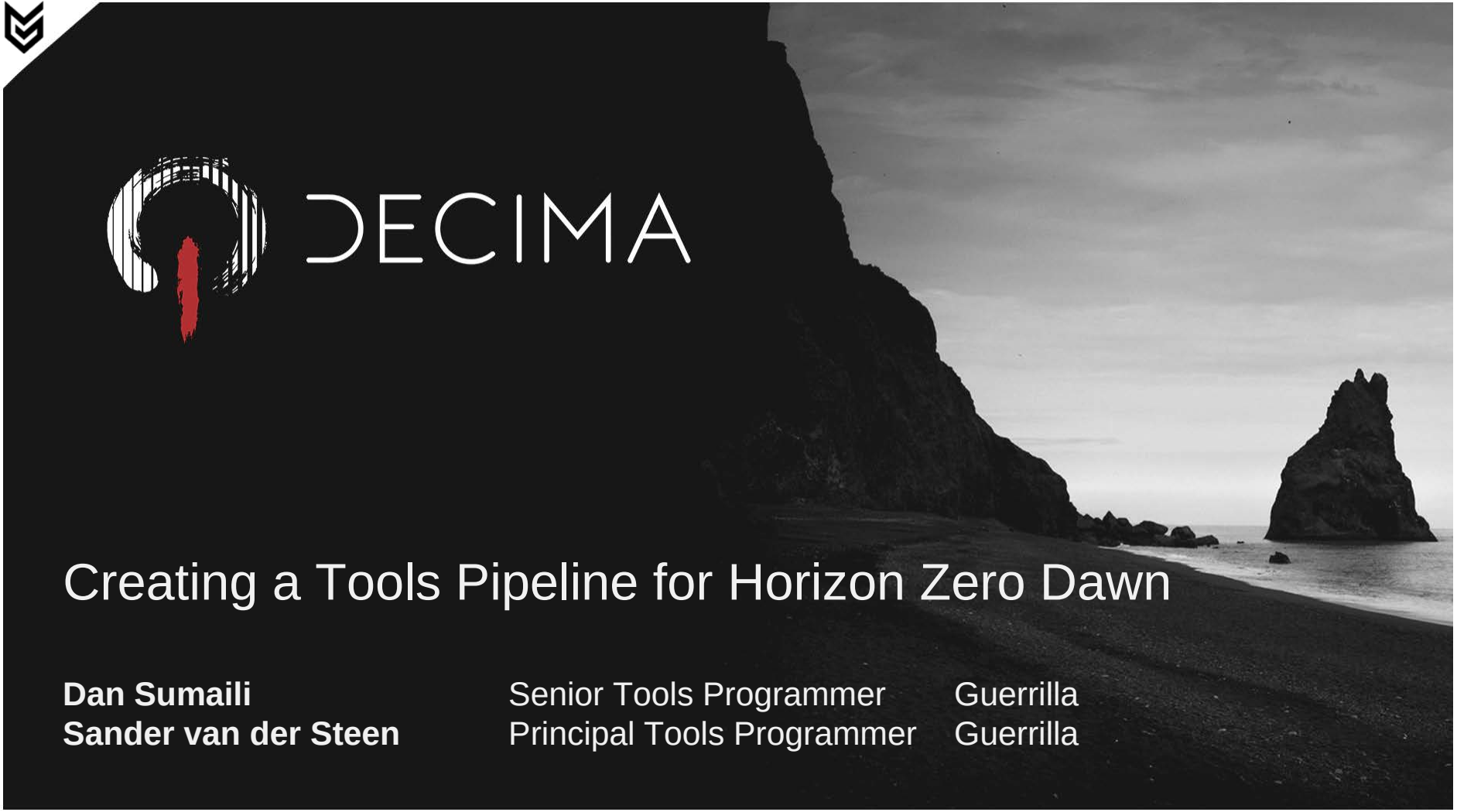
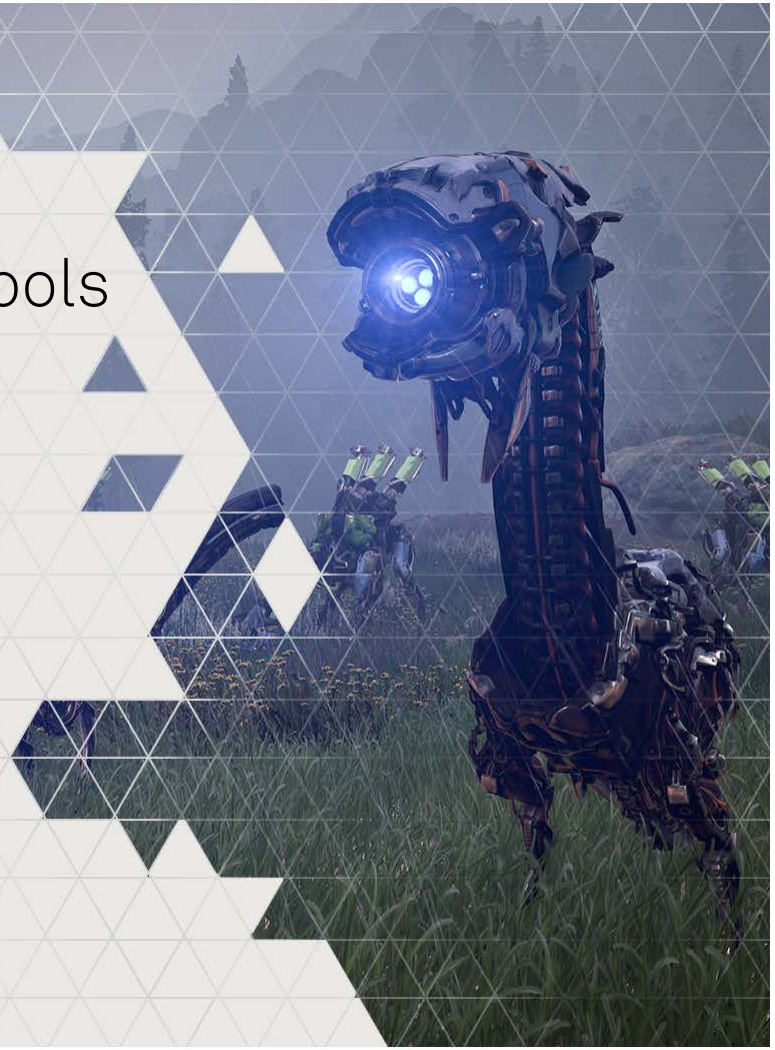


Creating a Tools Pipeline for Horizon Zero Dawn

Dan Sumaili Senior Tools Programmer Guerrilla
Sander van der Steen Principal Tools Programmer Guerrilla

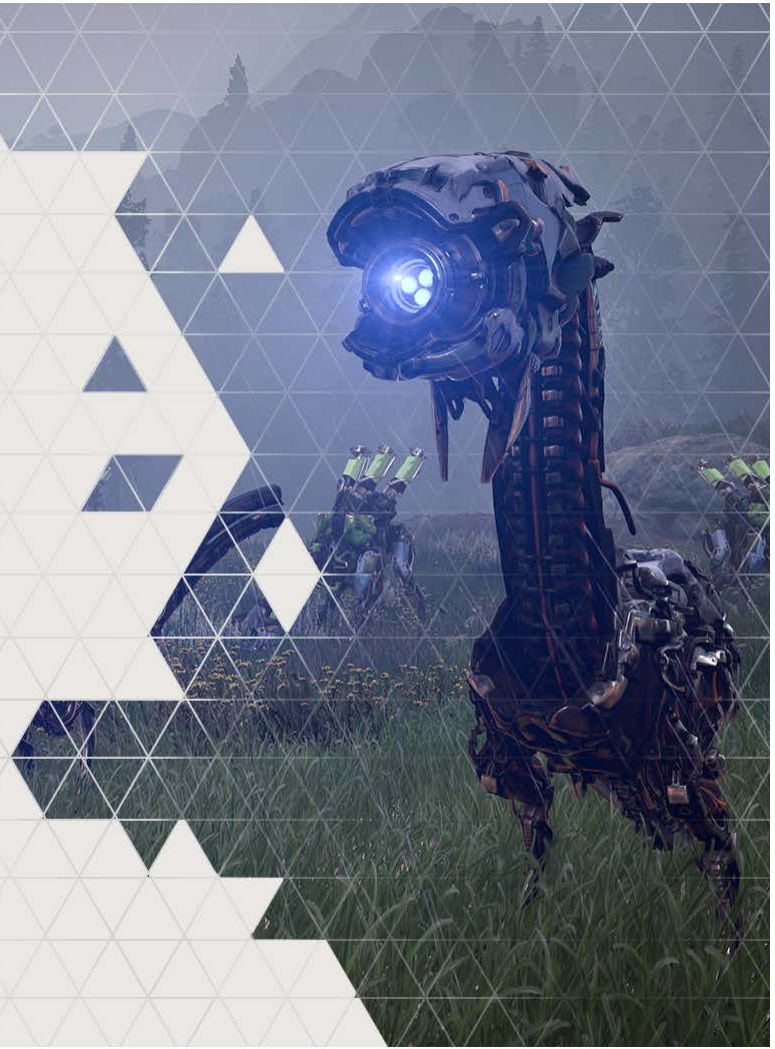
Takeaways

- 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

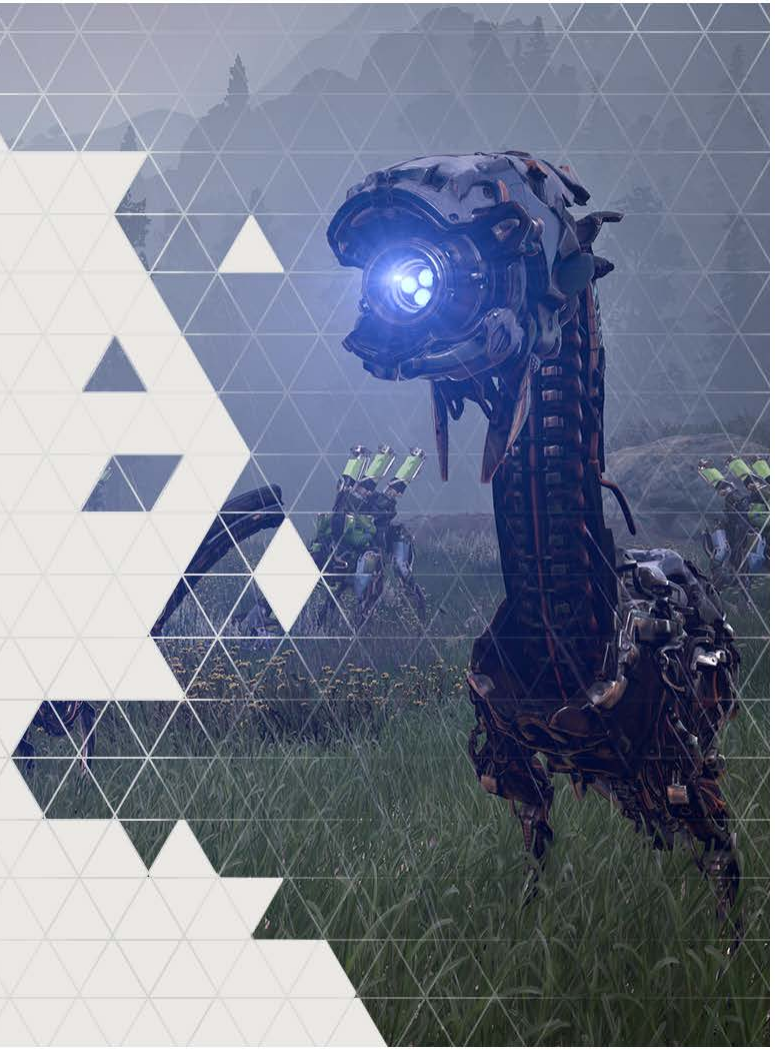


In the beginning... (Nov 2013)

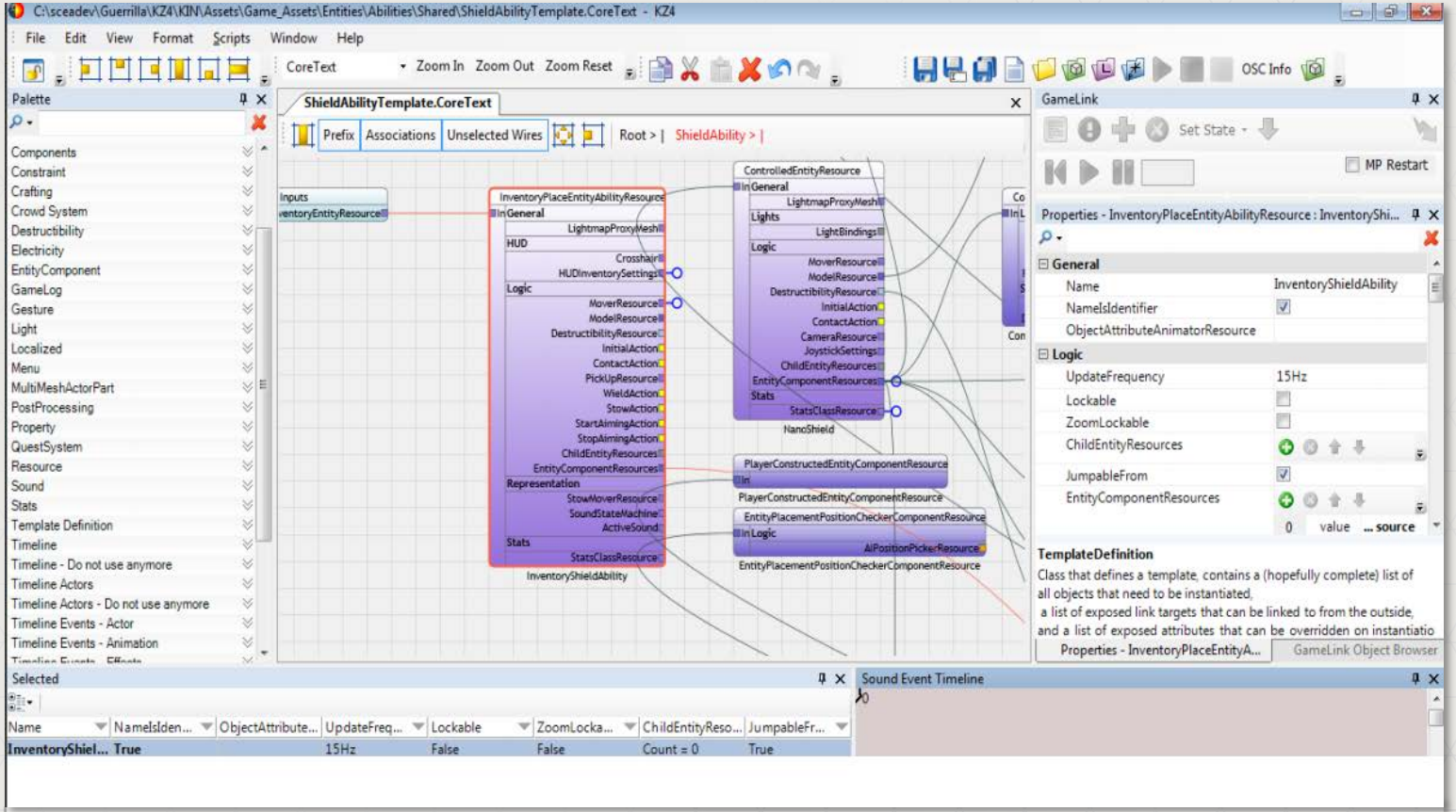
- We had fragmented tools
  - Mix of languages & frameworks
- Long iteration times (20-30 mins.)
- Dissatisfied Users
  - Far behind our competition

The CoreText Format

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

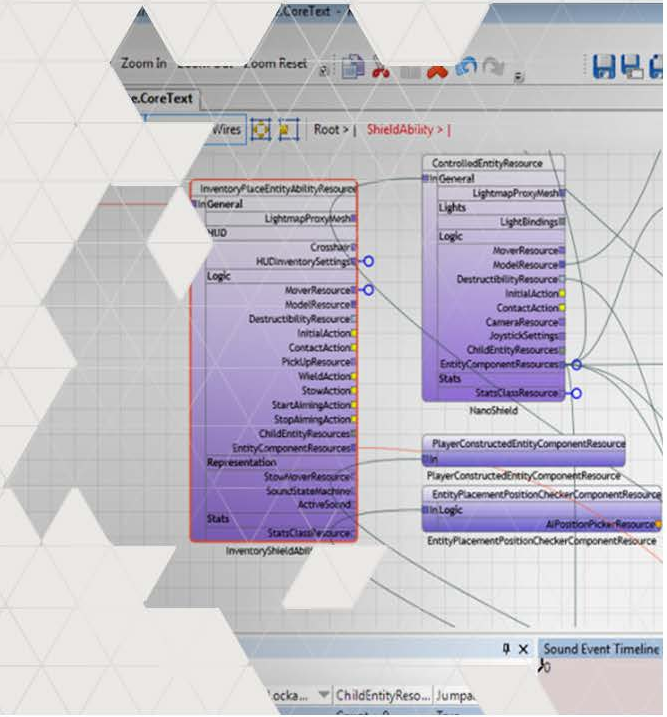


CoreText Editor

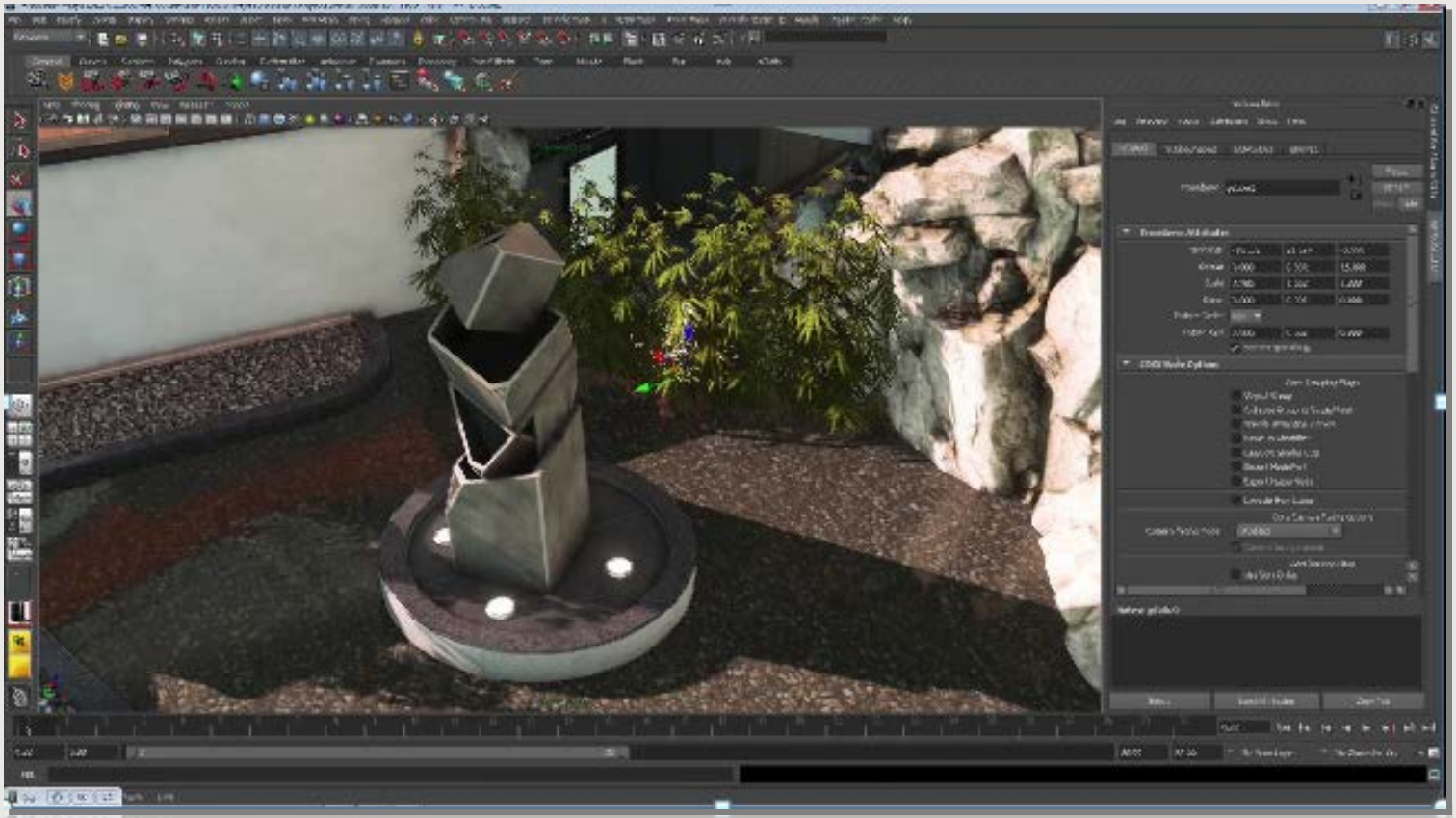


CoreText Editor

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



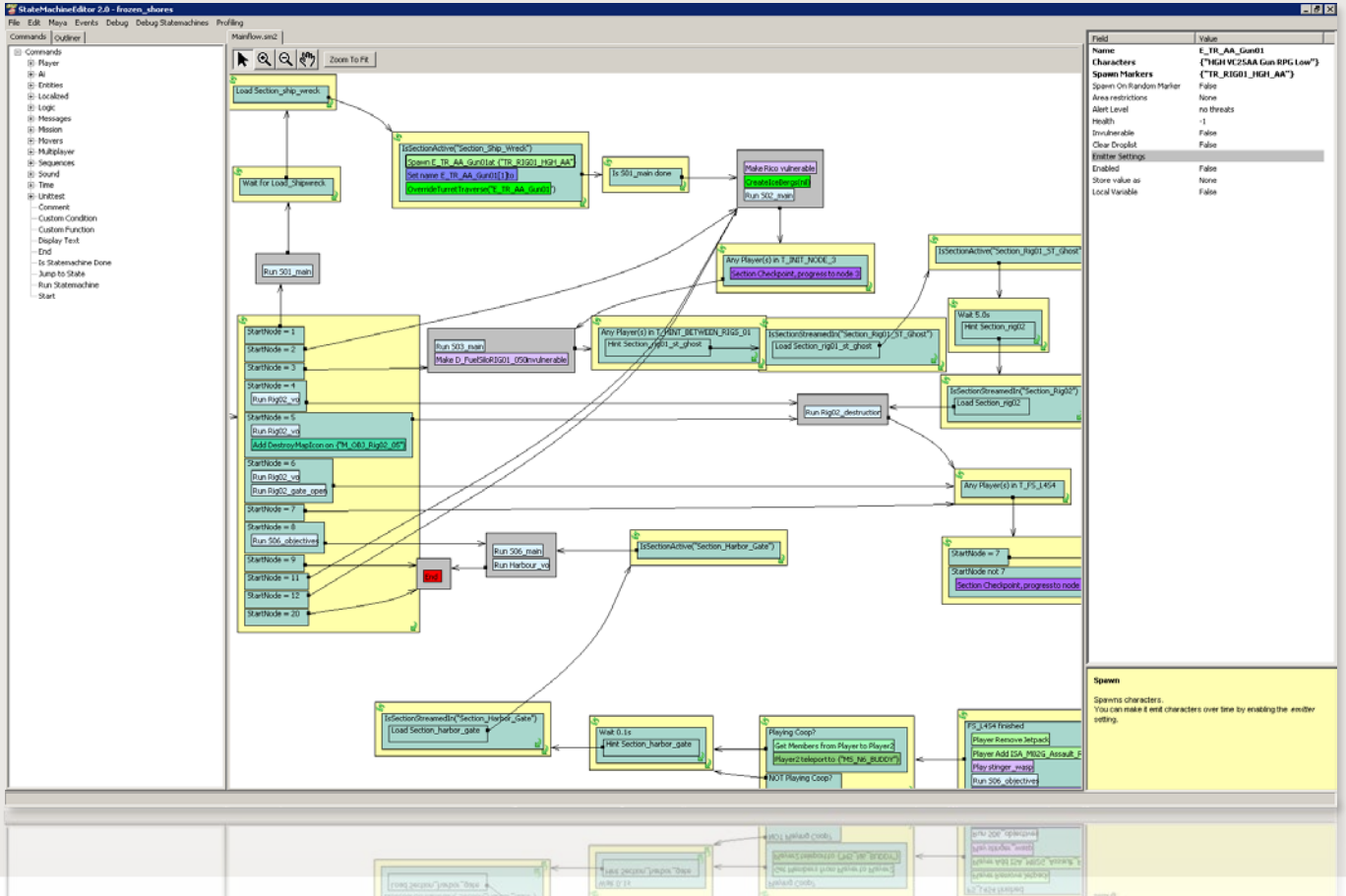
Maya Plugin



Maya Plugin

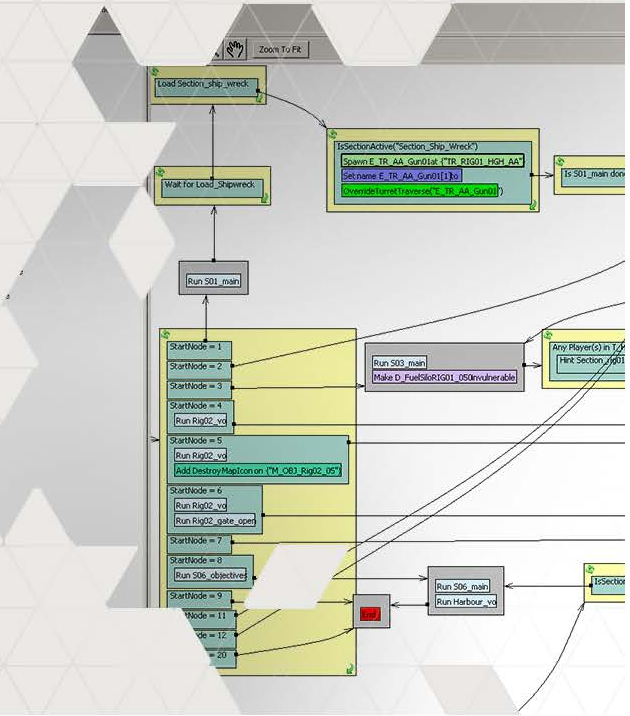
- Exports Maya data to .CoreText
- Used Decima's Renderer
- C++, Python, Maya API
- **Killzone Shadow Fall** Art Tools
  - http://www.gdcvault.com/play/1020711

