XRDC

Why Survios Builds New Tech For Every Title & Why You Should Too

ALEX SILKIN
Co-Founder / Chief Technology Officer
SURVIOS

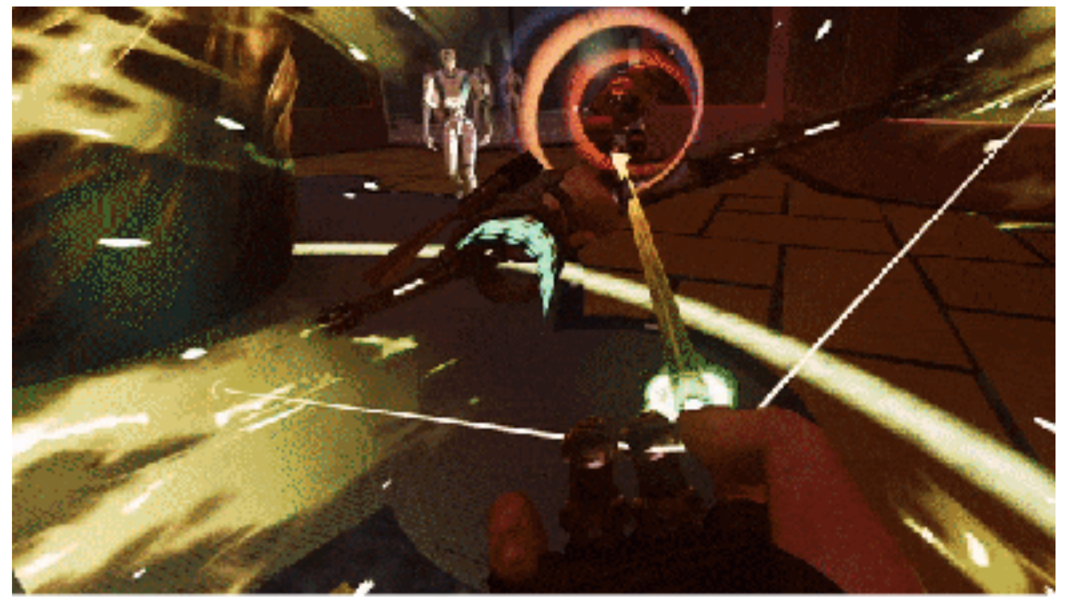
Key Takeaways

1. How to grow a flexible and extensible codebase for cross-platform VR game development

2. Mistakes to avoid

