MARCH 18-22, 2024
SAN FRANCISCO, CA

EFFICIENT XR DEV IN UE5

Alexander Silkin
Co-Founder & CTO at Survios

SPEAKER BIO

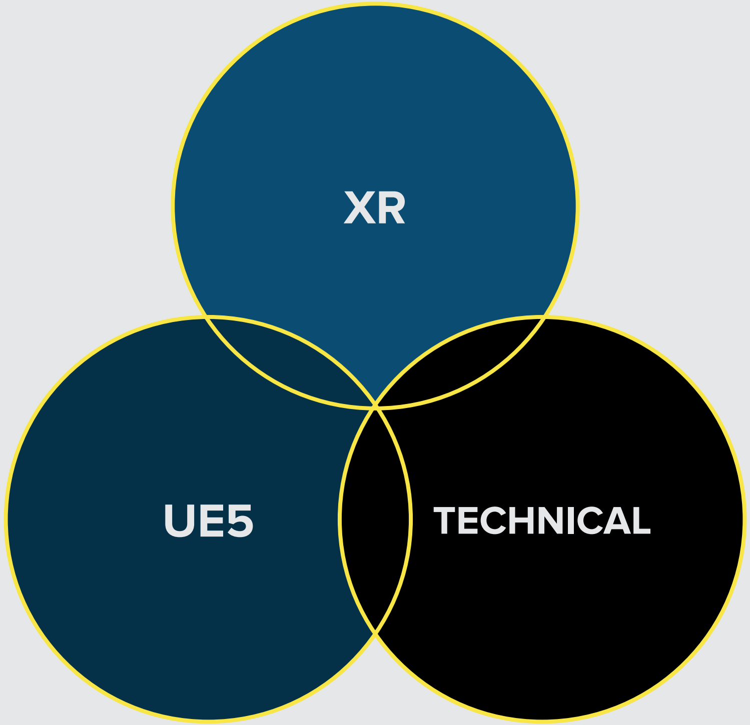
ALEXANDER SILKIN

CO-FOUNDER & CTO SURVIOS

Alex is a software engineer with an extensive background in consumer hardware, motion controls and software development. His professional experience includes stints at NASA, Microsoft, and the Information Sciences Institute at the University of Southern California (USC).

In 2012, Alex began working on VR at USC as a lead engineer on a student project - *Project Holodeck*. In 2013, the project gave birth to the creation of the **Survios** company.

TARGET AUDIENCE



KEY TAKEAWAYS

1 Best practices for XR tech stack in UE5

2 Tools and processes for optimal development

PREFACE

We work in our own branch of Unreal Engine (5.3.2)

↳ We **TRY** to avoid unnecessary modifications

We try to do things the "*Unreal Way*"

↳ Sometimes we **PURPOSEFULLY** stray off the beaten path

VR GAMES SHIPPED WITH UE4

7 VR Games released since 2015 across genres: fighting, shooting, surviving, racing, naval battling and puzzling.

Shipped on all the major VR stores and hardware platforms: Oculus PC + Quest, PS VR 1+2, SteamVR, Viveport.

RAW DATA SPRINT VECTOR CREED RISE TO GLORY WESTWORLD AWAKENING BATTLEWAKE WALKING DEAD ONSLAUGHT PUZZLE BOBBLE 3D VACATION ODYSSEY

MIGRATION TO UE5

Remastered in UE5 for PS VR2 and Quest 2+
(Check out Sylvie Sherman's talk on Wednesday @ 3:30!)

Next Gen VR development in UE5

TOPICS

INFRASTRUCTURE & WORKFLOWS

MODULAR REPOSITORY

P4V + Epic's robomerge + in house tools to simplify dependency merges.

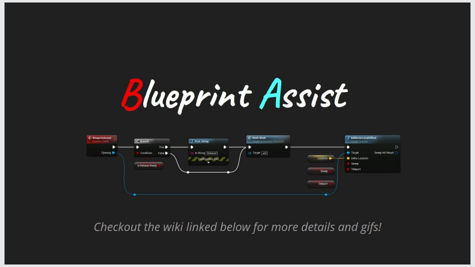
14 "tech depots" - contain Unreal Engine forks, plugins, and example projects.