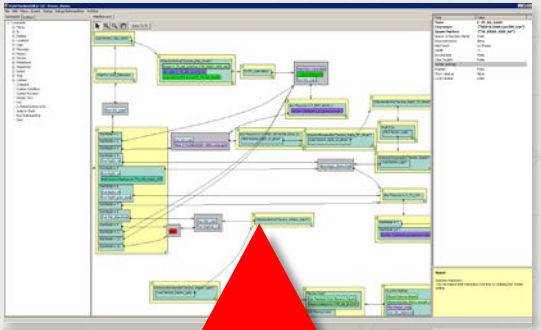
State Machine Editor – ‘SMED’

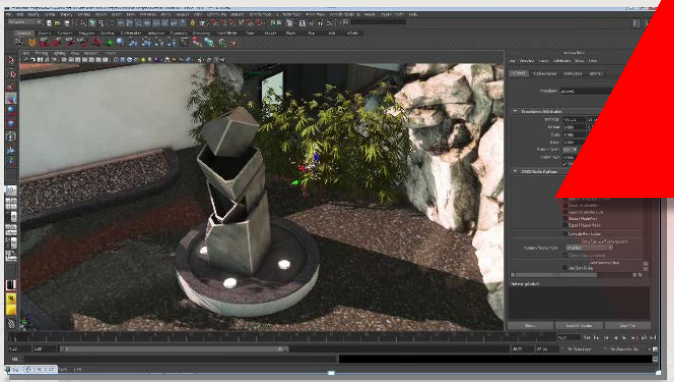


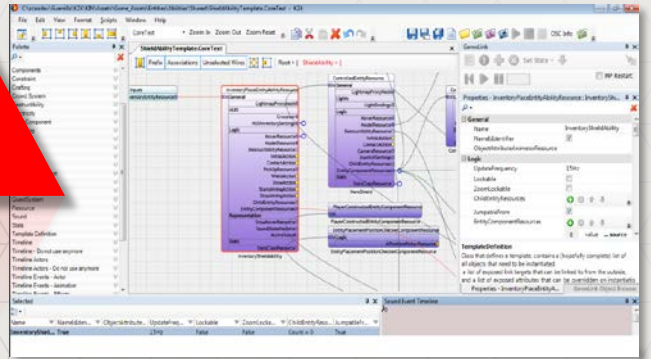
State Machine Editor – ‘SMED’

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

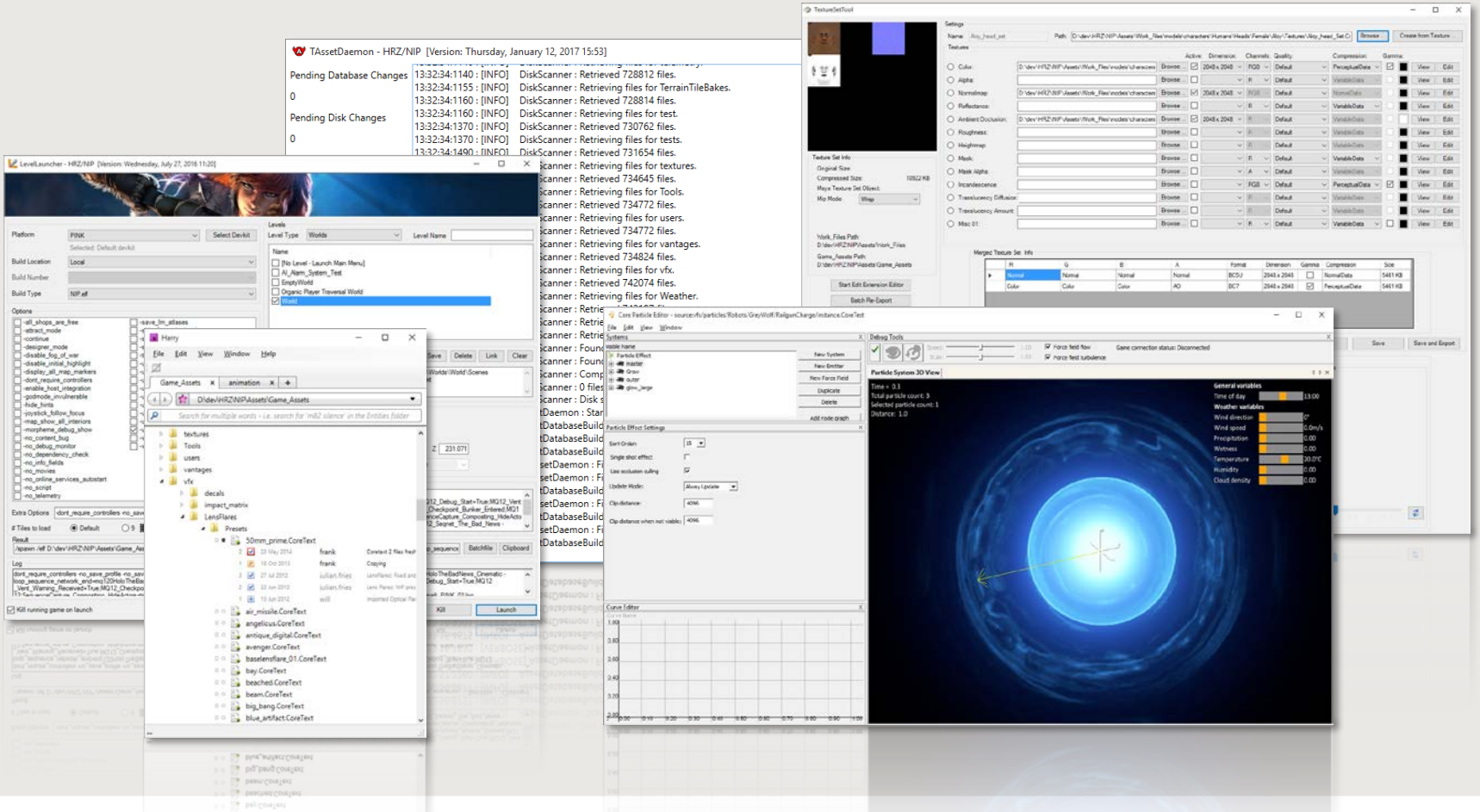








And many more...

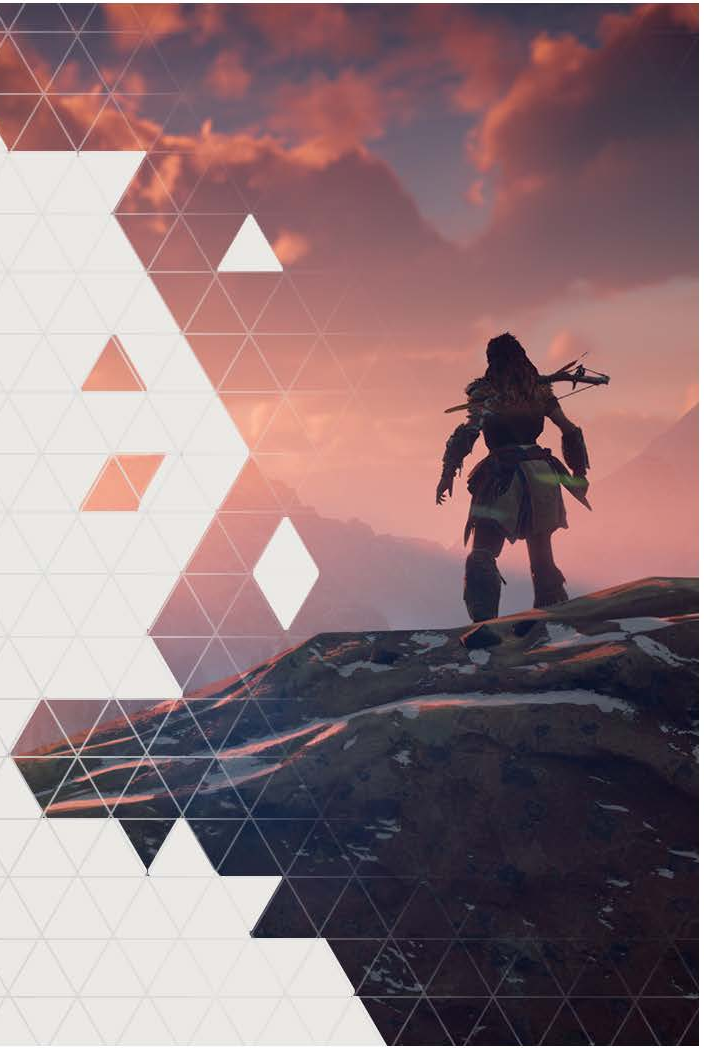


Debug UI

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

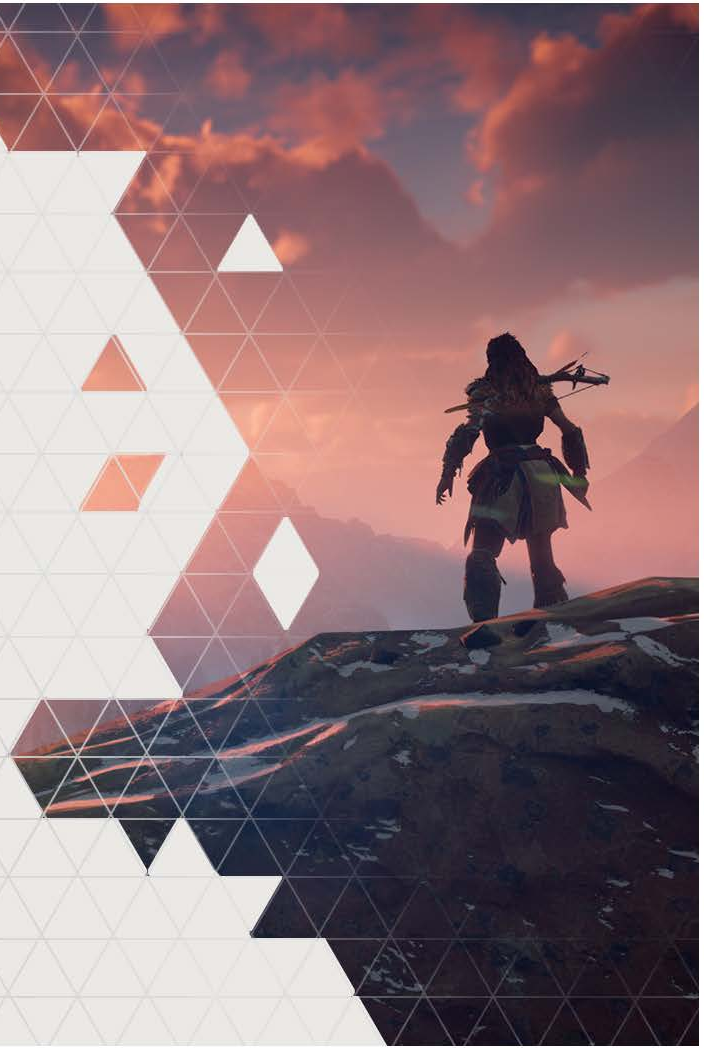
Horizon Zero Dawn

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



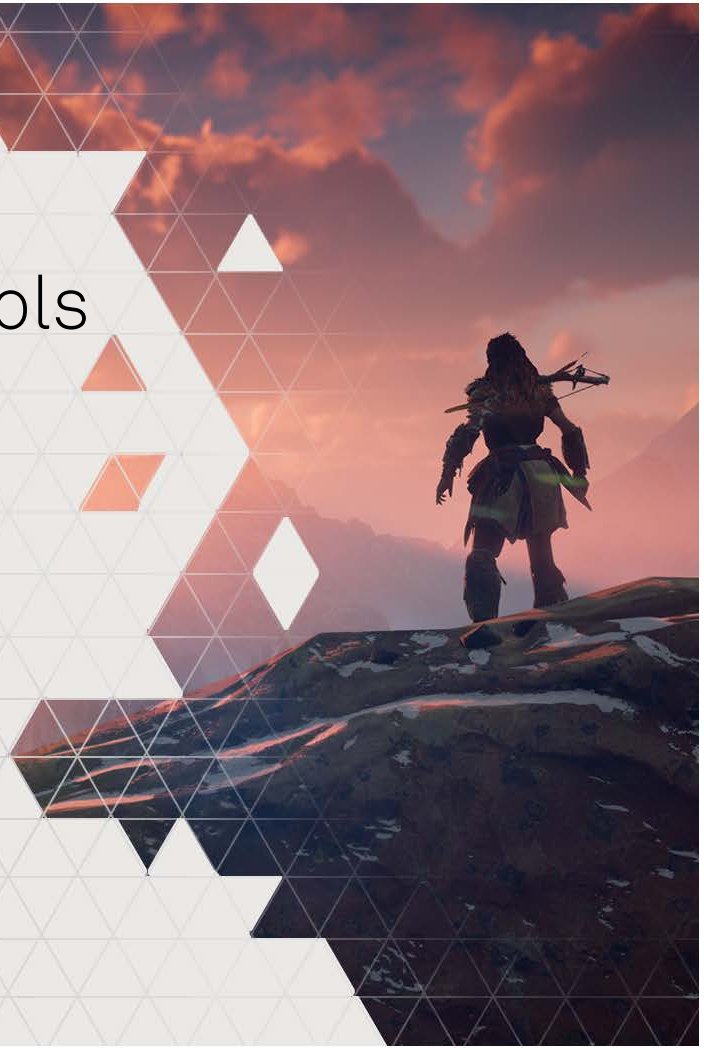
Horizon Zero Dawn

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



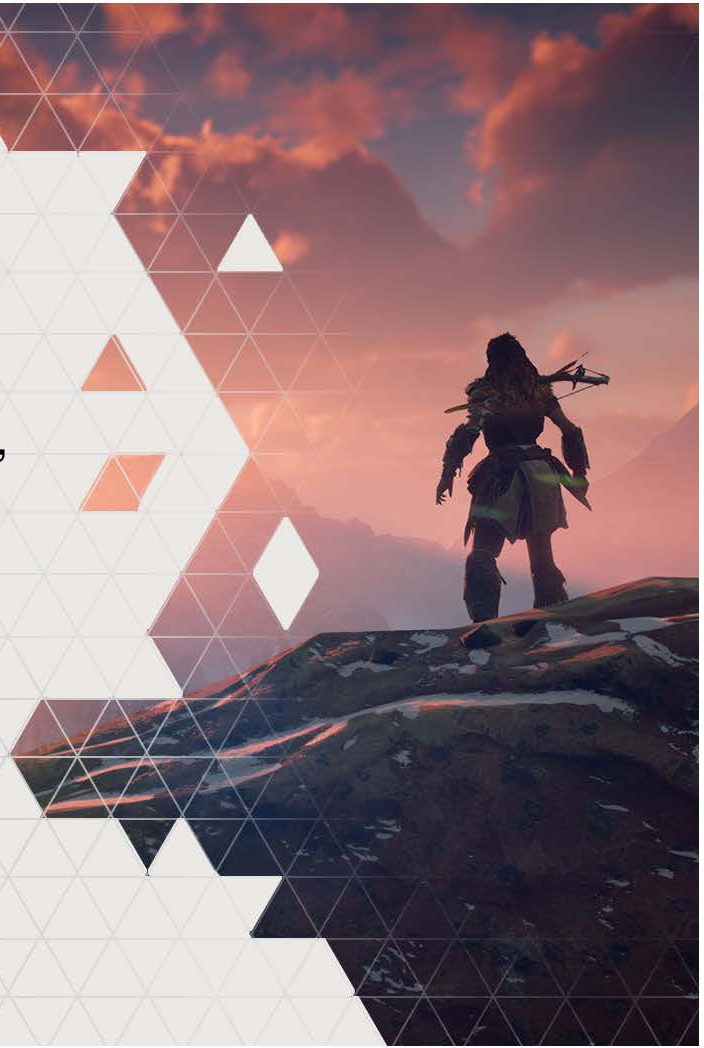
Tools Goals: Code

- 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



DECIMA

TOOLS FRAMEWORK

GAME CODE

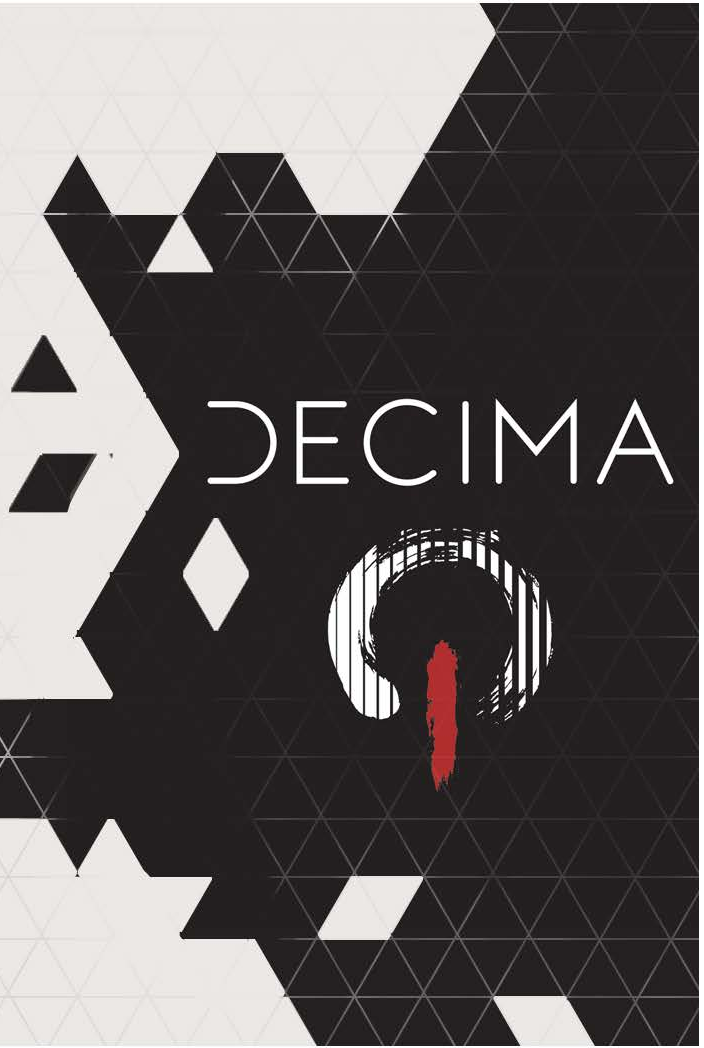
CORE

PIGS

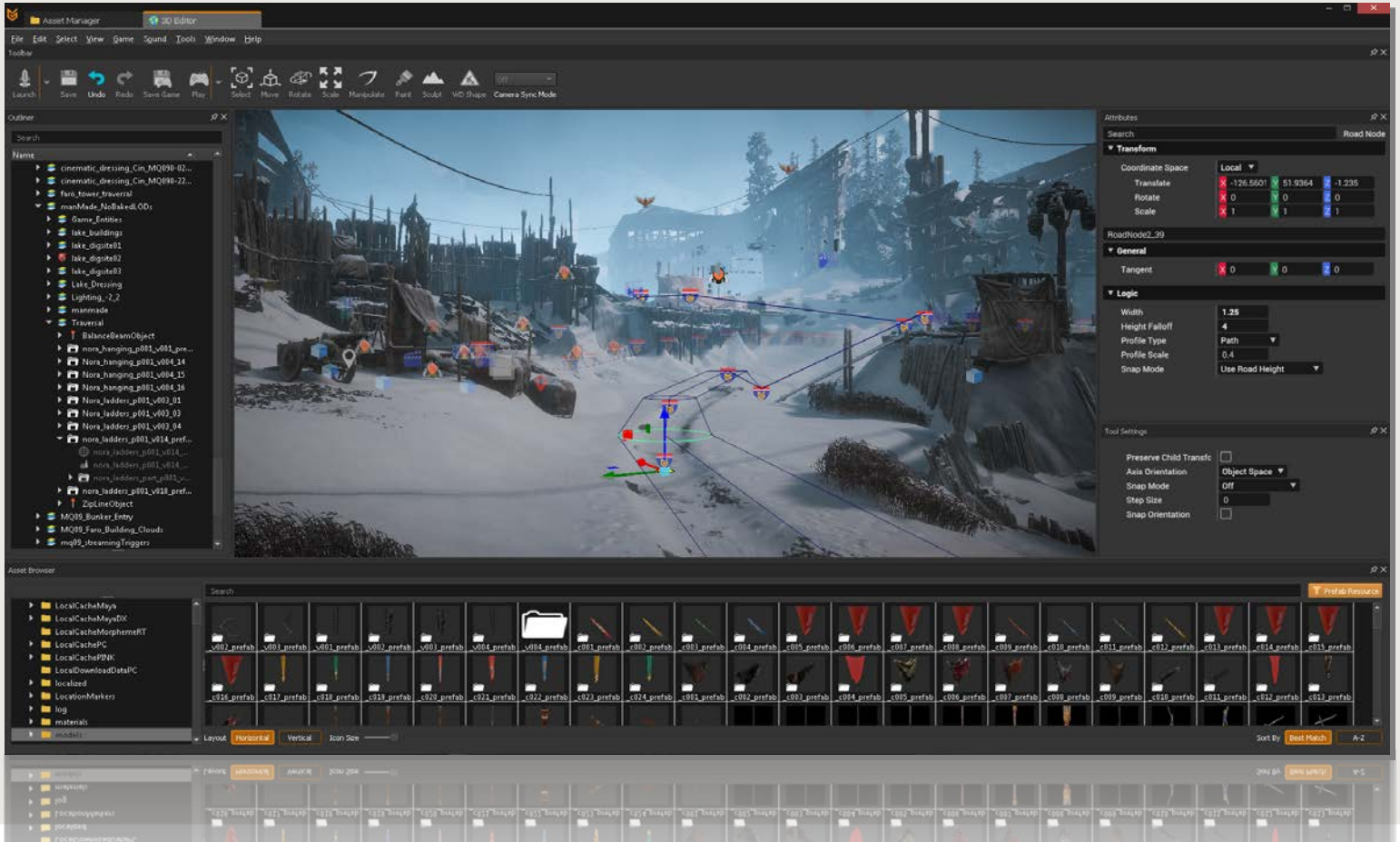
OS

Decima Engine

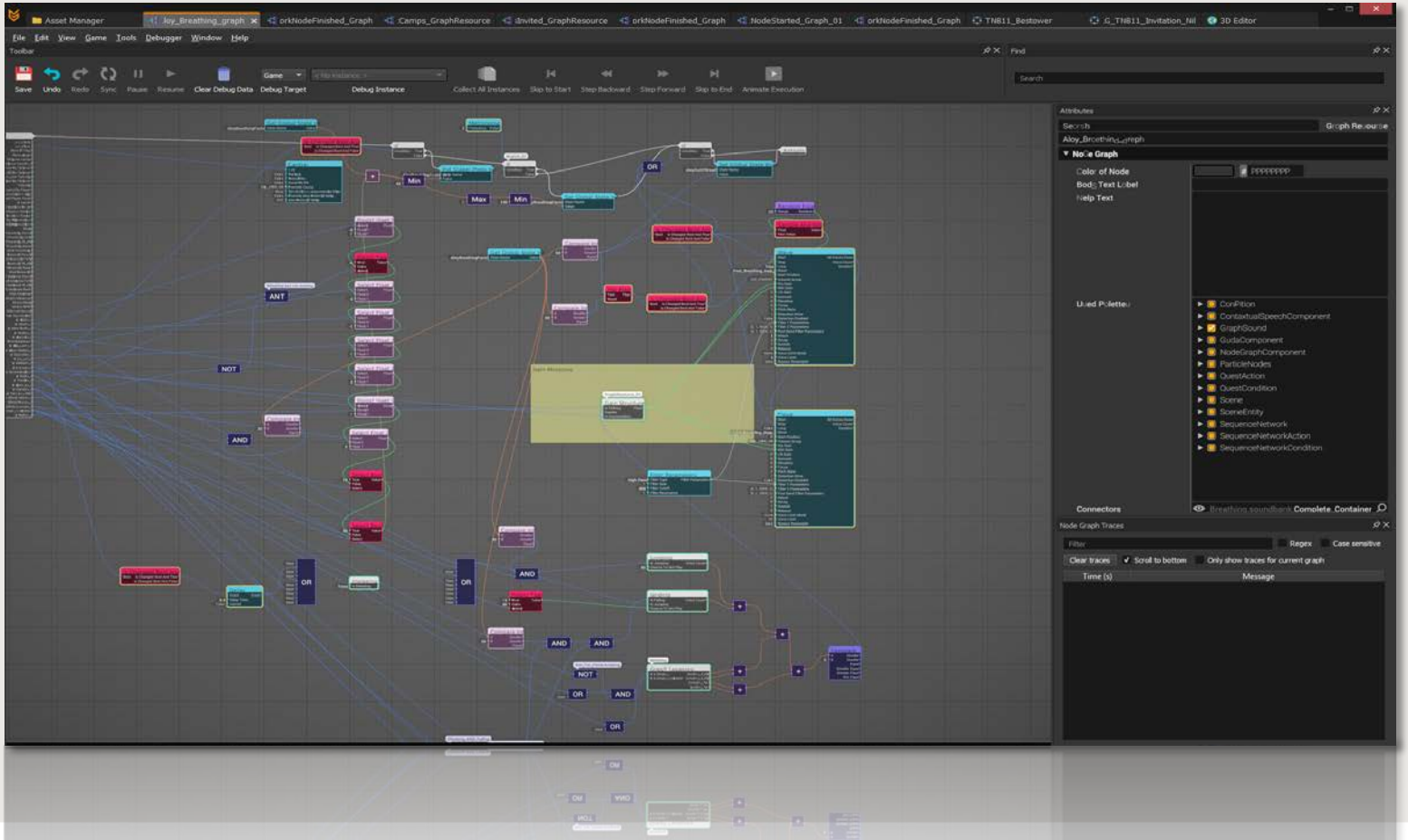
- Data Serialization & Deserialization
- Runtime Type Introspection (RTTI)
- File I/O, Networking, Threads, Jobs
- Image Readers & Writers
- Compression, Hashing, Algorithms
- Unit Testing Framework



