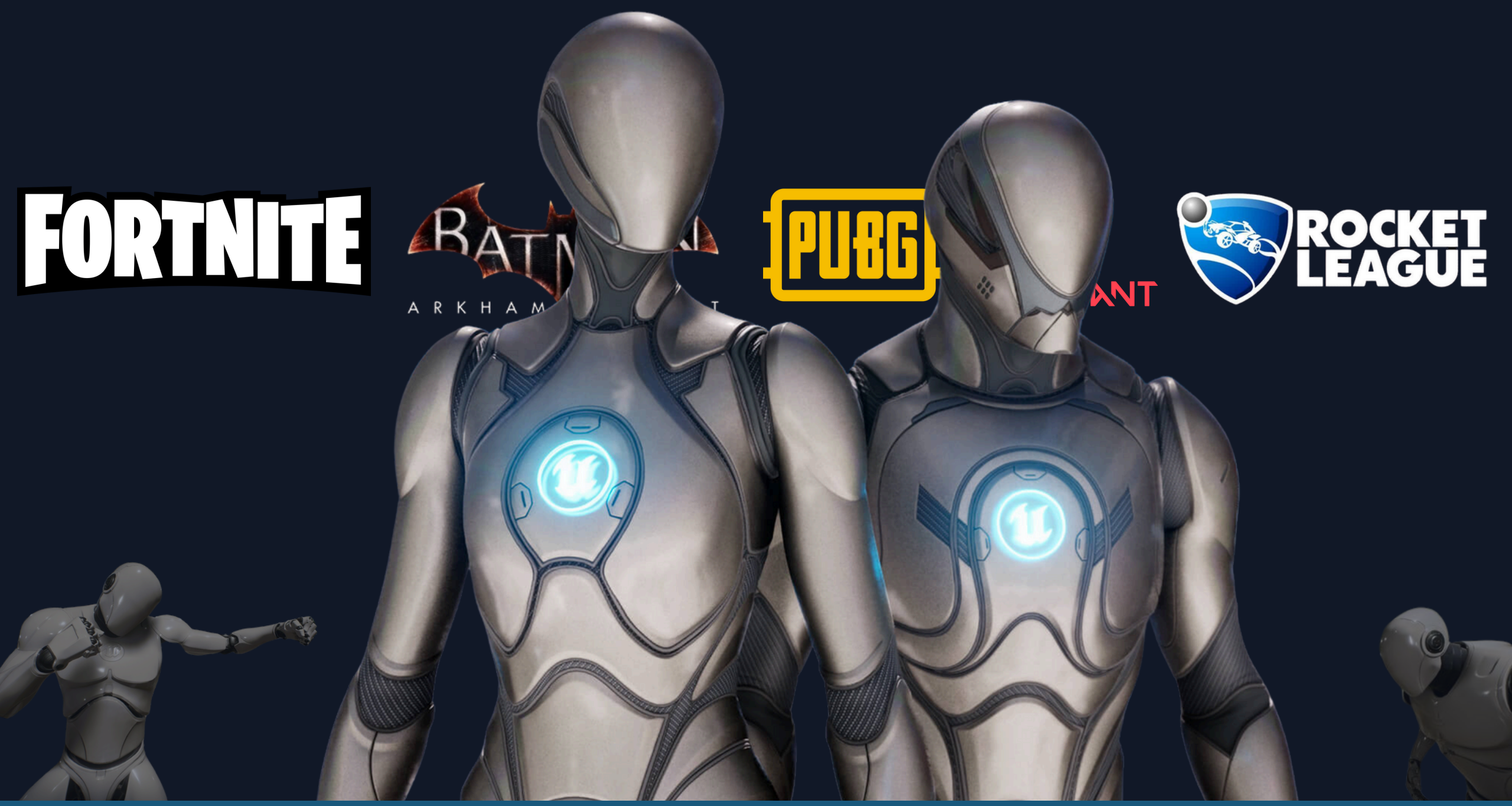
UNREAL CHEAT SHEETS

- Cheat Sheet 1: Gameplay Framework
- Cheat Sheet 2: Property Specifiers
- Cheat Sheet 3: Function Specifiers
- Cheat Sheet 4: Actor Lifecycle Functions
- Cheat Sheet 5: AActor Functions
- more coming soon...