3. Examples of some of our foundational technologies

Previous Titles



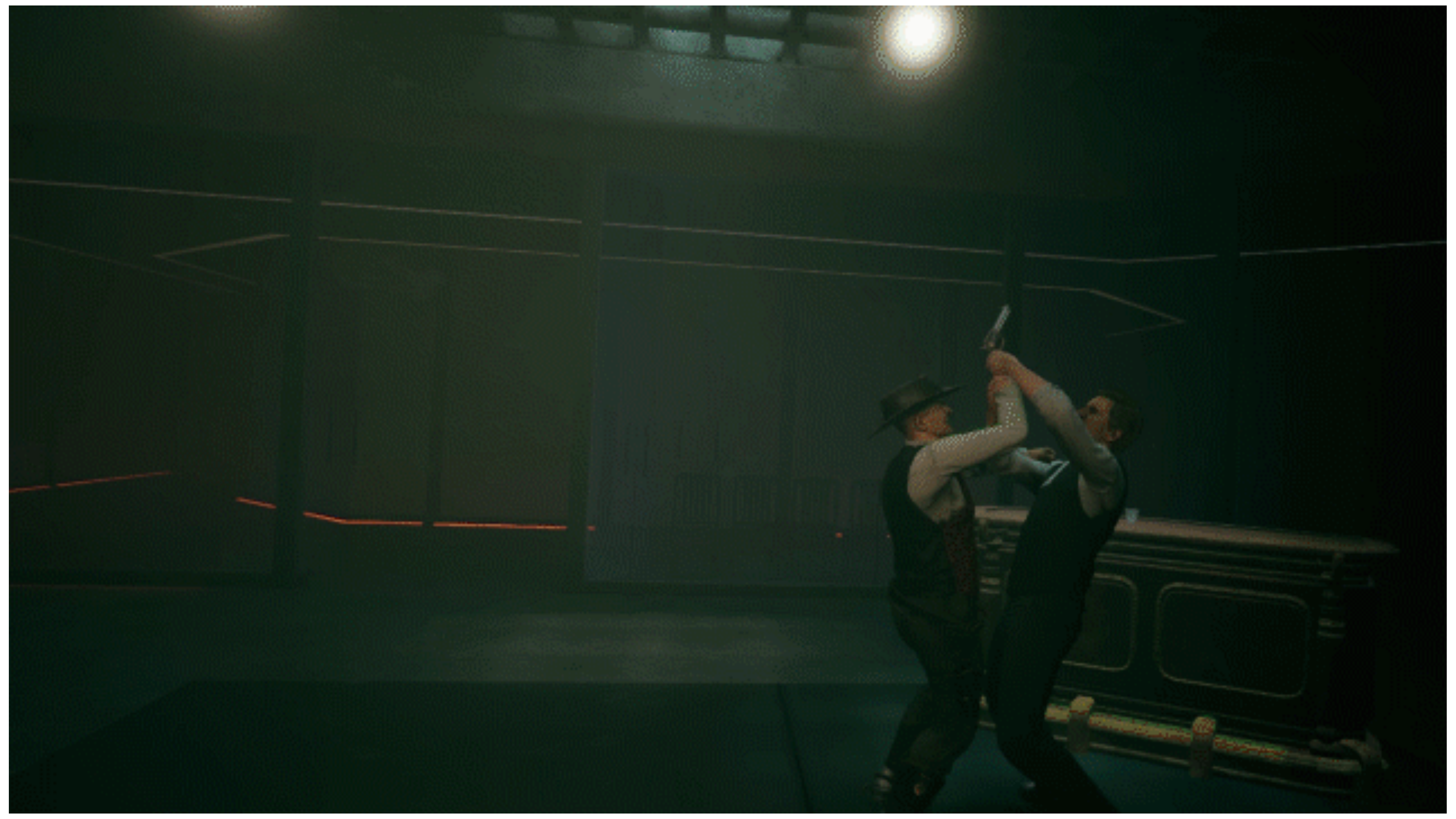
RAW DATA™

SPRINT VECTOR™

ELECTRONAUTS

CREED
RISE TO GLORY™

Newest Titles

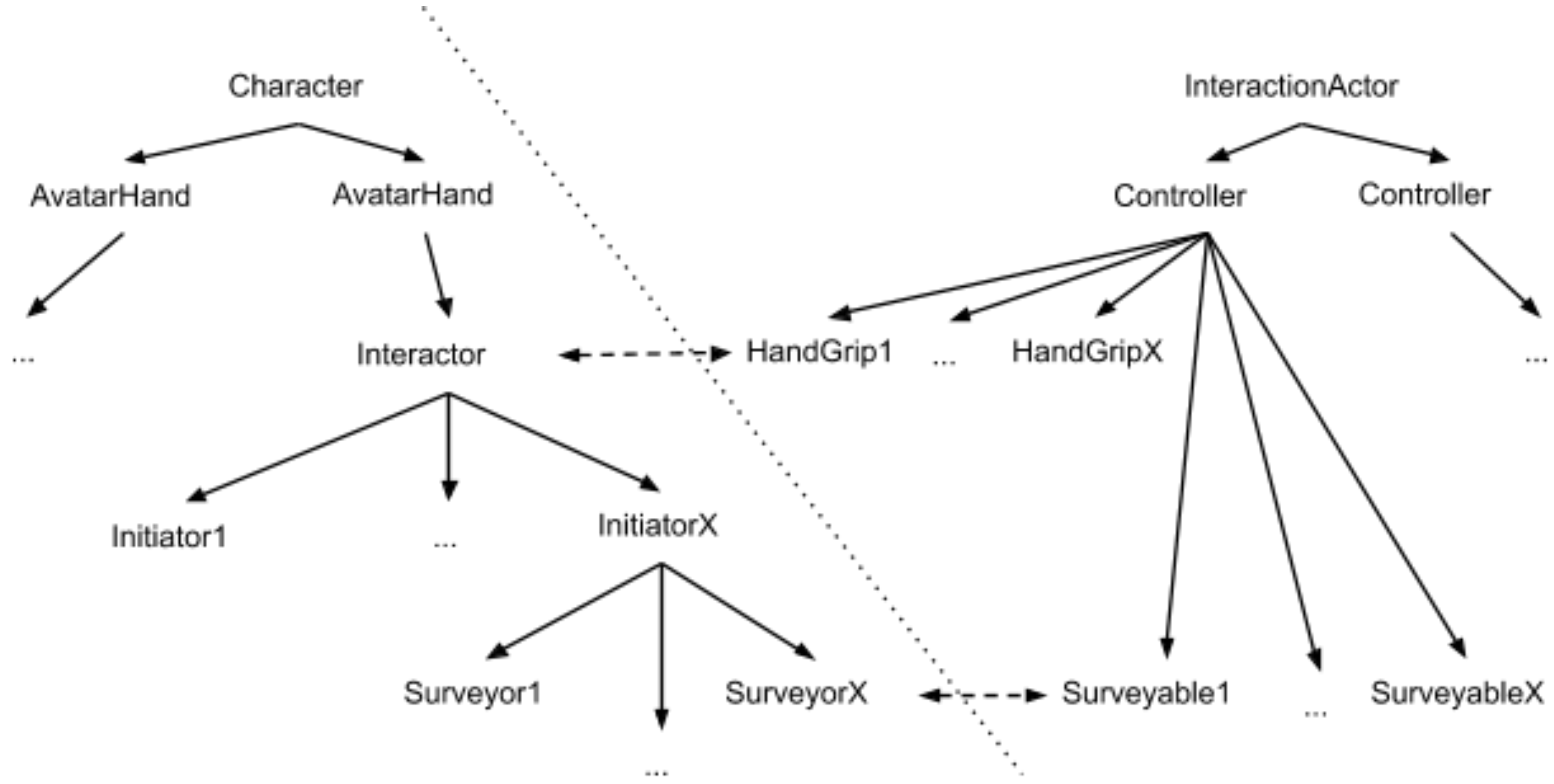


BATTLEWAKE

WESTWORLD AWAKENING

