MFF UK
NPGR033
23th March 2016

Unreal Engine 4 – Platform Independence

Game Engines – Part II

Jakub Gemrot

Based on "various sources"

Unreal Engine
History

- Unreal Engine 1 – May 1998
- Unreal Engine 2 – January 2001
- Unreal Engine 3 – March 2003
- Unreal Development Kit – November 2009
- Unreal Engine 4 – May 2012

~ 20 years of experiences
~ „No one knows every corner of UE4 sources."
-- Gerke Max Preussner, UE4 Senior Engineer

Unreal Engine 1

May 1998

- Software renderer and Glide API (3Dfx)
- Later Direct3D, OpenGL
- Easy to mod using UnrealScript
- Networking later on

Unreal

Unreal Engine 1

May 1998

- Software renderer and Glide API (3Dfx)
- Later Direct3D, OpenGL
- Easy to mod using UnrealScript
- Networking later on

Tactical Ops

Unreal Engine 2
January 2001

- Rewritten renderer
- PS2, Xbox, GameCube
- Karma Physics SDK
- 64-bit later on

America's Army

Unreal Engine 3

March 2002

- DX 9/10
- XBox 360, PS3
- Ported for Stage3D
- Many updates later on

Gears of War

Unreal Development Kit

November 2009

- UE3 made “public”
- 99$ upfront, after 5000$ sales 25% royalties
- Changed to free and no royalties under 50000$ sales

The Ball

Unreal Engine 4

May 2012

- Major rewrite
- Modularization
- UnrealScript dropped
- New Blueprint system
- ...

AQP City

Unreal Engine 4

Main Points

- Complete platform abstraction
- Many (cutting edge) rendering & anim. Features
  - Landscape features, Level streaming, 8192x8192 terrains
- Physics (no soft bodies yet), Audio, Networking
- UI system (also as in-game textures)
- Extensible editor
  - 2D Plugin, Blueprints
  - Own Game module
- No game specific stuff (inventories, weapons, …)

Unreal Engine 4

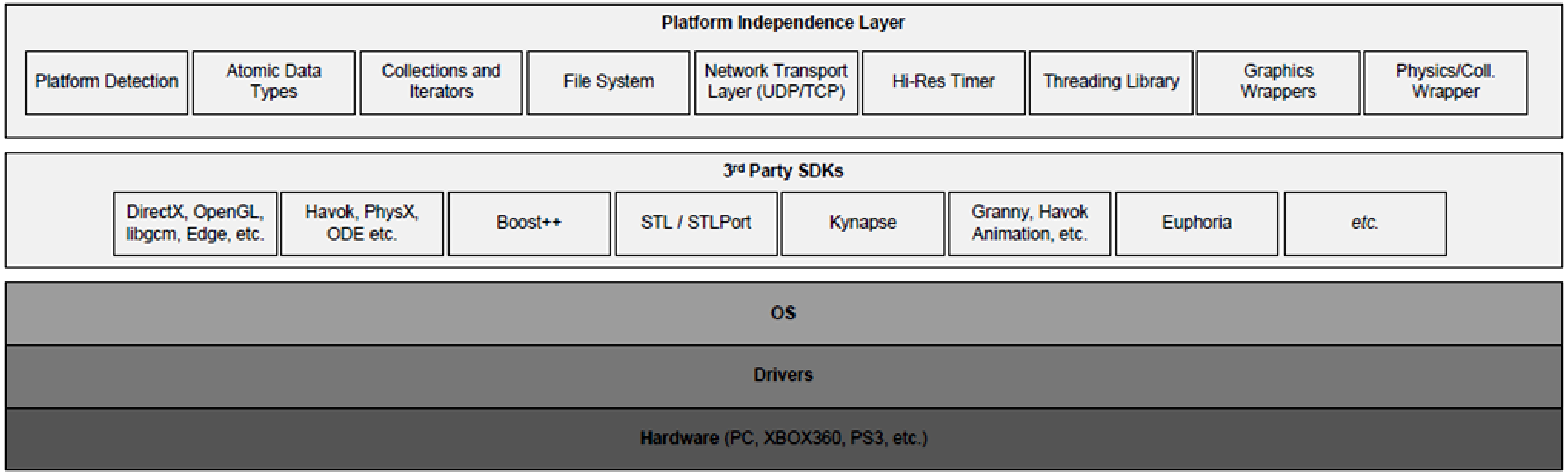
Main Points

- **Complete platform abstraction**
- Many (cutting edge) rendering & anim. Features
  - Landscape features, Level streaming, 8192x8192 terrains
- Physics (no soft bodies yet), Audio, Networking
- UI system (also as in-game textures)
- Extensible editor
  - 2D Plugin, Blueprints
  - Own Game module
- No game specific stuff (inventories, weapons, …)

Unreal Engine 4

Main Points

- Complete platform abstraction
  - Windows, Mac, Linux, Android, iOS, HTML5, XBox One, PS4



Unreal Engine 4

Complete Platform Abstraction

- Custom build tool chain *(your solution is a lie)*
  - Unreal Build Tool (UBT)
  - Unreal Header Tool (UHT)
  - Unreal Automation Tool (UAT)
  - *And a few others…*

Unreal Engine 4
Complete Platform Abstraction

- Modules
  - Whole engine is modularized
  - Many interfaces, which are then implemented for respective platforms
- Plug-ins
  - Works with the abstraction only
  - You can slip in custom plugins into your compiled editor and export them with your game

Unreal Engine 4
Complete Platform Abstraction

- Modules
  - Module Types
    - Developer – Used by Editor & Programs, not Games
    - Editor – Used by Unreal Editor only
    - Runtime – Used by Editor, Games & Programs
    - ThirdParty – External code from other companies
    - Plugins – Extensions for Editor, Games, or both
    - Programs – Standalone applications & tools
  - Module Dependency Rules
    - Runtime modules <u>must not</u> have dependencies to Editor or Developer modules
    - Plug-in modules <u>must not</u> have dependencies to other plug-ins

Unreal Engine 4

Complete Platform Abstraction

- Modules

Module Type	Editor	App	Game
Runtime	√	√	√
ThirdParty	√	√	√
Plugins	√	√	√
Developer	√	√	X
Editor	√	X	X