GET THEM ALL



UNREAL CHEAT SHEET
GAMEPLAY FRAMEWORK

PART - 1

```
// WORLD & GAME INSTANCE
UGameInstance                   // Manages persistent data across levels
                                // (e.g., save systems)
UGameInstanceSubsystem          // Global systems (e.g., matchmaking)
                                // created and managed by GameInstance
UWorld                          // Represents the current level;
                                // manages actors, tick, and systems


// ACTORS & DERIVATIVES
AActor                          // Base for all objects placed in a level
                                // (replicable, transformable)
APawn                           // Controllable object in the world
                                // (possessed by Controller)
ADefaultPawn                    // Simple pawn with default movement
ASpectatorPawn                  // Pawn for free-fly viewing or spectating
ACharacter                      // Humanoid pawn with movement and
                                // animation components
AController                     // Non-physical class that controls Pawn behavior
APlayerController               // Handles input and camera for human players
AAIController                   // Drives AI behavior via logic/behavior trees
AGameMode                       // Server-only; defines rules and
                                // spawns key framework actors
AGameState                      // Replicates shared match data
                                // (scores, players) to clients
APlayerState                    // Tracks individual player data
                                // (health, inventory, etc.)
AHUD                            // Draws UI overlays (health bars, scores) on screen
ACameraActor                    // Defines the camera's position and
                                // orientation in world


// ACTOR COMPONENTS
UActorComponent                 // Base for reusable logic attached to actors
USceneComponent                 // Adds transform (location, rotation, scale)
UAudioComponent                 // Plays 2D/3D sounds
UParticleSystemComponent        // Visual effects (fire, smoke, etc.)
UPrimitiveComponent             // Enables rendering and collision
                                // (base for visible components)
UCameraComponent                // Adds a view/camera to an Actor
USpringArmComponent             // Adjustable boom arm for attaching cameras
USkeletalMeshComponent          // Renders animated (skinned) character meshes
UStaticMeshComponent            // Renders non-animated 3D objects (e.g., props)

// STATIC UTILITIES
UGameplayStatics                // Static utility for spawning, sound,
                                // damage, player queries, etc.
```

UNREAL CHEAT SHEET
PROPERTY SPECIFIERS

PART - 2

```
// COMMONLY USED
UPROPERTY(EditAnywhere)                 // Editable in both Blueprint and
                                        // Details panel
UPROPERTY(EditDefaultsOnly)             // Editable only on Class Defaults (CDO)
UPROPERTY(EditInstanceOnly)             // Editable per instance, not on defaults
UPROPERTY(VisibleAnywhere)              // Read-only but visible in Blueprint and
                                        // Details panel
UPROPERTY(BlueprintReadWrite)           // Read/write access in Blueprints
UPROPERTY(BlueprintReadOnly)            // Read-only access in Blueprints
UPROPERTY(Transient)                    // Not saved or loaded (runtime only)
UPROPERTY(Category = "Category")        // Organizes property in editor category

// MODERATELY USED
UPROPERTY(Replicated)                   // Enables replication for multiplayer
UPROPERTY(ReplicatedUsing = Func)       // Triggers function when replicated
                                        // value updates
UPROPERTY(SaveGame)                     // Serialized in SaveGame object
UPROPERTY(ExposeOnSpawn)                // Allows property to be set
                                        // during construction
UPROPERTY(VisibleDefaultsOnly)          // Visible only on class defaults
UPROPERTY(AdvancedDisplay)              // Hidden unless "Show Advanced"
                                        // is clicked
UPROPERTY(meta = (AllowPrivateAccess = "true")) // Exposes private vars
                                                // in editor

// LESS COMMON
UPROPERTY(Instanced)                    // Marks reference as owned subobject
UPROPERTY(Config)                       // Loads/stores property from .ini config
UPROPERTY(GlobalConfig)                 // Uses engine/global .ini config
UPROPERTY(Interp)                       // Used in Matinee to interpolate properties
UPROPERTY(AssetRegistrySearchable)      // Adds property to asset registry metadata
UPROPERTY(NoClear)                      // Disables the Clear button in editor UI
UPROPERTY(DuplicateTransient)           // Skipped during object duplication
UPROPERTY(NonPIEDuplicateTransient)     // Skipped during PIE duplication only

// EXPERIMENTAL / RARE
UPROPERTY(SkipSerialization)            // Completely ignored during serialization
UPROPERTY(TextExportTransient)          // Skipped during text export
UPROPERTY(BlueprintGetter = "Func")     // Custom getter for Blueprints
UPROPERTY(BlueprintSetter = "Func")     // Custom setter for Blueprints
UPROPERTY(ShowOnlyInnerProperties)      // Displays only inner struct/object
                                        // properties inline
```