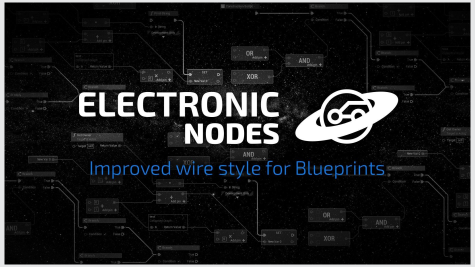
Tech depots have standardized stream structure:

- ↳ Dev
- ↳ Release-5.X
- ↳ Upgrade

Game depots merge dependencies from tech-depots and have similar structure.

MARKETPLACE (VENDOR) DEPOT





LOGIC DRIVER PRO
Prefabricator
Create Prefabs in Unreal Engine
Edit
Package
Blueprint Property Transfer Manager
Simple Editor Tool to help streamline transfering properties between unrelated Blueprints in Unreal Editor.
Edit
Package
Documentation

Wwise

SIMPLYGON™

37 "third party" plugins in NEW GAME depot

Plugins are maintained within tech-marketplace depot to manage:

- ↳ Modifications
- ↳ Upgrades
- ↳ Merges into game depots

PARTNER FORK DEPOTS

Meta and Sony UE forks are within their own vendor depots.

NEW GAME platform stream strategy:

- ↳ Dev stream avoids specific platform modifications
- ↳ Separate Meta and Sony streams merge from each fork

MODULAR CODEBASE

185 Plugins consisting of 310 Modules in NEW GAME depot

Minimize dependencies across modules:

- ↳ *[Plugin]Core* – contains interfaces and core structs
- ↳ *[Plugin]X/Y/Z* - implementations for X/Y/Z subsystems

All plugins are stored in their "depot folder" in *Engine/Plugins/Survios*

- ↳ eg. all plugins from tech-xr depot are in ***Engine/Plugins/Survios/tech-xr***
- ↳ Makes it easy to merge between game and tech depots

No code in "game" module – game specific plugins are in *Engine/Plugins/Survios/[GAME]*

- ↳ Convenient to have ALL the code under one directory
- ↳ Makes it easy to create separate project to target any specific plugins for challenging bug hunts

DEBUGGING > COMPILER OPTIMIZATION

DebugGame Editor – Recommended Daily Build Configuration

Engine Plugin modules ***build.cs*** need to be tagged one of these ways:

```
if (Target.Configuration == UnrealTargetConfiguration.DebugGame)
{
    OptimizeCode = CodeOptimization.Never;
}
```

```
bTreatAsEngineModule = false;
```

Alternative Solutions:

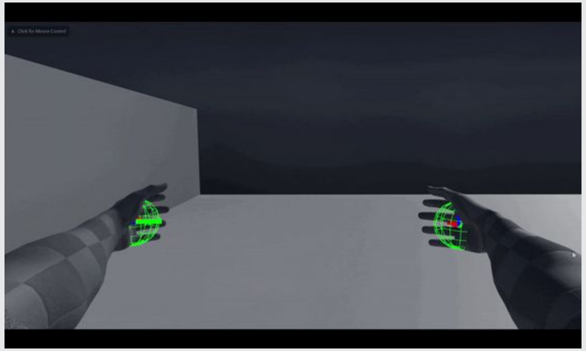
↳ Add modules to "DisableOptimizeCode" list in ***BuildConfiguration.xml***

