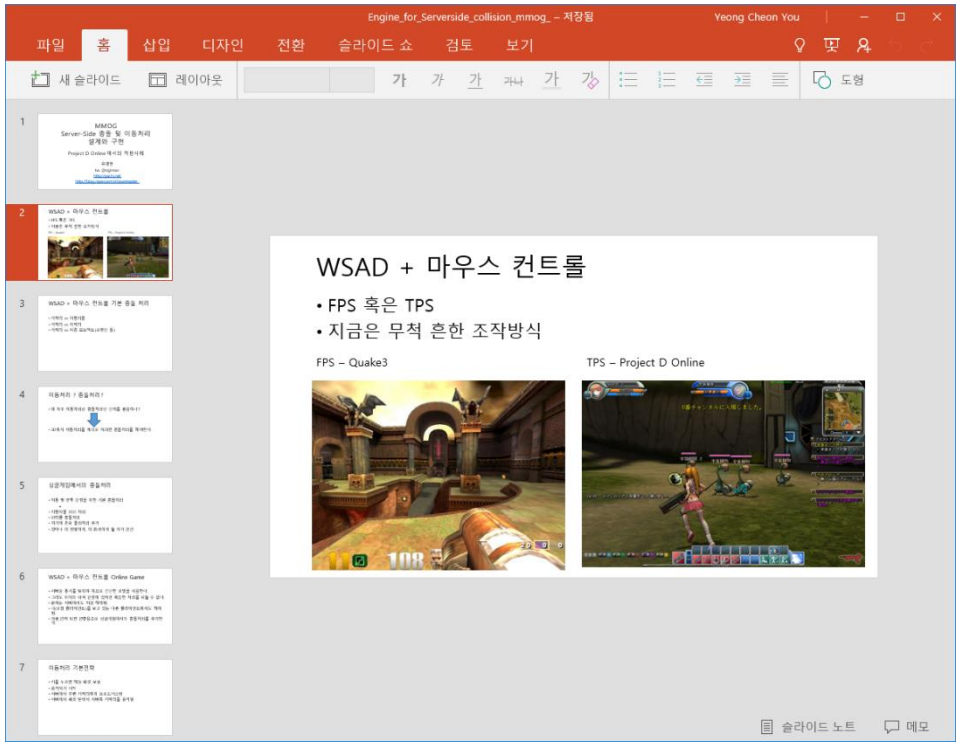


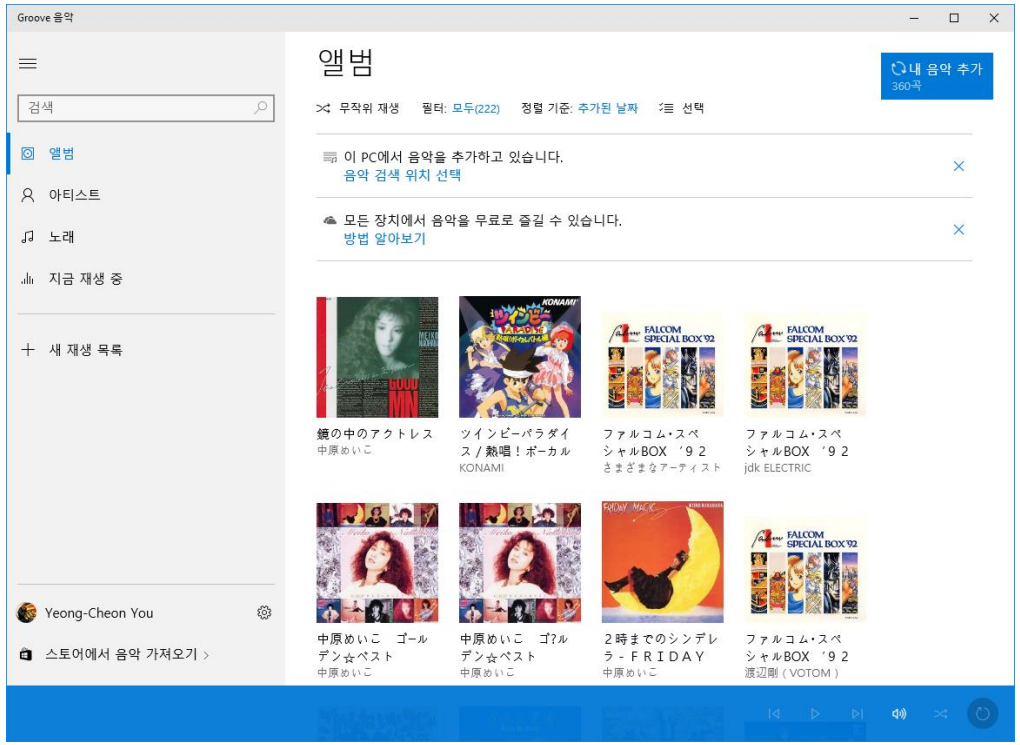
- Windows Platform의 새로운 API
- 모바일과 터치,웹 억세스에 특화
- 모든 Windows Device API의 대통합
- X86/x64/ARM 지원
- Windows 8/8.1 – ~~Metro App~~ / Windows Store App
- Windows 10 – UWP App

UWP(Universal Windows Platform)