UNREAL ENGINE CHEAT SHEET

FUNCTION SPECIFIERS

PART - 3

```
// COMMONLY USED
UFUNCTION(BlueprintCallable)                // Callable from Blueprints
UFUNCTION(BlueprintPure)                    // No side effects (read-only)
UFUNCTION(BlueprintImplementableEvent)      // Implemented in Blueprint,
                                            // called from C++
UFUNCTION(BlueprintNativeEvent)             // C++ default, can override
                                            // in Blueprint
UFUNCTION(Client, Reliable)                 // Runs on client from server
UFUNCTION(Server, Reliable)                 // Runs on server from client
UFUNCTION(NetMulticast, Reliable)           // Runs on all clients
                                            // from server

// MODERATELY USED
UFUNCTION(BlueprintAuthorityOnly)           // Only callable by server
                                            // (authority)
UFUNCTION(Unreliable)                       // Less bandwidth;
UFUNCTION(CallInEditor)                     // Callable in Editor
                                            // (e.g. button in Details)
UFUNCTION(Exec)                             // Callable via in-game console

// LESS COMMON
UFUNCTION(CustomThunk)                      // Used with Blueprints
                                            //  for custom call logic
UFUNCTION(meta = (WorldContext = "Context"))        // Used to inject
                                                    // world context
UFUNCTION(meta = (ExpandEnumAsExecs = "EnumParam")) // Converts enum
                                                    // to exec pins

// EXPERIMENTAL / RARE
UFUNCTION(meta=(NotBlueprintThreadSafe))            // Not safe for async
                                            // Blueprint usage
UFUNCTION(SealedEvent)                      // Prevents override
                                            // in Blueprint
UFUNCTION(BlueprintInternalUseOnly)         // Hides function from
                                            // standard BP use
```

UNREAL CHEAT SHEET
ACTOR LIFECYCLE

PART - 4

```
// INITIALIZATION & LOADING
UObject::PostLoad()                                  // After deserialization
UObject::PostDuplicate(bool bDuplicateForPIE)        // After PIE duplicate
AActor::PostActorCreated()                           // After spawn ctor
AActor::PreRegisterAllComponents()                   // Before component register
UActorComponent::OnComponentCreated()                // On component creation
AActor::PostRegisterAllComponents()                  // After component register
AActor::PreInitializeComponents()                    // Before component init
UActorComponent::InitializeComponent()               // Init component defaults
AActor::PostInitializeComponents()                   // After component init

// CONSTRUCTION
AActor::PostSpawnInitialize()                        // After spawn
AActor::ExecuteConstruction()                        // Run construction scripts
AActor::OnConstruction(const FTransform&)            // Blueprint construction hook
AActor::PostActorConstruction()                      // After construction scripts

// PLAY
AActor::BeginPlay()                                  // On play start

// TICKING
AActor::Tick(float DeltaSeconds)                     // Per-frame Actor tick
UActorComponent::TickComponent(float DeltaTime,
            ELevelTick, FActorComponentTickFunction&)
                                                     // Per-frame component tick

// REPLICATION
AActor::PreReplication(IRepChangedPropertyTracker&)
                                                     // Before property replication

// END & DESTROY
AActor::EndPlay(EEndPlayReason::Type)                // On play end or destroy
AActor::OnDestroyed(AActor*)                         // Legacy destroy callback
AActor::Destroy(bool bNetForce=false)                // Mark pending kill

// GARBAGE COLLECTION (UObject)
UObject::BeginDestroy()                              // Before GC cleanup
UObject::IsReadyForFinishDestroy()                   // Check GC readiness
UObject::FinishDestroy()                             // Final cleanup
```

