Designing games for Windows

Paul van Wingerden
Technical Evangelist

//build/

What is a PC?

SAMSUNG

Wednesday, 21

Start
ASUS

vimeo

What is PC gaming?

‘[Microsoft] has apparently elected to take a much riskier, and gutsier, approach: turn Windows 8 into a serious gaming platform. Microsoft can't help but get into "PC" gaming in a big way, because there's no such thing as PC gaming or tablet gaming or mobile gaming. It's all the same thing.’

Matt Asay, The Register

New gaming scenarios

Developers have an entire universe of new gameplay scenarios to explore with Windows 8.

Single device, single game, multiple environments

Single game, multiple inputs

Single game, single player, multiple device types

Single game, multi-player, multiple device types

Single place, multiple tablets

Working while playing, playing while working

Playing while playing

Cross-context taming

The best games go where you go and play the way you like to play without compromise.

Single device, single game, multiple environments

Single device at work, at home, on the go

Same device rendering to multi-monitor display, big screen TV, 10” screen

Single device, single game, multiple inputs

Games work well with touch, keyboard and mouse, sensors, gamepads

Single game, single player, multiple device types

Games purchased and played on laptop, tablet, desktop, all-in-one, flipbook, …

Cross-device gaming

The best multi-player games make the most of the users device and environment without negatively impacting consistency or competitiveness.

Single game, multi-player, multiple device types

Competitive and cooperative games using full range of devices

Games designed to level the playing field, leverage the affordances of each device

Single place, multiple tablets

Gaming in households and at BYOD events in new and unexpected ways using NFC, DLNA, Bluetooth, GPS, …

Distracted gaming

The most successful game developers will master keeping developers engaged even when they're distracted.

Working while playing, playing while working

Snap, live tiles, and toast notifications to stay in the game

Playing while playing

Consuming entertainment experiences simultaneously

Designing games for Windows

Paul van Wingerden
Technical Evangelist

//build/

Session agenda

Attention and engagement inside and out

Interaction models

Exploiting Windows features

Individual and social connectivity

Tiles

Cycle images and game information

Invite back and engage for longer

Show scores, achievements

Display turns, challenges, matches, missions

Secondary tile pinning for checkpoint or concurrent sessions

Active even while game is not

Served by Windows Live scale

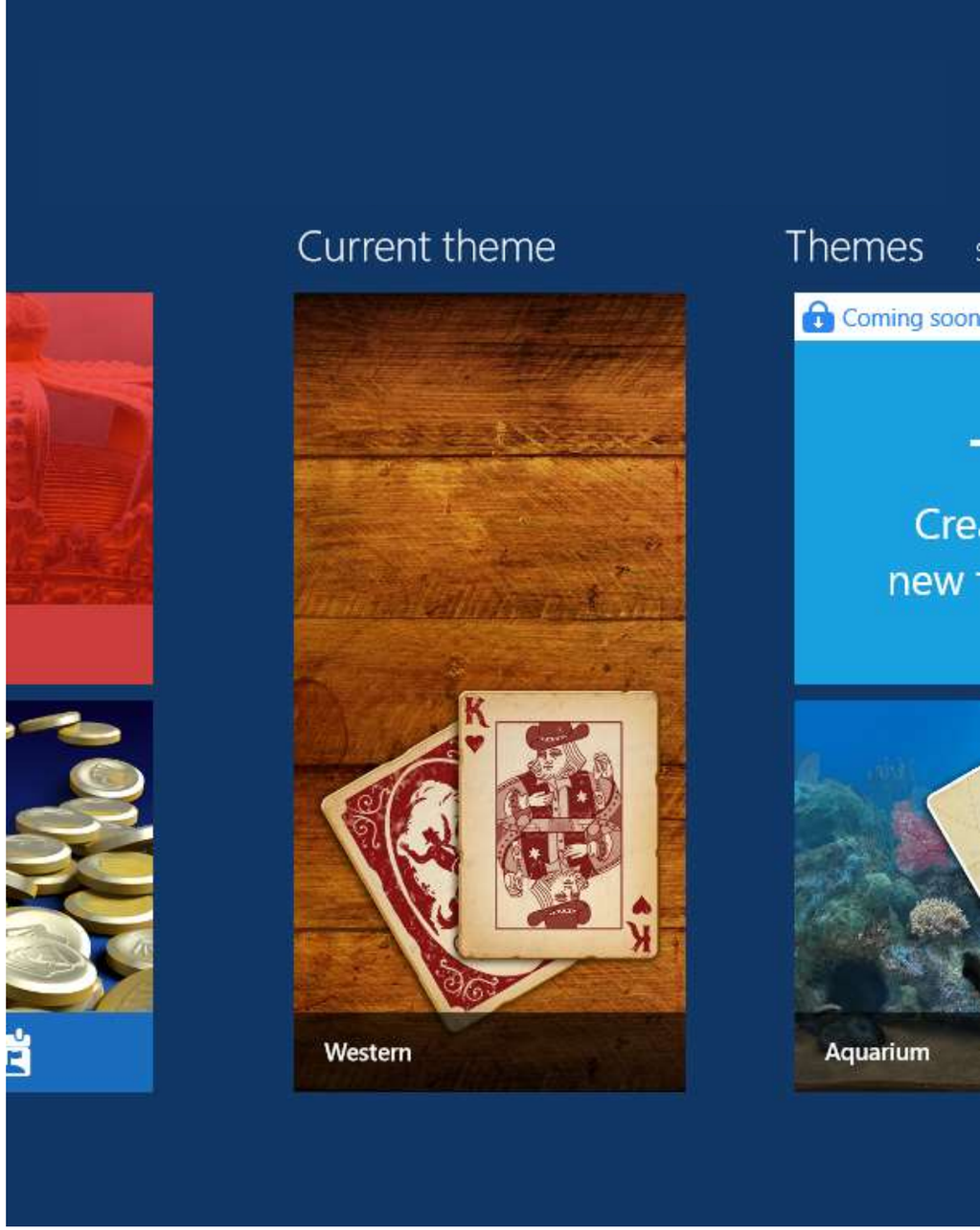
Square tiles and desktop application tiles