One Universal Windows Platform

UWP Apps

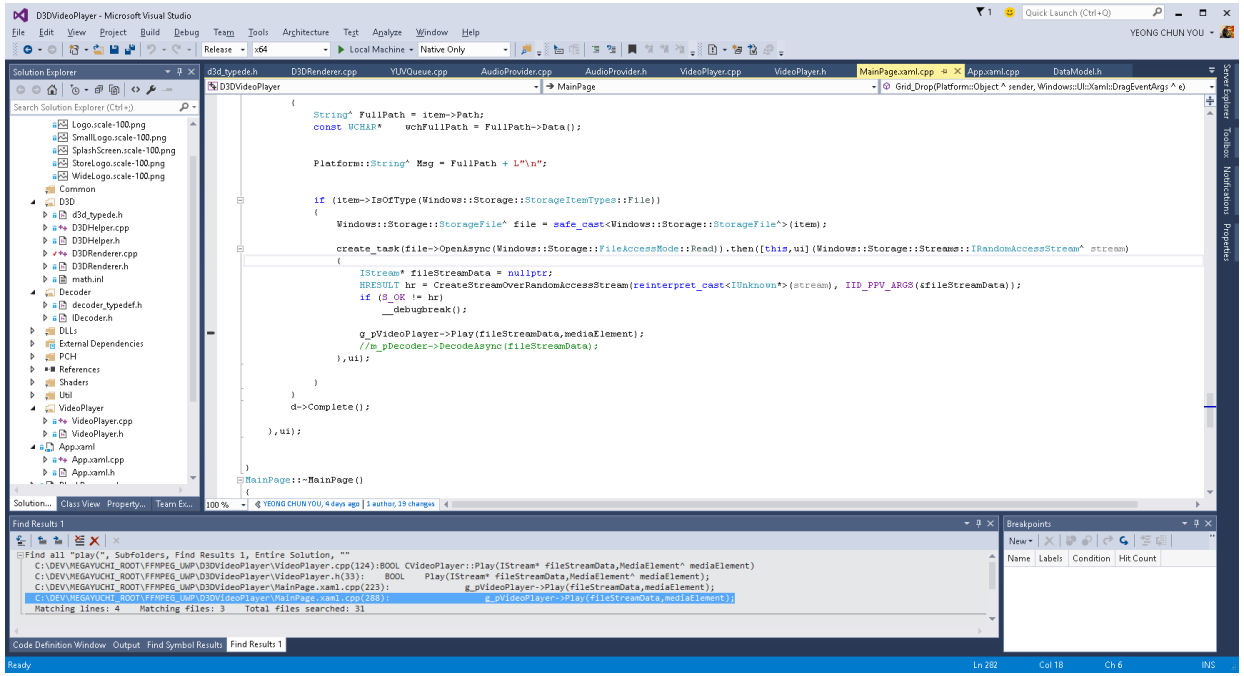


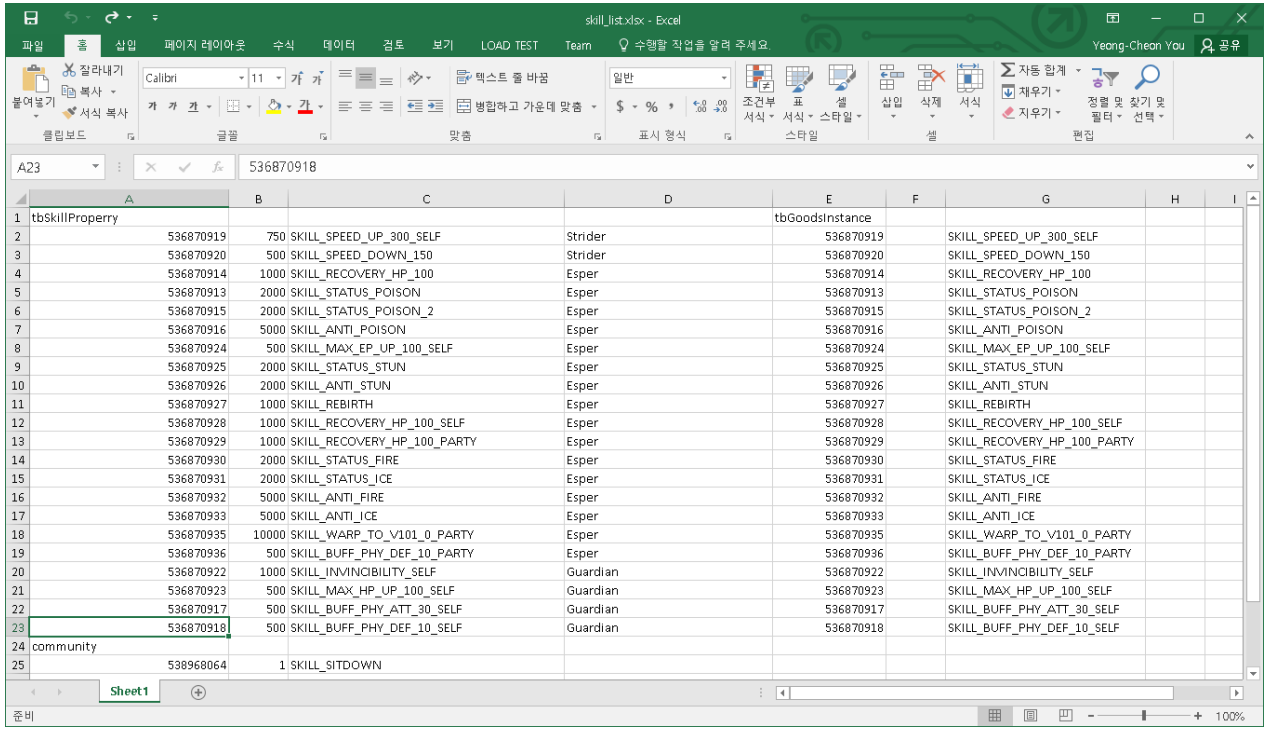


Microsoft

tech·days Korea 2015

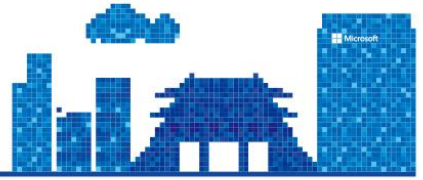
Desktop Applications





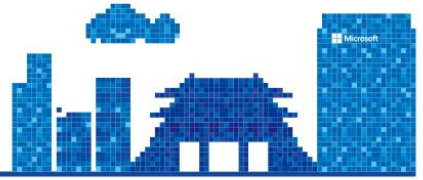
	A	B	C	D	E	F	G	H	I
1	tbSkillProperry				tbGoodsInstance				
2	536870919	750	SKILL_SPEED_UP_300_SELF	Strider	536870919		SKILL_SPEED_UP_300_SELF		
3	536870920	500	SKILL_SPEED_DOWN_150	Strider	536870920		SKILL_SPEED_DOWN_150		
4	536870914	1000	SKILL_RECOVERY_HP_100	Esper	536870914		SKILL_RECOVERY_HP_100		
5	536870913	2000	SKILL_STATUS_POISON	Esper	536870913		SKILL_STATUS_POISON		
6	536870915	2000	SKILL_STATUS_POISON_2	Esper	536870915		SKILL_STATUS_POISON_2		
7	536870916	5000	SKILL_ANTI_POISON	Esper	536870916		SKILL_ANTI_POISON		
8	536870924	500	SKILL_MAX_EP_UP_100_SELF	Esper	536870924		SKILL_MAX_EP_UP_100_SELF		
9	536870925	2000	SKILL_STATUS_STUN	Esper	536870925		SKILL_STATUS_STUN		
10	536870926	2000	SKILL_ANTI_STUN	Esper	536870926		SKILL_ANTI_STUN		
11	536870927	1000	SKILL_REBIRTH	Esper	536870927		SKILL_REBIRTH		
12	536870928	1000	SKILL_RECOVERY_HP_100_SELF	Esper	536870928		SKILL_RECOVERY_HP_100_SELF		
13	536870929	1000	SKILL_RECOVERY_HP_100_PARTY	Esper	536870929		SKILL_RECOVERY_HP_100_PARTY		
14	536870930	2000	SKILL_STATUS_FIRE	Esper	536870930		SKILL_STATUS_FIRE		
15	536870931	2000	SKILL_STATUS_ICE	Esper	536870931		SKILL_STATUS_ICE		
16	536870932	5000	SKILL_ANTI_FIRE	Esper	536870932		SKILL_ANTI_FIRE		
17	536870933	5000	SKILL_ANTI_ICE	Esper	536870933		SKILL_ANTI_ICE		
18	536870935	10000	SKILL_WARP_TO_V101_0_PARTY	Esper	536870935		SKILL_WARP_TO_V101_0_PARTY		
19	536870936	500	SKILL_BUFF_PHY_DEF_10_PARTY	Esper	536870936		SKILL_BUFF_PHY_DEF_10_PARTY		
20	536870922	1000	SKILL_INVINCIBILITY_SELF	Guardian	536870922		SKILL_INVINCIBILITY_SELF		
21	536870923	500	SKILL_MAX_HP_UP_100_SELF	Guardian	536870923		SKILL_MAX_HP_UP_100_SELF		
22	536870917	500	SKILL_BUFF_PHY_ATT_30_SELF	Guardian	536870917		SKILL_BUFF_PHY_ATT_30_SELF		
23	536870918	500	SKILL_BUFF_PHY_DEF_10_SELF	Guardian	536870918		SKILL_BUFF_PHY_DEF_10_SELF		
24	community								
25	538968064	1	SKILL_SITDOWN						

Why UWP App?



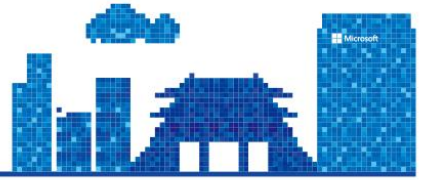
- 배포가 쉽다.
  - 별도의 디지털 서명이 필요없다.
  - Windows Store에 올릴 수 있다.
  - Sandbox형태로 깔끔한 설치/깔끔한 제거
- UI만들기 쉽다.
- Mobile 친화적이다.(전력관리/터치입력 등)
- Web과의 연동이 쉽다.
- 모든 Windows 10 device에서 다 돌아간다.

Why not UWP App? (불편한 진실)



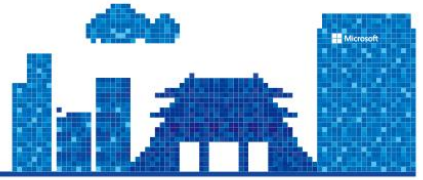
- 기존 코드를 얼마나 재사용할 수 있을까? (0% – 90%)
- 이놈의 App은 되는게 없네.
  - Sandbox 시스템 하에서의 권한 문제
  - API규모가 실제로 작음.
  - 아직 불안한 면도 있음.
  - 디바이스의 모든 기능을 사용할 수 없다.
- 사실 모든 Windows 10 device간 완벽 호환도 아니다.
  - Windows 10 Mobile / Windows 10간에 UWP는 거의 완벽 호환
  - Windows IoT와는 부분적으로 호환.
  - XBOX ONE 은 거의 100%일걸로 예상.
  - Hololense는 아직 모름.

Desktop App vs UWP App



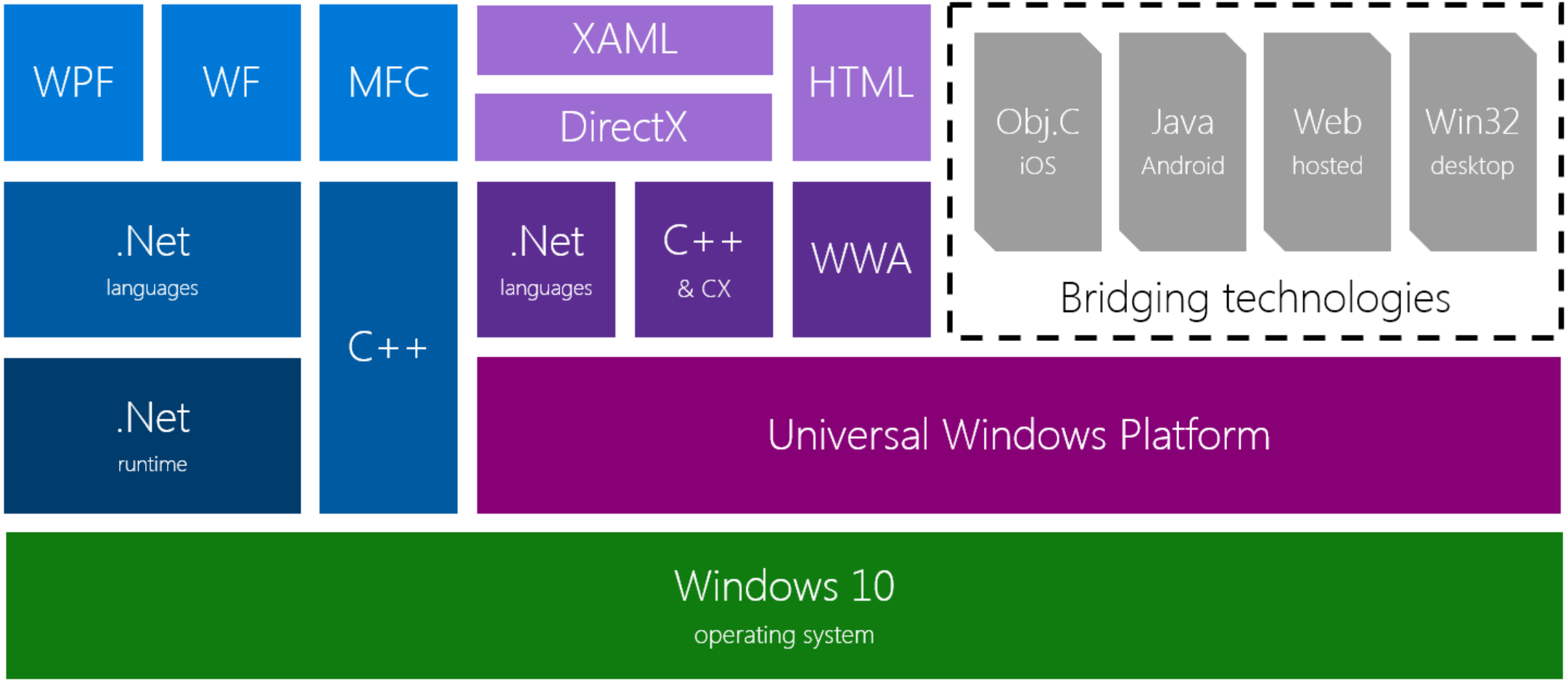
- 개발하기 쉽다
  - UWP App > Desktop App or Desktop App > UWP App ????
- 배포하기 쉽다(팔아먹기 쉽다)
  - UWP App >>> Desktop App
- 기능(할 수 있는 것)
  - Desktop App >>>> UWP App
- 타겟 디바이스 수
  - UWP App >> Desktop App

UWP의 기술적 특징

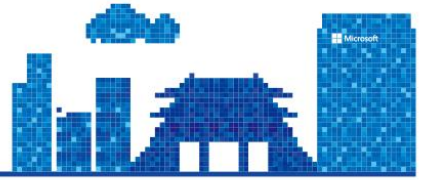


- Sandbox – (MS/Google/Apple중 MS가 가장 엄격함.)
- 앱의 명시적 종료가 따로 없음(iOS와 비슷)
- GDI사용할 수 없음. (WPF가 아니라면 UI는 새로 작성합니다.)
- UI는 XAML로 작성
- C++ / Java Script(HTML) , C#으로 개발가능
- C++로 개발할때
  - Win32일부 사용 가능.
  - Direct X 사용 가능.
  - 표준 C/C++ 라이브러리 어느 정도 사용 가능.

UWP의 기술적 특징


WPF
WF
MFC
XAML
DirectX
HTML
Obj.C
iOS
Java
Android
Web
hosted
Win32
desktop
Bridging technologies
.Net
languages
.Net
runtime
C++
.Net
languages
C++
& CX
WWA
Universal Windows Platform
Windows 10
operating system

Windows 10 App vs Windows 8.x App

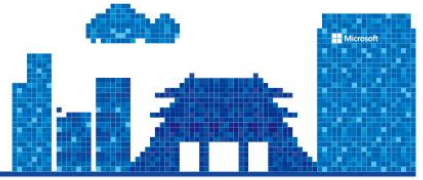


- 공식적으로는 둘다 Windows App으로 칭함.
- UWP :
  Windows Runtime API for Windows 10

- Windows 8.x App :
  Windows Runtime API for Windows 8/8.1

- Windows Runtime -> 줄여서 WinRT
- 저주받은 그 이름 Windows RT (Surface RT에 탑재된 Windows 8 on ARM)
- WinRT는 Windows RT를 연상시켜서 금기시되고 있다.
- 따라서 앞으론 UWP라고 부르기로 합니다.

UWP App과 Windows 8.x App의 차이

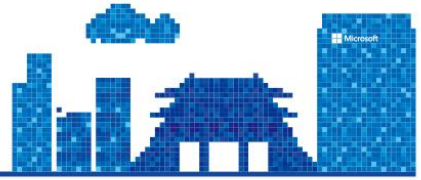


- 사실상 Windows 10 과 Windows 8.x에서의 ~~WinRT~~ API의 차이

- Universal App 호환성
  - Windows 8.1에선 Phone,PC 각각 XAML코드를 따로 작성해야됨.
  - Windows 10의 UWP에선 따로 작성할 필요없음.

- 다양한 사이즈의 디스플레이를 위한 Adaptive UI지원.

UWP App과 Windows 8.x App의 차이

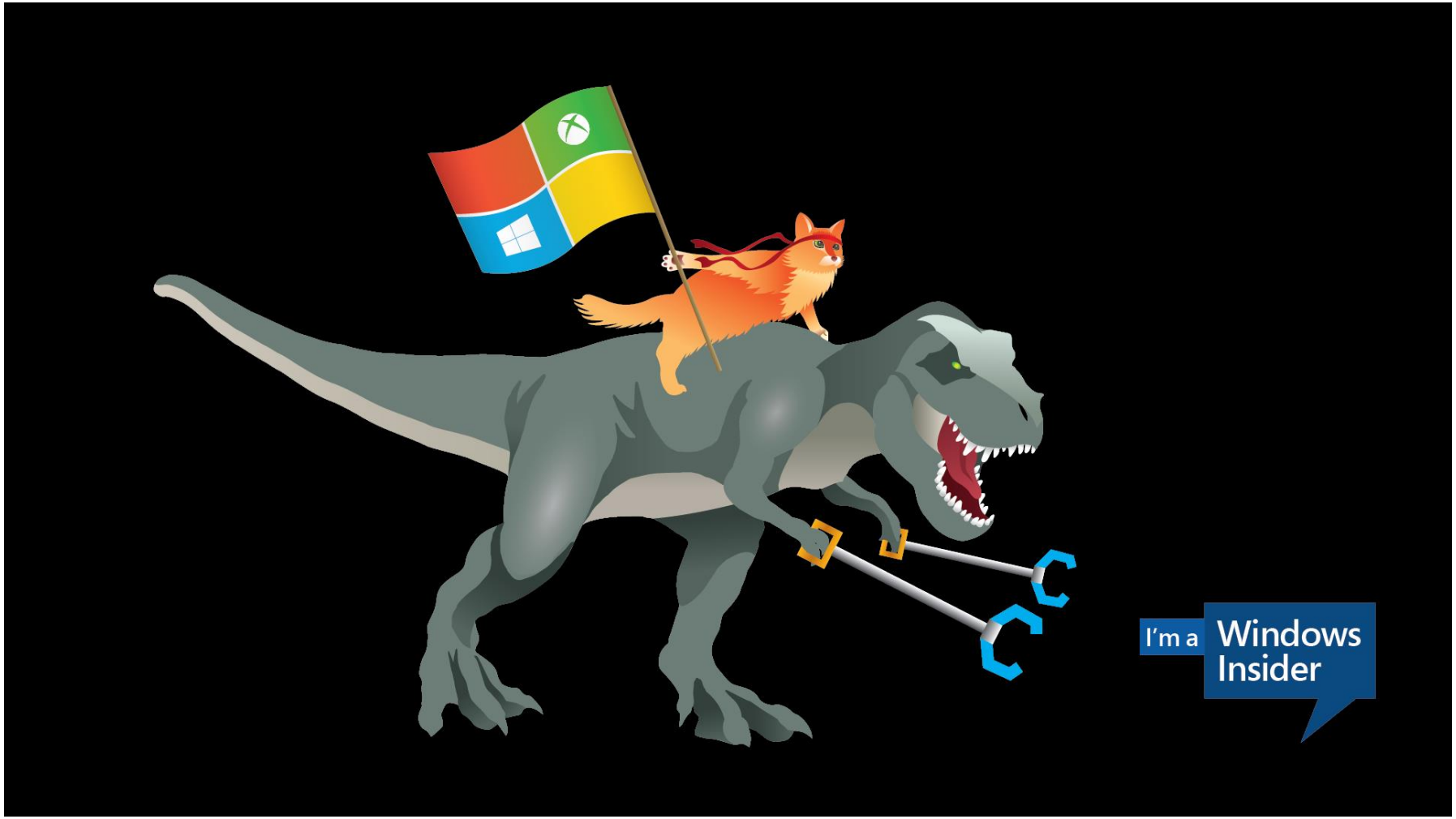


- 추가적인 Win32 API/CRT추가지원.(Winsock, process관련 CRT 등)
- 창모드 지원
- 드래그앤 드롭 지원.
- 여러개의 인스턴스 가능.(계산기,Edge브라우저 등)

