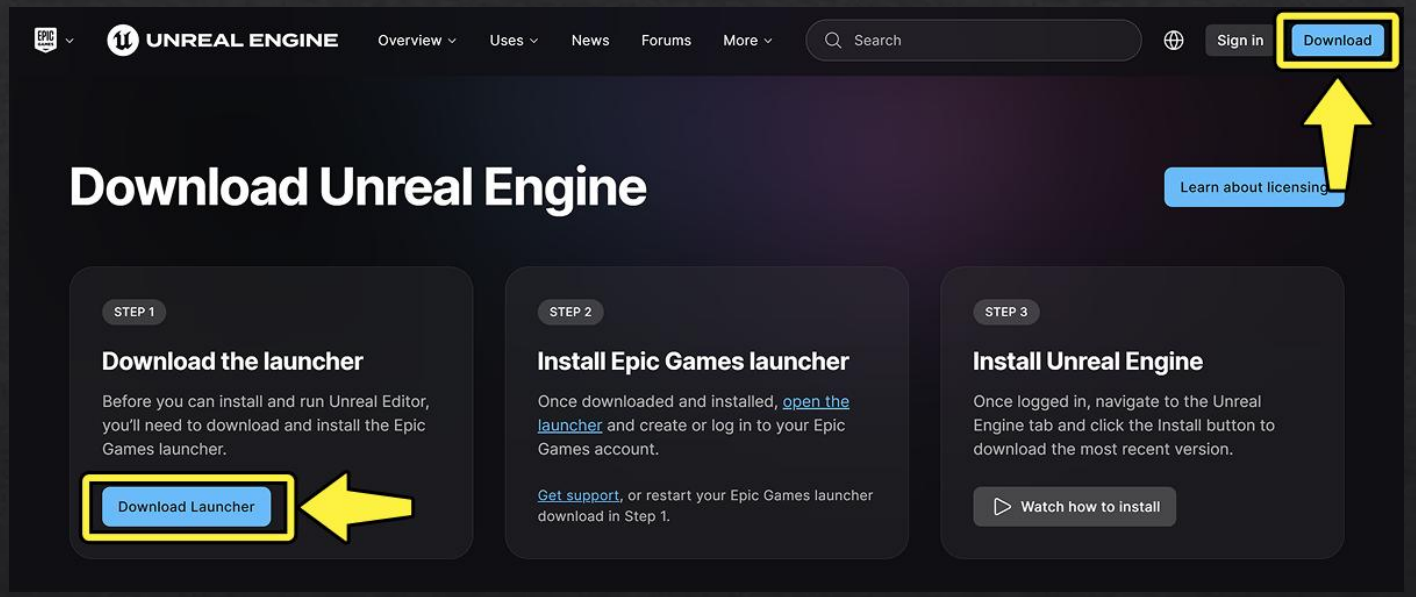
Introduction To Unreal Engine

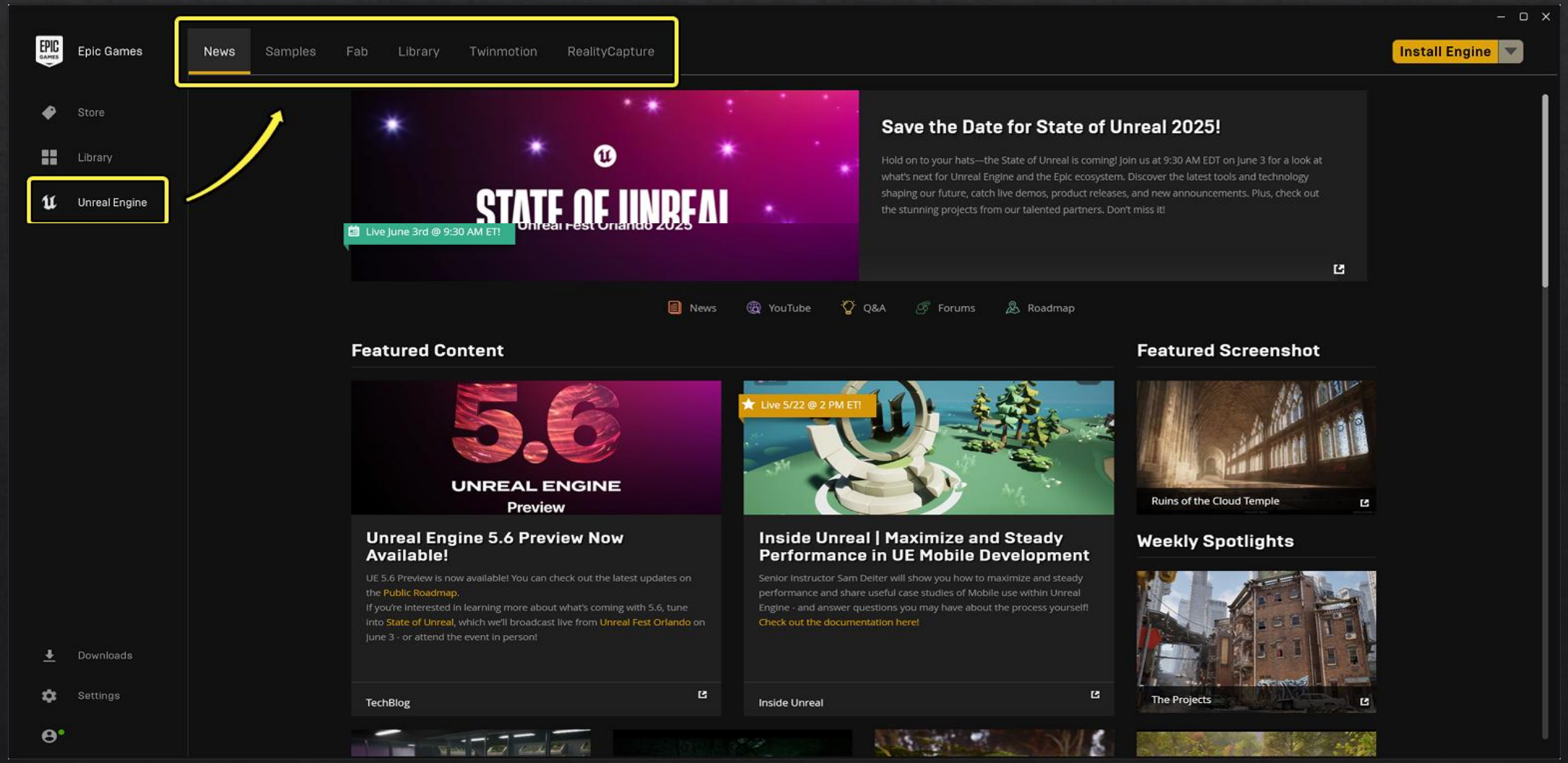
Mohamad Shaaban, Yonas Tefera

INSTALLATION

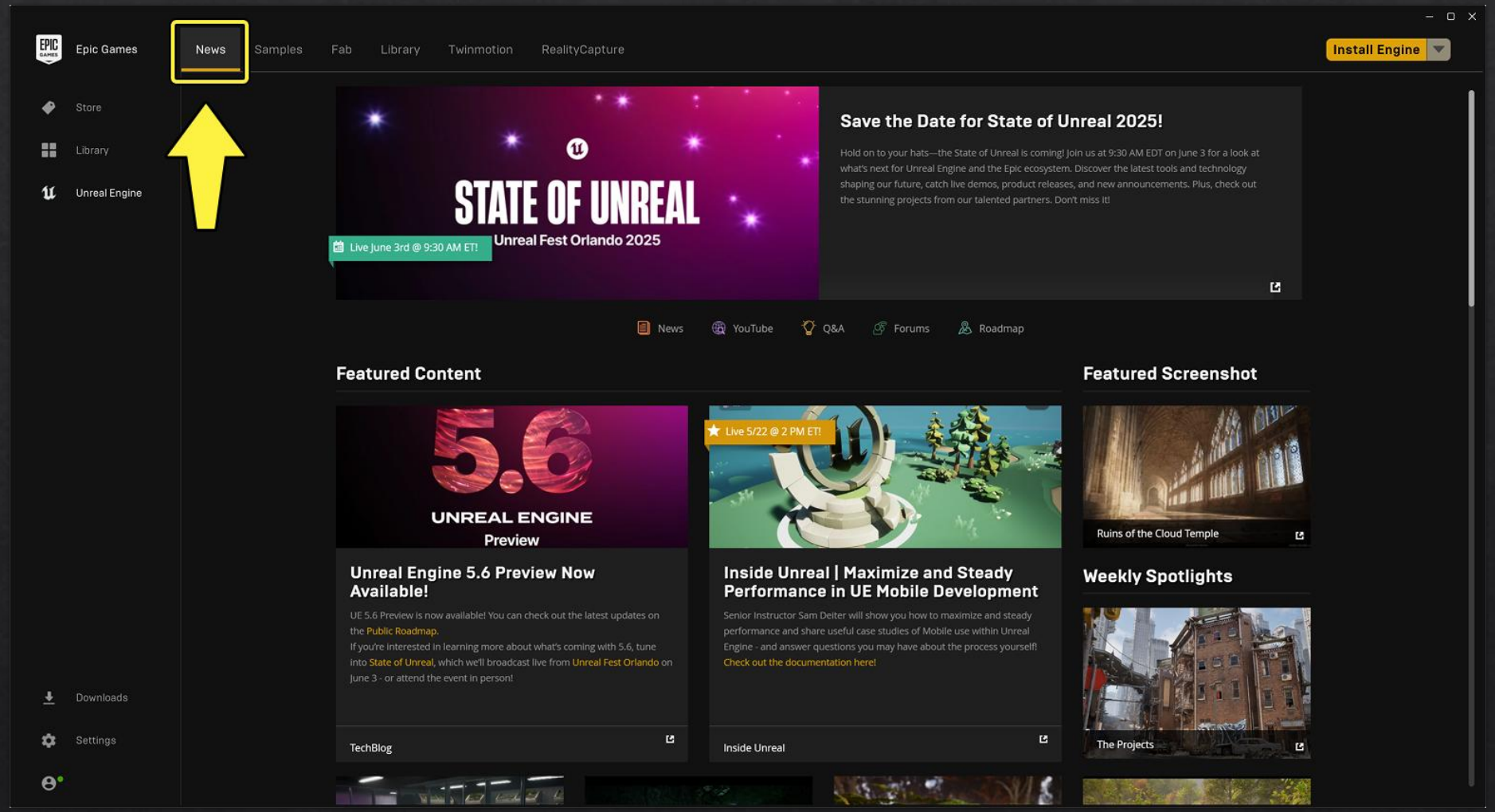
- Access the Download Unreal Engine page
- Download Epic Games Launcher