↳ PRAGMA_DISABLE_OPTIMIZATION

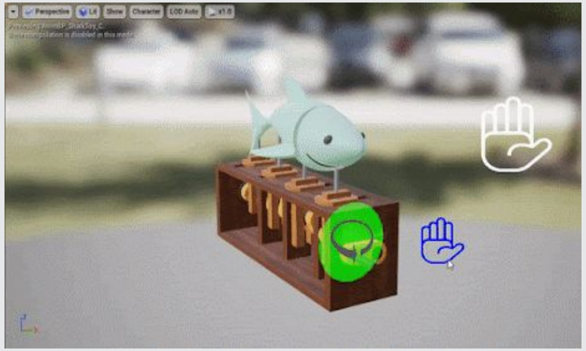
VR EMULATION TOOLS

Emulation tools aid rapid iteration outside of VR

Create and test your content outside of VR



USING MOUSE TO SIMULATE CONTROLLER IN-GAME



USING MOUSE TO SIMULATE INTERACTION IN EDITOR

CORE TECH

NEW PAWN CENTRIC STRUCTURE

Remove concept of a standalone hand Actor

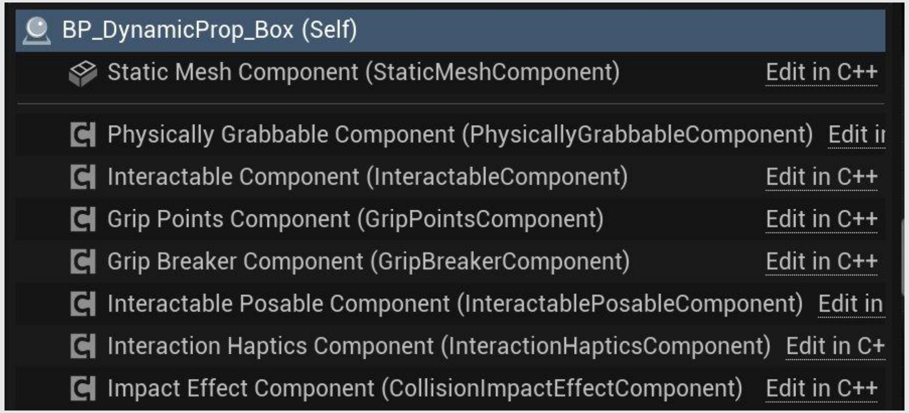
Encapsulate systems in components on player Pawn

- ↳ Each component manages both hands, usually through 2 instances of a UObject subclass
- ↳ Additional components attached to interactable actors for system specific data and logic
- ↳ Dependencies between systems minimized with the use of interfaces

SIMPLIFIED INTERACTABLE HIERARCHY

Broke apart systems across multiple components

Only 1 SceneComponent – the visual root component



BP_DynamicProp_Box (Self)
Static Mesh Component (StaticMeshComponent) Edit in C++
Physically Grabbable Component (PhysicallyGrabbableComponent) Edit ir
Interactable Component (InteractableComponent) Edit in C++
Grip Points Component (GripPointsComponent) Edit in C++
Grip Breaker Component (GripBreakerComponent) Edit in C++
Interactable Posable Component (InteractablePosableComponent) Edit in
Interaction Haptics Component (InteractionHapticsComponent) Edit in C+
Impact Effect Component (CollisionImpactEffectComponent) Edit in C++

TRACKING POLLING SYSTEM

We do not use ***UMotionControllerComponent***

Our system polls data with ***UHeadMountedDisplayFunctionLibrary::GetMotionControllerData***

FOpenXRHMD::GetMotionControllerData has caveats out of the box (UE 5.3):

- ↳ Tracker transforms are in world space
- ↳ Sets ***DeviceVisualType*** = ***EXRVisualType::Hand*** even when no valid hand tracking data but valid controller data

Taking control of the tracking code allows us to emulate VR with debug mouse and keyboard controls

AVATAR MOVEMENT COMPONENT

Movement modes' logic encapsulated in standalone classes

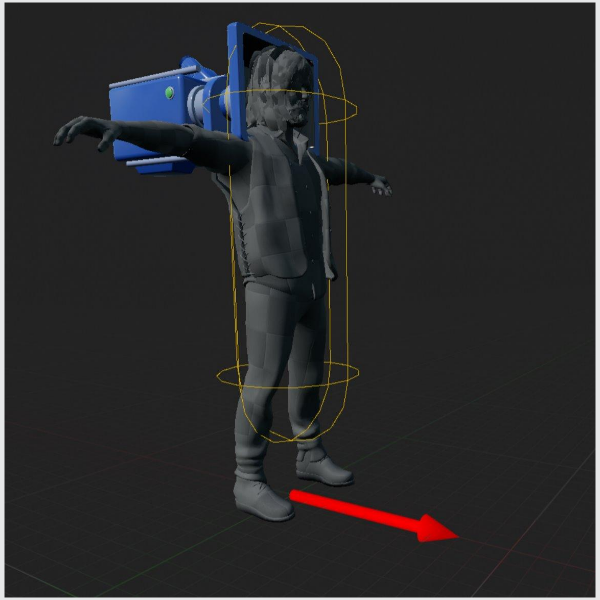
- ↳ Configuration stored in individual ***DataAssets***
- ↳ Modes are responsible for handling artificial locomotion and tracked head motion