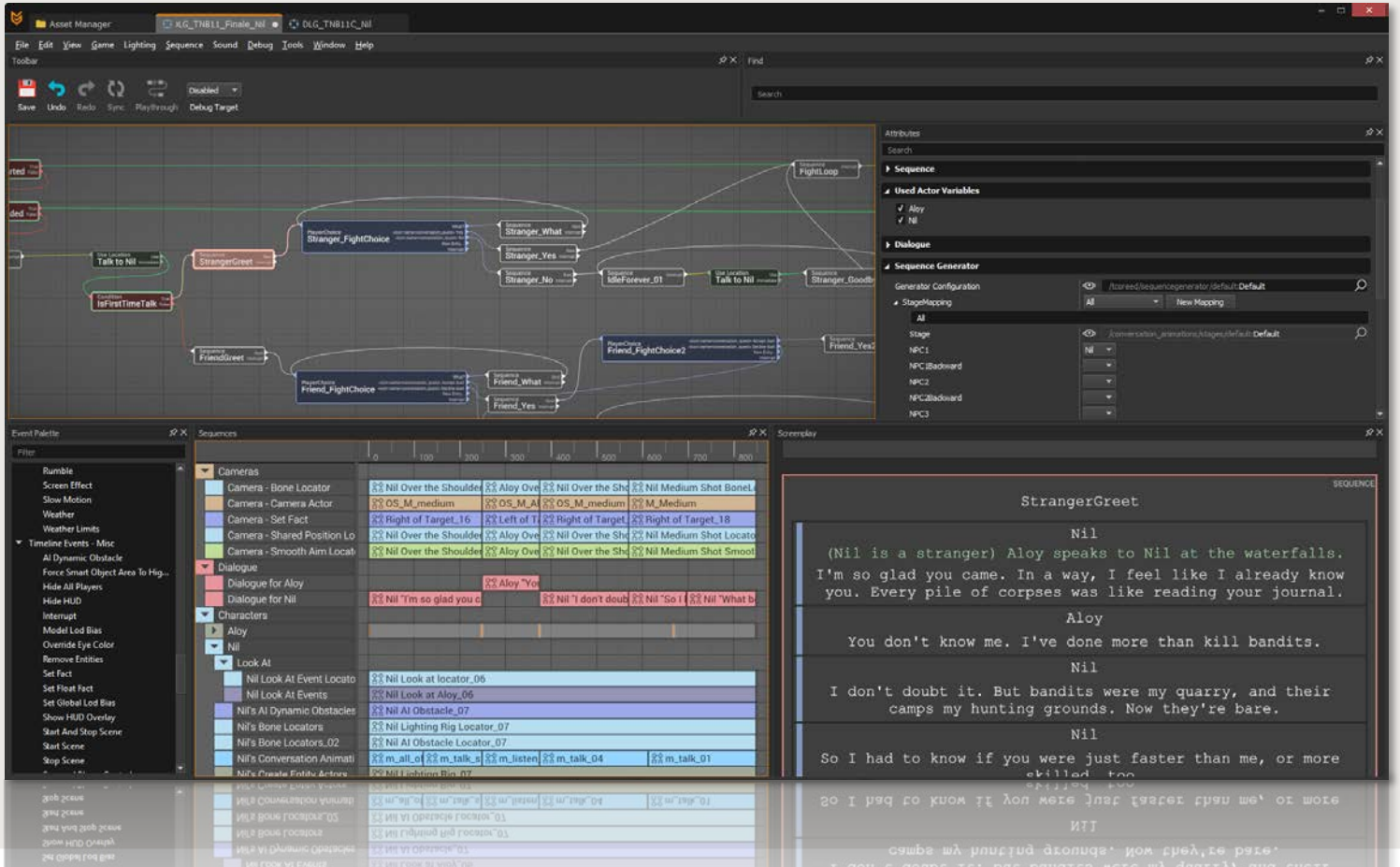
Level Editor



Visual Script Editor



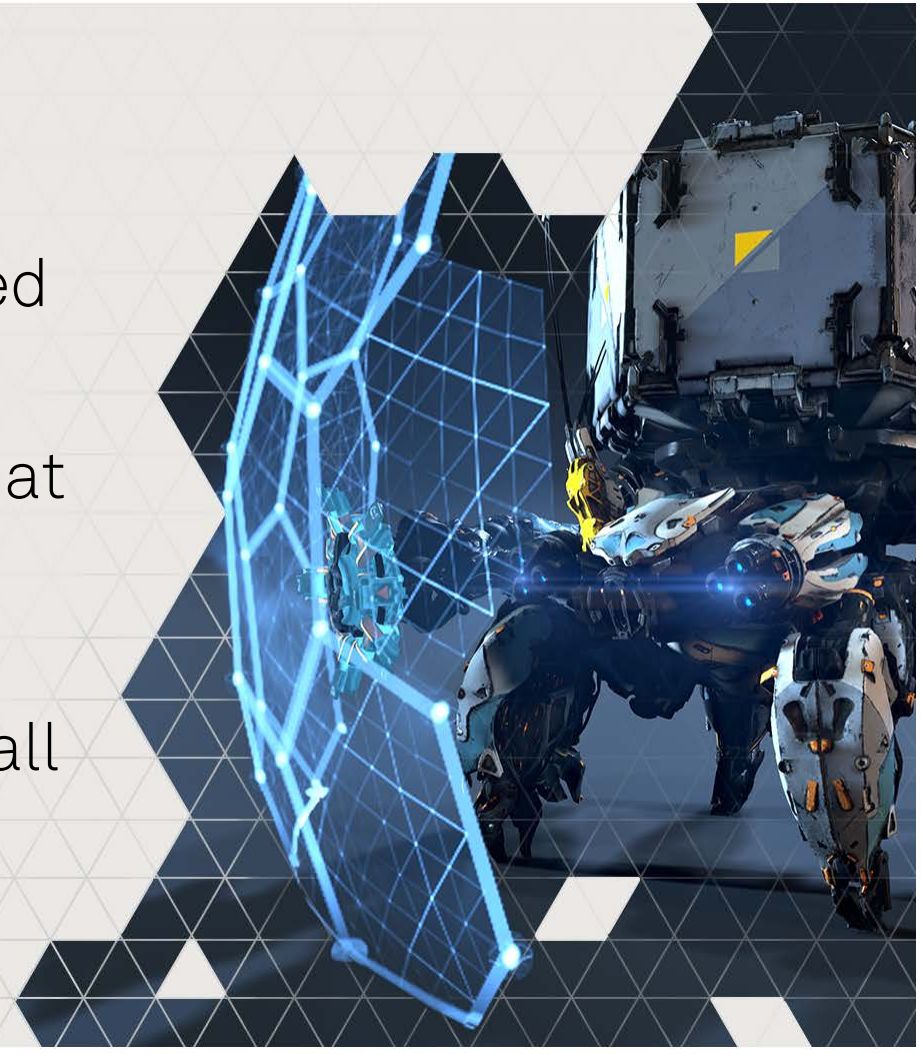
Conversation Editor



The first steps to a new editor

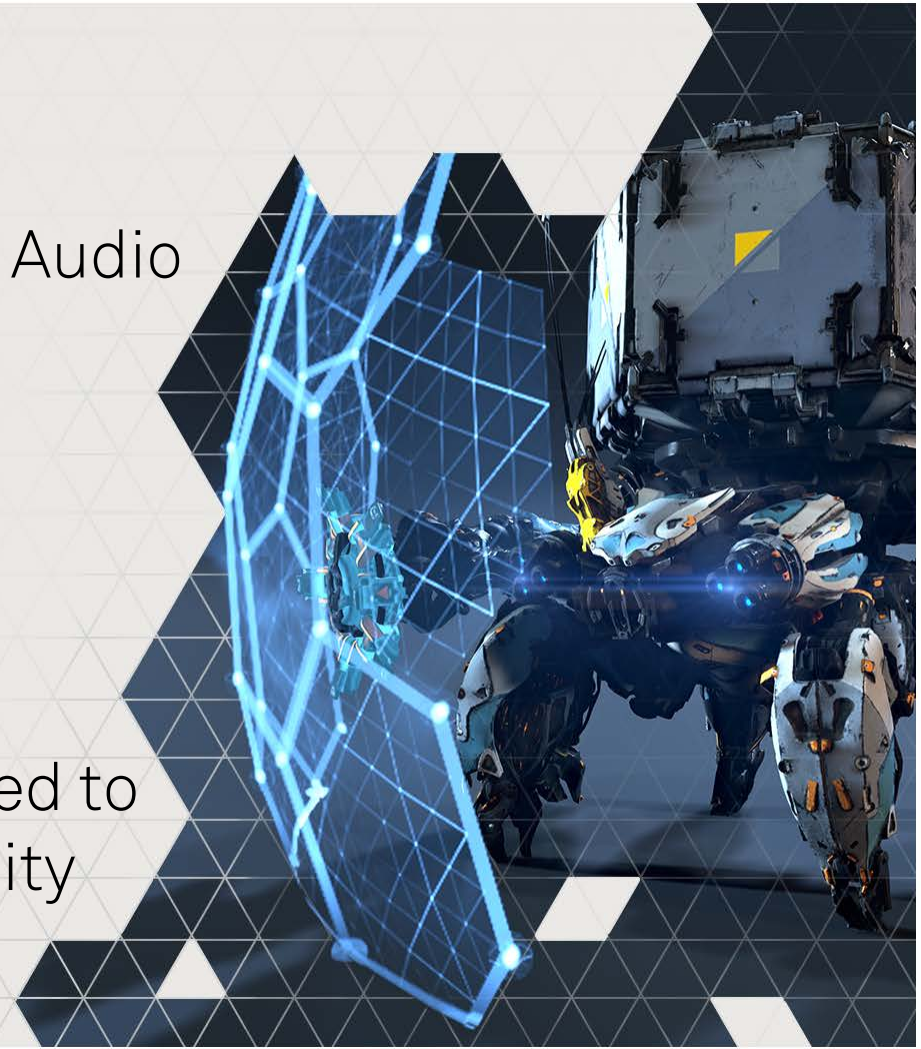
Where do we begin?

- 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?

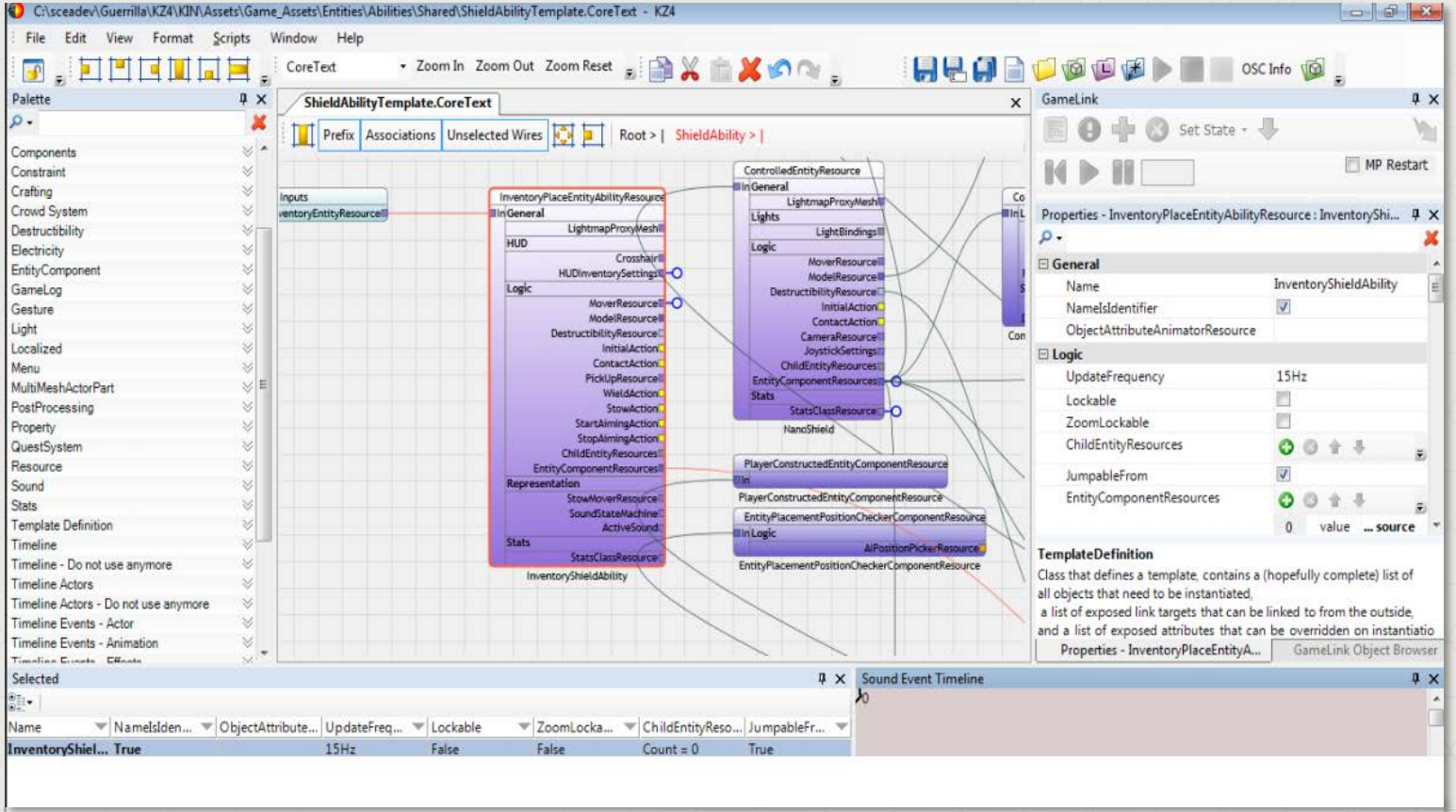
- We decided to focus on the Audio Team
- A small group of users with specific needs
- Their workflow could be used to test the Framework's viability



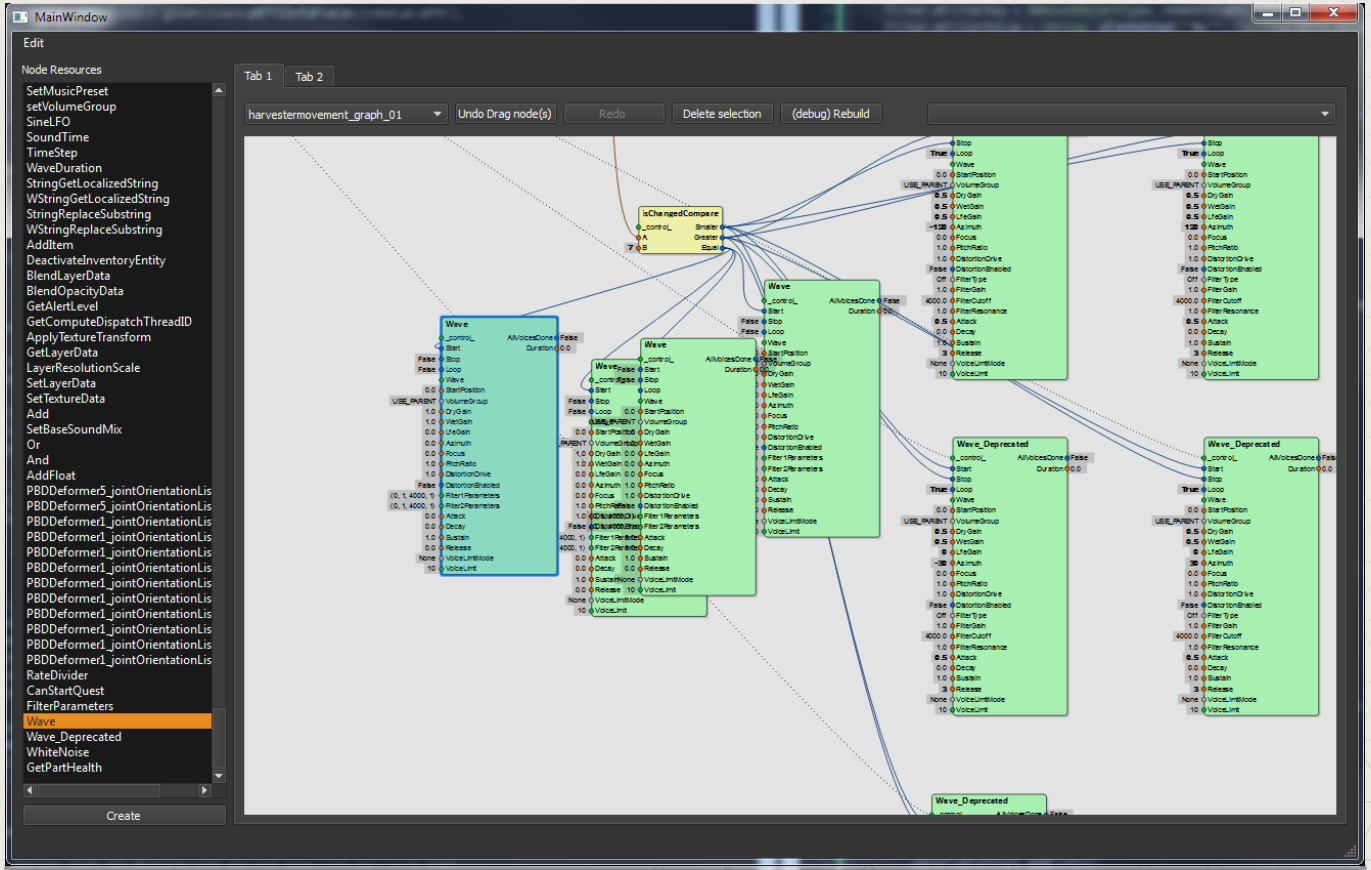
Where do we begin?

- 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]