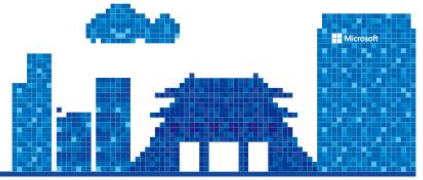
이제 Windows Store에서도 데스크탑에서 쓸만한 앱을 다운로드 할 수 있게 될 것이다.

megayuchi

UWP앱 개발

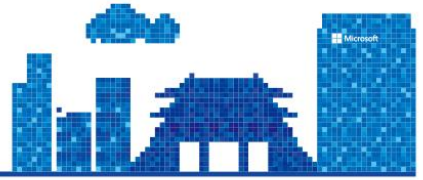


Why C++?



- 빠르다.(빠르게 짜거나 느리게 짤 수 있다.)
- 메모리 관리를 내 맘대로
  - 메모리 사용량을 줄일 수 있다.
  - GC기반 언어와 성능차이를 만들어내는 가장 큰 이유
- 기존 코드 재사용 가능 (게임/영상처리 등)
  - 영상 처리나 그래픽스 코드들의 상당수가 C++로 작성되어 있다.
  - 기존 PC/콘솔 게임들의 100%. 모바일 게임도 상당수가 C/C++로 작성되어 있다.
- 크로스플랫폼 지원 – 모든 플랫폼이 공통적으로 지원하는 언어는 C/C++ 뿐.

Cross-Platform 을 위한 C++사용 사례



Trivia!

Android

Java

C++

How many of the **top 50** applications on the **Android Playstore (U.S.)** leverage C++ code?

- None
- 10%
- 40%
- **80%**

Top 50 Android Playstore applications (U.S.)

Android

Java

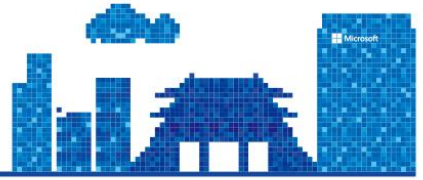
C++

- Facebook Messenger
- Facebook
- Pandora
- Clean Master
- Go Keyboard
- Instagram
- SnapChat
- Super-Bright LED
- Candy Crush Soda
- Spotify
- CrossRoads
- Netflix
- Subway Surfers
- Kik
- WhatsApp Messenger
- Skype
- Clash of Clans
- Jelly Jump
- DubsMash
- Temple Run 2
- Surgery Simulator
- Pinterest
- Candy Crush Saga
- CM Security Antivirus
- Trivia Crack
- Zedge Ringtones
- Apus Launcher
- Bingo Crush
- Amazon Shipping
- Texas Holdem
- ZigZag
- 8 ball pool
- Yahoo Mail
- Game of War
- Despicable Me
- Fast and Furious Legacy
- Five Nights at Freddy's
- Sound Cloud – Music and Audio
- iHeart Radio
- Twitter
- Fruit Ninja Free
- The Weather Channel
- Flow Free 2
- Minicraft
- Magic Piano
- ooVoo video call
- Solitare
- Wish Shopping Made Fun
- Google Earth
- Angry Birds

Building Cross-Platform Mobile Apps in C++ with Visual Studio 2015

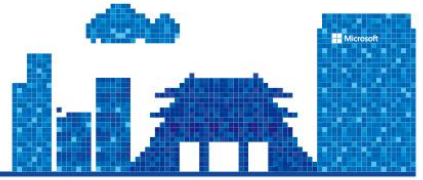
https://channel9.msdn.com/Events/Build/2015/3-714

C++로 UWP앱 개발



- C++만을 사용하면 좋겠지만 C++/CX를 조금은 사용해야함.
- API는 호출하는 방법만 약간 다르고 C#,C++,JS 기본적으로 동일.
- 예제가 없어요ㅠㅠ -> C#을 키워드에 넣어서 검색합니다.
- UI는 XAML을 사용한다. (C#,JS와 동일)
- C/C++ Runtime Library 사용 -> Universal CRT(ucrtbase.dll)
  - 8/8.1 세대의 CRT지원에 비해 제약이 상당히 줄어들었다.

C/C++로 개발할때의 중요한 변화

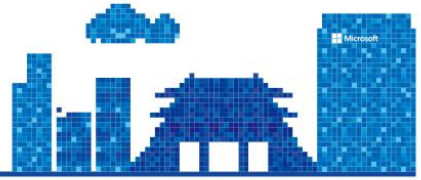


- _beginthread(), _endthread()등 process.h에 선언된 스레드관련 함수들 사용 가능

- SetCurrentDirectory(),GetCurrentDirectory() 사용 가능. 이전엔 모든 path는 절대 경로로 접근해야 했음.

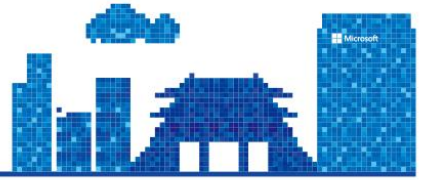
- ARM/x86에 상관없이 HLSL Shader를 run-time에 컴파일 가능. 8.x에선 미리 빌드한 바이너리만 사용 가능했다.