MovementCollisionComponent - custom ***PrimitiveComponent***

- ↳ Pivot on the "floor"
- ↳ Capsule center and dimensions are modified by tracked head motion and movement mode logic

Minimal overhead on Game Thread

- ↳ Logic is run on worker thread
- ↳ ***MovementCollisionComponent*** transform update and event broadcasts are on game thread



NEW UNREAL FRAMEWORKS

Enhanced Input and Gameplay Ability System

ENHANCED INPUT

Enhanced Input cannot mirror right to left hand:

- ↳ Have to duplicate all the data in the ***InputMappingContext*** and ***InputAction***
- ↳ Duplicate code to bind to right vs left ***InputAction***

Our workflow avoids duplication:

- ↳ Use legacy action bindings instead of ***MappableInputConfig*** - ***DefaultInput.ini*** maps every FKey to dummy action
- ↳ Input assets are authored for right hand
- ↳ System generates new ***InputMappingContexts*** and ***InputActions*** by duplicating the authored assets and binding to left keys
- ↳ Gameplay code binds to the mirrored ***InputActions*** when left hand is involved

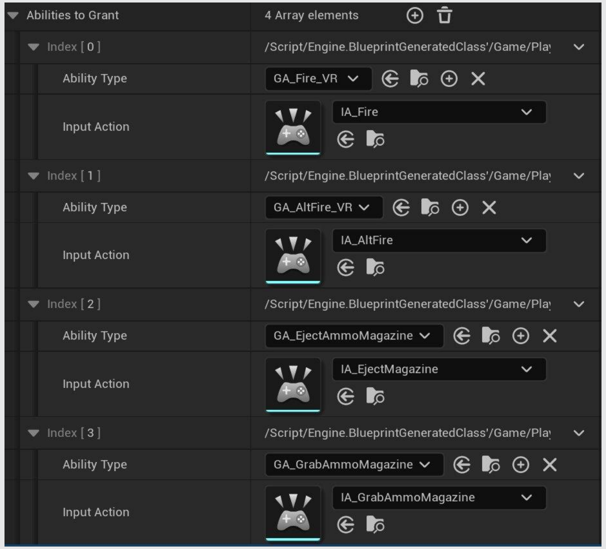
AbilityGrantComponent

Grants abilities and binds input

Added to the pawn for default abilities

Dynamically grants abilities while a gun is in hand

Set to request right vs left hand bindings



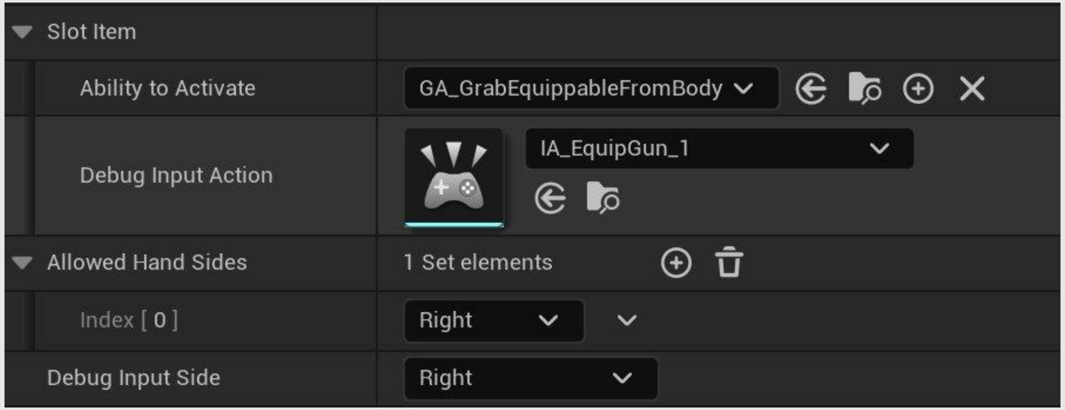
Abilities to Grant	4 Array elements
Index [0]	/Script/Engine.BlueprintGeneratedClass'/Game/Pla
Ability Type	GA_Fire_VR
Input Action	IA_Fire
Index [1]	/Script/Engine.BlueprintGeneratedClass'/Game/Pla
Ability Type	GA_AltFire_VR
Input Action	IA_AltFire
Index [2]	/Script/Engine.BlueprintGeneratedClass'/Game/Pla
Ability Type	GA_EjectAmmoMagazine
Input Action	IA_EjectMagazine
Index [3]	/Script/Engine.BlueprintGeneratedClass'/Game/Pla
Ability Type	GA_GrabAmmoMagazine
Input Action	IA_GrabAmmoMagazine

