



# DECIMA

## Creating a Tools Pipeline for Horizon Zero Dawn

**Dan Sumaili**  
**Sander van der Steen**

Senior Tools Programmer  
Principal Tools Programmer

Guerrilla  
Guerrilla







# Takeaways

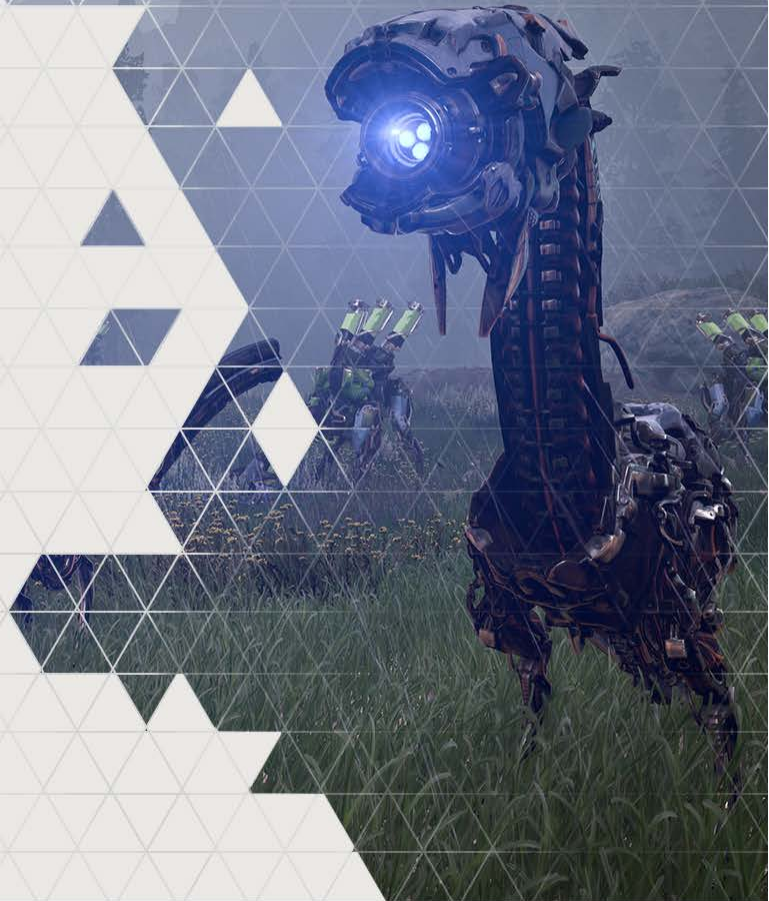
- Insight into why we rebuilt our tools
- How we did it during production
- How it paid off





# Overview

- Our legacy tools
- Rebuilding the tools
- Building a 3D Editor
- Editing data
- Other tools
- Where we are today





## In the beginning... (Nov 2013)

- We had fragmented tools
  - Mix of languages & frameworks
- Long iteration times (20-30 mins.)
- Dissatisfied Users
  - Far behind our competition







# The CoreText Format

- Intermediate plaintext format
- Used for most data
  - Except Textures and Audio
- **Converted** to binary for game





# CoreText Editor

C:\scodev\Guerrilla\KZ4\KIN\Assets\Game\_Assets\Entities\Abilities\Shared\ShieldAbilityTemplate.CoreText - KZ4

File Edit View Format Scripts Window Help

CoreText Zoom In Zoom Out Zoom Reset

Palette

- Components
- Constraint
- Crafting
- Crowd System
- Destructibility
- Electricity
- EntityComponent
- GameLog
- Gesture
- Light
- Localized
- Menu
- MultiMeshActorPart
- PostProcessing
- Property
- QuestSystem
- Resource
- Sound
- Stats
- Template Definition
- Timeline
- Timeline - Do not use anymore
- Timeline Actors
- Timeline Actors - Do not use anymore
- Timeline Events - Actor
- Timeline Events - Animation
- Timeline Events - Character

ShieldAbilityTemplate.CoreText

Prefix Associations Unselected Wires Root > | ShieldAbility > |

Inputs

InventoryEntityResource

InventoryPlaceEntityAbilityResource

ControlledEntityResource

PlayerConstructedEntityComponentResource

EntityPlacementPositionCheckerComponentResource

AIPositionPickerResource

InventoryShieldAbility

GameLink

Set State

MP Restart

Properties - InventoryPlaceEntityAbilityResource: InventoryShi...

General

Name InventoryShieldAbility

NamesIdentifier ☒

ObjectAttributeAnimatorResource

Logic

UpdateFrequency 15Hz

Lockable ☐

ZoomLockable ☐

ChildEntityResources

JumpableFrom ☒

EntityComponentResources

0 value ... source

TemplateDefinition

Class that defines a template, contains a (hopefully complete) list of all objects that need to be instantiated, a list of exposed link targets that can be linked to from the outside, and a list of exposed attributes that can be overridden on instantiation

Properties - InventoryPlaceEntityA... GameLink Object Browser

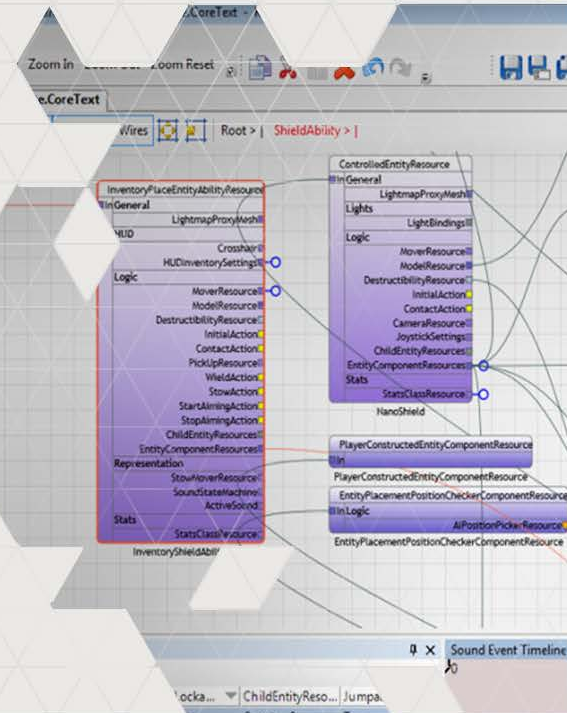
Sound Event Timeline

Name	NamesIden...	ObjectAttribute...	UpdateFreq...	Lockable	ZoomLocka...	ChildEntityReso...	JumpableFr...
InventoryShiel...	True		15Hz	False	False	Count = 0	True

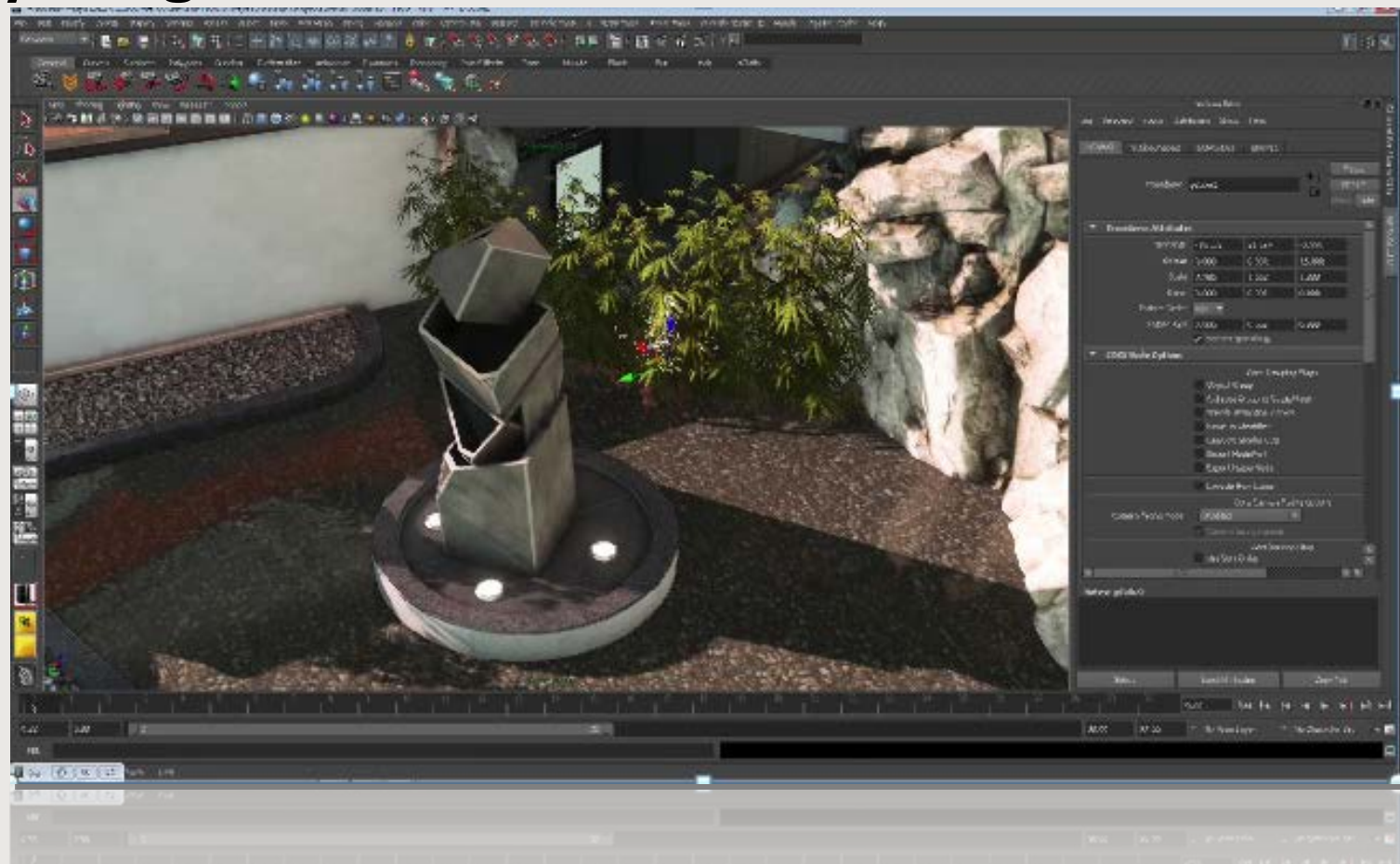


# CoreText Editor

- Used to create & configure game data
- Written in C#, WinForms, ATF
- Absurdly complex code









# Maya Plugin

- Exports Maya data to .CoreText
- Used Decima's Renderer
- C++, Python, Maya API
- Killzone Shadow Fall Art Tools
  - <http://www.gdcvault.com/play/1020711>

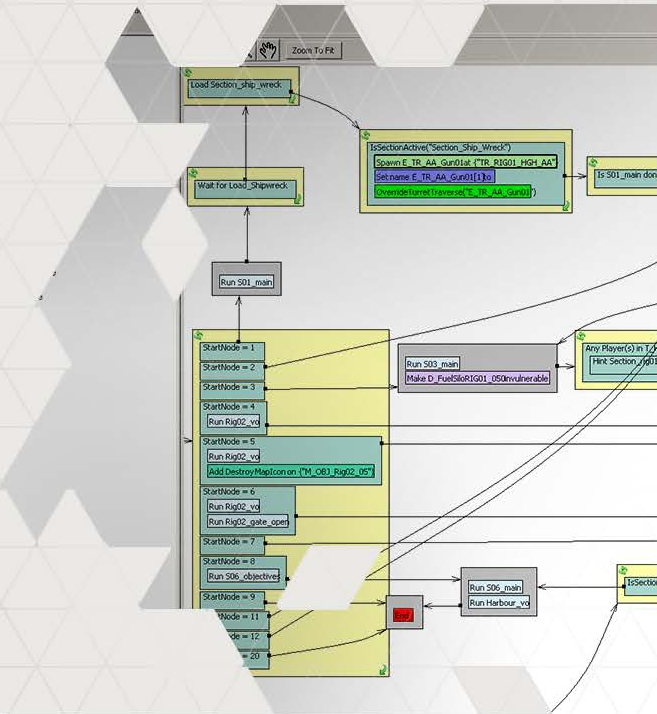


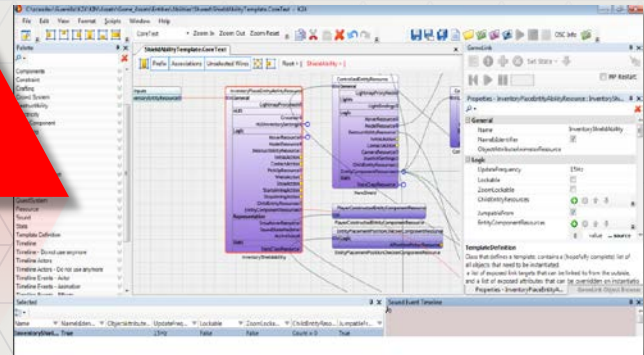
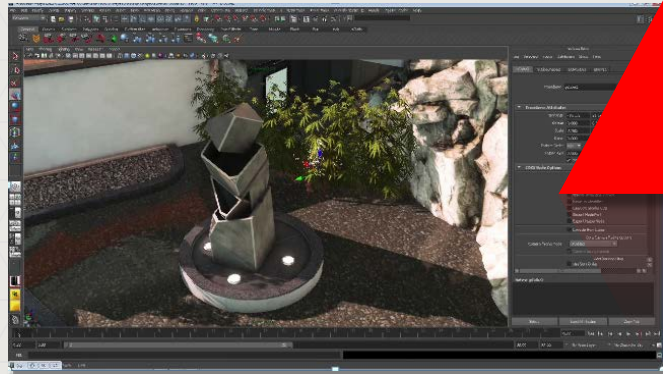
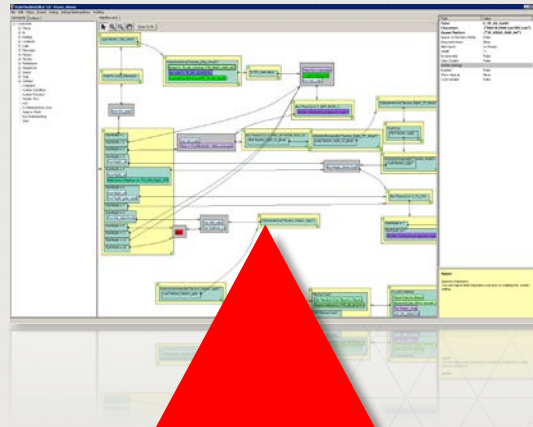




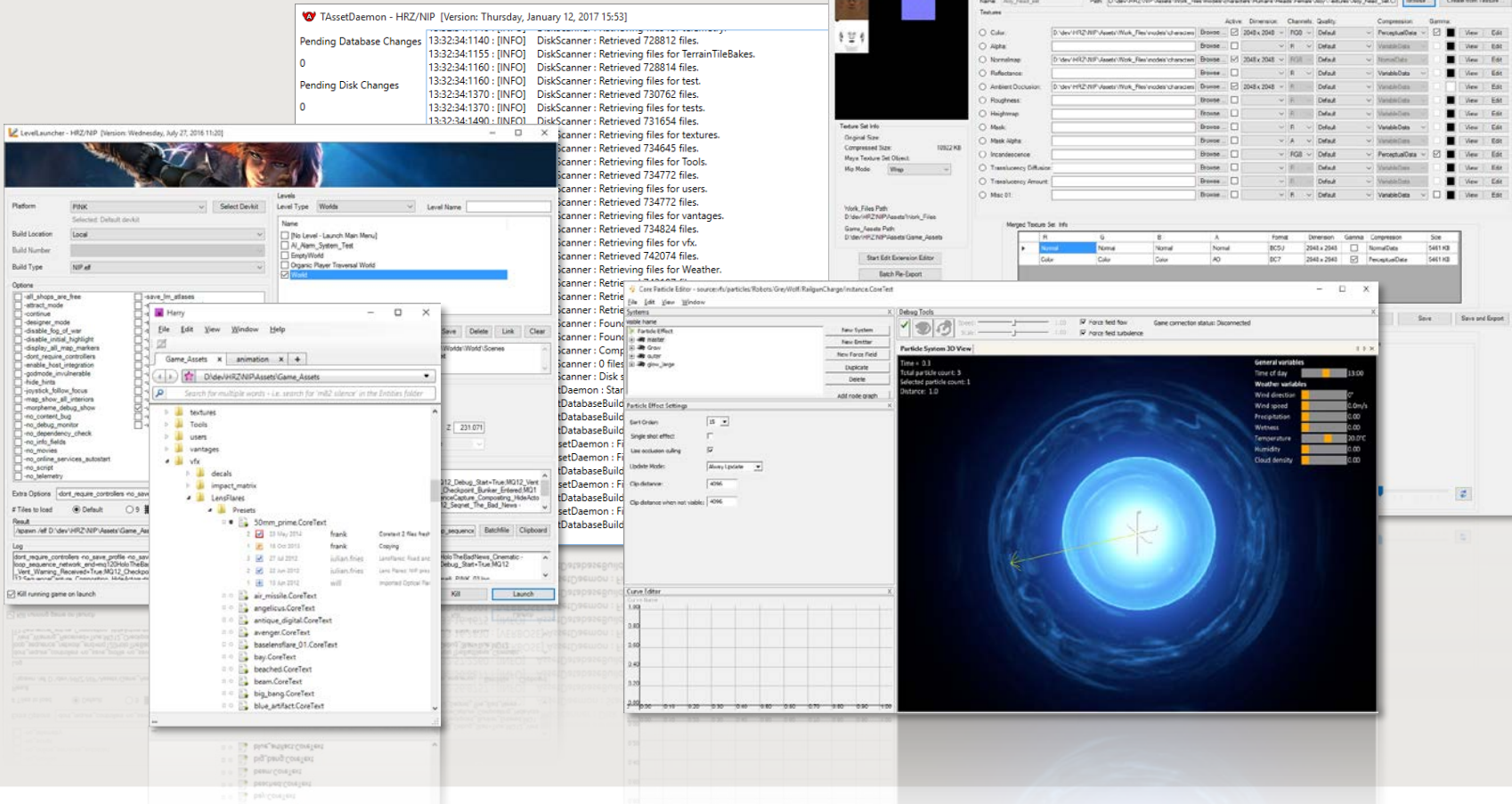
# State Machine Editor – ‘SMED’

