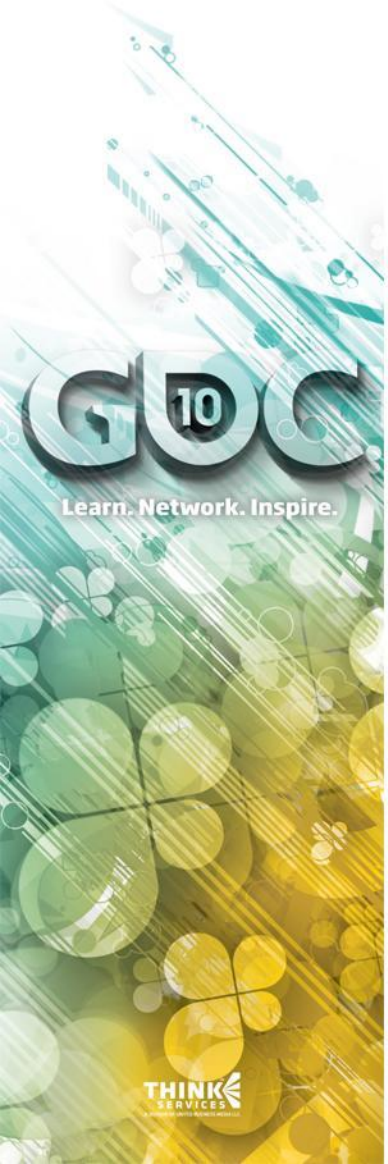
Game Developers Conference® | March 9-13, 2010 | Moscone Center | San Francisco, CA

iPhoneGames SUMMIT

Bringing UE3 to Apple's iPhone Platform

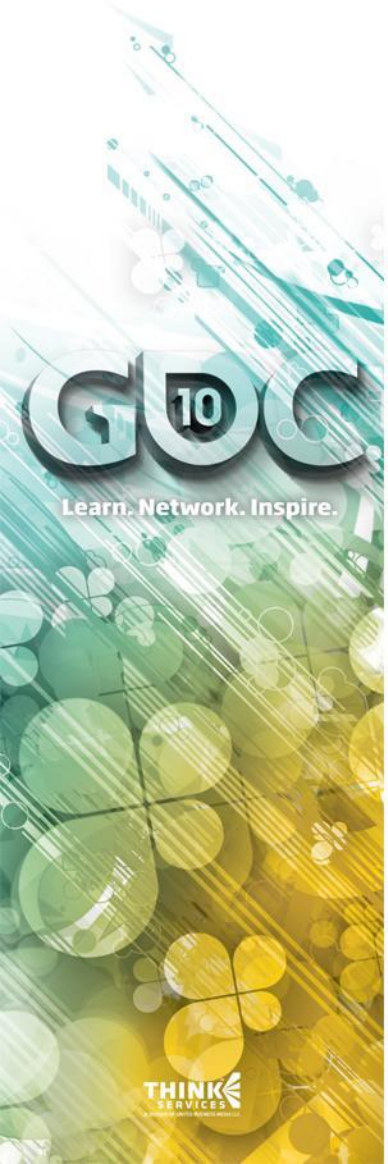
Josh Adams
Senior Console Programmer
Epic Games
josh.adams@epicgames.com



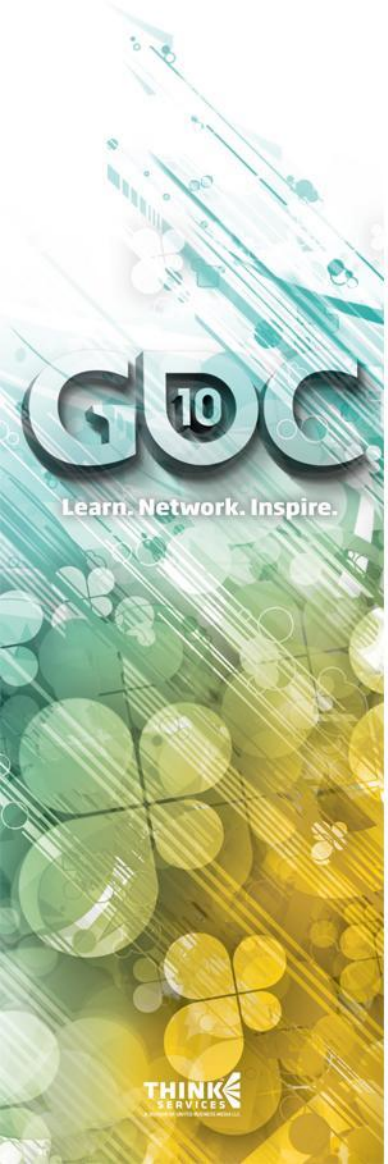
Topics

- Background
- Method of Attack
  - Bringup
  - What we kept
  - Changes we made
- Workflow Changes
- Where To Go From Here

Topics

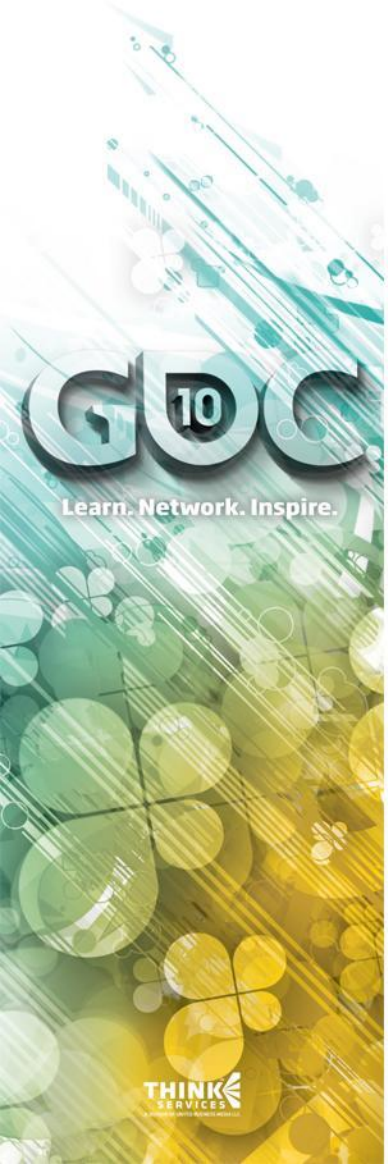


- Background
- Method of Attack
  - Bringup
  - What we kept
  - Changes we made
- Workflow Changes
- Where To Go From Here



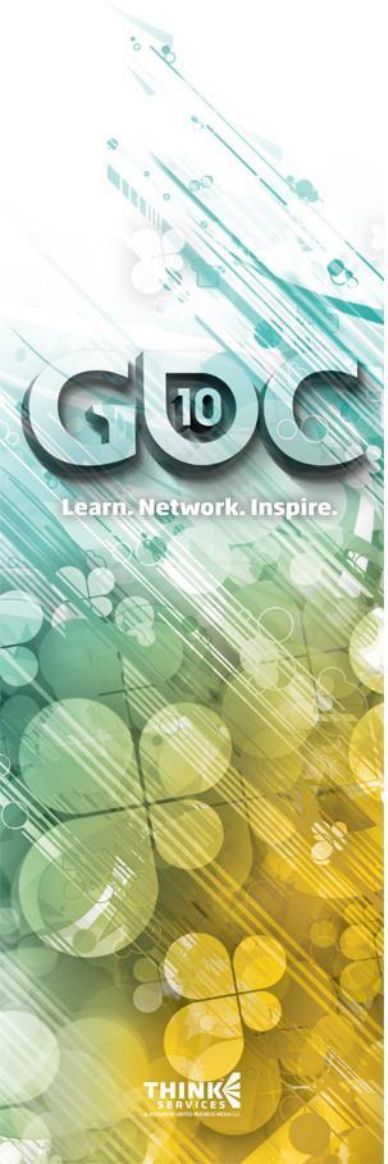
[Background]
Me

- Engine Programmer at Epic
  - 5+ years at Epic
  - 13+ years in industry
- Console focused
  - Unreal Tournament – Dreamcast
  - Unreal Engine 2 – PS2/Gamecube
  - Unreal Engine 3 – PS3
- iPhone = Console!



[Background]
You

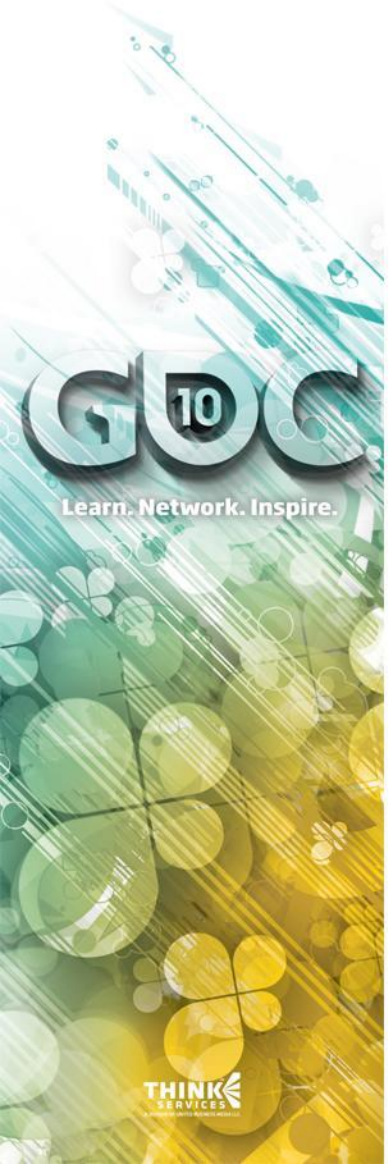
- Talk assumes some iPhone experience
  - You've compiled and run an app or two
  - Read some docs/sample code, etc
- If not, feel free to ask me after the talk!



[Background]
The Talk

- Sharing developer experiences
  - No small effort
    - Unreal Engine 3 – 2 million lines of code
    - iPhone – Fits in your pocket
  - Hope it can be of use to you!

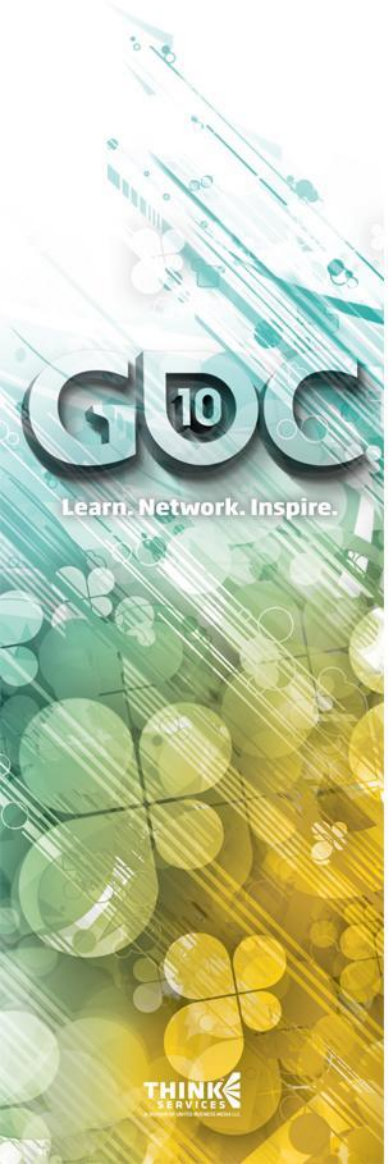
  - (Note: look for *'s – they are Gotchas!)



[Background]

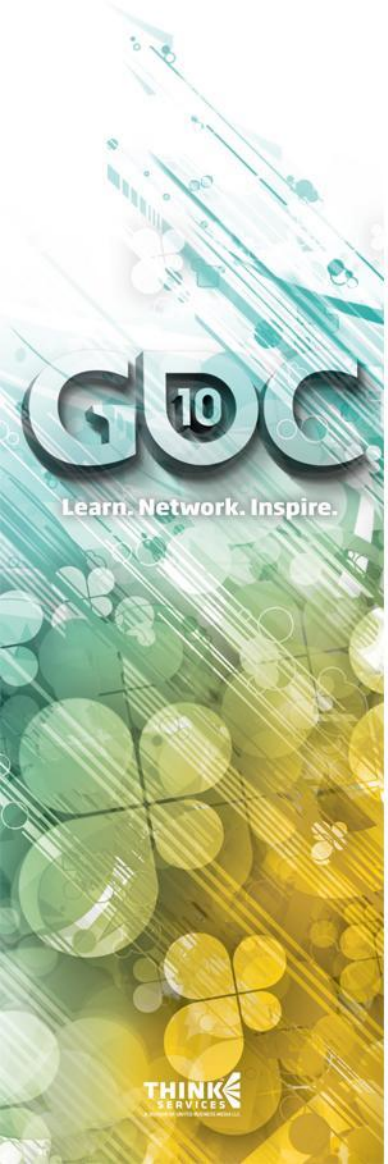
Why iPhone?

- 3GS has OpenGL ES 2.0
  - Programmable shaders
- Huge install base
  - Many are pre-3GS (for now, anyway)
- Fun, “can we do it?” project



[Background]
Unreal Engine 3

- Multiplatform
  - Shipped platforms
    - Windows, Xbox 360, PS3
    - UDK – free version of UE3
  - Unsupported platforms
    - iPhone, NVIDIA Tegra2, Linux, Mac
  - 2 layers
    - Platform independent – 90%
    - Platform specific (engine and DLLs)

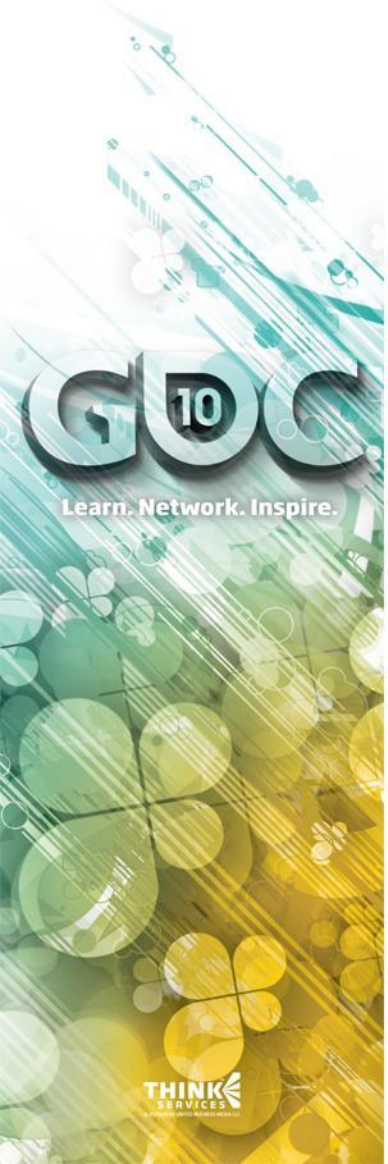


[Background]
Unreal Engine 3

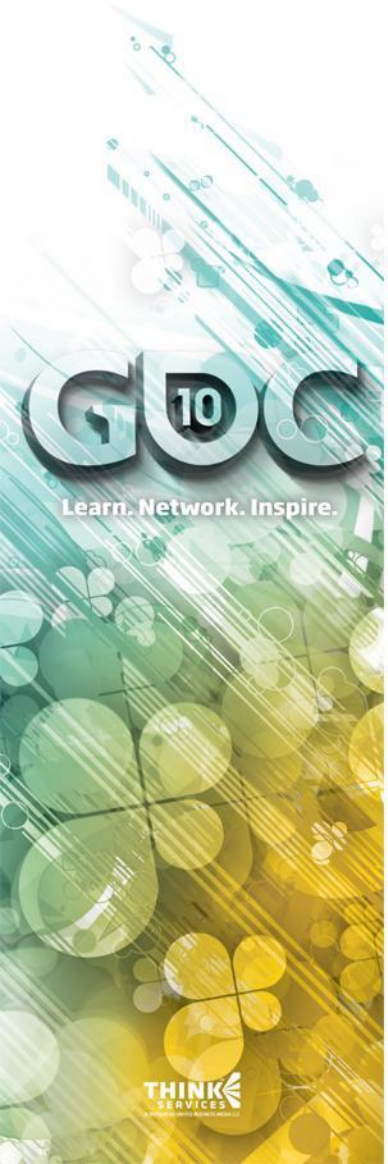
- Rendering Engine
  - Materials, streaming worlds, visibility, …
- Gameplay Engine
  - Script code, physics, AI, …
- Third party integration
  - SpeedTree, Scaleform, PhysX, Bink, …

Demo

Topics

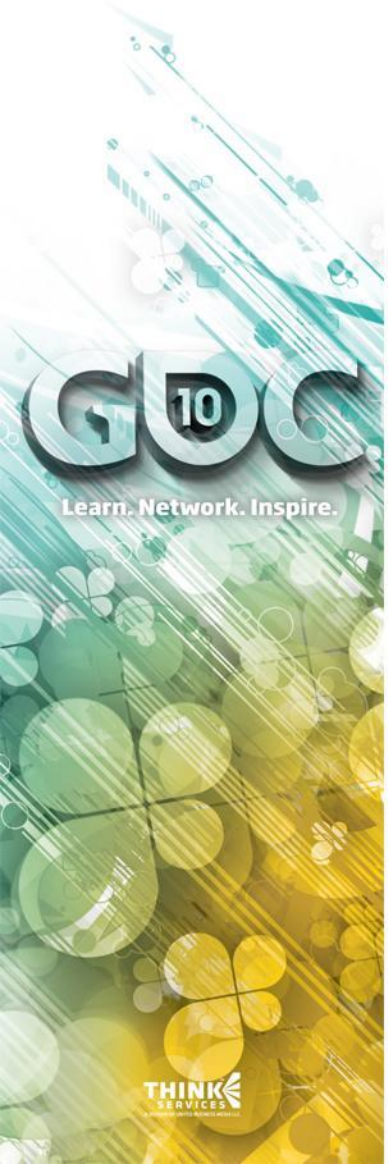


- Background
- Method of Attack
  - Bringup
    - Compiling w/ Xcode
    - Objective-C integration
  - What we kept
  - Changes we made
- Workflow Changes
- Where To Go From Here



Topics

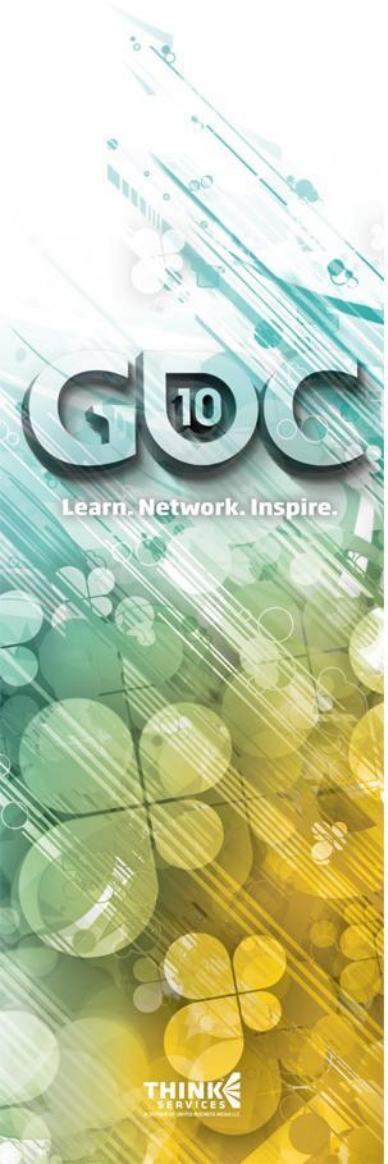
- Background
- Method of Attack
  - Bringup
    - Compiling w/ Xcode
    - Objective-C integration
  - What we kept
  - Changes we made
- Workflow Changes
- Where To Go From Here



[Bringup – Compiling]

Problem

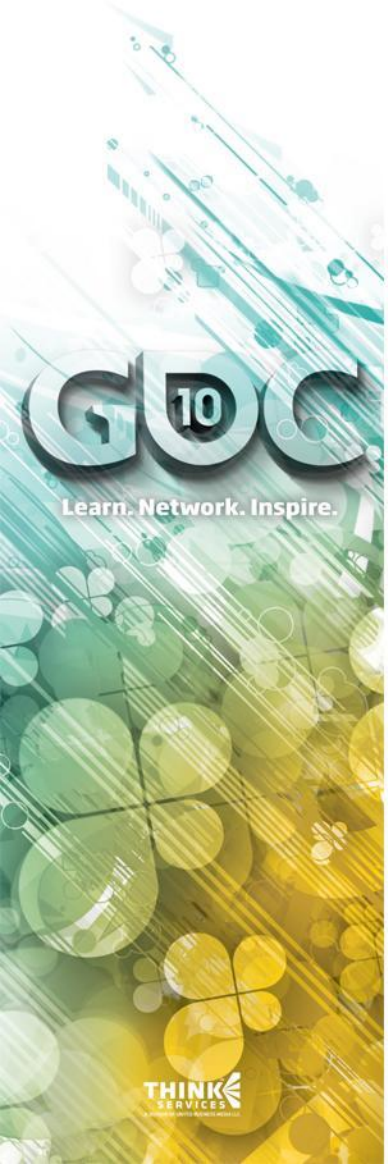
- iPhone is Mac-based
  - UE3 uses a Visual Studio solution
- UE3 is cross-platform
  - How to fill in the iPhone-holes?