Notifications

Additional engagement opportunities inside your game

Display “popup toast” to communicate:

- High scores, achievements, challenges or updates
- Game information (your turn, new DLC available)

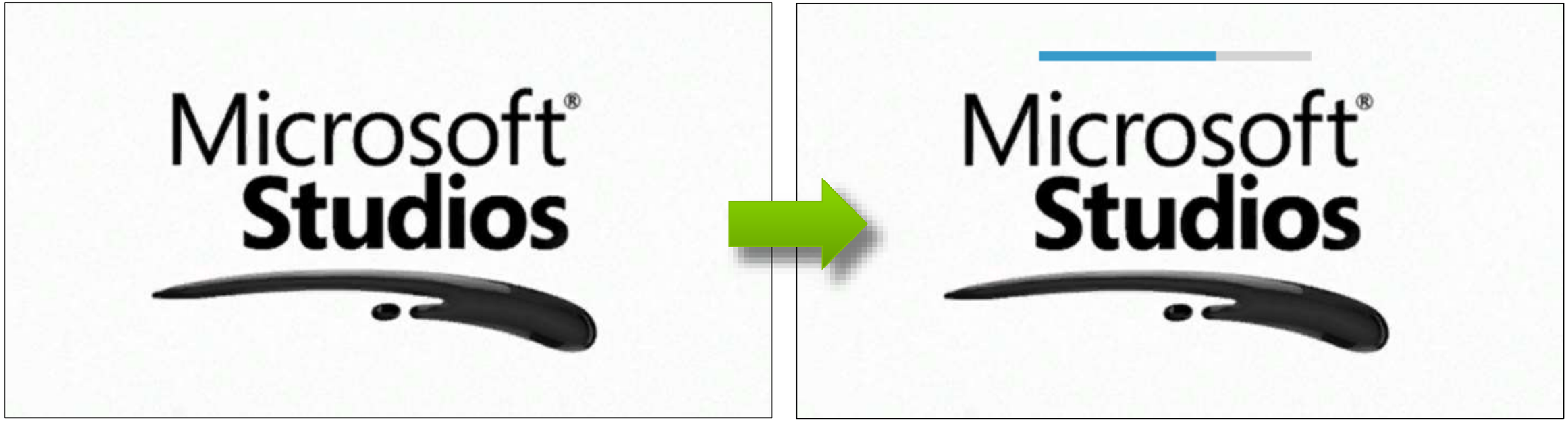


Splash screens and extended loading

Hand off from platform Start screen to game intro

Provide an extended splash screen for launches greater than 3 seconds

Engage by showing progress while loading assets



Layout and navigation

Landing page

Hub (hierarchical) for multiple experiences

- Leaderboards
- Achievements
- Themes

Consistent scrolling direction

Semantic (“pinch”) zoom

Layout and navigation

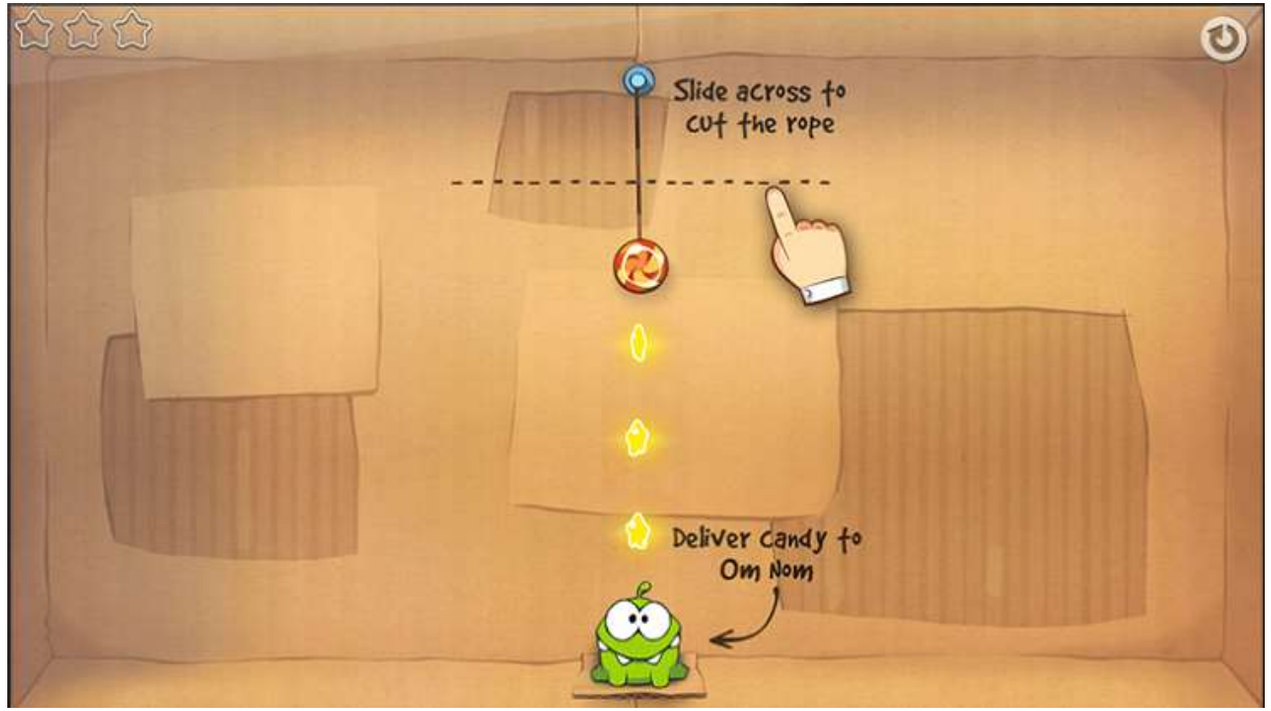
Flat for focused gameplay (boards, puzzles)