Hello World~



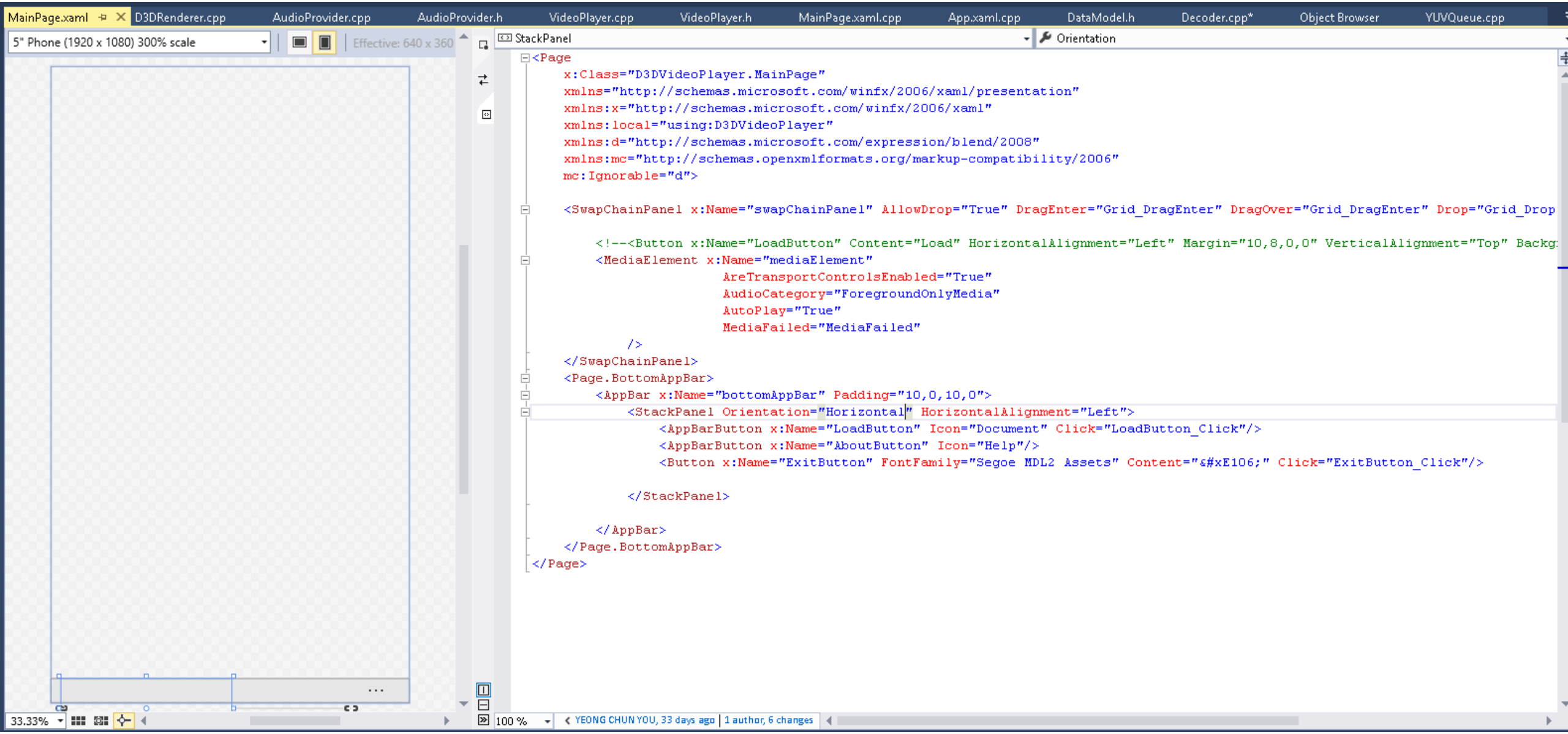
- 간단 라이브 코딩

XAML UI

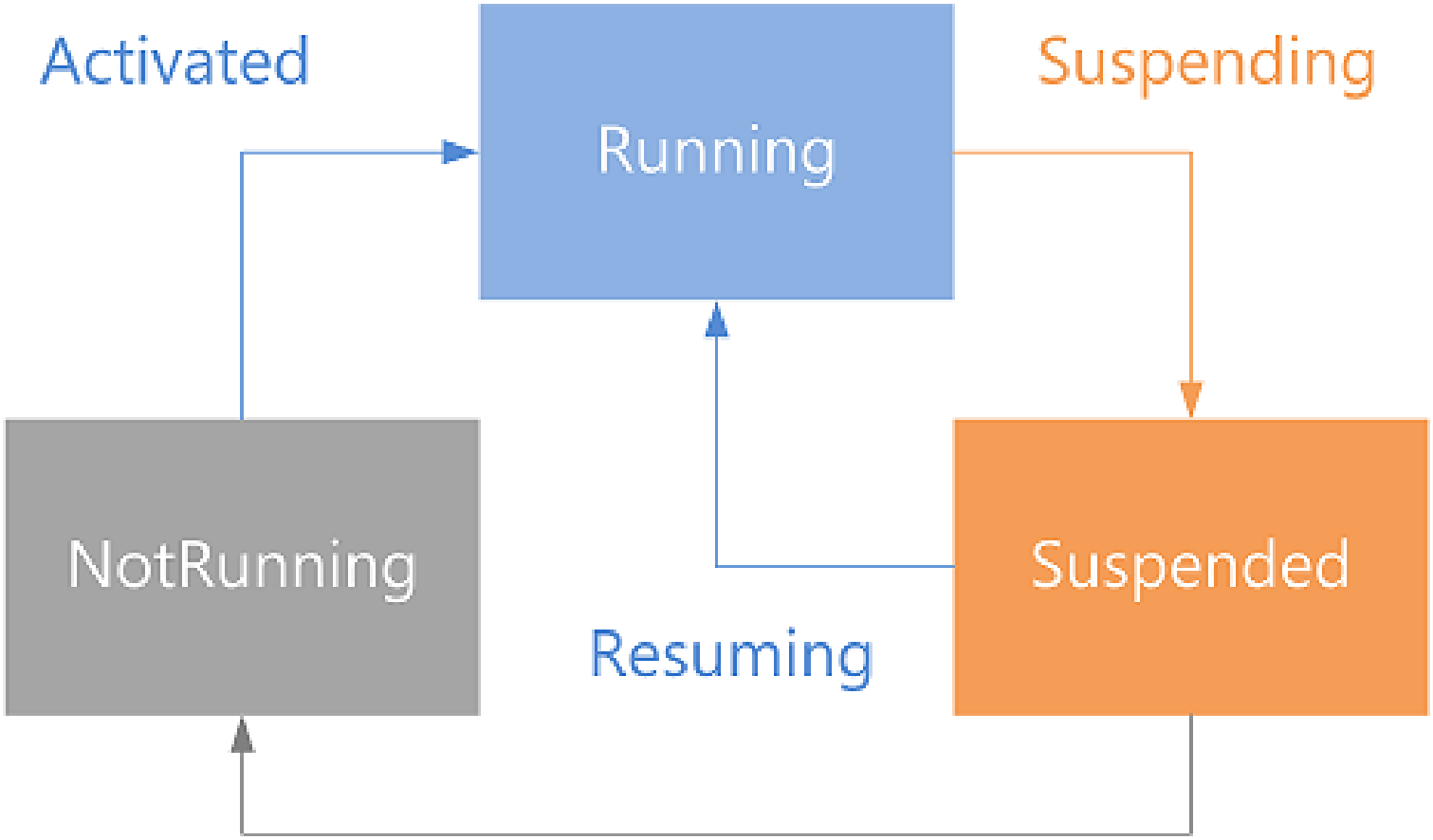


- XAML - MS의 UI 개발용 마크업 언어
- 어떤 언어를 사용하든 UWP앱 개발을 위해서 XAML은 무조건 사용해야함.
- Visual Studio에 기본 내장된 XAML Designer로 작성 가능.
- Visual Studio 패키지에 포함된 Blend로 작성 가능.

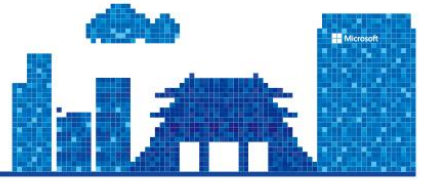
XAML Designer



App Lifecycle

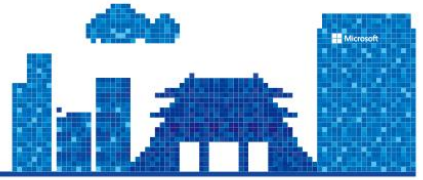


App Lifecycle – 이것만 알아둡시다.



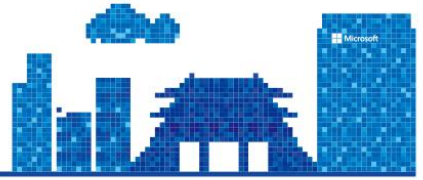
- void App::OnLaunched()
  - App이 실행됨. 최초 실행되었거나, 죽었다가 실행되었거나…
    - Suspend-> Killed -> Resume -> OnLanunched
    - 최초 설치됨 -> OnLaunched
- Void App::OnSuspending()
  - 더 이상 스케쥴링되지 않음.
    - Phone이나 태블릿모드에서 back버튼, Windows버튼 눌렀을때.
    - 전화왔을때.
    - Desktop에서 창을 최소화시켰을때.
- void App::OnResuming()
  - Susened상태에서 다행히(?) 죽지 않고 App으로 복귀한 경우.
    - OS는 메모리가 부족할 때 suspend 상태의 앱부터 죽인다.

C++/CX



- UWP API(Windows Runtime API)를 호출하기 위한 MS의 C++ 확장
- UWP의 모든 API는 객체지향. C++/CX의 ref class로 구현되어있다.
- 레퍼런스 카운팅 기반 C++
- 컴파일러가 분석해서 알아서 AddRef()와 Release()를 호출하는 게 아니고...ref class가 스마트 포인터를 내장하고 있다.

C++/CX



- C++에 C++/CLI 문법을 차용했다. -> C# 비슷한 느낌이 있다.
- ref class는 내부적으로 IInspectable COM 객체
- ref class의 핸들로 포인터 연산자 *대신 ^을 사용한다.
- Lambda 표현을 많이 사용함.
- ppl task를 많이 사용한다.

ref class 선언

```
ref class CSimpleObject sealed
{
    String^_Name;
    ~CSimpleObject();
public: // 외부에 메타데이타를 노출함. 심지어 다른 언어에서도 이 클래스의 public 멤버 호출 가능
    property Platform::String^ Name // get(),set() 직접 코딩
    {
        Platform::String^ get()
        {
            return _Name;
        }
        void set(Platform::String^ name)
        {
            _Name = name;
        }
    }
    property int Value; // get(),set()자동 생성
    CSimpleObject();
    CSimpleObject(String^ name,int value)
    {
        _Name = name;
        Value = value;
    }
internal: // 이 모듈(빌드되는 바이너리) 내에서만 public.
    CSimpleObject^ operator+(CSimpleObject^ obj);
};
```

ref class 구현

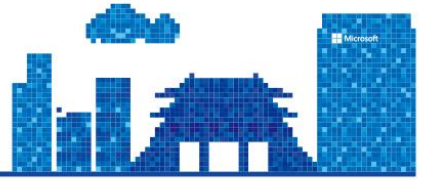
```
#include "pch.h"
#include "SimpleObject.h"

CSimpleObject::CSimpleObject()
{

}

CSimpleObject^ CSimpleObject::operator+(CSimpleObject^ obj)
{
    CSimpleObject^result = ref new CSimpleObject();
    result->Value = Value + obj->Value;
    result->Name = _Name + L" + " + obj->Name;

    return result;

}

CSimpleObject::~CSimpleObject()
{
    String^Message = _Name + L" destroyed\n";
    OutputDebugString(Message->Data());
}
```

ref class 사용

```
void MainPage::TestRefClass()
{
    CSimpleObject^ Obj0 = ref new CSimpleObject(L"Object 0",0);
    CSimpleObject^ Obj1 = ref new CSimpleObject(L"Object 1",1);

    CSimpleObject^ Obj2 = Obj0 + Obj1; // 새로운 객체 Obj2 생성

    CSimpleObject^ Obj3 = Obj2; // Obj3이 Obj2를 참조. ref count 1 증가


    Obj0 = nullptr; // ref count가 0이 되어 해제
    Obj1 = nullptr; // ref count가 0이 되어 해제
    Obj2 = nullptr; // ref count가 1 남아있으므로 해제되지 않음.
    Obj3 = nullptr; // ref count가 0이 되어 해제
}
```

```
<Output>
Object 0 destroyed
Object 1 destroyed
Object 0 + Object 1 destroyed
```

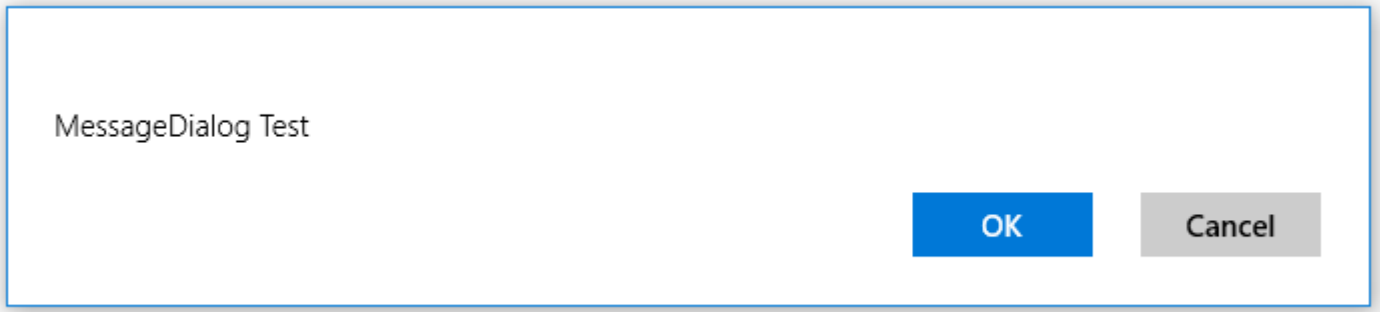
C++/CX - Lambda



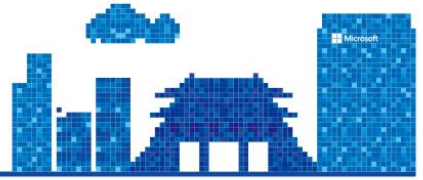
- 비동기 API호출을 위해 create_task(), then()을 많이 쓰다보니 Lambda식을 자연스럽게 많이 사용하게 됨.
- 이벤트 핸들러(Win32에서 콜백함수)의 상당수를 Lambda식으로 전달
- 그냥 쓰다 보면 익숙해짐. 나름 편함.
- Lambda에 대한 자세한 설명은 생략. 링크를 참조
  https://msdn.microsoft.com/ko-kr/library/dd293608.aspx

Lambda식 사용

```
void MainPage::MessageDialogOkCancel(Platform::String^ Message)
{
    MessageDialog^ msg = ref new MessageDialog(Message);

    // OK버튼을 눌렀을때의 이벤트 핸들러를 Lambda식으로 전달
    UICommand^ cmdOK = ref new UICommand("OK",ref new UICommandInvokedHandler([this](IUICommand^
    )
    {
        // On Pressed OK

    }));
    // Cancel버튼을 눌렀을때의 이벤트 핸들러를 Lambda식으로 전달
    UICommand^ cmdCanccel = ref new UICommand("Cancel",ref new UICommandInvokedHandler([this](IU
    ICommand^)
    {
        // On Pressed Cancel
    }));

    msg->Commands->Append(cmdOK);
    msg->Commands->Append(cmdCanccel);
    msg->DefaultCommandIndex = 0;
    msg->CancelCommandIndex = 1;
    msg->ShowAsync();
}
```



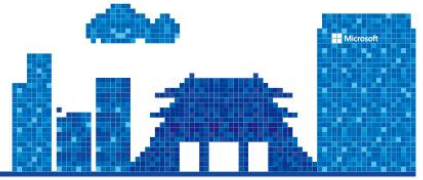
Async API



- Windows Runtime의 설계자들은 생각했다.
  - UI가 잠깐이라도 먹통 되는 것은 참을 수 없다!!!!
  - I/O작업은 미친듯이 느리다.

- 그래서 그들은 이렇게 결정했다.
  - 어떤 작업도 UI스레드를 block시켜서는 안된다.
  - 모든 I/O작업은 비동기로 처리한다.
  - 가능하면 UI 스레드는 다른 스레드에 작업을 넘길 뿐 직접 처리하지 않는다.

Async API



- I/O 관련 API는 모조리 Async
- Async API는 ppl task의 create_task()와 같이 사용한다.
- create_task().then().then().then()….
- create_task()는 Windows ThreadPool 사용하므로 task 생성 비용이 높지는 않다.
- 추후 C++에서도 await가 지원될 예정.
  - 사실 이 문서를 작성하는 시점에서 전혀 불편함을 못 느끼고 있음.