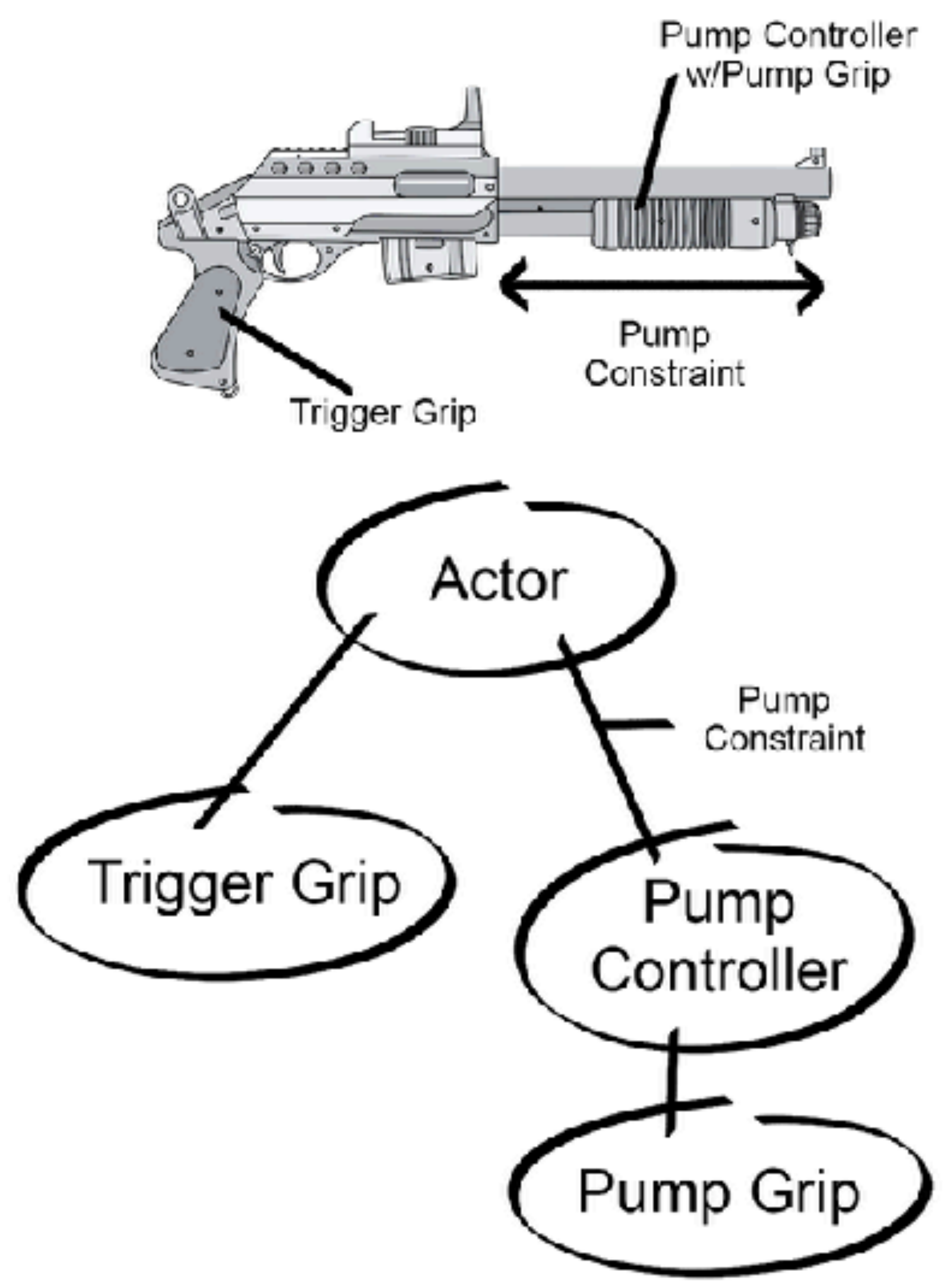
RAW DATA™
HOSTILE TAKEOVER

Interaction System

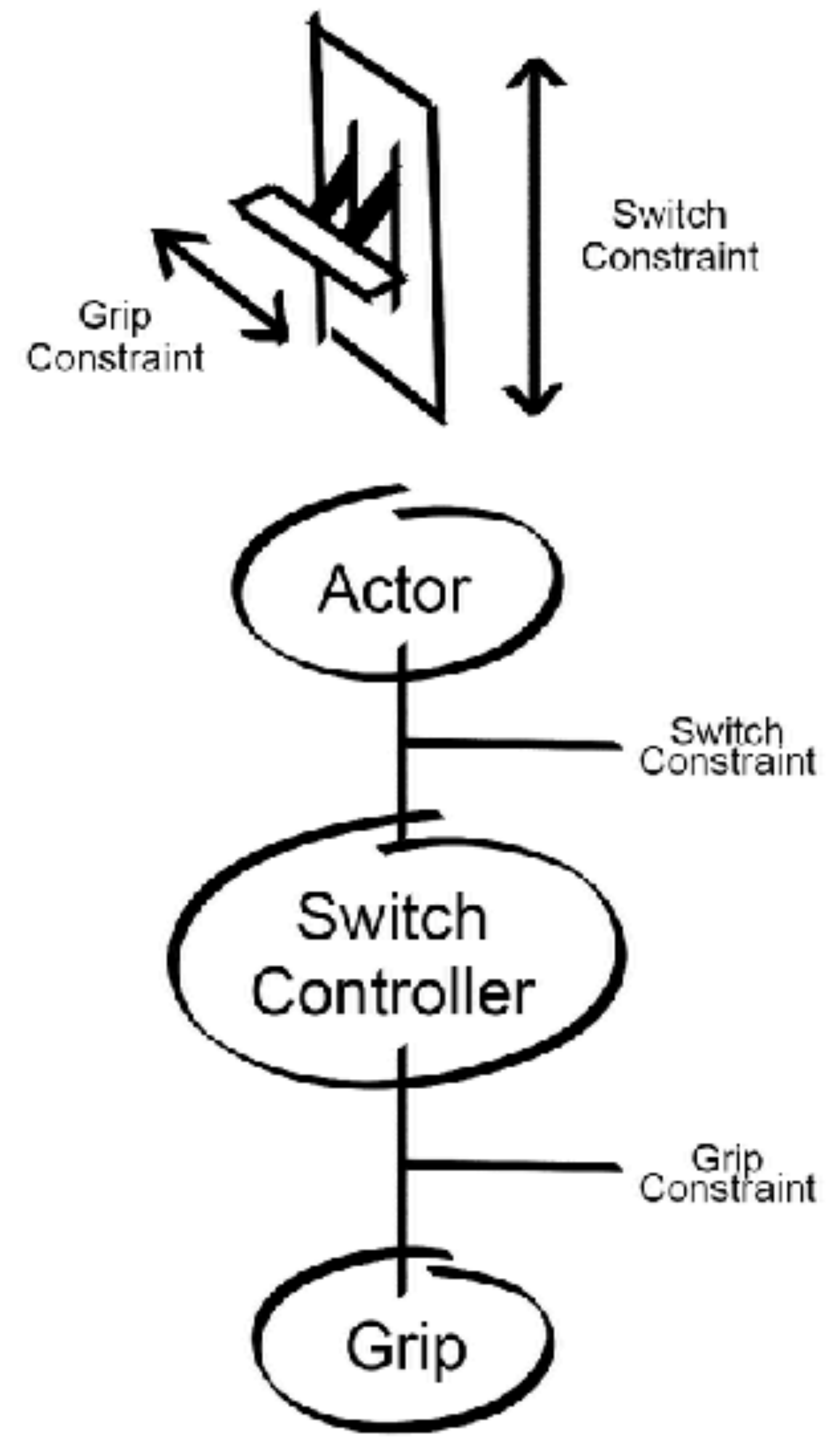


Interaction Positioning

Shotgun



Slidable Switch



Marionette

Weapon System

Modular and portable - use for all player and weapon guns as well as autonomous turrets