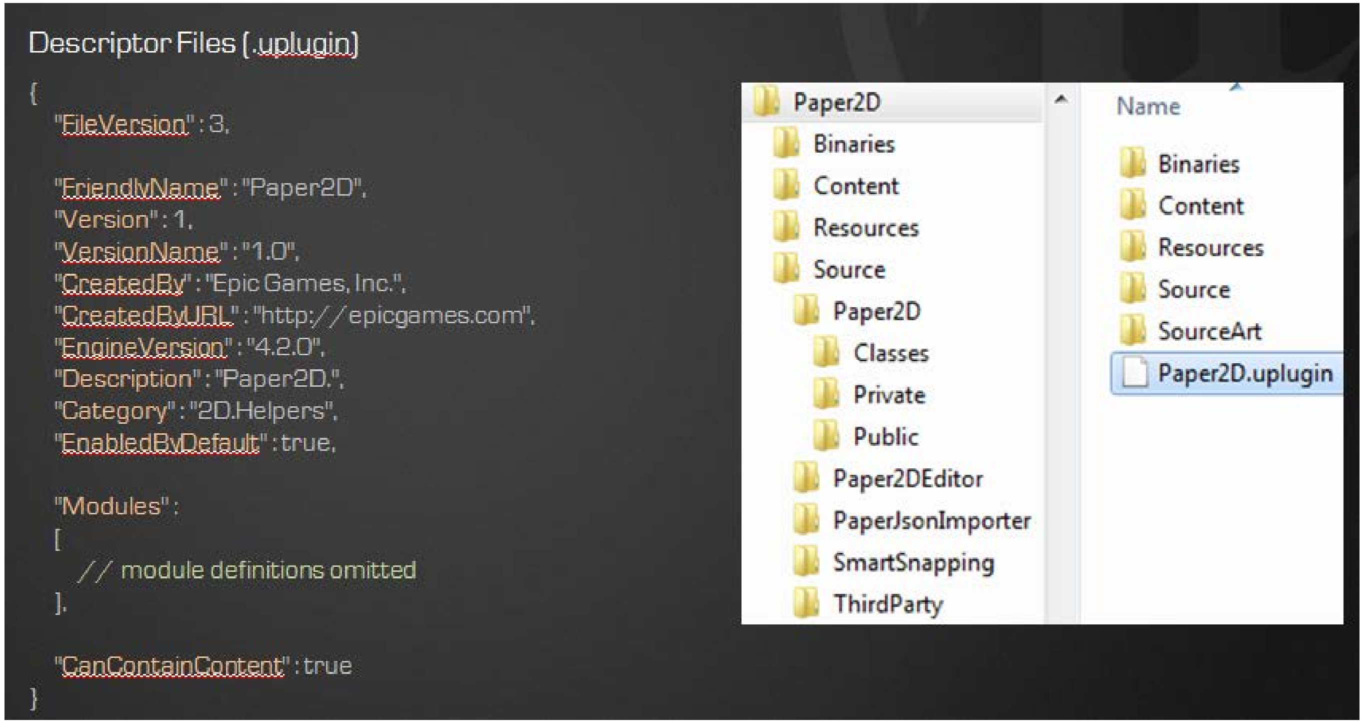
Unreal Engine 4

Complete Platform Abstraction

- Plug-ins
  - Loaded dynamically on startup
  - Should not depend on other plugins
  - Own source, binaries, content, config files

Unreal Engine 4
Complete Platform Abstraction

- Plug-ins



Descriptor Files (.uplugin)

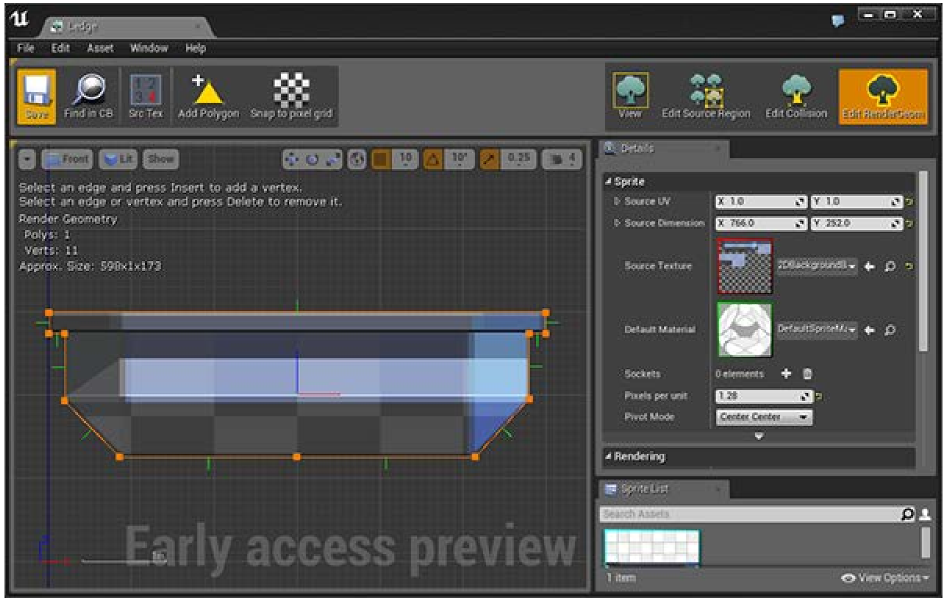
```
{
  "FileVersion" : 3,

  "FriendlyName" : "Paper2D",
  "Version" : 1,
  "VersionName" : "1.0",
  "CreatedBy" : "Epic Games, Inc.",
  "CreatedByURL" : "http://epicgames.com",
  "EngineVersion" : "4.2.0",
  "Description" : "Paper2D.",
  "Category" : "2D.Helpers",
  "EnabledByDefault" : true,

  "Modules" :
  [
     // module definitions omitted
  ],

  "CanContainContent" : true
}
```

```
Paper2D
  Binaries
  Content
  Resources
  Source
    Paper2D
      Classes
      Private
      Public
    Paper2DEditor
    PaperJsonImporter
    SmartSnapping
    ThirdParty
```

```
Name
Binaries
Content
Resources
Source
SourceArt
Paper2D.uplugin
```