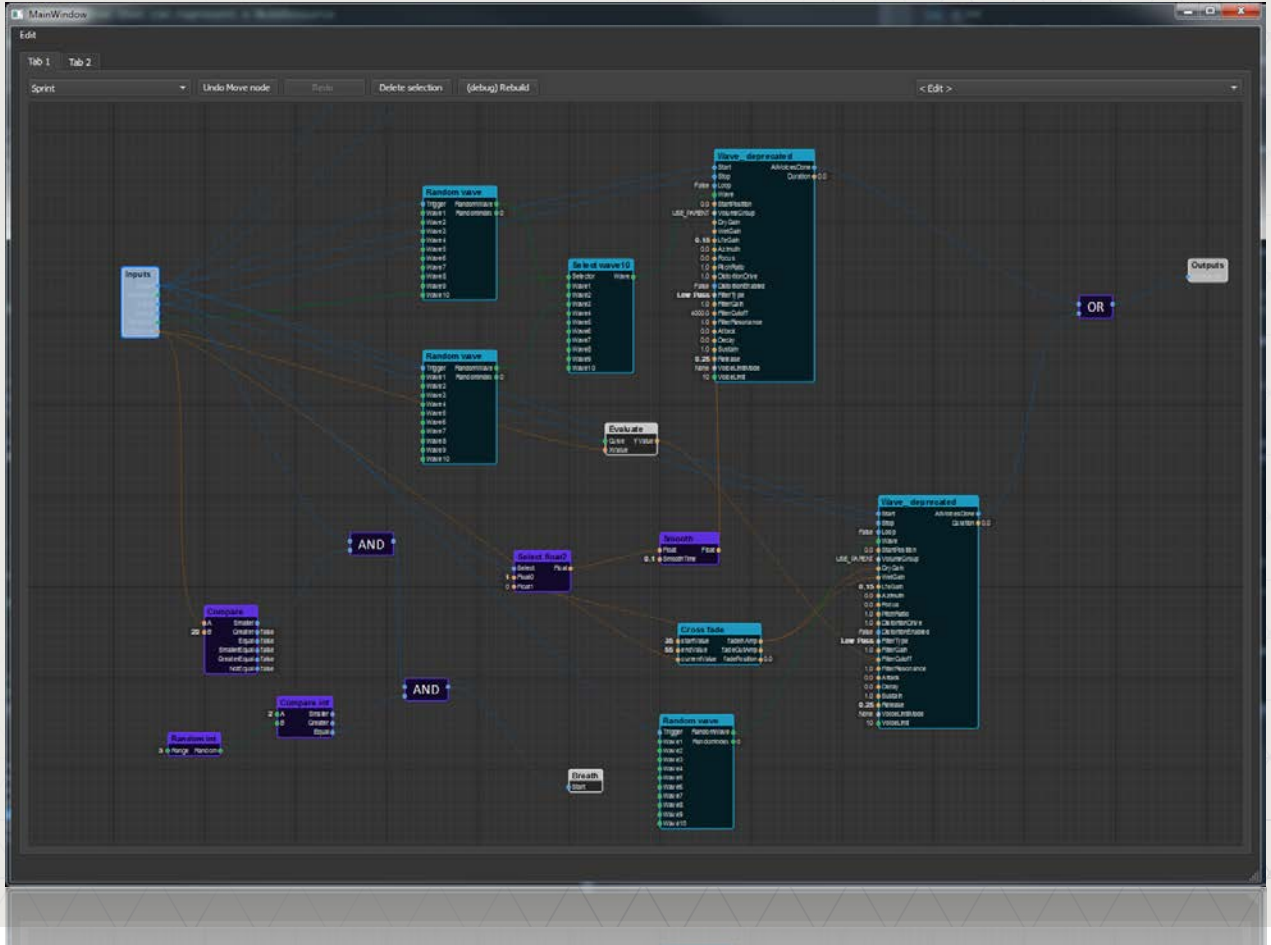
CoreText Editor



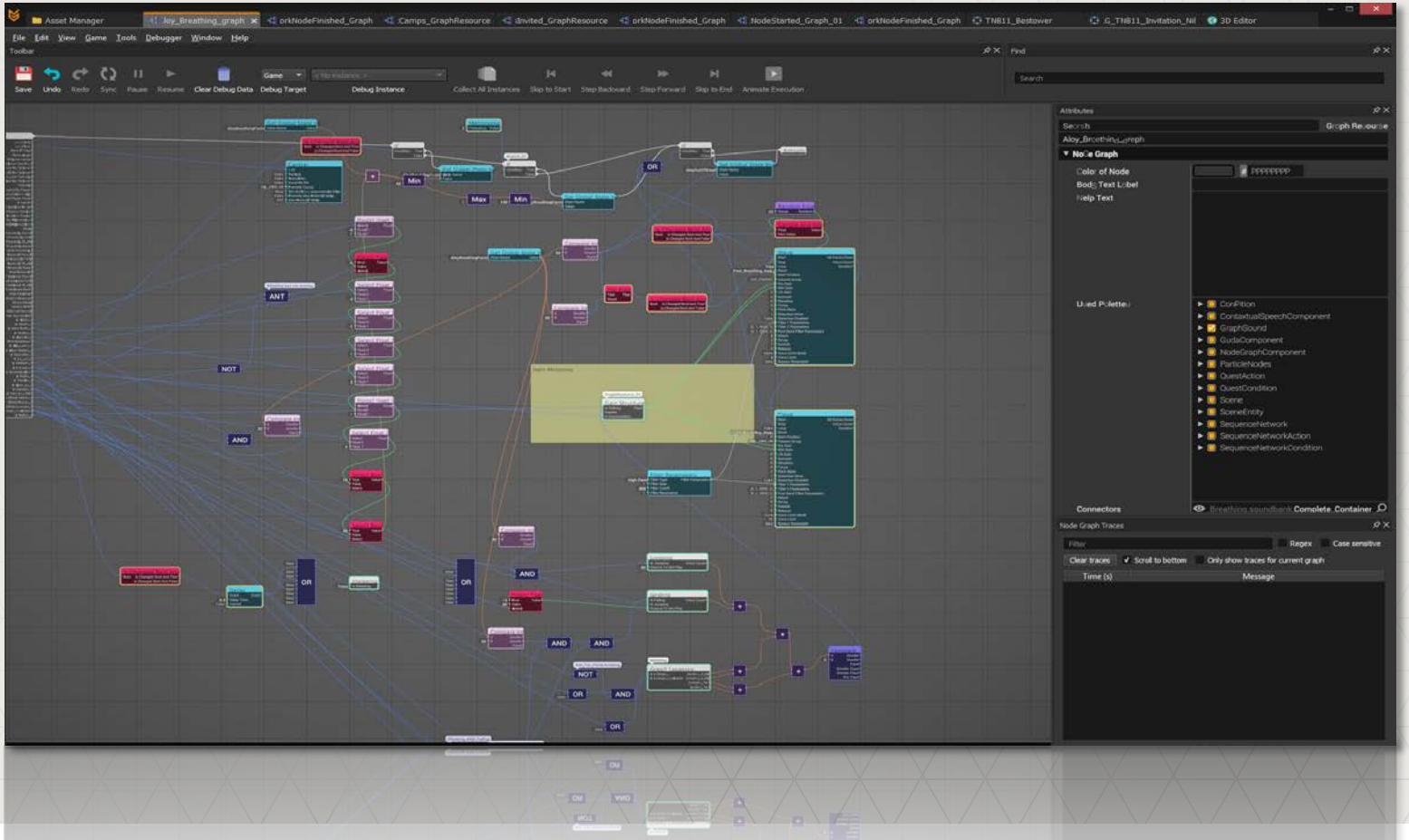
NodeGraph: Prototype #1



NodeGraph: Prototype #2

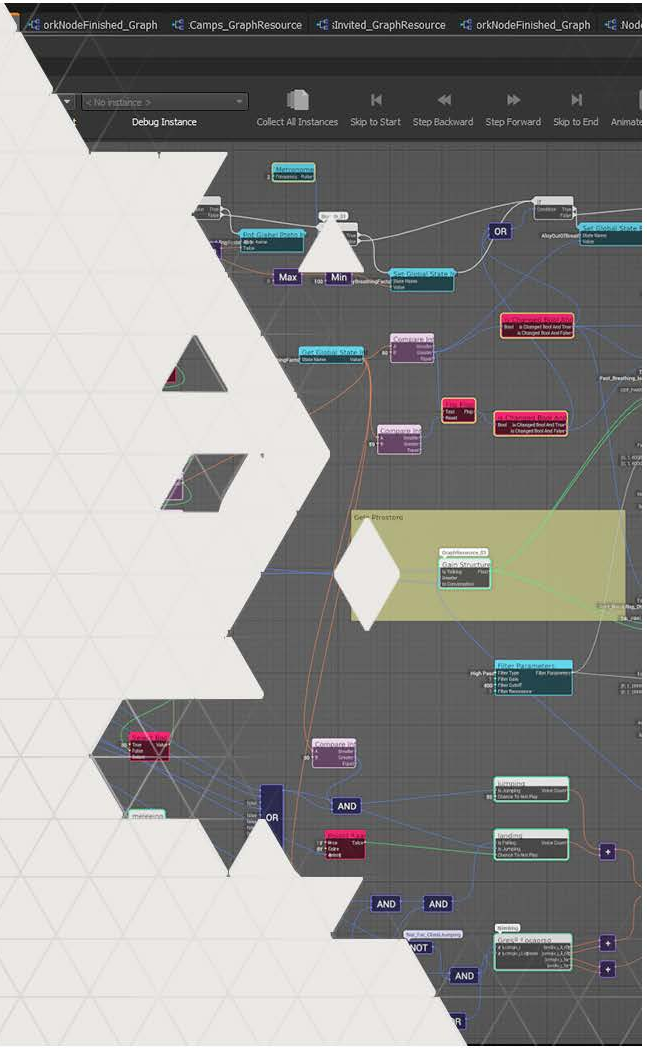


Node Graph Editor Context



Node Graph Editor

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

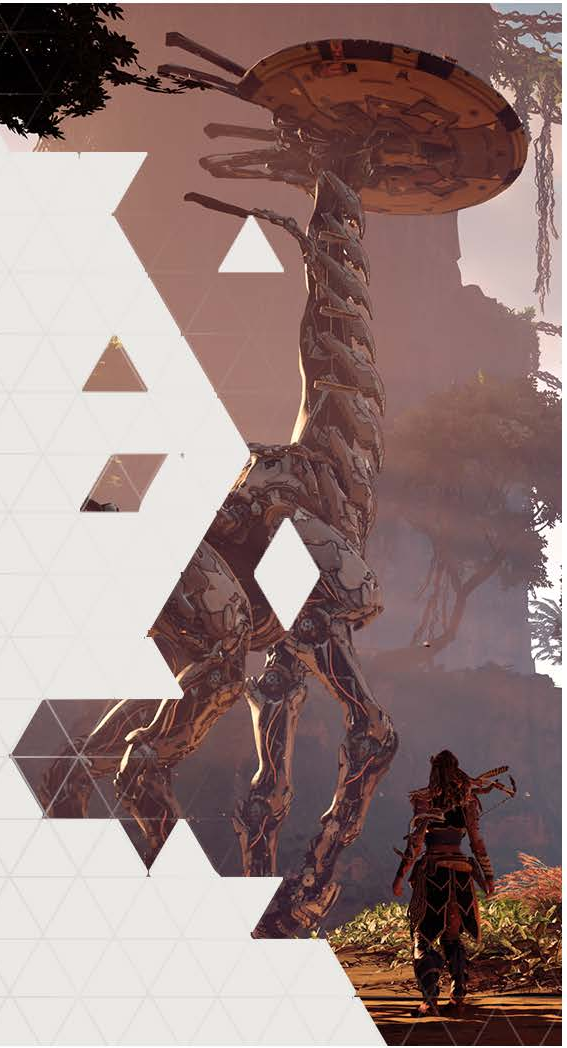


A new workflow for Game Design

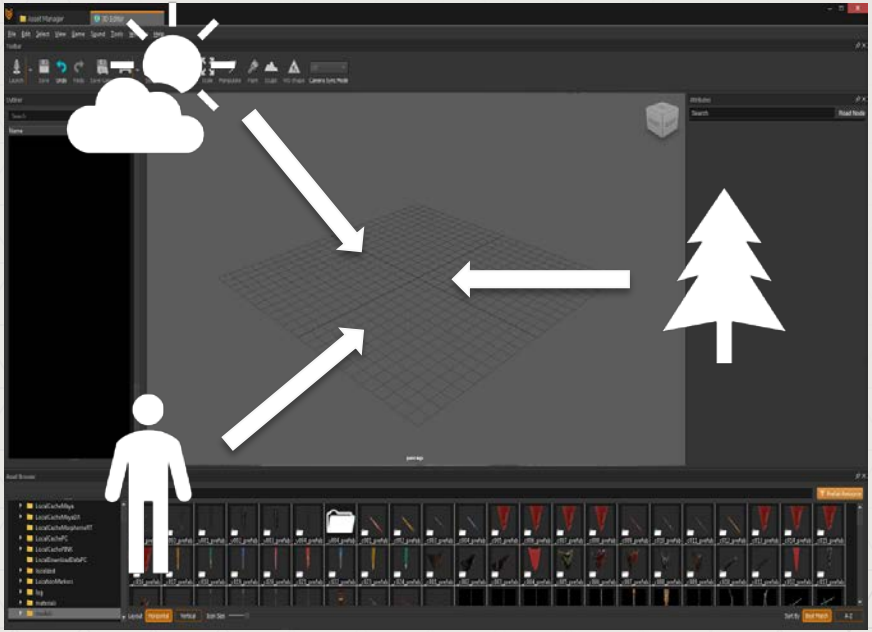
Creating a 3D Editor

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?



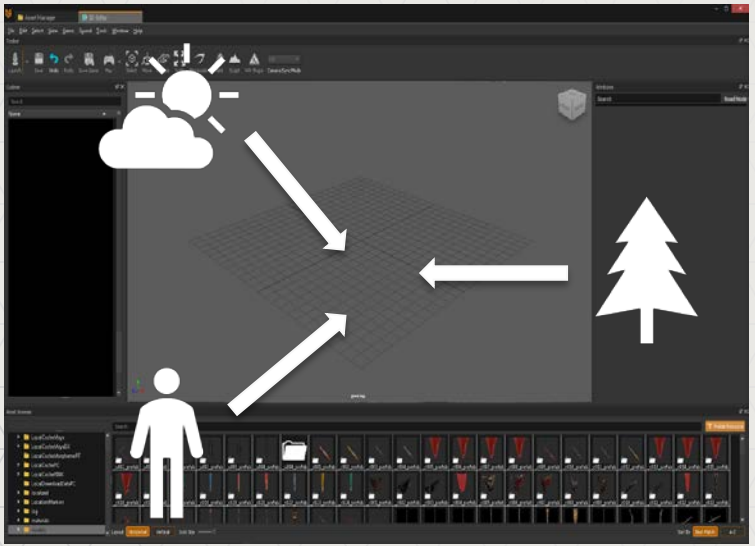
Integrate only sub-systems into Editor?

+ Purpose build, controllable
+ Fast

- New code-path

  Longer development

- Visual/behavioral disparity



Integrate full game into editor?

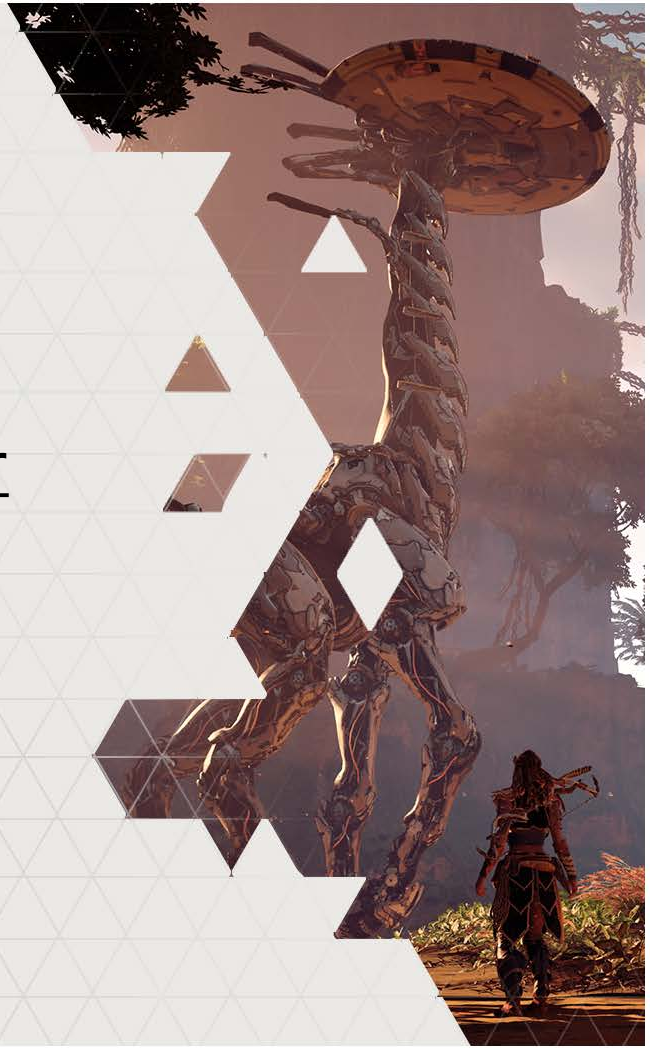
+ All game systems available
+ *DebugUI* (tools) for free
+ Single code path (play/edit)

- Overhead of running game

3D Editor – Architecture

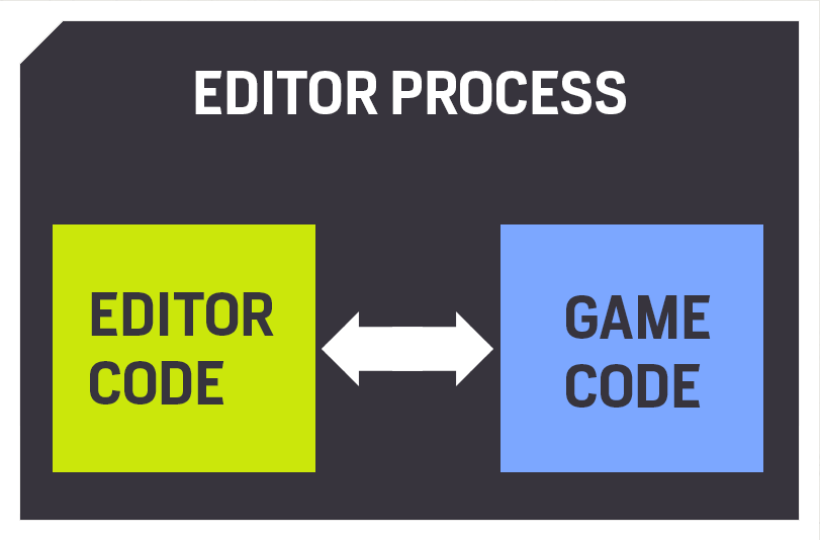
Due to time constraints:

Integrate full game into Editor

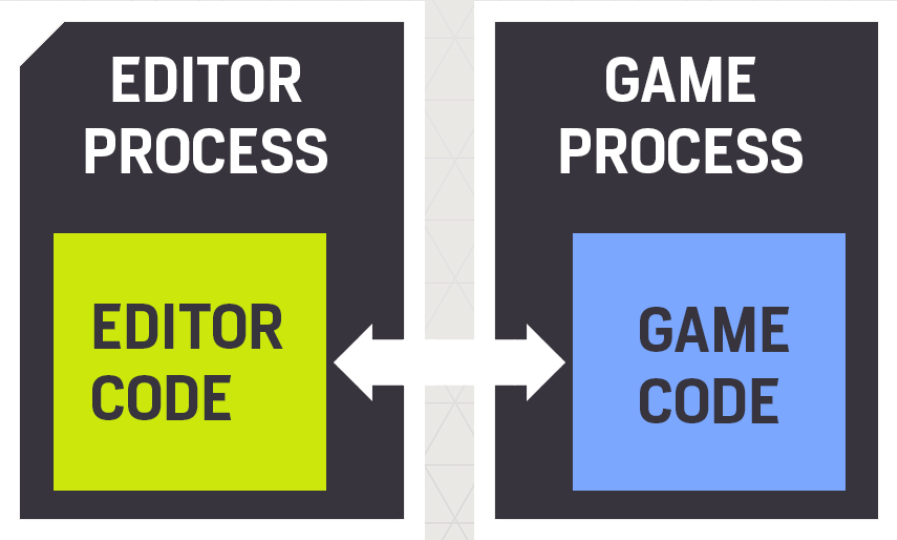


How to integrate a full game into Editor?

In process?



Out of process?

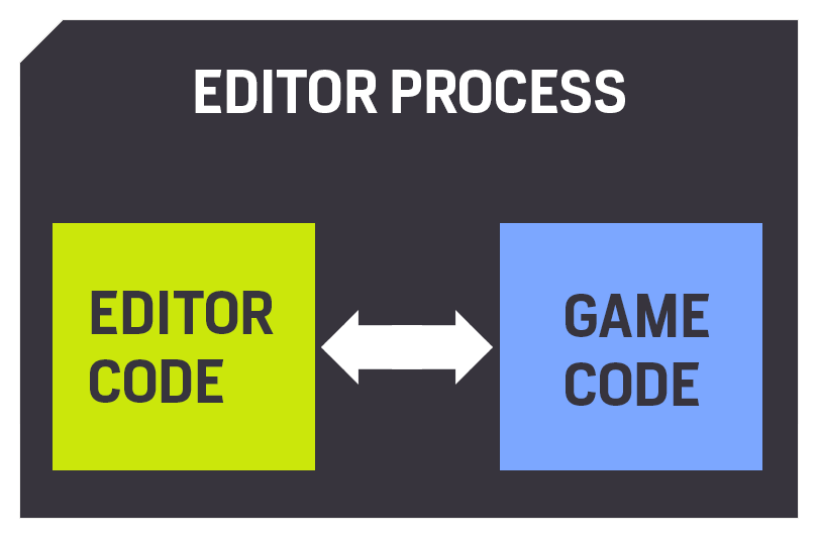


In-process integration

+ Fast to setup

+ Direct synchronous data changes

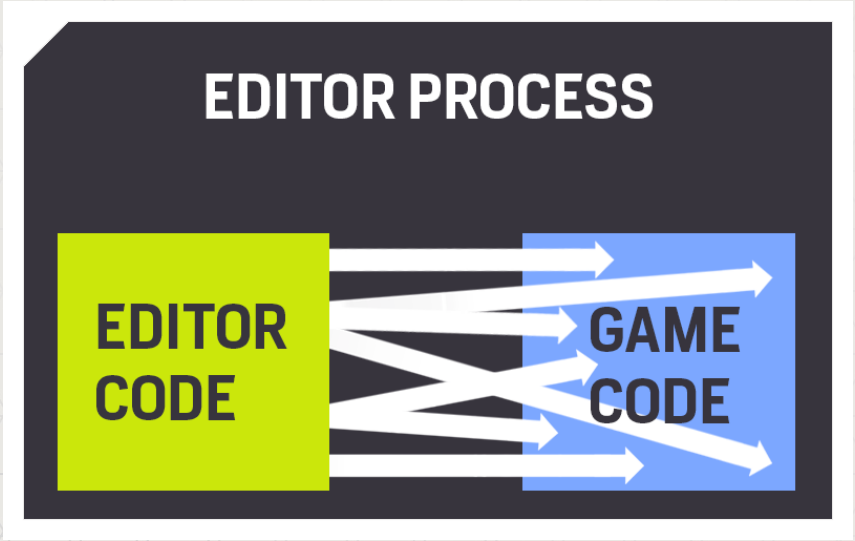
- Crash in game = crash in editor



In-process integration

+ Fast to setup
+ Direct synchronous data changes

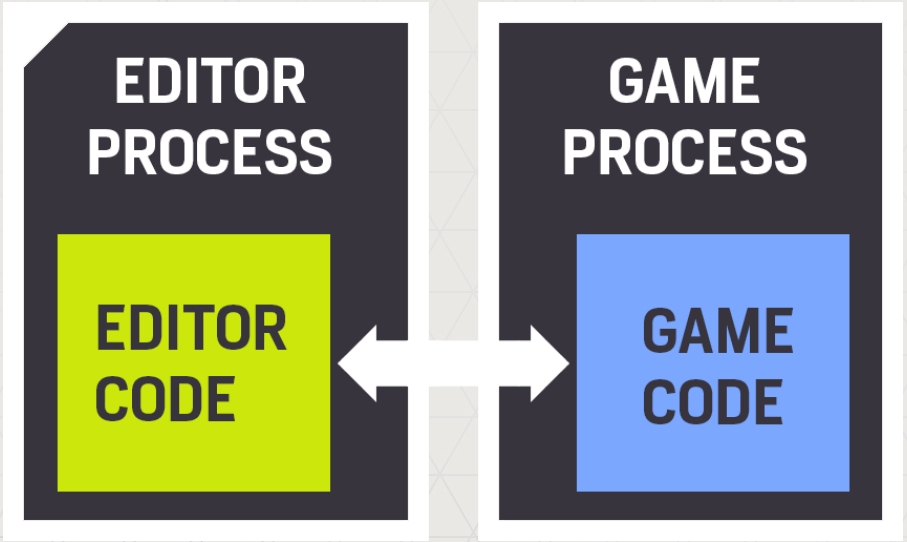
- Instability
- Enforcing code separation



Out of process integration

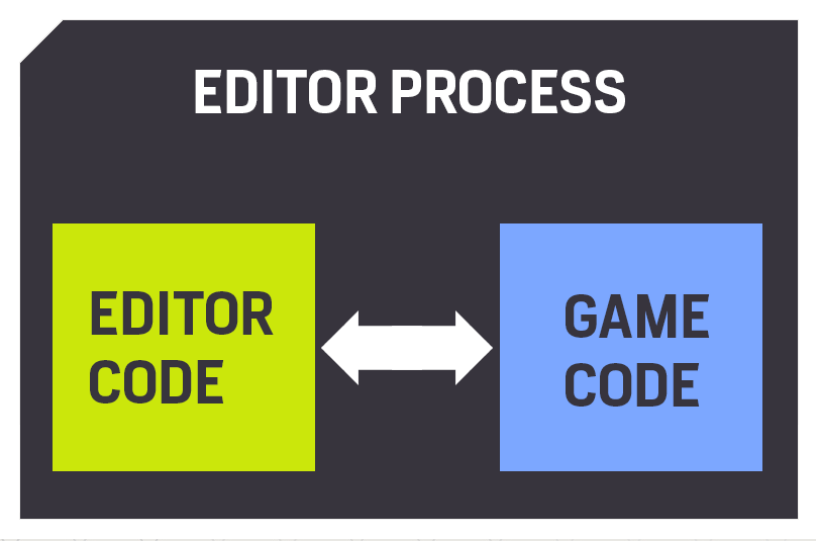
+ Stability
+ Cleaner code
+ Responsive UI