- Node-based State Machine editor
- Generates LUA script
- Maintained by a Technical Designer
- Written in Python











# Debug UI

- In-game UI framework
  - Created during **Shadow Fall**
- Programmers found it easy to use
  - Led to many in-game debug tools









# Horizon Zero Dawn

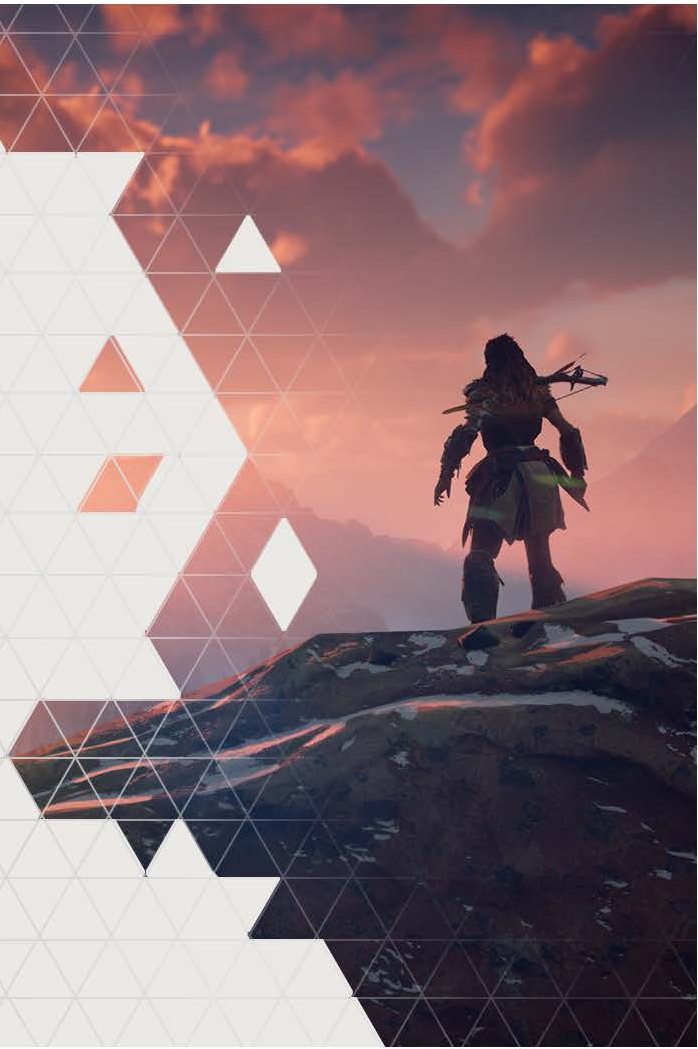
- Horizon was new for us
- New types of content
- Tools team was small
  - Had to use time and resources efficiently





# Horizon Zero Dawn

- The old ways would not scale
- We must begin again...





## Tools Goals: Code

- Create a Framework to build Tools
- Re-use **Decima** engine libs
- Making tools should be easy





# Tools Goals: Editor

- Users need tools that fit their work
  - Discipline-specific 'Editing Contexts'
  - Built from reusable components
- User should be able to stay in **one** Program
  - Decima Editor







TOOLS FRAMEWORK

GAME CODE

CORE

PIGS

OS





# Decima Engine

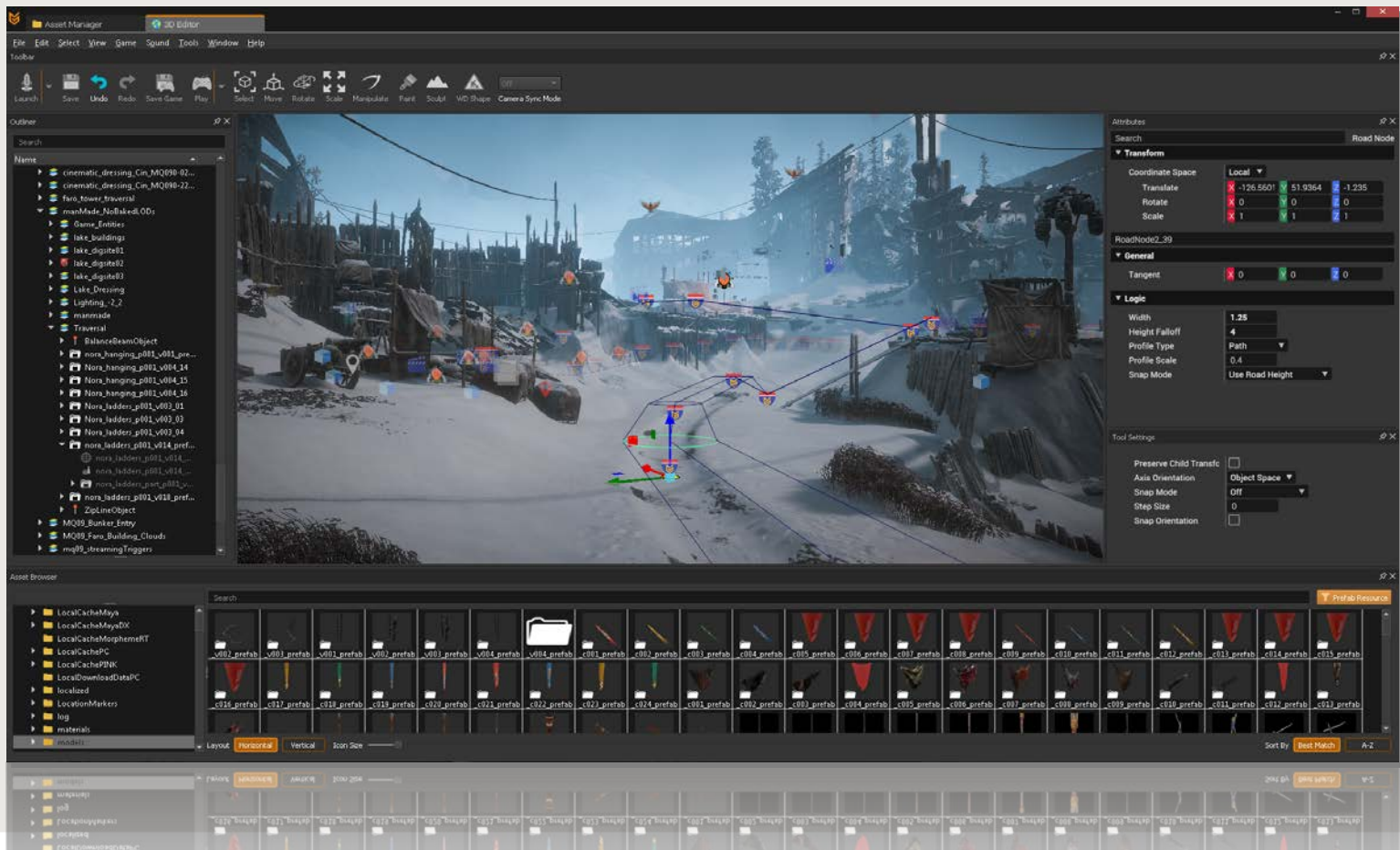
- Data Serialization & Deserialization
- Runtime Type Introspection (RTTI)
- File I/O, Networking, Threads, Jobs
- Image Readers & Writers
- Compression, Hashing, Algorithms
- Unit Testing Framework





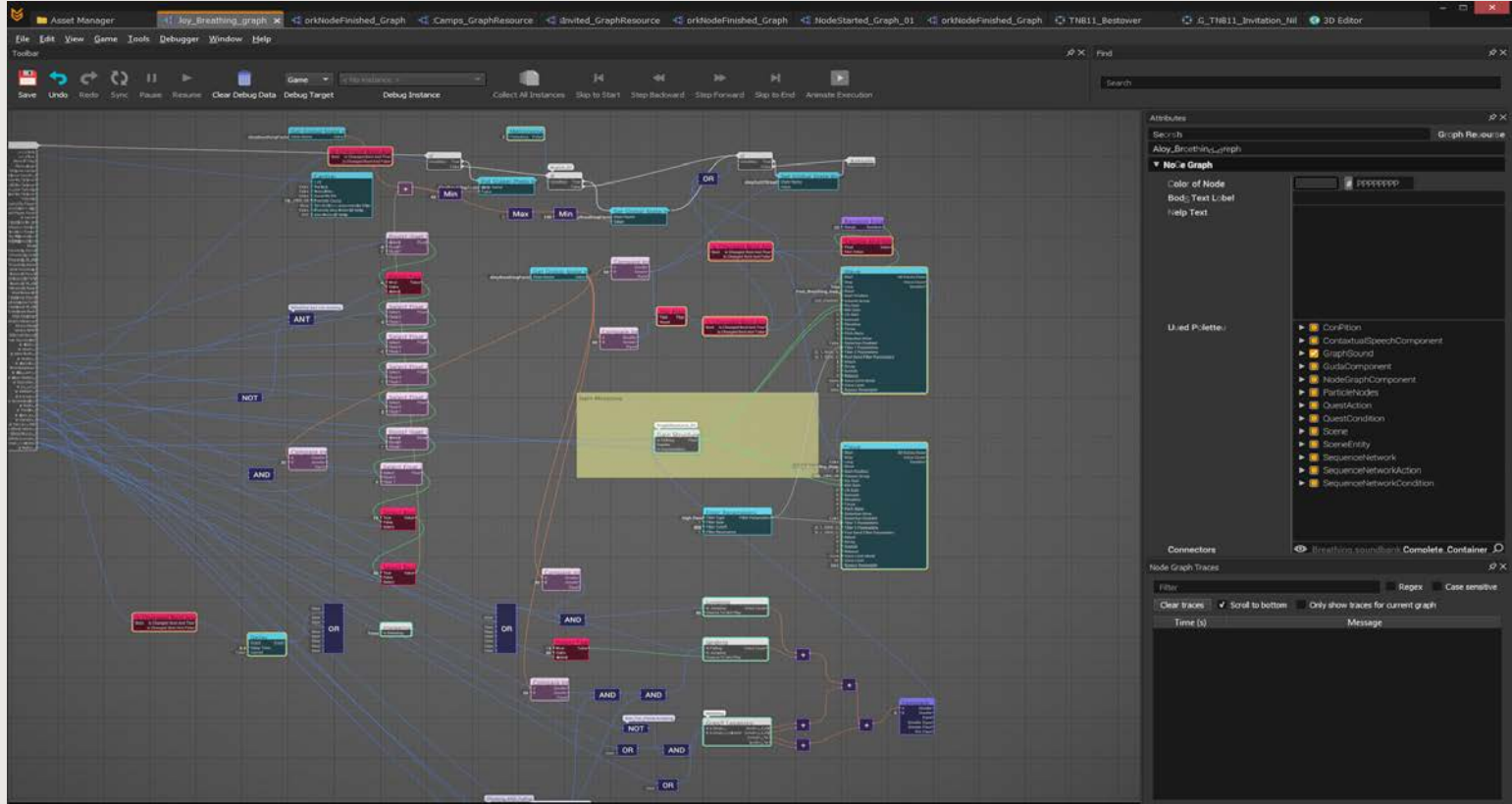


# Level Editor





# Visual Script Editor





# Conversation Editor

The screenshot displays the Conversation Editor interface, which is used for creating and editing AI conversations. The interface is divided into several panels:

- Asset Manager:** Located at the top left, it shows the current project assets, including "X/G\_TB111\_Finale\_Nil" and "O/G\_TB111C\_Nil".
- Toolbar:** Below the Asset Manager, it contains buttons for "Save", "Undo", "Redo", "Sync", "Playthrough", and "Debug Target".
- Main Canvas:** The central area displays a flowchart of the conversation logic. It starts with a "Talk to Nil" event, leading to a "StrangerGreet" event. From there, it branches into "Stranger\_FightChoice" and "Friend\_FightChoice". These lead to various dialogue options like "Stranger\_What", "Stranger\_Yes", "Stranger\_No", "Friend\_What", and "Friend\_Yes". The flowchart also includes a "FightLoop" and a "Friend\_FightChoice2" event.
- Event Palette:** Located on the left side, it lists various events and actions that can be added to the conversation, such as "Camera - Bone Locator", "Camera - Camera Actor", "Camera - Set Fact", "Camera - Shared Position Lo", "Camera - Smooth Aim Locat", "Dialogue for Alloy", "Dialogue for Nil", "Characters", "Alloy", "Nil", "Look At", "Nil Look At Event Locato", "Nil Look At Events", "Nil's AI Dynamic Obstacles", "Nil's Bone Locators", "Nil's Bone Locators.02", "Nil's Conversation Animati", and "Nil's Create Entity Actor".
- Sequences:** A timeline view at the bottom left showing the sequence of events over time. It includes a "Filter" section and a "Timeline" section with a "Time" axis ranging from 0 to 800. The timeline shows events like "Nil Look At Alloy\_06", "Nil Look At Alloy\_07", "Nil's AI Dynamic Obstacles", "Nil's Bone Locators", "Nil's Bone Locators.02", "Nil's Conversation Animati", and "Nil's Create Entity Actor".
- Screenplay:** A panel on the right side showing the dialogue text for the conversation. It includes a "Sequence" section and a "Screenplay" section. The dialogue is organized into a table with columns for "Character", "Dialogue", and "Action". The current sequence is "StrangerGreet", and the dialogue is as follows:

Character	Dialogue	Action
Nil	(Nil is a stranger) Alloy speaks to Nil at the waterfalls. I'm so glad you came. In a way, I feel like I already know you. Every pile of corpses was like reading your journal.	
Alloy	You don't know me. I've done more than kill bandits.	
Nil	I don't doubt it. But bandits were my quarry, and their camps my hunting grounds. Now they're bare.	
Nil	So I had to know if you were just faster than me, or more skilled too.	
Alloy	So I was so much faster than you, and I was so much more skilled too.	
Nil		
Alloy		

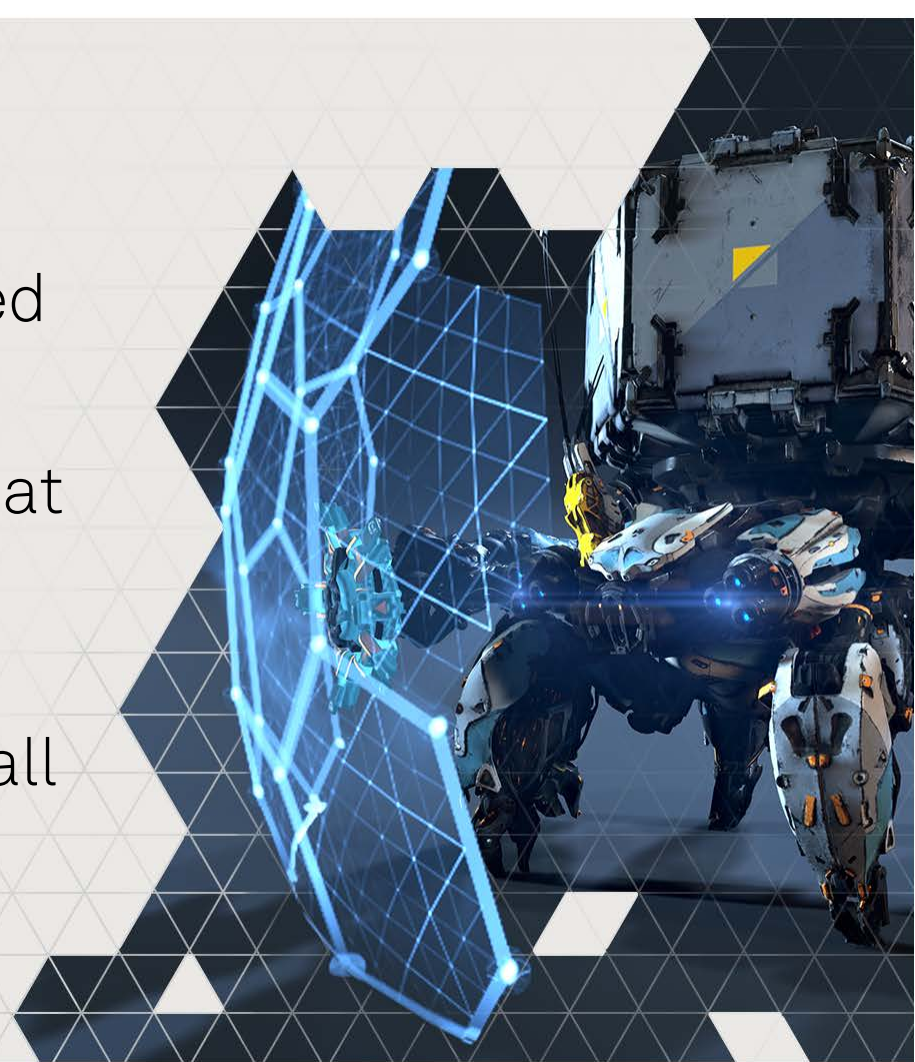


# The first steps to a new editor



# Where do we begin?

- Production is getting started
- Users already have tools that they're familiar with
- We would need to start small and and minimize risk







# Where do we begin?

- We decided to focus on the Audio Team
- A small group of users with specific needs
- Their workflow could be used to test the Framework's viability





# Where do we begin?

- The Audio Team primarily worked with NodeGraphs
- NodeGraph is our High Performance Visual Scripting language
  - Generates native C++ code, that runs on target
- Proven tech, used for audio in **Killzone: Shadow Fall**
  - <http://www.gdcvault.com/play/1020559> [Varga, Woldhek 2014]





# CoreText Editor

The screenshot displays the CoreText Editor interface, which is used for creating and editing game templates. The main workspace shows a visual graph of a template named "ShieldAbilityTemplate.CoreText". The graph is organized into several components, including "Inputs", "InventoryPlaceEntityAbilityResource", "ControlledEntityResource", and "PlayerConstructedEntityComponentResource". These components are interconnected with various resources and actions, such as "LightmapProxyMesh", "HUD", "Logic", "DestructibilityResource", "InitialAction", "ContactAction", "PickUpResource", "WieldAction", "StowAction", "StartAimingAction", "StopAimingAction", "ChildEntityResources", "EntityComponentResources", "Stats", "StatsClassResource", "InventoryShieldAbility", "PlayerConstructedEntityComponentResource", "EntityPlacementPositionCheckerComponentResource", and "AIPositionPickerResource".