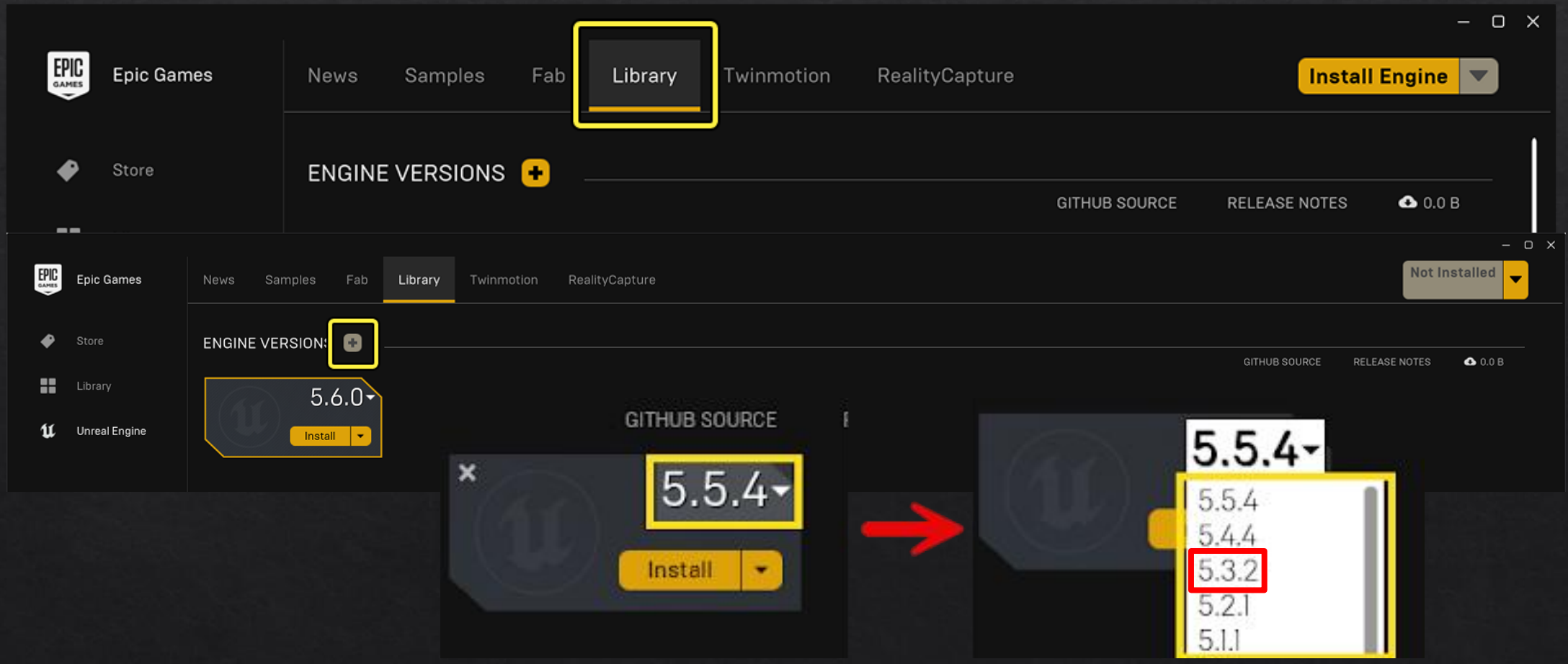
Epic Games Launcher



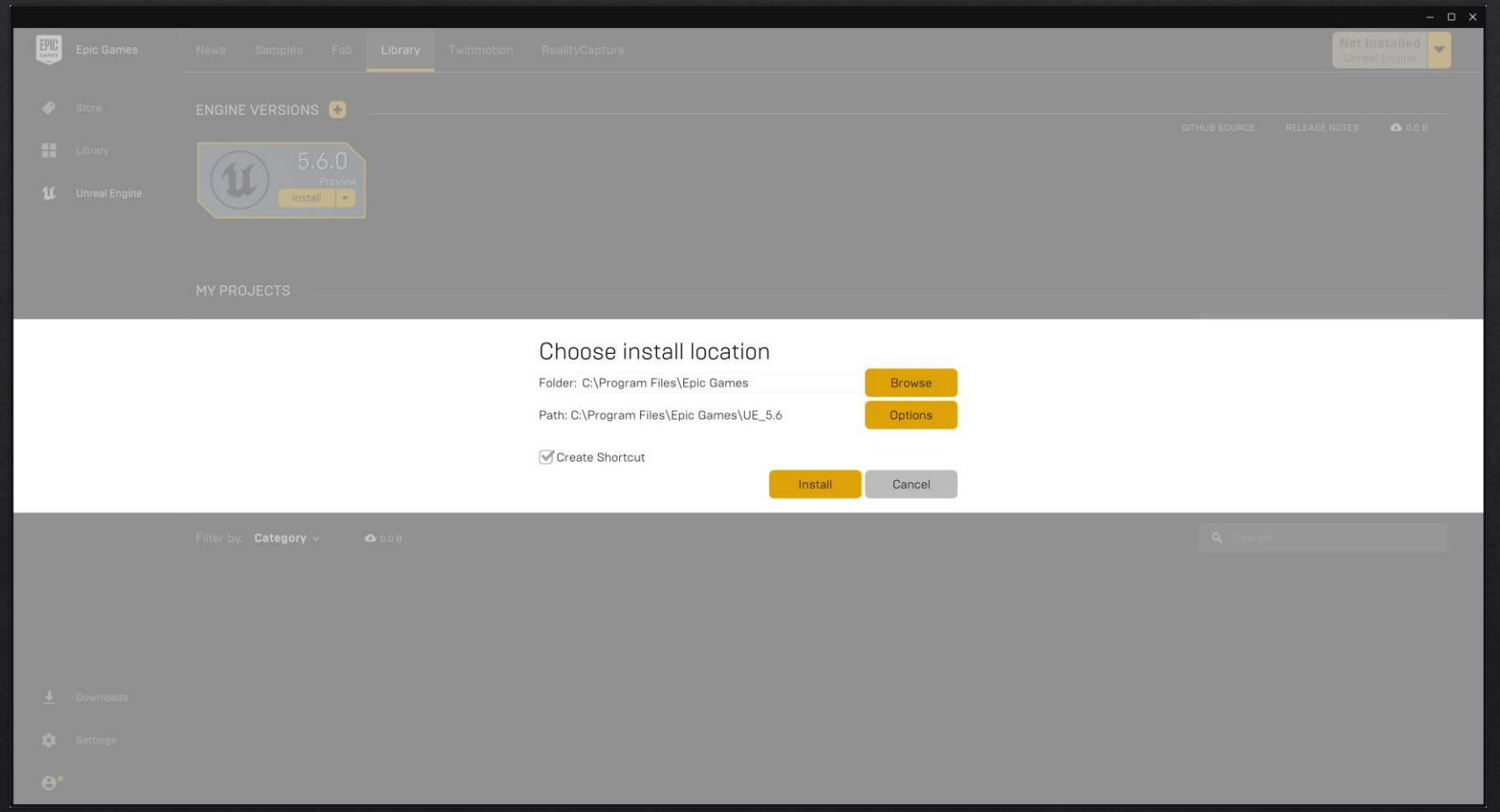
News Tab


Epic Games
News
Samples
Fab
Library
Twinmotion
RealityCapture
Install Engine
Store
Library
Unreal Engine
STATE OF UNREAL
Unreal Fest Orlando 2025
Live June 3rd @ 9:30 AM ET!
Save the Date for State of Unreal 2025!
Hold on to your hats—the State of Unreal is coming! Join us at 9:30 AM EDT on June 3 for a look at what's next for Unreal Engine and the Epic ecosystem. Discover the latest tools and technology shaping our future, catch live demos, product releases, and new announcements. Plus, check out the stunning projects from our talented partners. Don't miss it!
News
YouTube
Q&A
Forums
Roadmap
Featured Content
5.6
UNREAL ENGINE
Preview
Unreal Engine 5.6 Preview Now Available!
UE 5.6 Preview is now available! You can check out the latest updates on the Public Roadmap.
If you're interested in learning more about what's coming with 5.6, tune into State of Unreal, which we'll broadcast live from Unreal Fest Orlando on June 3 - or attend the event in person!
TechBlog
Live 5/22 @ 2 PM ET!
Inside Unreal | Maximize and Steady Performance in UE Mobile Development
Senior Instructor Sam Deiter will show you how to maximize and steady performance and share useful case studies of Mobile use within Unreal Engine - and answer questions you may have about the process yourself!
Check out the documentation here!
Inside Unreal
Featured Screenshot
Ruins of the Cloud Temple
Weekly Spotlights
The Projects
Downloads
Settings