Begin right away

On screen controls

App bar for secondary controls

Gameplay interactions



Direct manipulation via touch

Touch and gestures

Keyboard or mouse enabled

Dynamic input-mode transitions

On screen primary controls

Secondary controls in the app bar

Opportunities for monetization

Input and sensors

New breed of interaction opportunities beyond mouse and keyboard

Multi-touch, swipes, and defined gestures

Accelerometers can be used for steering cars or tilting game elements

Device movement to rotate a character or a camera viewpoint

Sensor Fusion to enable precise orientation and location data

Shake to defend from enemies or reset

Light sensor to change the mood or lighting of a game's rendering

Microphone or camera to integrate environmental elements

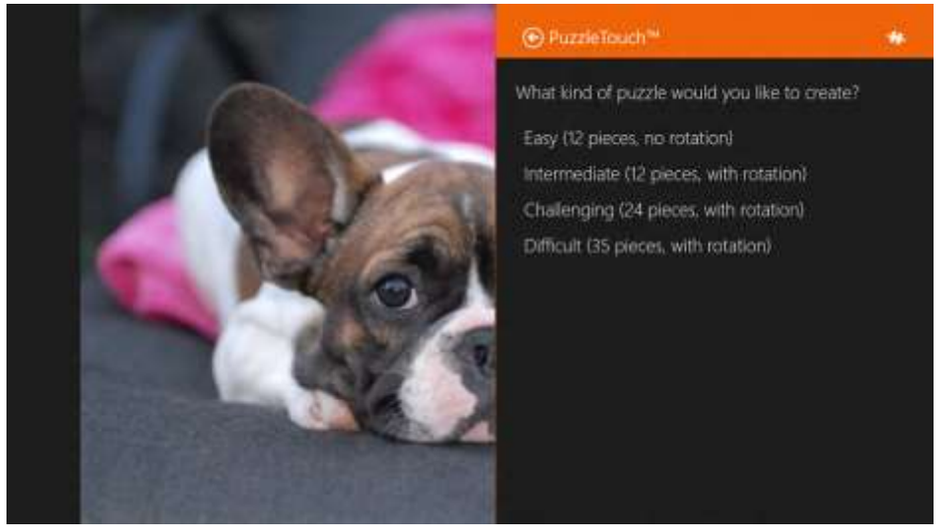
Contract charms

Better together; Branding and engagement opportunity

Share/Search: score, achievements, items, game play

Share/Search: initiator or provider

Context sensitive



Play To: extend for secret screen

Hand of cards

Letter tiles for word games

Battle ship

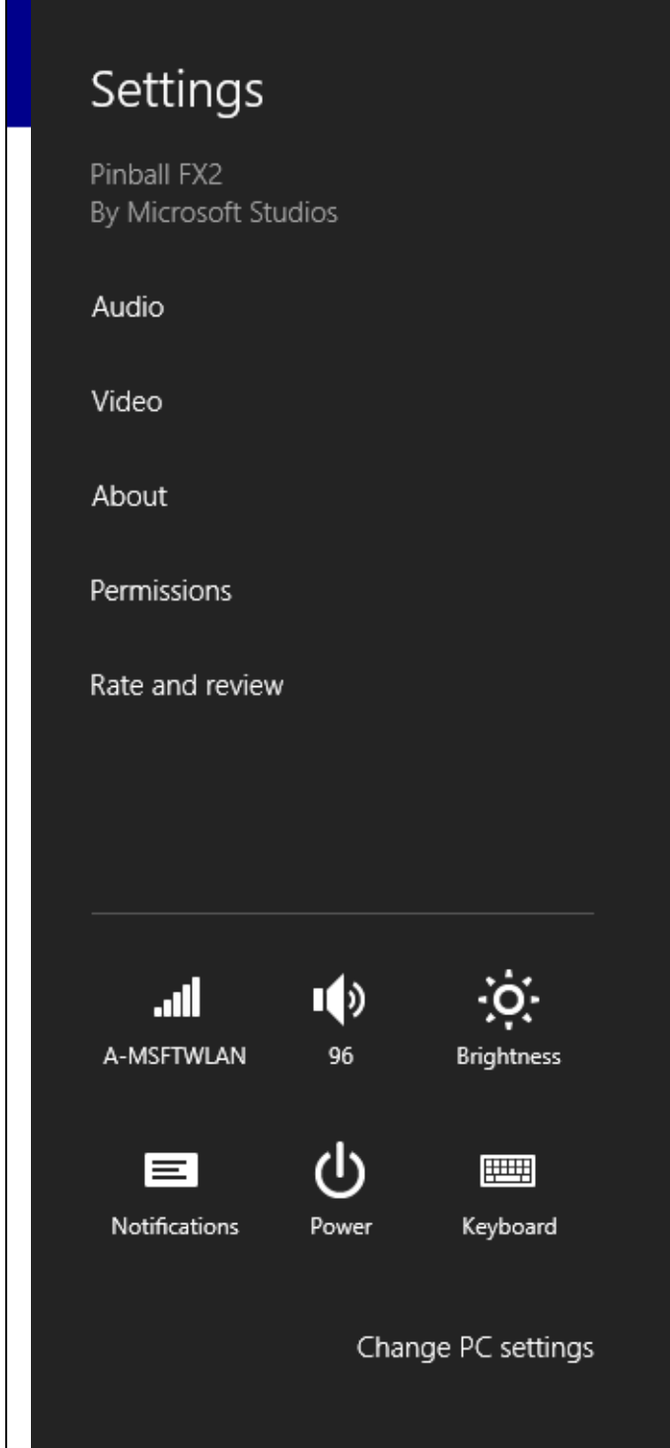
Settings

Arrange in logical groups

Options: difficulty levels, audio levels, game preferences