Unreal Engine 4
Complete Platform Abstraction

- Paper2D

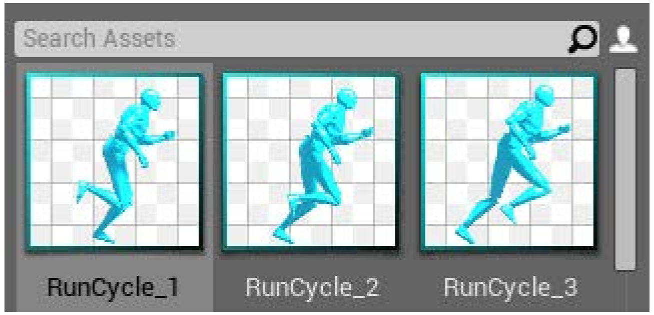


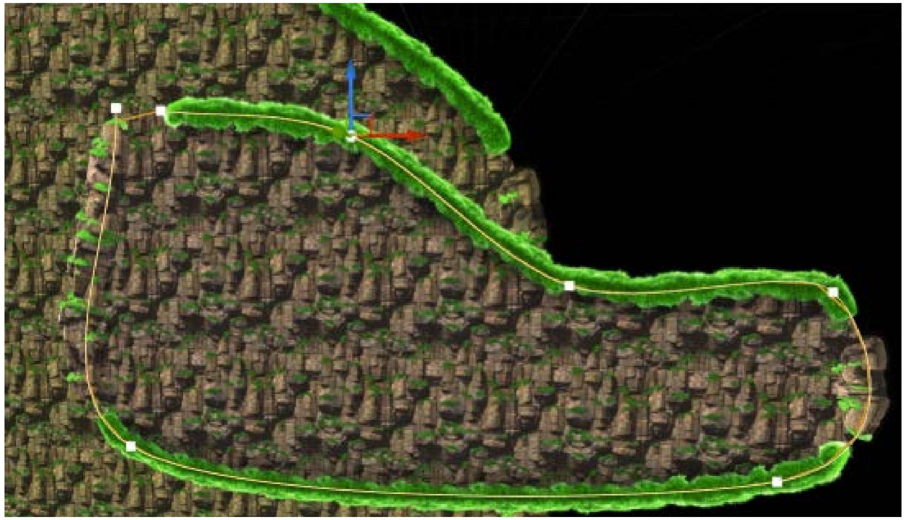
```
Select an edge and press Insert to add a vertex.
Select an edge or vertex and press Delete to remove it.
Render Geometry
 Polys: 1
 Verts: 11
Approx. Size: 598x1x173
```

Sprite
- Source UV: X 1.0, Y 1.0
- Source Dimension: X 766.0, Y 252.0
- Source Texture: 2DBackgroundl
- Default Material: DefaultSpriteM
- Sockets: 0 elements
- Pixels per unit: 1.28
- Pivot Mode: Center Center

Rendering

1 item





Unreal Engine 4

Complete Platform Abstraction

- Custom build tool chain *(your solution is a lie)*
  - Unreal Build Tool (UBT)
    - Written in C# (may convert to C++ in the future)
    - Scans solution directory for modules and plug-ins
    - Determines all modules that need to be rebuilt
    - Invokes UHT to parse C++ headers
    - Creates compiler & linker options from .Build.cs & .Target.cs
    - Executes platform specific compilers (VisualStudio, LLVM)
    - Auto-generates DLL on Windows
    - Solution file generation
    - Remote compilation (iOS, MacOS)

Unreal Engine 4
Complete Platform Abstraction

- Custom build tool chain *(your solution is a lie)*
  - Unreal Header Tool (UHT)
    - Written in C++
    - Parses all C++ headers containing UClasses
    - Generates glue code for all Unreal classes & functions
      - Preprocess specific macros (RTTI, network replication, in-editor exposure)
    - Generated files stored in Intermediates directory

Unreal Engine 4

Complete Platform Abstraction

- Custom build tool chain *(your solution is a lie)*
  - Unreal Automation Tool (UAT)
    - Written in C# (may convert to C++ in the future)
    - Automates repetitive tasks through Automation Scripts
    - Build, cook, package, deploy and launch projects
    - Invokes UBT for compilation
    - Distributed compilation (XGE) & build system integration
    - Generate code documentation
    - Automated Testing of code and content
    - Configurable

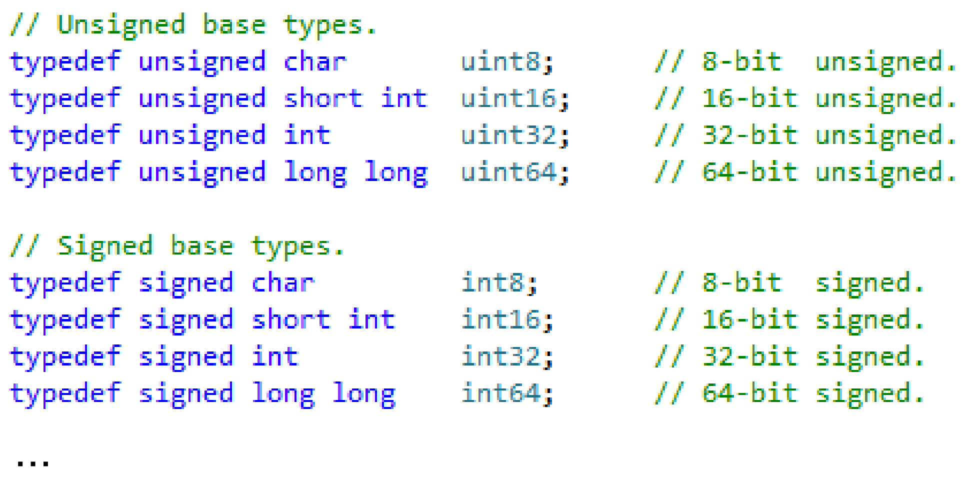
Unreal Engine 4
Complete Platform Abstraction

- Speaking UE4 Language
  - Fundamental types (primitives + a few others)
  - Containers
  - Delegates
  - Common game domain related structures
  - Smart pointers (UE4 is not using Boost…)
  - Strings
  - Macros
  - UObjects
  - Design principles in general

Unreal Engine 4

Speaking UE4 Language

- Fundamental types
  - Custom typedef's for ints & strings
  - *GenericPlatform.h*



```
// Unsigned base types.
typedef unsigned char        uint8;      // 8-bit  unsigned.
typedef unsigned short int   uint16;     // 16-bit unsigned.
typedef unsigned int         uint32;     // 32-bit unsigned.
typedef unsigned long long   uint64;     // 64-bit unsigned.

// Signed base types.
typedef signed char          int8;       // 8-bit  signed.
typedef signed short int     int16;      // 16-bit signed.
typedef signed int           int32;      // 32-bit signed.
typedef signed long long     int64;      // 64-bit signed.

...
```

Unreal Engine 4

Speaking UE4 Language

- Fundamental types
  - Numeric type traits
  - *NumericLimits.h*

```
#define MIN_uint8        ((uint8)     0x00)
#define MIN_uint16       ((uint16)    0x0000)
#define MIN_uint32       ((uint32)    0x00000000)
#define MIN_uint64       ((uint64)    0x0000000000000000)
#define MIN_int8         ((int8)      -128)
#define MIN_int16        ((int16)     -32768)
#define MIN_int32        ((int32)     0x80000000)
#define MIN_int64        ((int64)     0x8000000000000000)
...
```

```
template<>
struct TNumericLimits<uint8>
{
    typedef uint8 NumericType;

    static NumericType Min()
    {
        return MIN_uint8;
    }

    static NumericType Max()
    {
        return MAX_uint8;
    }

    static NumericType Lowest()
    {
        return Min();
    }
};
```

Unreal Engine 4

Speaking UE4 Language

- Containers
  - TArray, TSparseArray – Dynamic arrays
  - TLinkedList, TDoubleLinkedList
  - TMap – Key-value hash table
  - TQueue – Lock free FIFO
  - TSet – Unordered set (without duplicates)
  - *More in Core module*