GrabSlot

Triggers an ability when user presses grab button within range

Can set ***InputAction*** to trigger ability for dev purposes

Used for body slot interactions



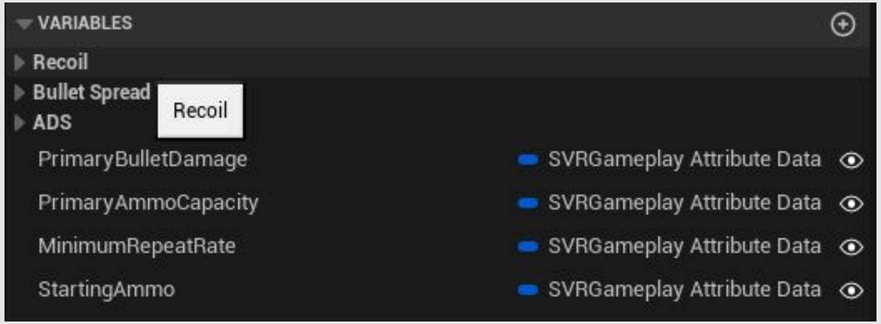
Slot Item	
Ability to Activate	GA_GrabEquippableFromBody
Debug Input Action	IA_EquipGun_1
Allowed Hand Sides	1 Set elements
Index [0]	Right
Debug Input Side	Right

Adding Attributes in Blueprint

Engine modification to allow blueprint subclasses of ***AttributeSet***

- ↳ Modified *FGameplayAttribute::GetAllAttributeProperties* and *SAttributeListWidget::UpdatePropertyOptions*
- ↳ We can add new attributes in BP instead of C++

FSVRGameplayAttributeData – subclass to expose "InitialBaseValue"

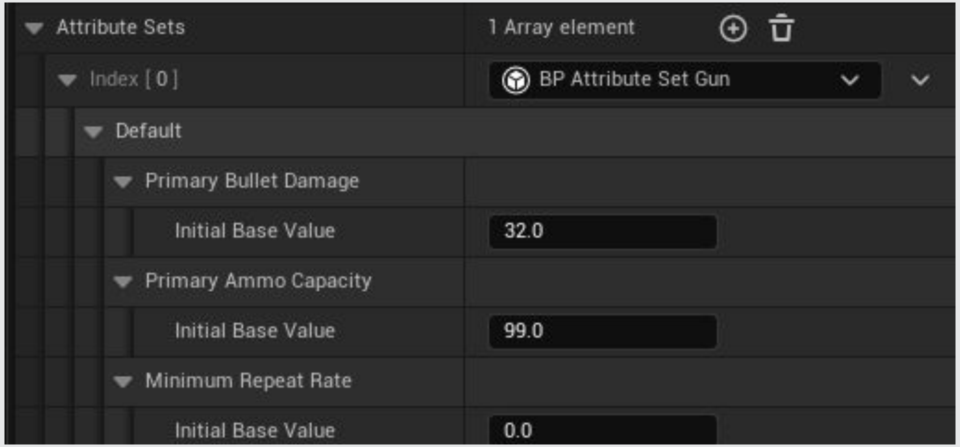


PrimaryBulletDamage	SVRGameplay Attribute Data
PrimaryAmmoCapacity	SVRGameplay Attribute Data
MinimumRepeatRate	SVRGameplay Attribute Data
StartingAmmo	SVRGameplay Attribute Data