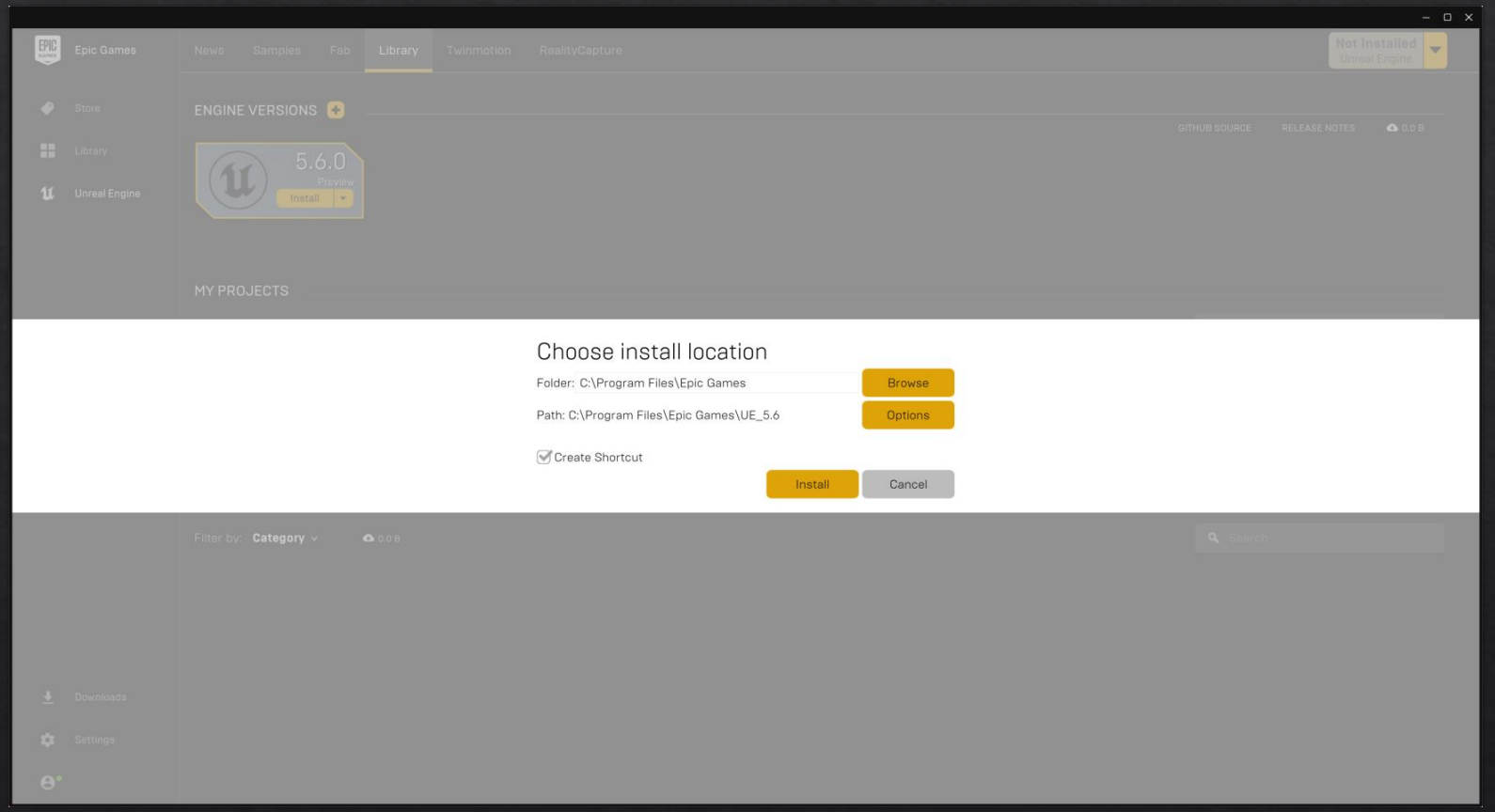
Installation



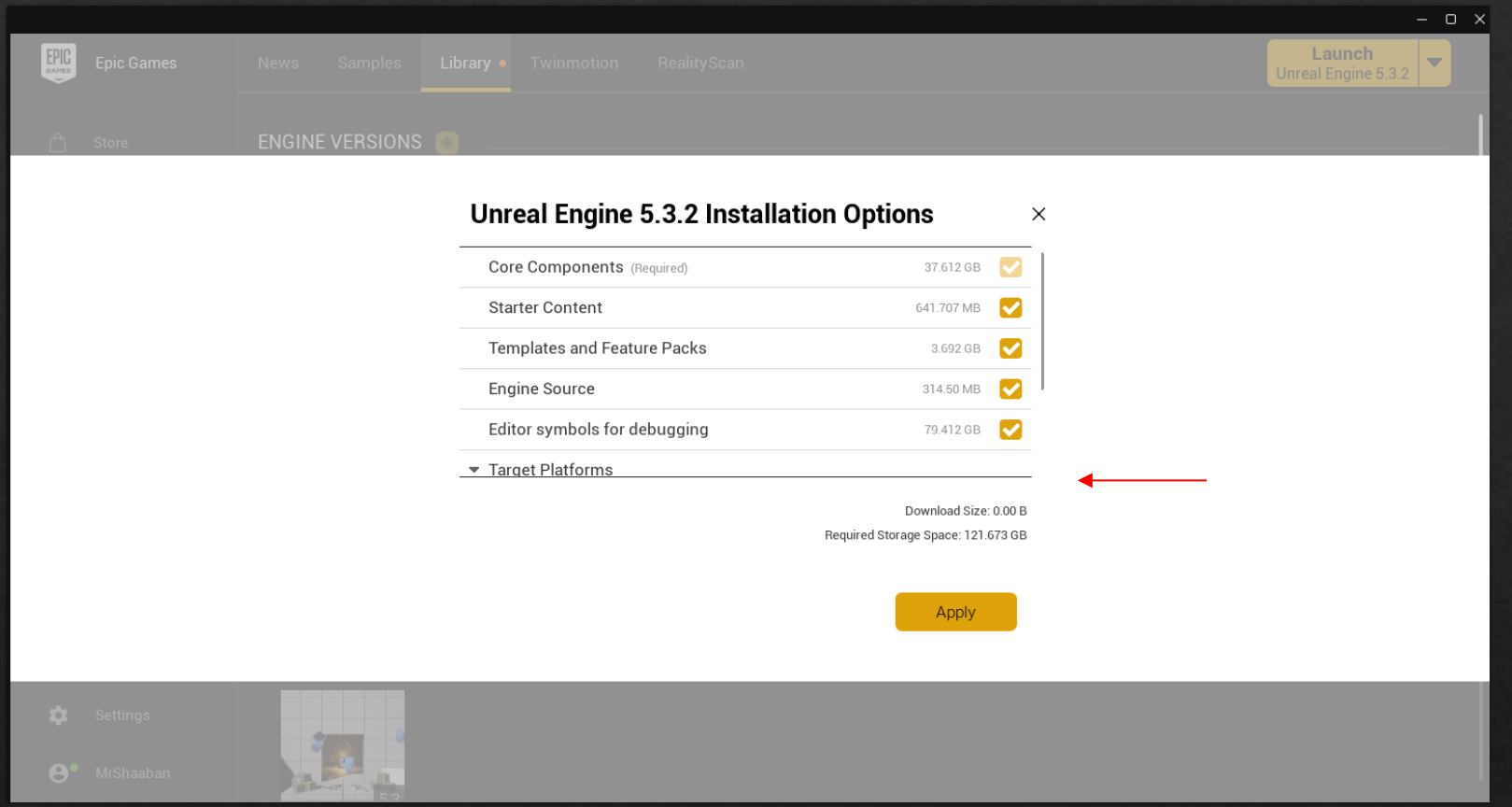
Installation



Installation



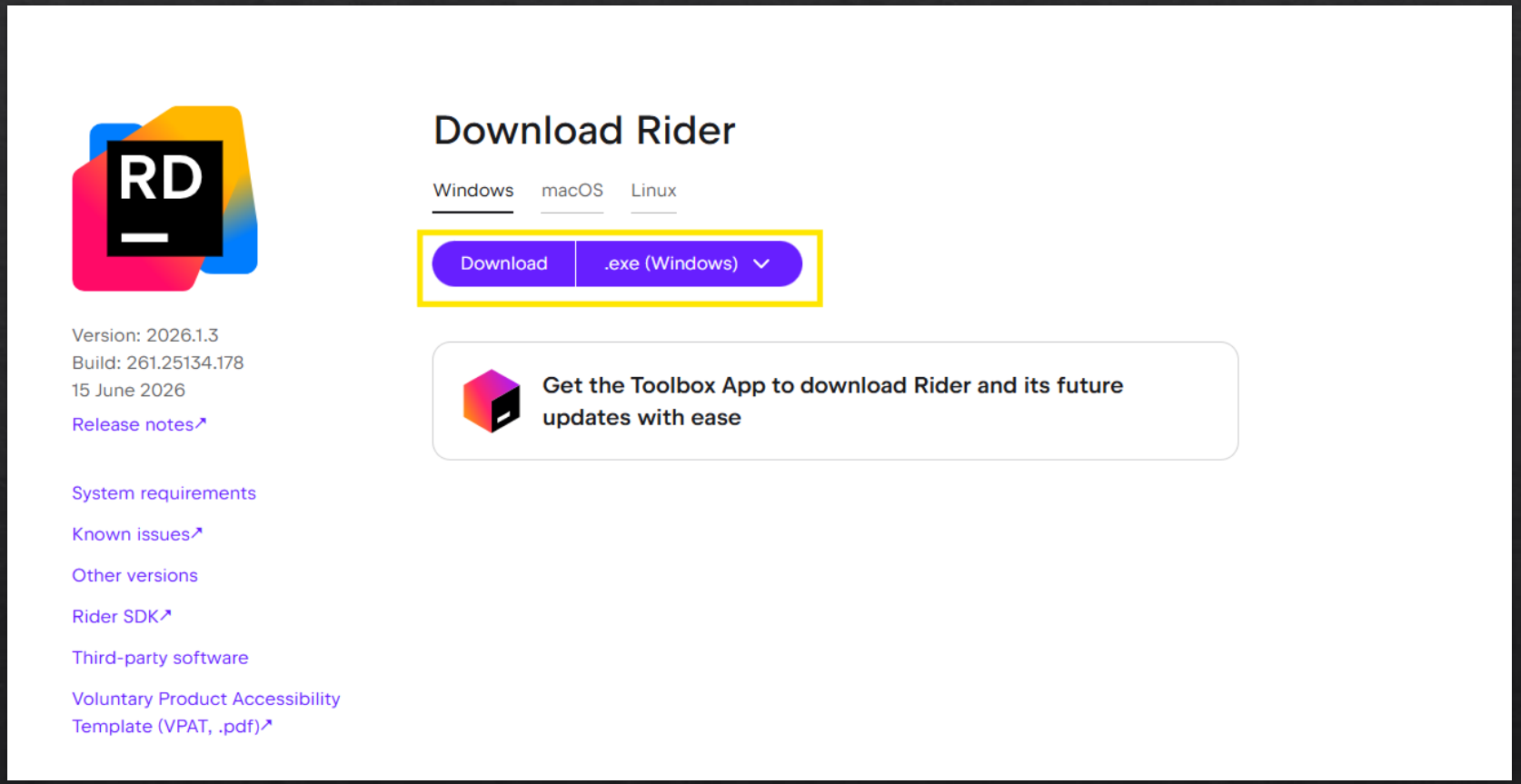
Installation



Rider for Unreal Engine

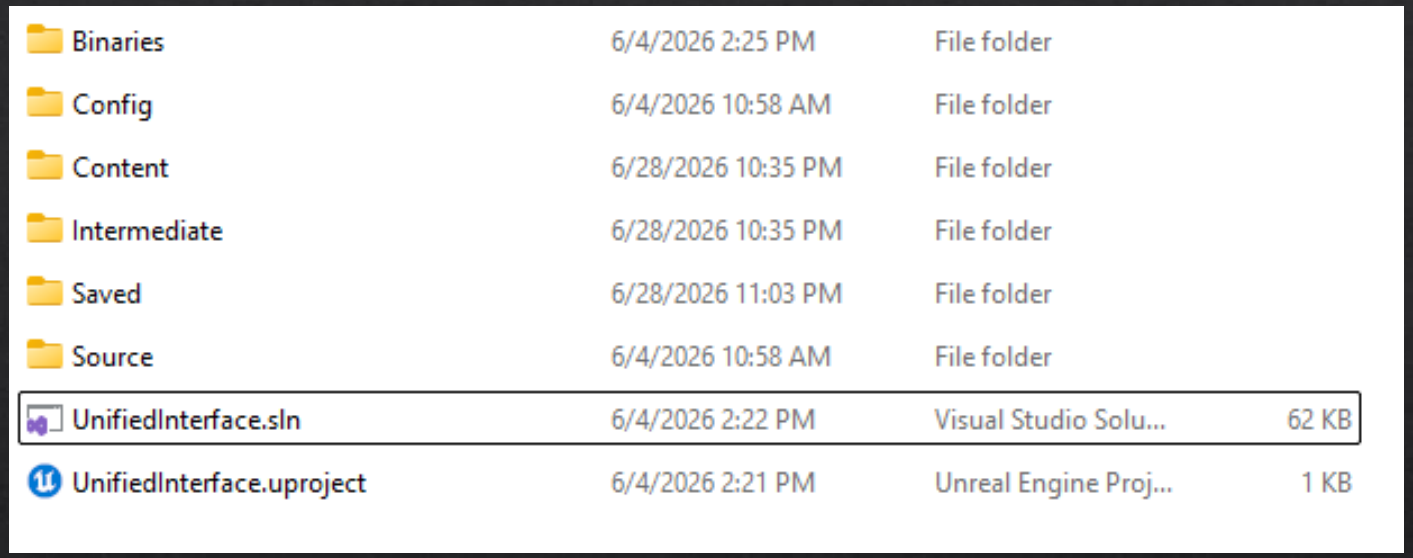
- Access the Download page

Rider for Unreal Engine



Unreal Project Structure

- **.uproject:** JSON descriptor for engine version and plugins.
- **.sln:** Project solution for IDEs.
- **Config/:** Project settings per platform/category.