Firemode component

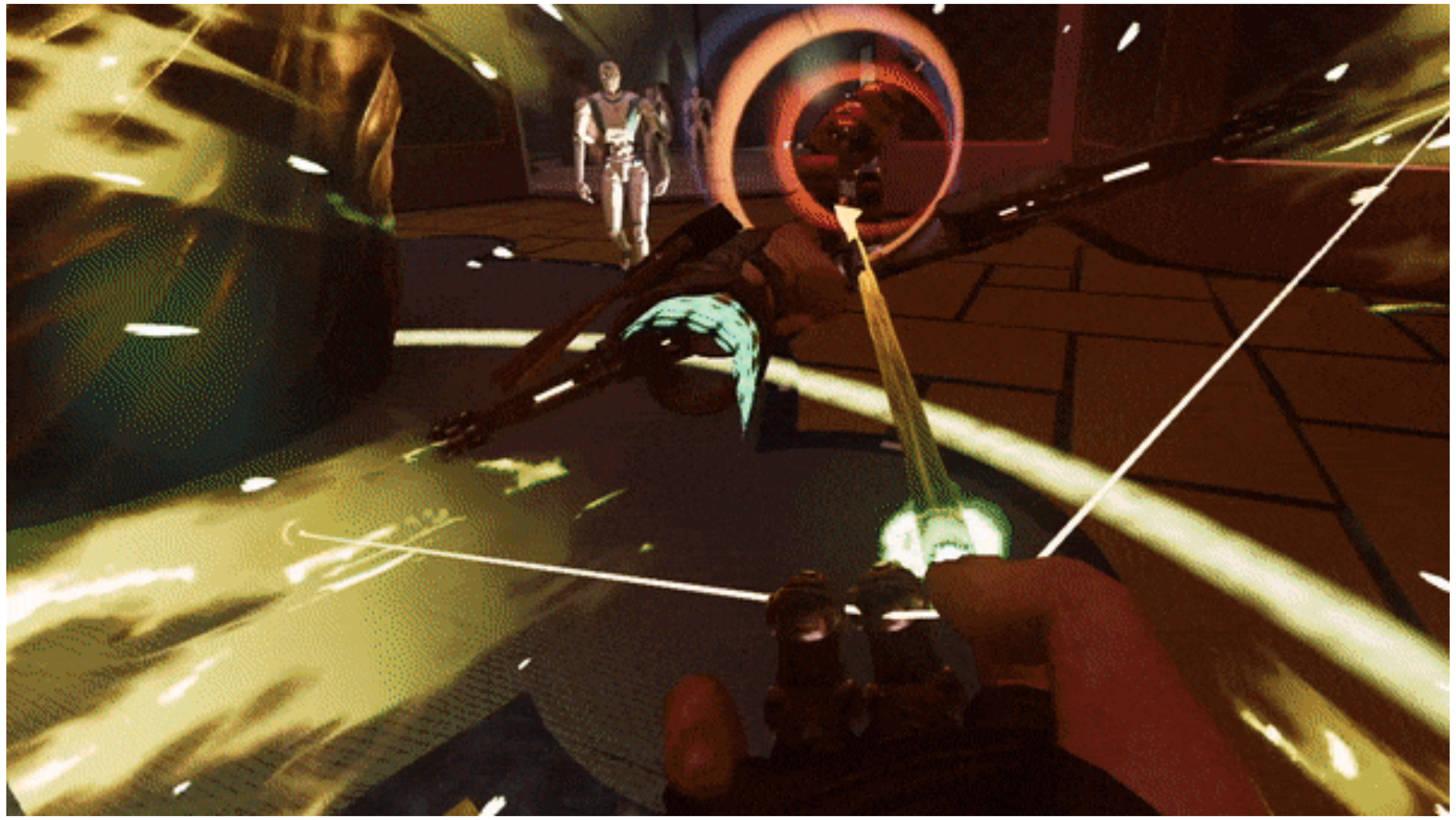
- Automatic
- Burst Fire
- Spooling
- Charge Shot

Damager Component

- Raytrace (hitscan)
- Projectile
- Volume Based

Ammo System

Firing Effects



Damage Systems

Damageable

- Per body region health
- Custom attributes (eg armor)
- Supports dismemberment and headshots

Hit Reaction

- Associate damage events with appropriate response
- Play animation
- Ragdoll physicalization
- Rotate according to damage direction

Multiplayer

All our gameplay systems are built with multiplayer in mind

Homegrown solutions for online services

- Dedicated Servers
- PC crossplatform
- Leaderboards

Porting Raw Data to PS VR

PS VR came out during Raw Data early access - we decided to port mid development!