Setting & Overriding InitialBaseValue

AttributeSetConfigurations – blueprintable object

- ↳ Collection of ***AttributeSet*** instances that expose "InitialBaseValue" for each attribute
- ↳ Subclassed to provide variants, for example for different weapons

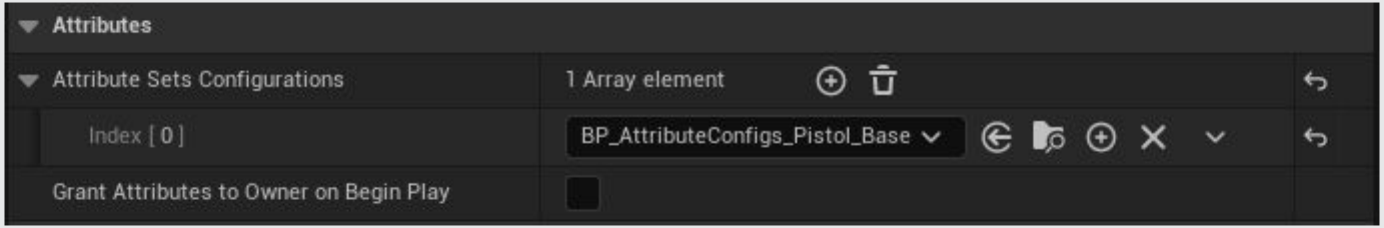


Attribute Sets	1 Array element
Index [0]	BP Attribute Set Gun
Default	
Primary Bullet Damage	
Initial Base Value	32.0
Primary Ammo Capacity	
Initial Base Value	99.0
Minimum Repeat Rate	
Initial Base Value	0.0

AttributeSetGrantComponent

AttributeSetGrantComponent – grants and initializes attribute sets

- ↳ Editable list of ***AttributeSetConfigurations*** to be granted
- ↳ Primary use case – when gun is held, grant the attributes and initialize them with the defaults for that gun



Attributes

Attribute Sets Configurations	1 Array element
Index [0]	BP_AttributeConfigs_Pistol_Base
Grant Attributes to Owner on Begin Play	☐

DAMAGE WITH GAS ATTRIBUTES

AActor::TakeDamage - deprecated in UE5

Request damage and healing with GameplayEffects (GE) that modify attributes in ***DamageAttributeSet***:

- ↳ **Damage** – amount of hitpoints to decrement
- ↳ **DamageScratchPad** - temporary variable for GE modifiers to override before updating Damage
- ↳ **Heal** – amount of hitpoints to add

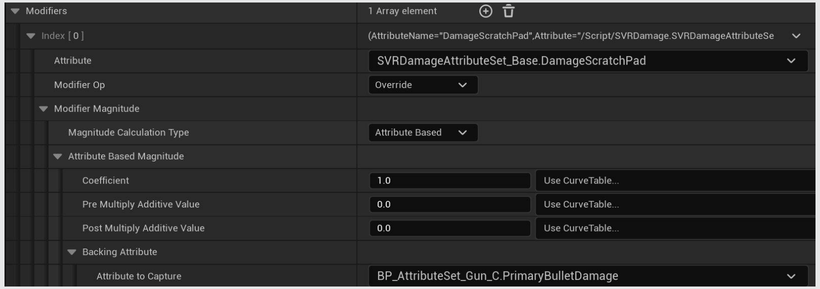
Override ***DamageAttributeSet::PostGameplayEffectExecute***

- ↳ Handle changes in attributes to broadcast to the DamageableComponent

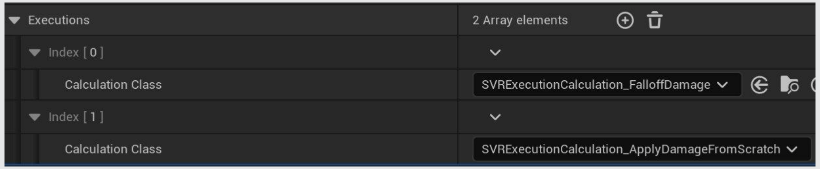
DamageScratchPad CHAIN

Damage GEs calculate damage based off attributes and *GameplayEffectExecutionCalculations*