- **Content/:** All game assets (.uasset, .umap).
- **Source/:** C++ header and implementation files.

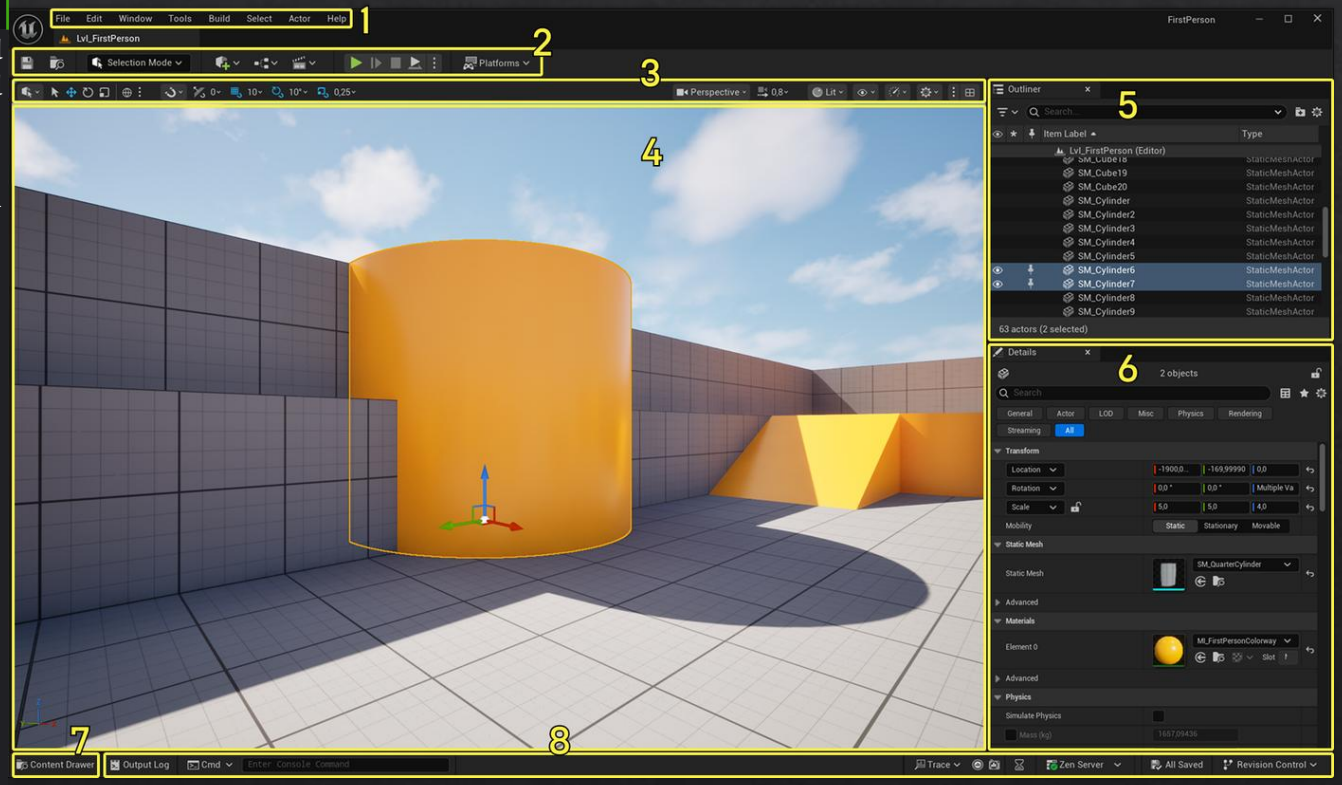


Unreal Project Structure

Folder	Status	Description
Source / Content	Static	Must be shared via Version Control (Git/P4).
Config	Static	Essential project-wide settings.
Binaries	Generated	Compiled executables and DLLs. Regenerated.
Intermediate	Generated	Build artifacts. Safe to delete.
Saved	Generated	Logs, local backups, and autosaves.