Async call – File Open

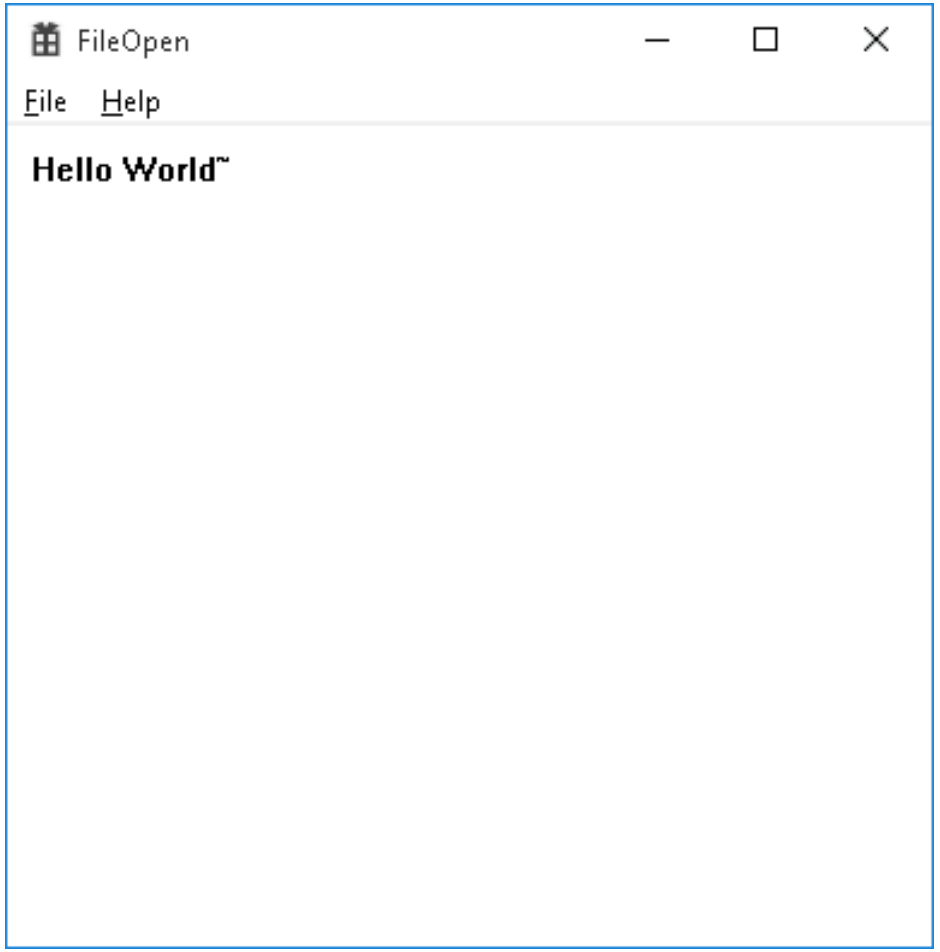
```
void FileOpenUWP::MainPage::PickButton_ClickSingle(Platform::Object^ sender, Windows::UI::Xaml:
:RoutedEventArgs^ e)
{
    FileOpenPicker^ openPicker = ref new FileOpenPicker();
    openPicker->ViewMode = PickerViewMode::Thumbnail;
    openPicker->SuggestedStartLocation = PickerLocationId::PicturesLibrary;
    openPicker->FileTypeFilter->Append(".txt");

    //auto ctx = task_continuation_context::use_arbitrary();
    auto ctx = task_continuation_context::use_default();
    create_task(openPicker->PickSingleFileAsync()).then([this](Windows::Storage::StorageFile^ fi
    le)
    {
        if (file)
        {
            ReadTextFromFile(file);
        }
    },ctx);
}
```

then().then().then()이 너무 정신 없다!!!

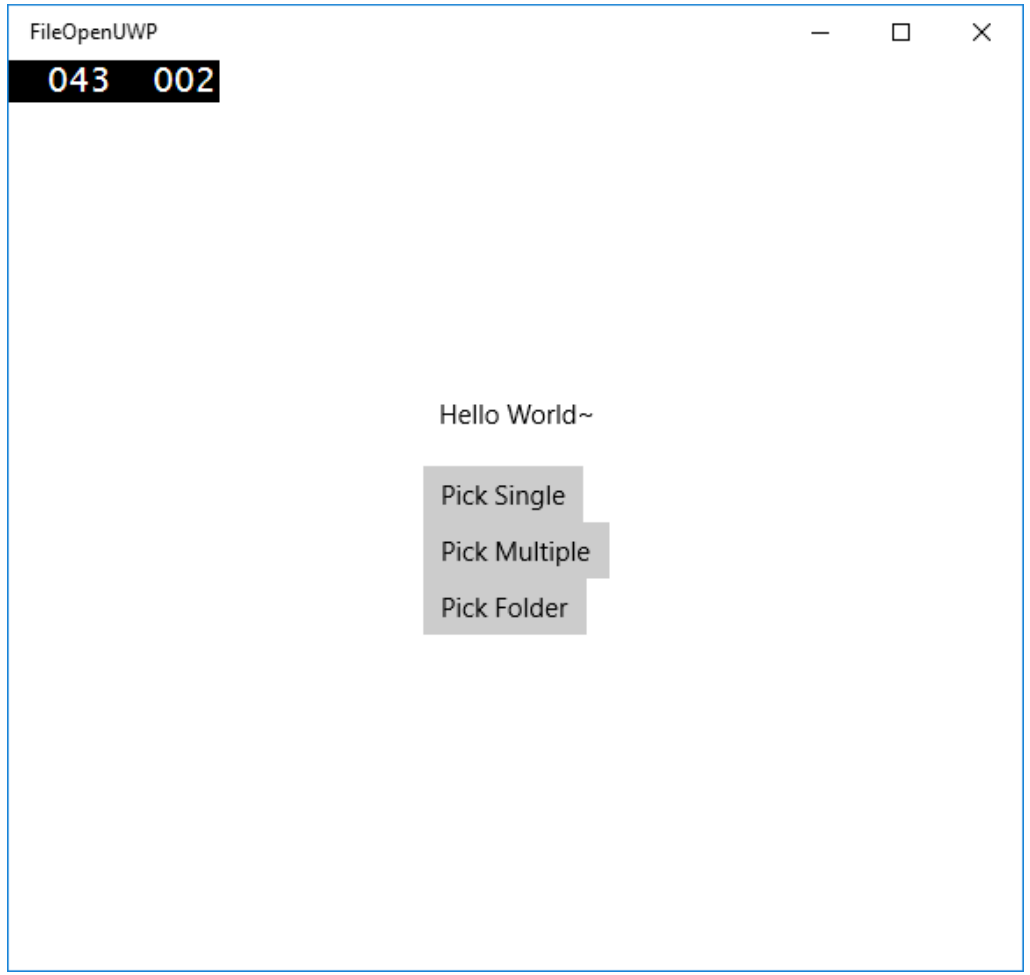
UI스레드가 아닌 스레이라면 task::get()을 사용해서 Asyc 태스크의 완료를 wait 할 수 있다.
Then()문이 필요 없다.

```
Windows::Storage::CreationCollisionOption opt = CreationCollisionOption::ReplaceExisting;

auto task_file = create_task(folder->CreateFileAsync(FileName,opt));
StorageFile^ file = nullptr;
try
{
    file = task_file.get();
}
catch (Platform::Exception^ e)
{
    String^ err = e->Message;
    String^ errMsg = L"Failed to CreateFileAsyc - " + err + L"\n";
    WriteDebugStringW(errMsg->Data());
}
```

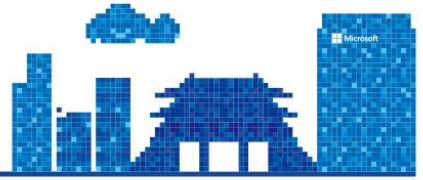
파일 오픈 예제(Desktop Application)



파일오픈 예제 (UWP)


FileOpenUWP
043 002
Hello World~
Pick Single
Pick Multiple
Pick Folder

감을 잡으셨나요?

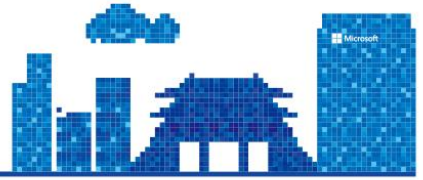


- 추가적으로 학습해야 할 내용
  - XAML
  - C++/CX
  - Async API + ppl task

그때그때 검색해서 사용하다 보면 금방 익숙해집니다.

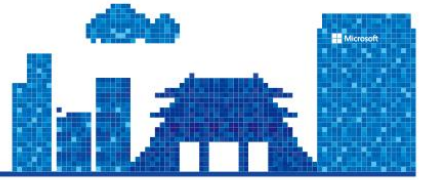
만들어 봅시다.

UWP앱, 무엇을 만들까?



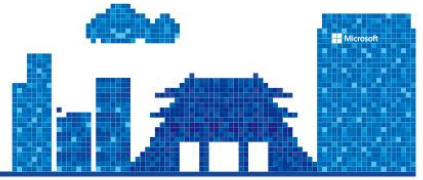
- smi자막과 다양한 코덱을 지원하는 동영상 플레이어
- 강제 resize없고 exif 표시되는 이미지 뷰어
- Winamp같은 뮤직 플레이어
- 그림판보단 좀 그럴싸한 이미지 에디터-아이콘 에디터라든가
- Mp3 tag
- 가벼운 3D 모델링 툴. Sket**up같은거?
- 네트워크 툴 – ftp등
- 게임

UWP앱, 무엇을 만들까?



- 소비성이 아닌 생산성 앱이면서 무겁지 않은 앱이 적합하다.
- PC/콘솔 게임과 동등한 스케일, 혹은 그에 준하는 스케일의 게임
- 이런 앱이 모바일에서도 똑같이 작동한다면 매력적이지 않은가?

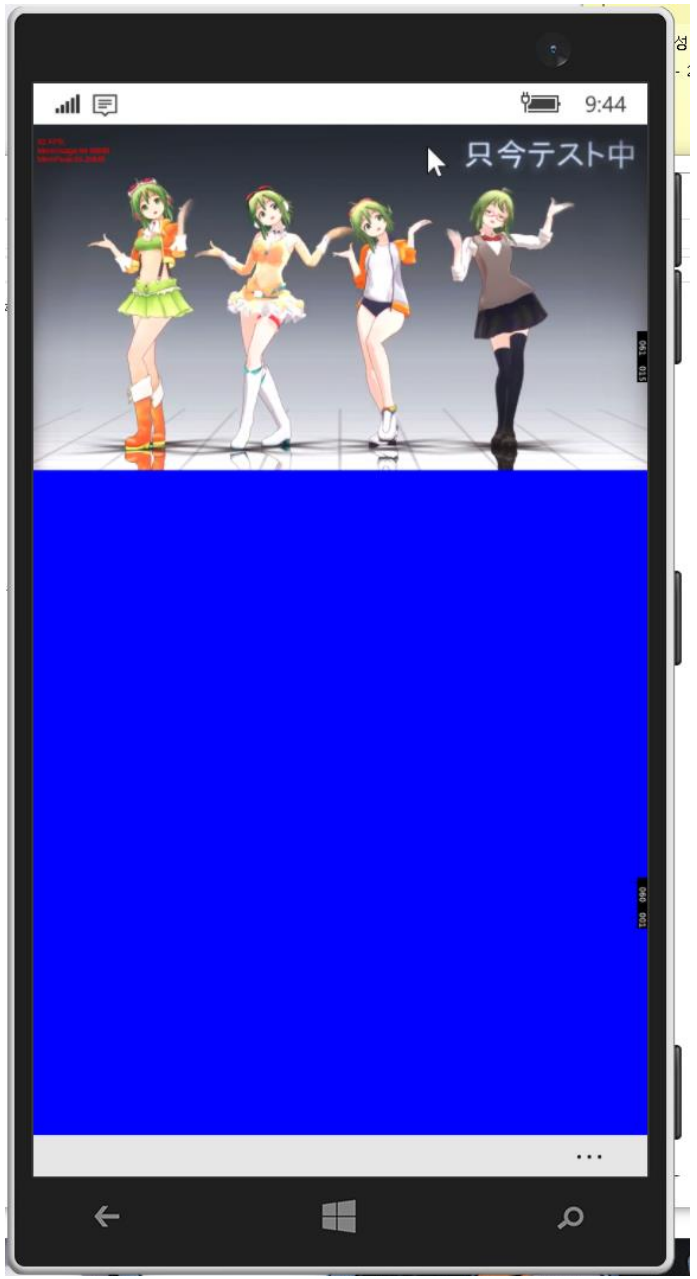
이런걸 만들어 봤습니다.



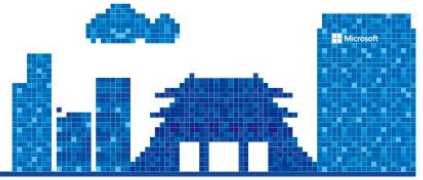
- D3DPlayer - ffmpeg 동영상 플레이어
  - 최초 발단은 Windows Phone에서 돌아가는 스트리밍 게임 클라이언트를 만들 생각이었다.
  - ffmpeg , Direct X(Direct 3D, Direct 2D) , C++
  - ffmepg interop 참고 (https://github.com/Microsoft/FFmpegInterop)
  - ffmpeg는 따로 빌드해야함.