On the left side, there is a "Palette" containing a list of components and resources that can be added to the graph. The "Selected" panel at the bottom shows the properties of the selected component, "InventoryShieldAbility".

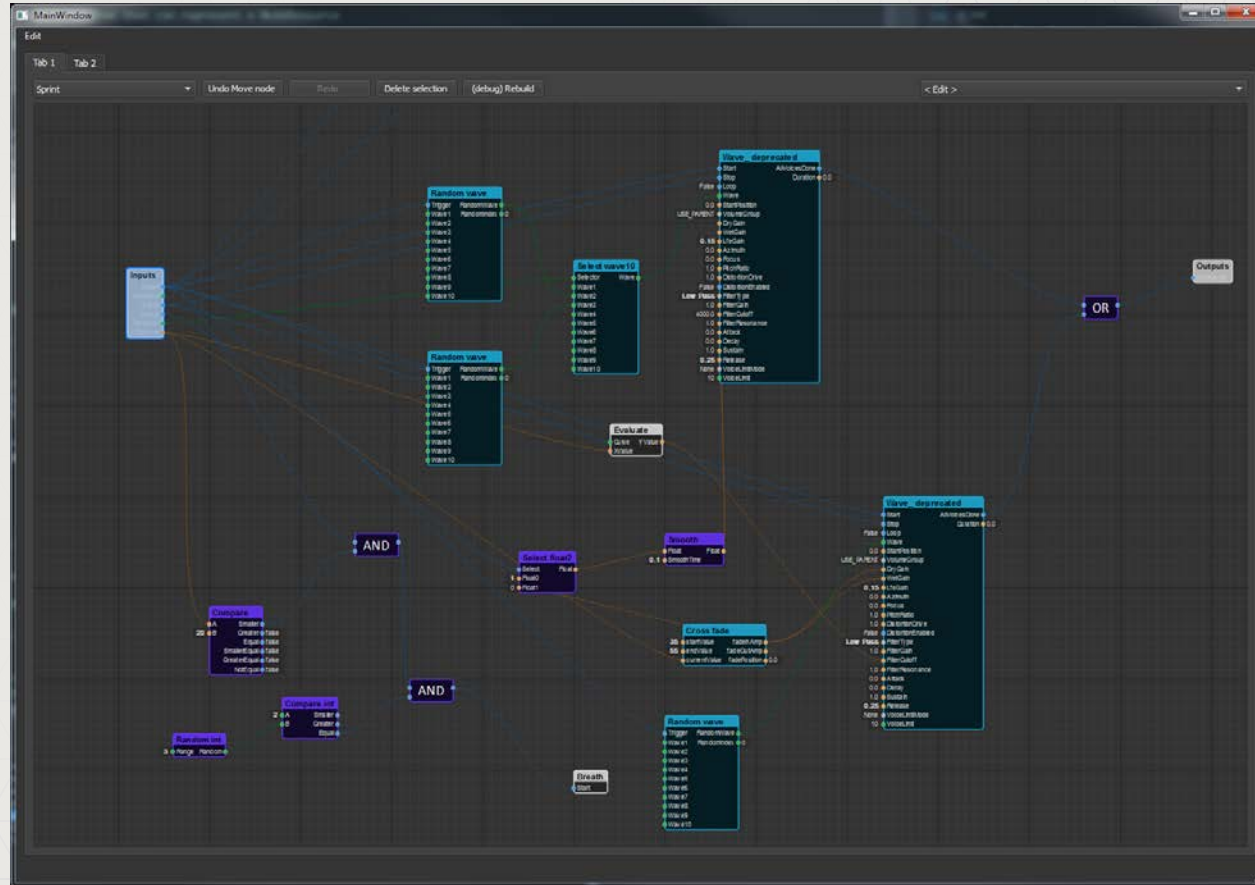
On the right side, there is a "Properties" panel for the selected component, "InventoryPlaceEntityAbilityResource: InventoryShieldAbility". It shows various properties and settings, including:

- General**
  - Name: InventoryShieldAbility
  - NamesIdentifier: ☒
  - ObjectAttributeAnimatorResource
- Logic**
  - UpdateFrequency: 15Hz
  - Lockable: ☐
  - ZoomLockable: ☐
  - ChildEntityResources: ☒
  - JumpableFrom: ☒
  - EntityComponentResources: ☒

Below the Properties panel, there is a "TemplateDefinition" section that provides a brief description of the template and its components. It states: "Class that defines a template, contains a (hopefully complete) list of all objects that need to be instantiated, a list of exposed link targets that can be linked to from the outside, and a list of exposed attributes that can be overridden on instantiation." Below this, there are two tabs: "Properties - InventoryPlaceEntityA..." and "GameLink Object Browser".

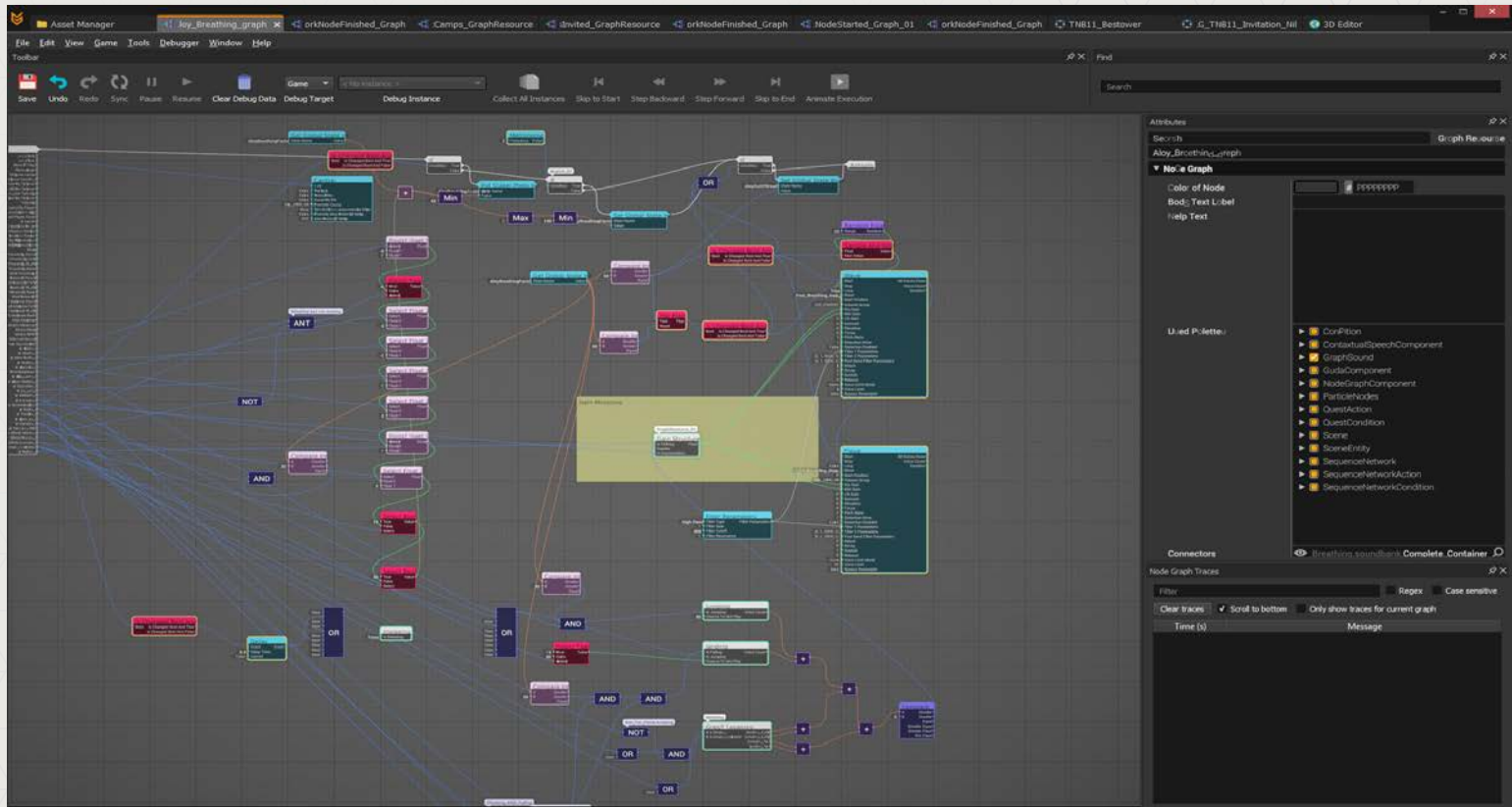
At the bottom of the interface, there is a "Sound Event Timeline" panel, which is currently empty.







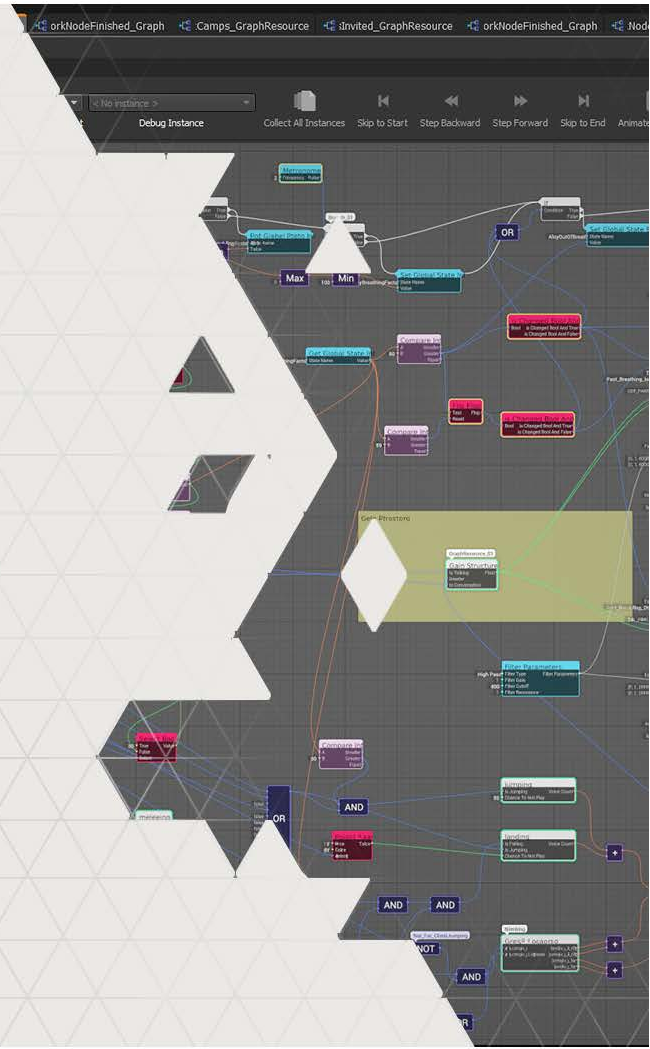
# Node Graph Editor Context







- We reached feature parity within a few months
- New features could be implemented easily
- Programmers from other teams could also contribute







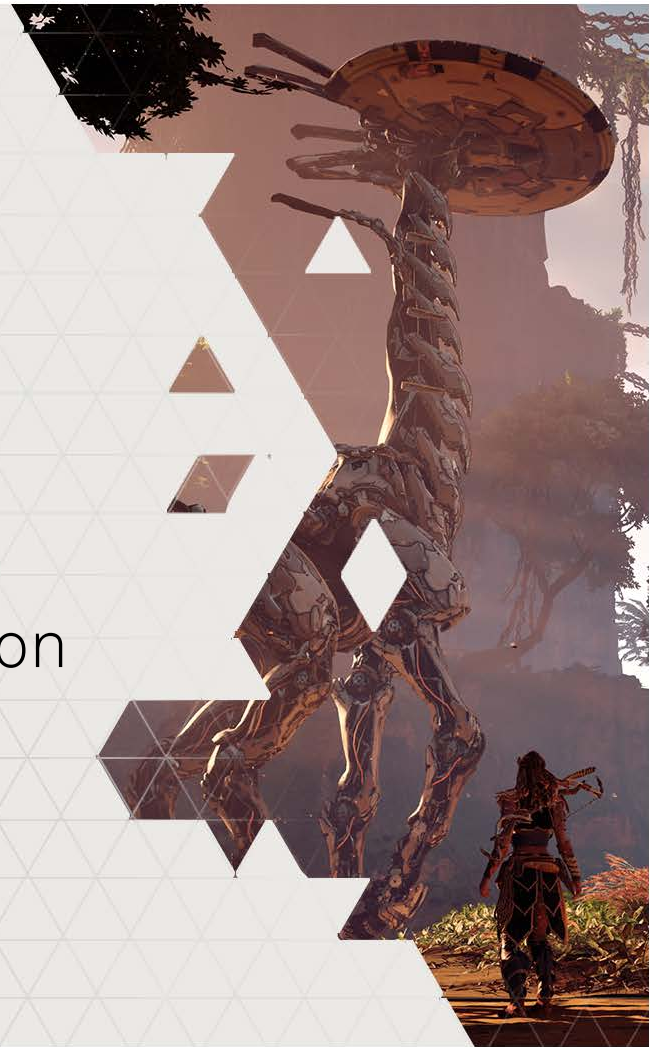
# A new workflow for Game Design

Creating a 3D Editor



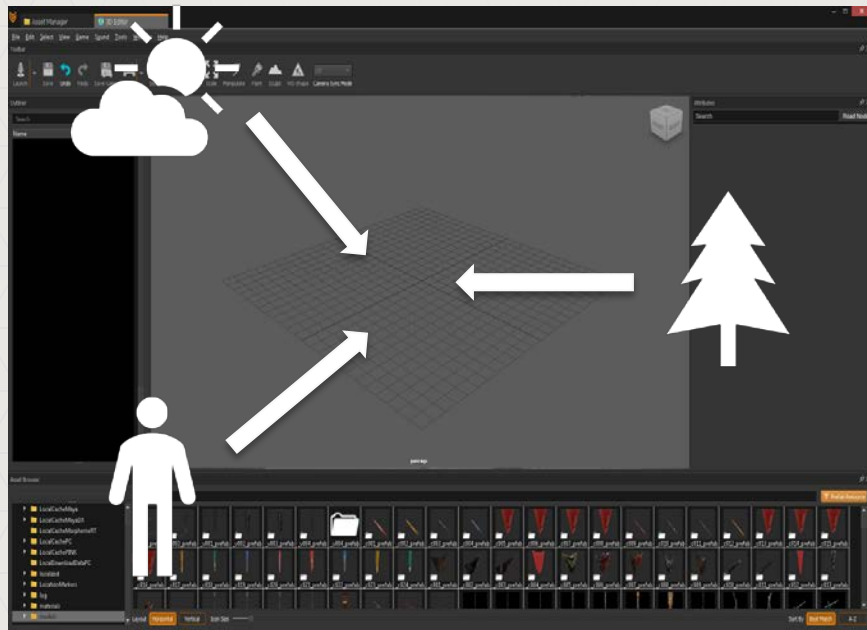
# Why a 3D Editor?