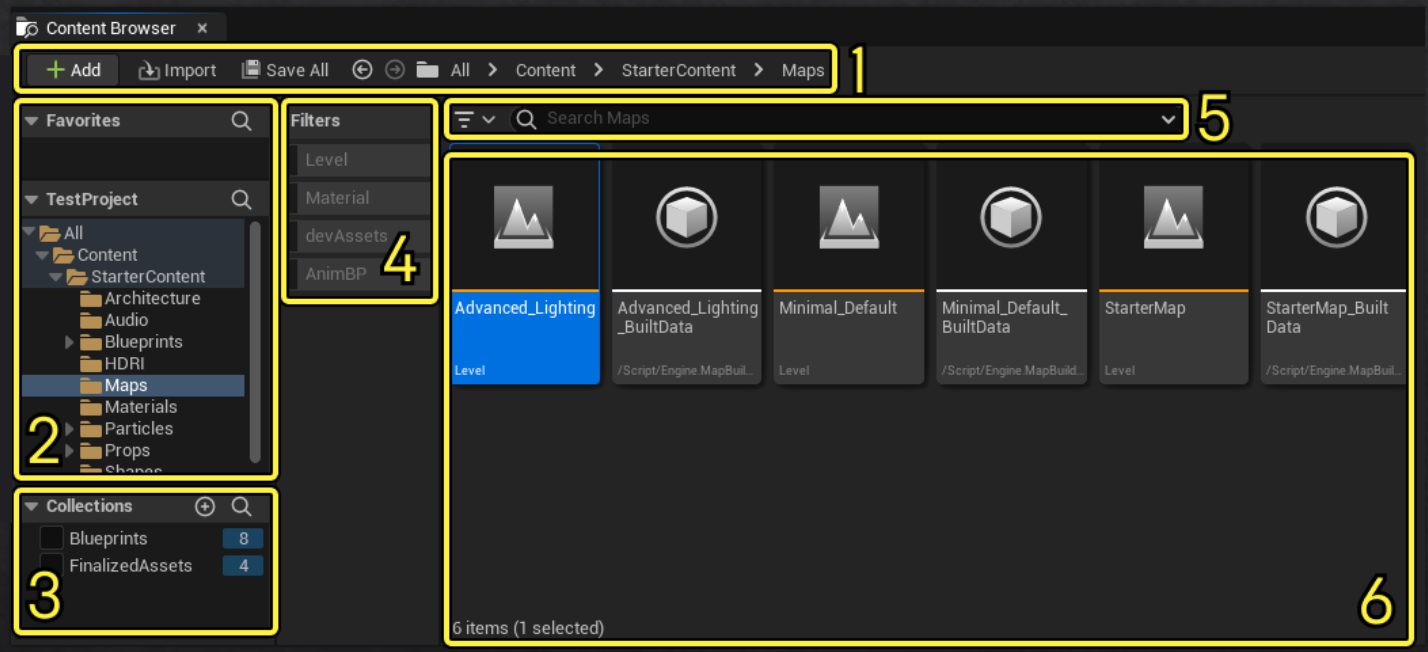
Unreal Editor

#	Name	Description
1	Menu Bar	Uses these menus to access common application actions, like saving, and creating new levels. You'll also find options for opening editor windows and tools that are useful for specific functions like debugging and more.
2	Main Toolbar	Contains shortcuts for some of the most common tools and editors in Unreal Engine, as well as shortcuts to enter Play mode (run your game inside Unreal Editor) and to deploy your project to other platforms.
4	Level Viewport	Displays the contents of your Level, such as Cameras, Actors, Static Meshes, and so on.
5	Outliner	Displays a hierarchical tree view of all content in your Level.
6	Details panel	Appears when you select an Actor. Displays various properties for that Actor, such as its Transform (position in the Level), Static Mesh, Material, and physics settings. This panel displays different settings depending on what you select in the Level Viewport.
7	Content Drawer	Opens the Content Drawer, a temporary Content Browser window, from which you can access all of the Assets in your Project.
8	Bottom Toolbar	Contains shortcuts to the Command Console, Output Log, and Derived Data functionality. Also displays source control status.



Content Browser

Number	Name
1	Navigation Bar
2	Sources Panel
4	Filters
5	Search Bar
6	Asset View



UNREAL UNITS (UU)

1 Unreal Unit = **1** Centimeter

Gravity, Physics, and Lighting depend on this scale.

A standard door is approx. 210cm (210 UU) tall.

UNREAL Coordinate System

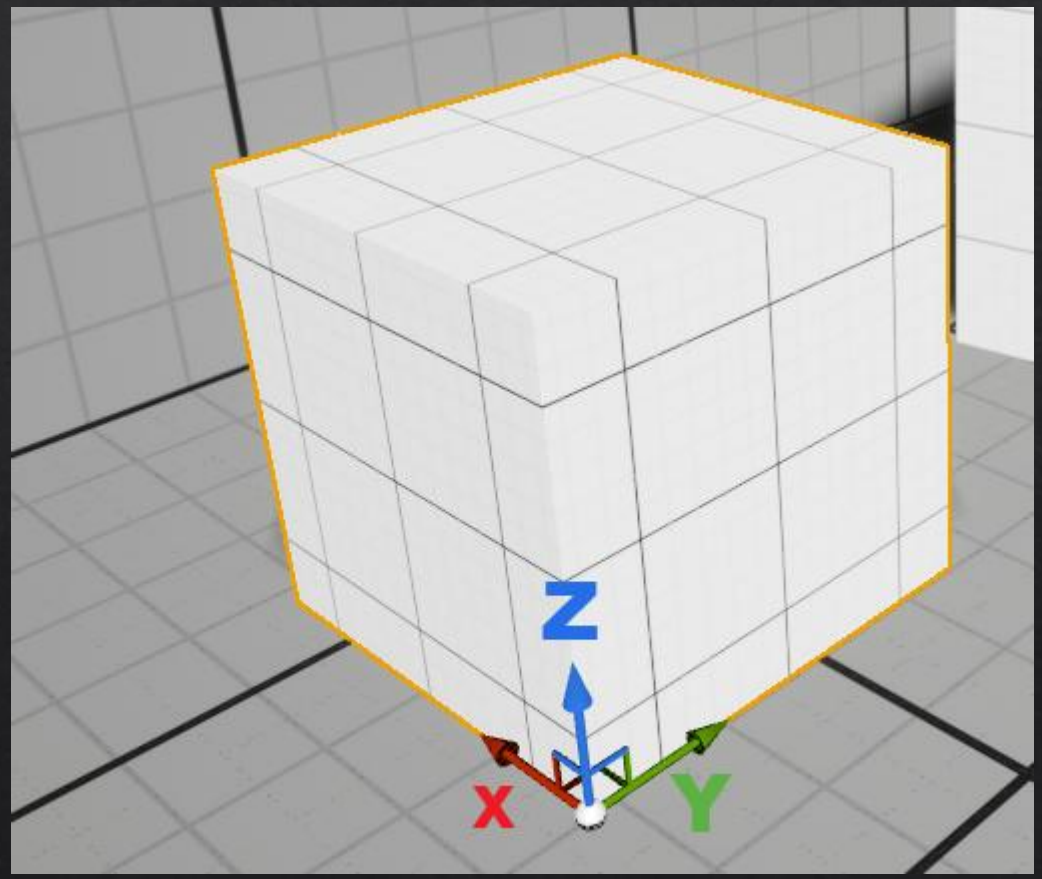
LHS Orientation

X-Axis (Red): Forward

Y-Axis (Green): Right

Z-Axis (Blue): Up

Rotation: Yaw (Z), Pitch (Y), Roll (X).