Avoid porting - develop for most constrained platform as lead SKU

Performance

- Actor pooling system
- UMG Widget pooling system
- Async overlap system

Button mapping

- Redesign some mechanics
- Had to explicitly check for platform to determine button behavior

Get Set!
Summer 2015

Inventing Fluid Locomotion

SPRINT VECTOR
machinima

Focus Testing

SPRINT
VECTOR
BETA

CREED
RISE TO GLORY™

Inventing Phantom Melee Tech

MATHES
CREED
RISE TO GLORY

SURVIO

Sprint Vector & Creed Post Mortem

Problem: Majority of code is shared through one Survios plugin

Flexibility and Extensibility Issues:

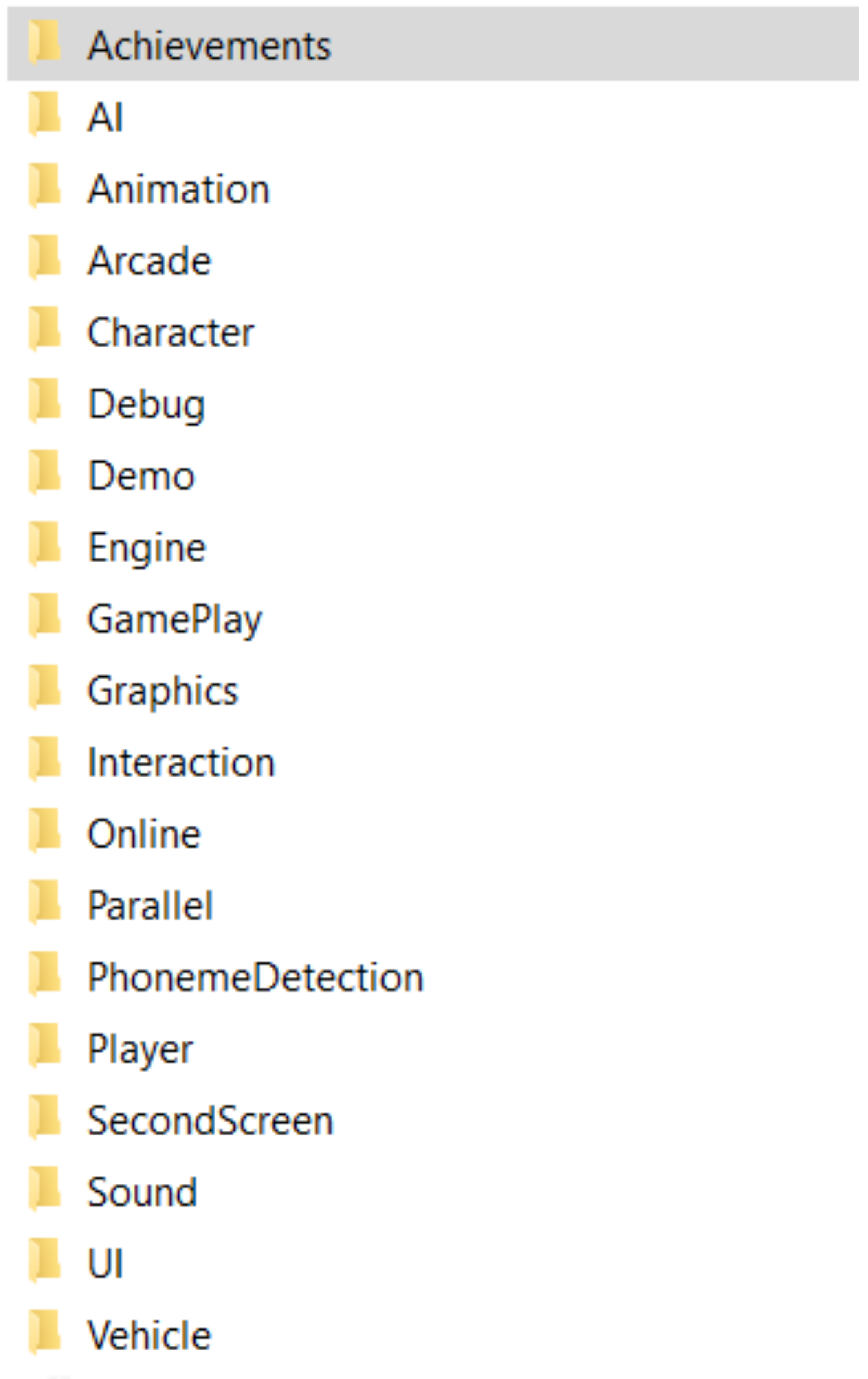
- Too many assumptions about different games' structure and needs
- Difficult to refactor/extend/replace/debug systems in isolation

Solutions:

- Get rid of base classes (SVRGameMode, SVRGameState, SVRPawn, SVRPlayerState)
- Decouple systems into separate independent plugins with abstraction layers
- Template provides sample configuration of systems for projects to branch from