- *Maya* not suitable for quest content
- *3D Editor* was urgently needed
- Focus on Object placement & creation





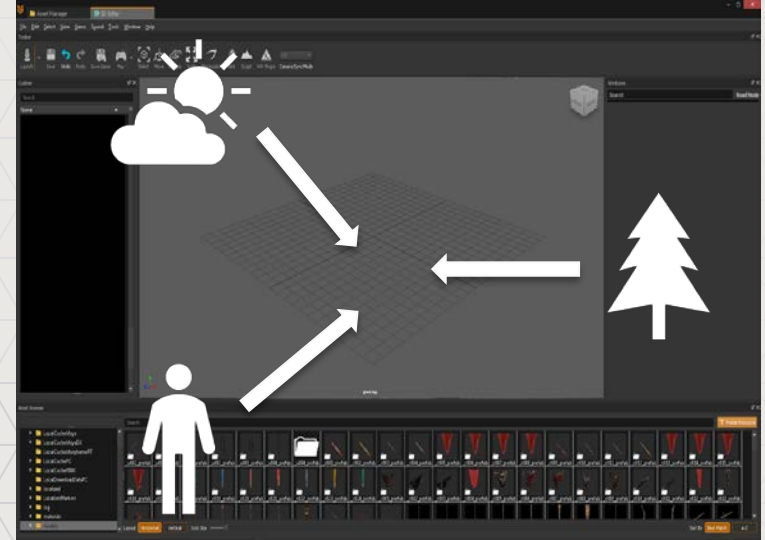
# Integrate sub-systems or full game?





# Integrate only sub-systems into Editor?

- + Purpose build, controllable
- + Fast
- New code-path  
Longer development
- Visual/behavioral disparity







# Integrate full game into editor?

- + All game systems available
- + *DebugUI* (tools) for free
- + Single code path (play/edit)
- Overhead of running game

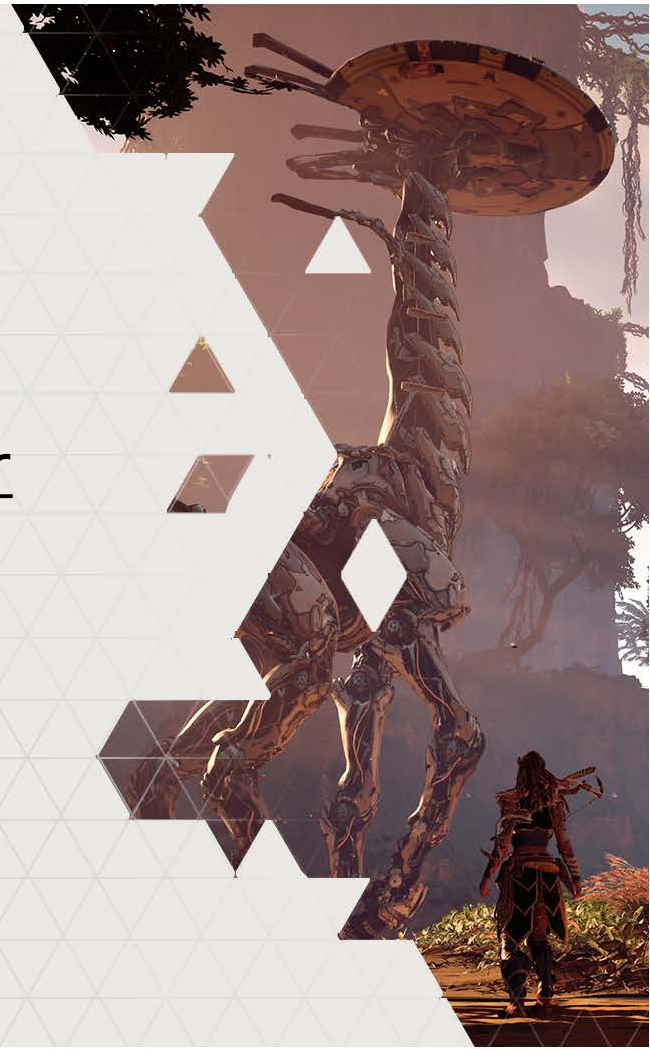




# 3D Editor – Architecture

Due to time constraints:

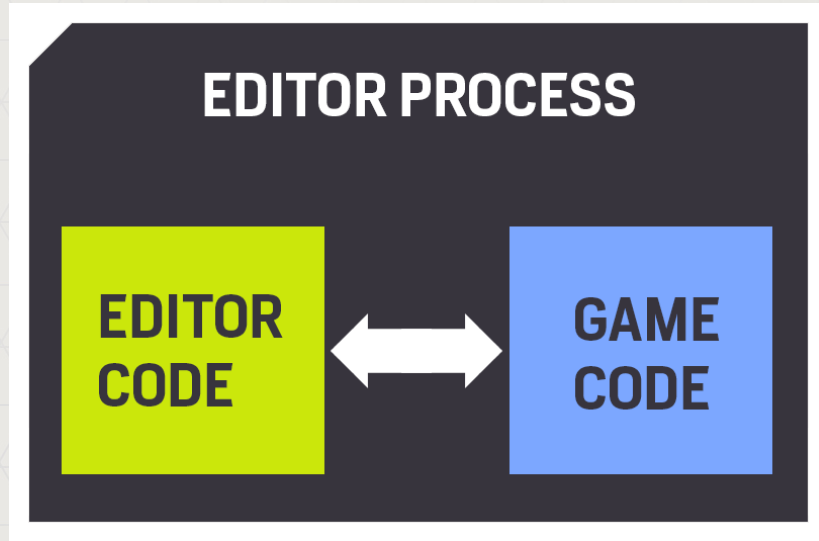
Integrate full game into Editor



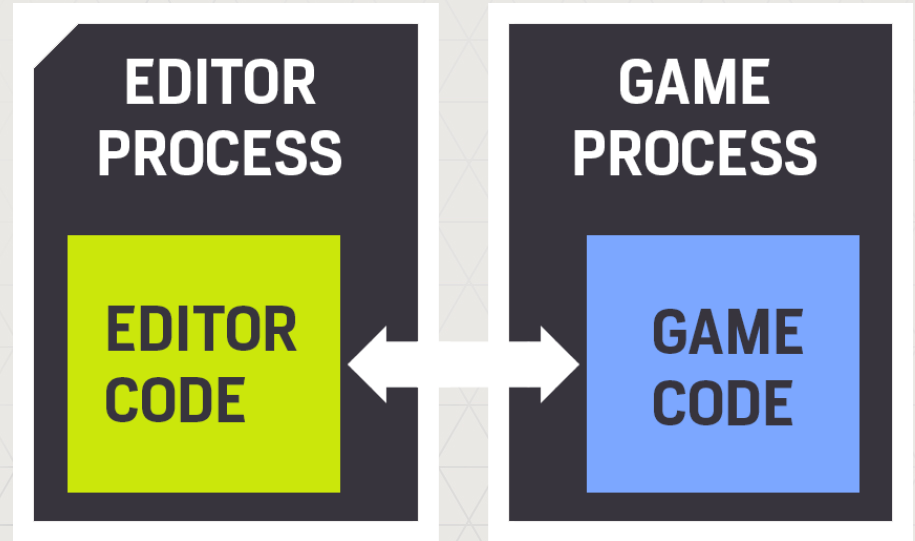


# How to integrate a full game into Editor?

*In process?*



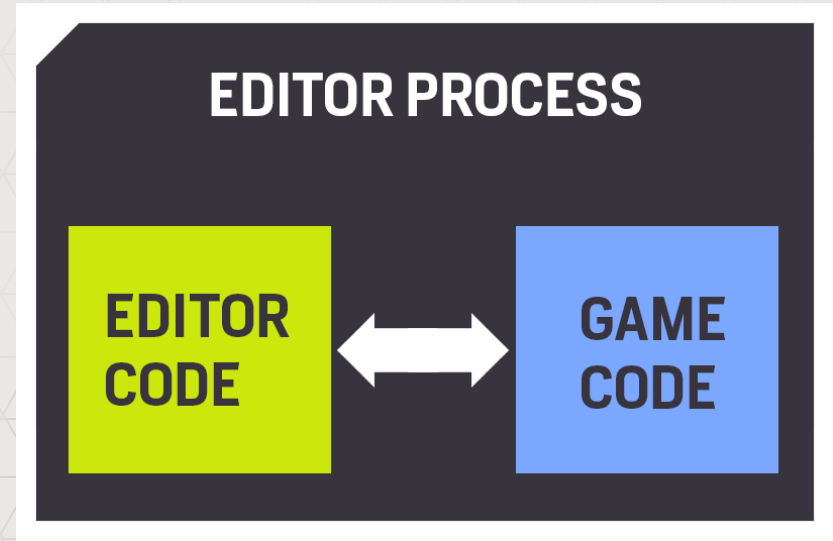
*Out of process?*





# In-process integration

- + Fast to setup
- + Direct synchronous data changes
- Crash in game = crash in editor







# In-process integration

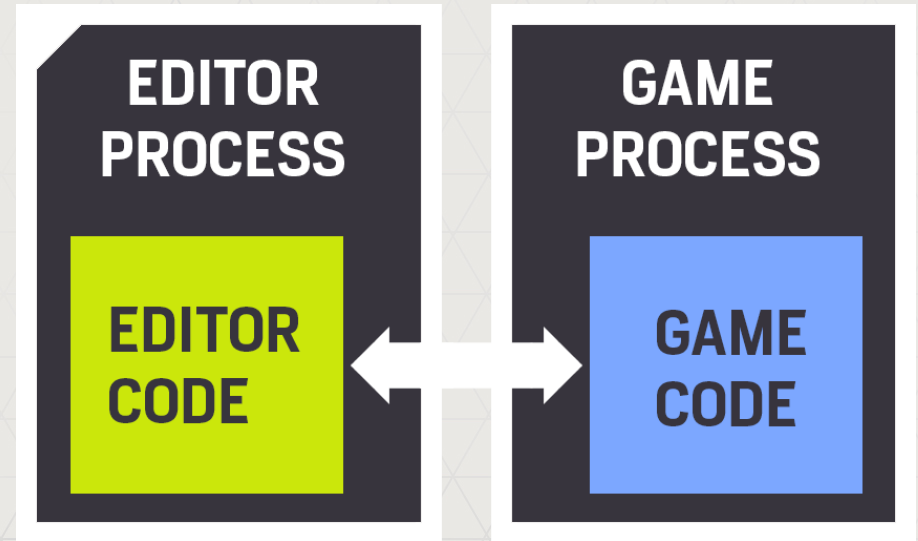
- + Fast to setup
- + Direct synchronous data changes
- Instability
- Enforcing code separation





# Out of process integration

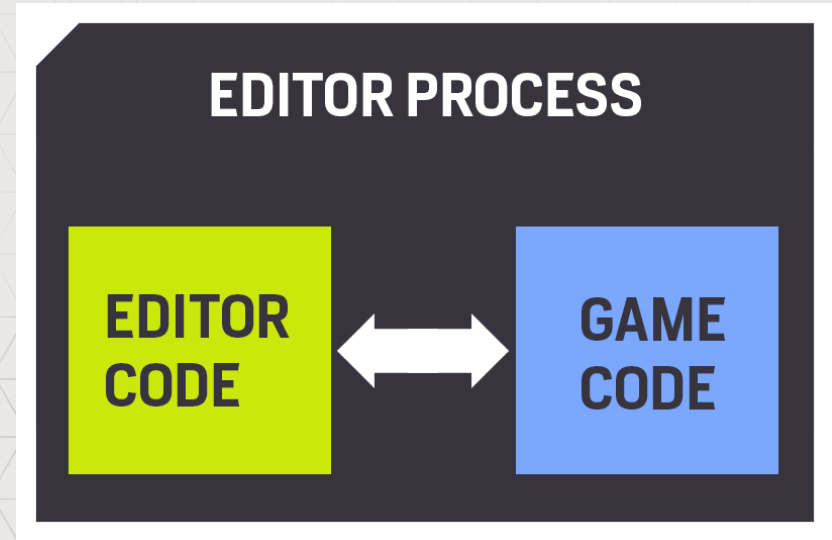
- + Stability
- + Cleaner code
- + Responsive UI
- Longer initial development
- Harder to debug 2 asynchronous processes





## 3D Editor: Initial design

- Develop in-process with a full game
- Goal: Out of process should remain possible









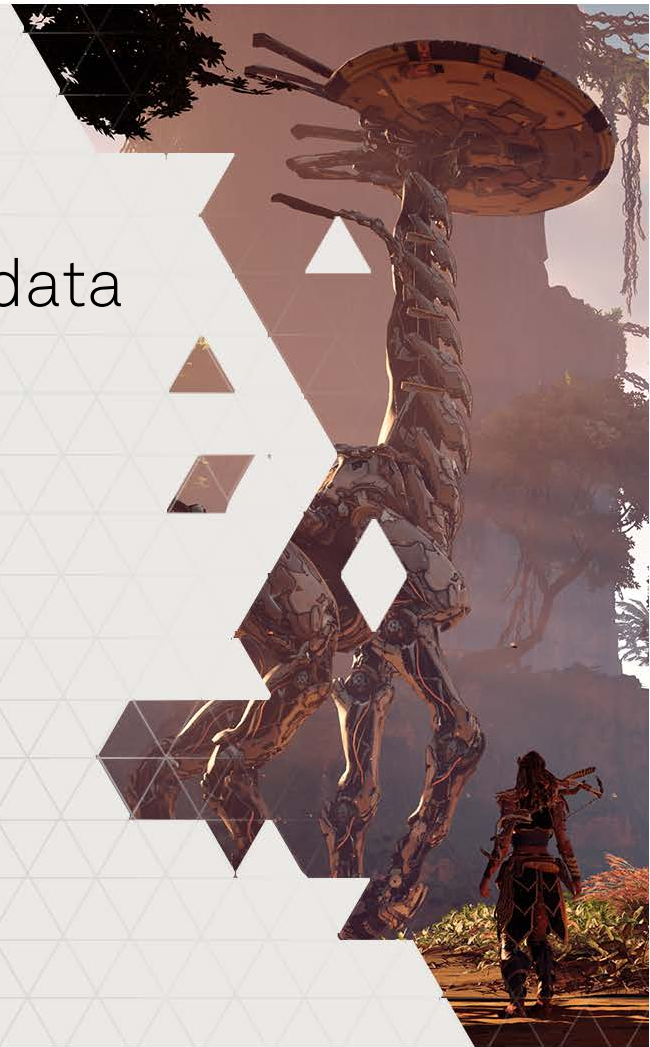
# Editing content in a running game

Required sub-systems for Level Editor



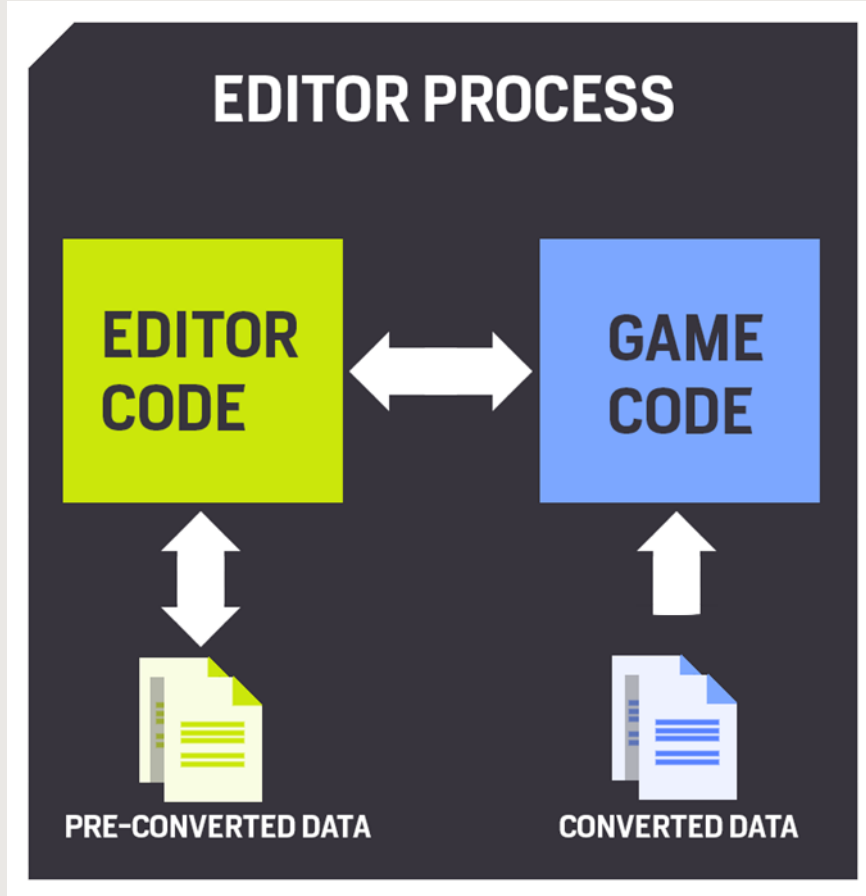
# Game data

- Problem: Game reads binary game data
- Creation is *destructive*
  - Shader compilation
  - Meshes optimization
  - Data packing