Privacy policy: protection of personal data and liability
Help: context sensitive tutorials, hints, support and contact information
About: developer, publisher, version, build
Login: Sign in/out or switch gamer service
Credits: kudos, thanks and “production babies”

Real time sliders

Standard or branded UI



View states and orientation changes

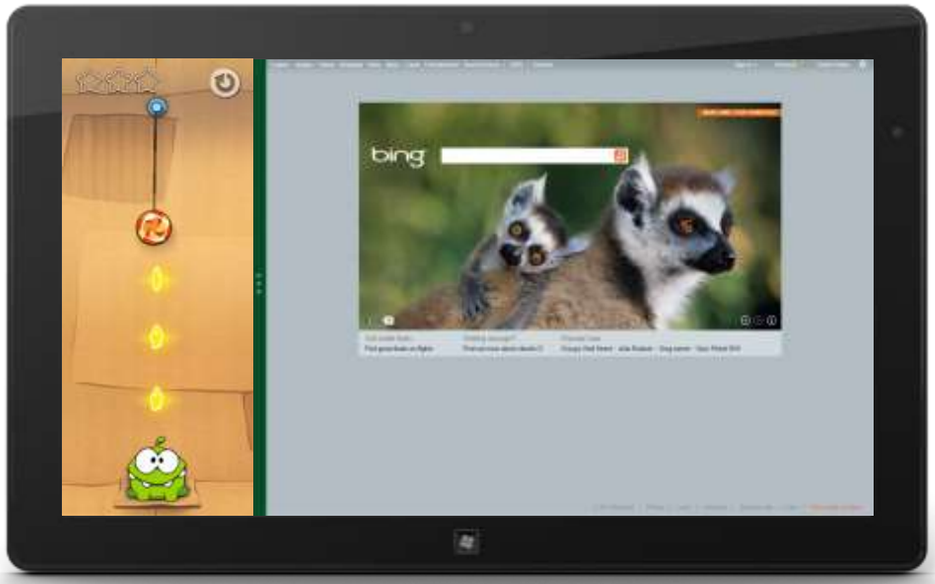
Full screen, full bleed, border and menu free

Snap View: 320 pixels – offer secondary engagement

Fill View: 1366 – 320 = 1046 (4:3) maintain primary engagement

Portrait View: “optional” – consider rotation animation

App bar considerations for each case



Pausing the game

Loss of focus or view changes

Careful about rotation

App bar swipes (or RMB)

Toggle play/pause control

Minimal or total obfuscation

Countdown to return

Roaming across devices

PLM State management

Checkpoints and background game play saving

Play, pause, resume experiences

Continuous gameplay across devices

Consistent settings and options

Player accounts

Connect to gaming services

Tracking players progress

Link to social network

Friction-free commerce

Leverage Microsoft ID federation

Advantages of the new app model

Fast and fluid UI interaction "touch first"