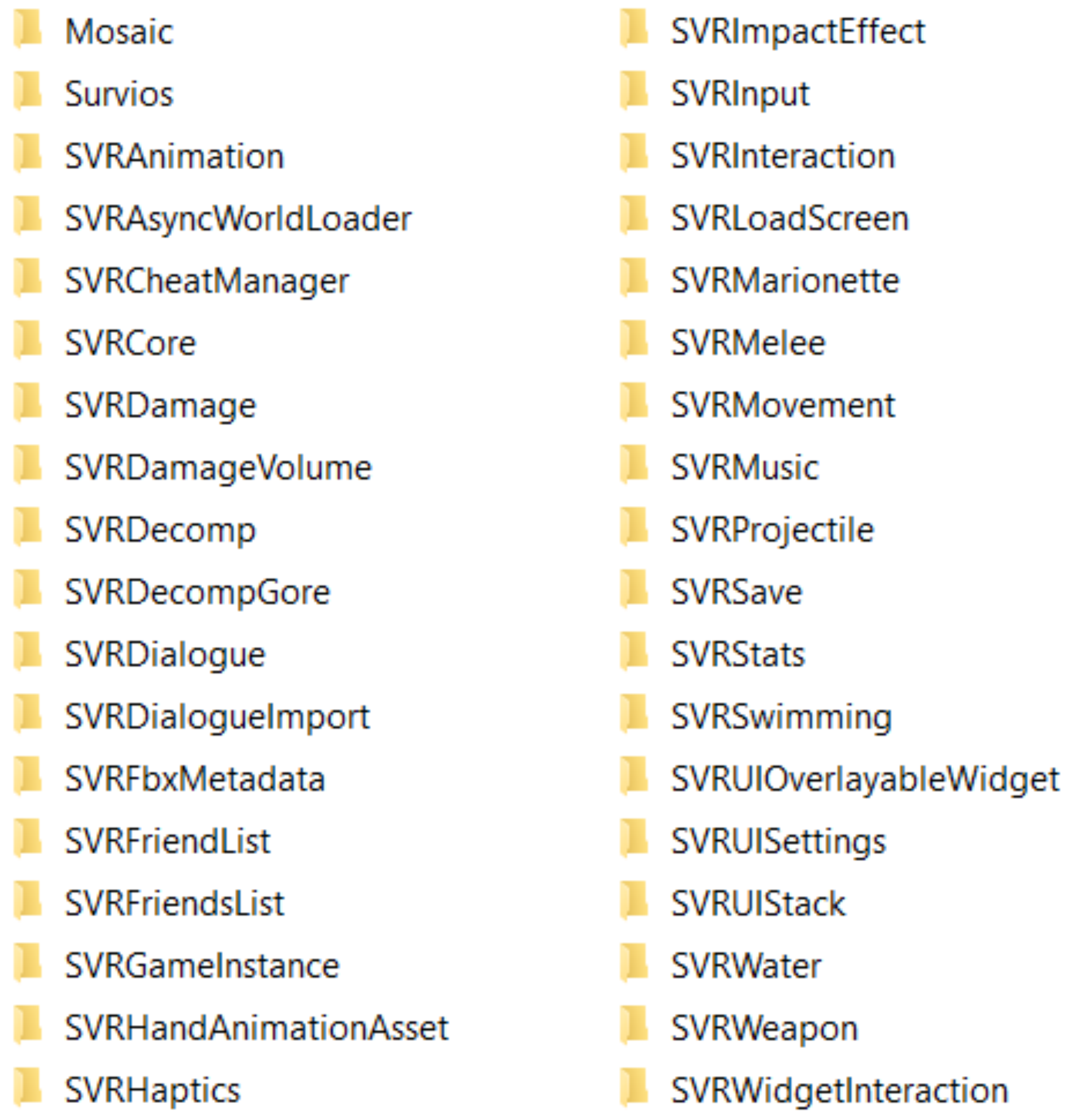
The Great Pluginification

Contents of old Survios plugin



- Achievements
- AI
- Animation
- Arcade
- Character
- Debug
- Demo
- Engine
- GamePlay
- Graphics
- Interaction
- Online
- Parallel
- PhonemeDetection
- Player
- SecondScreen
- Sound
- UI
- Vehicle

37 Current Plugins



- Mosaic
- Survios
- SVRAnimation
- SVRAsyncWorldLoader
- SVRCheatManager
- SVRCore
- SVRDamage
- SVRDamageVolume
- SVRDecomp
- SVRDecompGore
- SVRDialogue
- SVRDialogueImport
- SVRFbxMetadata
- SVRFriendList
- SVRFriendsList
- SVRGameInstance
- SVRHandAnimationAsset
- SVRHaptics
- SVRImpactEffect
- SVRInput
- SVRInteraction
- SVRLoadScreen
- SVRMarionette
- SVRMelee
- SVRMovement
- SVRMusic
- SVRProjectile
- SVRSave
- SVRStats
- SVRSwimming
- SVRUIOverlayableWidget
- SVRUISettings
- SVRUIStack
- SVRWater
- SVRWeapon
- SVRWidgetInteraction

BATTLEWAKE

Vehicular Locomotion

ECS Projectile System

Projectile Definition

Heavy use of editinlinenew instanced subobjects in a data asset

SVRProjectileDefinitionDataAsset

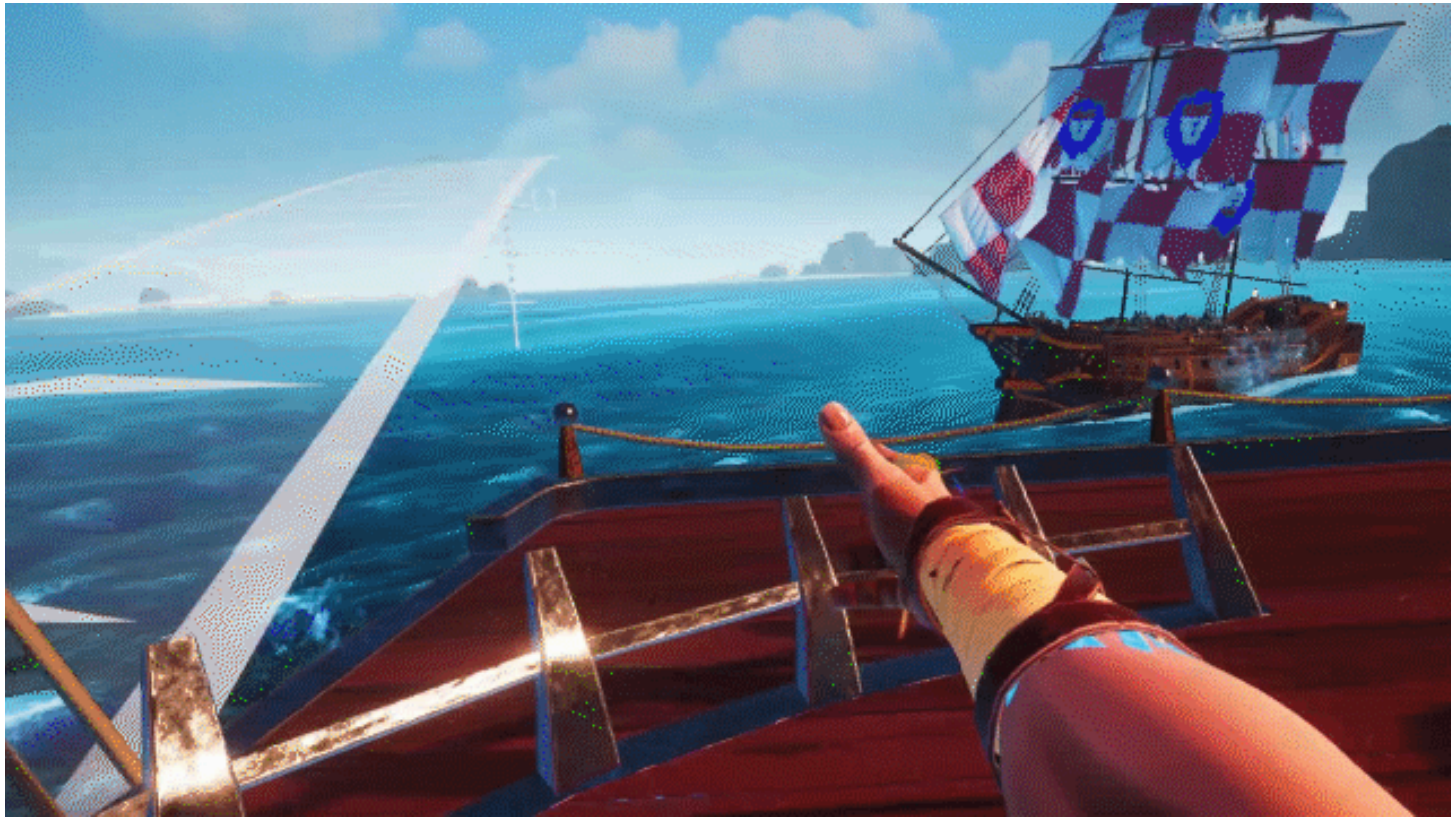
- SVRDamageAccessBase
- SVRProjectileMovementBase
- SVRProjectileCollisionBase
- SVRProjectileFXBase