[Bringup – Compiling]

Xcode project

- Used OpenGL template project
- Mimics Visual Studio project
  - Have to keep in sync, though
- Only game subsystems
  - No editor or other Windows-only stuff



[Bringup – Compiling]

Xcode project

- Copied over preprocessor defines
  - User defined variables -> awesome!
  - **CONFIG_DEFINES:** `-DFINAL_RELEASE=1`
  - **TARGET_DEFINES:** `-DGAMENAME=UTGAME`
  - **GLOBAL_DEFINES:** `-DIPHONE=1`
  - **Other C++ Flags:**
    - `$(OTHER_CFLAGS) $(CONFIG_DEFINES) $(GLOBAL_DEFINES) $(TARGET_DEFINES)`
    - (same for all configs/targets)

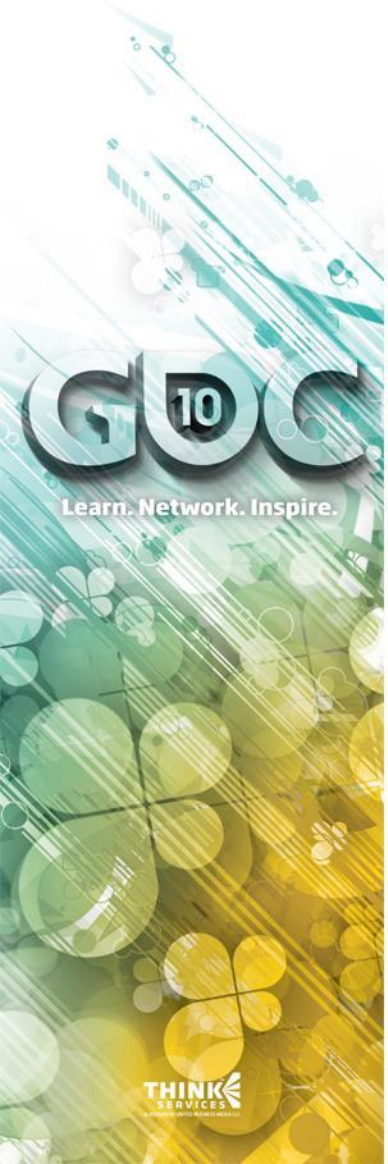
[Bringup – Compiling]

Missing Pieces

- Added new UE3 platform
  - Header to map types

```
typedef uint64_t QWORD; // 64-bit unsigned
```

  - Implement interface sub-classes
    - UE3 has base classes for major interfaces
      - Memory allocator
      - File manager
      - Threading routines
  - IPhoneTools.dll

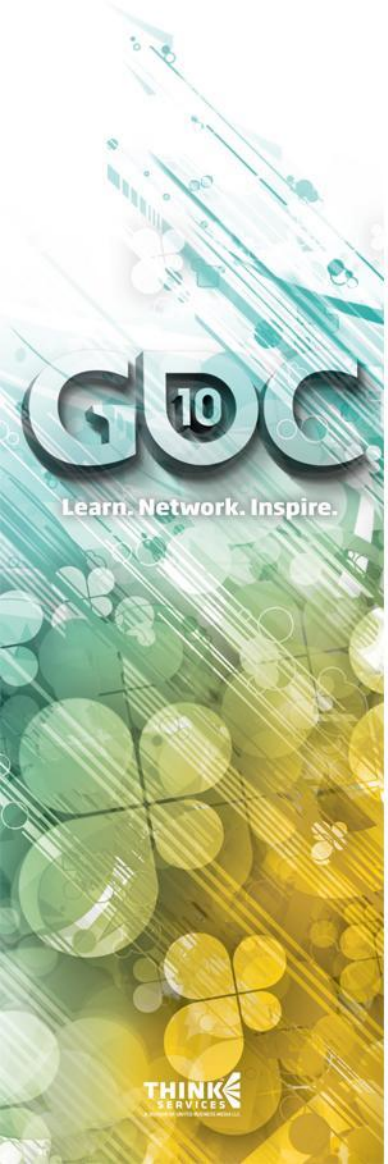


[Bringup – Compiling]

Missing Pieces

- One conceptual piece at a time
  - Core, Engine, Net, GameFramework, UT3
- Had done gcc for PS3, helped, but:
  - *wchar_t is 4 bytes!*
    - Had to handle UE3 Unicode 2-byte chars on disk vs. libc functions needing 4-bytes

```
typedef uint16_t UNICHAR;// on disk
typedef wchar_t TCHAR; // in memory
#define TCHAR_IS_4_BYTES 1
```



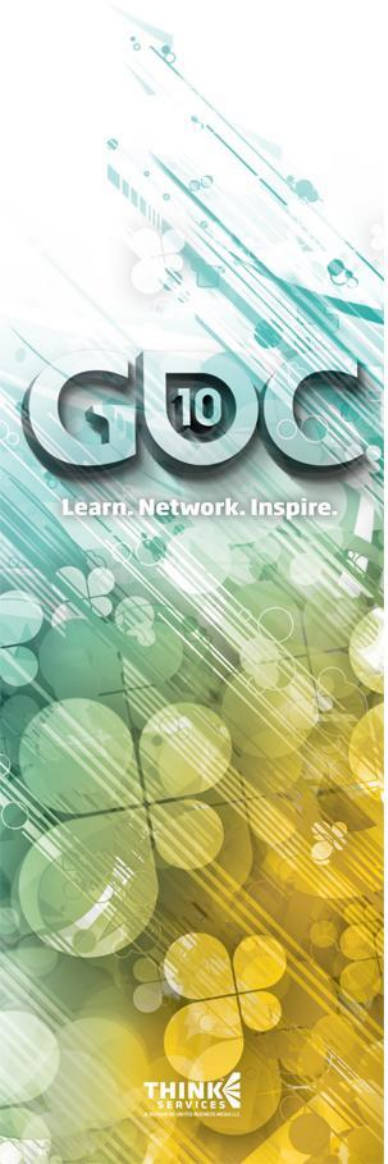
[Bringup – Compiling]

Vector intrinsics

- High level intrinsic “language”
  - Each platform defines type and functions
  - Implemented Neon intrinsics
  - *Xcode 3.2 Internal Compiler Error*

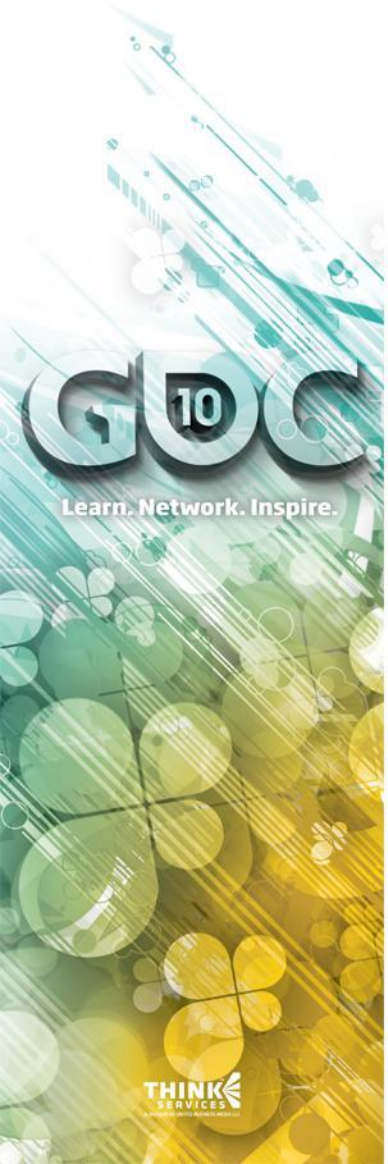
```
#define VectorRegister float32x4_t

FORCEINLINE VectorRegister VectorAdd( VectorRegister
Vec1, VectorRegister Vec2 )
{
    return vaddq_f32( Vec1, Vec2 );
}
```



Topics

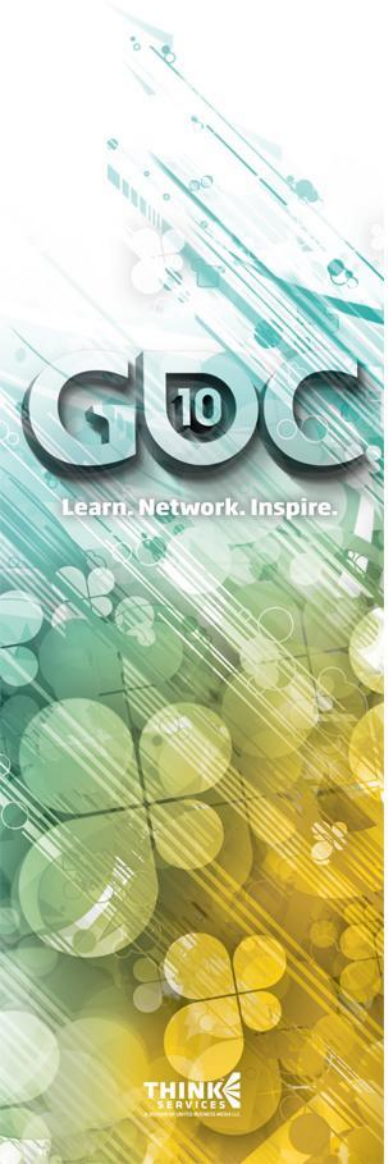
- Background
- Method of Attack
  - Bringup
    - Compiling w/ Xcode
    - Objective-C integration
  - What we kept
  - Changes we made
- Workflow Changes
- Where To Go From Here



[Bringup – Obj-C]

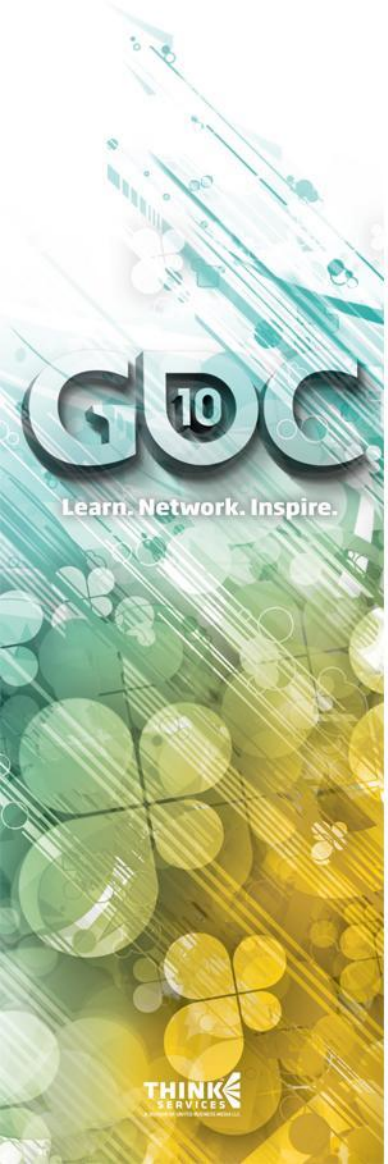
Problem

- UE3 is all C++
  - What about this Objective-C stuff?
    - Callbacks
    - Animation system



[Bringup – Obj-C]
iPhone<–> UE3 “glue”

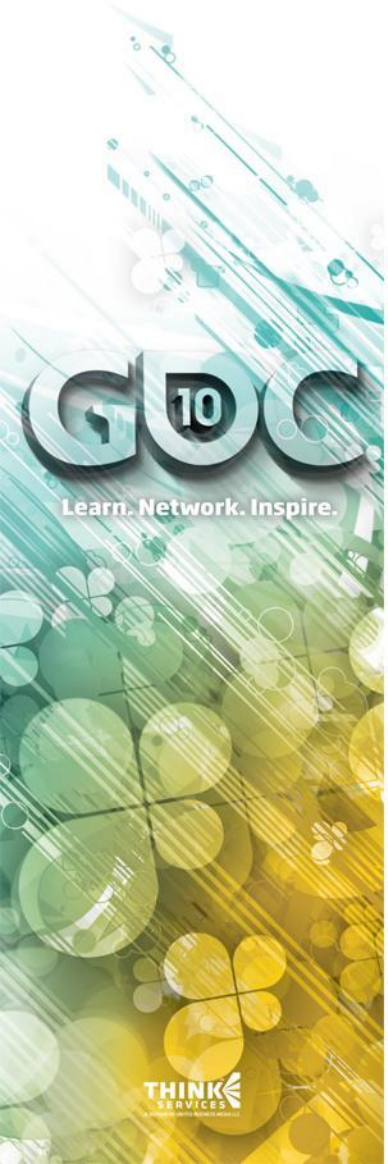
- Objective-C integration
  - Whole game doesn’t need to be Obj-C!
    - Total of 4 .mm files (could be less)
  - Some is still needed, i.e.:
    - Startup (UI, GL init)
    - API wrapper functions (Called from C++)
      - presentRenderBuffer
      - NSSearchPathForDirectoriesInDomains
    - Input/tilt callbacks from OS



[Bringup – Obj-C]

Startup process

- `applicationDidFinishLaunching`
  - Creates main game thread
    - Engine is now “independent” entity
    - No Animation, CADisplayLink or Timers
  - Enables accelerometer
    - Sets app as delegate for callbacks
  - Show splash screen UI layer
    - Layer on top of EAGL layer
    - Shows the Default.png



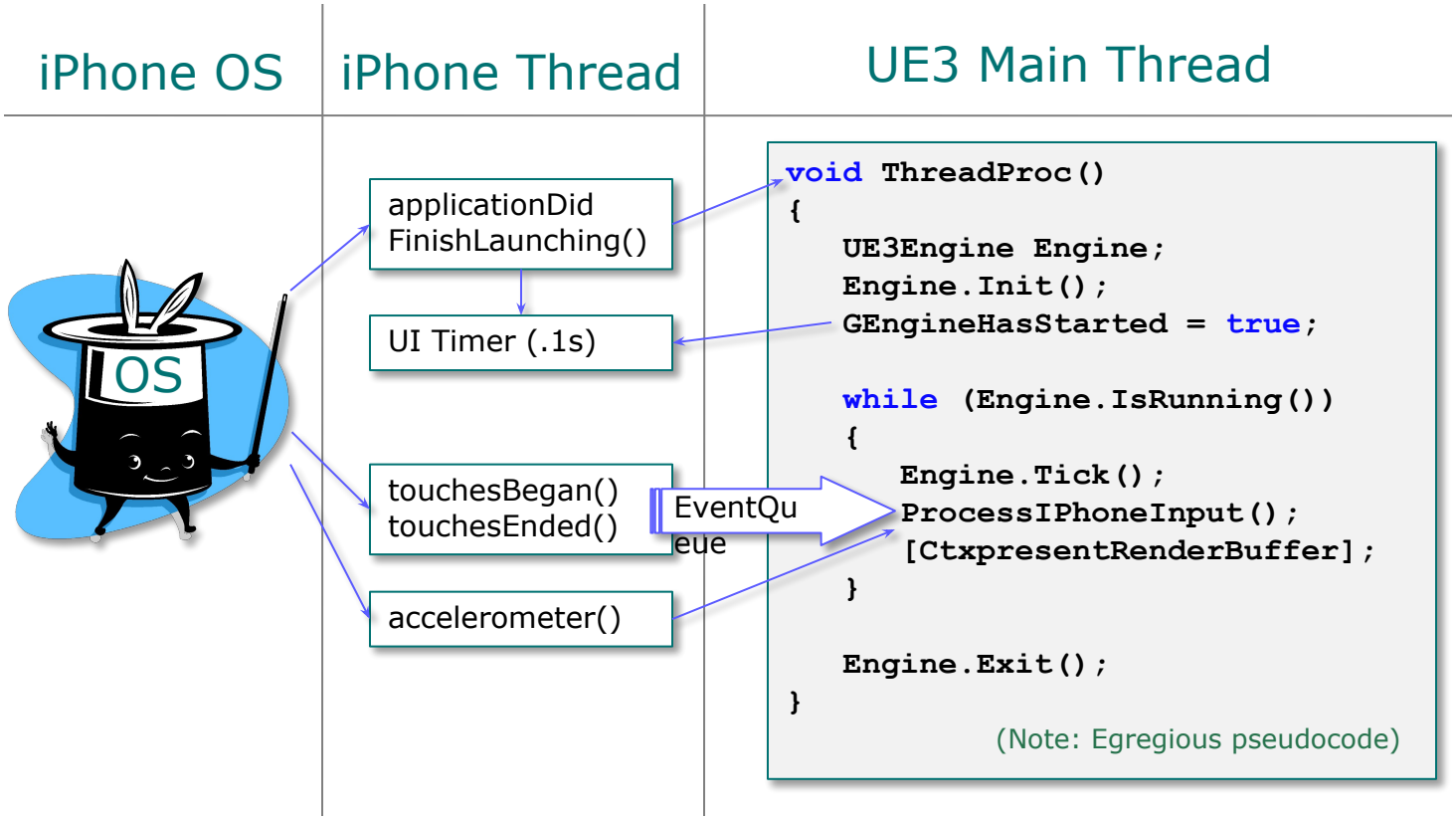
[Bringup – Obj-C]