- FileTransfer - TCP기반 파일 전송 앱
  - 내 Windos Phone으로 파일을 전송하고 싶은데 케이블이 없네.
  - 원격지의 친구에게 빠르게(peer to peer) 파일을 전송하고 싶다.
  - Winsock , C++

D3DVideoPlayer

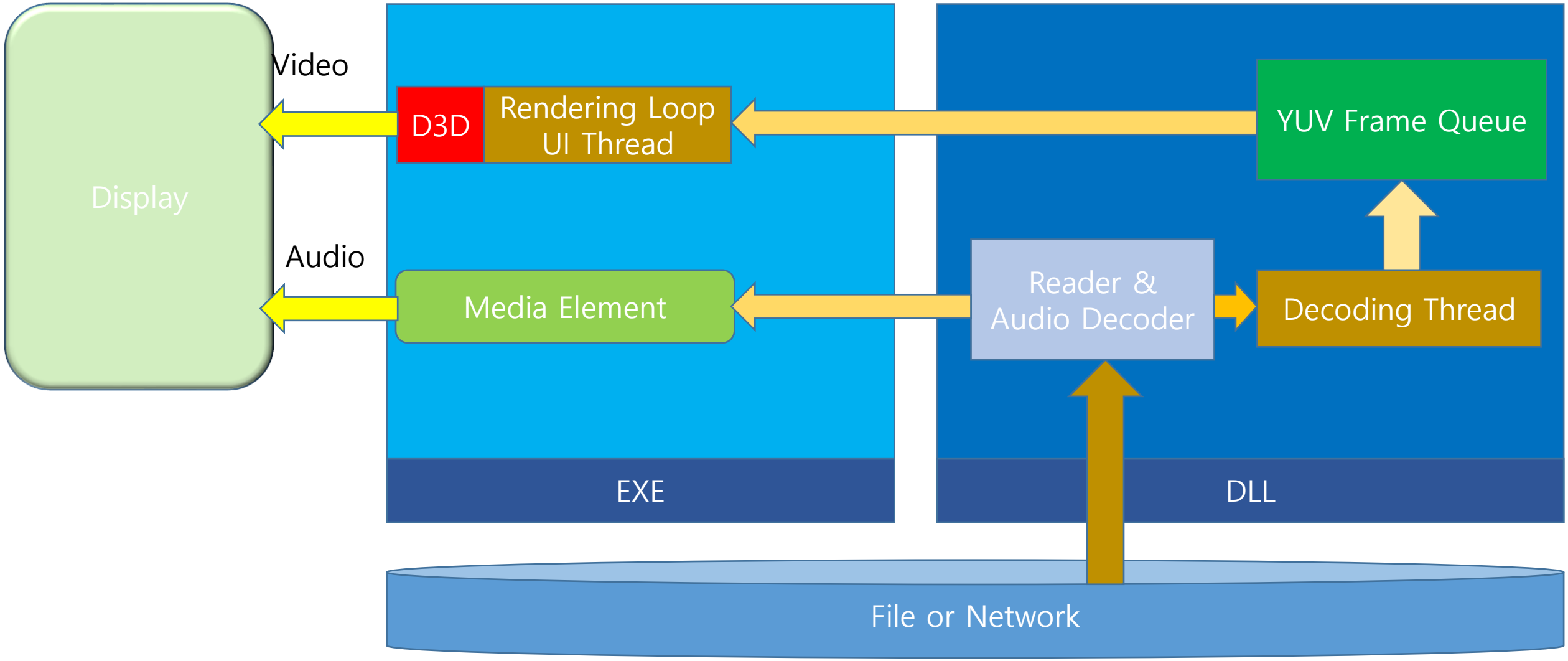


D3DPlayer – 목표 기능

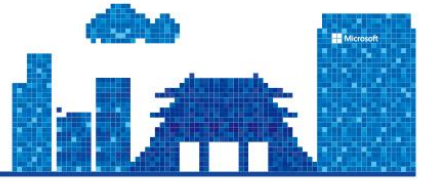


- ffmepg를 이용한 동영상 재생
- Shader를 이용한 간단한 후처리
- 비디오 출력은 D3D를 이용해서 직접 출력함.
- D2D를 이용한 텍스트 출력
- Windows Phone/Tablet/Desktop 동일 작동.

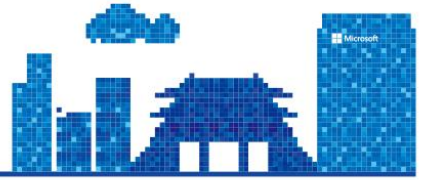
D3DVideoPlayer – 앱 구조



D3DVideoPlayer 데모



D3DPlayer – 디코딩

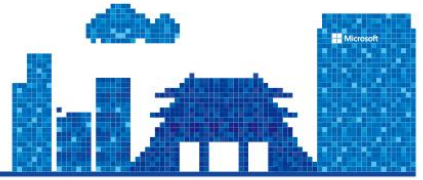


- MediaElement사용
  - 타이머 겸 오디오 스트림 출력
  - XAML에서 `<MediaElement x:Name="mediaElement"></MediaElement>` 한 줄로 끝.
- 오디오 스트림을 가지고 MediaStreamSource 세팅.
- 오디오 샘플 요청이 들어오면 큐를 채움.
- 비디오 샘플의 경우 디코딩 스레드가 백그라운드로 디코딩.
- 디코딩해서 얻은 YUV 프레임을 큐에 쌓아둠.

```
void CDecoder::DecodePacket(AVPacket avPacket)
{
    BOOLbGotPicture = FALSE;
    if (avcodec_decode_video2(m_pVCtx, m_pVFrame, &bGotPicture, &avPacket) >= 0)
    {
        if (bGotPicture)
        {
            DWORDstride = m_pVFrame->linesize[0];
            DWORDwidth = m_pVCtx->width;
            DWORDheight = m_pVCtx->height;

            if (avPacket.pts == AV_NOPTS_VALUE && avPacket.dts != AV_NOPTS_VALUE)
                avPacket.pts = avPacket.dts;

            Windows::Foundation::TimeSpan pts = { LONGLONG(av_q2d(m_pFmtCtx->streams[m_nVSI]->ti
            me_base) * 10000000 * avPacket.pts) };
            Windows::Foundation::TimeSpan dur = { LONGLONG(av_q2d(m_pFmtCtx->streams[m_nVSI]->ti
            me_base) * 10000000 * avPacket.duration) };

            PushYUVFrame(&bDestroyed, width, height, m_pVFrame->data[0], m_pVFrame->data[1], m_p
            VFrame->data[2], stride, (__int64)pts.Duration);
        }
    }
}
```

AVPacket으로부터 비디오 프레임 디코딩

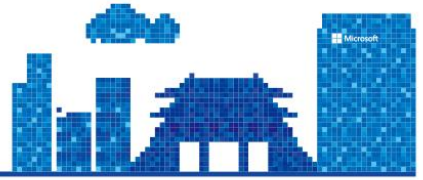
Direct3D/Direct2D 생성 및 초기화



- Desktop앱과 UWP앱의 DirectX 사용법은 95% 같다.
- Desktop앱에선 D3D 생성에 HWND가 필요하다.
- UWP앱 D3D생성에는 XALM로 페이지에 배치한 SwapChainPanel 객체가 필요하다.

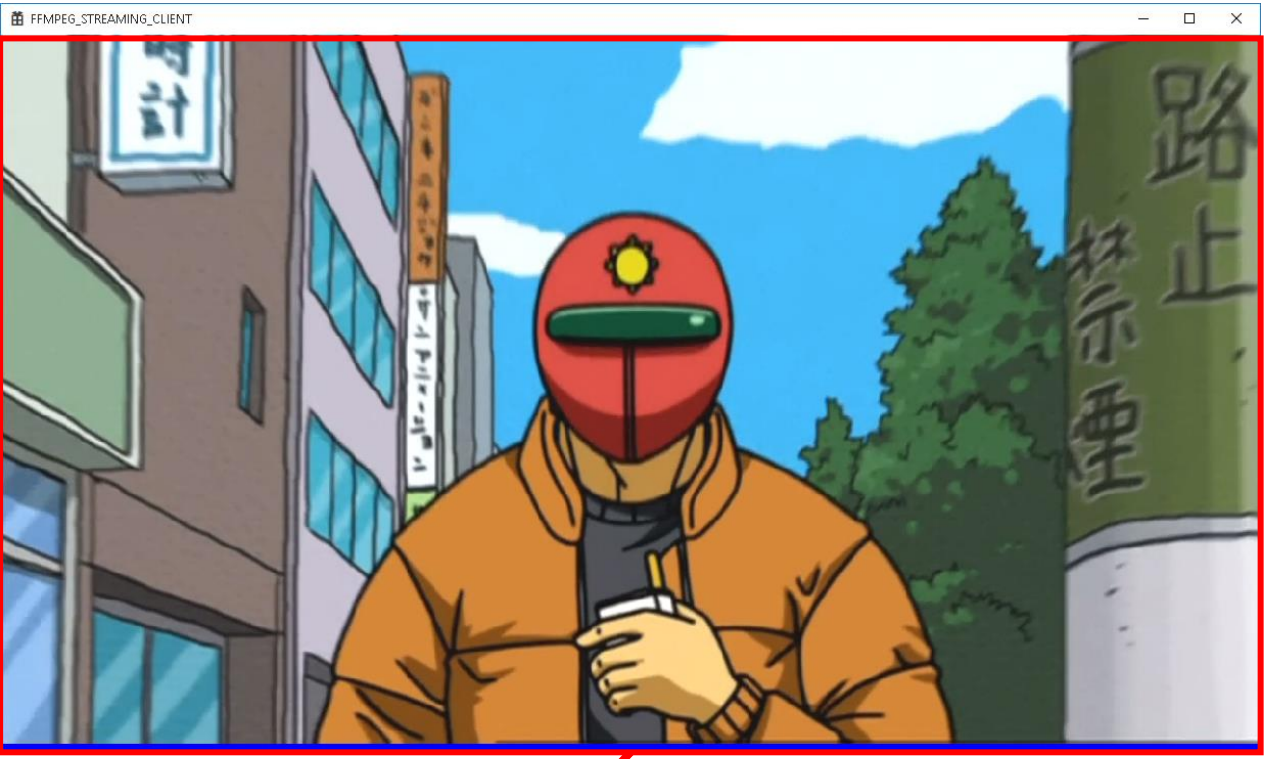
```
<SwapChainPanel x:Name="swapChainPanel"/>
```

Direct3D/Direct2D 생성 및 초기화



- UWP/Desktop 모두 IDirect3DDevice로부터 ID2DDevice 인터페이스를 얻는다.
- 단 Desktop API는 HWND로부터 Direct2D 객체를 직접 생성할 수 있다.
- DirectX를 사용함에 있어 UWP앱과 Desktop앱의 가장 큰 차이점은 SwapChain을 설정하는 방법이다.

Desktop Application – win32



GDI Windows Handle - HWND

UWP App

SwapChainPlanel

```
<SwapChainPanel x:Name="swapChainPanel"/>
```

in XAML

UWP앱에서의 SwapChain생성

```
ID3D11Device*pD3DDevice = NULL;
// create D3DDevice...blabla

IDXGIDevice1*pdxgiDevice = NULL;
pD3DDevice->QueryInterface(__uuidof(IDXGIDevice1), (void**) &pdxgiDevice);

IDXGIAdapter*pdxgiAdapter = NULL;
pdxgiDevice->GetAdapter(&pdxgiAdapter);

IDXGIFactory2*pdxgiFactory = NULL;
pdxgiAdapter->GetParent(__uuidof(IDXGIFactory2), (void**) &pdxgiFactory);

IDXGISwapChain1*pSwapChain = NULL;
DXGI_SWAP_CHAIN_DESC1 swapChainDesc = { 0 };
// Fill swapChainDesc…blabla

pdxgiFactory->CreateSwapChainForComposition(pD3DDevice, &swapChainDesc, nullptr, &pSwapChain);

ISwapChainPanelNative*pSwapChainPanelNative = nullptr;
reinterpret_cast<IUnknown*>(swapChainPanel)->QueryInterface(__uuidof(ISwapChainPanelNative), (v
oid**) &pSwapChainPanelNative);

pSwapChainPanelNative->SetSwapChain(m_pSwapChain);
```

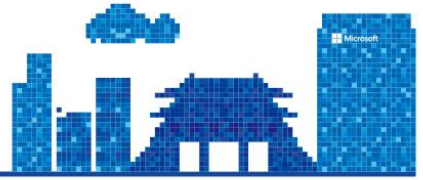
XAML에서 페이지에 배치한 <SwapChainPanel>객체를 D3D에서 생성한 IDXGISwapChain1 객체와 연결시킨다.