1. Initialize DamageScratchPad with modifier backing attribute
2. Add a chain of *GameplayEffectExecutionCalculation* to modify *DamageScratchPad*
3. *ApplyDamageFromScratchPad* – final execution set Damage to *DamageScratchPad* value



Modifiers	1 Array element	
Index [0]	(AttributeName="DamageScratchPad",Attribute="/Script/SVRDamage.SVRDamageAttributeSe	
Attribute	SVRDamageAttributeSet_Base.DamageScratchPad	
Modifier Op	Override	
Modifier Magnitude		
Magnitude Calculation Type	Attribute Based	
Attribute Based Magnitude		
Coefficient	1.0	Use CurveTable...
Pre Multiply Additive Value	0.0	Use CurveTable...
Post Multiply Additive Value	0.0	Use CurveTable...
Backing Attribute		
Attribute to Capture	BP_AttributeSet_Gun_C.PrimaryBulletDamage	



Executions	2 Array elements
Index [0]	
Calculation Class	SVRExecutionCalculation_FalloffDamage
Index [1]	
Calculation Class	SVRExecutionCalculation_ApplyDamageFromScratch

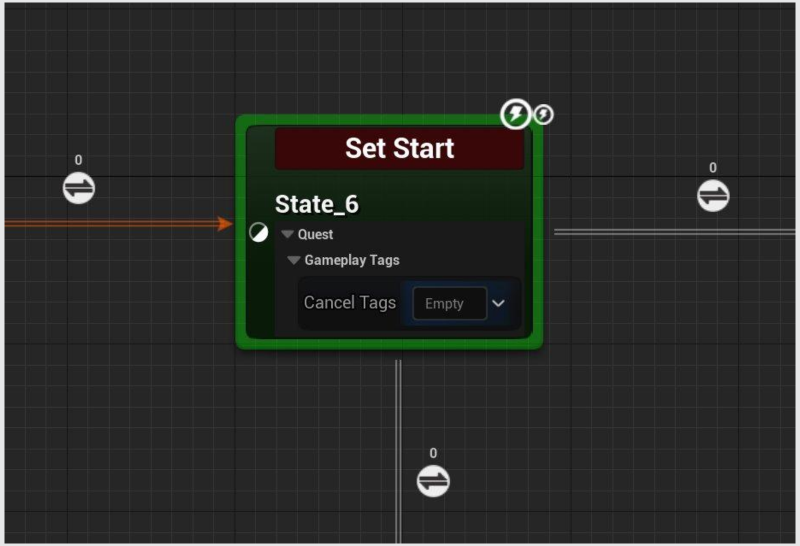
SCRIPTING & SAVING

SCRIPTING SYSTEM

Built "quest" system on top of Logic Driver Pro

Added custom functionality:

- ↳ Setting start node to launch PIE
- ↳ Cheats for skipping active state



SAVE SYSTEM

Modular Save Components

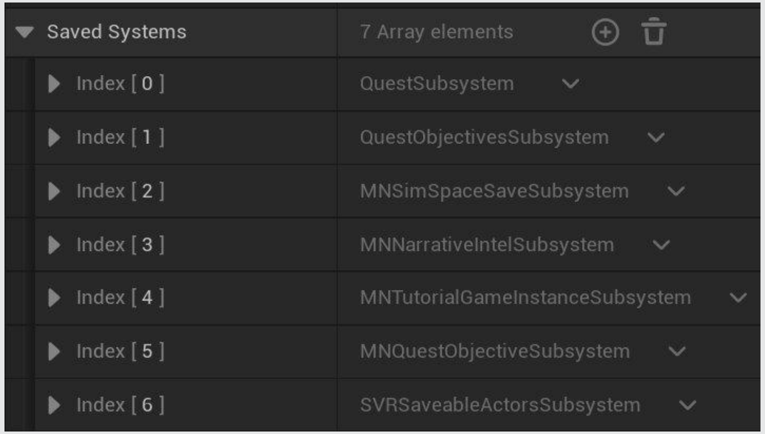
- ↳ ***GameInstanceSubsystems*** implement ***ISaveComponent***
- ↳ Each save component generates USaveGame
- ↳ ***USaveGames*** are combined into 1 master save file