Projectile Manager

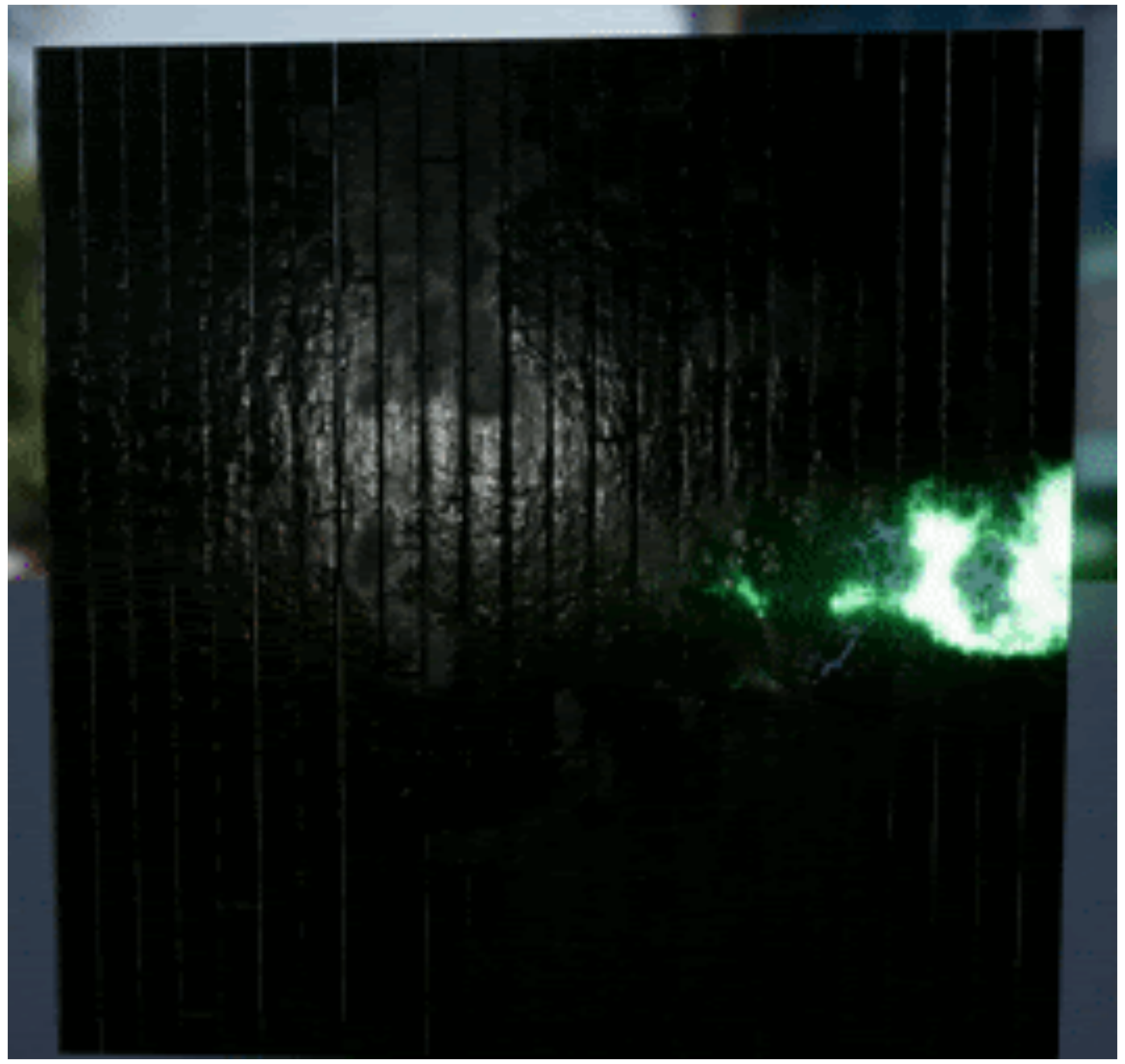
ASVRProjectileManager

- FSVRProjectileCollection array
  - SVRProjectileDefinitionDataAsset
  - FSVRProjectileInstance array
    - Transform
    - Velocity

Damage Decal Composition



Damage Decal Example



Future Developments

Player Movement Refractor

Completely detach from UE native CharacterMovementComponent

A state machine approach:

- Locomotion schemes encapsulated within USVRMovementState subclasses
- “Additive” movement state, ie USVRTurnInPlaceAdditiveMoveState for artificial yaw rotation