UNREAL CHEAT SHEET
AACTOR FUNCTIONS

PART - 5

```
// TRANSFORM & MOVEMENT
FVector  GetActorLocation()                                  // world-space position
bool     SetActorLocation(const FVector&, bool bSweep)       // move / teleport
void     AddActorWorldOffset(const FVector&, bool bSweep)    // translate (world)
void     AddActorLocalOffset(const FVector&, bool bSweep)    // translate (local)
FRotator GetActorRotation()                                  // current rotation
bool     SetActorRotation(const FRotator&, bool bSweep)      // snap/rotate
void     AddActorWorldRotation(const FRotator&, bool bSweep) // rotate (world)
void     AddActorLocalRotation(const FRotator&, bool bSweep) // rotate (local)
FVector  GetActorScale3D()                                   // read scale
void     SetActorScale3D(const FVector&)                     // set scale
void     SetActorTransform(const FTransform&, bool bSweep)   // loc+rot+scale in one go
bool     K2_TeleportTo(const FVector&, const FRotator&)      // BP-friendly teleport
float    GetDistanceTo(const AActor*)                        // 3-D distance
float    GetHorizontalDistanceTo(const AActor*)              // X-Z plane distance
FVector  GetVelocity()                                       // current linear velocity
void     SetActorEnableCollision(bool bOn)                   // master collision toggle
bool     IsRootComponentMovable() const                      // static vs movable?
void     SetActorHiddenInGame(bool bHide)                    // show / hide actor
void     SetLifeSpan(float Seconds)                          // auto-destroy timer
UWorld*  GetWorld() const                                    // grab the world context

// ATTACHMENT & HIERARCHY
bool     AttachToActor(AActor*, const FAttachmentTransformRules&)
bool     AttachToComponent(USceneComponent*, const FAttachmentTransformRules&)
bool     K2_AttachToActor(AActor*, FName, EAttachLocation, bool)
bool     K2_AttachToComponent(USceneComponent*, FName, EAttachLocation, bool)
bool     K2_AttachRootComponentTo(USceneComponent*, FName, EAttachLocation, bool)
void     DetachFromActor(const FDetachmentTransformRules&)
void     DetachRootComponentFromParent()
AActor*  GetAttachParentActor() const
void     GetAttachedActors(TArray<AActor*>&) const
void     GetAllChildActors(TArray<AActor*>&, bool bRecursive) const
bool     IsAttachedTo(const AActor*) const
bool     IsBasedOnActor(const AActor*) const
void     SetRootComponent(USceneComponent*)
void     SetOwner(AActor*)
bool     HasAuthority() const

// COMPONENTS & CHILD-ACTOR UTILITIES
USceneComponent*       GetRootComponent() const              // root scene comp
template<typename T> T* GetComponentByClass() const          // first comp of type
template<typename T> T* FindComponentByClass() const         // null-safe lookup
void     GetComponentsByClass(UClass*, TArray<UActorComponent*>&) const
void     GetComponents(TArray<UActorComponent*>&) const      // every component
void     ForEachComponent(bool, FActorComponentIteratorFunction) // lambda iterate
void     RerunConstructionScripts()                          // rebuild like editor
```