Startup process

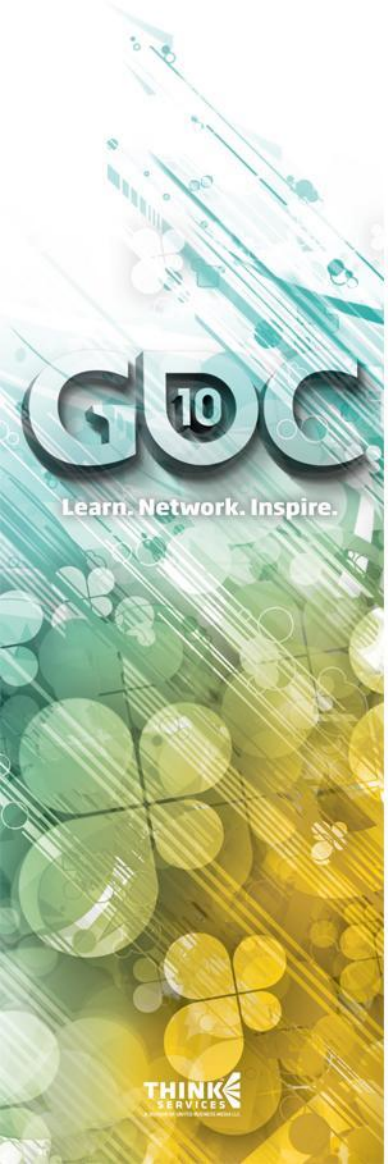
- `applicationDidFinishLaunching`
  - Start 'hide splash' timer
    - Timer function called every 100 ms
    - Looks for main thread 'has booted' flag
    - When flag is set:
      - Hides UI layer
      - Kills timer
  - Returns to OS quickly
    - OS watchdog kills app if too slow (15 sec)

[Bringup – Obj-C]

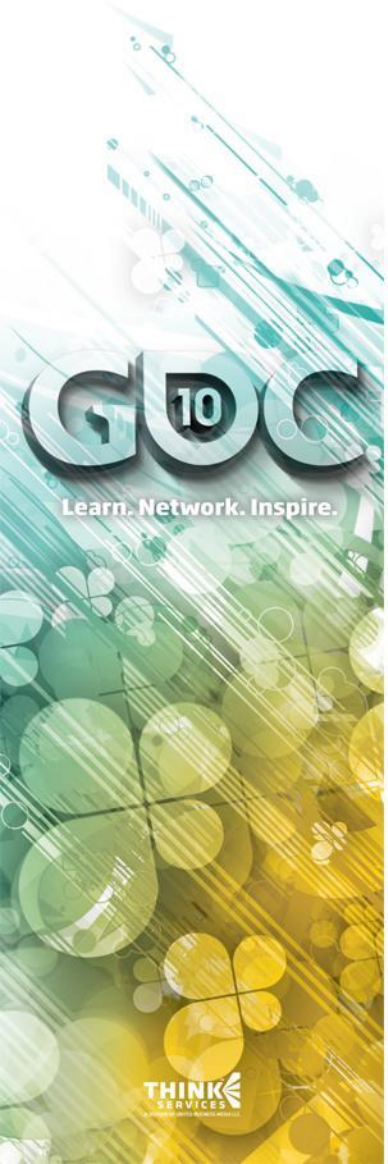
Thread structure



Topics

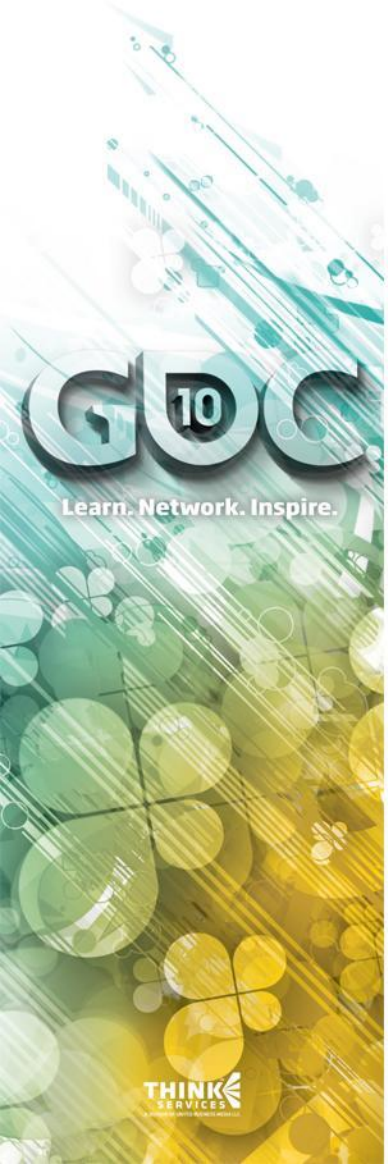


- Background
- Method of Attack
  - Bringup
  - What we kept
    - Code
    - Tools
  - Changes we made
- Workflow Changes
- Where To Go From Here



Topics

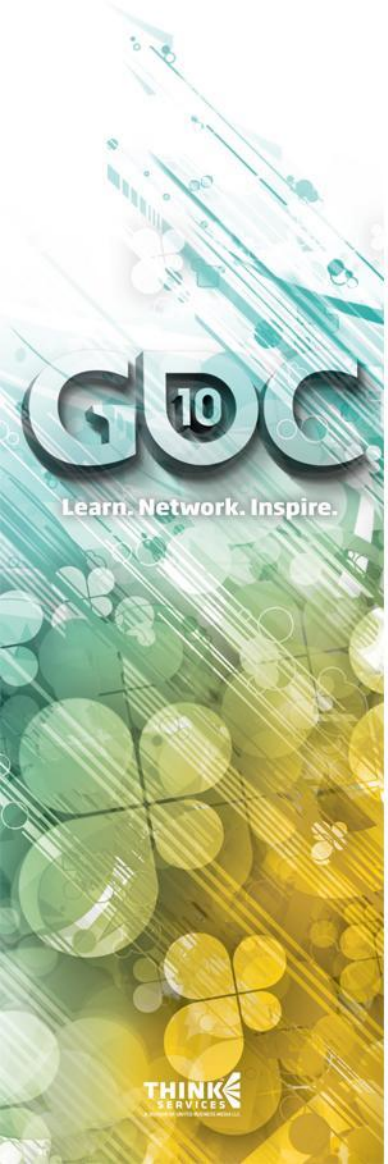
- Background
- Method of Attack
  - Bringup
  - What we kept
    - Code
    - Tools
  - Changes we made
- Workflow Changes
- Where To Go From Here



[What we kept – Code]

General

- Almost everything!
  - File formats
  - Math routines
  - Collision
  - Gameplay
  - etc



[What we kept – Code]

General

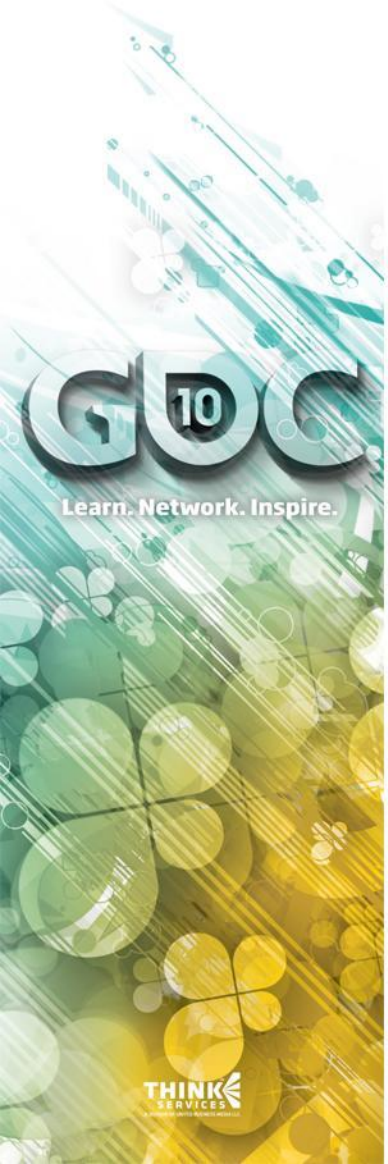
- Script code
  - Same compiled code as all platforms
  - Some runtime platform checks, though
- Wrapper code from Linux
  - Types (INT, QWORD, etc)
  - Threading (BSD sockets)
  - FileManager (fopen, etc)

[What we kept – Code]

File management

- File writing redirection from Win32
  - Win32:
    - Program Files security restricts writes
  - iPhone:
    - Can't write to signed .app dir
  - Only write to Documents directory
  - Try to read from Documents dir, fallback to install dir

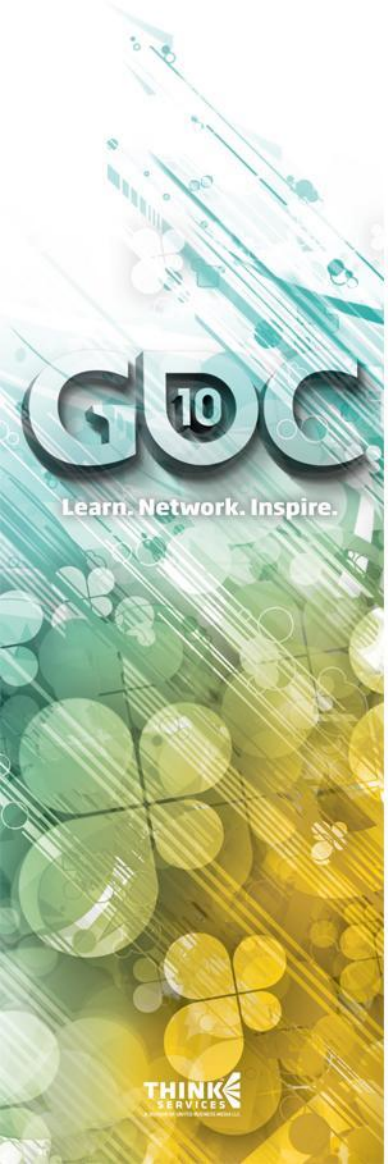
```
// use the API to retrieve the Document dir
NSString* Dir = [NSSearchPathForDirectoriesInDomains
    (NSDocumentDirectory, NSUserDomainMask, YES)
    objectAtIndex: 0];
```



[What we kept – Code]

RHI

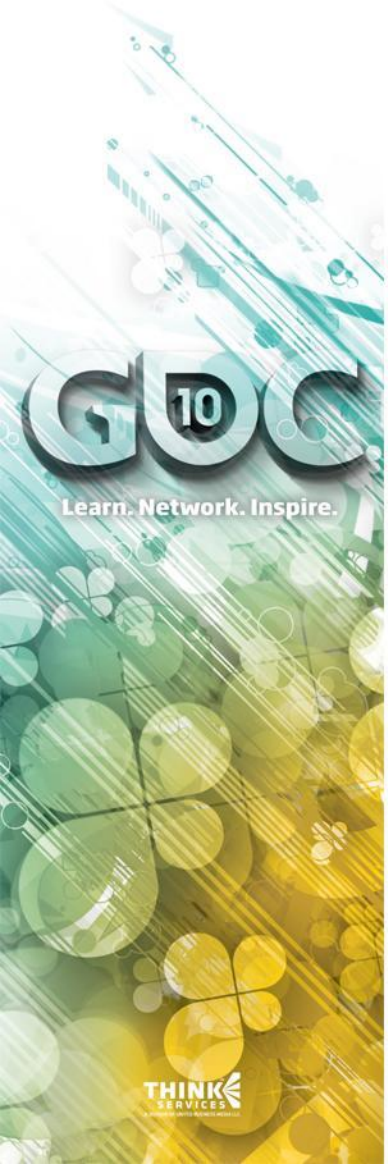
- RHI (RenderHardwareInterface)
  - Thin layer between rendering thread and platform's API
    - RHICreateVertexBuffer()
    - RHISetBlendState()
    - RHIDrawIndexedPrimitive()
  - Hides D3D, GL, etc from engine



[What we kept – Code]

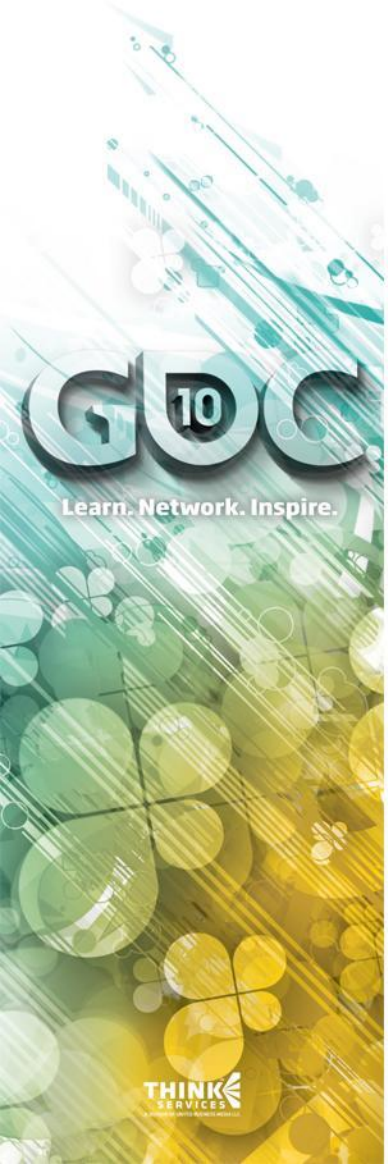
Lighting

- LightEnvironments
  - Gathers static and/or dynamic lights
  - Many lights into one or two lights
    - Directional, Spherical Harmonic, Ambient
  - Updated ‘infrequently’
  - Great for iPhone
    - Rendering cost of 1 light
    - Visually, many lights from artists/gameplay



Topics

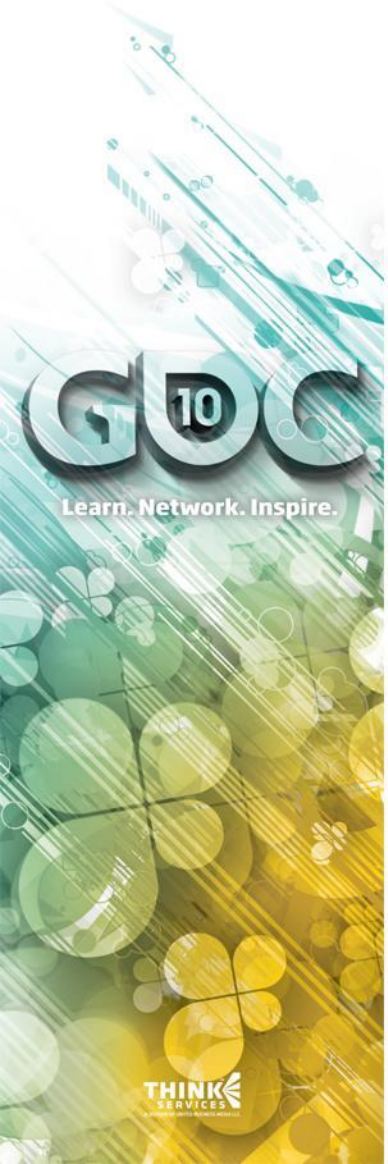
- Background
- Method of Attack
  - Bringup
  - What we kept
    - Code
    - Tools
  - Changes we made
- Workflow Changes
- Where To Go From Here



[What we kept – Tools]

Content

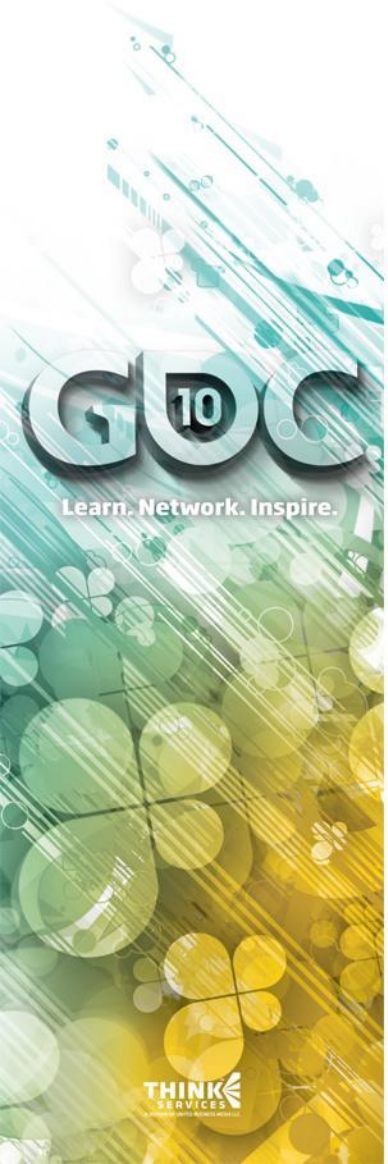
- Editor – critical!
  - Artists build levels the same way
    - Mesh importing
    - Material creation
    - Gameplay placements
- Cooker
  - Same pipeline of moving PC-format assets to consoles



[What we kept – Tools]