Functionality to view and edit saves in Editor

- ↳ Import .sav file to create ***SaveGameDataAsset***

Use "debug" saves to test different parts of game

- ↳ Designers configure debug saves to be generated at various points using scripting system
- ↳ Saves are packaged with the game as ***SaveGameDataAssets***



Saved Systems	7 Array elements
Index [0]	QuestSubsystem
Index [1]	QuestObjectivesSubsystem
Index [2]	MNSimSpaceSaveSubsystem
Index [3]	MNNarrativeIntelSubsystem
Index [4]	MNTutorialGameInstanceSubsystem
Index [5]	MNQuestObjectiveSubsystem
Index [6]	SVRSaveableActorsSubsystem

SCRIPT SKIPPING vs DEBUG SAVES

Script skipping gives full flexibility to play at any location

- ↳ The behavior is not always correct as it requires manually configuring the game state
- ↳ Great for development iteration

Debug saves give an accurate snapshot for replay

- ↳ Active development quickly invalidates saves
- ↳ Great for hardened builds at the end of milestone

PERFORMANCE

ACTOR ACTIVATION

Engine modification introduces concept of ***Actor*** "activation"

- ↳ Similar to ***GameObject.SetActive*** in Unity
- ↳ State automatically propagates to child attached actors

Toggles core systems:

- ↳ ***Actor*** and ***ActorComponent*** ticking – registers & unregisters tick functions
- ↳ Primitive visibility and collision – adds & removes from render and physics scene

OnActivateActor & ***OnDeactivateActor*** – virtual callbacks on ***Actor*** and ***ActorComponent***

ACTOR POOLING

Spawning actors still too expensive - enemy ACharacter spawn times:

- ↳ PS5 - ~5ms
- ↳ Quest 3 - ~8.5ms

PoolManager spawns a preset number of actors on ***BeginPlay***

Pooling system uses the "Actor Activation" system

- ↳ ***OnActivateActor(bool bResetGameState)*** - bool parameter used to notify actors and components to reset state when leaving pooled state

Engine modified to call into ***PoolManager***:

- ↳ ***UWorld::SpawnActor*** – takes out actor from a pool by activating actor and resetting state
- ↳ ***UWorld::DestroyActor*** - put actor back into pool by deactivating actor

GPU LIGHT BAKES

Static lighting is a must for visual quality and performance in VR

- ↳ Precomputed Visibility is necessary to lower culling costs

Lumen is great for real time preview in Editor

GPU light bakes are much faster than CPU Lightmass!

Automating GPU light bakes is tricky

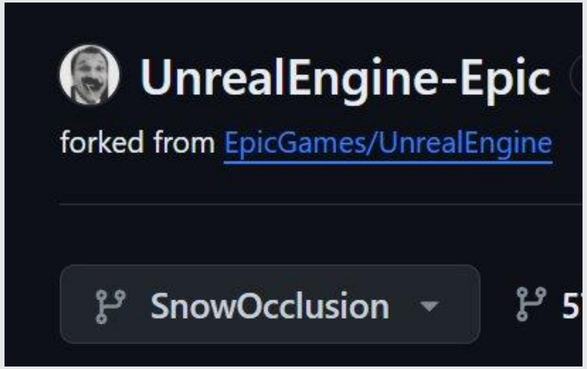
- ↳ Cannot do a GPU bake from a headless client since GPU is needed
- ↳ We are launching an Editor from Jenkins and have a plugin that responds to launch parameters

SOFTWARE OCCLUSION ON QUEST

Epic removed Software Occlusion from UE5

Fast Travel Games has graciously open sourced "Snow Occlusion" plugin for UE5

Fast Travel Games


UnrealEngine-Epic
forked from EpicGames/UnrealEngine

QUESTIONS?

Reach out to keep the discussion going!

Alexander Silkin

Discord: alex.silkin

alex.silkin@survios.com