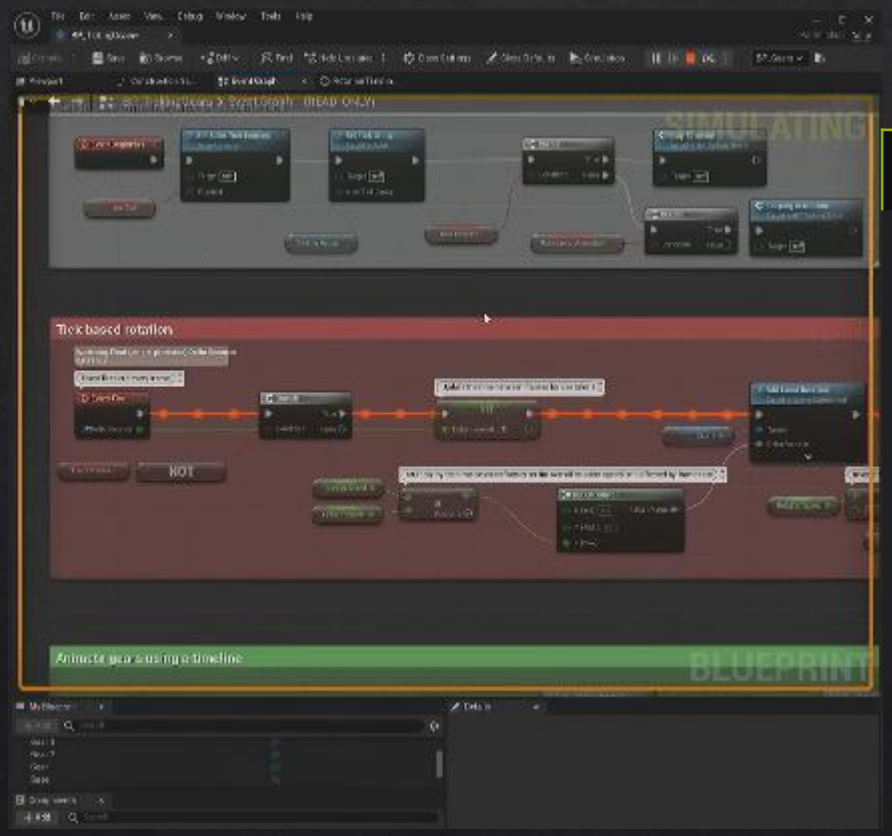


Unreal Engine Plugins

Enable or disable engine features modularly. Plugins can contain C++ code, shaders, and content(Edit → Plugins).

- Built-in Epic plugins.
- Marketplace acquired tools.
- Custom project-local plugins.

Programming Inside Unreal



Blueprint

Blueprint	C++
Scripting Language	Programming Language
Faster iteration	Faster runtime
Broader Accessibility	Broader Access
Broader Extensibility	Greater Discoverability

```cpp
UCLASS()
No derived blueprint classes
class UNIFIEDINTERFACE_API UUIWindowManagerSubsystem : public UGameInstanceSubsystem
{
    GENERATED_BODY()

public:
    UFUNCTION(BlueprintCallable, Category = "Window Management")
    void OpenSecondWindow(UUserWidget* HUD);  1 blueprint usage

    UFUNCTION(BlueprintCallable, Category = "Window Management")
    void CloseSecondWindow();  No blueprint usages

    UFUNCTION(BlueprintPure, Category = "Window Management")
    UUserWidget* GetSecondWindowHUD() const { return SecondWindowHUDInstance; }

protected:
    virtual void Deinitialize() override;

private:
    TSharedPtr<SWindow> SecondWindow;

    // Pointer to the UMG Widget inside the window
    UPROPERTY()
    UUserWidget* SecondWindowHUDInstance;

};
```

C++

Unreal Framework

UObject: Base meta-data class (Garbage Collected).

AActor: Can be placed/spawned in a Level.

APawn: An Actor that can be "Possessed".

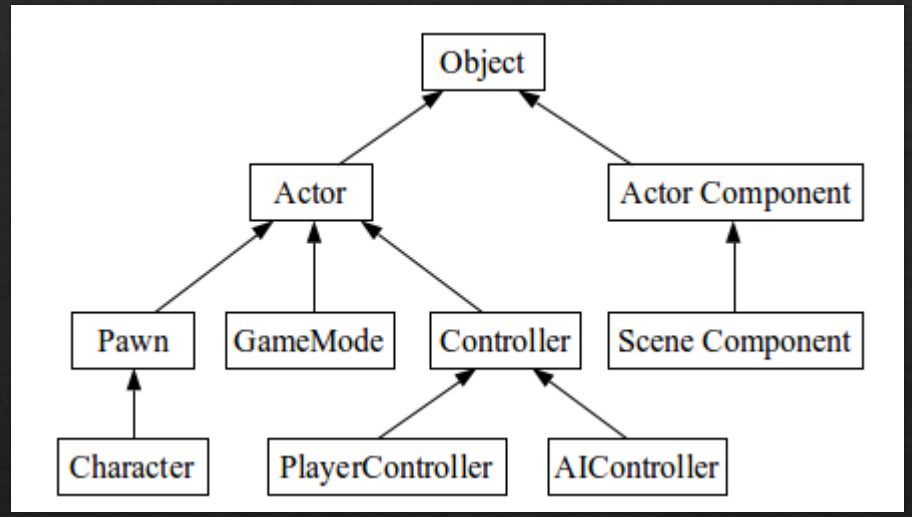
ACharacter: Pawn with walking/physics logic.

Player Controller: Possess a Pawn and control it.

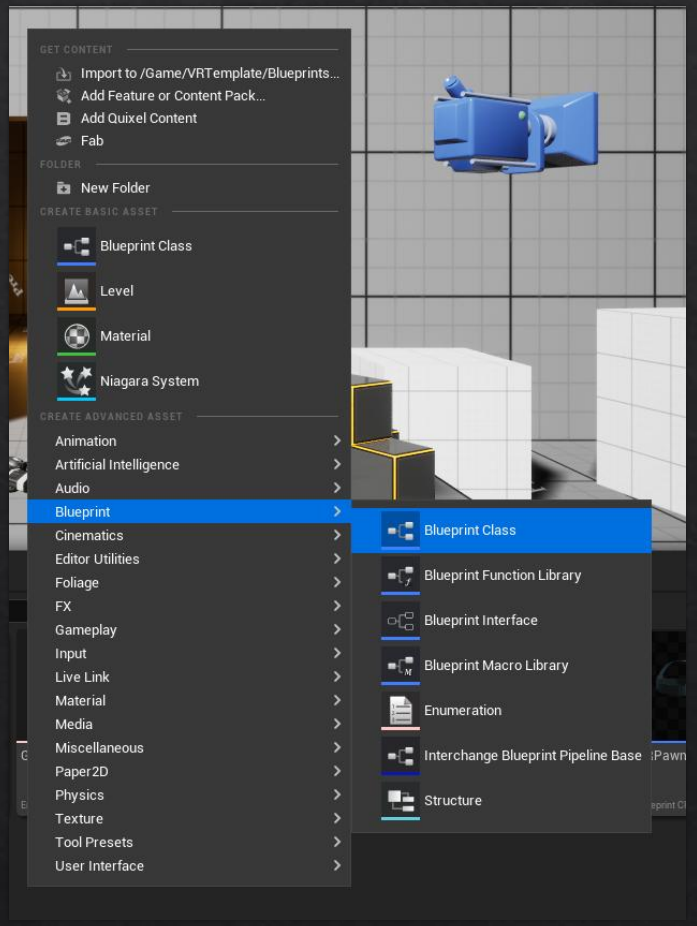
Actor Component: Extend Actor with new logic.

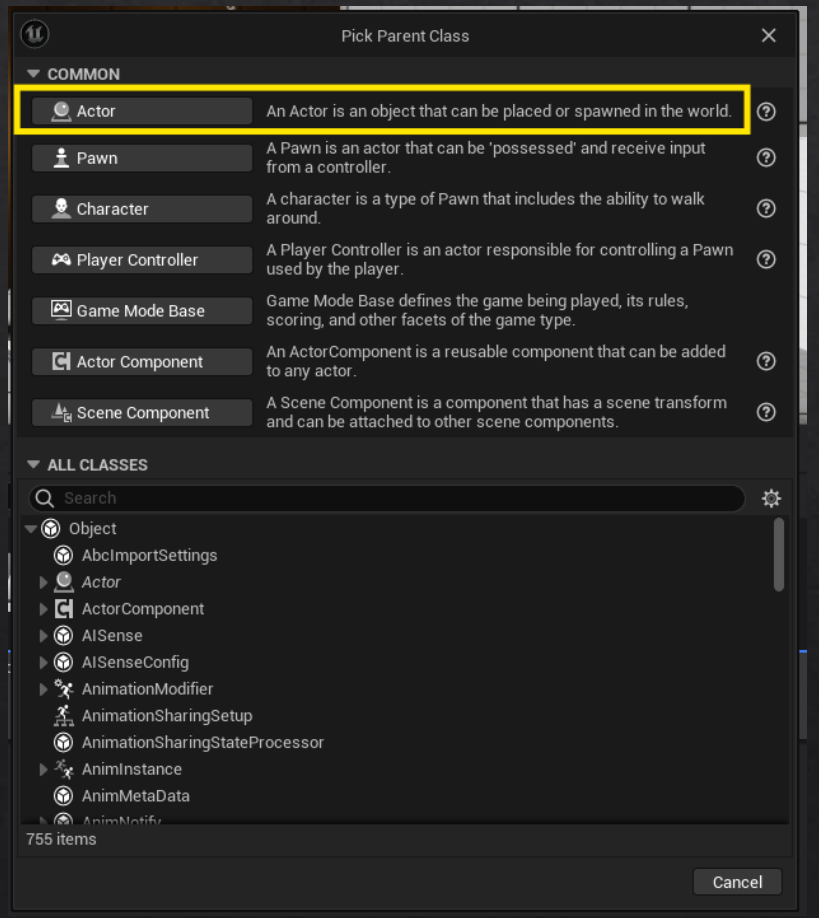
Scene Component: A component that can render.