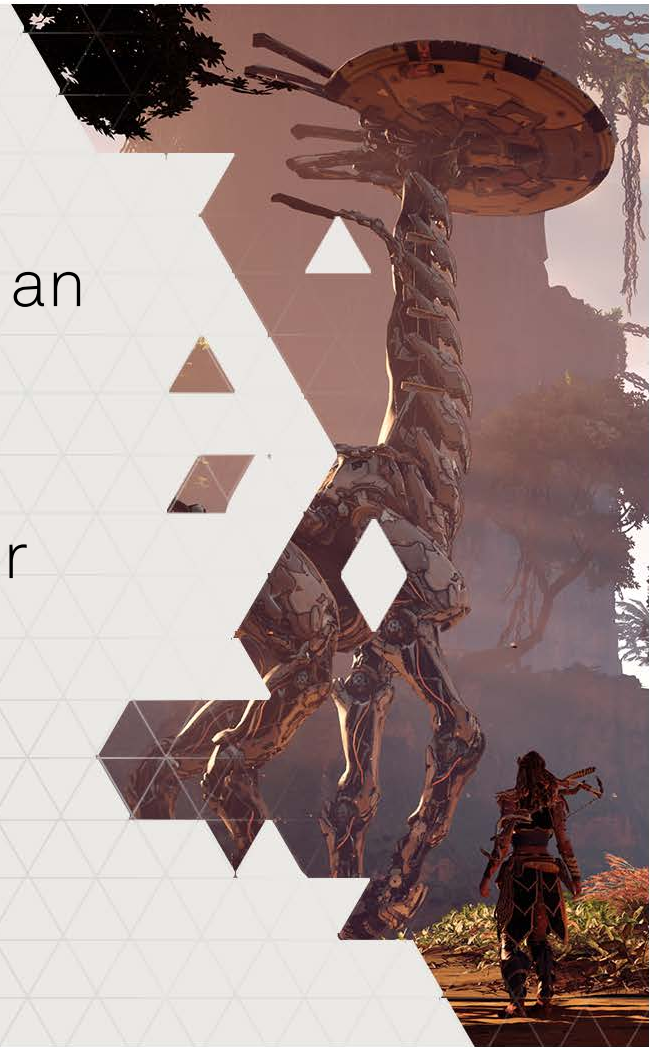
# 3D Editor: Initial design





# EditorNode

- Provides **all** editing functionality for an object in game
- Game object does not know of Editor Node
- Keeps code clean, no editor specific code interwoven in game code





# Only WorldSpace in-game

- Problem: Less hierarchy in game
- Flat list of objects
  - Fast and efficient
  - Transforms in World Space







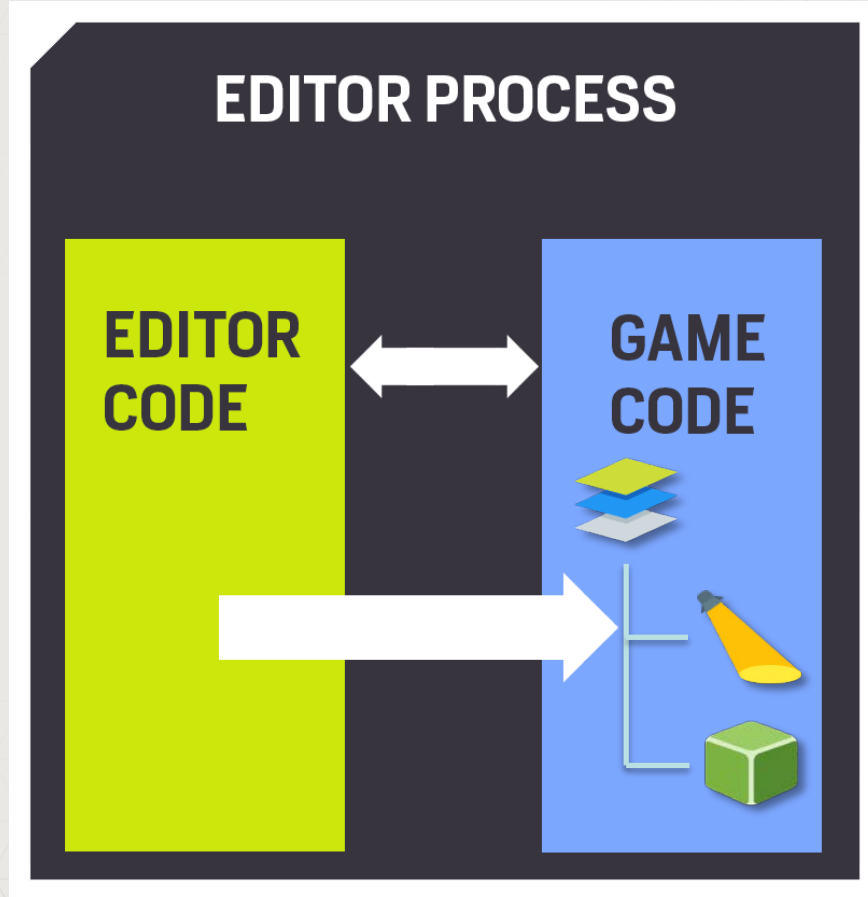
# EditorNodeTree

- Hierarchical structure of in-game data
- Constructed from game data, reflects hierarchy of objects in-game.
  - When entering “*edit mode*”
  - Top down (starting at root)



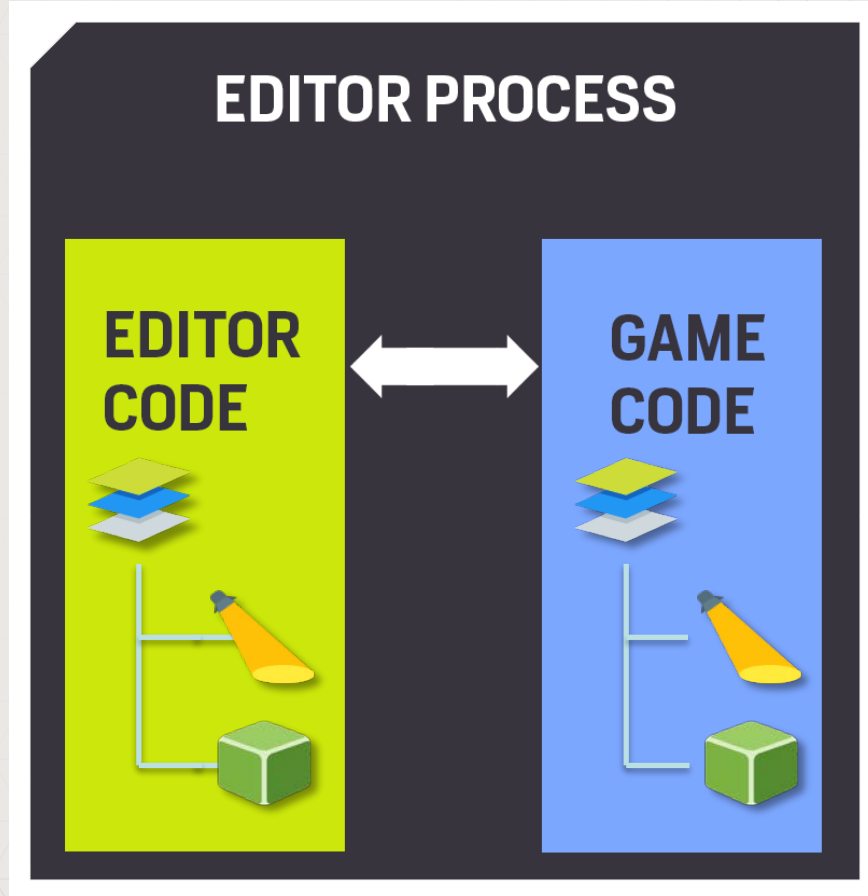


# Accessing the EditorNodeTree





## 2 Editor Node Trees



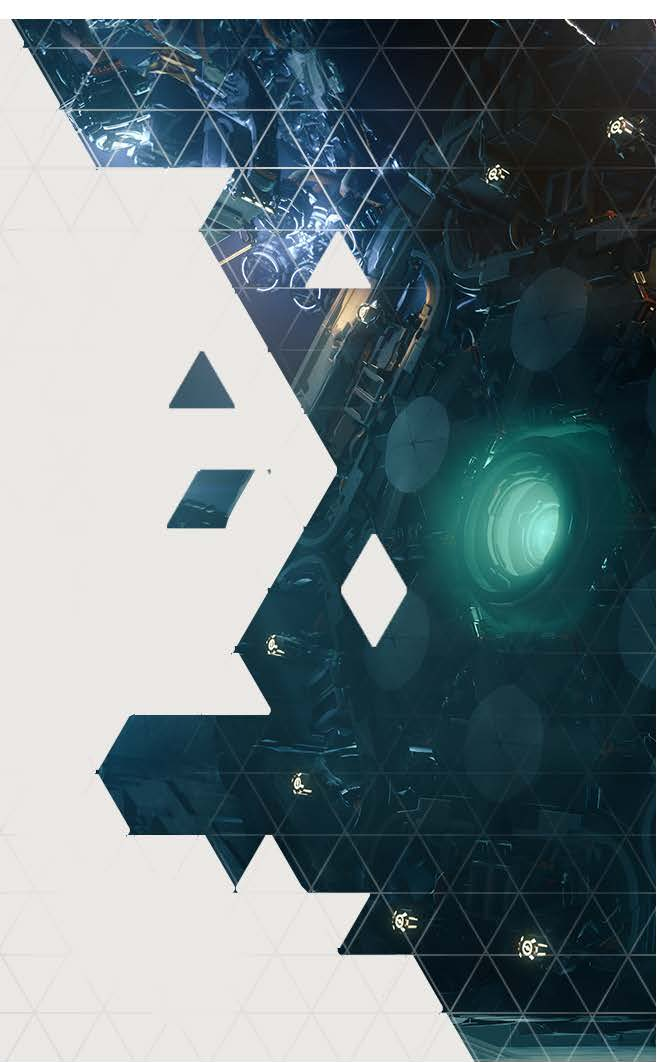


# Making Changes to Objects



# What we didn't want

- Proxy representations
- String-based accessors
- Editor-specific boilerplate
- Code generation steps in the build

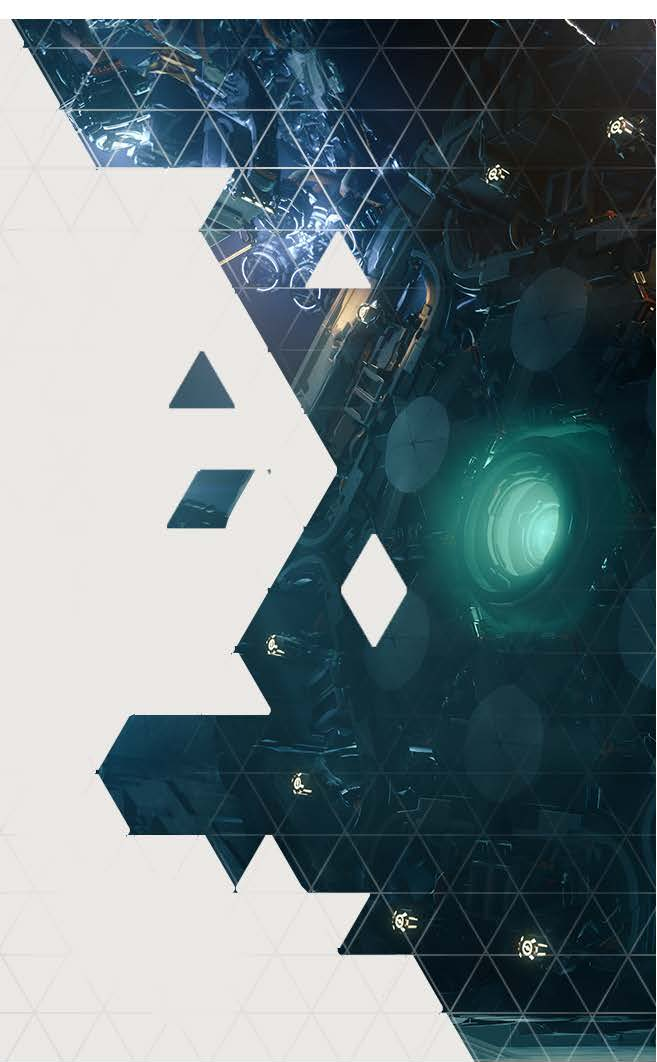






# What we wanted

- Work directly with concrete objects
- Directly modify object members
- Re-use existing systems and utilities to create and transform data



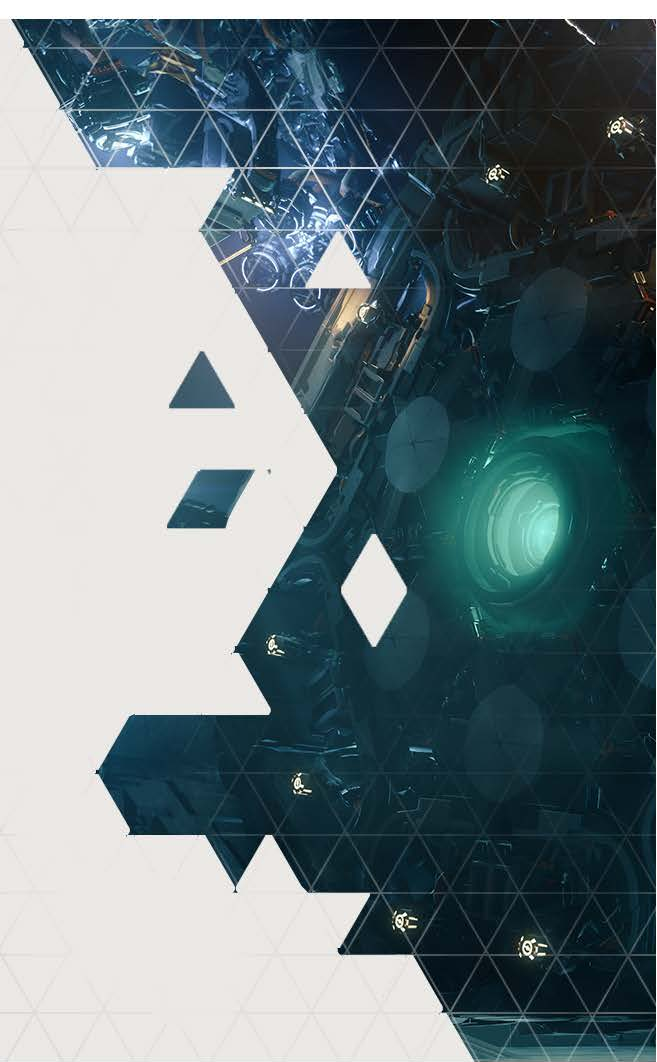


# Making changes to objects

- Being able to make changes is good, but...

“What is best in life?”