Desktop앱에서의 SwapChain생성

HWND를 받아서 DXGI_SWAP_CHAIN_DESC 구조체에 넣는다.
이 구조체로 SwapChain,D3DDevice,D3DDeviceContext를
한번에 생성한다.

```
D3D_FEATURE_LEVEL featureLevels[] =
{
    D3D_FEATURE_LEVEL_11_0
};
UINT numFeatureLevels = ARRAYSIZE( featureLevels );

DXGI_SWAP_CHAIN_DESC swapChainDesc = {0};
// Fill swapChainDesc…blabla
swapChainDesc.OutputWindow = hWnd;

IDXGISwapChain*pSwapChain = NULL;
ID3D11Device*pD3DDevice = NULL;
ID3D11DeviceContext*pImmediateContext = NULL;

HRESULT hr = D3D11CreateDeviceAndSwapChain( NULL, D3D_DRIVER_TYPE_HARDWARE, NULL, createDevic
eFlags, featureLevels, numFeatureLevels,D3D11_SDK_VERSION, &swapChainDesc, &pSwapChain, &pD3DDe
vice, &m_FeatureLevel, &pImmediateContext );
```

D3DPlayer – 렌더링



- 게임루프 처리하듯이 60프레임으로 폴링
  - 클라우드 게이밍용 클라이언트가 목적이므로
- 큐에서 현재 타임 스탬프와 일치하는 YUV프레임을 얻어옴.
- YUV포맷의 이미지를 직접 텍스쳐에 써넣는다.
- YUV포맷의 텍스쳐를 쉐이더 안에서 RGB로 변환
- 필터 모드에 따라 쉐이더를 다르게 선택
- 텍스트는 Direct2D를 이용해서 출력

유휴시간을 모두 사용하는 (게임용) 렌더링 루프

```
void MainPage::Init()
{
    EventHandler<Object^>^ ev = ref new EventHandler<Object^>(this, &MainPage::OnUpdate);
    Windows::Foundation::EventRegistrationToken RenderingEventToken = CompositionTarget::Renderi
    ng::add(ev);
}
void MainPage::OnUpdate(_In_ Object^ sender ,_In_ Object^  args)
{
    if (g_pVideoPlayer)
    {
        g_pVideoPlayer->OnRender();
    }
}
```

YUV프레임으로부터 텍스쳐 업데이트 - Lock

```
void CD3DRenderer::UpdateYUVTexture(DWORD dwWidth,DWORD dwHeight,BYTE* pYBuffer,BYTE* pUBuffer,
BYTE* pVBuffer,DWORD Stride)
{
    D3D11_MAPPED_SUBRESOURCE mappedResource = {0}
    pDeviceContext->Map(m_pYUVTexture,0,D3D11_MAP_WRITE_DISCARD,0,&mappedResource);

    DWORDStrideHalf = Stride / 2;
```

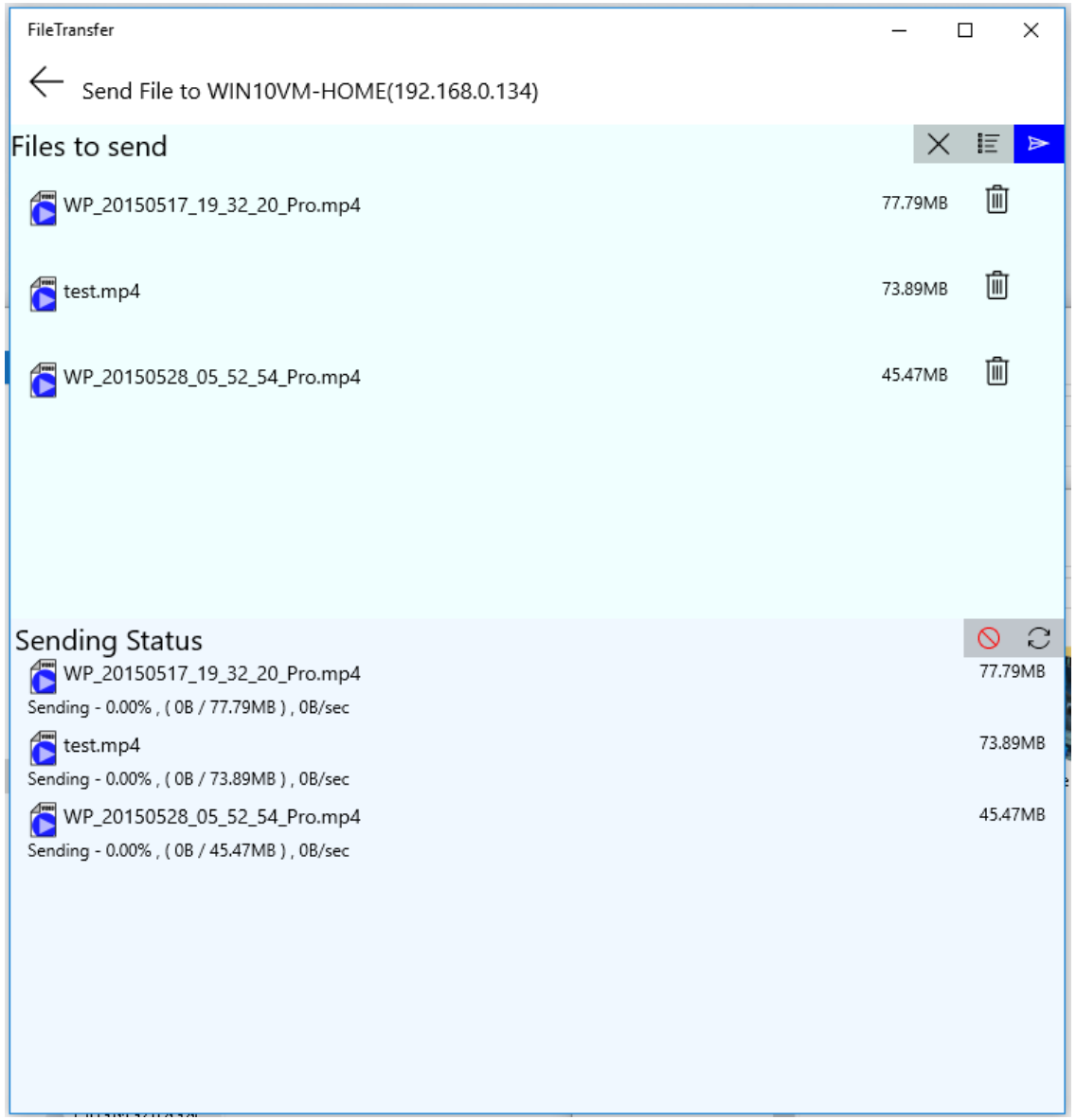
```
for (DWORD y=0; y<dwHeight; y++)
{
    BYTE* y_buffer_entry = pYBuffer + (Stride*y);
    BYTE* u_buffer_entry = pUBuffer + (StrideHalf*(y>>1));
    BYTE* v_buffer_entry = pVBuffer + (StrideHalf*(y>>1));

    BYTE*pDestEntry = (BYTE*)mappedResource.pData + (mappedResource.RowPitch*y);

    for (DWORD x=0; x<dwWidth; x++)
    {
        pDestEntry[0] = *y_buffer_entry;
        pDestEntry[1] = *u_buffer_entry;
        pDestEntry[2] = *v_buffer_entry;
        pDestEntry[3] = 0xff;

        y_buffer_entry++;

        DWORDuv_inc = x & 0x00000001;
        u_buffer_entry += uv_inc;
        v_buffer_entry += uv_inc;

        pDestEntry += 4;
    }
}
```

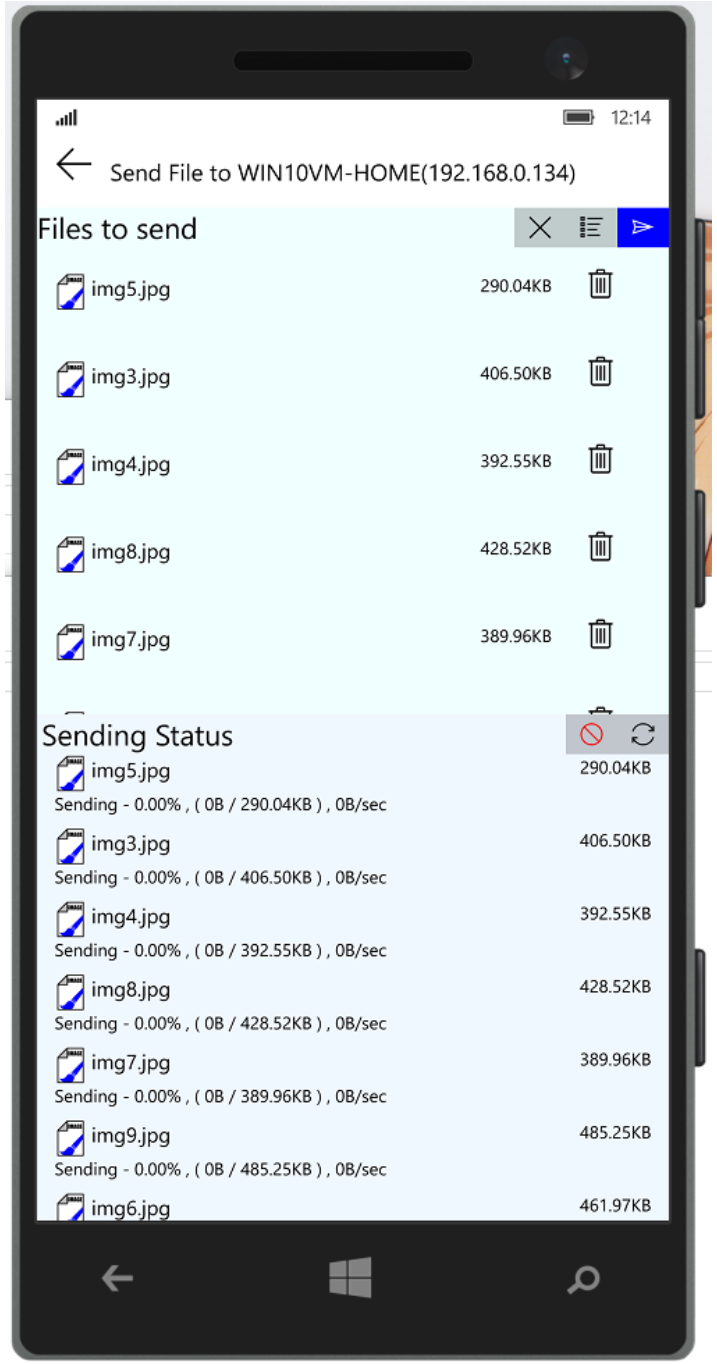
YUV프레임으로부터 텍스쳐 업데이트 - Wirte

```
    pDeviceContext->Unmap(m_pYUVTexture,0);
}
```

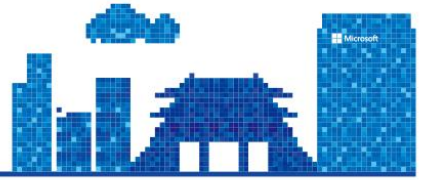
YUV프레임으로부터 텍스쳐 업데이트 - Unlock

FileTransfer



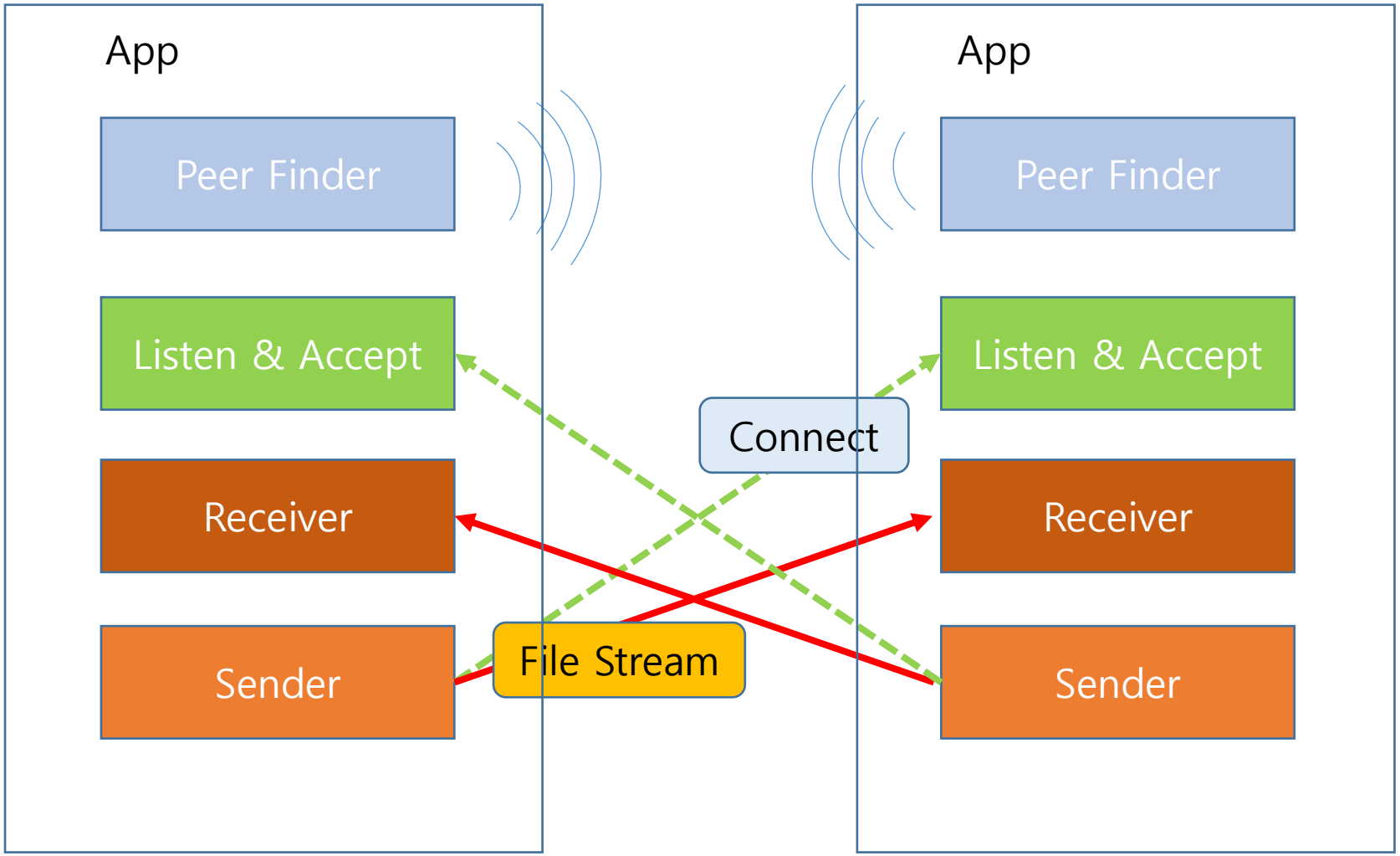


FileTransfer - 목표기능

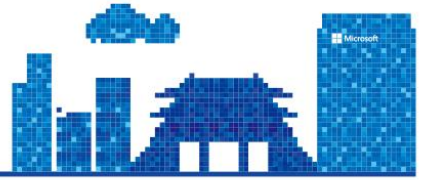


- Windows 10 디바이스간 파일 전송
- 여러 개의 파일 전송 가능
- 동시에 여러 개의 peer와 파일 전송 가능
- 저장할 폴더 선택 가능
- Windows Phone/Tablet/Desktop 동일 작동.
- 대부분의 코드를 공유하여 Win32 CUI버전도 개발

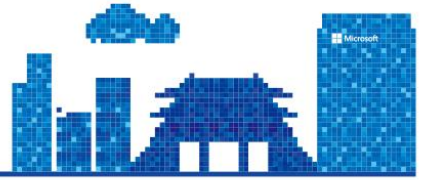
FileTransfer – 앱구조



FileTransfer - 데모

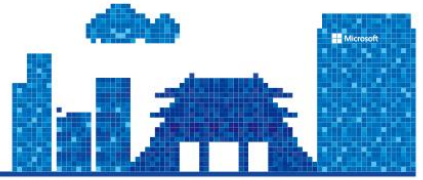


FileTransfer – peer간 접속



- UDP로 브로드캐스팅
- UDP패킷을 수신하면 TCP 포트를 열어서 listen
- 파일 전송 허가를 기다림

FileTransfer – 파일 송신/수신



- block모드 socke함수 send(),recv()사용.
- 전송 이벤트 하나당 스레드 하나씩
- Win32의 경우 소켓 에러 -1만 처리하면 됨.
- 파일 송수신 코드는 UWP/Desktop 99% 동일
- UWP의 경우 소켓 에러 상황에서 리턴값 외에 execption 처리가 필요할 수 있음. (1%의 차이)

패킷 수신

```
BOOL RecvPacket(SOCKET s,char* pBuffer,int size,int* piOutSocketError)
{
    *piOutSocketError = 0;

    BOOLbResult = TRUE;
    while (size > 0)
    {
        BOOLbException = FALSE;
        intiErrCodeOnException = 0;

        intrecv_result = -1;
        __try
        {
            recv_result = recv(s,pBuffer,size,0);
        }
        __except(GetExceptionCode() == EXCEPTION_INVALID_HANDLE ? EXCEPTION_EXECUTE_HANDLER : EX
        CEPTION_CONTINUE_SEARCH)
        {
            iErrCodeOnException = 10038;
            bException = TRUE;
            WriteDebugStringW(L"Exception was occured in recv().\n");
        }
```

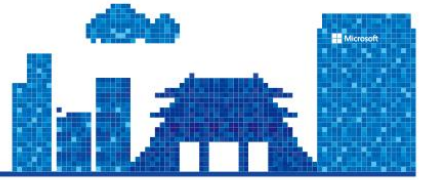
UWP에선 소켓에러 상황에서 SOCKET_ERROR를 리턴하는것 외에 추가적으로 익셉션을 발생시킨다.

StreamSocket이 내부적으로 winsock을 사용하기 StreamSocket을 위해 이렇게 작동하는 것으로 보인다. StreamSocket은 소켓 에러 상황을 익셉션으로 처리함.

패킷 수신

```
        if (0 == recv_result)
        {
            *piOutSocketError = 0;
            bResult = FALSE;
            break;
        }
        if (SOCKET_ERROR == recv_result)
        {
            if (bException)
            {
                *piOutSocketError = iErrCodeOnException;
            }
            else
            {
                *piOutSocketError = WSAGetLastError();
            }
            bResult = FALSE;
            break;
        }
        pBuffer += recv_result;
        size -= recv_result;
    }
    return bResult;
}
```

FileTransfer – 파일 읽기/쓰기



- UWP
  - Windows::Storage::StorageFile로부터 IRandomAccessStream^ fileStream을 얻는다.
  - FIleStream으로 DataWriter / DataReader를 생성,
  - 파일에 쓰기 - DataWriter::WriteBytes() , DataWriter::StoreAsync()
  - 파일로부터 읽기 - DataReader::LoadAsync() , DataReader::ReadBytes()

- Desktop – win32
  - CreateFile()로 파일 생성 및 오픈
  - 파일에 쓰기 – WriteFile()
  - 파일로부터 읽기 ReadFile()

폴더로부터 파일 생성

```
BOOL CFileReceiver::CreateFileAndRecv(Windows::Storage::StorageFolder^ folder , String^ FileName , ....
{
    Windows::Storage::CreationCollisionOption opt = CreationCollisionOption::ReplaceExisting;

    auto task_file = create_task(folder->CreateFileAsync(FileName,opt));
    StorageFile^ file = nullptr;
    try
    {
        file = task_file.get();
    }
    catch (Platform::Exception^ e)
    {
        String^ err = e->Message;
        String^ errMsg = L"Failed to CreateFileAsyc - " + err + L"\n";
        WriteDebugStringW(errMsg->Data());
    }
```

UWP앱은 완전한 Sandbox시스템이므로 아무 폴더나 임의로 억세스 할수 없다.
명시적인 방법으로 Windows::Storage::StorageFolder 객체를 얻어와야 한다.

파일 오픈

```
auto task_open = create_task(file->OpenAsync(FileAccessMode::ReadWrite));
IRandomAccessStream^ fileStream = task_open.get();

UINT64 CurPos = fileStream->Position;
fileStream->Seek(ui64StartPos);
CurPos = fileStream->Position;


DataWriter^ dataWriter = ref new DataWriter(fileStream);
```

UI Thread가 아니므로 task::get()으로 wait가능

파일/네트워크 스트림에 써넣기 위해서는
Windows::Storage::Stream::DataWriter를 사용한다.

파일에 수신한 스트림 저장

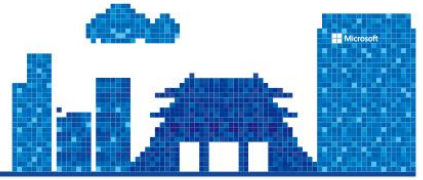
```
while (ui64FileSize > 0)
{
    int recv_size = (int)MAX_RECV_SIZE_PER_ONCE;
    if ((UINT64)(DWORD)recv_size > ui64FileSize)
    {
        recv_size = (int)(DWORD)ui64FileSize;
    }

    if (!RecvPacket(sock,m_pRecvBuffer,recv_size,piOutSocketError))
    {
        break;
    }
```

WriteBytes()에서 Write작업이 바로 이루어지지는 않는다.

```
    dataWriter->WriteBytes(Platform::ArrayReference<BYTE>((BYTE*)m_pRecvBuffer, (unsigned int)re
    cv_size));

    auto task_write = create_task(dataWriter->StoreAsync());
    size_t WrittenBytes = task_write.get();

    ui64FileSize -= recv_size;
}
```

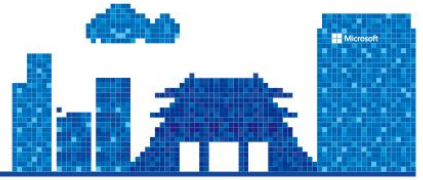
실제 write작업은 StoreAsync()호출 후에 비동기로 이루어진다.

잠깐! C#으론 못만들어요?

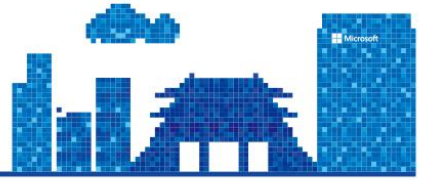


- 물론 됩니다.
- D3D , D2D , ffmepg , winsock을 사용하는 부분만 C++로 작성하고 C#에서 호출할 수 있도록 ref class로 포장하면 됩니다.
- 저라면 이렇게 복잡하게 하느니 차라리 전체를 C++로 작성하겠습니다.

결론

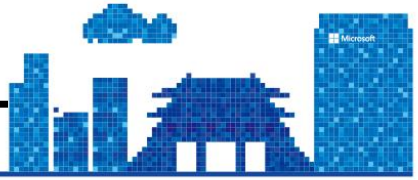


- 같은 성능이라면 사용자는 간편하고 깔끔한 설치/제거를 원한다.
- UWP앱으로 Desktop 앱의 영역을 어느 정도 대체 가능하다.
- C++을 사용해서 UWP앱을 개발하면 효율적으로 Desktop앱의 코드를 UWP앱으로 옮겨갈 수 있다.
- C++은 크로스 플랫폼 개발에 가장 효율적인 언어이다.
- 몇 가지 내용만 학습하면 기존 C++ 프로그래머들이 어렵지 않게 UWP앱을 개발할 수 있다.



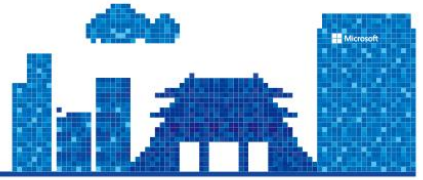
Q / A

Classic Win32 개발자들이 자주 하는 질문



- MessageBox를 띄우고 싶어요.
- 작업 중간에 MessageBox를 띄우고 응답이 올때까지 작업을 대기시키고 싶어요.
- WriteFile() ,ReadFile() ,fread() ,fwrite()를 사용하고 싶은데 문제가 있나요?
- 임의로 아무 폴더의 파일을 open할수 있는가?
- 폴더(디렉토리)간 이동을 하려면?
- DLL은 사용 가능한가요?
- Memory Leak을 확인하려면?
- create_task().then()이 짜증나요.
- Lambda 쓰기 싫어요.

Reference



- C++/CX의 ref class객체들을 STL컨테이너와 함께 사용할 경우 ref count관리는 어떻게 이루어지는가 (http://wp.me/p6sbBj-aZ)
- Windows 10 UWP에서 D3D객체들이 완전히 해제되었는지 확인하기 (http://wp.me/p6sbBj-6P)
- C++로 WIndows 10 UWP앱 개발할때 드래그 앤 드롭 처리하기 (http://wp.me/p6sbBj-7i)

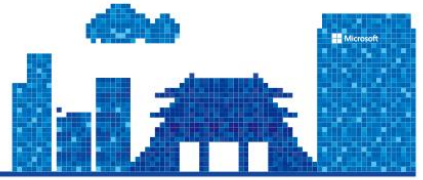
Microsoft

감사합니다.

- MSDN Forum http://aka.ms/msdnforum
- TechNet Forum http://aka.ms/technetforum

tech·days
Korea 2015

2015.10.27(화) 09:00 AM -09:00 PM
세종대학교 컨벤션센터 B2



TechDays Korea 2015에서 놓치신 세션은
Microsoft 기술 동영상 커뮤니티 **Channel 9**에서
추후에 다시 보실 수 있습니다.

http://aka.ms/td2015_again

Microsoft

tech·days Korea 2015