Continuous app operation

App autonomy/independence

App collaboration

New hardware form factors

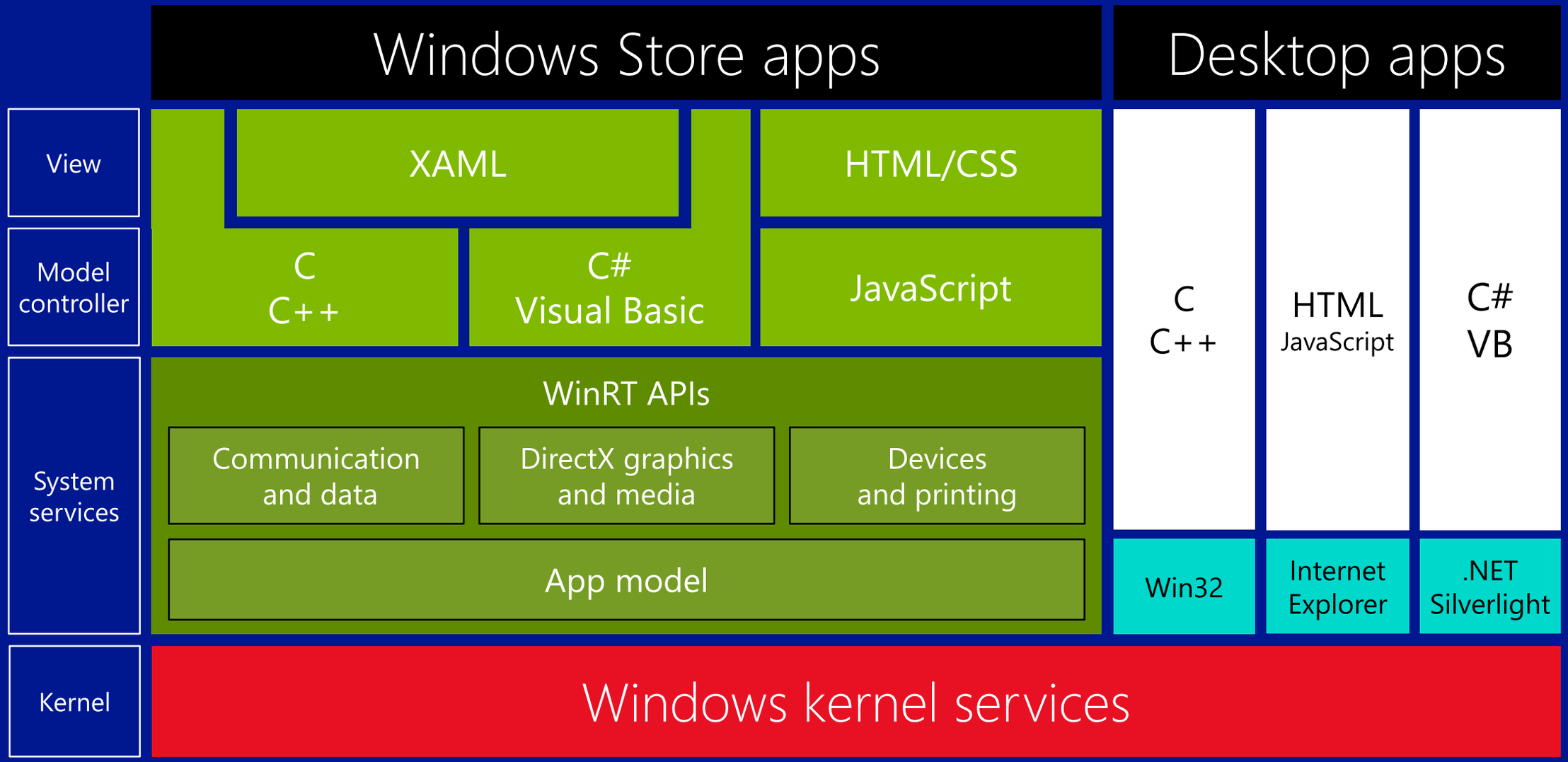
Reduced power consumption

Clear development process

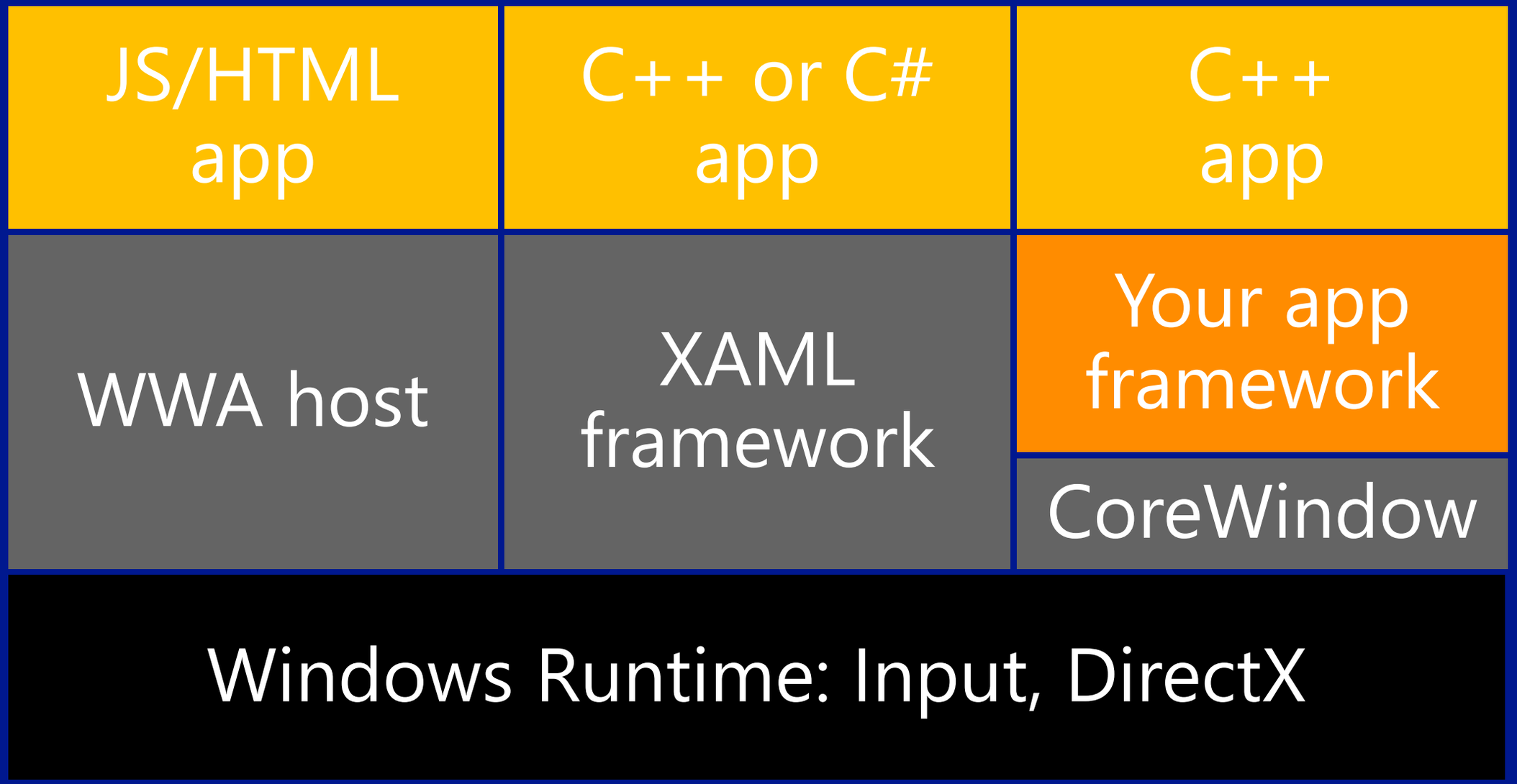
Latest APIs for each area

Focused tools and docs

Efficient distribution


Windows Store apps
Desktop apps
View
Model controller
System services
Kernel
XAML
HTML/CSS
C
C++
C#
Visual Basic
JavaScript
WinRT APIs
Communication and data
DirectX graphics and media
Devices and printing
App model
C
C++
HTML
JavaScript
C#
VB
Win32
Internet Explorer
.NET Silverlight
Windows kernel services

App structure options



JavaScript games

UI and layout in CSS/HTML

Gameplay in HTML5 canvas

JavaScript access to WinRT APIs

Access game controllers, video, audio, and networking via C++

Social games, arcade games, sprite games, puzzles, adventure games, card or board games

C# games

XAML UI kit for menus

Ability to render via SwapChainBackgroundPanel

Option to drive via C# or C++

3-D games (FPS, RPG)

Demo SwapBackGroundPanel

C++ Games

Highest performance =
broadest range of hardware

XAML UI kit for menus

"Twitch" games

Action platformers

3-D games (FPS, RPG)

Augmented reality games

The Windows Runtime

.Net runtime

Interpreted runtime, runs in a sandbox
Garbage collected
Supports XAML UI toolkit
Supports managed languages

Windows Runtime

Native runtime, apps run in a sandbox
Reference counted
Supports XAML UI toolkit
Can call from JS, managed langs, and C++
- More efficient binding to C# than p/Invoke

High-performance graphics

Expected feature levels

Hardware until mid-2014:

X86 systems are DX9_3 through DX11

So no need for optimizing to a DX9_1 codepath on x86

ARM systems are DX9_1 through DX9_3

So no need to ship large DirectX 11 assets in the ARM package yet