Unreal Engine 4

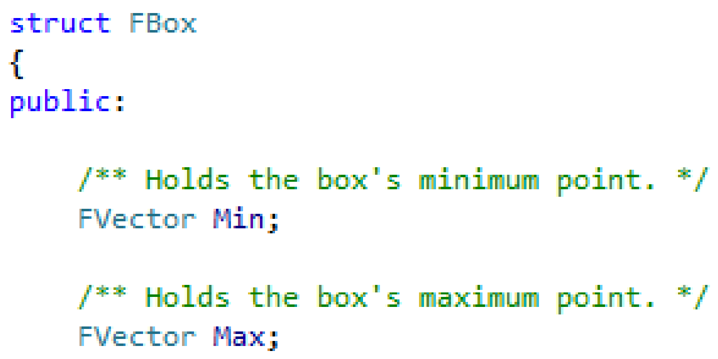
Speaking UE4 Language

- Delegates
  - Single / Multicast / UObject
    - ExecuteIfBound (as opposed to C#)
  - Limited signature
    - Up-to 4 parameters
    - Can be with / without return value
  - *More info in Delegate.h*

Unreal Engine 4

Speaking UE4 Language

- Common structures
  - FBox, FColor, FGuid, FVariant, FVector, TBigInt, TRange
  - Box.h



```cpp
struct FBox
{
public:

	/** Holds the box's minimum point. */
	FVector Min;

	/** Holds the box's maximum point. */
	FVector Max;
```

Unreal Engine 4

Speaking UE4 Language

- Smart pointers *(~ garbage collection)*
  - TSharedPtr, TSharedRef – for regular C++ objects
  - TWeakPtr – for regular C++ objects
  - TWeakObjPtr – for UObjects
  - TAutoPtr, TScopedPtr
  - TUniquePtr
  - Also thread-safe variants
  - Similar to boost:: & std:: implementations

Unreal Engine 4

Speaking UE4 Language

- Smart pointers

Benefit	**Description**
Clean syntax	You can copy, dereference, and compare shared pointers just like regular C++ pointers.
Prevents memory leaks	Resources are destroyed automatically when there are no more shared references.
Weak referencing	Weak pointers allow you to safely check when an object has been destroyed.
Thread safety	Includes thread safe version that can be safely accessed from multiple threads.
Ubiquitous	You can create shared pointers to virtually any type of object.
Runtime safety	Shared references are never null and can always be de-referenced.
No reference cycles	Use **weak pointers to break reference cycles**.
Confers intent	You can easily tell an object owner from an observer.
Performance	Shared pointers have minimal overhead. **All operations are constant-time**.
Robust features	Supports const, forward declarations to incomplete types, type-casting, etc.
Memory	Only **twice the size of a C++ pointer in 64-bit** (plus a shared 16-byte reference controller.)

Unreal Engine 4

Speaking UE4 Language

- Smart pointers *(~ garbage collection)*
  - Various helper functions ~ *MakeSharable(void*)*
  - Up-casting is implicit, just like with C++ pointers
  - Dynamically-allocated arrays are not supported yet
  - Related documentation

Unreal Engine 4

Speaking UE4 Language

- String Types
  - FString – Regular string
  - FText – Localized string, used heavily in Slate UI
  - FName – String hash, used heavily in UObjects, case-insensitive!

- String Literals
  - TEXT()
    - Creates a regular(!) string, i.e. TEXT("Hello");
  - LOCTEXT()
    - Creates a localized string, i.e. LOCTEXT("Namespace", "Name", "Hello");
  - NSLOCTEXT()
    - LOCTEXT with scoped namespace, i.e. NSLOCTEXT("Name", "Hello");

Unreal Engine 4
Speaking UE4 Language

- Macros (heavily used!)
  - Logging
    - UE_LOG, also GLog->Logf()
  - Assertions
    - check(), checkSlow(), ensure()
  - Localization
    - LOCTEXT_NAMESPACE, LOCTEXT, etc.
  - Slate (UI Framework)
    - SLATE_BEGIN_ARGS, SLATE_ATTRIBUTE, etc.
  - And many others

Unreal Engine 4

Speaking UE4 Language

- UObject
  - Run-time reflection of class properties and functions
  - Serialization from/to disk and over the network
  - Garbage collection
  - Meta data
  - Also: Blueprint integration
- Decorated regular C++ Classes with UHT Macros
  - UCLASS – for class types
  - USTRUCT – for struct types
  - UFUNCTION – for class and struct member functions
  - UPROPERTY – for class and struct variables

Unreal Engine 4

Speaking UE4 Language

- UObject
  - No dynamic allocation

```
UMyObjClass* DynamicObj = NewObject<UMyObjtClass>(this);
```

  - Prototype-like
    - Using a class default object for initialization of "new UObject"
  - Can be root-set (won't be auto-GCed)

```
YourObjectInstance->SetFlags(RF_RootSet);
```

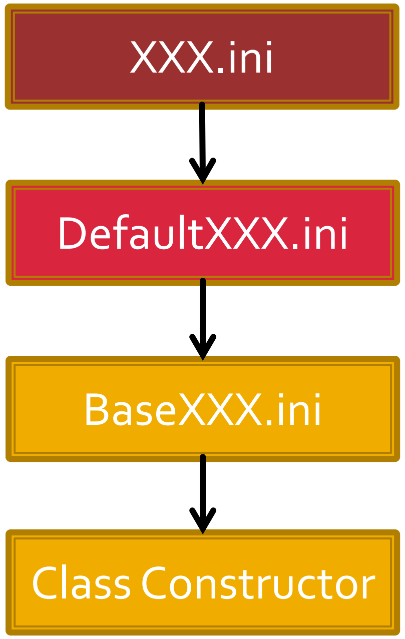
  - Always need to be checked for existence

```
if(!MyGCProtectedObj) return;
if(!MyGCProtectedObj->IsValidLowLevel()) return;
```