- 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



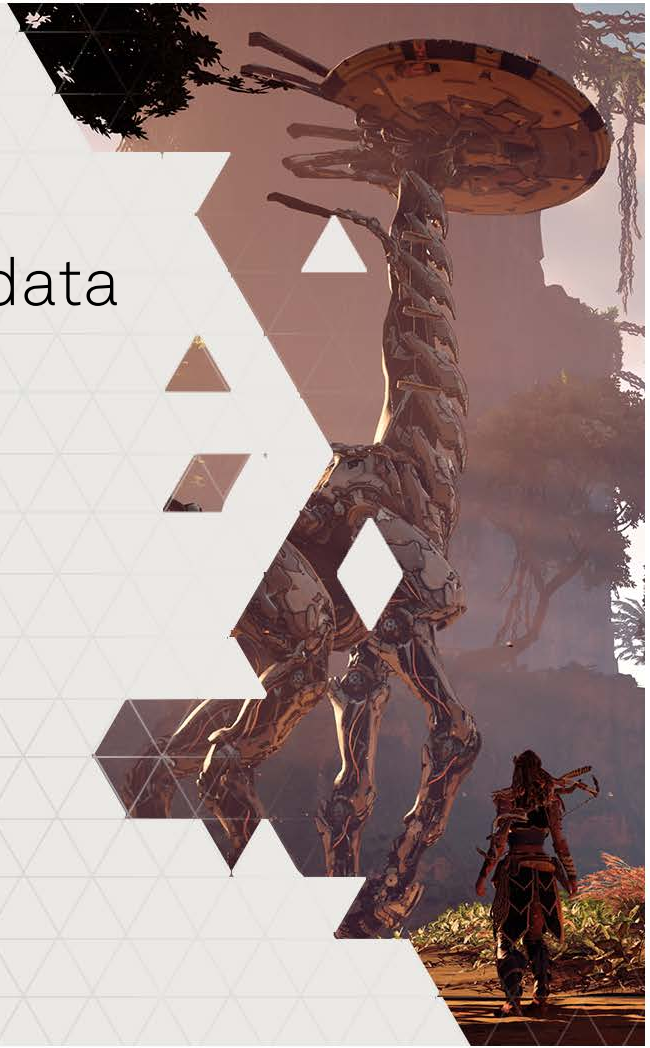
Editing content in a running game

Required sub-systems for Level Editor

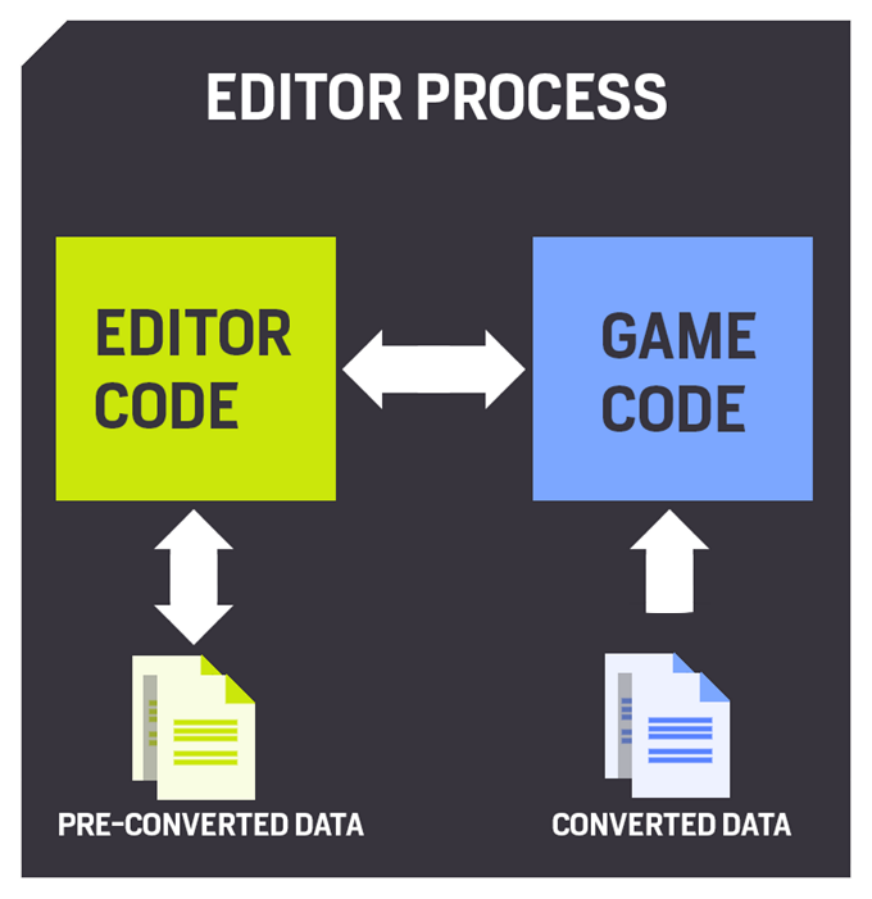
Game data

- **Problem**: Game reads binary game data

- Creation is *destructive*
  - Shader compilation
  - Meshes optimization
  - Data packing

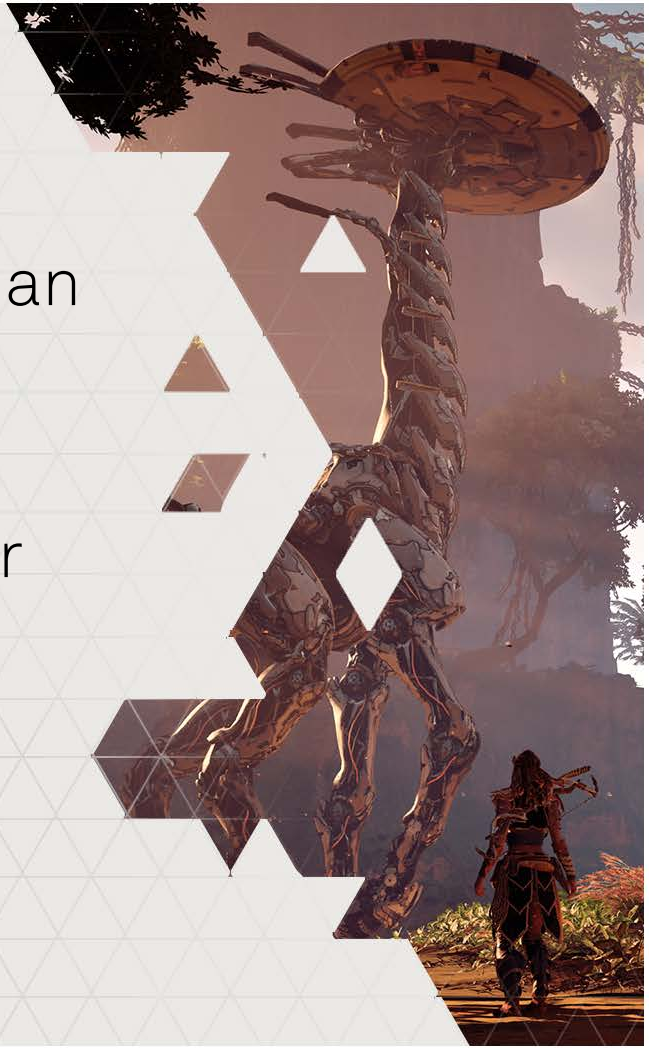


3D Editor: Initial design



EditorNode

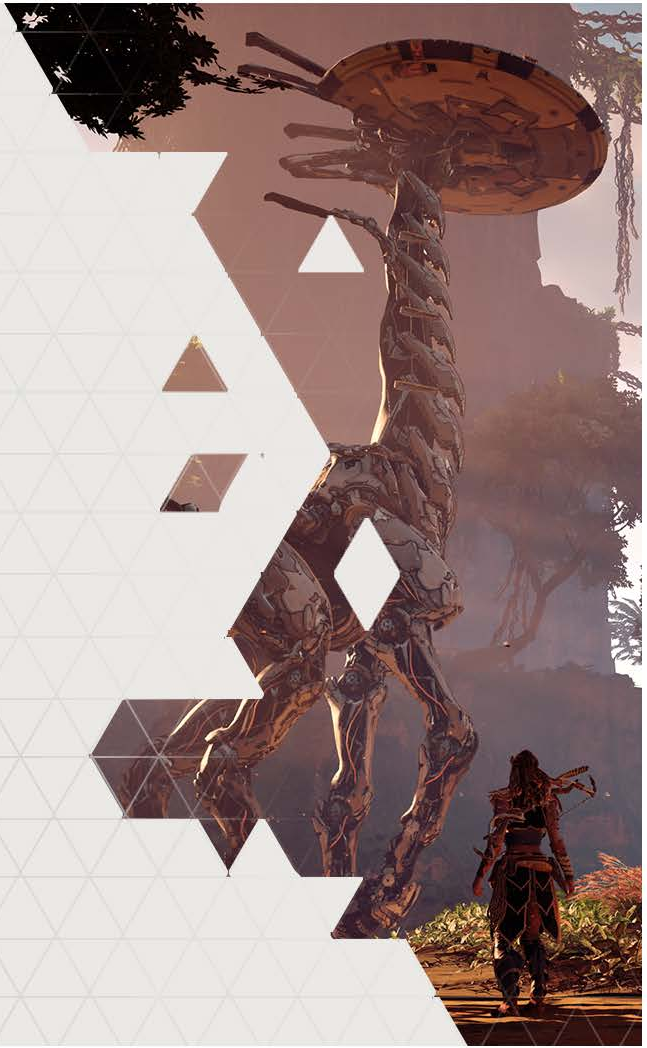
- Provides **all** editing functionality for an object in game
- Game object does **not** know of Editor Node
- Keeps code clean, no editor specific code interwoven in game code



Only WorldSpace in-game

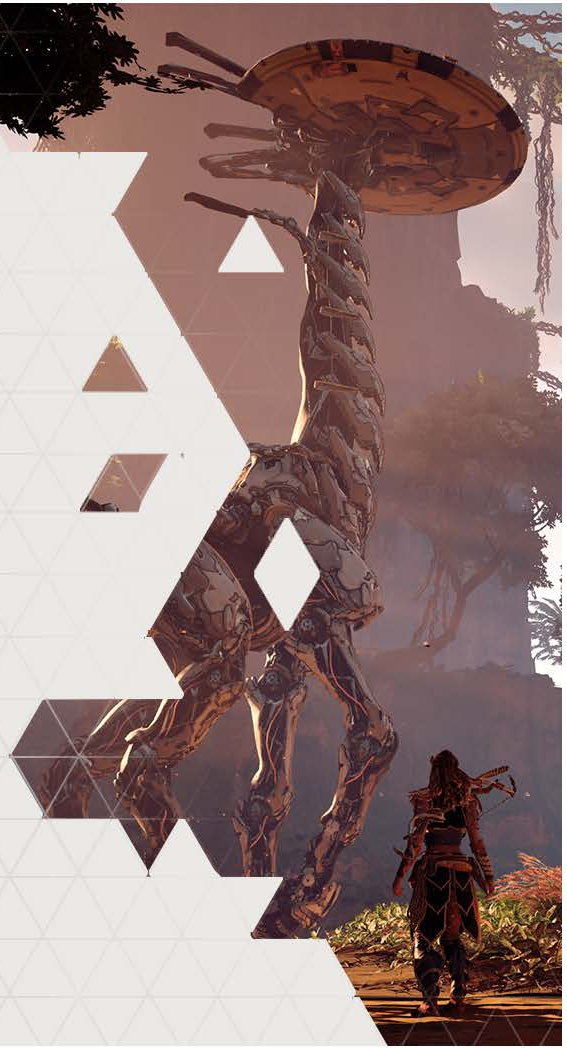
- **Problem**: Less hierarchy in game

- Flat list of objects
  - Fast and efficient
  - Transforms in World Space

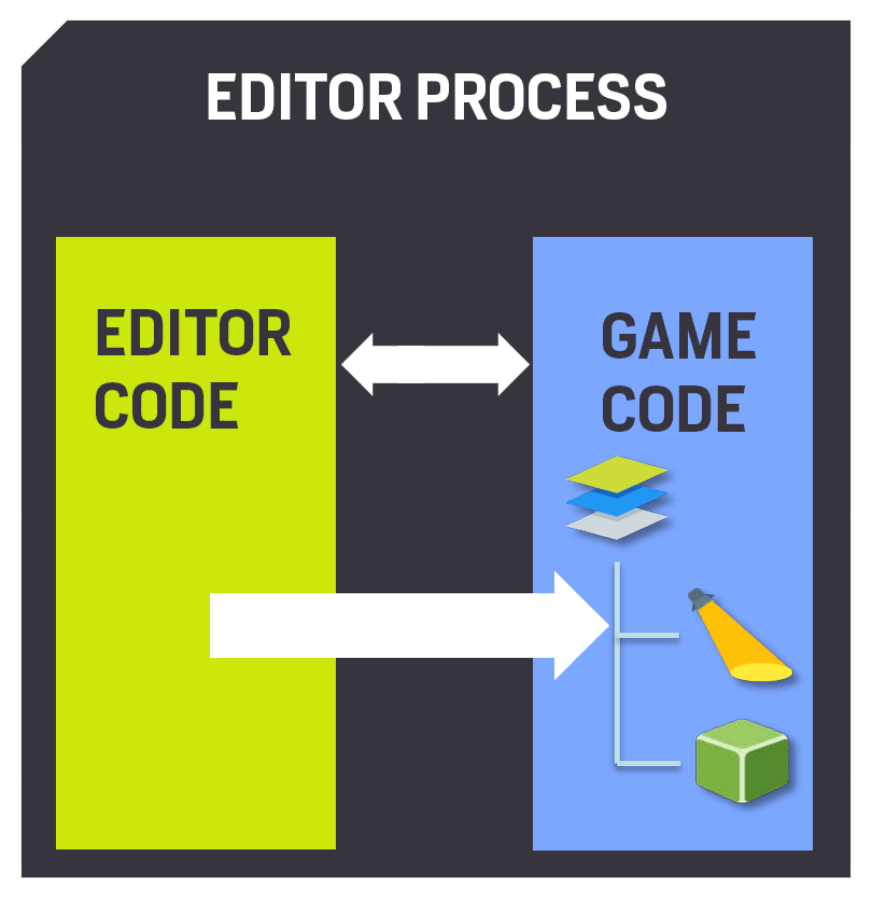


EditorNodeTree

- 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

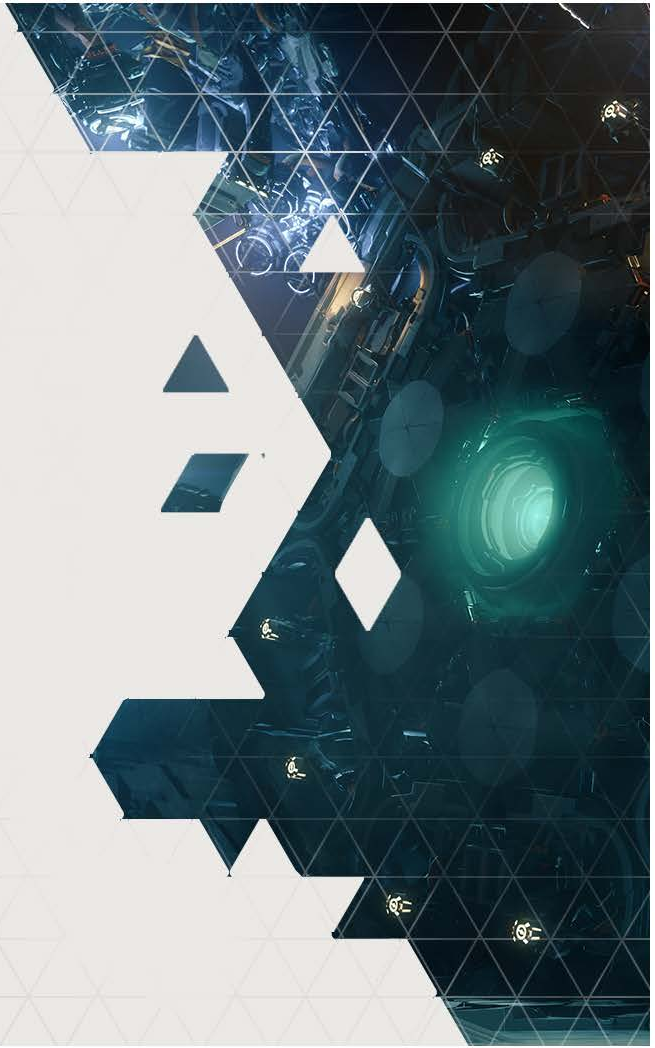


2 Editor Node Trees

Making Changes to Objects

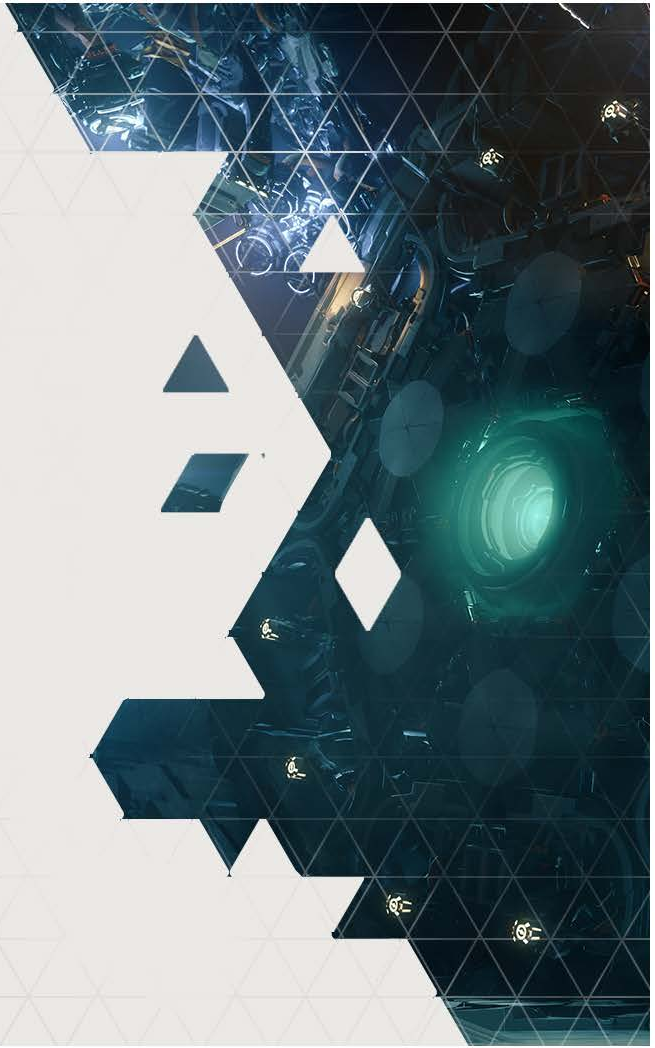
What we didn't want

- 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

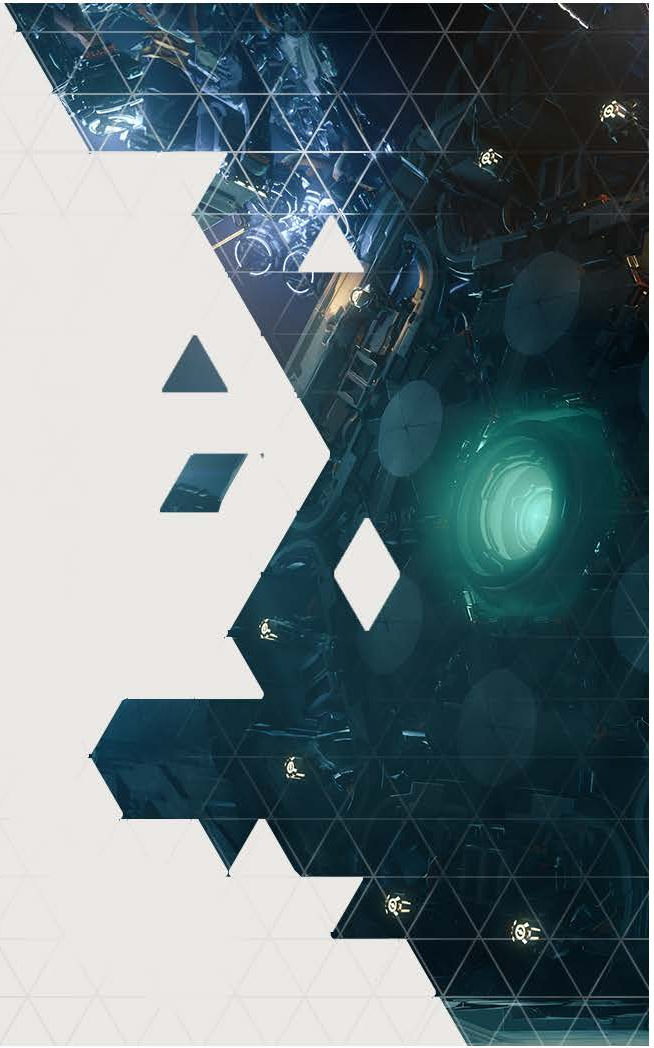


Making changes to objects

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

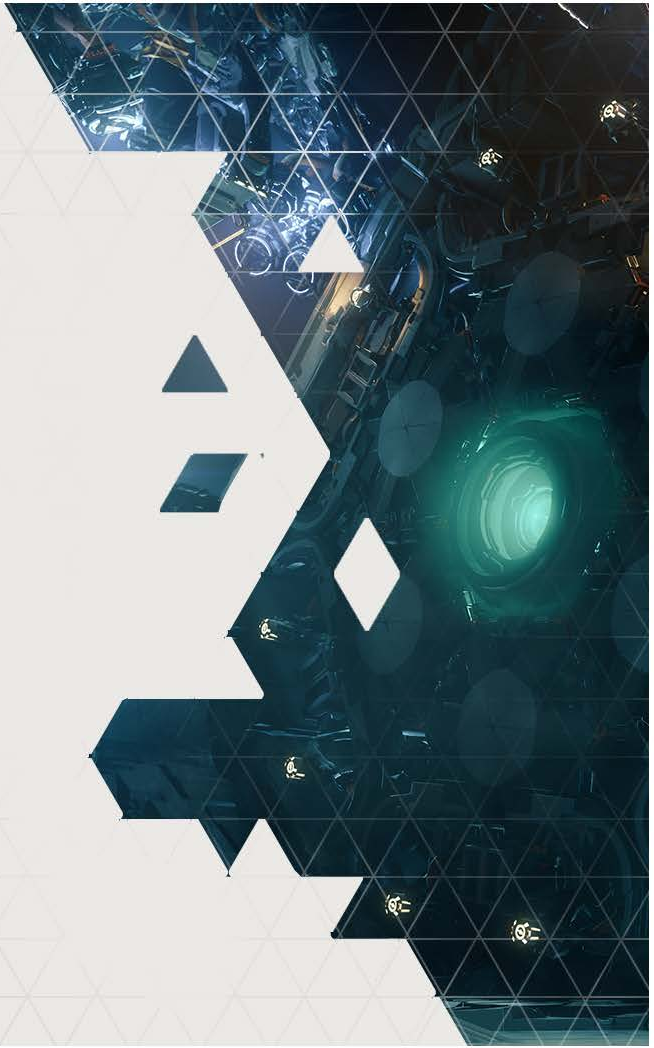
"What is best in life?"