Helpers

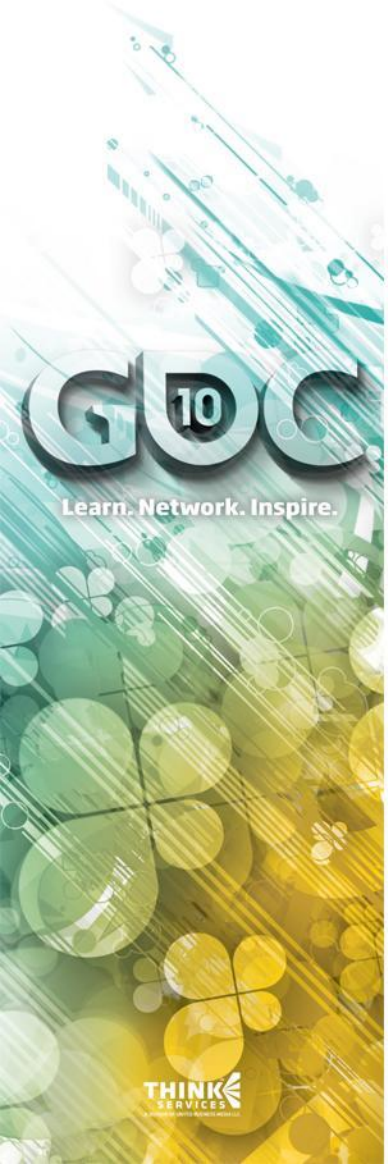
- UnrealFrontend
  - Controls UE3 games
    - Launching game
    - Cooking content
    - Compiling script
    - Syncing files
- UnrealConsole
  - Captures output from all platforms
  - Reports crashes



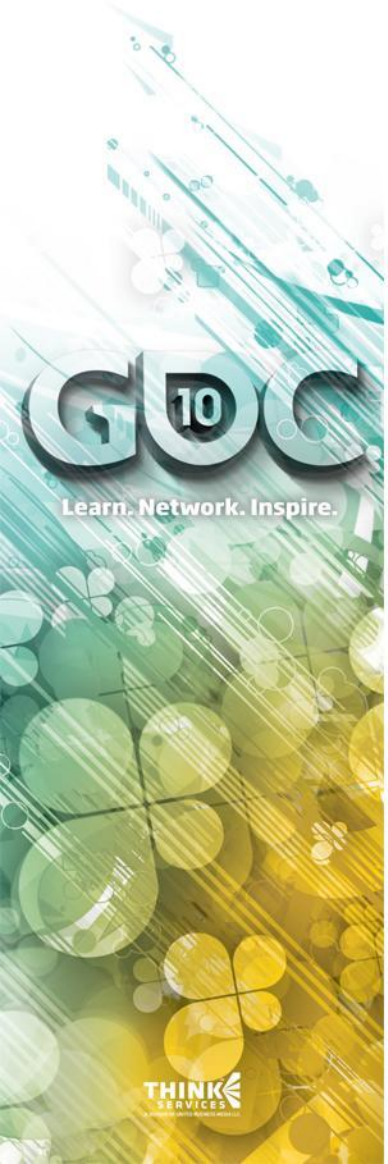
Topics

- Background
- Method of Attack
  - Bringup
  - What we kept
  - Changes we made
    - Input
    - Renderer
    - Tools
- Workflow Changes
- Where To Go From Here

Topics



- Background
- Method of Attack
  - Bringup
  - What we kept
  - Changes we made
    - Input
    - Renderer
    - Tools
- Workflow Changes
- Where To Go From Here



[Changes we made – Input]

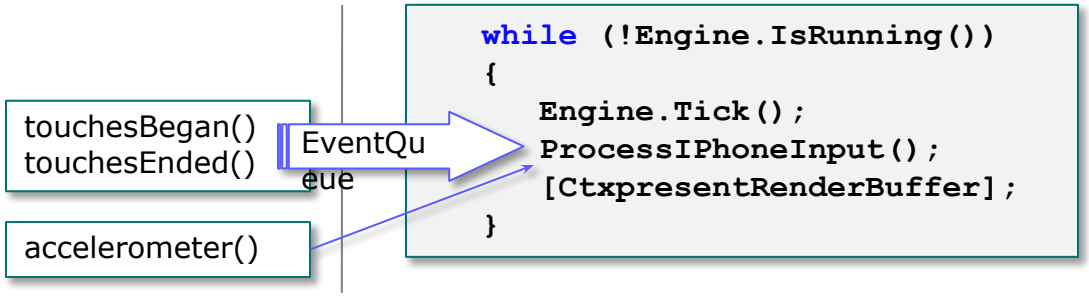
Problem

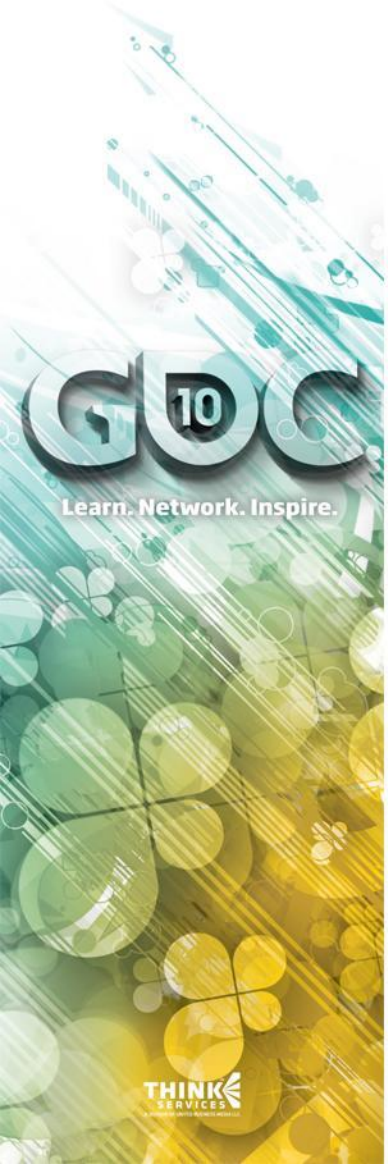
- No keyboard/mouse/controller
  - How to use touch events effectively?

[Changes we made – Input]

Input "glue"

- Touch callbacks in Objective-C
  - Pushed to game thread in a queue
    - Frequency higher than game thread
  - Game thread pulls off once a frame
    - Process all outstanding input, in order

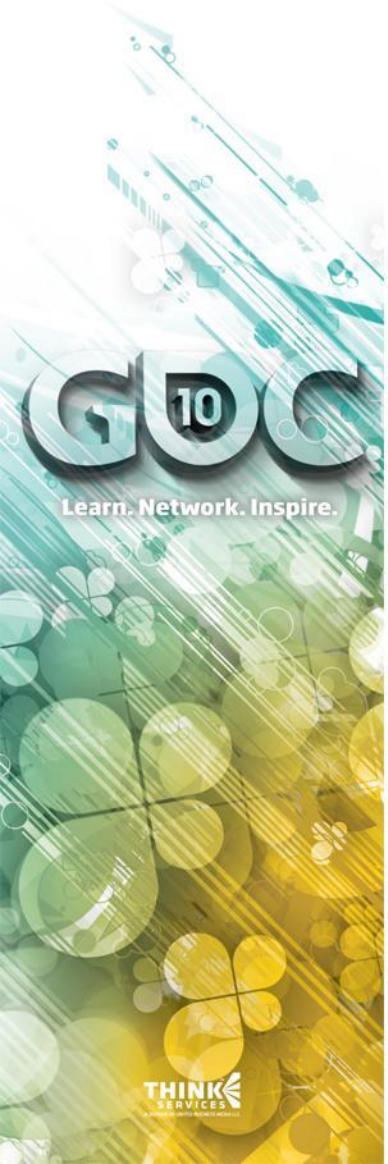




[Changes we made – Input]

Input "glue"

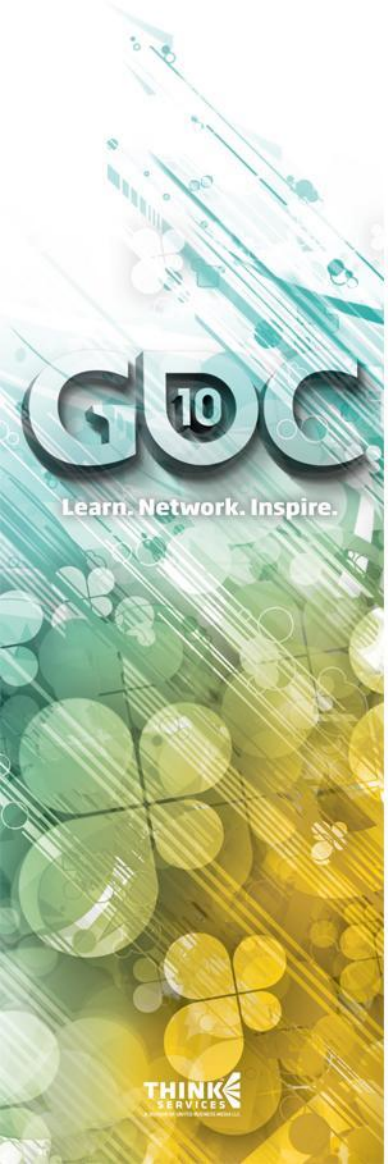
- Game thread processing
  - Tracks each touch over time
    - Looks for closest touch from last frame
    - Close finger drags can confuse it
    - *Drag off edge and back breaks tracking*
  - Turns into Press, Hold, Release events
    - Standard UE3 input messages



[Changes we made – Input]

Input "glue"

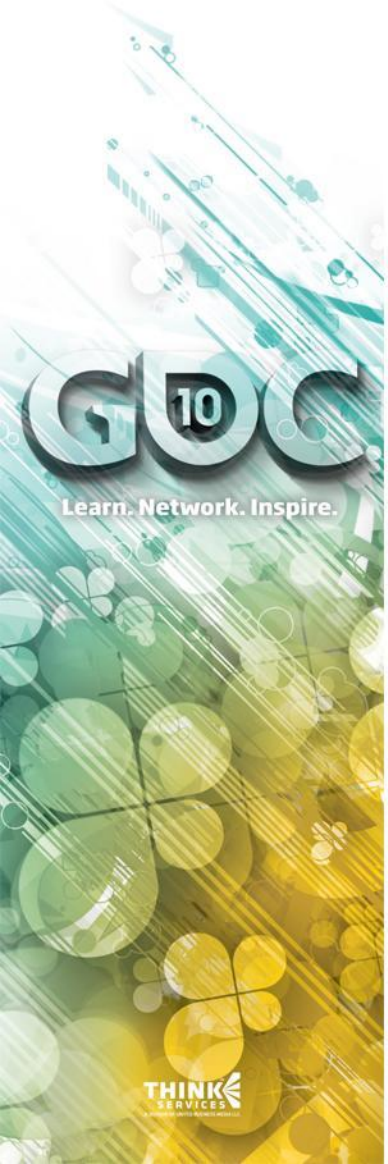
- Accelerometer (tilt) also from API
- Not a queue
  - Updates game thread with latest values
- Tried using magnetometer
  - Figured it could enhance turning info
  - Unusable results
  - Also, quite CPU-intensive



[Changes we made – Input]

Input feedback

- Input Zones
  - Mimics a gamepad button or stick
    - Hooked up to UE3 Input Binding system
  - Data driven touch screen zones
  - "Button" types
    - Tracks clicks/drags
    - Imagine Windows Button controls
  - "Stick" types
    - Sends floating point X/Y axes to game



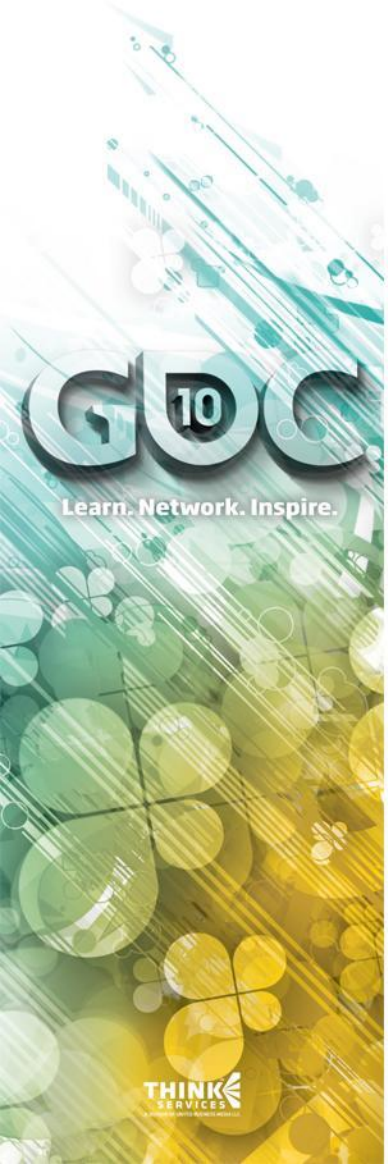
[Changes we made – Input]

Input feedback

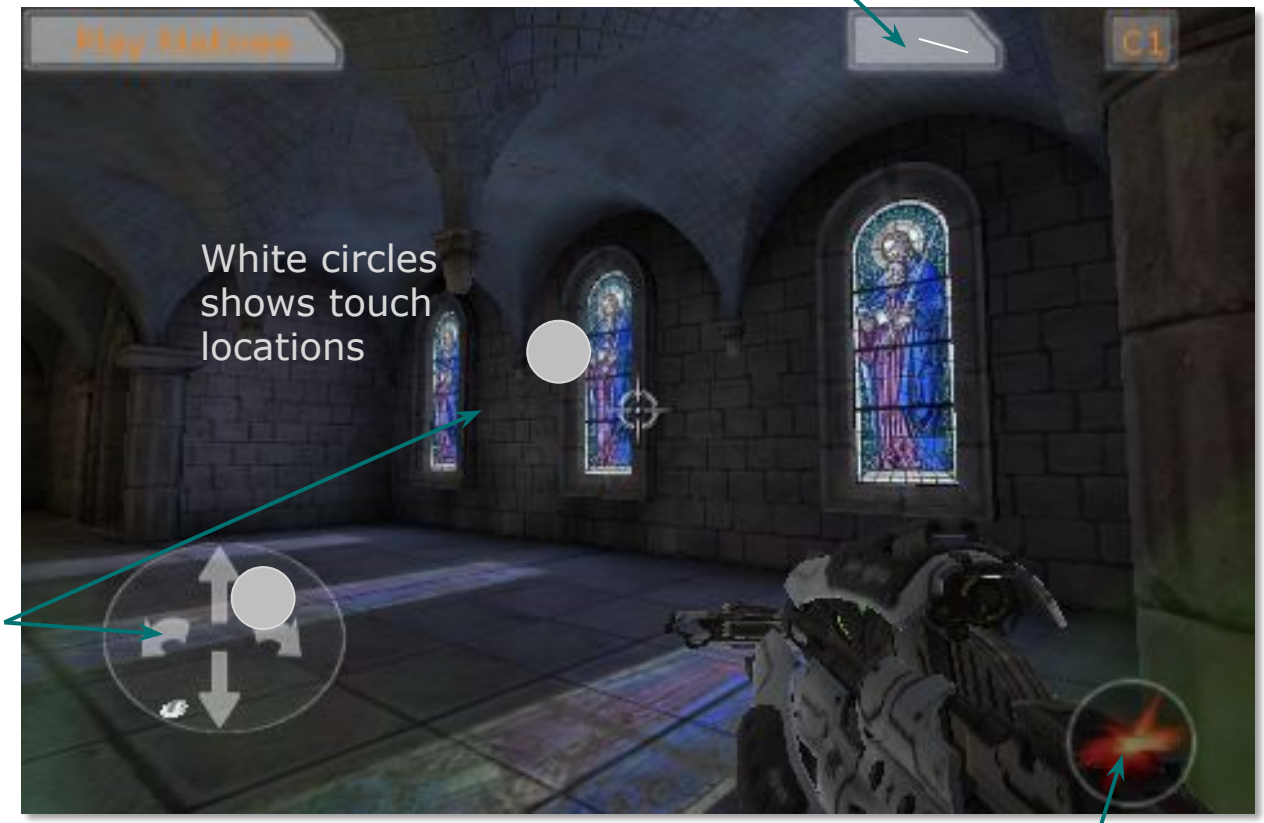
- Input Zones (cont'd)
  - "Tilt" types (hidden)
    - Sends floating point X/Y axes to game
    - Important to have a calibrate option
  - MobileHUD base HUD class
    - Draws zones and state info
      - Highlighted while pressed
      - Stick types show current location
    - Input handling code updates Zone state
    - Can draw on PC for testing

[Changes we made – Input]

How it looks

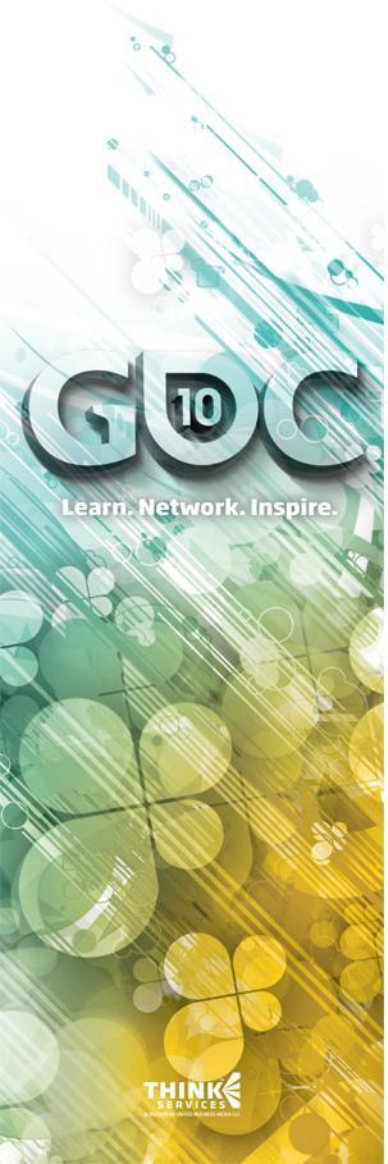


Tilt zone (shows current tilt)



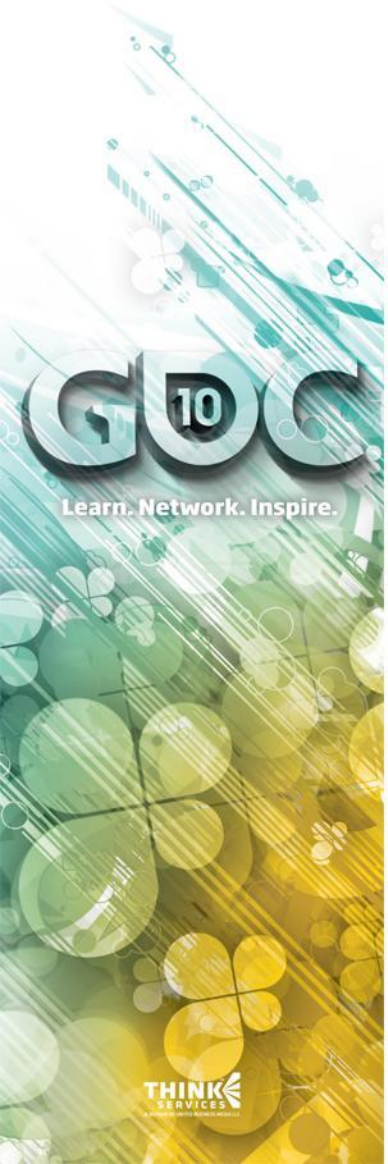
Stick zones (whole screen draggable)