- An **Undo** system is essential for users to remain in control and have the freedom to experiment





# Undo System - Goals

- Simplicity & Reliability
- No virtual 'Do' and 'Undo' functions
- No new undo code for new features
  - Prevent transformation asymmetry





# Undo System

- The ideal scenario:
  - An undo system that automatically detects object changes, and knows how to revert them
- The compromise, in our case:
  - Beforehand, code indicates which objects will be modified, and when the modifications are completed



# Undo System - Transactions

- Describes all modifications in a change
  - Commands represent individual changes
- Built on the RTTI system
- Subscribers are notified of all transactions

**ADD OBJ**

**ADD OBJ**

**SET VALUE**

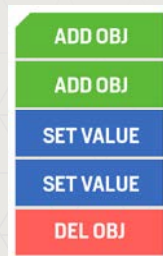
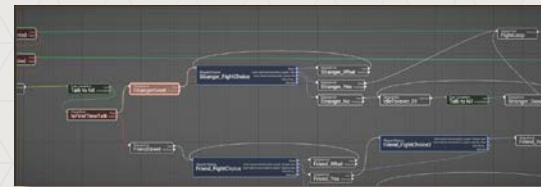
**SET VALUE**

**DEL OBJ**





# Transaction Notifications

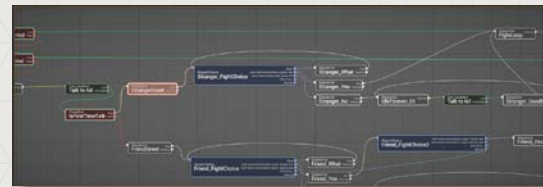




# Transaction Notifications



- Code Separation
- Changes can be made from anywhere
- No knowledge of do and undo in editor code





# Undo System - DiffUtil

- Building transactions is hard
- **DiffUtil** builds them
  - Produces transactions by looking at changes that have already happened





# Undo System - DiffUtil

- Before changes are made:
  - DiffUtil takes **snapshot** of objects' state
- After changes:
  - Analyzes what has changed between the snapshot and the current state







## DiffUtil – Code Example

```
// Get the spotlight to edit...
pSpotLightResource spotlight = GetSpotlightToEdit();

// Take a snapshot of the spotlight's current state
DiffUtil diff;
diff.TakeSnapshot(spotlight);

// Update the cone angle
spotlight->SetConeAngle(30.0f);

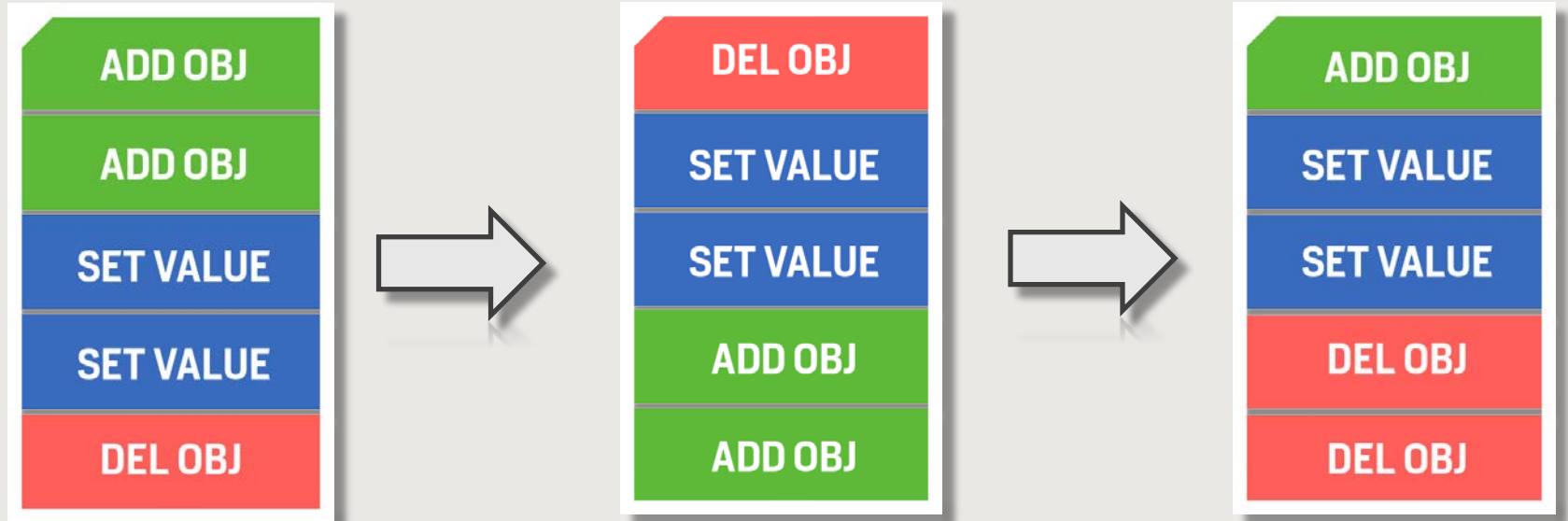
// Indicate that we have finished making changes
diff.CommitChanges("Set Cone Angle to 30");
```

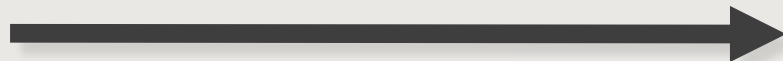
```
diff.CommitChanges("Set Cone Angle to 30");
\\ Indicate that we have finished making changes
```





# Undo System – Reverting Changes

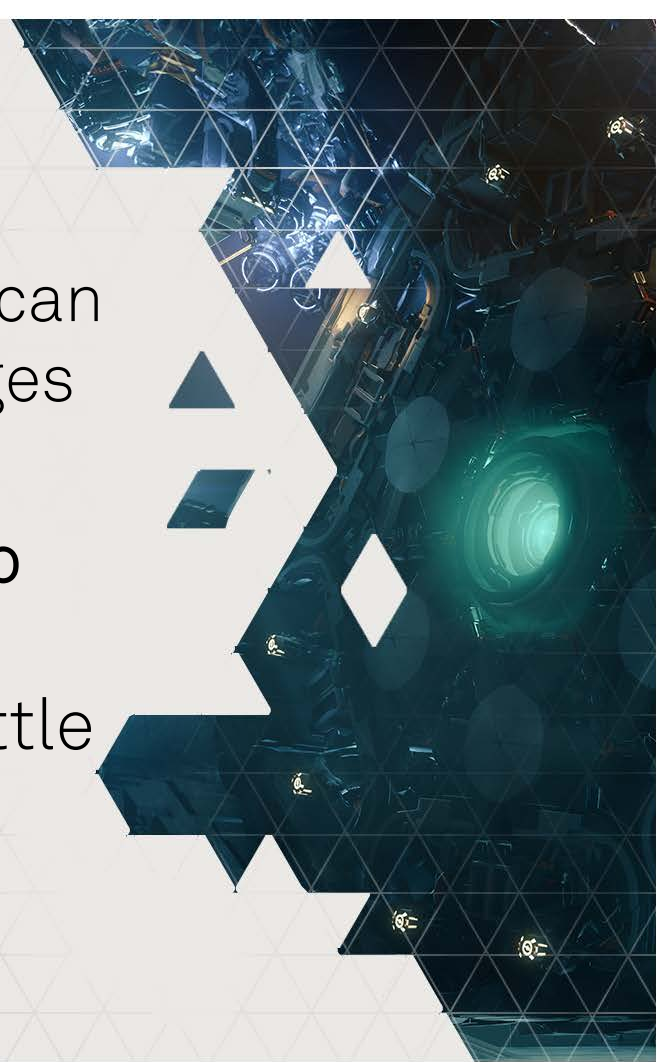






# What worked?

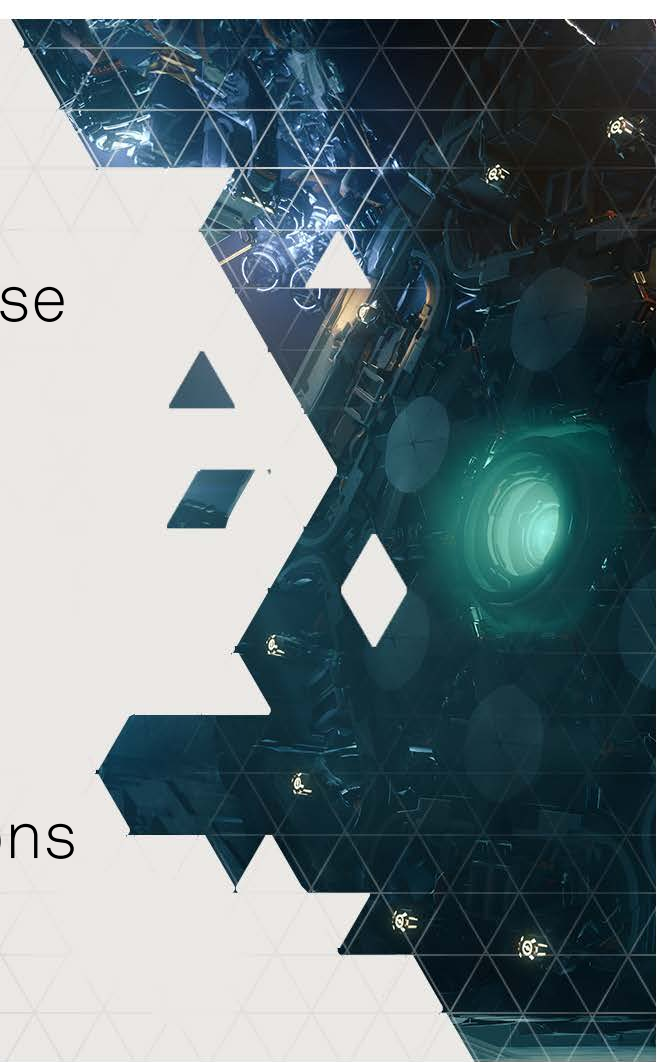
- A reliable, transparent system that can represent arbitrarily complex changes
- No distinction between **do** and **undo**
- Stable code, that changed very little in the time since it was written





# What didn't work?

- Easy for programmers to forget to use a `DiffUtil`
- Transactions validated in Debug
  - But this doesn't catch everything
- Doesn't support filesystem operations





# Transactions in the 3D Editor





# Changes in the 3D Editor

- Transactions to the Game



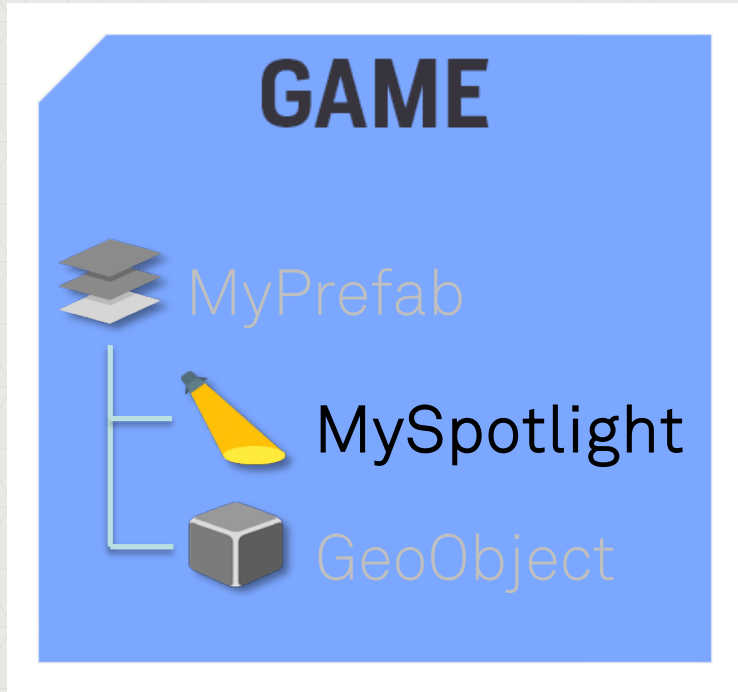


# EditorNodeTree in space conversion





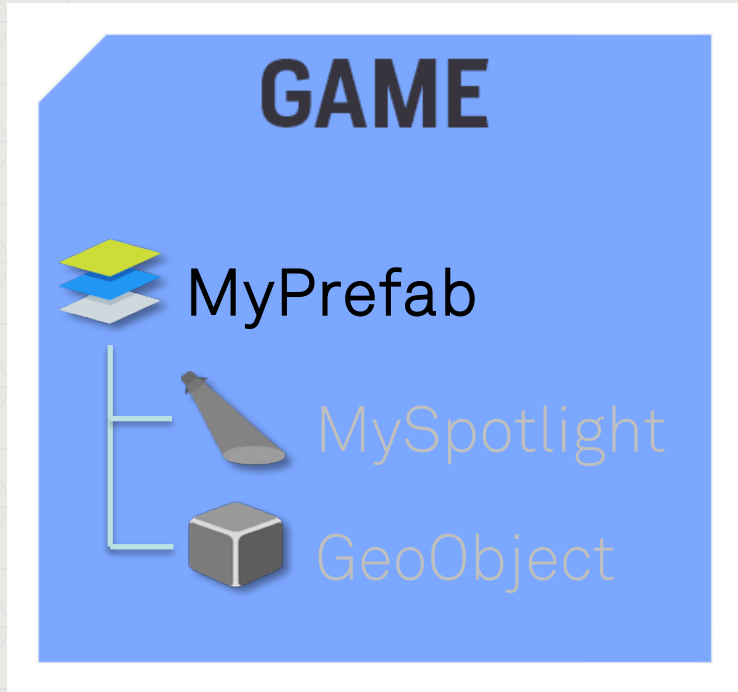
# 3D Editor – Applying a change in the tree



- Modify local space of *MySpotlight*



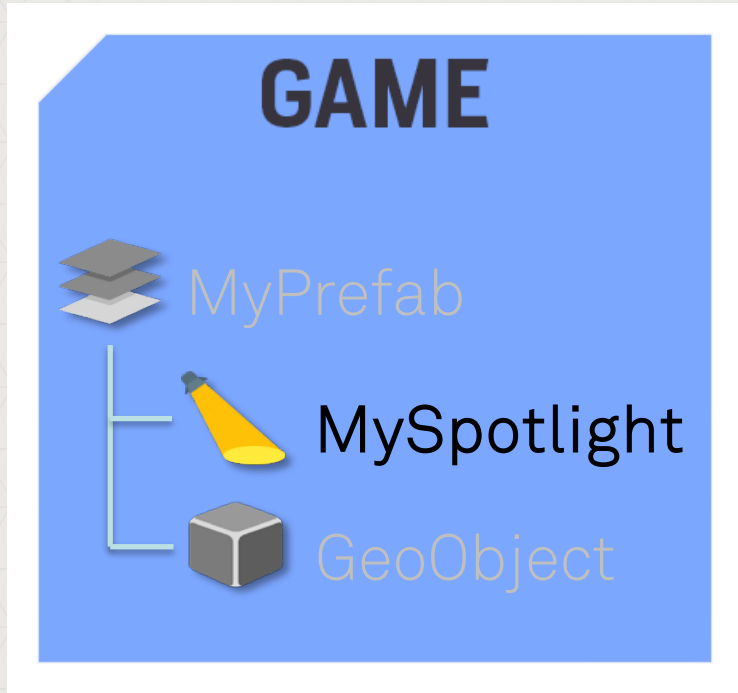
## 3D Editor – Applying a change in the tree



- *MyPrefab* multiplies transform change with it's local matrix
- Modified transaction passed to *MySpotLight*



## 3D Editor – Applying a change in the tree

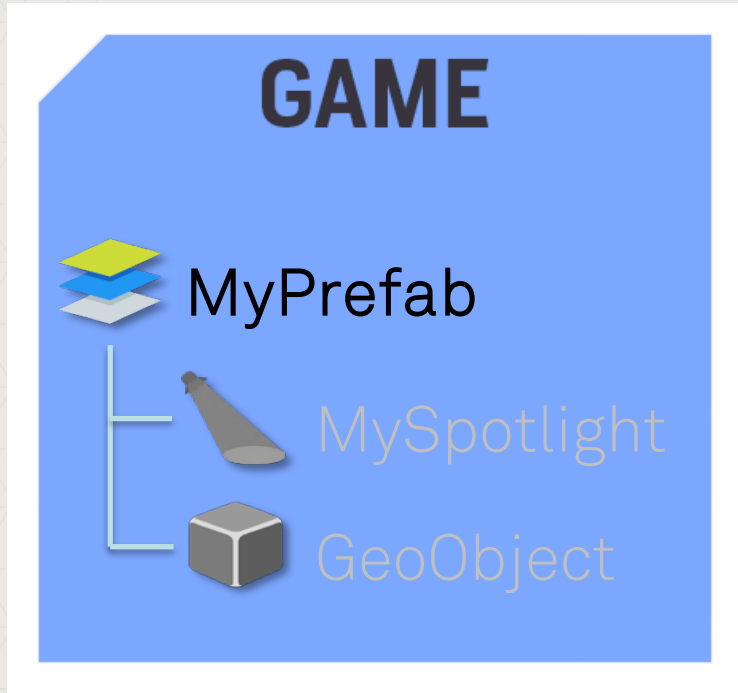


- *MySpotlight* consumes the change
- The modified transaction is in WorldSpace





## 3D Editor – Applying a change in the tree



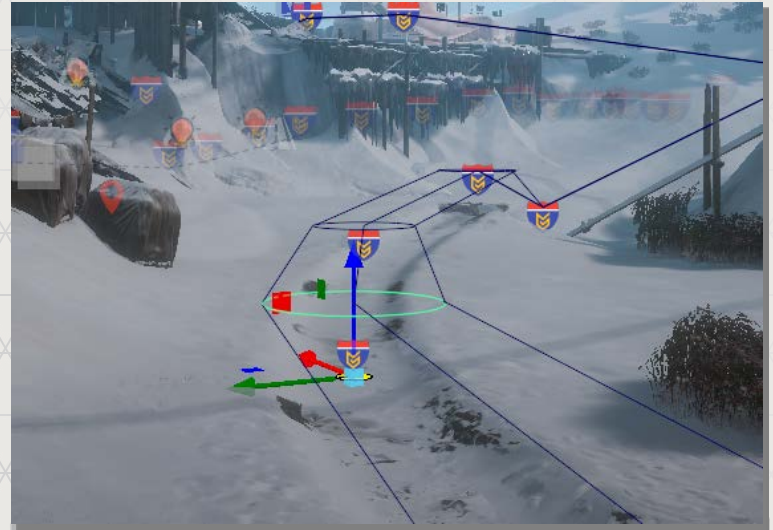
- *[Optional]: MyPrefab* invalidates cached data or updates dependencies





# Interactive changes

- **Problem:** Interactive changes start *in-game*
  - Orientate an object in viewport
  - Painting vegetation
  - Terrain sculpting
  - Etc.