- 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



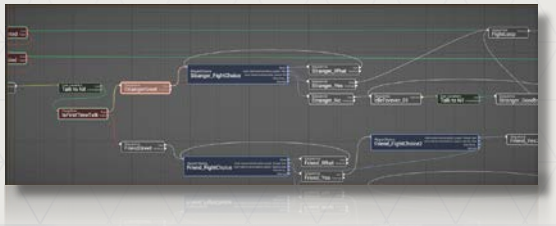
Undo System

- 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

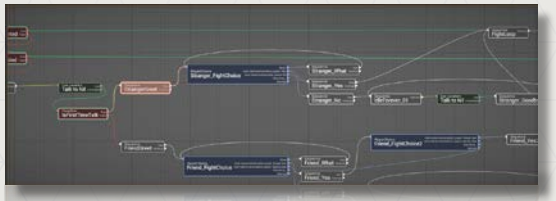
- Describes all modifications in a change
  - Commands represent individual changes
- Built on the RTTI system
- Subscribers are notified of all transactions

Transaction Notifications



ADD OBJ
ADD OBJ
SET VALUE
SET VALUE
DEL OBJ

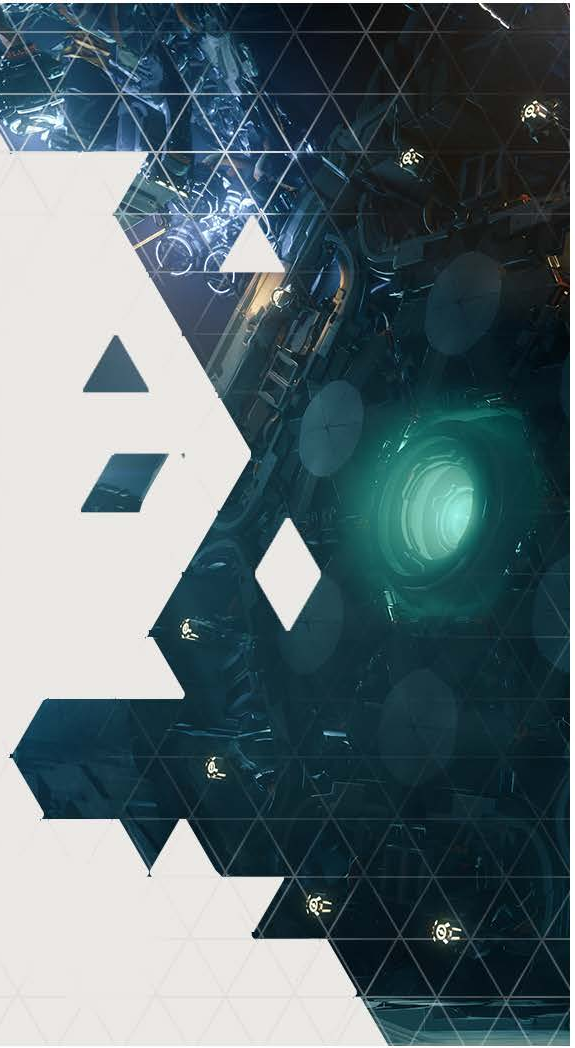
Transaction Notifications



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

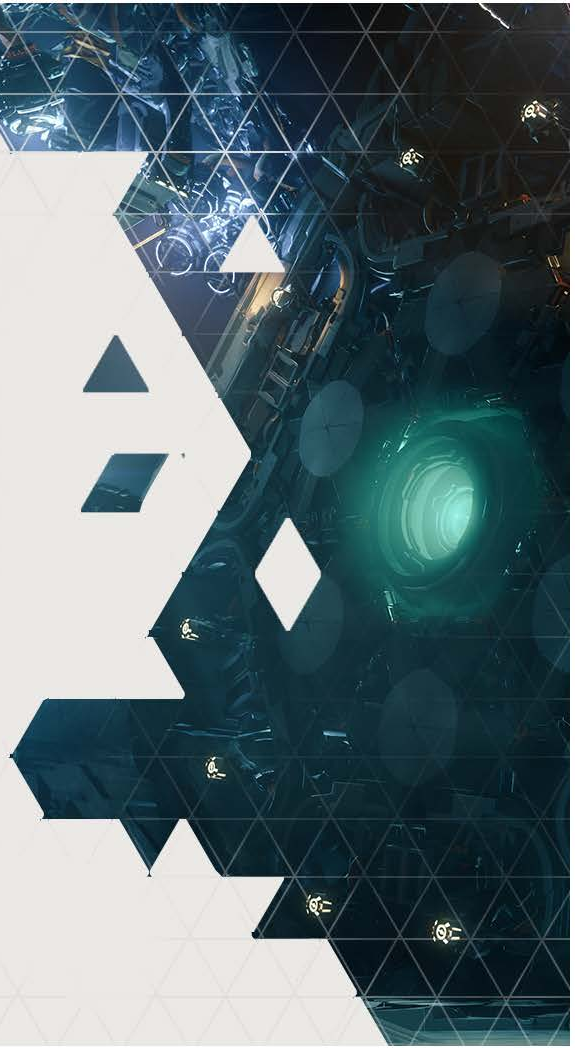
Undo System – DiffUtil

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



Undo System – DiffUtil

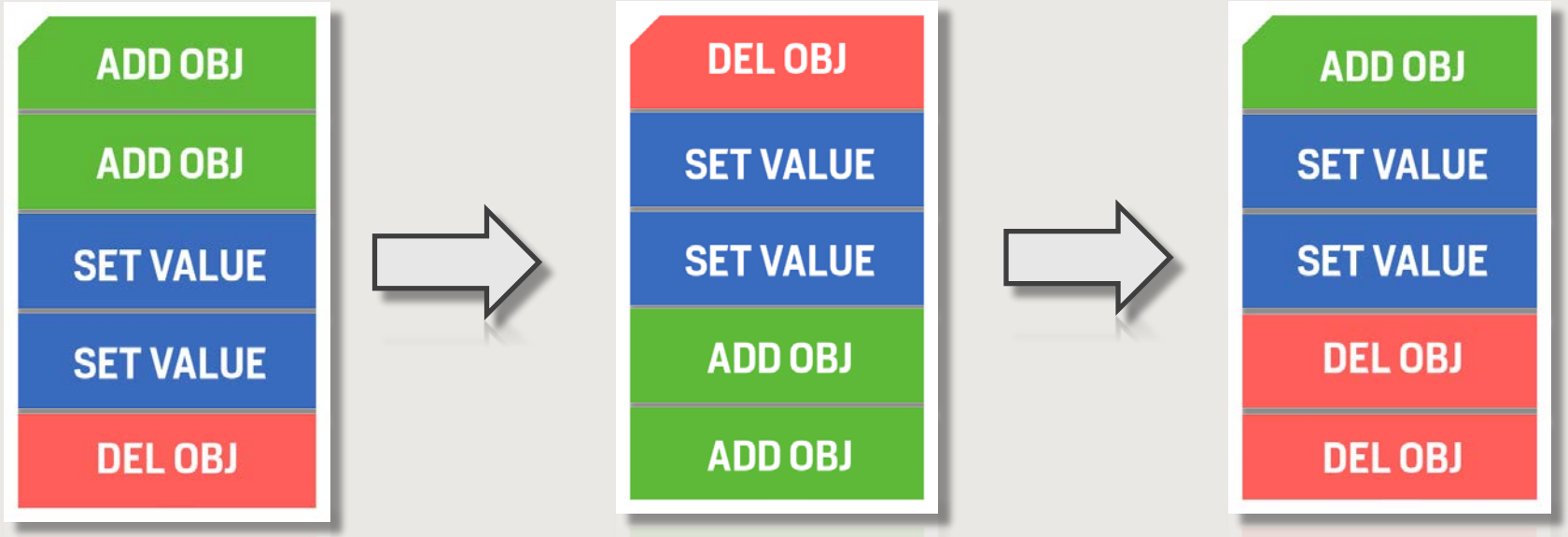
- Before changes are made:
  - DiffUtil takes **snapshot** of objects' state
- After changes:
  - Analyzes what has changed between the snapshot and the current state



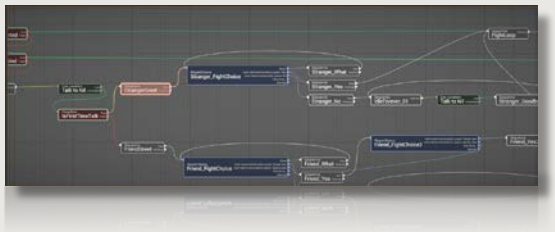
DiffUtil – Code Example

```
// 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");
```

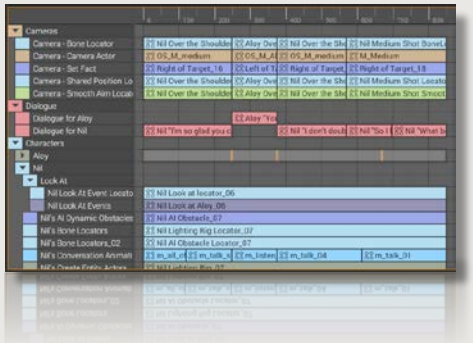
Undo System – Reverting Changes



Applying the Undo Transaction

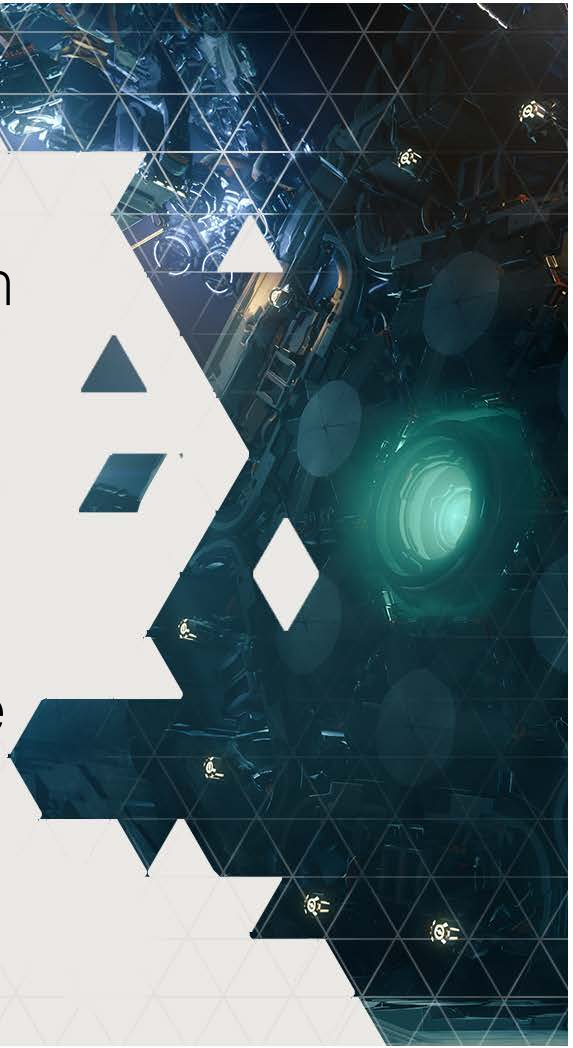


ADD OBJ
SET VALUE
SET VALUE
DEL OBJ
DEL OBJ



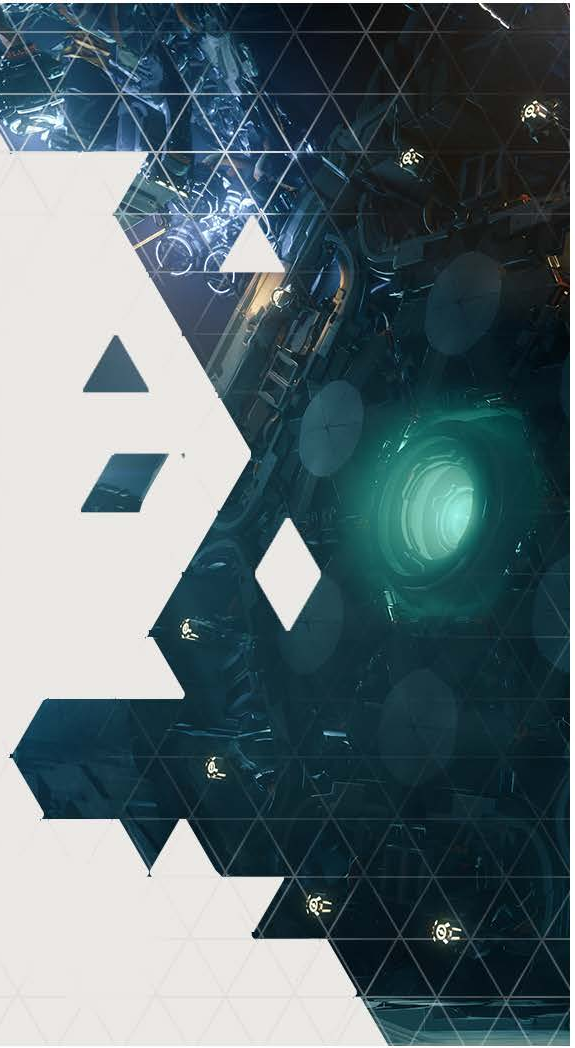
What worked?

- 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



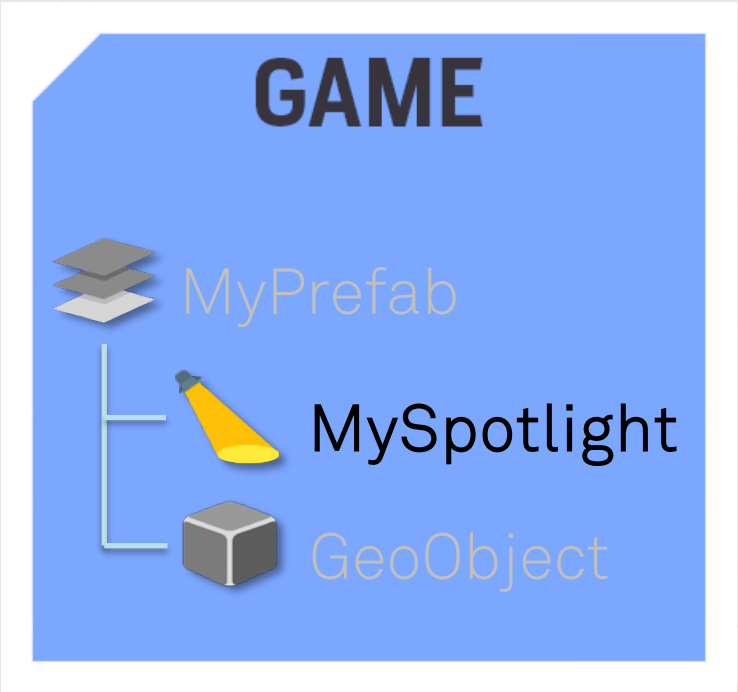
Transactions in the 3D Editor

Changes in the 3D Editor

- Transactions to the Game

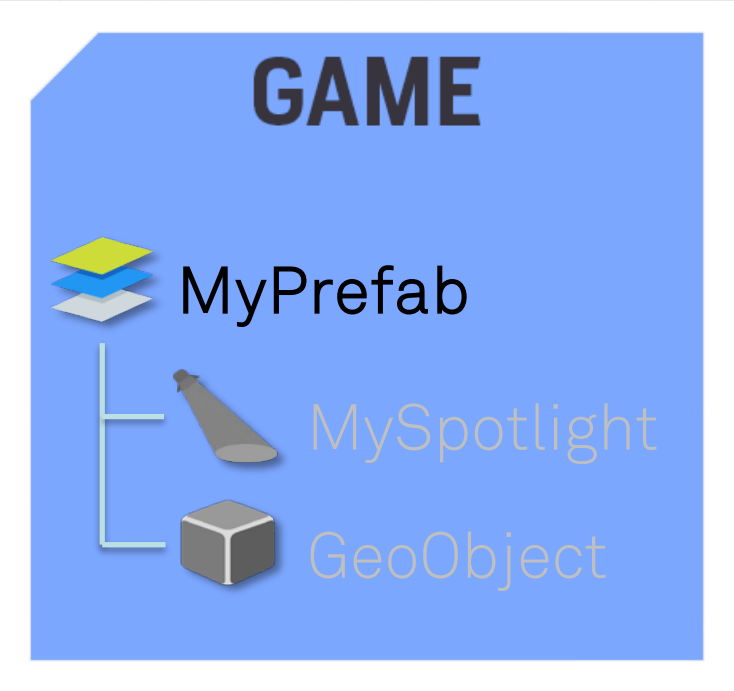
EditorNodeTree in space conversion

3D Editor – Applying a change in the tree



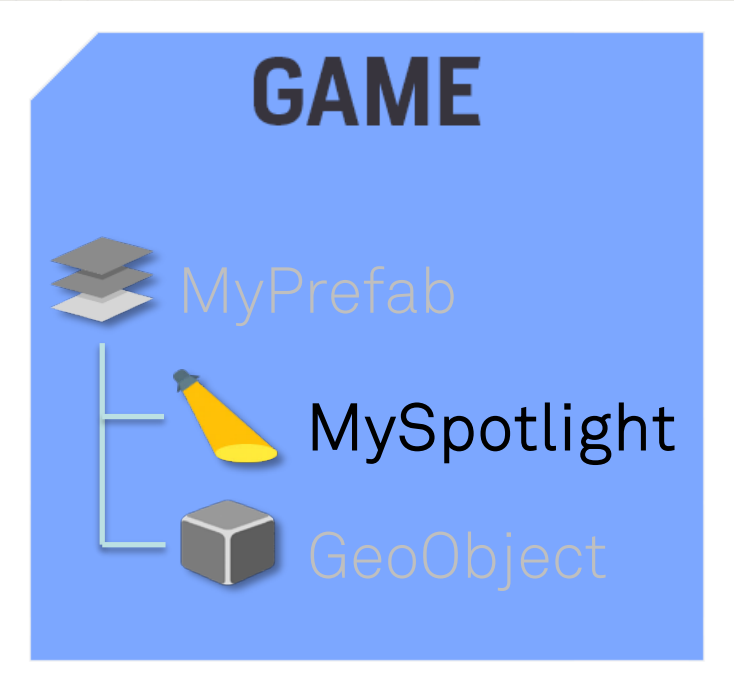
- Modify local space of *MySpotlight*

3D Editor – Applying a change in the tree



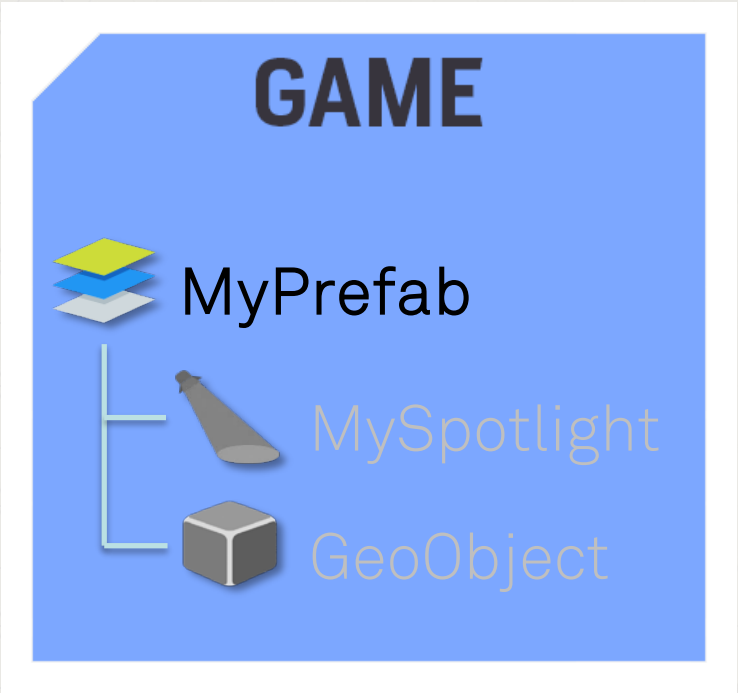
- *MyPrefab* multiplies transform change with it's local matrix
- Modified transaction passed to *MySpotLight*

3D Editor – Applying a change in the tree



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

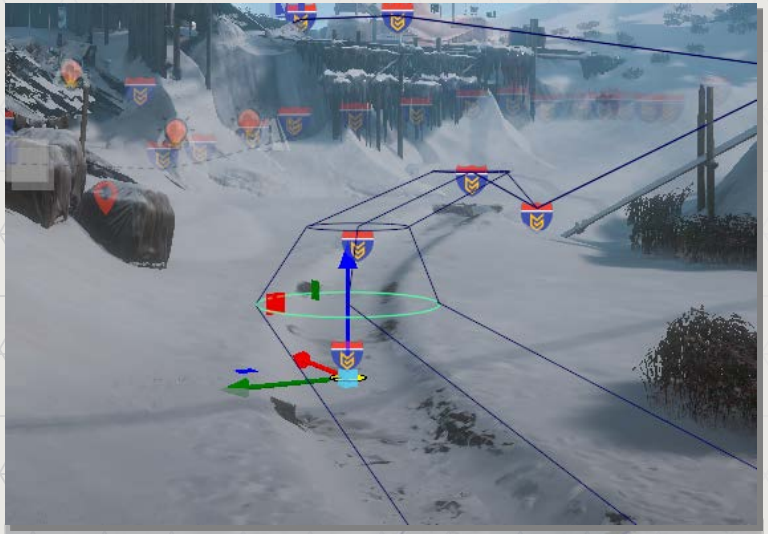
3D Editor – Applying a change in the tree



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

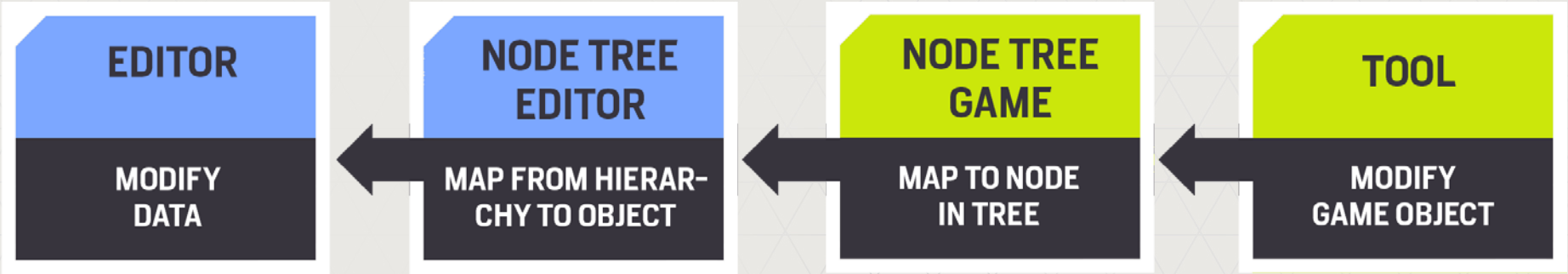
Interactive changes

- **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 ?

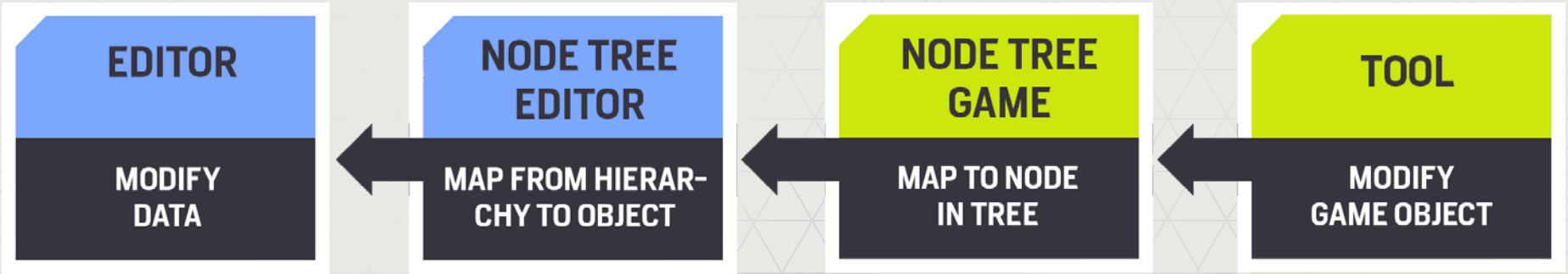


Naïve implementation of tool change

- (Tool) Transactions from the Game ?

Naïve implementation of tool change

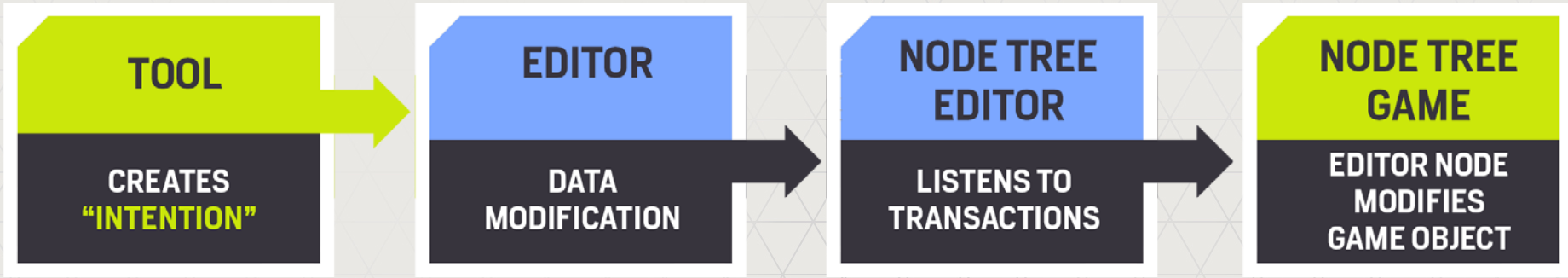
- Who has authority over data?
- Impossible due to dataloss in conversion
- Creates a cycle going back to game



Tool changes

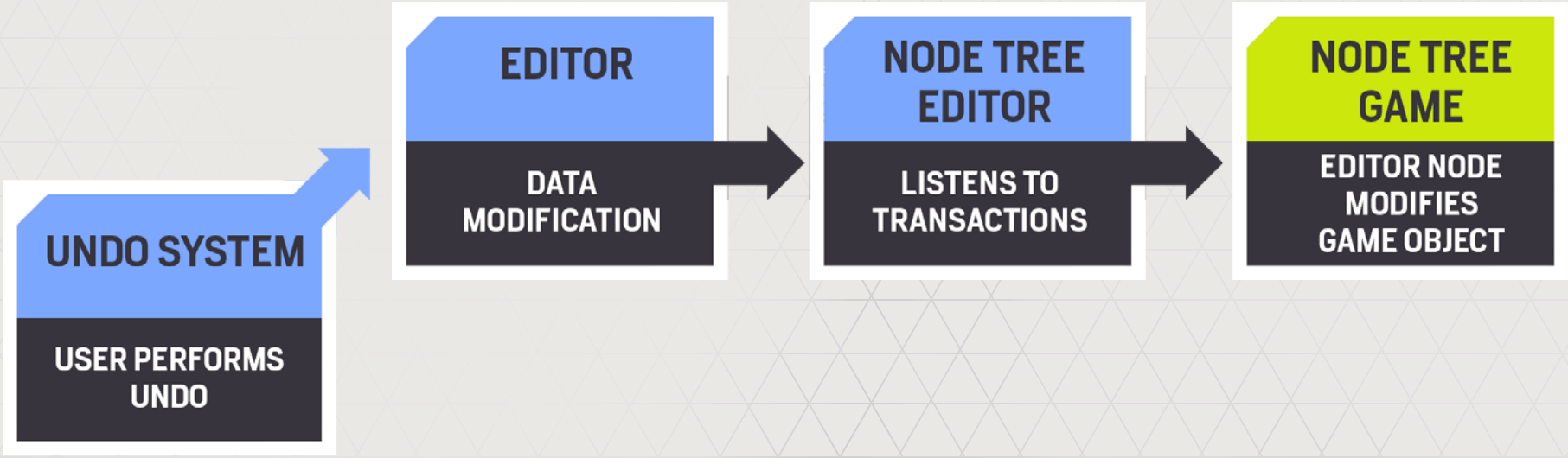
Solution:

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



Undoing tool changes

Undo works the same way!