Button zone



Topics

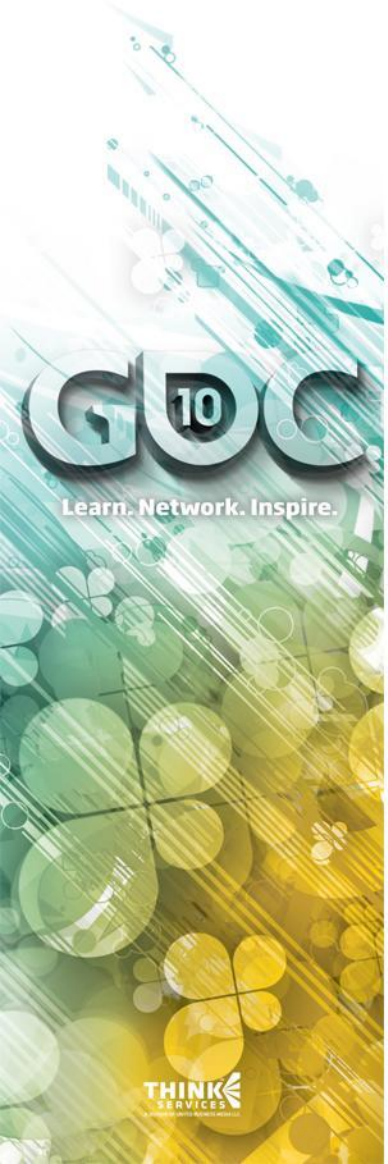
- Background
- Method of Attack
  - Bringup
  - What we kept
  - Changes we made
    - Input
    - Renderer
    - Tools
- Workflow Changes
- Where To Go From Here



[Changes we made – Renderer]

Problem

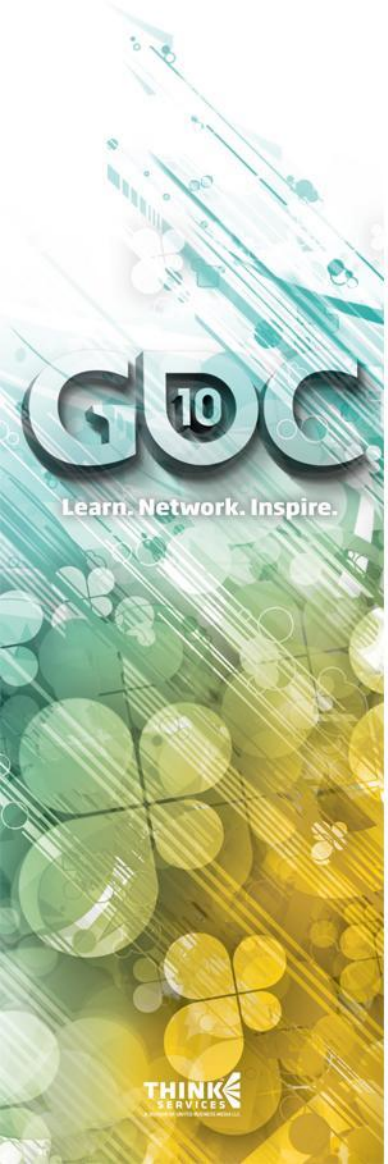
- Mobile GPU
  - Less powerful
  - OpenGLES
- UE3 games have thousands of shaders



[Changes we made – Renderer]

OpenGLES

- Added OpenGLES RHI
  - Started with existing OpenGL RHI
- Rewrote much for ES
  - New shader system
  - Added PVRTC support
  - GL driver CPU overhead noticeable in perf
    - State caching, reduce draw calls
  - *16-bit indices*



[Changes we made – Renderer]

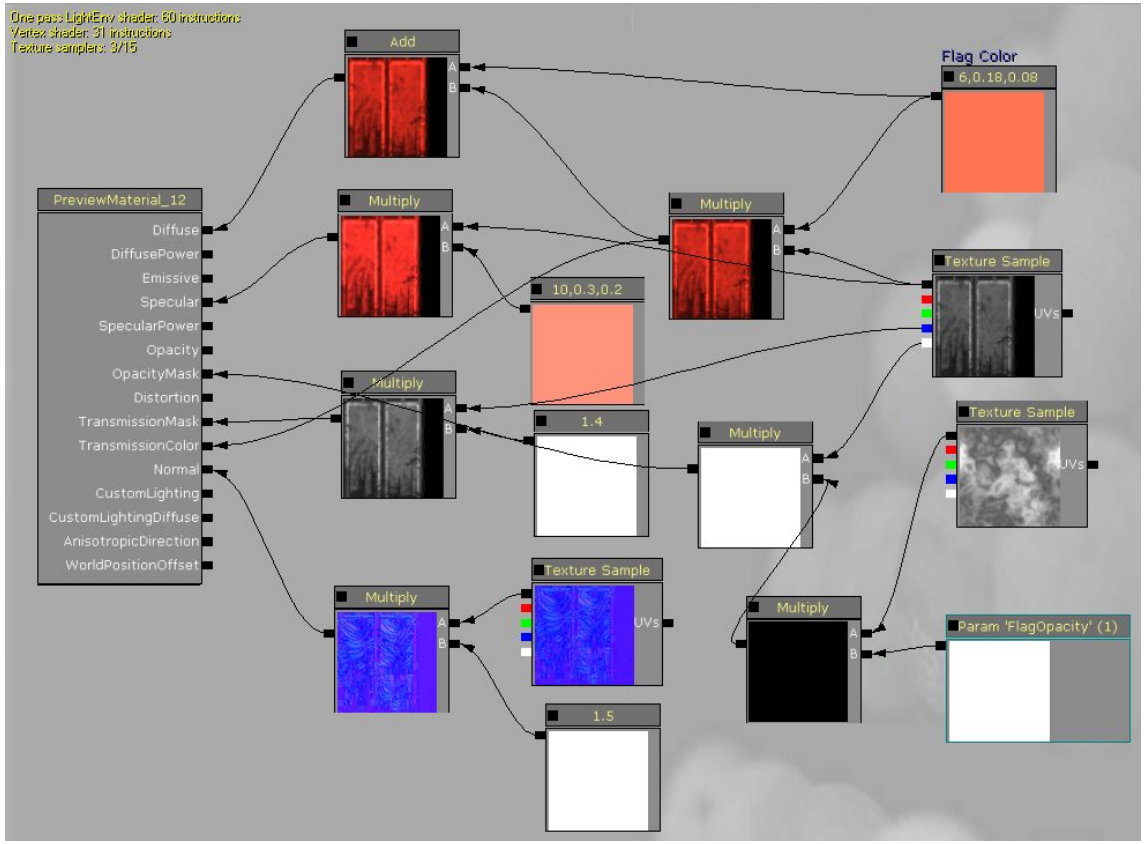
Shaders

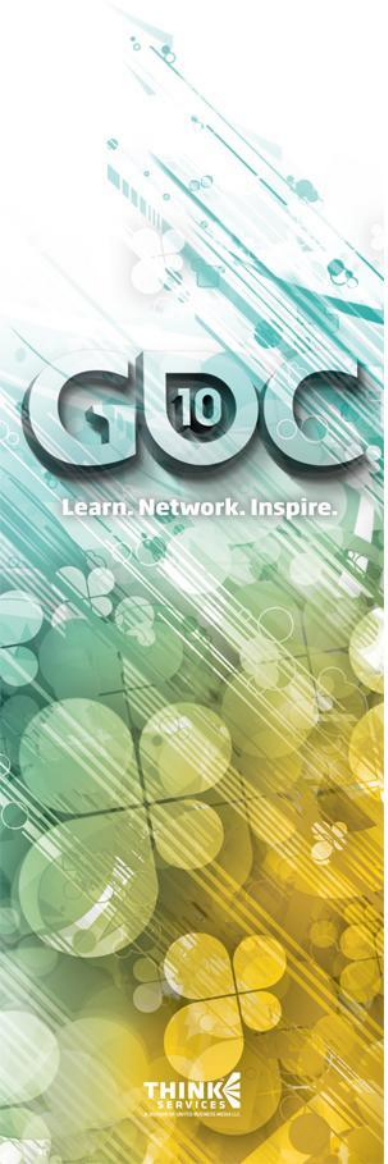
- UE3 = thousands of shaders
  - Several (5-20) shaders per artist material
- iPhone = no offline compiling
- Something has to change!

[Changes we made – Renderer]

Shaders

- Example UE3 material:

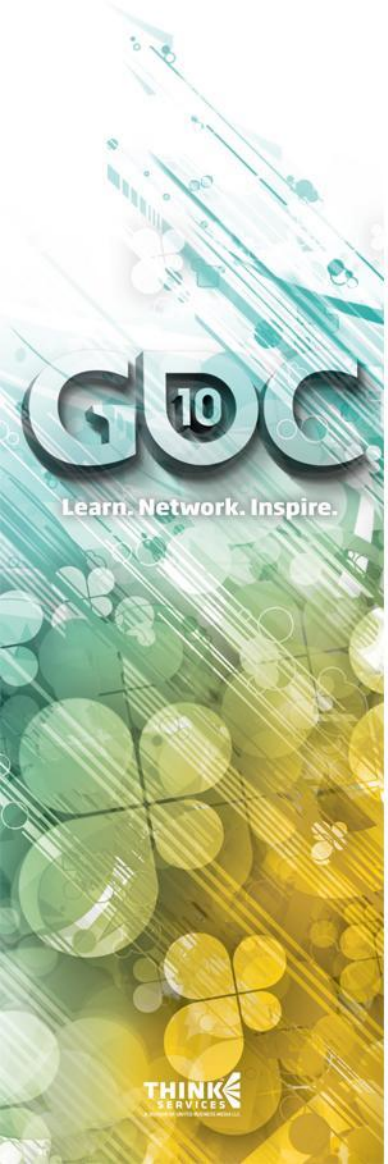




[Changes we made – Renderer]

Shaders

- Handwritten shaders
  - Based on primitive/lighting
    - Static mesh with texture lightmap
    - Static mesh with vertex lightmap
    - Static mesh unlit
    - Particle
    - Skeletal mesh with lighting
    - …
  - But, needs to look like original material!



[Changes we made – Renderer]

Shader simplification

- Material Flattening
  - Materials are auto-flattened to a texture
  - Can override auto-texture with hand-painted version
  - Editor/PC game can emulate with UE3 materials that mimic ES shaders

[Changes we made – Renderer]

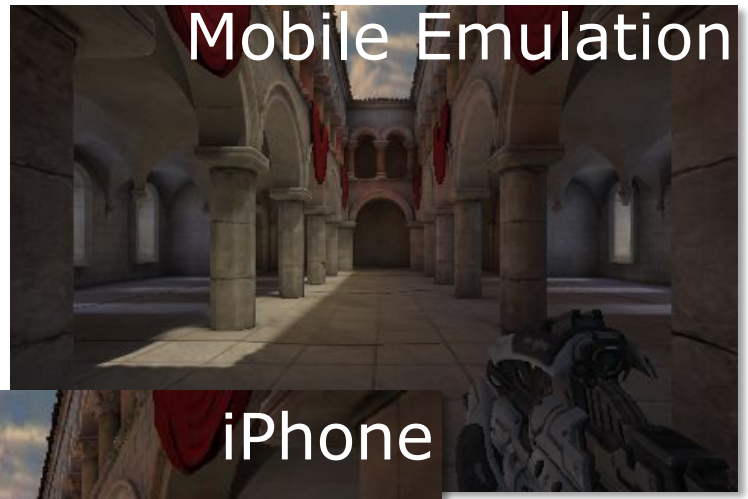
Shader simplification

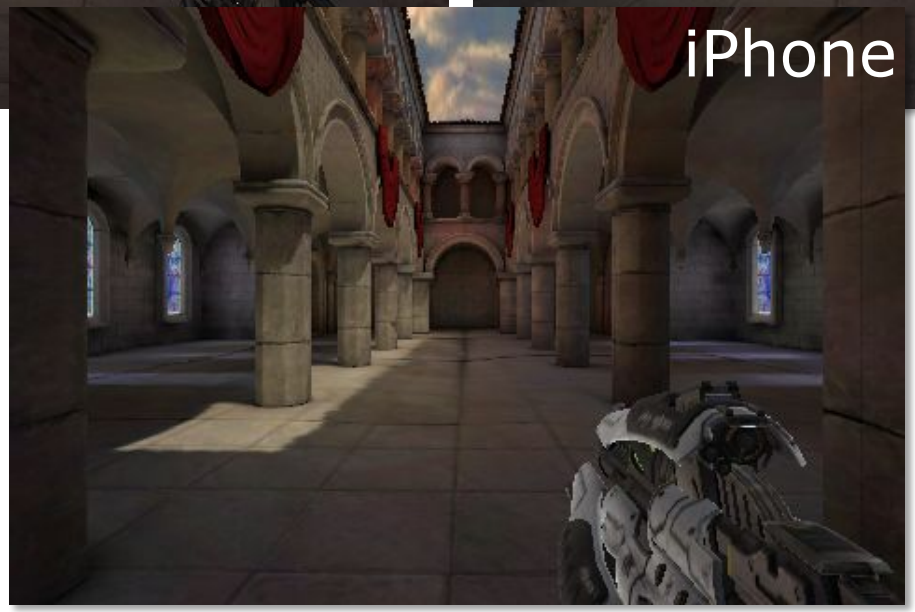
UE3 material, 60 instructions

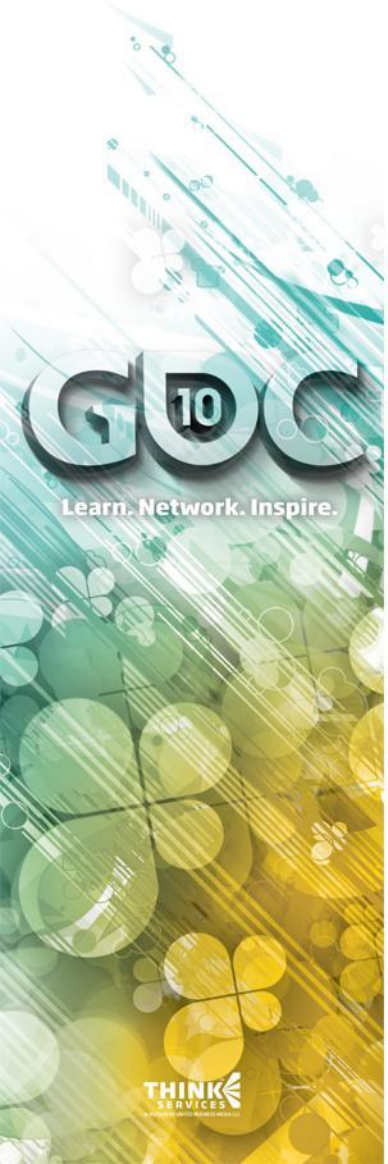
Single texture, few instructions

[Changes we made – Renderer]

Shader simplification



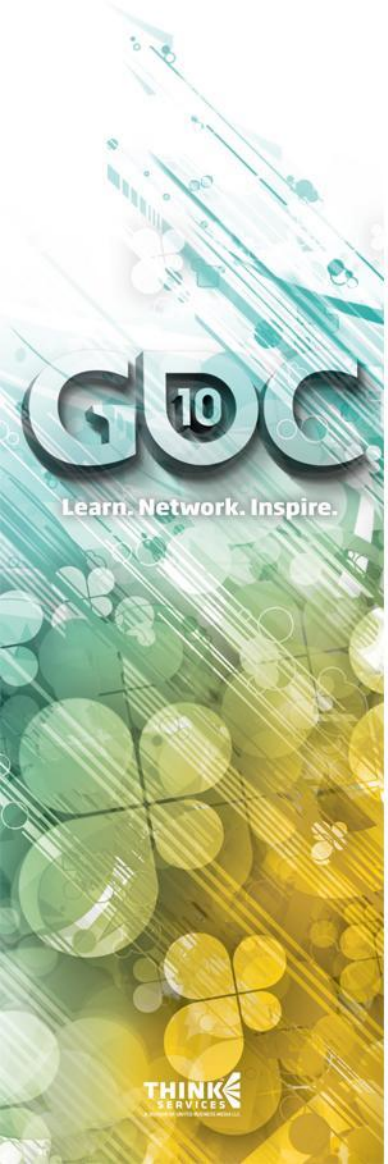




[Changes we made – Renderer]

Shader simplification

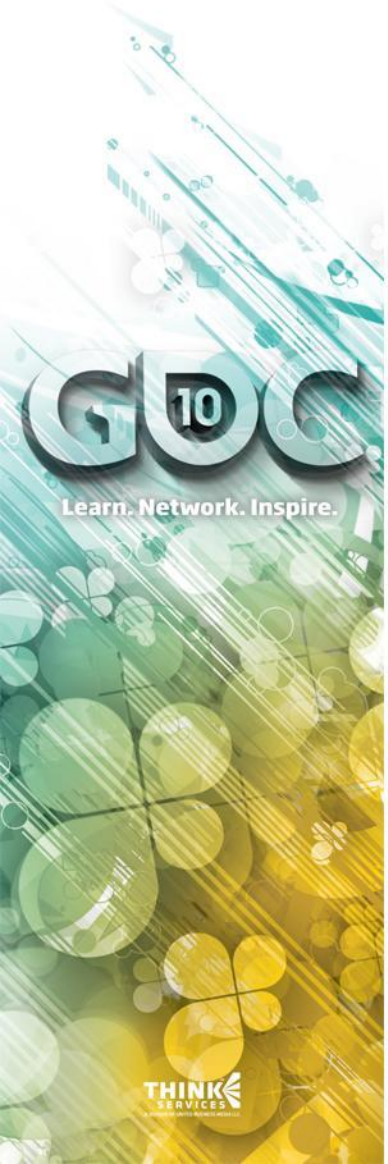
- Benefits
  - Allows for normal art pipeline
  - Fast runtime on mobile devices
  - Only a few shaders to compile
  - Usually fewer textures to load
    - One per material instead of N
- Drawback
  - Can’t share textures between materials



[Changes we made – Renderer]

Textures

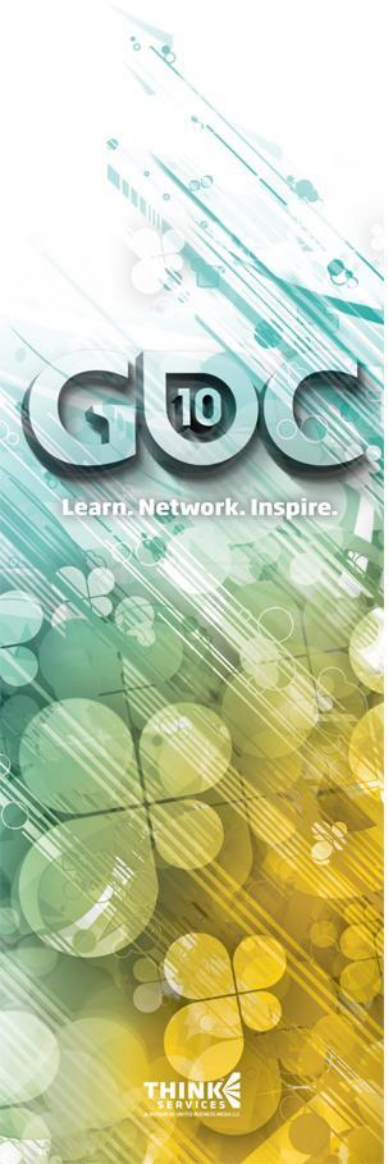
- UE3 = DXT textures
  - ~98% of textures are DXT
- iPhone = PVRTC
- Something has to change!