# Naïve implementation of tool change

- (Tool) Transactions from the Game ?





# Naïve implementation of tool change

- (Tool) Transactions from the Game ?







# Naïve implementation of tool change

- Who has authority over data?
- Impossible due to dataloss in conversion
- Creates a cycle going back to game

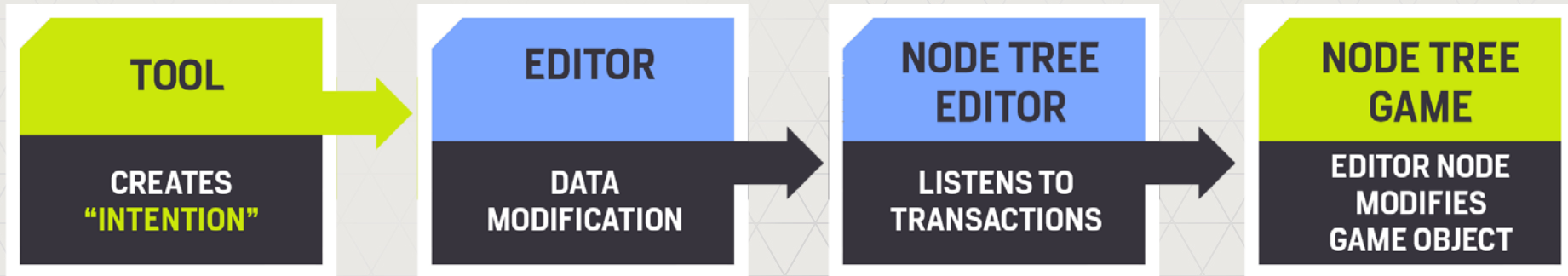




# Tool changes

Solution:

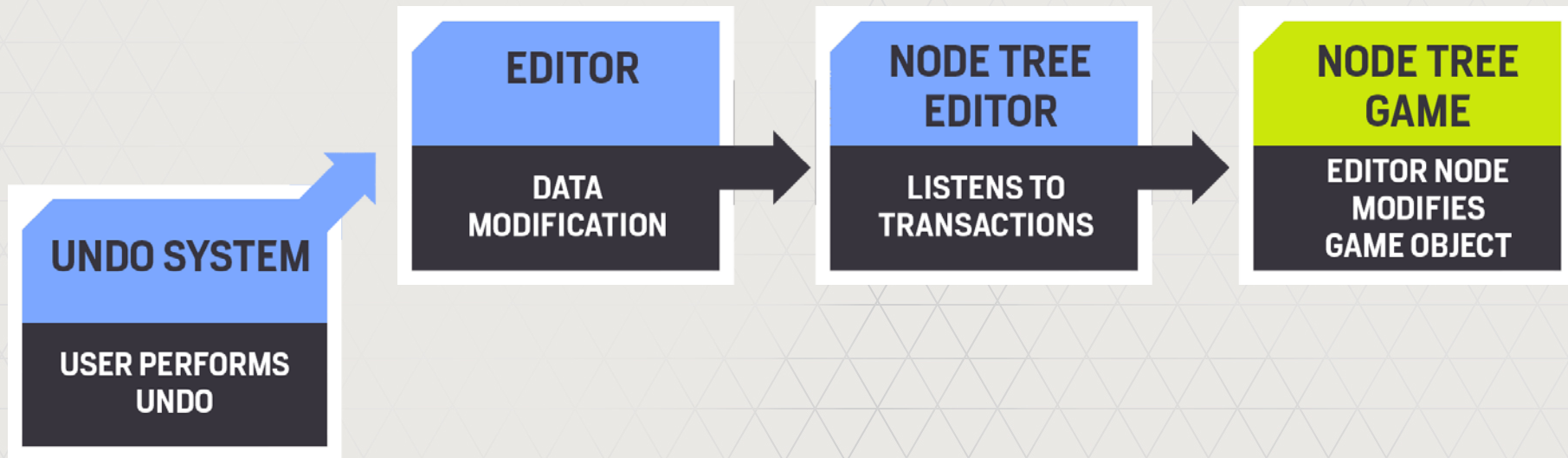
- Tool does not modify data
- Editor keeps authority over data





# Undoing tool changes

Undo works the same way!





# GameSync

Updating structural changes in the game



# GameSync

- *Gamesync* handles structural changes to the game
- Pre-dates Tools Framework
- Creates patch file to apply in game







# GameSync

- Create a copy of *Spotlight*



MyPrefab



Spotlight

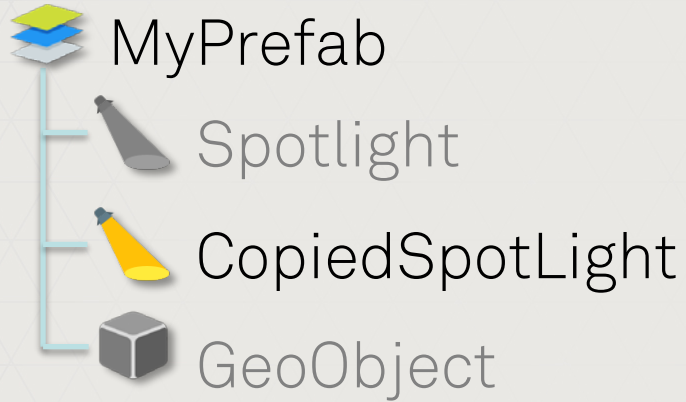


GeoObject





# GameSync – Analyze change





# GameSync – Patch file



MyPrefab (*modified*)

CopiedSpotLight

**ANALYSE  
CHANGE**



**CREATE  
PATCH FILE**



**CONVERT  
PATCH FILE**



**PAUSE GAME  
SYSTEMS**



**APPLY  
PATCH**



**RESUME  
GAME**



# GameSync - Conversion



MyPrefab (*modified*)

CopiedSpotLight



01000  
110100  
010110

ANALYSE  
CHANGE



CREATE  
PATCH FILE



CONVERT  
PATCH FILE



PAUSE GAME  
SYSTEMS



APPLY  
PATCH



RESUME  
GAME





# GameSync



**ANALYSE  
CHANGE**



**CREATE  
PATCH FILE**



**CONVERT  
PATCH FILE**



**PAUSE GAME  
SYSTEMS**



**APPLY  
PATCH**

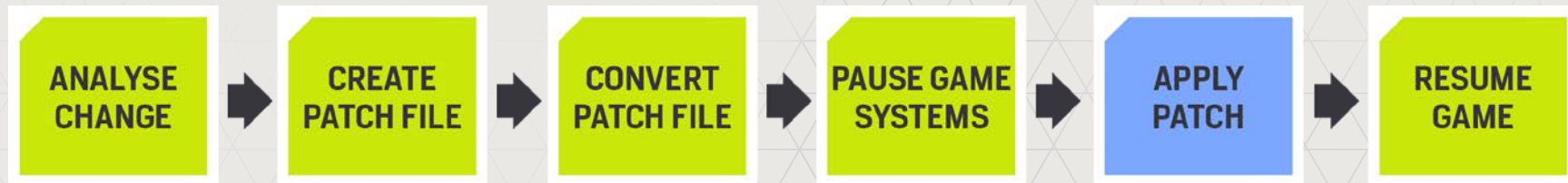
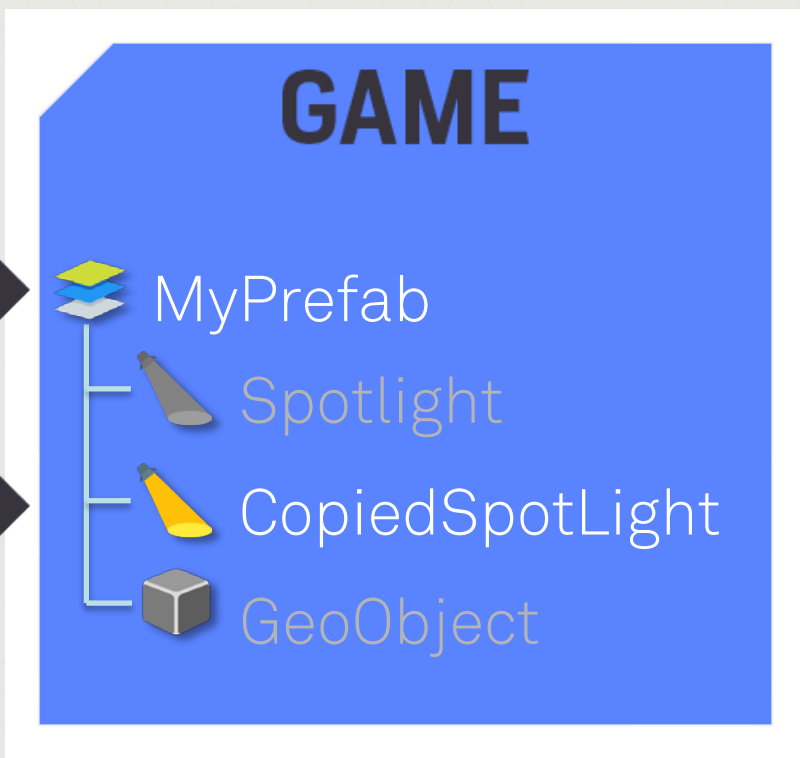


**RESUME  
GAME**



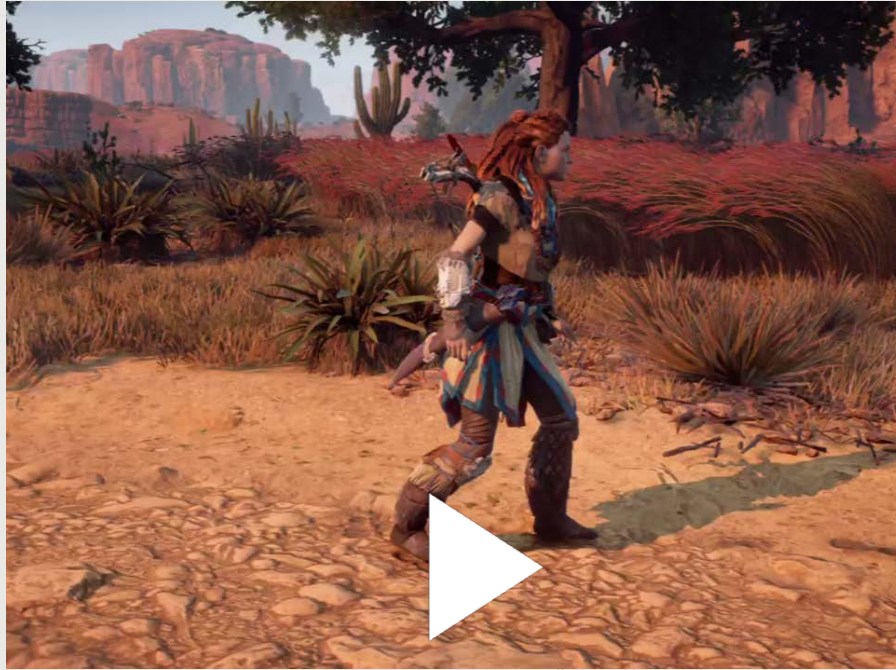


# GameSync





# GameSync



**ANALYSE  
CHANGE**



**CREATE  
PATCH FILE**



**CONVERT  
PATCH FILE**



**PAUSE GAME  
SYSTEMS**



**APPLY  
PATCH**



**RESUME  
GAME**



# GameSync – The Good

- Works on all changes throughout Editor
- Can sync to target as well (PS4)
- No game restart required





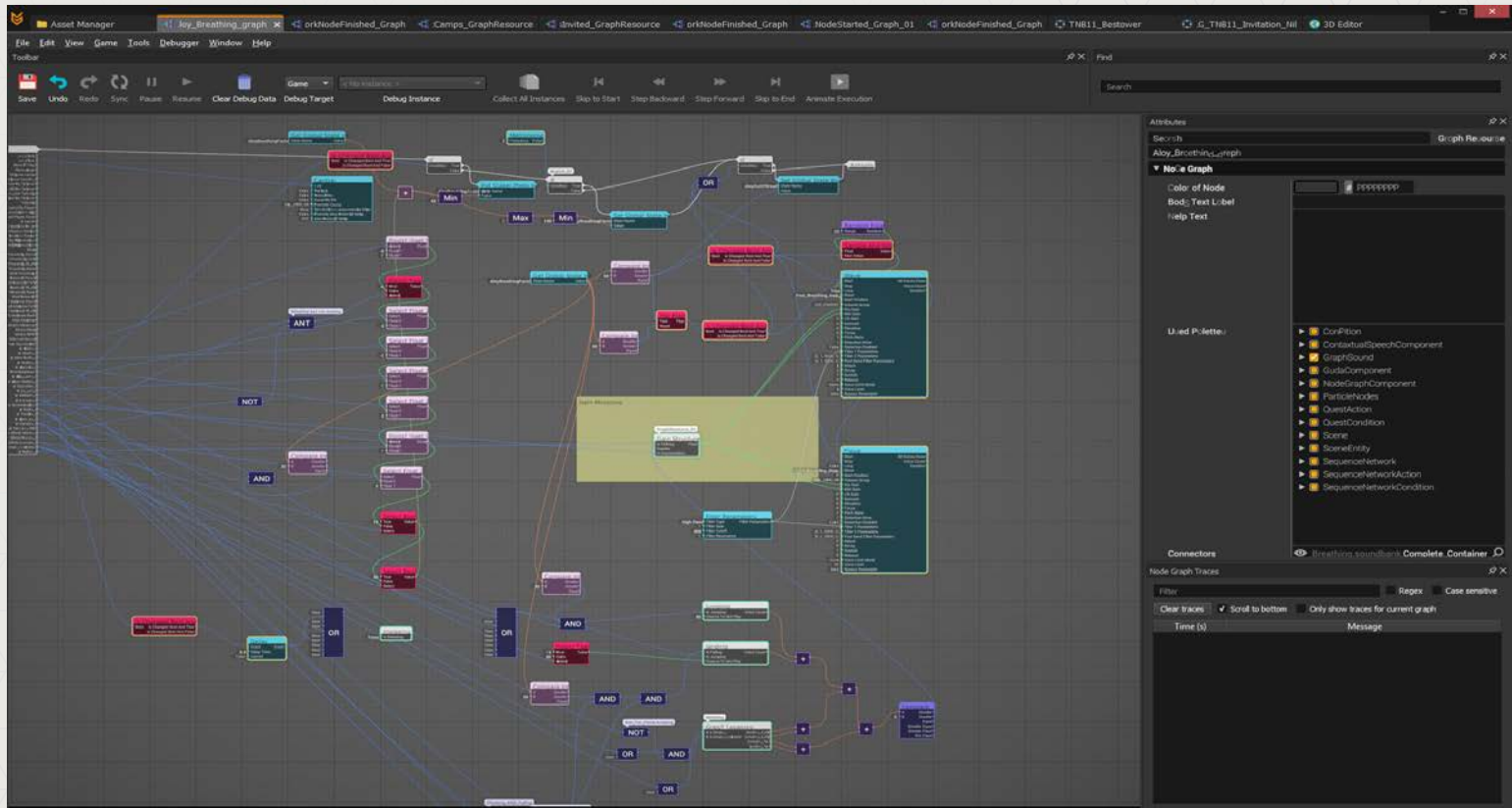
## GameSync - The Bad

- Systems were not designed to work with GameSync
- The change analysis code became very complex
- Performance issues
- Inconsistent behavior
  - Crashes, incomplete updates
- Low level of confidence from users



# Other Tools







# NodeGraph Debugger

- Powerful debugging features
  - Variable inspection
  - Execution visualization
  - Historical debugging
- Minimal involvement from the tools team
  - Almost all of the work was done by game tech







# Interactive Conversations

- A new feature for Horizon
- Free to build a workflow from scratch
- Built specifically for:
  - Writers
  - Quest designers
  - Cinematic artists





# Conversation Editor Context

The screenshot displays the Conversation Editor interface, which is used for creating and editing dialogue sequences. The interface is divided into several panels:

- Asset Manager:** Located at the top left, it shows the current project files, including `X/G_TB111_Final_Nil` and `O/G_TB111C_Nil`.
- Toolbar:** Below the Asset Manager, it contains standard editing tools like Save, Undo, Redo, Sync, Playthrough, and Debug Target.
- Main Editor:** The central area displays a flowchart of the conversation sequence. It starts with `StrangerGreet`, which branches into `Stranger_FightChoice` and `Friend_FightChoice`. These lead to various dialogue options like `Stranger_What`, `Stranger_Yes`, `Stranger_No`, `Friend_What`, and `Friend_Yes`. The flowchart also includes `Stranger_Goodbye` and `Friend_Yes` paths.
- Event Palette:** On the left, it lists various events and actions that can be added to the sequence, such as `Rumble`, `Screen Effect`, `Slow Motion`, `Weather`, `Timeline Events - Misc`, `Force Smart Object Area to Hig...`, `Hide All Players`, `Hide HUD`, `Interrupt`, `Model Lod Bias`, `Override Eye Color`, `Remove Entities`, `Set Fact`, `Set Heat Fact`, `Set Global Lod Bias`, `Show HUD Overlay`, `Start And Stop Scene`, `Start Scene`, `Stop Scene`, and `Nil's Create Entity Action`.
- Sequences:** A timeline view at the bottom left shows the sequence of events over time, with markers for `Nil` and `Aloy` actions.
- Screenplay:** On the right, it displays the dialogue text in a screenplay format. The current sequence is `StrangerGreet`, which includes the following dialogue:
  - Nil:** (Nil is a stranger) Aloy speaks to Nil at the waterfalls. I'm so glad you came. In a way, I feel like I already know you. Every pile of corpses was like reading your journal.
  - Aloy:** You don't know me. I've done more than kill bandits.
  - Nil:** I don't doubt it. But bandits were my quarry, and their camps my hunting grounds. Now they're bare.
  - Nil:** So I had to know if you were just faster than me, or more skilled too.