GameSync

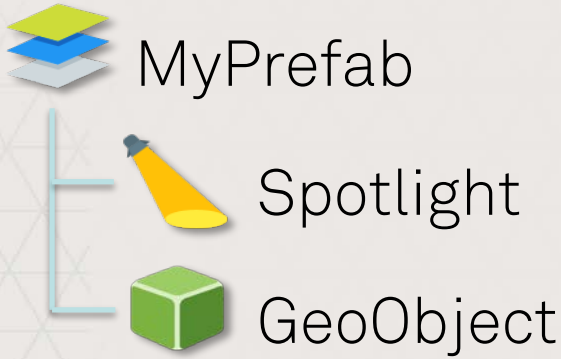
Updating structural changes in the game

GameSync

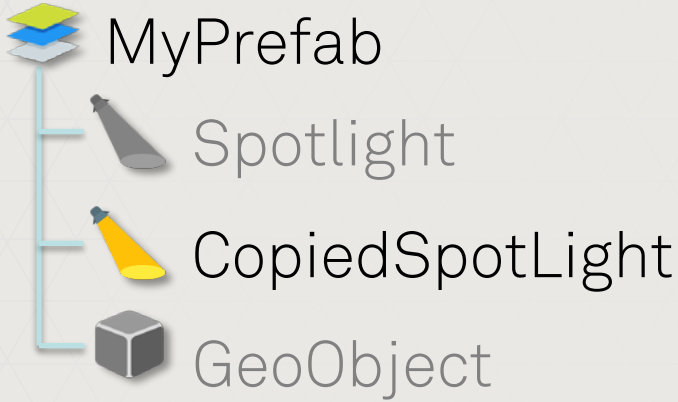
- *Gamesync* handles structural changes to the game
- Pre-dates Tools Framework
- Creates patch file to apply in game

GameSync

- Create a copy of *Spotlight*

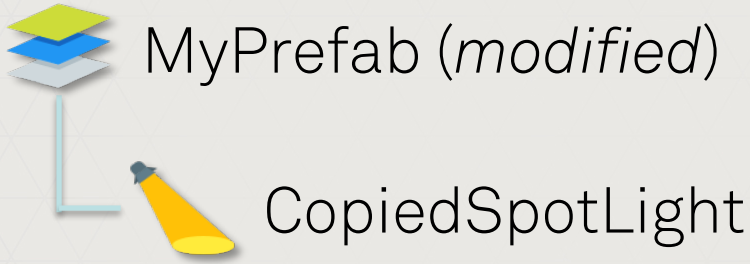


GameSync – Analyze change



GameSync – Patch file

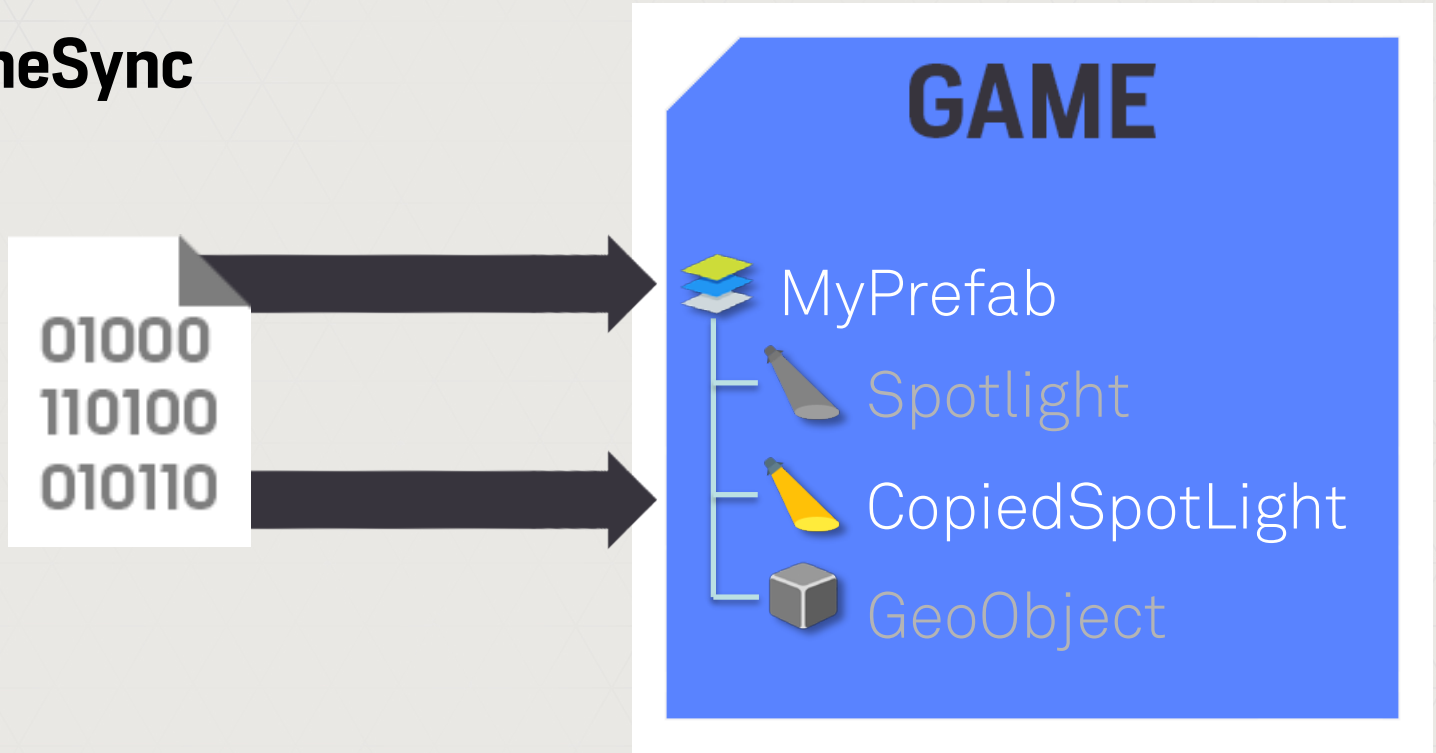
GameSync – Conversion



GameSync

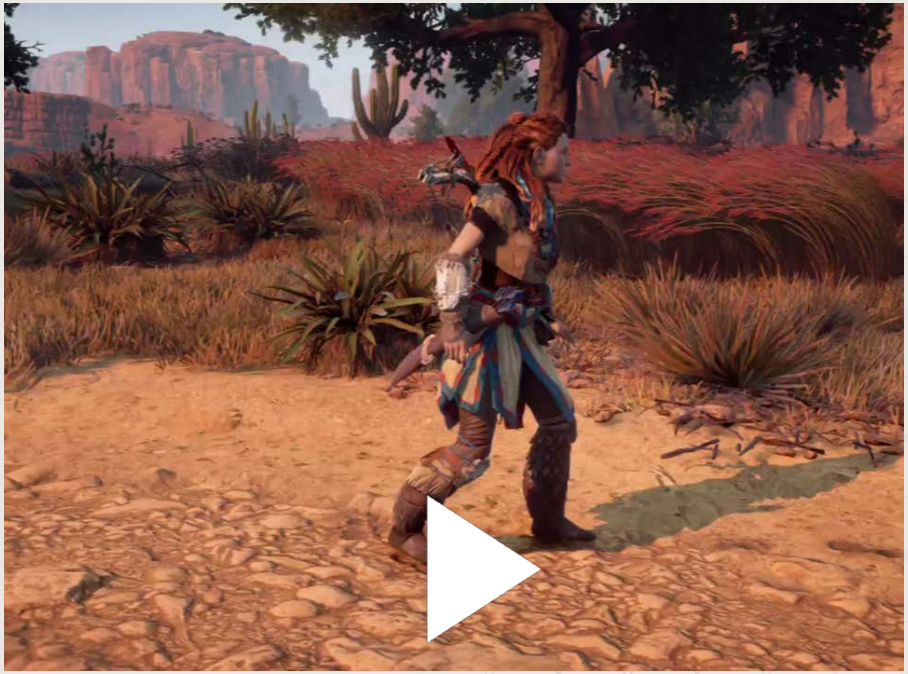
ANALYSE CHANGE → CREATE PATCH FILE → CONVERT PATCH FILE → PAUSE GAME SYSTEMS → APPLY PATCH → RESUME GAME

GameSync



ANALYSE CHANGE → CREATE PATCH FILE → CONVERT PATCH FILE → PAUSE GAME SYSTEMS → APPLY PATCH → RESUME GAME

GameSync



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

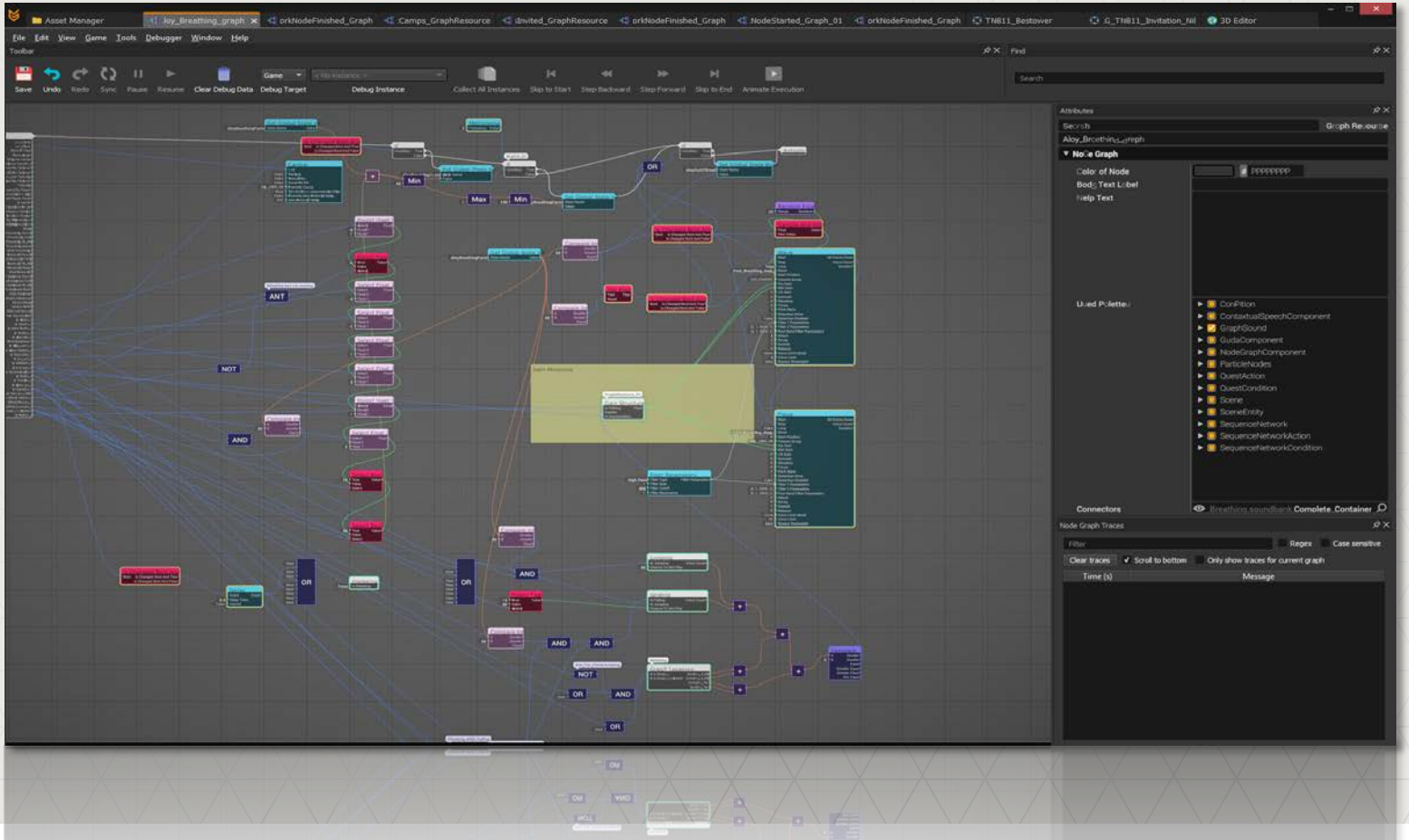
GameSync – The Bad

- 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

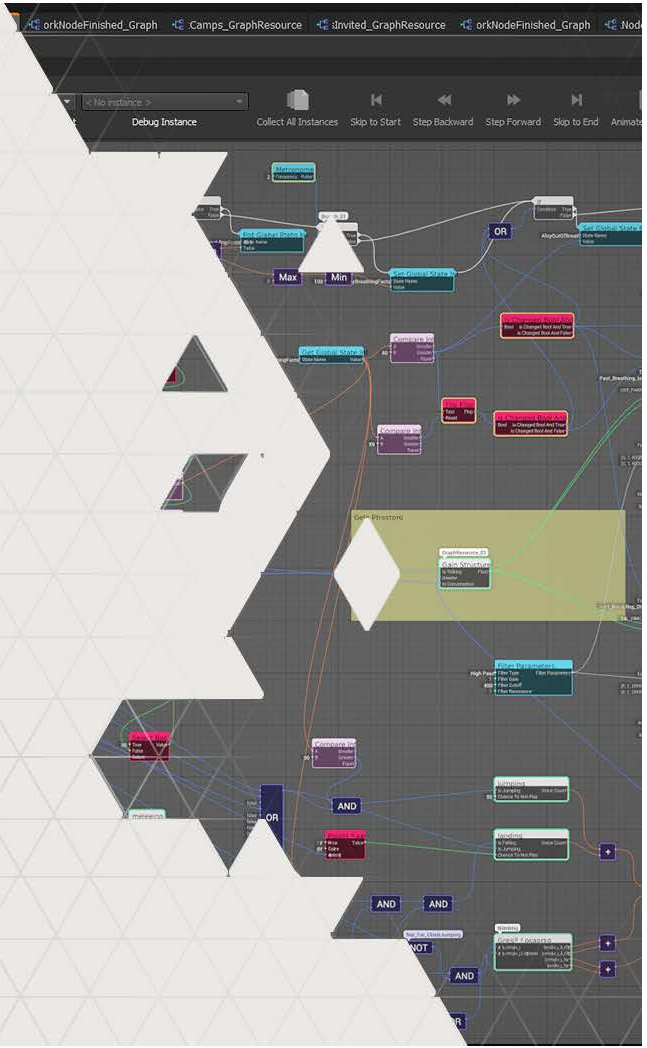
Other Tools

NodeGraph Debugger



NodeGraph Debugger

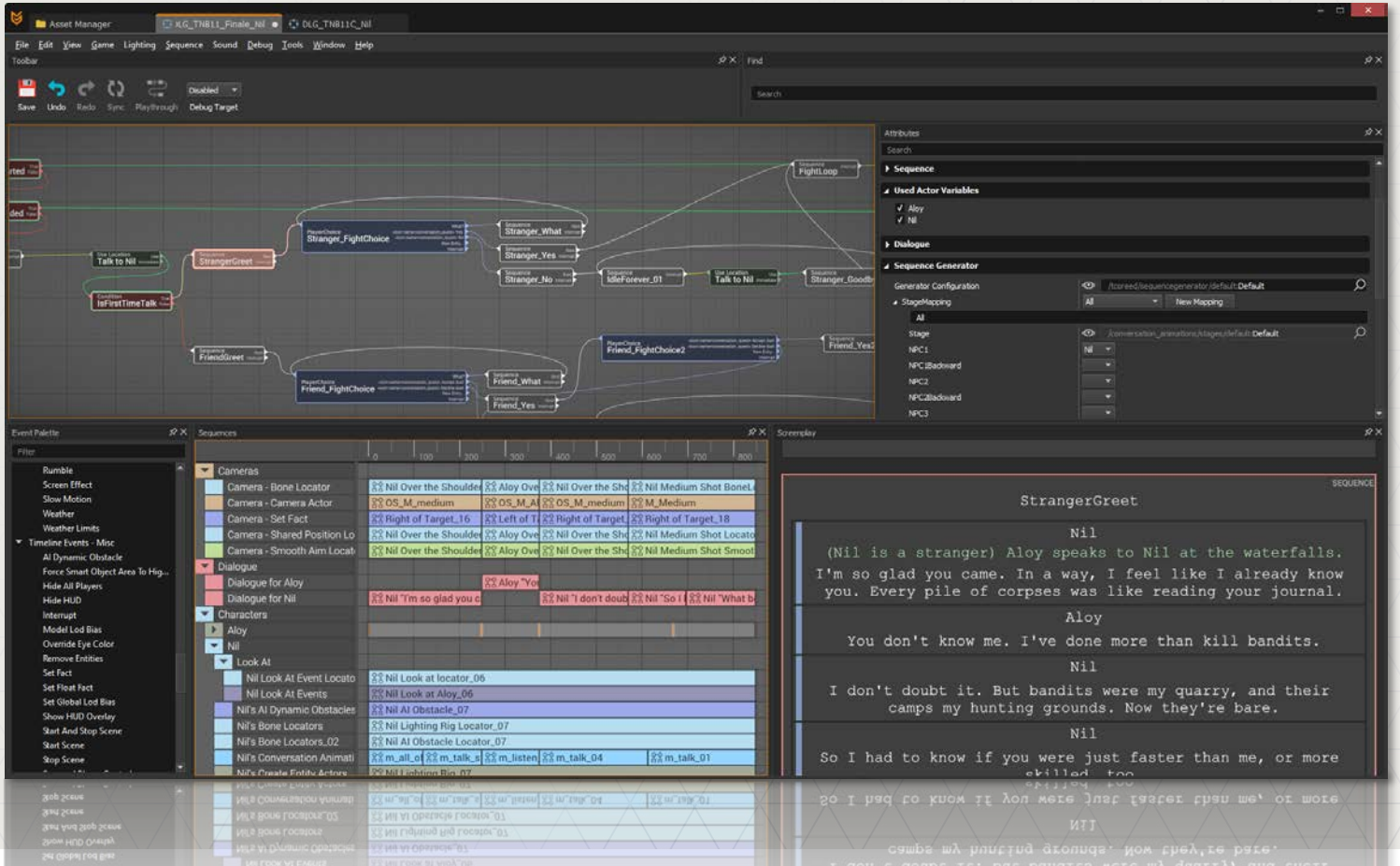
- Powerful debugging features
  - Variable inspection
  - Execution visualization
  - Historical debugging
- Minimal involvement from the tools team
  - Almost all of the work was done by game tech



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



Conversation Generator

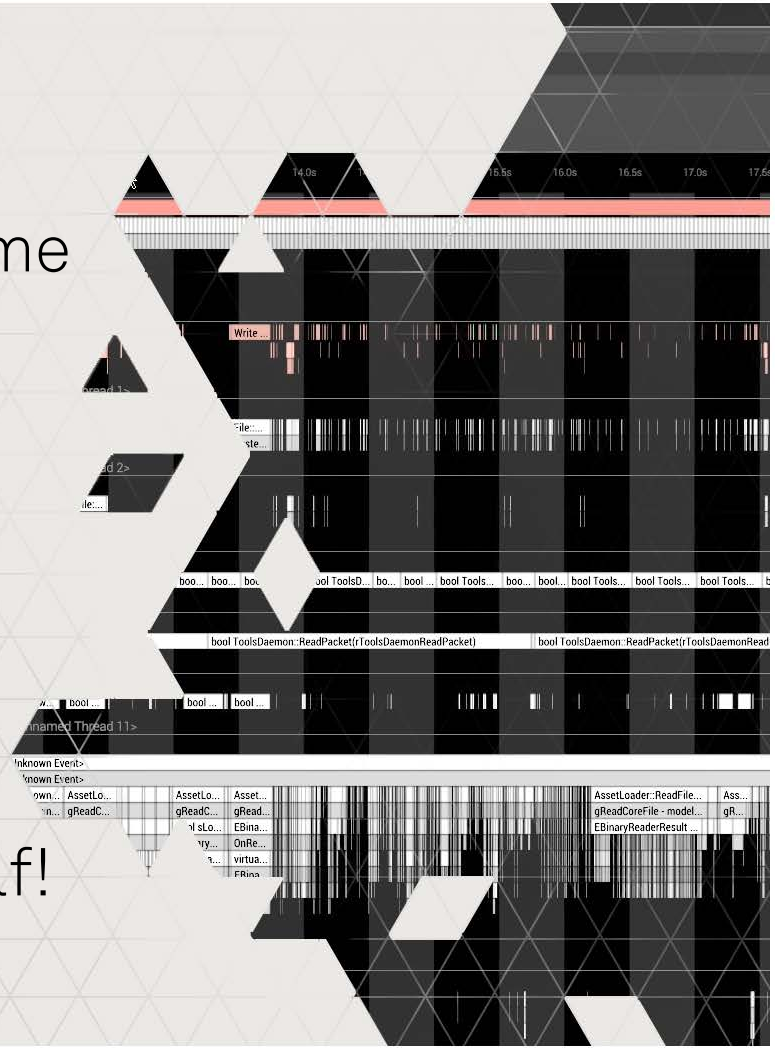
- This conversation was auto-generated
- Camera cuts, animation, gestures selected by rule-based system
- **RoboVoice** and lip-sync animations created for mock-up content

Conversation Generator

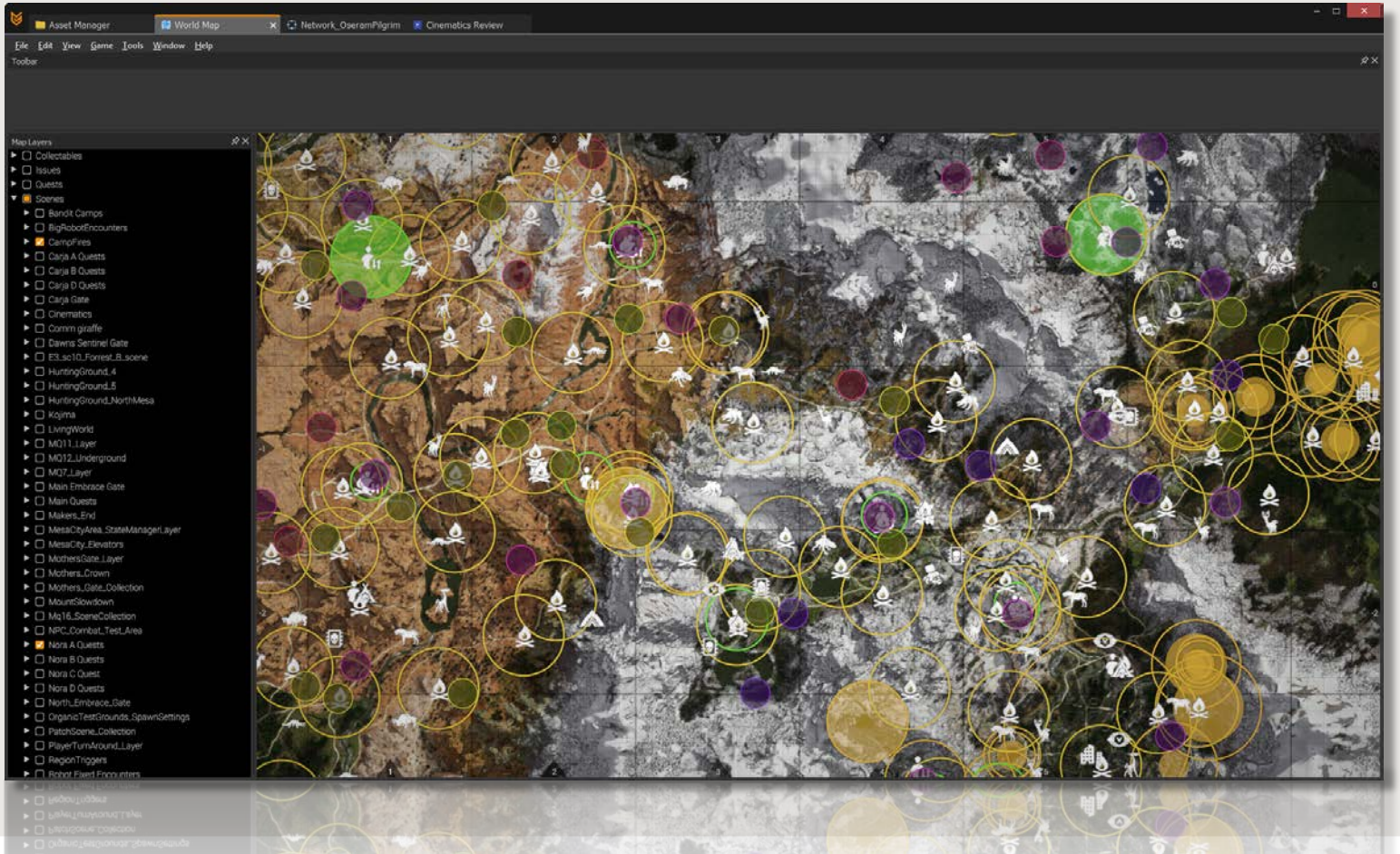
- Inspired by the work of **CD Projekt RED** and **BioWare**
- Was possible because it was easy to build in the new framework
- Behind the Scenes of Cinematic Dialogues in ‘The Witcher 3: Wild Hunt’
  - http://www.gdcvault.com/play/1023285/

Profiler

- 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!