GameMode: Defines the rules of the game.



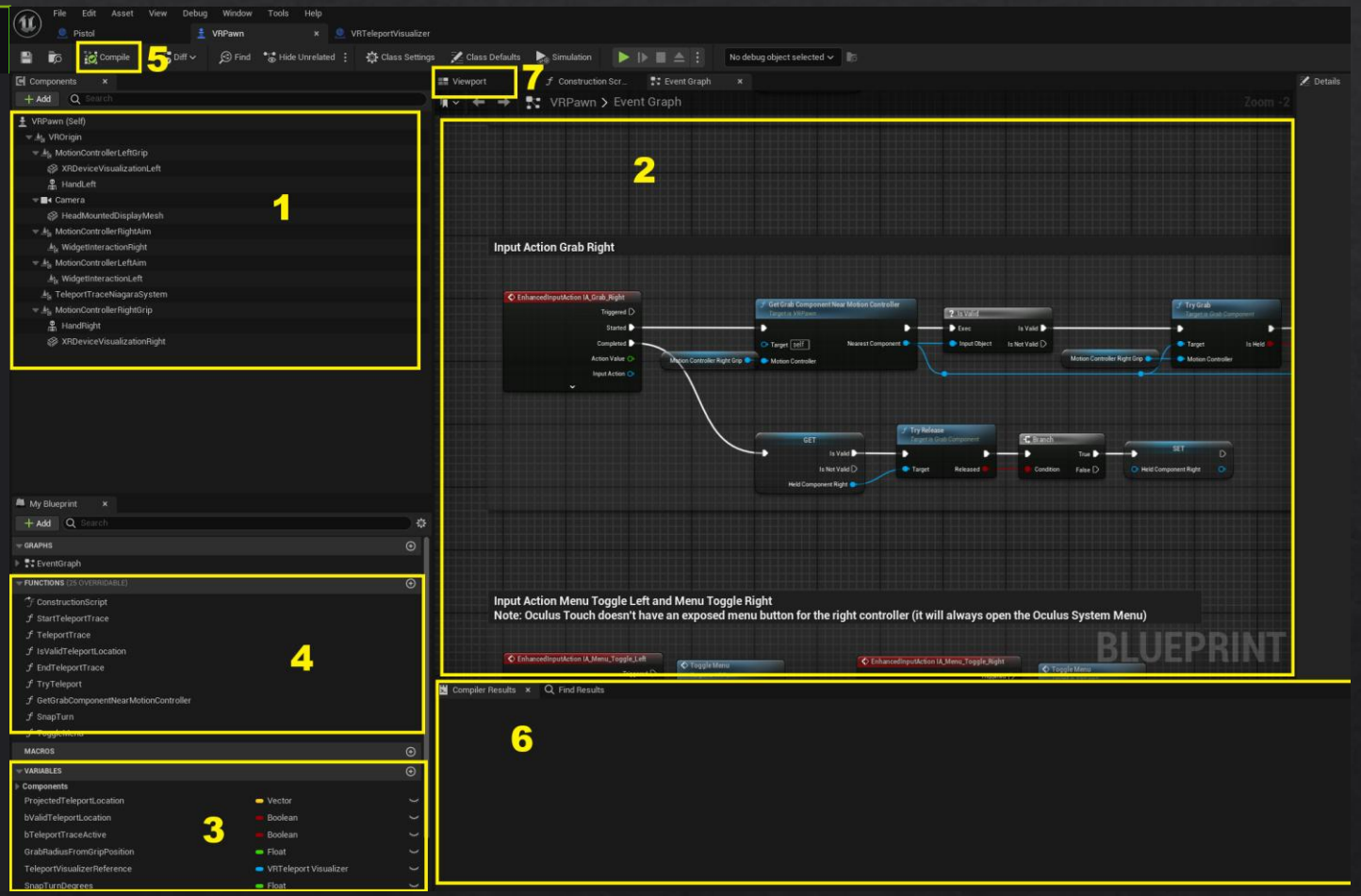
Creating Your First Actor





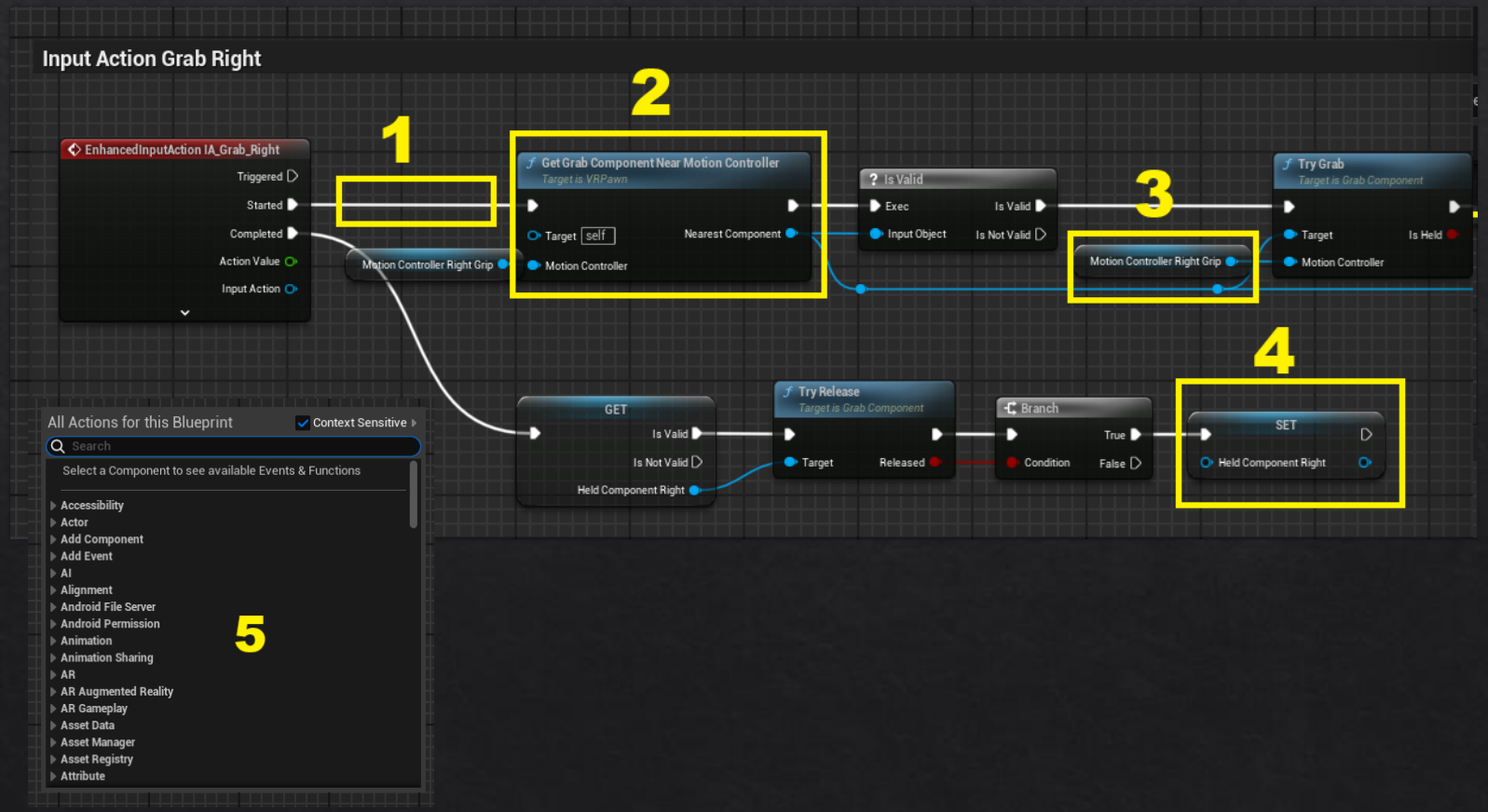
Blueprint Editor

#	Name	Description
1	Components	Add/Remove new components to extend the actor logic.
2	Event Graph	Create new Events(aka Function with no return) and manage your blueprint graphs.
3	Variables	Add/Remove variables such as Int32/Fstring.
4	Functions	Create new Blueprint functions that can be placed in the Event graph.
5	Compile	After finishing your logic, you must press compile.
6	Compiler Results	Shows compiling Logs/Errors.
7	Viewport	Can be used to visualize your actor.