# Conversation Generator

- This conversation was auto-generated
- Camera cuts, animation, gestures selected by rule-based system
- **RoboVoice** and lip-sync animations created for mock-up content



More about you...

Farewell ↩



# Conversation Generator

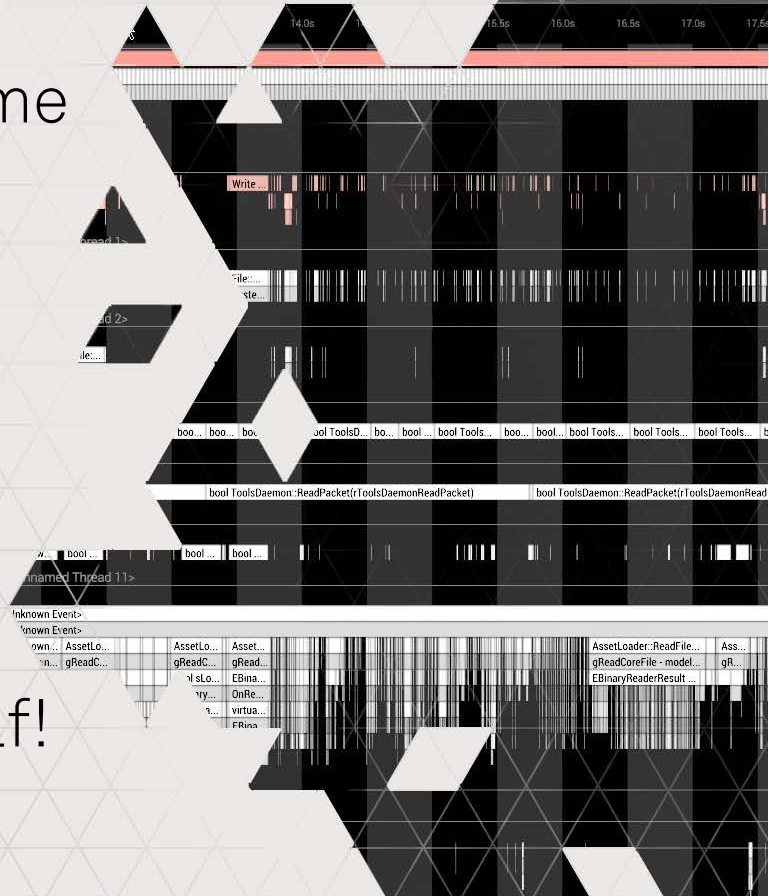
- Inspired by the work of CD Projekt RED and BioWare
- Was possible because it was easy to build in the new framework
- Behind the Scenes of Cinematic Dialogues in 'The Witcher 3: Wild Hunt'
  - <http://www.gdcvault.com/play/1023285/>







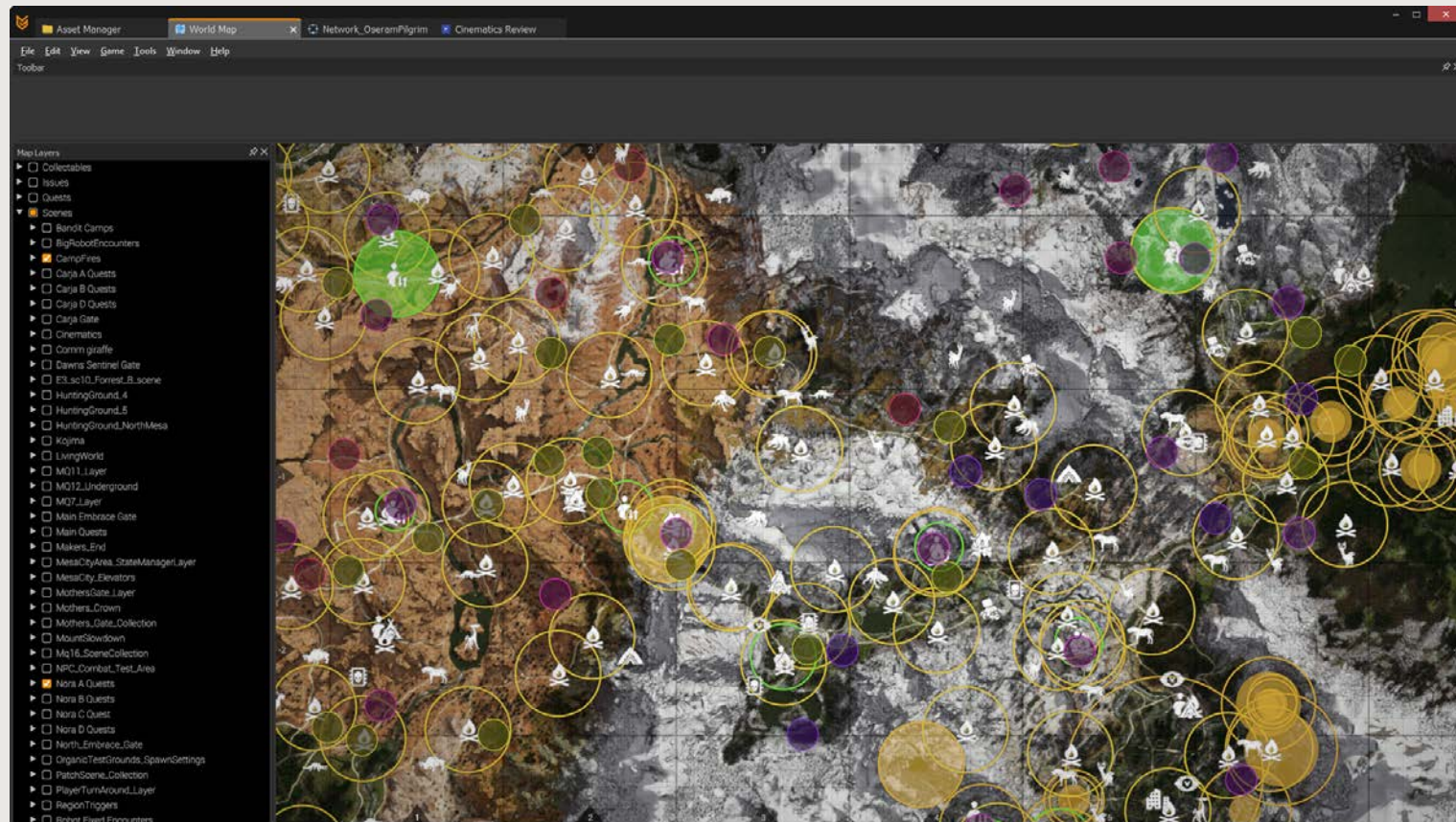
- Built on top of an existing in-game profiler framework
- Real-time data from any app running the profiler framework
- Editor runs the profiler framework, so it can profile itself!







# World Map





# World Map

- Visualizes game features on a 2D world map
- Pluggable system for overlays
  - Quests
  - JIRA bugs
- Came online late in the project, and wasn't heavily used









# World Map – Perf Report Overlay

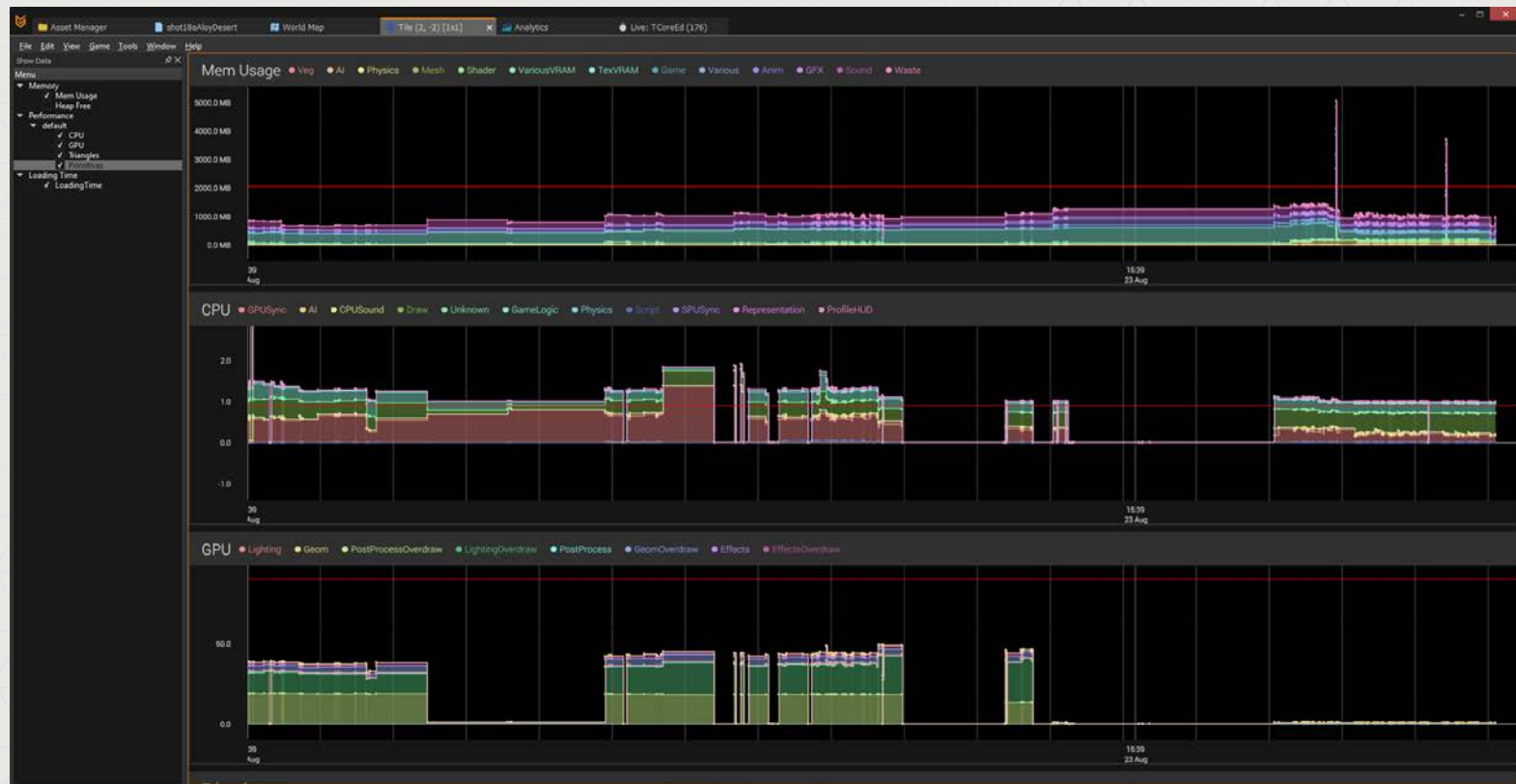
The screenshot displays a game engine interface with a world map. The top menu bar includes 'Asset Manager', 'shot13RaMyDesert', 'World Map', 'Analytics', and 'Live: TCore6d (176)'. The left sidebar shows a 'Map Layers' panel with a tree structure: 'Autoblot' (expanded), 'Memory' (expanded), '1x1' (expanded), 'Mem Usage' (expanded), and '3x3' (expanded). Below this are checkboxes for 'Collectables', 'Items', 'Quests', 'Scenes', 'World Activities', and 'World Encounters'. The main map area shows a green, textured terrain with a grid overlay. A performance report overlay is visible in the bottom right corner of the map area, displaying a table of metrics for a selected tile (X:2, Y:2).

Tile (X:2, Y:2)	
Veg:	0.01 MB
AI:	1.20 MB
Physics:	85.72 MB
Mesh:	28.70 MB
Shader:	7.34 MB
VariousVRAM:	62.44 MB
Total:	967.24 MB
TexVRAM:	257.08 MB
Game:	12.43 MB
Various:	34.72 MB
Anim:	194.77 MB
GFX:	54.51 MB
Sound:	186.98 MB
Waste:	43.90 MB

The overlay also includes a small thumbnail image of the game scene corresponding to the selected tile.



# Perf Report







# Decima API

- C-API to expose Tools Framework functionality
  - File loading/saving
  - Object manipulation
- Python wrapper
  - Artists and designers can create scripts manipulate game content





# Decima API

```
# Get the spotlight to edit...
spotlight = GetSpotlightToEdit()

# Open a transaction, to start modifying objects
with DecimaAPI.Transaction() as transaction:

    # Update the cone angle
    spotlight.ConeAngle = 30.0

    # Indicate that we have finished making changes
    transaction.Commit()
```

```
transaction.Commit()
```



# Future goals

- Move other teams into new editor
  - (VFX, Environment Art, ..)
- Build new Editor Contexts
  - Entity editor
  - Particle editor
  - Shader/Material editor





# Future goals

- Rewrite GameSync system
- Take the 3D editor out of process
- Improve UI/UX of the editor





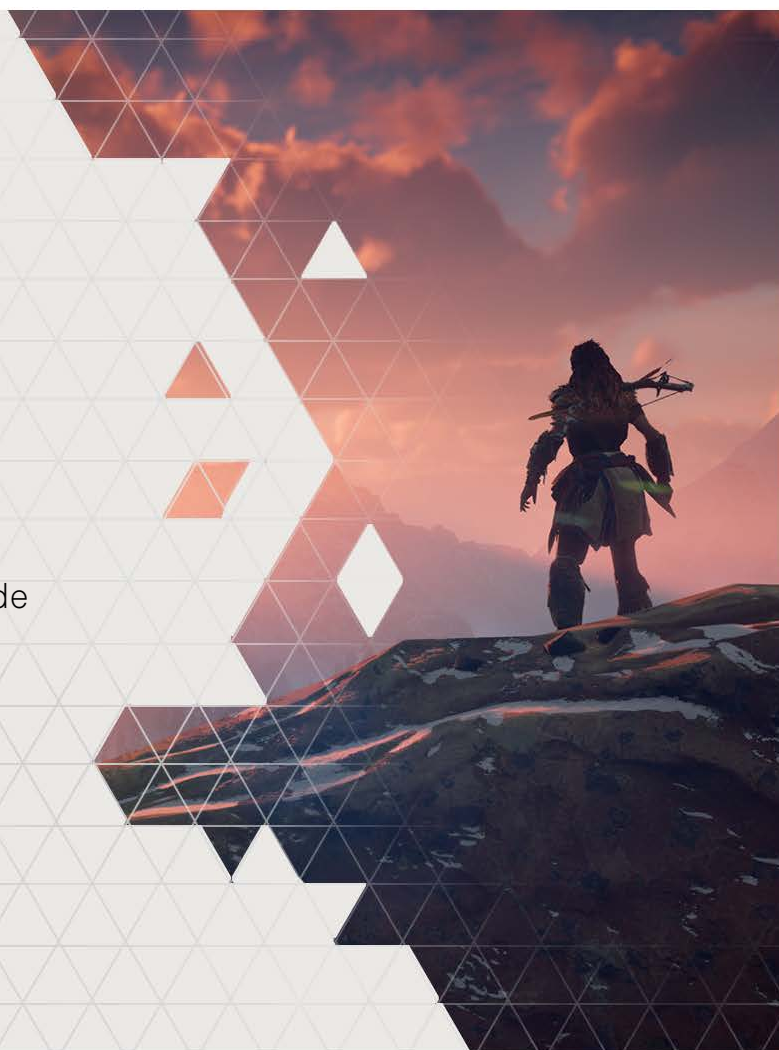
# Thanks

- Everyone at Guerrilla

- Special Thanks

- Michiel van der Leeuw
- Jelle van der Beek
- Ritesh Oedayrajsingh Varma
- Guido de Haan
- Laurens Simonis
- Kenzo ter Elst
- Johanna Persson

- Olaf Jansen
- Bryan Keiren
- Emile van der Heide
- Remco Straatman
- Stefan Reek
- Nathan Vos







# We're Hiring!

<https://www.guerrilla-games.com/join>





# Contact

- [dan.sumaili@guerrilla-games.com](mailto:dan.sumaili@guerrilla-games.com)
- [sander.vandersteen@guerrilla-games.com](mailto:sander.vandersteen@guerrilla-games.com)





DECIMA

Questions?