Unreal Engine 4

Speaking UE4 Language

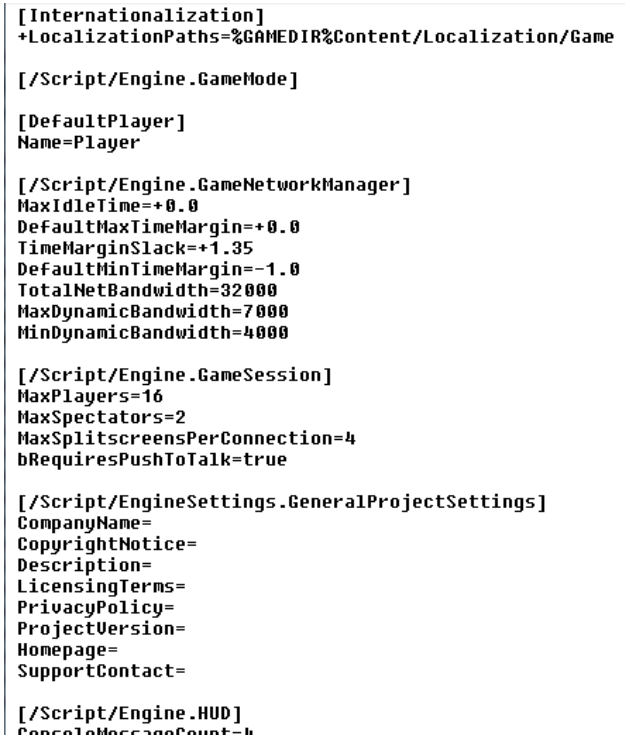
- UObject and INI files
  - Hold class default properties
  - Will be loaded into CDOs on startup
  - Organized in a hierarchy
  - Higher INIs override lower ones
  - Organized in sections
  - Key-value pairs within sections
  - Important ones exposed in Editor UI
  - Low-level access with FConfig



Unreal Engine 4

Speaking UE4 Language

- UObject and INI files



```
[Internationalization]
+LocalizationPaths=%GAMEDIR%Content/Localization/Game

[/Script/Engine.GameMode]

[DefaultPlayer]
Name=Player

[/Script/Engine.GameNetworkManager]
MaxIdleTime=+0.0
DefaultMaxTimeMargin=+0.0
TimeMarginSlack=+1.35
DefaultMinTimeMargin=-1.0
TotalNetBandwidth=32000
MaxDynamicBandwidth=7000
MinDynamicBandwidth=4000

[/Script/Engine.GameSession]
MaxPlayers=16
MaxSpectators=2
MaxSplitscreensPerConnection=4
bRequiresPushToTalk=true

[/Script/EngineSettings.GeneralProjectSettings]
CompanyName=
CopyrightNotice=
Description=
LicensingTerms=
PrivacyPolicy=
ProjectVersion=
Homepage=
SupportContact=

[/Script/Engine.HUD]
ConsoleMessageCount=4
```

Sections for UObjects

- [/Script/ModuleName.ClassName]

Sections for Custom Settings

- [SectionName]

Supported Value Types

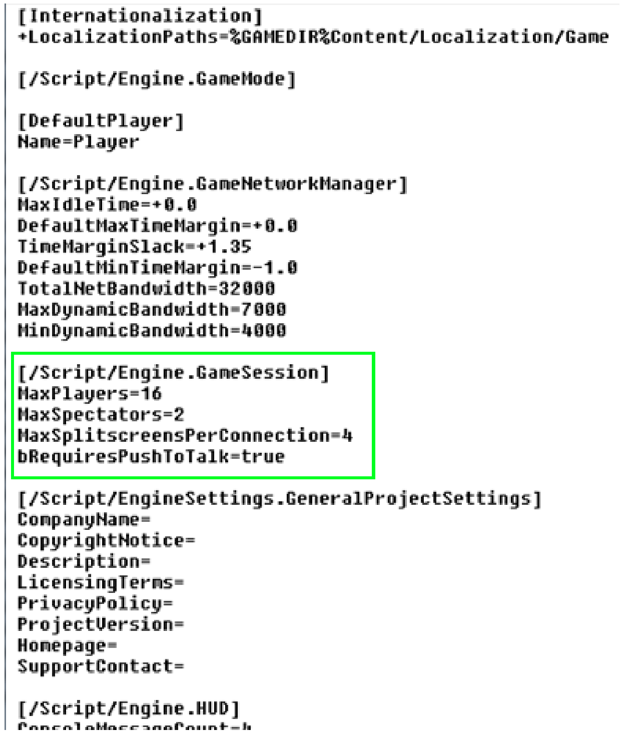
- Numeric values, strings, enums
- Structured data
- Static and dynamic arrays