Input handling for each state implemented in a separate object that extends USVRMovementStateInputManager

SVRInput System

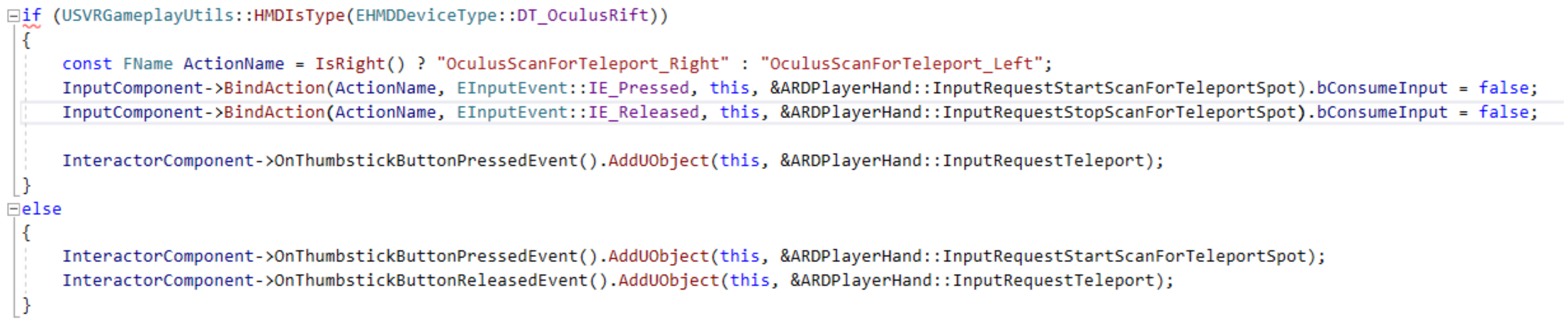
INI config binding based - built on top of Unreal Engine vanilla implementation

System dynamically modifies input mappings in response to:
- Which motion controller (important on PC)
- Control scheme variant selection
- Dominant hand selection (left hand vs right hand)

Input components explicitly enabled for left vs right hand
- Convenient for interaction system when you can grab objects with either or both hands

Example Raw Data Binding

C++ code initializing a player’s hand actor:

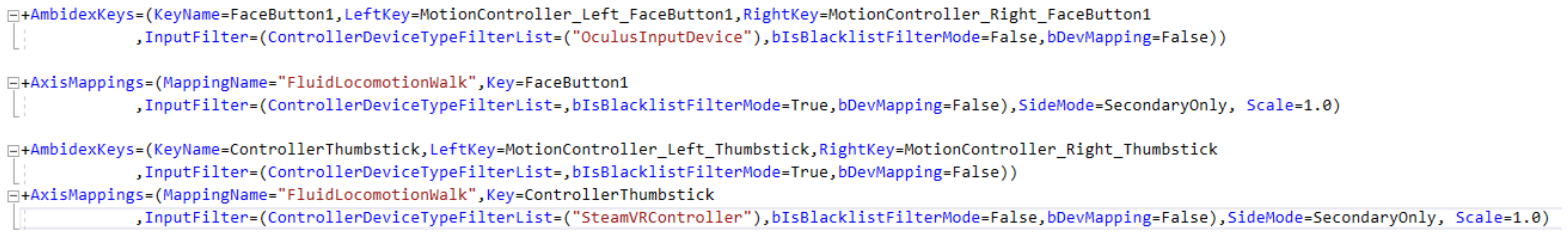


```cpp
if (USVRGameplayUtils::HMDIsType(EHMDDeviceType::DT_OculusRift))
{
    const FName ActionName = IsRight() ? "OculusScanForTeleport_Right" : "OculusScanForTeleport_Left";
    InputComponent->BindAction(ActionName, EInputEvent::IE_Pressed, this, &ARDPlayerHand::InputRequestStartScanForTeleportSpot).bConsumeInput = false;
    InputComponent->BindAction(ActionName, EInputEvent::IE_Released, this, &ARDPlayerHand::InputRequestStopScanForTeleportSpot).bConsumeInput = false;

    InteractorComponent->OnThumbstickButtonPressedEvent().AddUObject(this, &ARDPlayerHand::InputRequestTeleport);
}
else
{
    InteractorComponent->OnThumbstickButtonPressedEvent().AddUObject(this, &ARDPlayerHand::InputRequestStartScanForTeleportSpot);
    InteractorComponent->OnThumbstickButtonReleasedEvent().AddUObject(this, &ARDPlayerHand::InputRequestTeleport);
}
```

Example SVRInput Binding

Settings in DefaultInput.ini for fluid locomotion:



```
+AmbidexKeys=(KeyName=FaceButton1,LeftKey=MotionController_Left_FaceButton1,RightKey=MotionController_Right_FaceButton1
             ,InputFilter=(ControllerDeviceTypeFilterList=("OculusInputDevice"),bIsBlacklistFilterMode=False,bDevMapping=False))

+AxisMappings=(MappingName="FluidLocomotionWalk",Key=FaceButton1
             ,InputFilter=(ControllerDeviceTypeFilterList=,bIsBlacklistFilterMode=True,bDevMapping=False),SideMode=SecondaryOnly, Scale=1.0)

+AmbidexKeys=(KeyName=ControllerThumbstick,LeftKey=MotionController_Left_Thumbstick,RightKey=MotionController_Right_Thumbstick
             ,InputFilter=(ControllerDeviceTypeFilterList=,bIsBlacklistFilterMode=True,bDevMapping=False))
+AxisMappings=(MappingName="FluidLocomotionWalk",Key=ControllerThumbstick
             ,InputFilter=(ControllerDeviceTypeFilterList=("SteamVRController"),bIsBlacklistFilterMode=False,bDevMapping=False),SideMode=SecondaryOnly, Scale=1.0)
```

Conclusion

- Break-up systems into decoupled modular blocks
- Always keep multiplayer and multiplatform in mind
- Stay nimble
- Think about future games

Thank you.