[Changes we made – Renderer]

Textures

- Offline conversion
  - DXT1 -> PVRTC2 (2 bits per texel)
  - DXT3,5 -> PVRTC4 (4 bits per texel)
- Cache converted mips w/ source
  - *PVRTC conversion is slow*
- ES2 RHI remaps format
  - Engine textures marked as DXT
  - RHI treats DXT as PVR, under the hood

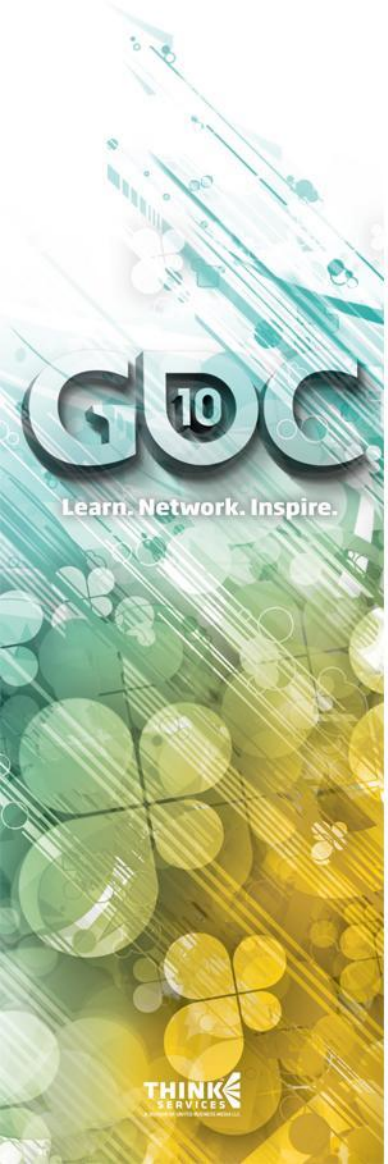


[Changes we made – Renderer]

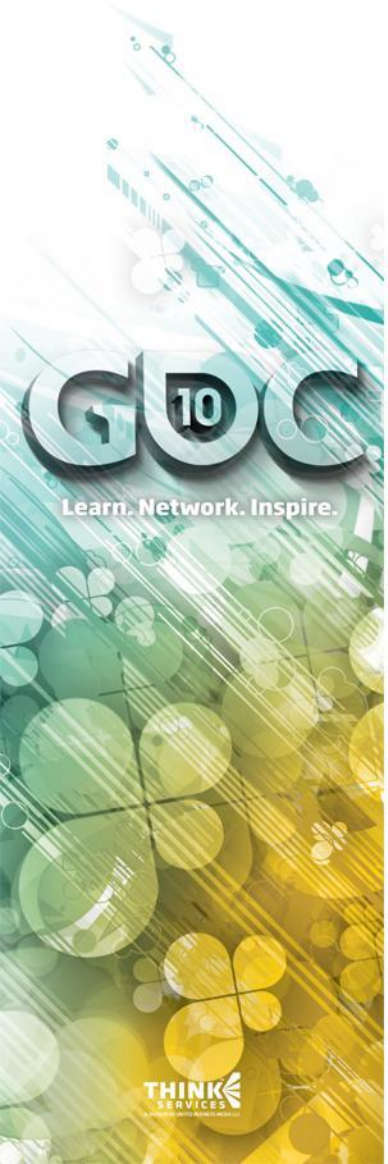
Rendering Passes

- Simplified rendering passes
  - Render world
  - Render foreground into depth partition
  - No depth-only pass
  - No per-light passes
    - Skeletal mesh shader supports one merged light
  - No occlusion queries
    - Unfortunately!

Topics



- Background
- Method of Attack
  - Bringup
  - What we kept
  - Changes we made
    - Input
    - Renderer
    - Tools
- Workflow Changes
- Where To Go From Here



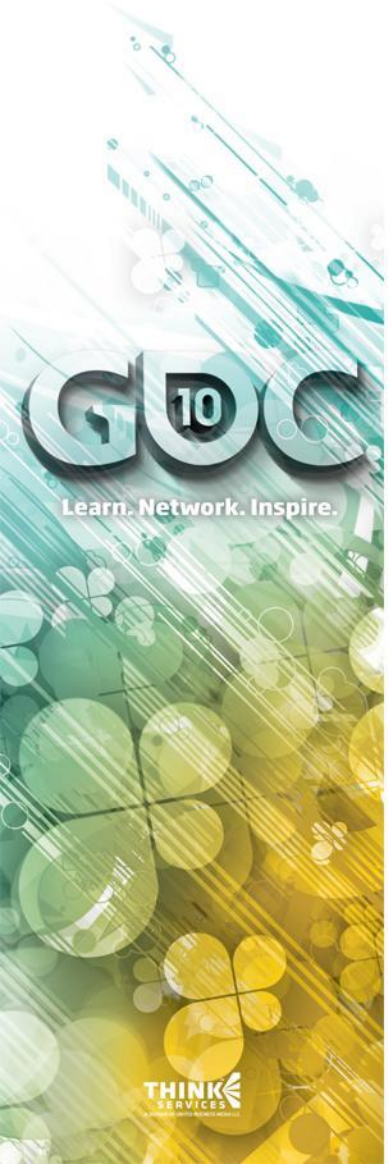
[Changes we made – Tools]

Content

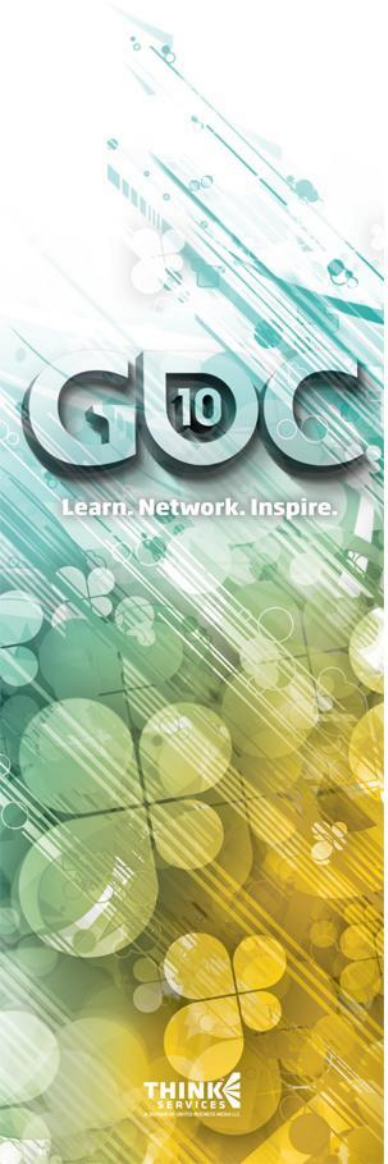
- Editor
  - Material flattening support
  - Mobile emulation
- Cooker
  - Cook on PC, then copy to Mac via script
  - Xcode script to copy cooked data into .app
    - A Run Script phase in each Target

```
SRC=${PROJECT_DIR}/../FromPC/${TARGET_NAME}/Cooked
DST=${BUILT_PRODUCTS_DIR}/${UNLOCALIZED_RESOURCES_FOLDER_PATH}
cp -Rf ${SRC} ${DST}
```

Topics

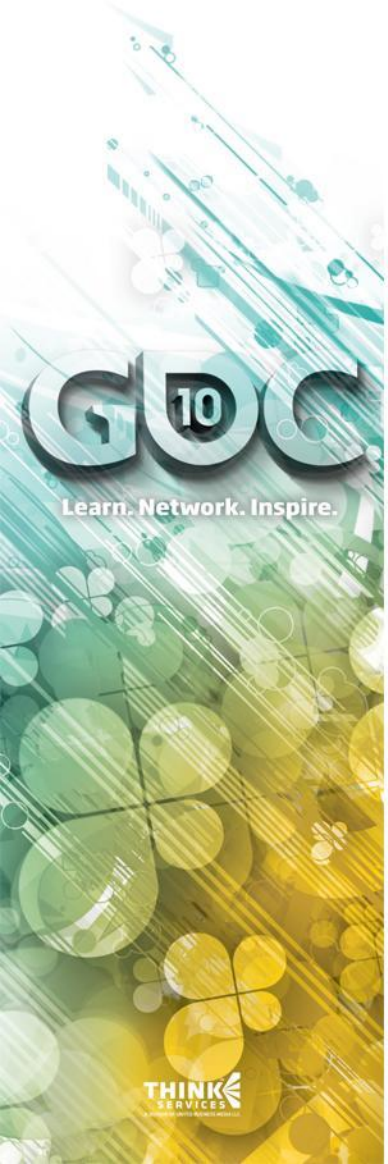


- Background
- Method of Attack
- Workflow Changes
  - Compiling
  - Signing
  - Installing
  - Debugging
- Where To Go From Here



Topics

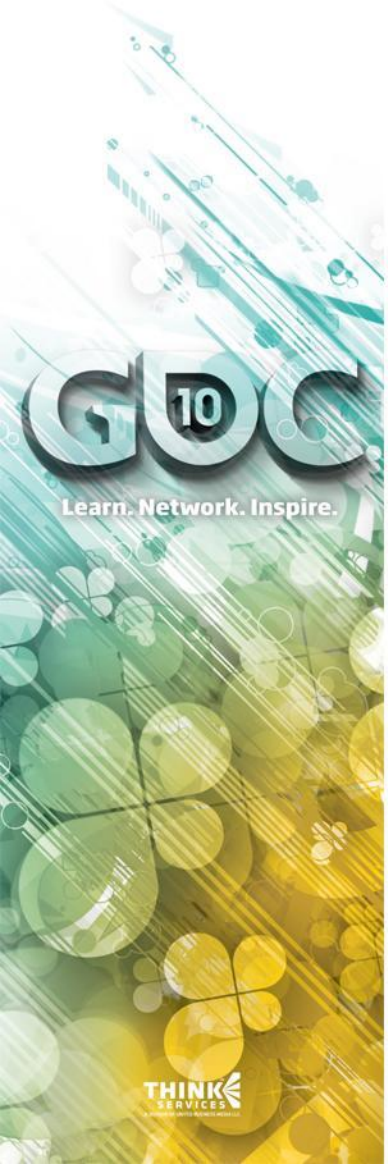
- Background
- Method of Attack
- Workflow Changes
  - Compiling
  - Signing
  - Installing
  - Debugging
- Where To Go From Here



[Workflow Changes – Compiling]

Problem

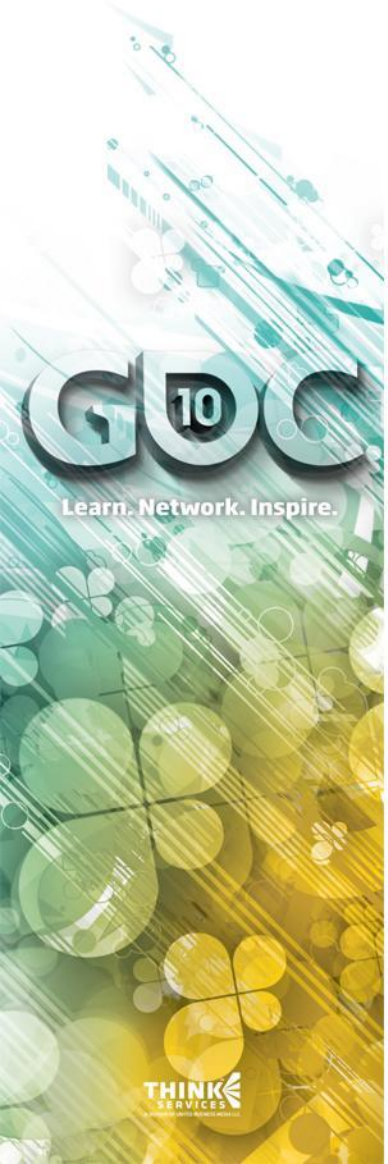
- UE3 toolchain is Windows based
- How to leverage?
  - Can’t remove Mac from process
  - No Jailbreaking, please!
  - Windows + Mac = ☺



[Workflow Changes – Compiling]

Moving to Windows

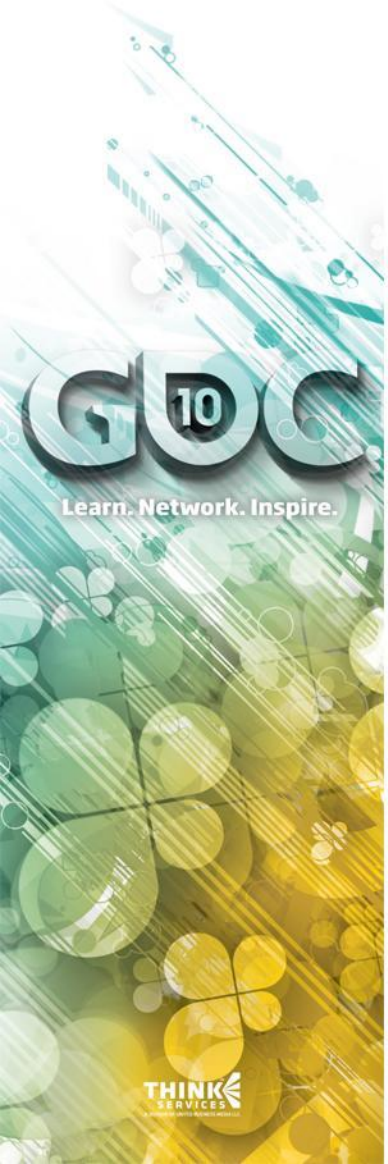
- We want to compile from Windows
- Mac still does the meat
  - Compiling, Linking, Signing
  - Simulator
  - Debugging
- First, add iPhone bits to VS .sln
- Now, we can use UnrealBuildTool



[Workflow Changes – Compiling]

UnrealBuildTool

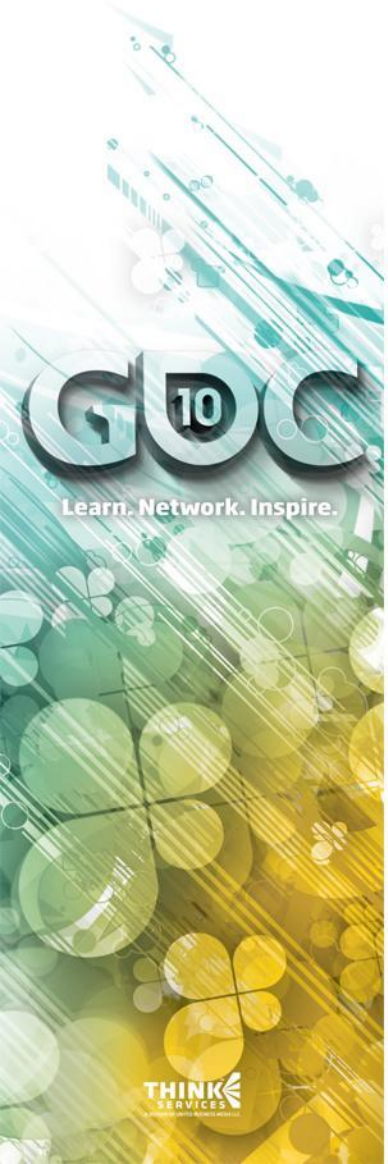
- C# utility that controls compilation
  - .sln is just a file reservoir
  - UBT parses .vcproj files
  - Creates build Actions and Dependencies
- Added Copy to Mac action
  - Dependency (input) is local file
  - Output file is remote (Mac) file
  - Action is file copy (if input newer)



[Workflow Changes – Compiling]

UnrealBuildTool

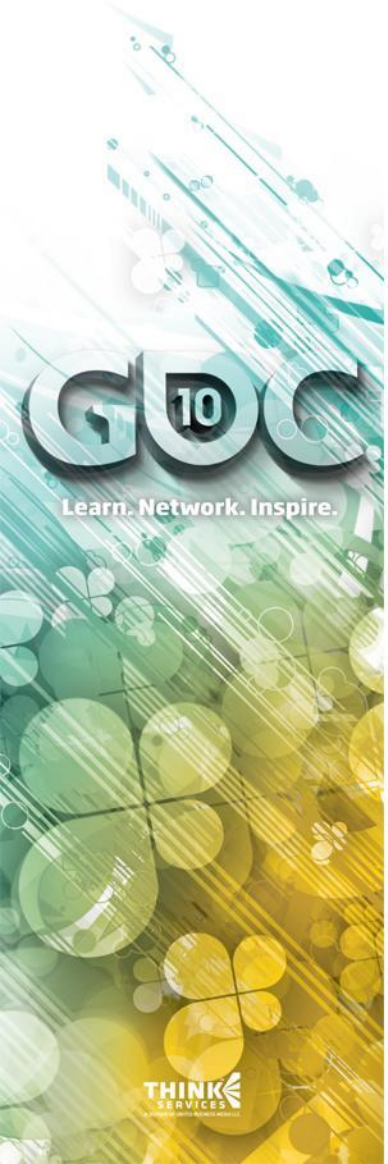
- Needs per-user environment variables:
  - MacDevPath – Mac path to dev root
  - PCDevPath – UNC path to Mac dev root
  - MacName – Name of user's Mac



[Workflow Changes – Compiling]

pscp and plink

- We use PuTTY command line tools
- pscp
  - Windows network copies messes up permissions (for us anyway)
    - A read-only file could never be overwritten
  - pscp uses proper user logins
  - *Can't "write if newer" however*
    - UBT does that for us for compiling
    - But, not for bulk copy scripts



[Workflow Changes – Compiling]

pscp and plink

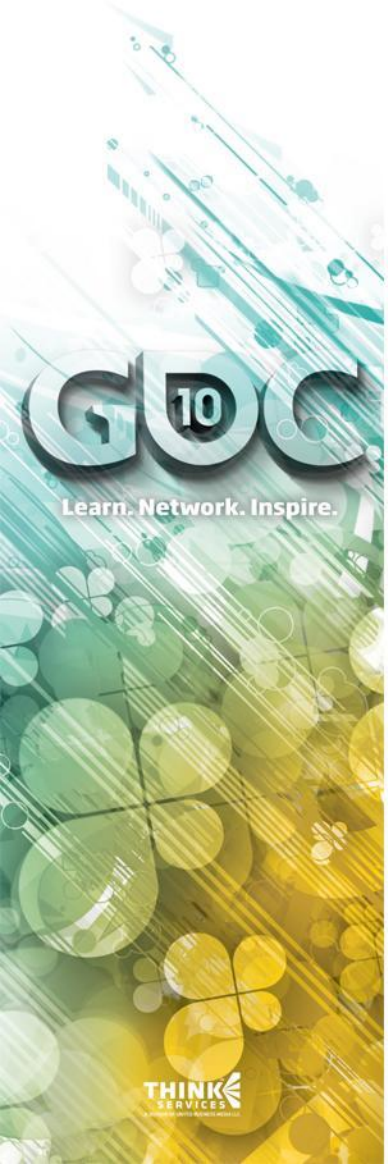
- plink
  - Performs a command over SSH
  - Used to run gcc and other remote ops
- Use PuTTY's UI to setup auth
  - Private/public key for SSH auth
  - Set auto-authenticate for pscp/plink
    - Setup PuTTY, then save Default config
  - Allows for one script to work globally

[Workflow Changes - Compiling]

gcc

- Boost gcc commands from Xcode

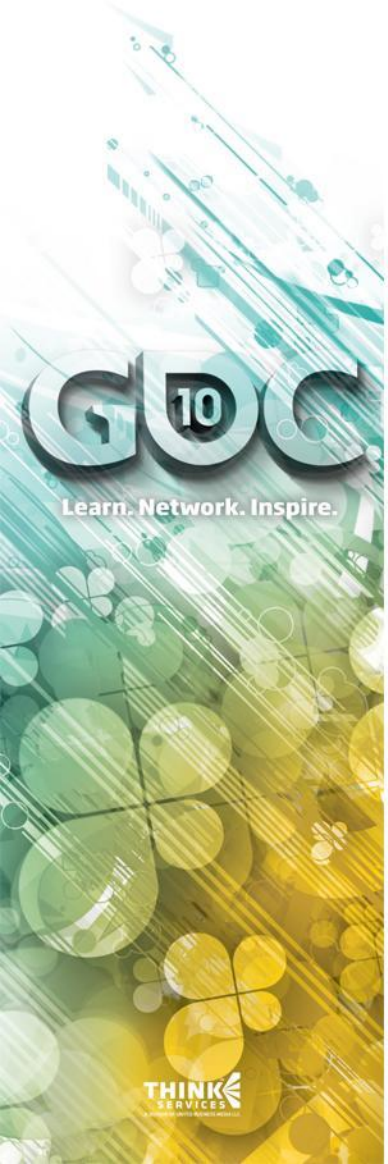
  - Some cleanup possible



[Workflow Changes – Compiling]

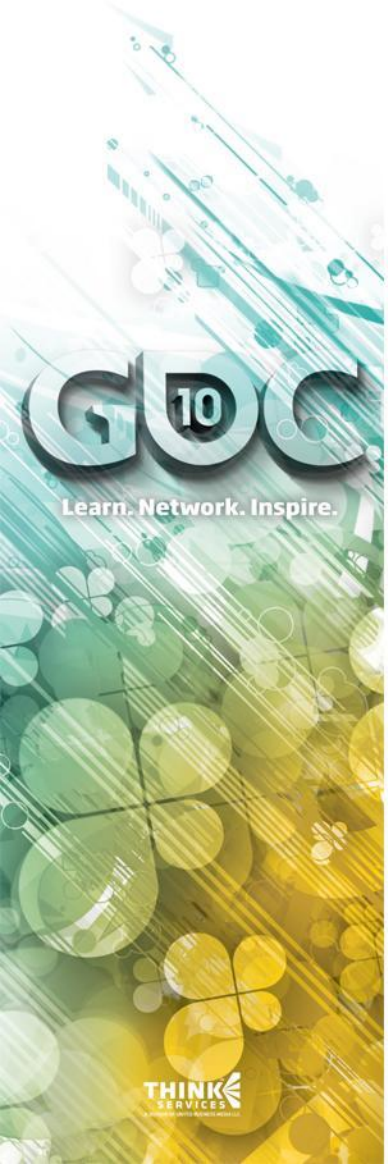
Compiling/linking

- Info.plist
  - Xcode-created file with app settings
  - Need to replace
    - ${PRODUCT_NAME}
    - ${EXECUTABLE_NAME}
  - UBT calls sed, output into .app folder



Topics

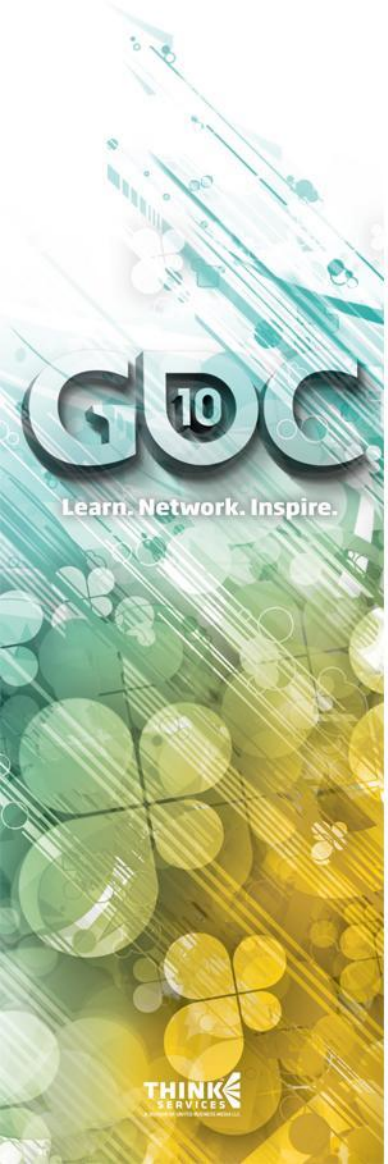
- Background
- Method of Attack
- Workflow Changes
  - Compiling
  - Signing
  - Installing
  - Debugging
- Where To Go From Here



[Workflow Changes – Signing]

Problem

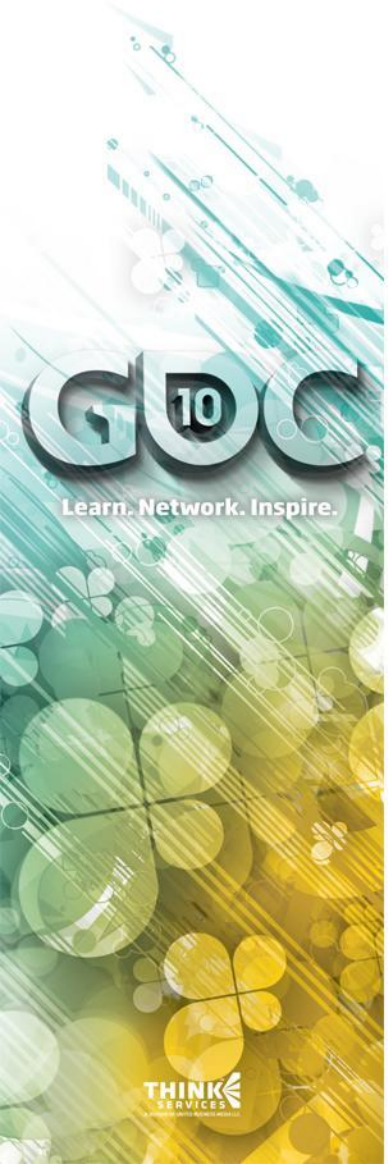
- Signing without Xcode is tricky
  - Commandline tools
  - Keychain issues



[Workflow Changes – Signing]

Signatures

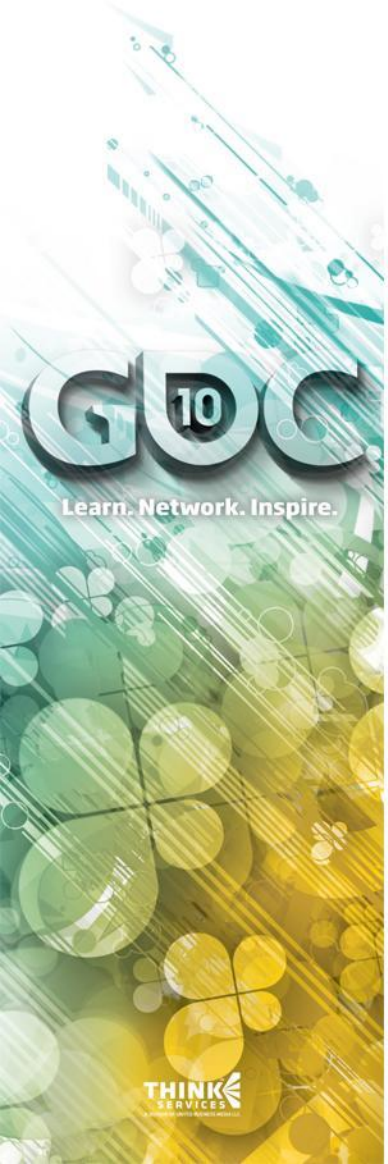
- Xcode internal command:
  - <com.tools.product-pkg-utility>
    - (shown as “ProcessProductPackaging”)
  - Generates:
    - .xcent, embedded.mobileprovision
  - Compile empty project with same target name
    - .xcent has target name inside



[Workflow Changes – Signing]

Signatures

- <com.tools.product-pkg-utility>
  - Remake .mobileprovision whenever ProvisioningProfile is updated
  - Copy generated files to PC
    - Along with ResourceRules.plist
  - Check-in for all devs to use in signing



[Workflow Changes – Signing]

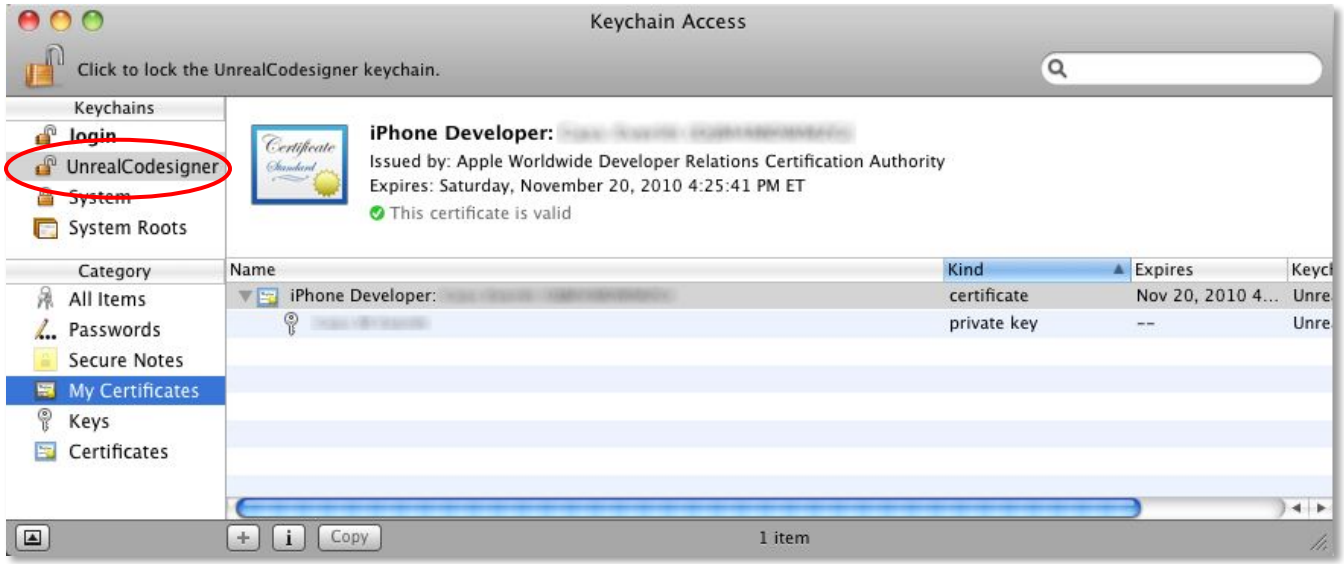
Keychain

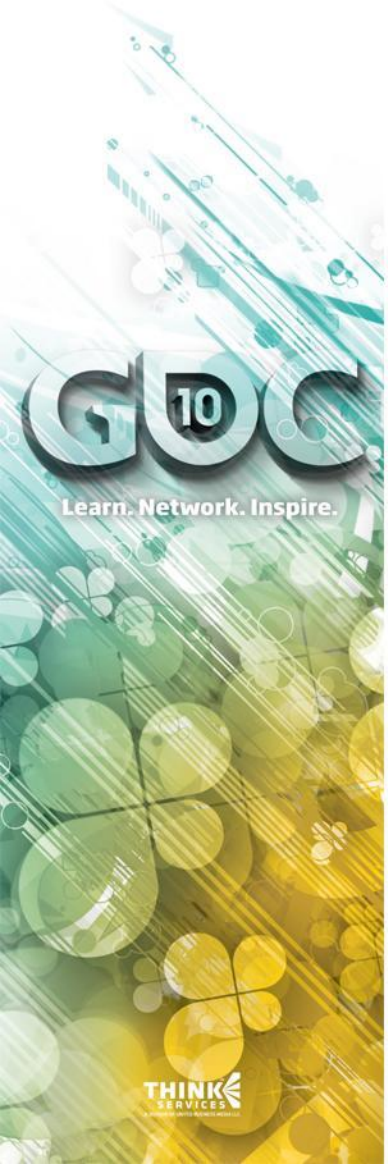
- Uses Mac Keychain, locked in SSH:
  - Run Keychain Access app
  - New keychain, same name for all devs
  - Give it an insecure password, same for all
  - Install your iPhone cert to *login* keychain
    - *Do not install directly to new keychain!*
  - Drag certificate from login to new keychain
    - Key will come over with it if "My Certificates" is selected

[Workflow Changes – Signing]

Keychain

- Keychain Access setup:





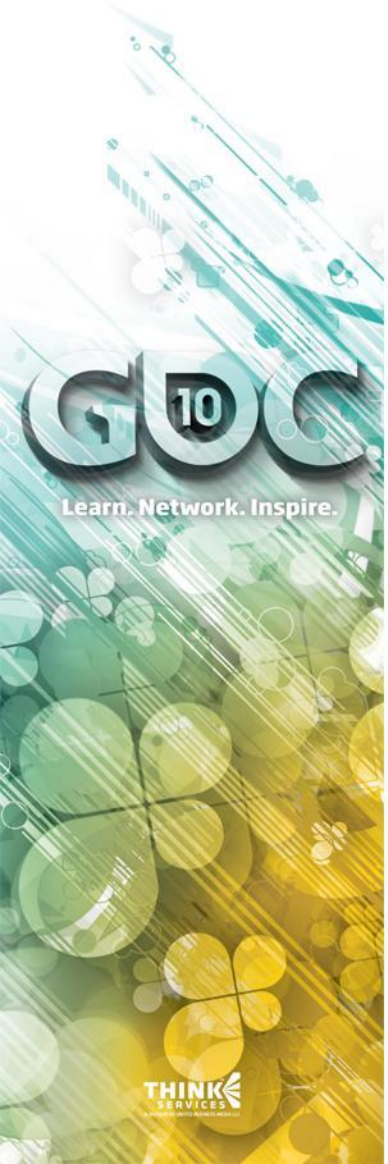
[Workflow Changes – Signing]

Commandline

- Our plink command:

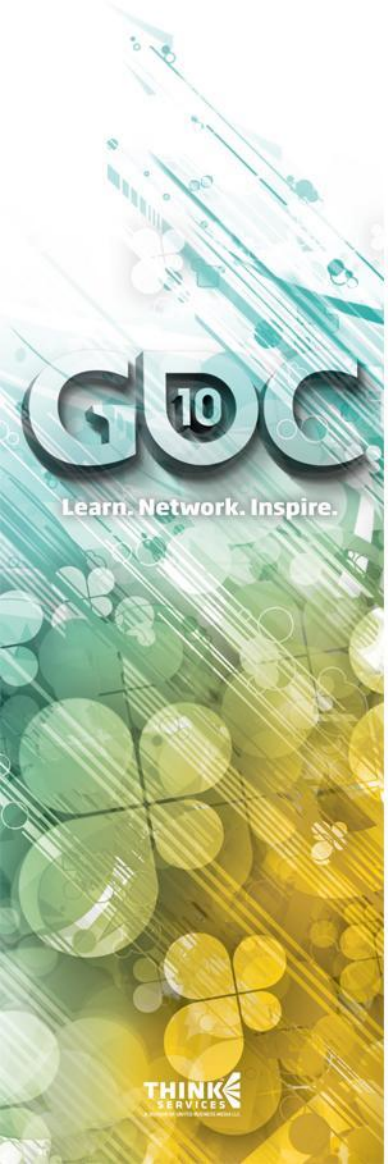
```
plink %MacName%
export CODESIGN_ALLOCATE=/Developer/Platforms/
iPhoneOS.platform/Developer/usr/bin/codesign_allocate;
security unlock-keychain -p pwd UnrealCodesigner.keychain;
/usr/bin/codesign -f -s \"iPhone Developer\"
--resource-rules=%MacAppDir%/ResourceRules.plist
-entitlements %MacBuildDir%/%1.xcent %MacAppDir%
```