Automatic serialization for UObject properties

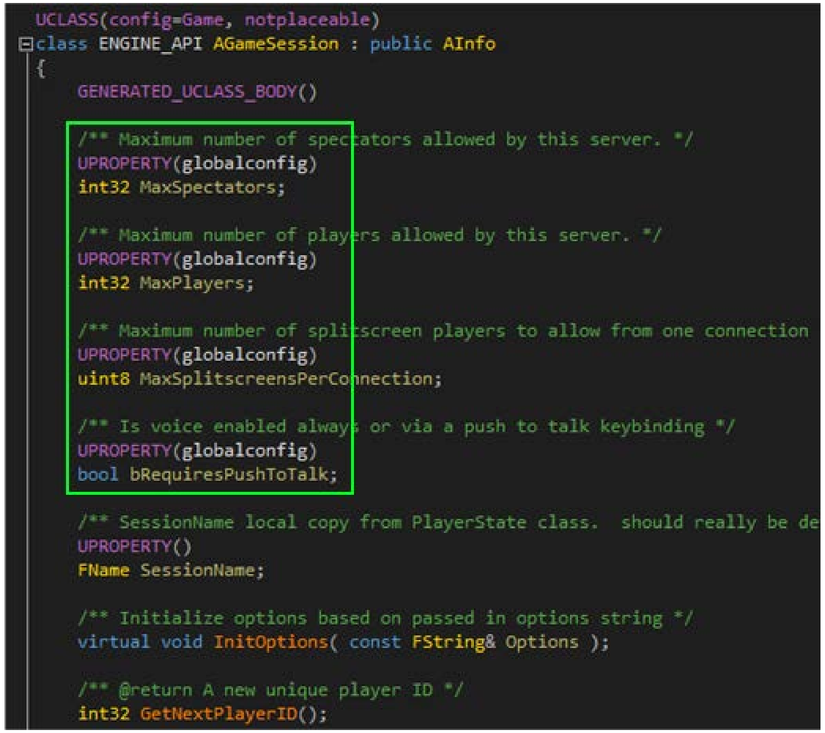
Unreal Engine 4

Speaking UE4 Language

- UObject and INI files



```
[Internationalization]
+LocalizationPaths=%GAMEDIR%Content/Localization/Game

[/Script/Engine.GameMode]

[DefaultPlayer]
Name=Player

[/Script/Engine.GameNetworkManager]
MaxIdleTime=+0.0
DefaultMaxTimeMargin=+0.0
TimeMarginSlack=+1.35
DefaultMinTimeMargin=-1.0
TotalNetBandwidth=32000
MaxDynamicBandwidth=7000
MinDynamicBandwidth=4000

[/Script/Engine.GameSession]
MaxPlayers=16
MaxSpectators=2
MaxSplitscreensPerConnection=4
bRequiresPushToTalk=true

[/Script/EngineSettings.GeneralProjectSettings]
CompanyName=
CopyrightNotice=
Description=
LicensingTerms=
PrivacyPolicy=
ProjectVersion=
Homepage=
SupportContact=

[/Script/Engine.HUD]
ConsoleMessageCount=4
```



```
UCLASS(config=Game, notplaceable)
class ENGINE_API AGameSession : public AInfo
{
    GENERATED_UCLASS_BODY()

    /** Maximum number of spectators allowed by this server. */
    UPROPERTY(globalconfig)
    int32 MaxSpectators;

    /** Maximum number of players allowed by this server. */
    UPROPERTY(globalconfig)
    int32 MaxPlayers;

    /** Maximum number of splitscreen players to allow from one connection
    UPROPERTY(globalconfig)
    uint8 MaxSplitscreensPerConnection;

    /** Is voice enabled always or via a push to talk keybinding */
    UPROPERTY(globalconfig)
    bool bRequiresPushToTalk;

    /** SessionName local copy from PlayerState class.  should really be de
    UPROPERTY()
    FName SessionName;

    /** Initialize options based on passed in options string */
    virtual void InitOptions( const FString& Options );

    /** @return A new unique player ID */
    int32 GetNextPlayerID();
```

Unreal Engine 4
Speaking UE4 Language

- Principles
  - KISS, YAGNI
  - Composition vs. inheritance
  - Avoid tight coupling of code and modules
  - Many trivial instead of few complicated components
- Design Patterns
  - SOLID
  - Hollywood Principle (especially for Slate & game code)
  - GOF, EIP

Unreal Engine 4

Speaking UE4 Language

Initial	Stands for	Concept
S	SRP	Single responsibility principle a class should have only a single responsibility (i.e. only one potential change in the software's specification should be able to affect the specification of the class)
O	OCP	Open/closed principle “software entities … should be open for extension, but closed for modification.”
L	LSP	Liskov substitution principle “objects in a program should be replaceable with instances of their subtypes without altering the correctness of that program.” See also design by contract.
I	ISP	Interface segregation principle “many client-specific interfaces are better than one general-purpose interface.”
D	DIP	Dependency inversion principle one should “Depend upon Abstractions. Do not depend upon concretions.”

Unreal Engine 4
Speaking UE4 Language

- Prefixes for All Types
  - U – UObject derrived class, i.e. UTexture
  - A – AActor derrived class, i.e. AGameMode
  - F – All other classes and structs, i.e. FName, FVector
  - T – Template, i.e. TArray, TMap, TQueue
  - I – Interface class, i.e. ITransaction
  - E – Enumeration type, i.e. ESelectionMode
  - b – Boolean value, i.e. bEnabled
- PascalCase
  - Function names and function parameters, too
  - Even local and loop variables!

Unreal Engine 4
Concurrency & Parallelism

- Concurrency
  - Atomics
  - Locking
  - Signaling & Waiting
  - Waiting
  - Containers

Unreal Engine 4

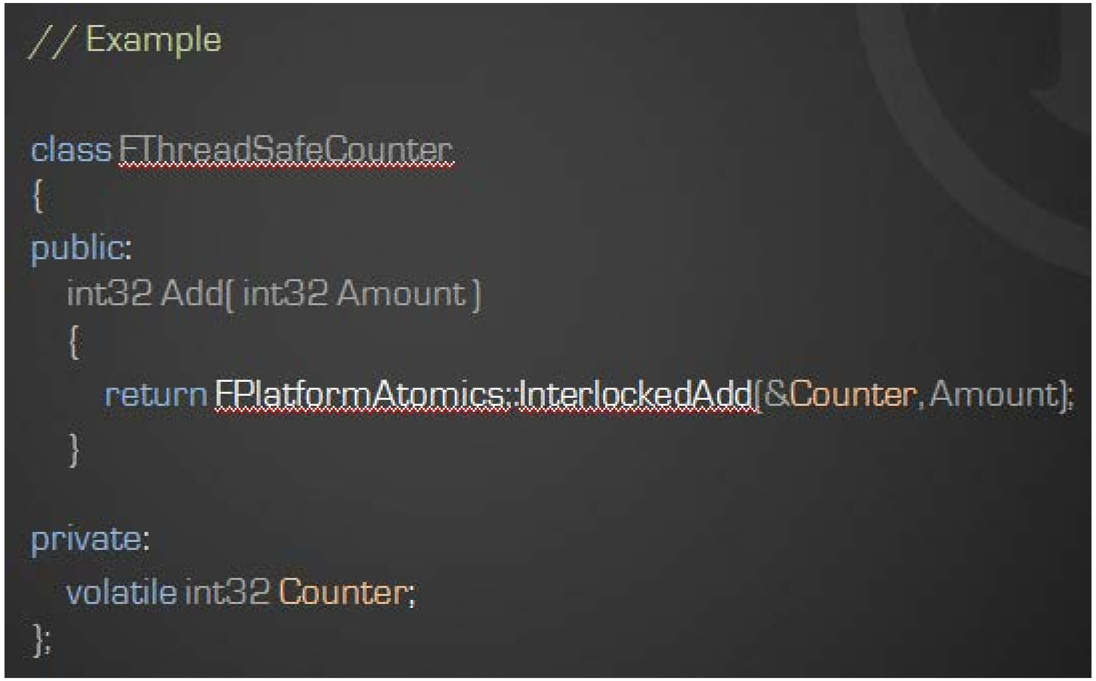
Concurrency

- Atomics
  - FPlatformAtomics
    - InterlockedAdd
    - InterlockedCompareExchange (-Pointer)
    - InterlockedDecrement (-Increment)
    - InterlockedExchange (-Pointer)
  - *FPlatformAtomics is "typedefed by platform"*

Unreal Engine 4

Concurrency

- Atomics



```
// Example

class FThreadSafeCounter
{
public:
    int32 Add( int32 Amount )
    {
        return FPlatformAtomics::InterlockedAdd(&Counter, Amount);
    }

private:
    volatile int32 Counter;
};
```

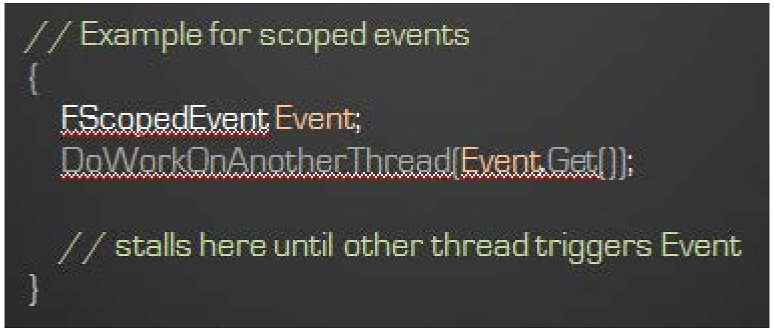
Unreal Engine 4
Concurrency

- Locking
  - Critical Sections
    - FCriticalSection implements synchronization object
    - FScopeLock for scope level locking using a critical section
    - Fast if the lock is not activated
  - Spin Locks
    - FSpinLock can be locked and unlocked
    - Sleeps or spins in a loop until unlocked
    - Default sleep time is 0.1 seconds