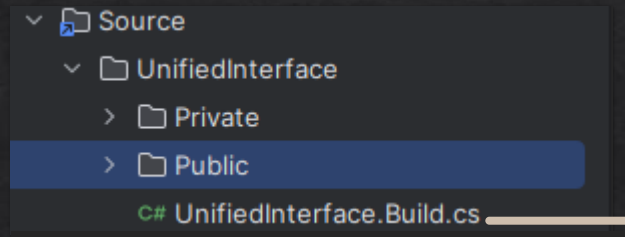
Blueprint Editor

#	Description
1	Execution Line.
2	A function to be executed. Inputs to the left/ Outputs to the right.
3	Variable getter.
4	Variable setter.
5	Search for a variable, function or an event.



Modules

- In Unreal Engine, a module is the fundamental building block of the engine's architecture. Both the engine itself and your game projects are constructed entirely from collections of these modules. Your game will have at least one module.
- Each module is structured as follows:




```
Source/
└── ExampleModule/                  <-- The Root Directory of the module
    │
    ├── ExampleModule.Build.cs      <-- The UBT compilation script
    │
    ├── Public/                     <-- Headers exposed to other modules
    │   ├── ExampleModule.h         <-- The Module Interface header
    │   ├── MyPublicClass.h         <-- A standard class header
    │   └── MyPublicStructs.h
    │
    └── Private/                    <-- Hidden implementation and internal headers
        ├── ExampleModule.cpp       <-- The Module Interface implementation
        ├── MyPublicClass.cpp       <-- Implementation of the public class
        ├── MyInternalHelper.h      <-- A header ONLY used inside this module
        └── MyInternalHelper.cpp
```

```
using UnrealBuildTool;

? usages  ? inheritors  Mohamad Shaaban *  ? extension methods  ? exposing APIs
public class UnifiedInterface : ModuleRules
{
    ? usages  ? overrides  Mohamad Shaaban *  ? extension methods  ? exposing APIs
    public UnifiedInterface(ReadOnlyTargetRules Target) : base(Target)
    {
        PCHUsage = PCHUsageMode.UseExplicitOrSharedPCHs;

        PublicDependencyModuleNames.AddRange( collection: new string[] { "Core", "CoreUObject", "Engine",

        PrivateDependencyModuleNames.AddRange( collection: new string[] {  });

        bEnableUndefinedIdentifierWarnings = false;

    }
}
```

Naming Convention

U & A Classes

U: Represents classes derived directly from UObject. Used for generic data or components.

A: Represents classes derived from Actor. These are objects that live in the physical game world.

F & E Types

F: Standard C++ structures (Structs).

E: Enumerations (Enums). Used to define specific lists of named constants for states or settings.

I Interfaces

I: Interfaces. Used primarily to hide implementation details and allow polymorphic interaction.

Unreal Reflection System

Unreal Header Tool

C++ doesn't natively support reflection. Unreal uses UHT to parse macros(**UCLASS/UFUNCTION/UPROPERTY**) and generate code(**#include "*.generated.h"**) to expose variables/functions to:

- Garbage Collection
- Blueprint Editor
- Serialization / Save Games
- Network Replication

```
#include "CoreMinimal.h"
#include "Subsystems/GameInstanceSubsystem.h"
#include "UIWindowManagerSubsystem.generated.h"

UCLASS()
No derived blueprint classes
class UNIFIEDINTERFACE_API UUIWindowManagerSubsystem : public UGameInstanceSubsystem
{
    GENERATED_BODY()

public:
    UFUNCTION(BlueprintCallable, Category = "Window Management")
    void OpenSecondWindow(UUserWidget* HUD);  1 blueprint usage

    UFUNCTION(BlueprintCallable, Category = "Window Management")
    void CloseSecondWindow();  No blueprint usages

    UFUNCTION(BlueprintPure, Category = "Window Management")
    UUserWidget* GetSecondWindowHUD() const { return SecondWindowHUDInstance; }

protected:
    virtual void Deinitialize() override;

private:
    TSharedPtr<SWindow> SecondWindow;

    // Pointer to the UMG Widget inside the window
    UPROPERTY()
    UUserWidget* SecondWindowHUDInstance;

};
```

Reflection Macros

Macro	Description
UCLASS	Used immediately above a class declaration. It tells the engine to generate reflection data for this class. The class must inherit from Uobject.
UPROPERTY	Placed above variable declarations. It registers the variable with Unreal's Garbage Collector (preventing memory leaks/crashes) and dictates how the variable interacts with the Editor and Blueprints.
UFUNCTION	Placed above method declarations. It exposes C++ functions to the engine, which is required for calling functions from Blueprints, binding inputs, binding delegates (events), or setting up network replication (RPCs).
USTRUCT	Makes a C++ struct usable in Blueprints and standardizes its serialization.
UENUM	Allows C++ enum class types to be used in the Editor and Blueprints.

Common Specifiers

Macro	Specifier	Example
UCLASS	• **Blueprintable**: Allows Blueprints to be created based on this C++ class • **BlueprintType**: Allows this class to be used as a variable type in Blueprints.	UCLASS(Blueprintable) class AMyActor : public AActor
UPROPERTY	• **EditAnywhere / VisibleAnywhere**: Determines if the variable can be edited or just viewed in the Editor's Details panel. • **BlueprintReadWrite / BlueprintReadOnly**: Determines if Blueprints can get/set the variable • **Category = "MyCategory"**: Organizes the variable into a specific drop-down menu in the Editor.	UPROPERTY(EditAnywhere, BlueprintReadWrite,Category = "Stats") int32 Health;
UFUNCTION	• **BlueprintCallable**: Allows the function to be executed via a node in Blueprints. • **BlueprintPure**: Creates a Blueprint node without execution pins (typically used for "getter" functions that just return data).	UFUNCTION(BlueprintCallable, Category = "Actions") void FireWeapon();

Class Communications

Unreal Engine supports three types of communications between classes:

- Casting: checks if an object is of a specific class.
- Interfaces: instead of casting to find out what an object is, you check if it implements an interface to find out what it can do.
- Events/Delegates: This is an observer pattern. The sender broadcasts a message, not knowing who is listening. Other classes bind their own functions to this event, so they are triggered when the broadcast fires.

Casting

Standard C++ Casting

static_cast: Performs no runtime check. Ideal for casting clearly related types where success is guaranteed.

dynamic_cast: Used for safe downcasting. Works only with polymorphic methods (virtual functions). Will silently return nullptr if the cast fails.

Unreal Engine Casting

CastChecked(): Acts like an assertion; it will instantly crash the engine if the cast fails. Best used when absolute logic certainty is required.

Cast(): Provides a safe runtime check specifically built for UObjects. Returns nullptr on failure.

Interfaces

- In Unreal Engine, an Interface is equivalent to a pure virtual class.
- Any actor can implement an interface either in C++ or Blueprint
- To check if an actor implement an interface :
  - Actor->Implements<UInterfaceName>()

- To execute a function of an interface on an actor:
  - IInterfaceName ::Execute_FunctionName(Actor)

```
UINTERFACE()
No derived blueprint classes
class USceneCaptureInterface : public UInterface
{
    GENERATED_BODY()
};


1 derived blueprint class
class UNIFIEDINTERFACE_API ISceneCaptureInterface
{
    GENERATED_BODY()
public:
    UFUNCTION(BlueprintNativeEvent,BlueprintCallable)
    void AttachTo(UTextureRenderTarget2D* InRenderTarget, int32 InSourceId);

    UFUNCTION(BlueprintNativeEvent,BlueprintCallable)
    bool CanBeTeleported();  No blueprint usages  Not implemented in blueprints
};
```

Delegates

- Unreal Engine support four types of delegates:
  - Single (Single subscriber)
  - Multicast (Multiple subscribers)
  - Dynamic Single (Single with Blueprint)
  - Dynamic Multicast (Multi with Blueprint)

```
DECLARE_DELEGATE_OneParam(FOnHpChange, int32)

DECLARE_MULTICAST_DELEGATE_OneParam(FOnHpChangeMulti, int32)

DECLARE_DYNAMIC_DELEGATE_OneParam(FOnHpChangeDynamic, int32 )

DECLARE_DYNAMIC_MULTICAST_DELEGATE_OneParam(FOnHpChangeDynamicMulti, int32)
```

```
DECLARE_DYNAMIC_MULTICAST_DELEGATE_OneParam(FOnKeyPress, int32, KeyId);

UCLASS()
No derived blueprint classes
class APlayerControllerBase : public APlayerController
{
    GENERATED_BODY()

public:
    // This should bne public
    // BlueprintAssignable makes it accessible from blueprints
    UPROPERTY(BlueprintAssignable)
    FOnKeyPress KeyPress;  No blueprint usages
};
```

Thank You!


VICARIOS LAB
iit
ADVR
ISTITUTO ITALIANO
DI TECNOLOGIA
ADVANCED ROBOTICS
INAIL
ISTITUTO NAZIONALE PER L'ASSICURAZIONE
CONTRO GLI INFORTUNI SUL LAVORO
IEEE
Telepresence