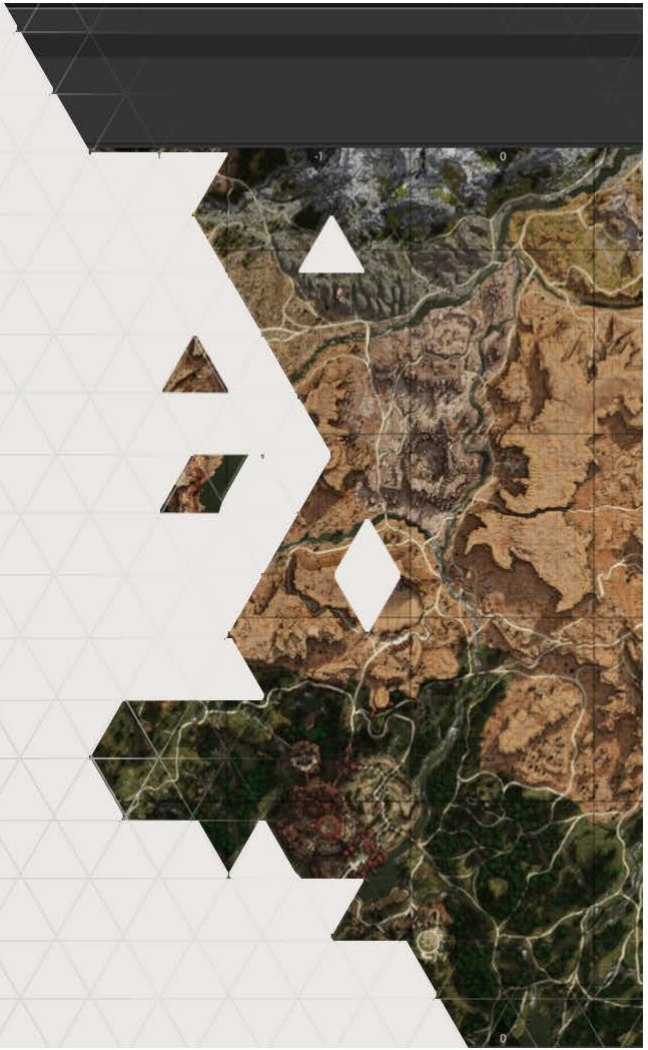


World Map

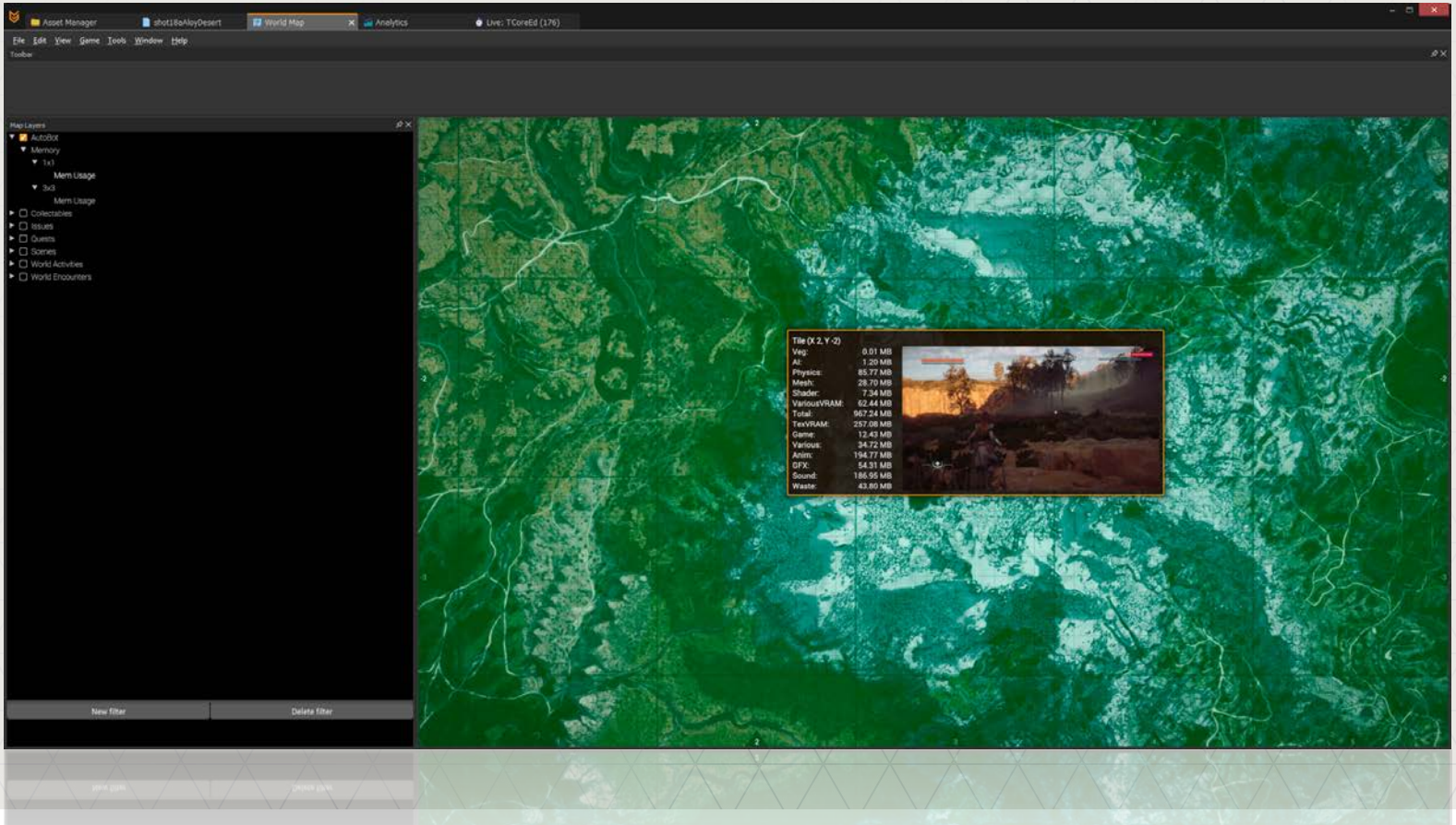

Asset Manager
World Map
Network_OseramPilgrim
Cinematics Review
File Edit View Game Tools Window Help
Toolbar
Map Layers
Collectables
Issues
Quests
Scenes
Bandit Camps
BigRobotEncounters
CampFires
Carja A Quests
Carja B Quests
Carja D Quests
Carja Gate
Cinematics
Comm giraffe
Dawns Sentinel Gate
E3_sc10_Forrest_B_scene
HuntingGround_4
HuntingGround_5
HuntingGround_NorthMesa
Kojima
LivingWorld
MQ11_Layer
MQ12_Underground
MQ7_Layer
Main Embrace Gate
Main Quests
Makers_End
MesaCityArea_StateManagerLayer
MesaCity_Elevators
MothersGate_Layer
Mothers_Crown
Mothers_Gate_Collection
MountSlowdown
Mq16_SceneCollection
NPC_Combat_Test_Area
Nora A Quests
Nora B Quests
Nora C Quest
Nora D Quests
North_Embrace_Gate
OrganicTestGrounds_SpawnSettings
PatchScene_Collection
PlayerTurnAround_Layer
RegionTriggers

World Map

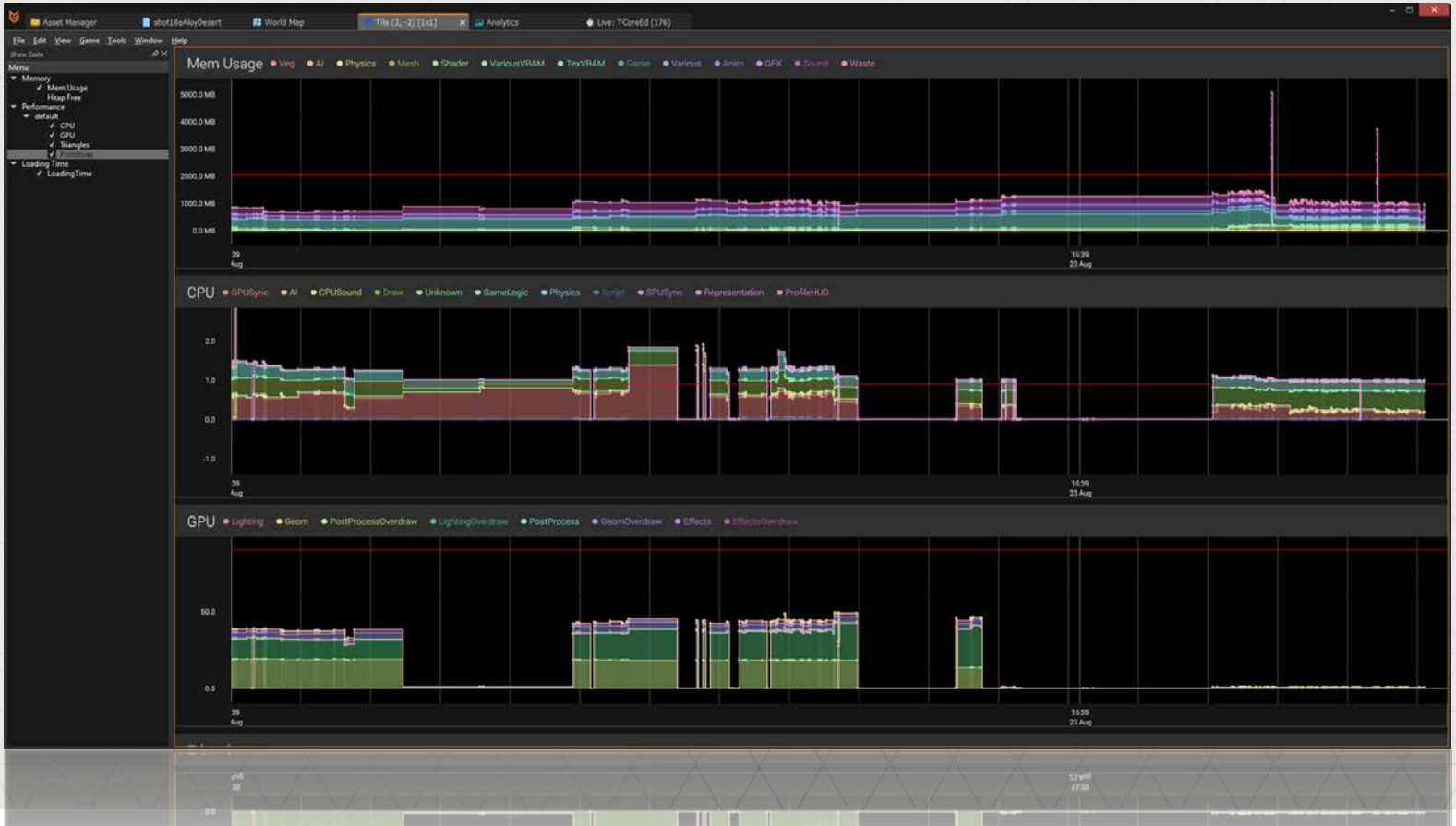
- Visualizes game features on a 2D world map
- Pluggable system for overlays
  - Quests
  - JIRA bugs
- Came online late in the project, and wasn't heavily used



World Map – Perf Report Overlay

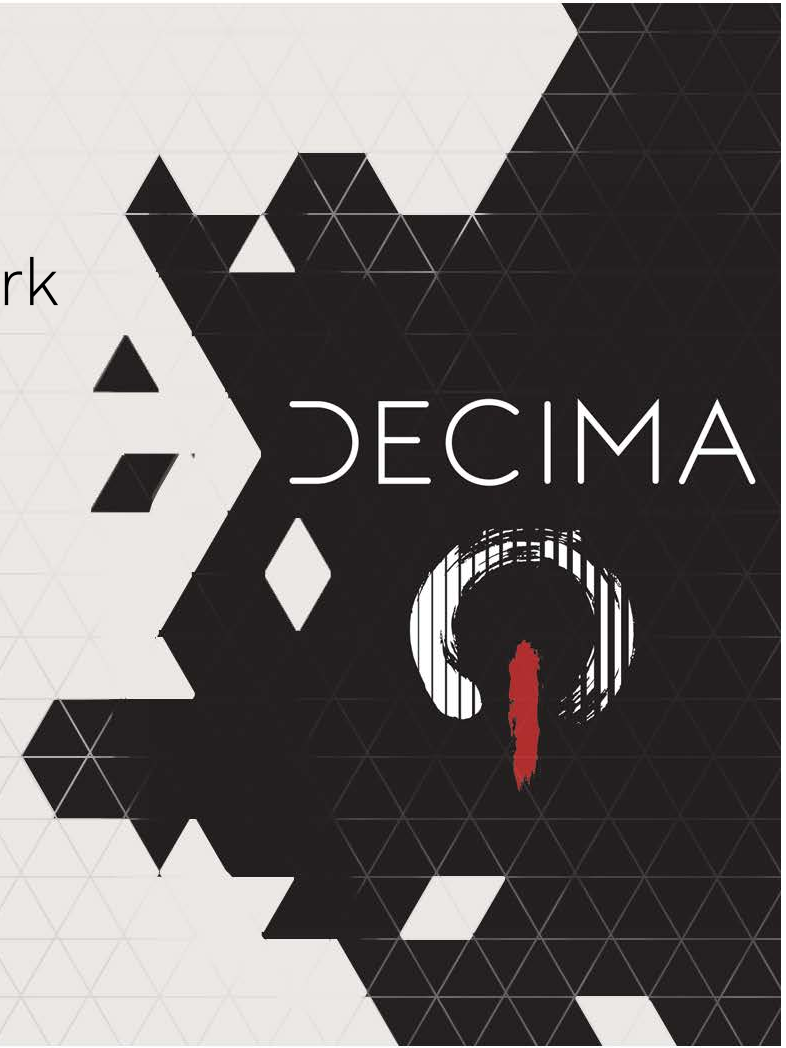


Perf Report


Asset Manager
shot18aAlloyDesert
World Map
Tile (2, -2) [1x1]
Analytics
Live: TCoreEd (176)
File Edit View Game Tools Window Help
Show Data
Menu
Memory
Mem Usage
Heap Free
Performance
default
CPU
GPU
Triangles
Loading Time
LoadingTime
Mem Usage
Veg
AI
Physics
Mesh
Shader
VariousVRAM
TexVRAM
Game
Various
Anim
GFX
Sound
Waste
5000.0 MB
4000.0 MB
3000.0 MB
2000.0 MB
1000.0 MB
0.0 MB
CPU
GPUSync
AI
CPUSound
Draw
Unknown
GameLogic
Physics
Script
SPUSync
Representation
ProfileHUD
2.0
1.0
0.0
-1.0
GPU
Lighting
Geom
PostProcessOverdraw
LightingOverdraw
PostProcess
GeomOverdraw
Effects
EffectsOverdraw
60.0
0.0
16:39
23 Aug

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

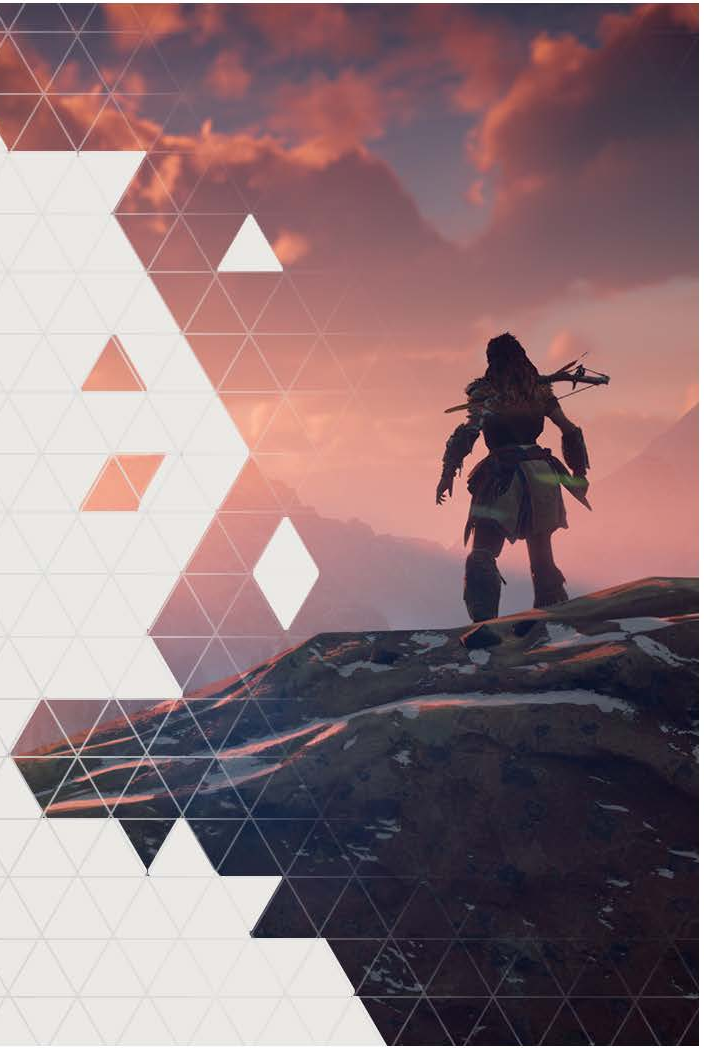


Decima API

```
# 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()
```

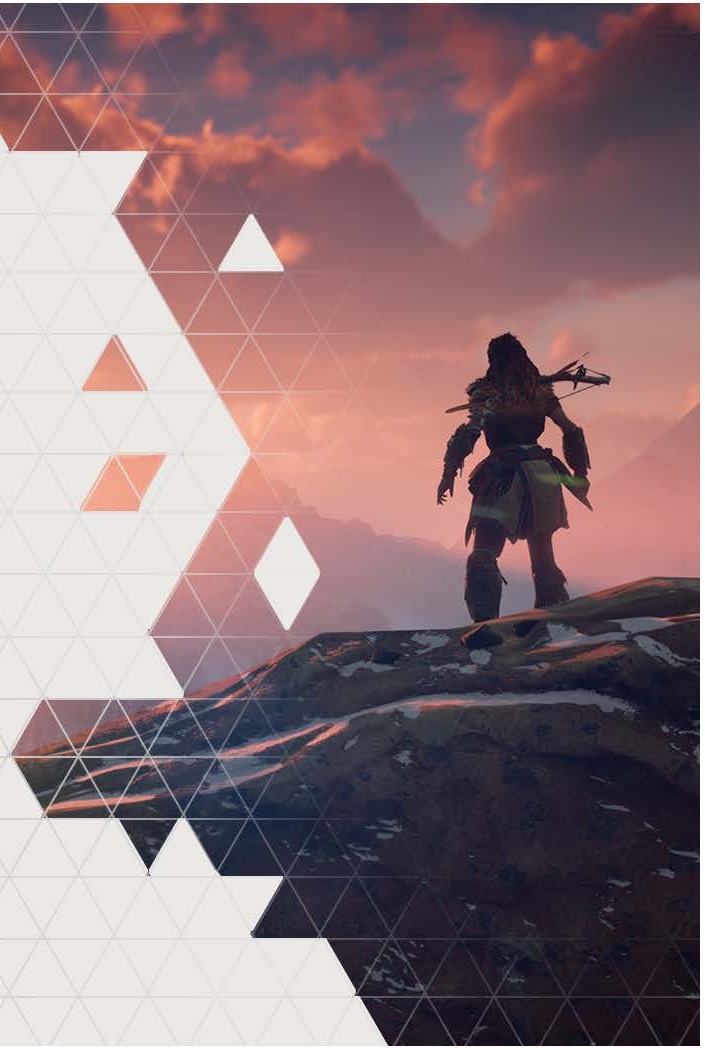
Future goals

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



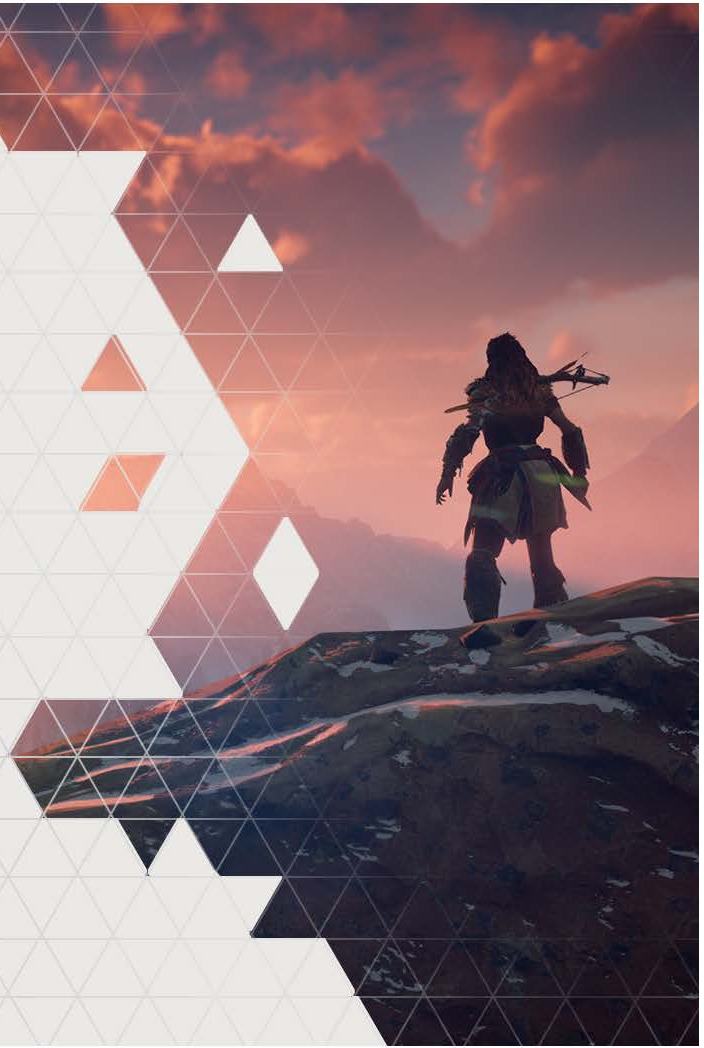
Future goals

- 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

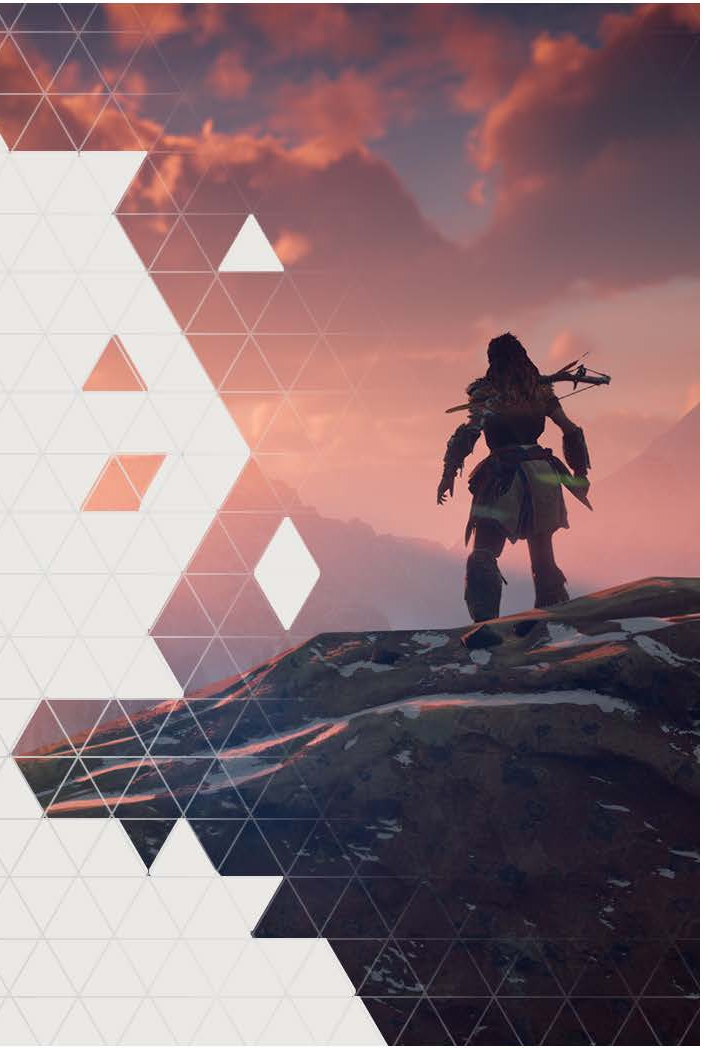


We’re Hiring!

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

Contact

- dan.sumaili@guerrilla-games.com
- sander.vandersteen@guerrilla-games.com



DECIMA
Questions?