Select feature levels to support

```
D3D_FEATURE_LEVEL featureLevels[] =
    {
        D3D_FEATURE_LEVEL_11_1,
        D3D_FEATURE_LEVEL_11_0,
        D3D_FEATURE_LEVEL_10_1,
        D3D_FEATURE_LEVEL_10_0,
        D3D_FEATURE_LEVEL_9_3,
        D3D_FEATURE_LEVEL_9_1
    };

UINT creationFlags = D3D11_CREATE_DEVICE_BGRA_SUPPORT;
```

A best practice

If using DirectX and CoreWindow, be sure to pause rendering (only check for input) if your window is occluded (**`OnVisibilityChanged`** arg is false)

This is lets users easily switch to or from the Start experience while your game is running

This makes the system run better, and reduces the chance of your app being suspended or pulled out of memory

Threading and processors

The challenge of multiple processors

Extracting every last cycle of execution from a set of cores may not pay off vs. the programming effort required

And may make it harder for other OS or app or middleware components to run

Multithreading helper code

We offer some sample code that emulates a subset of the Win32 CreateThread API on top of the WinRT ThreadPool

It won't work quite the same, but it should make the porting process easier

One thing it does try to support is the concept of a long-running thread, such as for ‘background’ processes

Createthread for Windows Store apps

Thread pool resources

http://msdn.microsoft.com/en-us/library/windows/desktop/ms684841(v=vs.85).aspx

http://msdn.microsoft.com/en-us/library/windows/desktop/ee416321(v=vs.85).aspx

http://msdn.microsoft.com/en-us/magazine/cc163327.aspx

http://msdn.microsoft.com/en-us/magazine/cc163405.aspx

Game input for all Windows 8 PCs

Windows 8 PC spectrum

Game control schemes

Q: Which input or control mechanisms should I use?

A: As many as you can!