- Must be all one plink command
  - Keychain remains unlocked



Topics

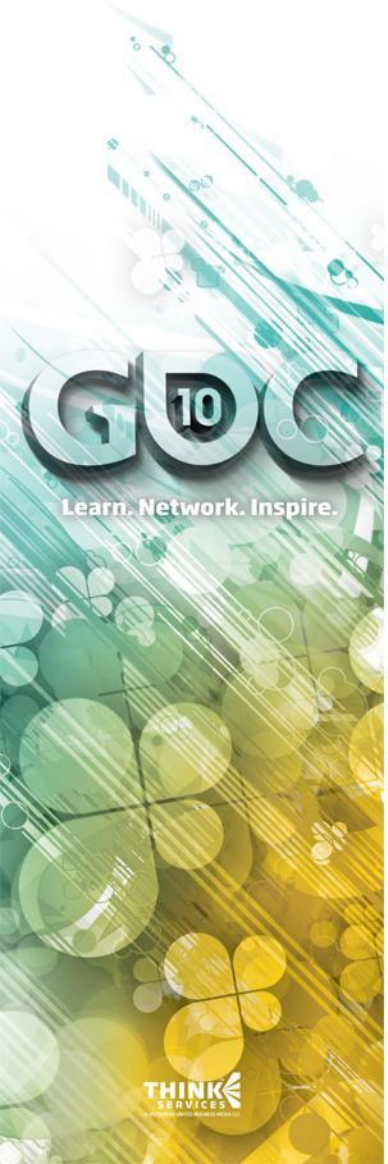
- Background
- Method of Attack
- Workflow Changes
  - Compiling
  - Signing
  - Installing
  - Debugging
- Where To Go From Here



[Workflow Changes – Installing]

Problem

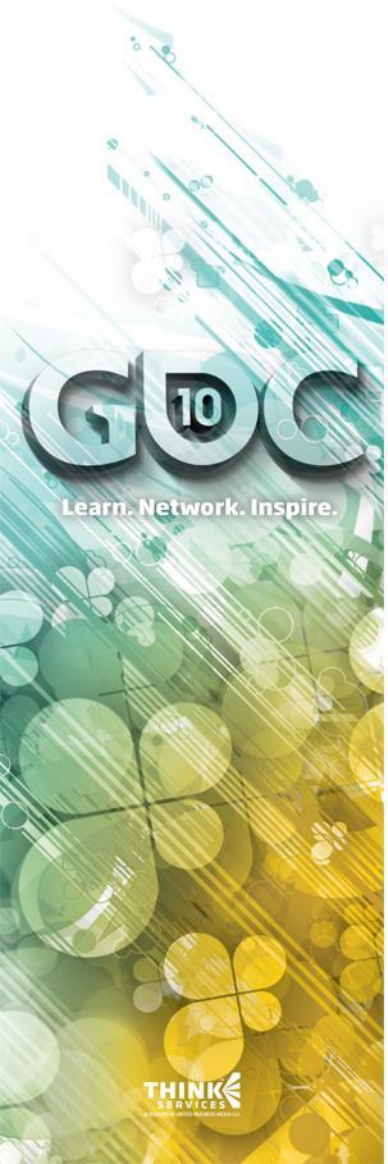
- Xcode automates installing
  - Remote commandline – not so automatic



[Workflow Changes – Installing]

Device

- Organizer window in Xcode
  - Easier than iTunes



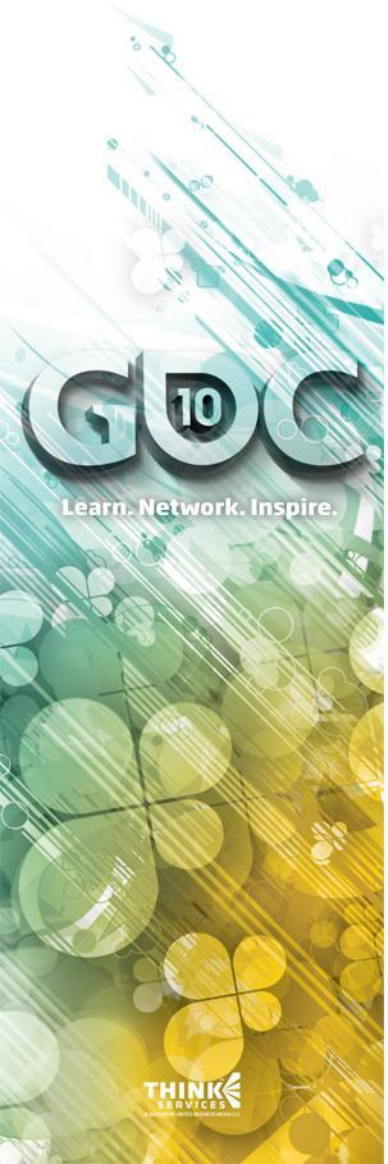
[Workflow Changes – Installing]

Simulator

- Installing an app in Simulator
  - Not straightforward via plink
  - Use killall to kill running sim app
  - Copy .app folder to:

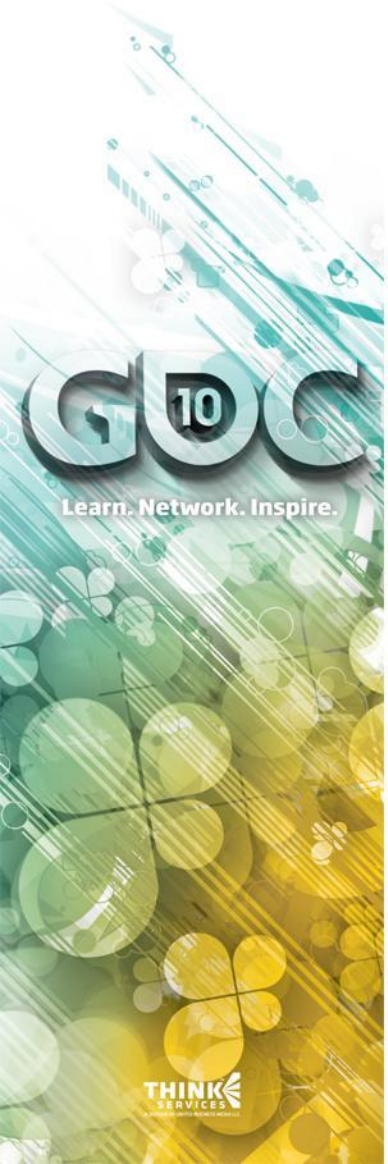
```
~/Library/Application Support/iPhone Simulator/User/Applications/PC
```

  - Delete all GUID directories in Applications that contain your game .app
    - Xcode makes these when debugging
  - Run game in Simulator by hand



Topics

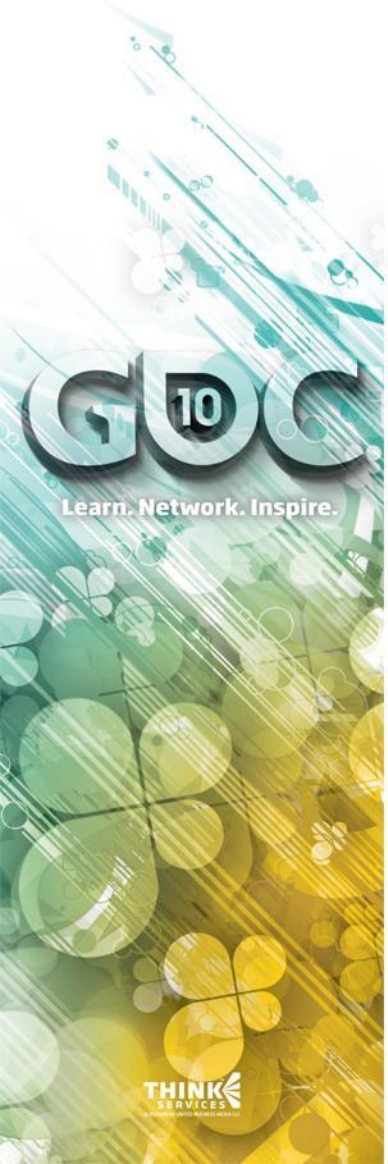
- Background
- Method of Attack
- Workflow Changes
  - Compiling
  - Signing
  - Installing
  - Debugging
- Where To Go From Here



[Workflow Changes – Debugging]

Problem

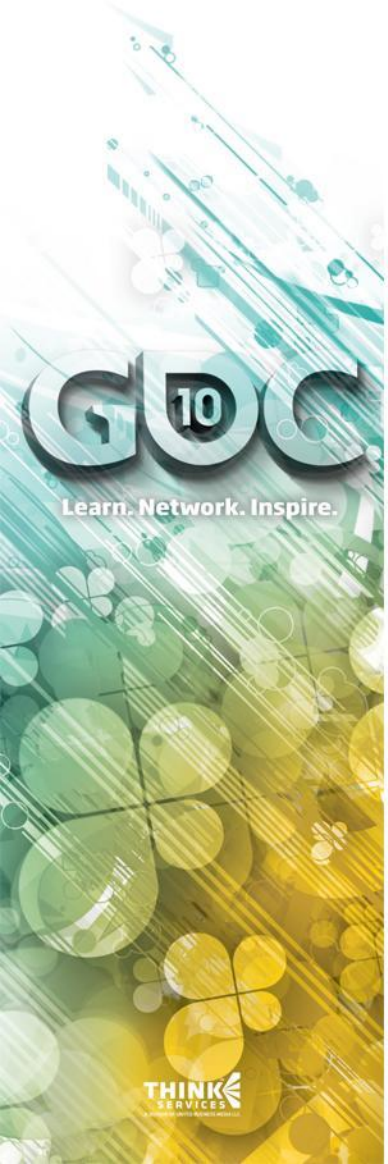
- How to debug in Xcode?
  - Xcode didn’t build the .app
  - Xcode needs a project to debug



[Workflow Changes – Debugging]

Xcode project

- DebuggerProject
  - Dead simple Xcode project
    - Target and Executable for each game
    - No code needed
  - Choose Simulator/Device, Debug/Release
  - Run will install and run “PC compiled” .app
    - Don’t use Build and Run
    - Replace Build and Run in toolbar ☺

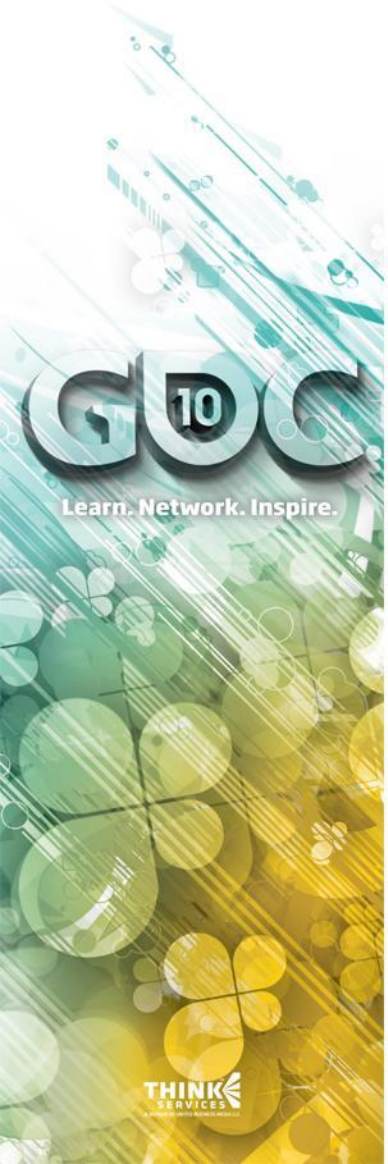


[Workflow Changes – Debugging]

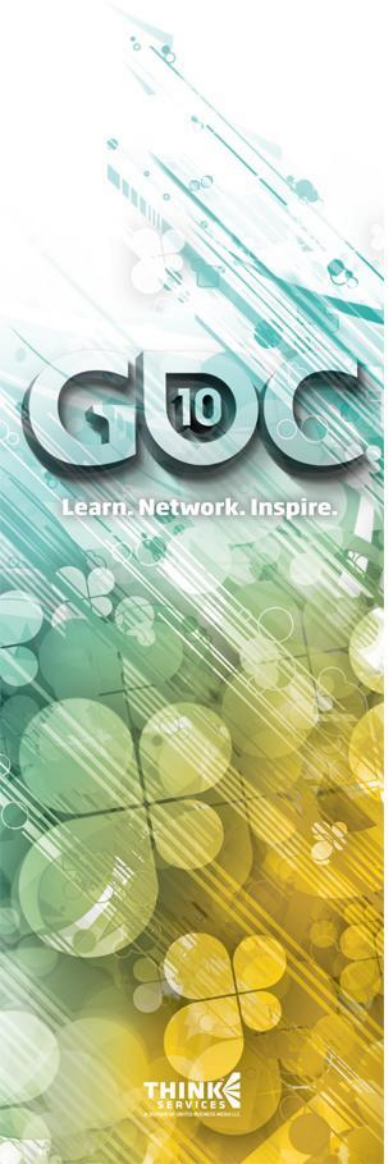
Debug info

- .dSYM file
  - Takes ~1 minute to generate with UE3
  - Xcode will always generate it
    - Even with 'DWARF' vs. 'DWARF with dSYM'
  - We leave debug info in executable
    - Make sure to strip for final build!
  - Still able to use breakpoints, etc.
  - Profiling:
    - *dSYM is needed for Instruments*

Topics



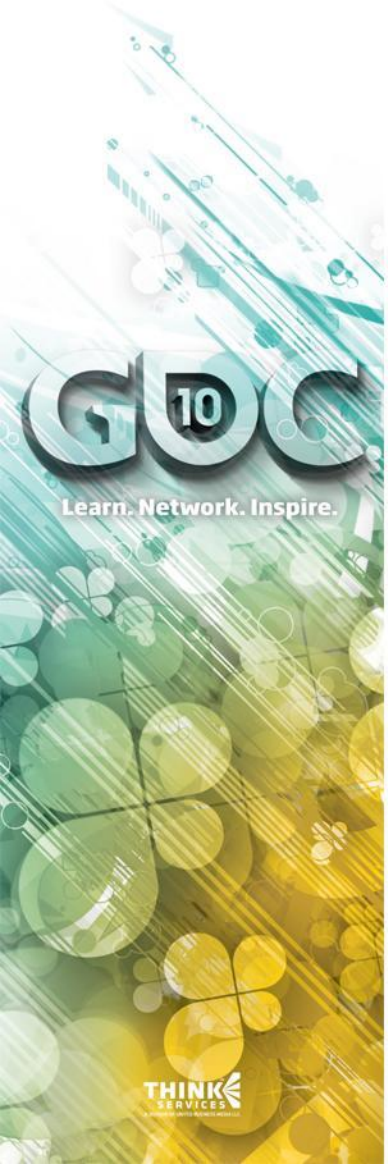
- Background
- Method of Attack
  - Bringup
  - What we kept
  - Changes we made
- Workflow Changes
- Where To Go From Here



[Where To Go From Here]

Future direction

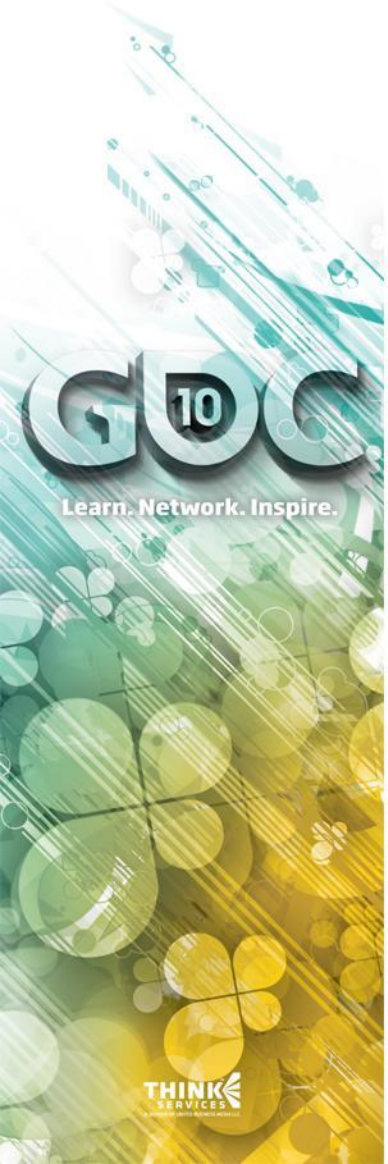
- Newer chips
  - More power!
  - Add normal or specular maps, etc
- Newer GL drivers
  - Occlusion queries
  - SRGB
  - Use full materials on some
    - “Hero” pieces get full support
    - Needs offline compiling, really



[Where To Go From Here]

Future direction

- 3rd party library support
  - Add them when we get them
  - PhysX, GameSpy, etc
- Generate the special Xcode files
  - .xcent is just .plist with some binary goo
  - .mobileprovision is painful by hand

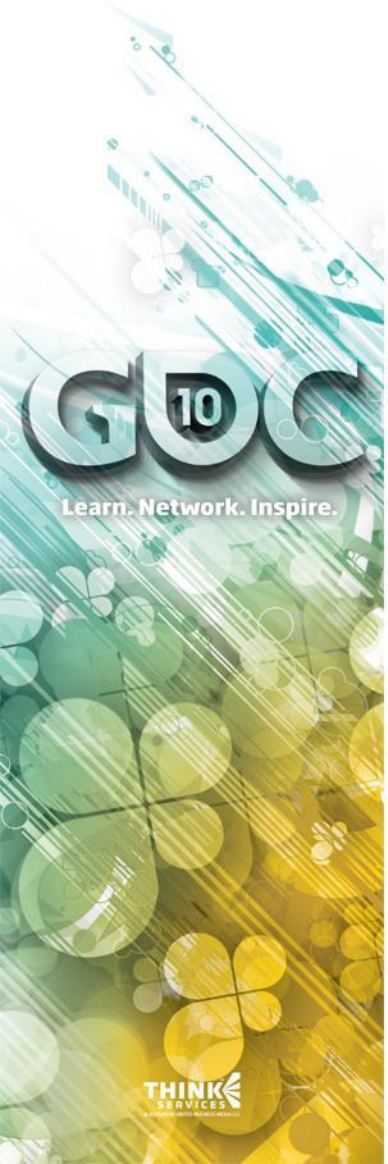


[Where To Go From Here]

Future direction

- Compiler fix for Neon
  - Apple says it's fixed in upcoming version
- Auto-run game on Simulator
  - Undocumented MacOS framework:
    - iPhoneSimulatorRemoteClient

Q&A



- Any questions?
  - Josh Adams
    - josh.adams@epicgames.com
  - Epic Games
    - www.epicgames.com
  - Unreal Technology
    - www.unrealtechnology.com
  - GDC Booth
    - ES 202, South Hall
    - (aka BS200)