Unreal Engine 4

Concurrency

- Signaling & Waiting
  - FEvent
    - Blocks a thread until triggered or timed out
    - Frequently used to wake up worker threads
  - FScopedEvent
    - Wraps an FEvent that blocks on scope exit



```
// Example for scoped events
{
   FScopedEvent Event;
   DoWorkOnAnotherThread(Event.Get());

   // stalls here until other thread triggers Event
}
```

Unreal Engine 4

Concurrency & Parallelism

- Containers
  - General Thread-safety
    - Most containers (TArray, TMap, etc.) are not thread-safe
    - Use synchronization primitives in your own code where needed
  - TLockFreePointerList
    - Lock free
    - Used by Task Graph system
  - TQueue
    - Uses a linked list under the hood
    - Lock and contention free for SPSC
    - Lock free for MPSC

Unreal Engine 4

Concurrency & Parallelism

- Parallelism
  - Threads
  - Task Graph
  - Processes
  - Messaging

Unreal Engine 4
Concurrency & Parallelism

- Threads
  - FRunnable
    - Platform agnostic interface
    - Implement Init(), Run(), Stop() and Exit() in your sub-class
    - Launch with FRunnableThread::Create()
    - FSingleThreadRunnable when multi-threading is disabled
  - FQueuedThreadPool
    - Carried over from UE3 and still works the same way
    - Global general purpose thread pool in GThreadPool
    - Not lock free

Unreal Engine 4

Concurrency & Parallelism

- Threads
  - Game Thread
    - All game code, Blueprints and UI
    - UObjects are not thread-safe!
  - Render Thread
    - Proxy objects for Materials, Primitives, etc.
  - Stats Thread
    - Engine performance counters

Unreal Engine 4

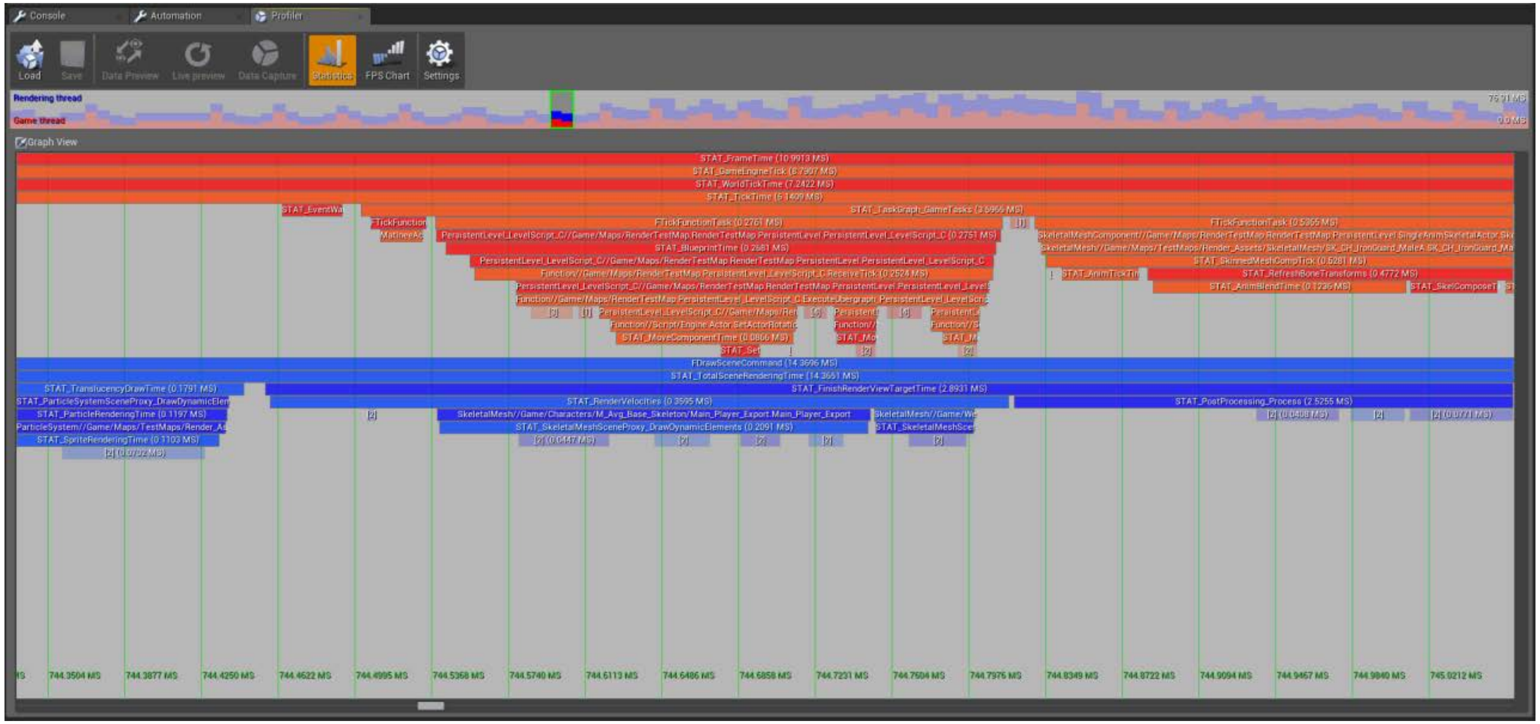
Concurrency & Parallelism

- Threads
  - Task Based Multi-Threading
    - Small units of work are pushed to available worker threads
    - Tasks can have dependencies to each other
    - Task Graph will figure out order of execution
    - Used by an increasing number of systems
  - Animation evaluation
    - Message dispatch and serialization in Messaging system
    - Object reachability analysis in garbage collector
    - Render commands in Rendering sub-system
    - Various tasks in Physics sub-system
    - Defer execution to a particular thread