Control schemes for gameplay

FPS: First-person mouse-look (and touch-look)

Always on during gameplay, off during menus

Touch (touchlook?) also works for this as shown in popular titles

3-D RPG: First-person mouselook/touch

Enabled based on mouse button down (usually right mouse button)

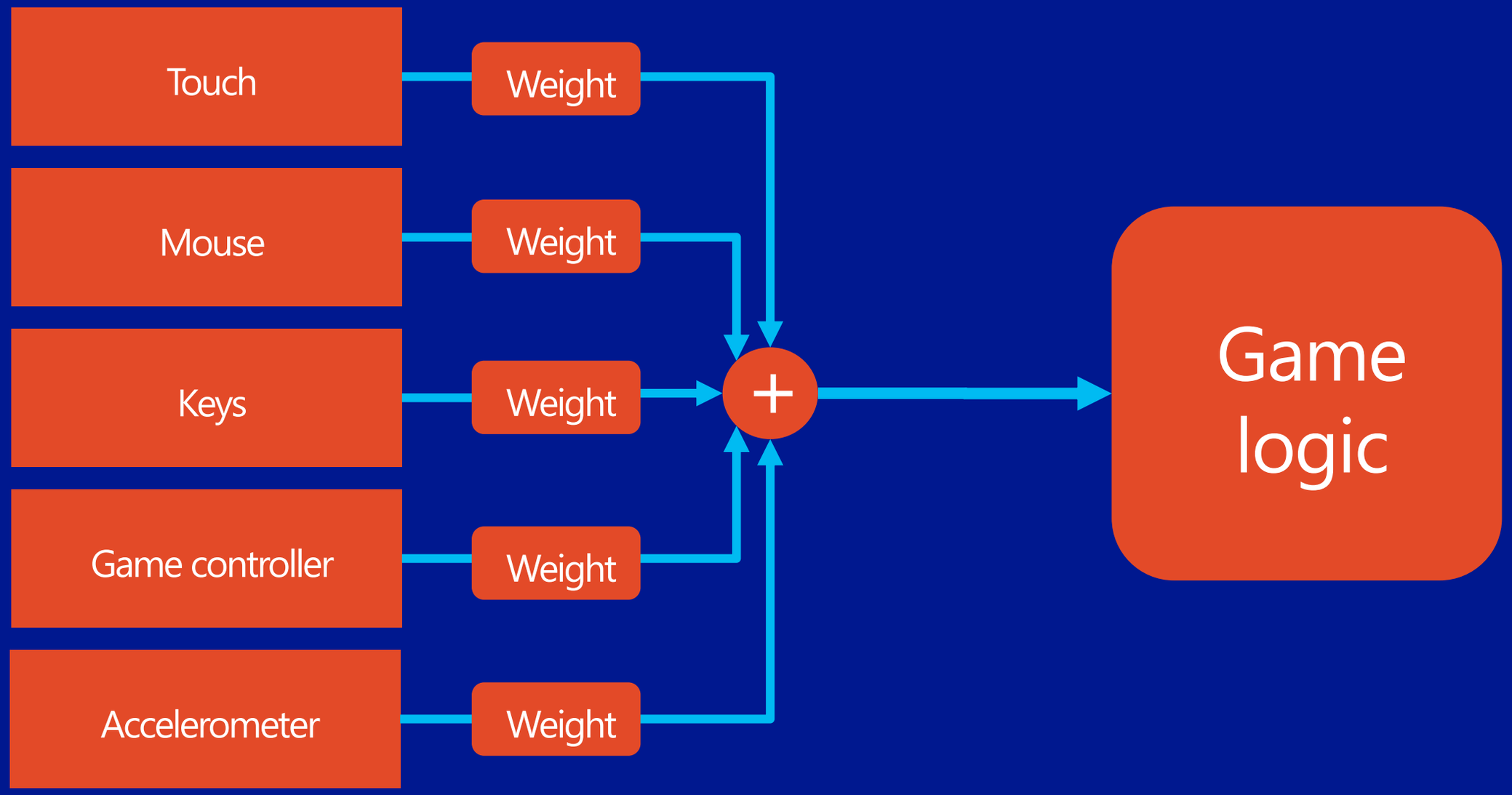
'Always on' with touch devices

Third-person action RPG: 'tap to move'

Picking up loot is tough with touch:

Make the 'hit' zone at least as large as a fingertip

Controls: Input combination during play



First-person mouse-look controls

First-person games use the mouse in relative mode

Mouse API returns incremental values since last frame

Absolute values still available and used to position cursor

When relative mouse handler is bound, and cursor is hidden:

```
window->PointerCursor = nullptr;
```

Then absolute mouse position is fixed until cursor is unhidden:

```
window->PointerCursor = ref new CoreCursor(CoreCursorType::Arrow, 0);
```

[DirectX Shooting Game sample]

[DirectX touch input sample]

```
 // for mouse-only use
 // register handler for relative mouse movement events
 Windows::Devices::Input::MouseDevice::GetForCurrentView()->MouseMoved +=
    ref new TypedEventHandler<MouseDevice^, MouseEventArgs^>
        (this, &MoveLookController::OnMouseMoved);

// Handle Mouse Input via dedicated relative movement handler
void MoveLookController::OnMouseMoved(
    _In_ MouseDevice^ mouseDevice,
    _In_ MouseEventArgs^ args
    )
{
    float2 pointerDelta;
    pointerDelta.x = static_cast<float>(args->MouseDelta.X);
    pointerDelta.y = static_cast<float>(args->MouseDelta.Y);

    . . .
```

Relative mouse handler

Consistent controls scheme – menus

Command	Windows Phone	Touch tablet	Keyboard	Game controller
Play/resume	**“Play” button**	**“Play” button**	**“Play” button/ menu option**	**Play item/option, triggered with (A) button**
Game-mode to Menu-mode (pause)	**Physical ← key**	**Swipe-in from top or bottom edge**	**Escape key**	**Physical pause button**
Navigate in menus	**Touch taps**	**Touch taps**	**Arrow keys, enter key to select, mouse clicks**	**Digital D-pad**
Previous menu	**Back control**	**Back control**	**Backspace key** Escape key can work in addition	**Back(B) button**

Managing larger apps

Splitting up your app into separate ‘screens’ can make development simpler. There are multiple options

Separate .exes in the same package
Bound to different activation contracts

Multiple IFrameworkView objects
Many apps have multiple screens today

Multiple CoreWindows in the same view
Each with its own SwapChain

Creating another FrameworkView

```
m_dxView = Windows::ApplicationModel::Core::CoreApplication::
CreateNewView("GameScreen.DirectXView","Example Entrypoint String");
```

Lets you have a full XAML environment for UI-heavy menus, options, etc.

Then have a separate DirectX-only screen

Yet another approach to XAML interop beyond SwapChainBackgroundPanel

Best practices for pause and rotation

Decision – Rotation

Let OS rotate

Recommended for most apps

May be disruptive of some types of gameplay

Recommended for

For games that can pause

Event-based games

- *Solitaire, Mahjong*

Any game that is paused or at a menu

Non-game apps

Block rotation by OS

No potential for glitching gameplay

If user rotates

OS UI is invoked from different edge

Notifications may be sideways or inverted

Recommended for

Full-real time games

- *For example, Arcade games*

Accelerometer/gyro games

- *For example, Marble Maze*

Games that can't pause

Rotation preferences

```
{
   DisplayOrientations  None                // Enable rotation by OS/Accelerometer
   DisplayOrientations  Landscape          // Lock rotation by OS/Accelerometer
   DisplayOrientations  LandscapeFlipped   //     and enable this orientation
   DisplayOrientations  Portrait
   DisplayOrientations  PortraitFlipped
}

using namespace Windows::Graphics::Display;

DisplayProperties::AutoRotationPreferences = DisplayOrientations::Landscape;

DisplayOrientations::LandscapeFlipped;
```

Persistent storage model

Persistent storage model

File system hierarchy

App package installation folder – locked/protected

Program files

"C:\Users\chasb.REDMOND\AppData\Local\Packages\Microsoft.Bing_8wekyb3d8bbwe"

AppData/roaming – limited size (store links to saved games)

Sideloading game data files

Windows Store currently limits packages to 2GB total size

Third-party service can provide additional assets in AppData folder (not app package installation folder)

Asset files like this cannot contain executable code

Copying additional assets to AppData/Local

```
void myGame::OnActivated(CoreApplicationView^ applicationView,
                         IActivatedEventArgs^ args)
{
    CoreWindow::GetForCurrentThread()->Activate();
    if(args->Kind == ActivationKind::File)
    {
        auto item = safe_cast<FileActivatedEventArgs^>(args)->Files->GetAt(0);
        if(item->Name == "texturepack.MyGame_wad")
        {
            safe_cast<StorageFile^>(item)->CopyAsync(
               ApplicationData::Current->LocalFolder,
               "data.wad",
               NameCollisionOption::ReplaceExisting );
        }
    }
}
```

The biggest opportunity

Tablets are a new thing, with their own identity/character

Longer sessions than phone, better display than phone

Broader range of users, times for playing than PCs

Most tablet games today are upgraded phone games

‘New’ technologies are available from PCs: Multiplayer, 3D

The killer app for tablets is not yet written.

Your move.

Get started

Check out the DirectX Samples:

Visual Studio project templates	File->New Project->C++->DirectX
Direct3D Tutorial Sample	D3DTutorial
DirectX Touch Input Sample	Simple3DTouch
Direct3D Sprite Sample	Simple3DSprites
Direct3D Resource Loading Sample	Direct3DResourceLoading
Direct3D Bump Mapping Sample	BumpMapping
DirectX Marble Maze Game Sample	Marble Maze
DirectX 3D Shooting Game Sample	Simple3DGame

…

Get your app onto the Windows Store

Resources:

May 2012 Camp: Developing Windows Store Apps in C++

Resources

Link to more detailed talks referenced:

May 2012 Camp: Developing Windows Store Apps in C++

Please submit session evals on the Build Windows 8 App or at http://aka.ms/BuildSessions

Resources

- Develop: http://msdn.microsoft.com/en-US/windows/apps/br229512
- Design: http://design.windows.com/
- Samples: http://code.msdn.microsoft.com/windowsapps/Windows-8-Modern-Style-App-Samples
- Videos: http://channel9.msdn.com/Windows

Please submit session evals by using the Build Windows 8 app or at http://aka.ms/BuildSessions

Resources

- Develop: http://msdn.microsoft.com/en-US/windows/apps/br229512
- Design: http://design.windows.com/
- Samples: http://code.msdn.microsoft.com/windowsapps/Windows-8-Modern-Style-App-Samples
- Videos: http://channel9.msdn.com/Windows

Please submit session evals by using the Build Windows 8 app or at http://aka.ms/BuildSessions

© 2012 Microsoft Corporation. All rights reserved. Microsoft, Windows, Windows Vista and other product names are or may be registered trademarks and/or trademarks in the U.S. and/or other countries.
The information herein is for informational purposes only and represents the current view of Microsoft Corporation as of the date of this presentation. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information provided after the date of this presentation. MICROSOFT MAKES NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, AS TO THE INFORMATION IN THIS PRESENTATION.