Unreal Engine 4

Concurrency & Parallelism

- Threads


Console
Automation
Profiler
Load
Save
Data Preview
Live preview
Data Capture
Statistics
FPS Chart
Settings
Rendering thread
Game thread
Graph View
STAT_FrameTime (10.9913 MS)
STAT_GameEngineTick (8.7907 MS)
STAT_WorldTickTime (7.2422 MS)
STAT_TickTime (6.1409 MS)
STAT_TaskGraph_GameTasks (3.5966 MS)
FTickFunctionTask (0.2761 MS)
FTickFunctionTask (0.5365 MS)
STAT_BlueprintTime (0.2681 MS)
STAT_SkinnedMeshCompTick (0.5281 MS)
STAT_RefreshBoneTransforms (0.4772 MS)
STAT_AnimBlendTime (0.1236 MS)
STAT_MoveComponentTime (0.0866 MS)
FDrawSceneCommand (14.3696 MS)
STAT_TotalSceneRenderingTime (14.3651 MS)
STAT_TranslucencyDrawTime (0.1791 MS)
STAT_FinishRenderViewTargetTime (2.8931 MS)
STAT_RenderVelocities (0.3595 MS)
STAT_PostProcessing_Process (2.5255 MS)
STAT_ParticleRenderingTime (0.1197 MS)
STAT_SkeletalMeshSceneProxy_DrawDynamicElements (0.2091 MS)
STAT_SpriteRenderingTime (0.1103 MS)
744.3504 MS
744.3877 MS
744.4250 MS
744.4622 MS
744.4995 MS
744.5368 MS
744.5740 MS
744.6113 MS
744.6486 MS
744.6858 MS
744.7231 MS
744.7604 MS
744.7976 MS
744.8349 MS
744.8722 MS
744.9094 MS
744.9467 MS
744.9840 MS
745.0212 MS

Unreal Engine 4

Concurrency & Parallelism

- Processes
  - FPlatformProcess
    - CreateProc() executes an external program
    - LaunchURL() launches the default program for a URL
    - IsProcRunning() checks whether a process is still running
    - Plus many other utilities for process management
  - FMonitoredProcess
    - Convenience class for launching and monitoring processes
    - Event delegates for cancellation, completion and output

Unreal Engine 4

Concurrency & Parallelism

- Messaging
  - Unreal Message Bus (UMB)
    - Zero configuration intra- and inter-process communication
    - Request-Reply and Publish-Subscribe patterns supported
    - Messages are simple UStructs
  - Transport Plug-ins
    - Seamlessly connect processes across machines
    - Only implemented for UDP right now (prototype)

Game Engine

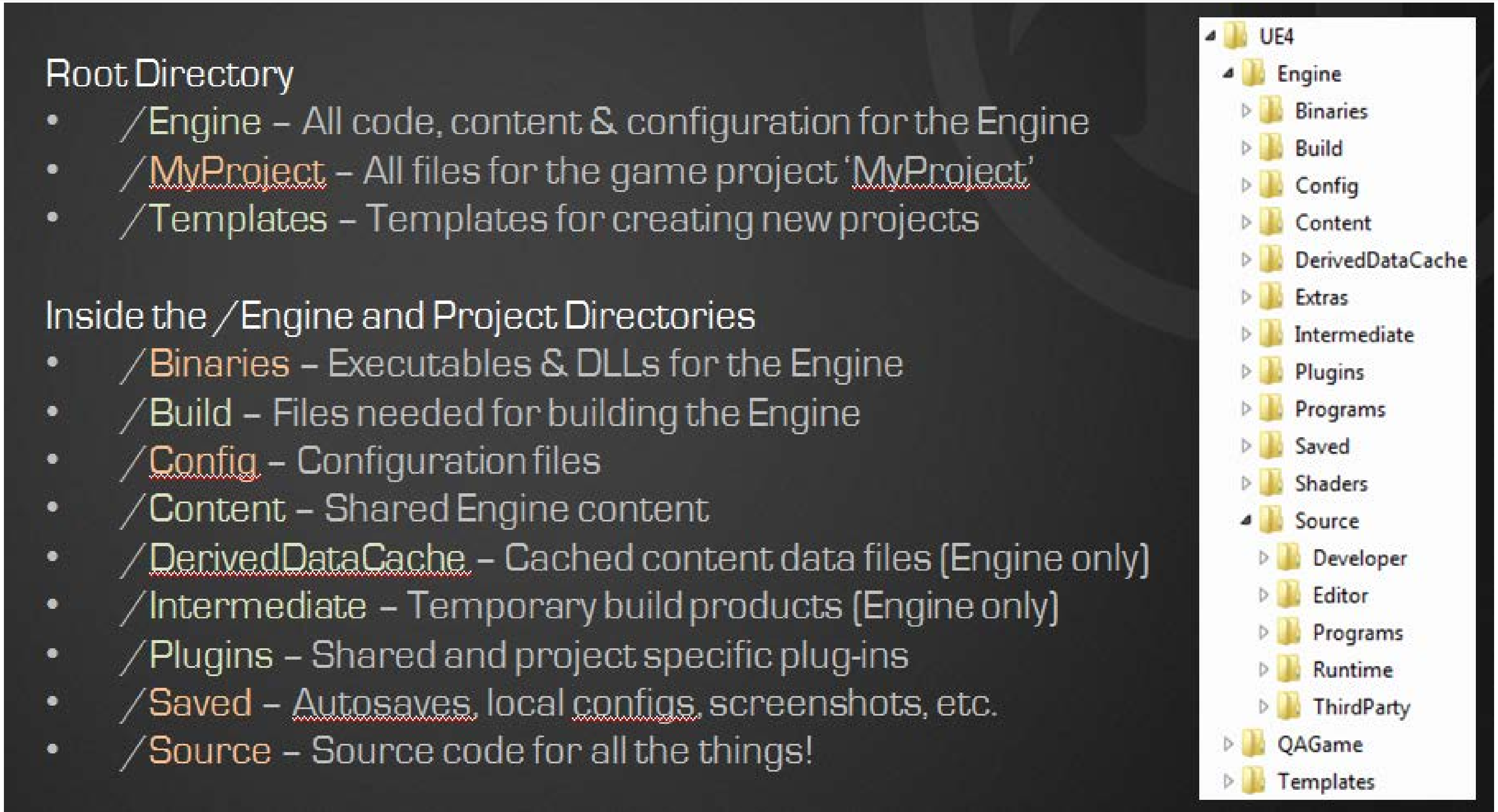
Thank you for you attention!

THAT'S IT FOR TODAY!

LABS => HLSL Part III (last one)

Unreal Engine 4

Solution Structure



Root Directory

- /Engine – All code, content & configuration for the Engine
- /MyProject – All files for the game project 'MyProject'
- /Templates – Templates for creating new projects

Inside the /Engine and Project Directories

- /Binaries – Executables & DLLs for the Engine
- /Build – Files needed for building the Engine
- /Config – Configuration files
- /Content – Shared Engine content
- /DerivedDataCache – Cached content data files (Engine only)
- /Intermediate – Temporary build products (Engine only)
- /Plugins – Shared and project specific plug-ins
- /Saved – Autosaves, local configs, screenshots, etc.
- /Source – Source code for all the things!

```
UE4
  Engine
    Binaries
    Build
    Config
    Content
    DerivedDataCache
    Extras
    Intermediate
    Plugins
    Programs
    Saved
    Shaders
    Source
      Developer
      Editor
      Programs
      Runtime
      ThirdParty
  QAGame
  Templates
```

Game Engine

Thanks you for you attention!

Some interesting stuff: