
# Horizon and Beyond

## A look into Tomb Raider's Tools

Jason Yao (jyao@crystald.com)
Senior Tools Software Engineer
GDC 2013

CRYSTAL DYNAMICS SQUARE ENIX

# Horizon and Beyond:
# A Look into Tomb Raider's Tools

**Jason Yao**
Senior Tools Software Engineer
Crystal Dynamics

GAME DEVELOPERS CONFERENCE
SAN FRANCISCO, CA
MARCH 25-29, 2013
EXPO DATES: MARCH 27-29
2013

# Agenda

1. Introduction
2. Horizon Features
3. Technologies and Process
4. Pros and Cons
5. Summary


TOMB RAIDER

# About You?

# Horizon is…

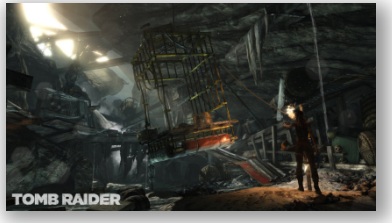
An editor for building arbitrary large worlds that support art & design workflow independence, object construction, multi-user and real-time editing.

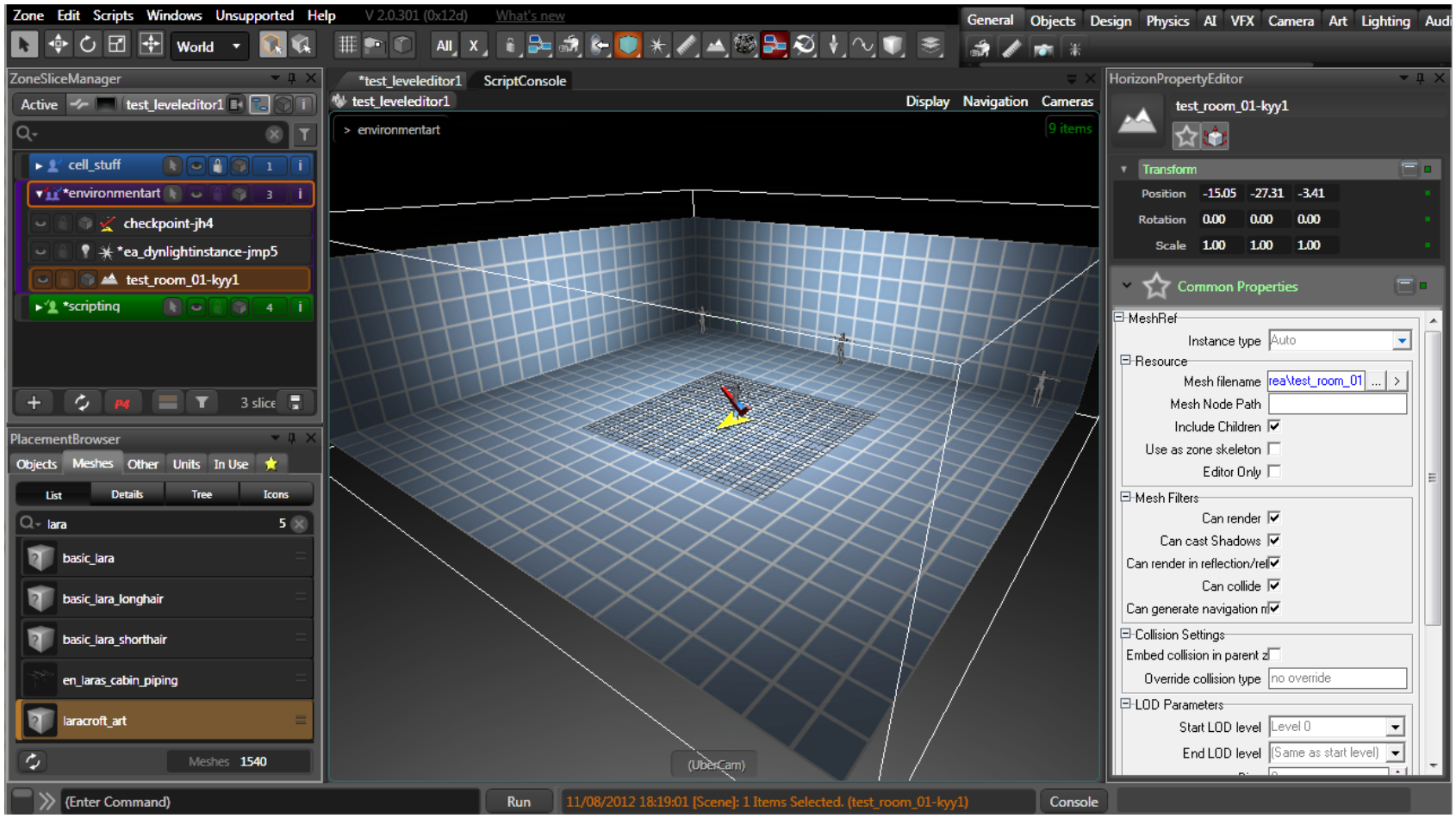
# Horizon is…

One of the tools in Foundation, our game platform.

Foundation allows us to build the most complex and largest levels yet at Crystal Dynamics




Zone Edit Scripts Windows Unsupported Help V 2.0.301 (0x12d) What's new
General Objects Design Physics AI VFX Camera Art Lighting
World
ZoneSliceManager
Active test_leveleditor1
cell_stuff
*environmentart
checkpoint-jh4
*ea_dynlightinstance-jmp5
test_room_01-kyy1
*scripting
PlacementBrowser
Objects Meshes Other Units In Use
List Details Tree Icons
lara
basic_lara
basic_lara_longhair
basic_lara_shorthair
en_laras_cabin_piping
laracroft_art
Meshes 1540
*test_leveleditor1 ScriptConsole
test_leveleditor1
Display Navigation Cameras
> environmentart
9 items
(UberCam)
HorizonPropertyEditor
test_room_01-kyy1
Transform
Position -15.05 -27.31 -3.41
Rotation 0.00 0.00 0.00
Scale 1.00 1.00 1.00
Common Properties
MeshRef
Instance type Auto
Resource
Mesh filename rea\test_room_01
Mesh Node Path
Include Children
Use as zone skeleton
Editor Only
Mesh Filters
Can render
Can cast Shadows
Can render in reflection/ref
Can collide
Can generate navigation m
Collision Settings
Embed collision in parent z
Override collision type no override
LOD Parameters
Start LOD level Level 0
End LOD level (Same as start level)
(Enter Command)
Run
11/08/2012 18:19:01 [Scene]: 1 Items Selected. (test_room_01-kyy1)
Console

# Horizon

- A World and Object Editor
- Began Four Years ago
- 4 to 7 developers
- Familiar features (to other 3D editors)
- Focus on Productivity and Iteration
- Sits on a feature rich, legacy win32 toolset

# Then and Now

## 2008

- C++ World Editor
- Win32 User Interface
- File based
- Mixed authoring workflow
- Multi-discipline editing
- Monolithic Workflow
- Despised World Editor

## Now

- C# World Editor
- Windows Presentation Foundation
- Database oriented
- Well defined Authoring Workflow
- Multi-user editing
- Modular Workflow
- Loved World Editor

# Agenda

1. Introduction
2. **Horizon Features**
3. Technologies and Process
4. Pros and Cons
5. Summary

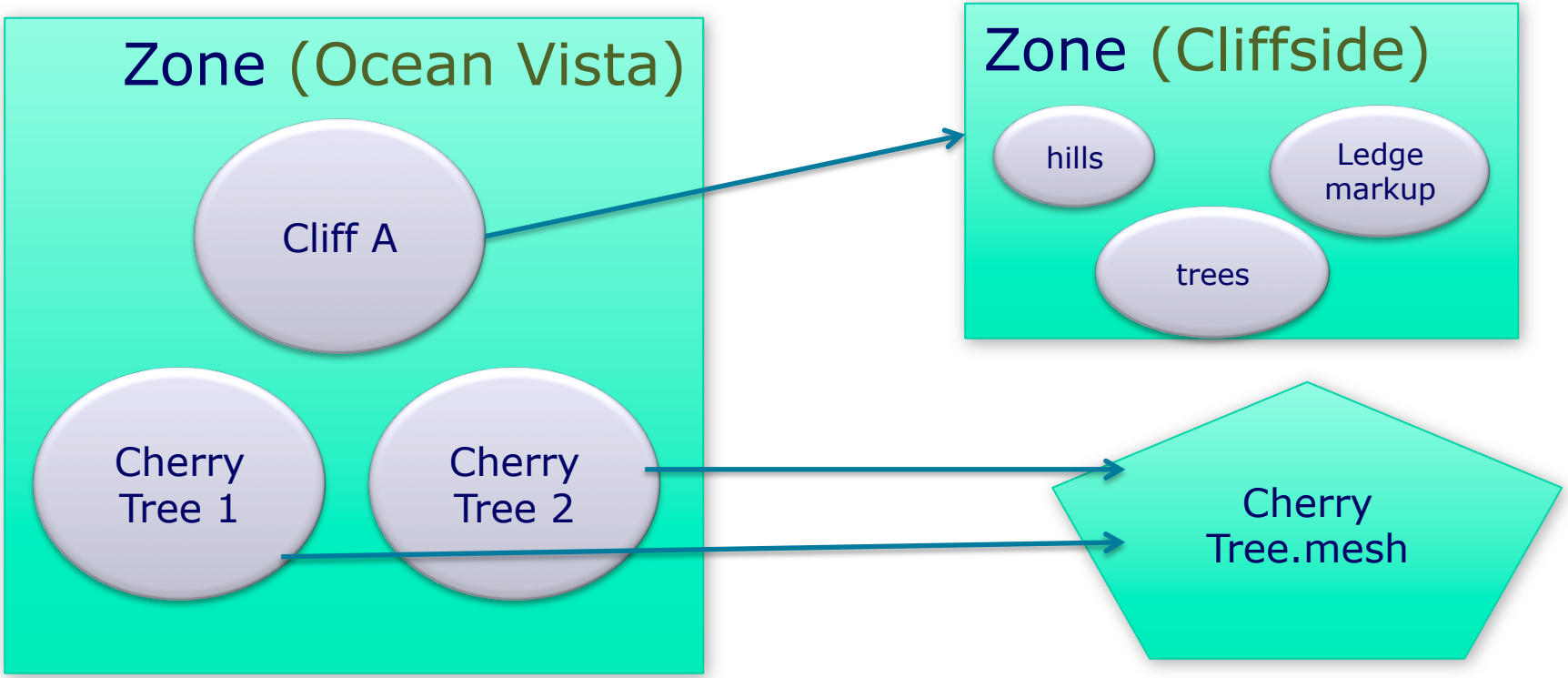
# The Basics

- A **Zone** is a container of placements
  - A Level is a **Zone**
  - An Object is a **Zone**
- We call the our placements **Zone Items**

- Horizon creates **Zones with Zone Items**

# The Basics Demo

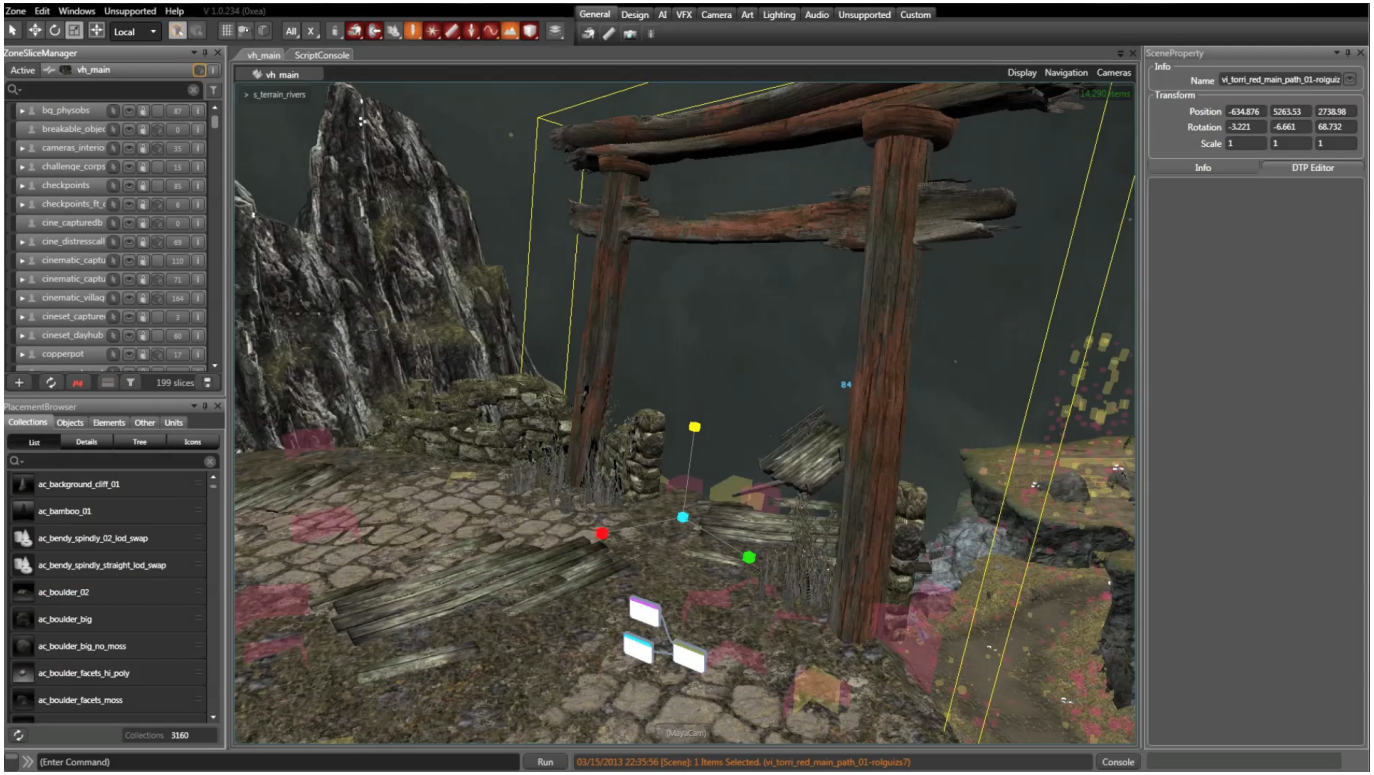
# Zones and Zone Items



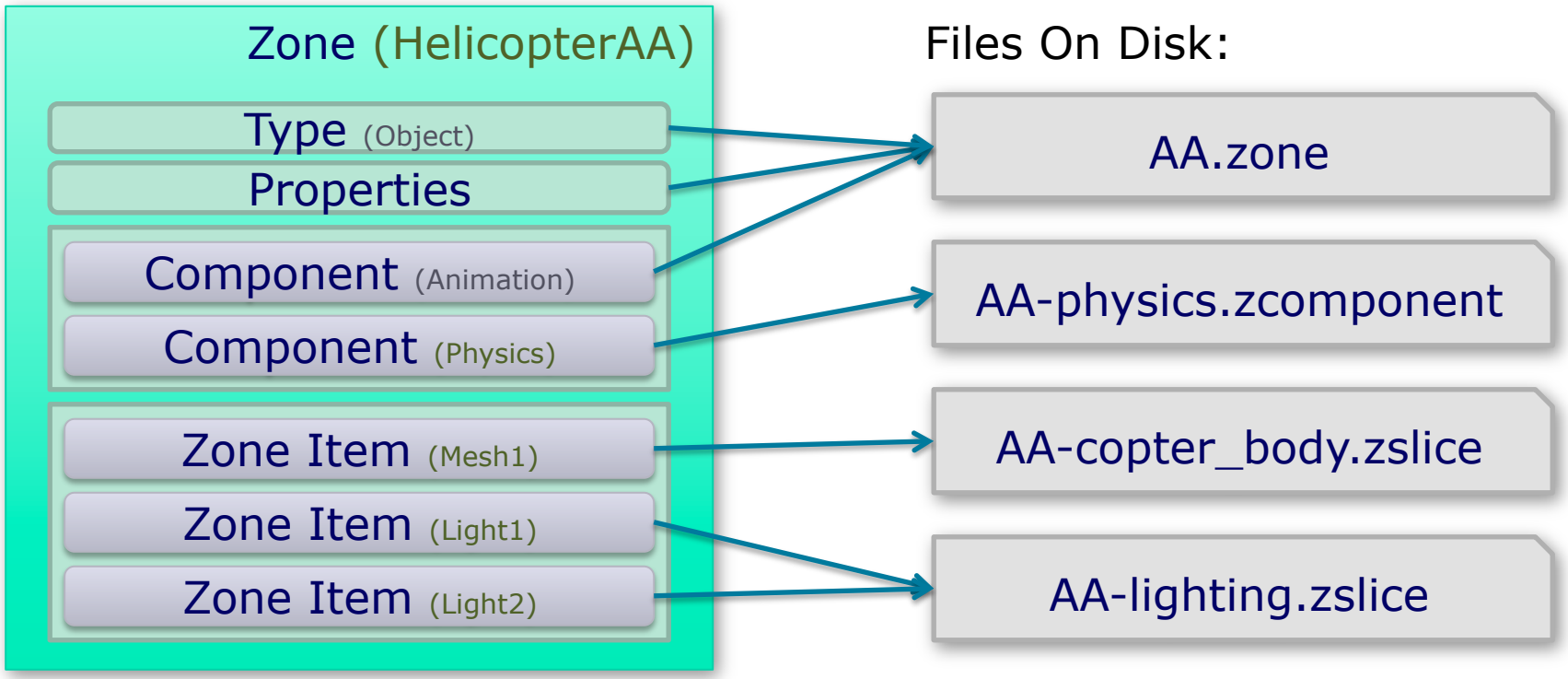
# Concurrent Authoring

- Many authors working in the same zone
- We use **Slices**
- Slicing up a zone by user and task
- An arbitrary number of slices
- Show user editing status

# Concurrent Authoring Example



# Storage for Concurrent Authoring



# Modular Construction

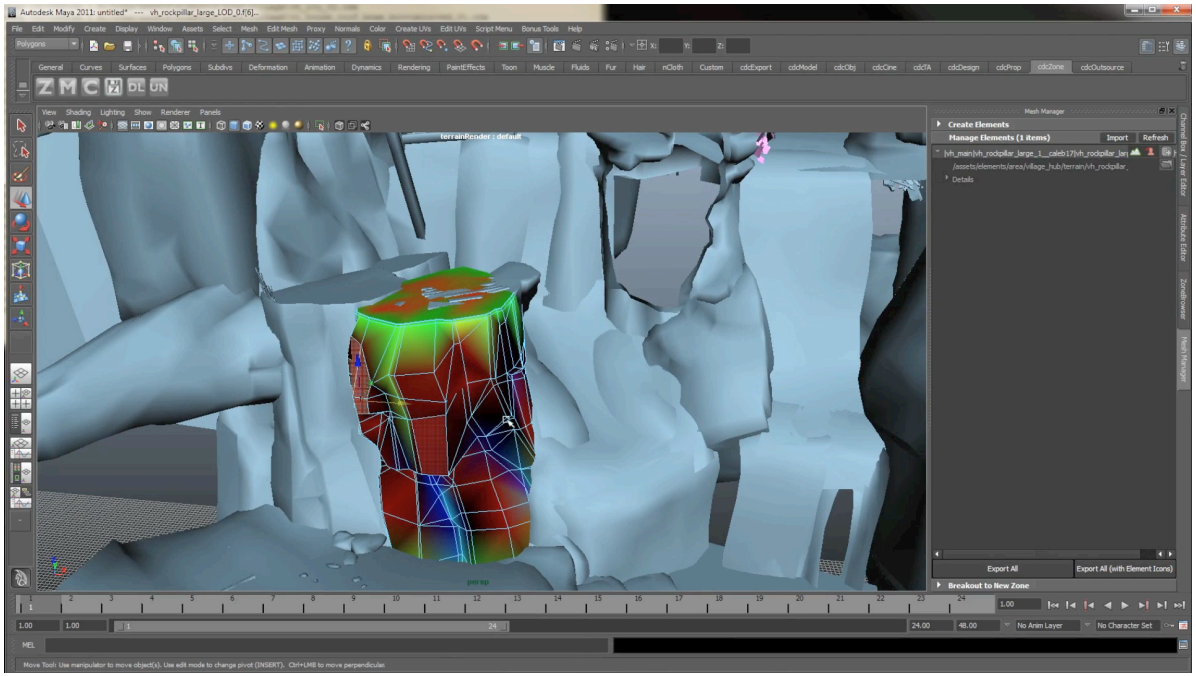
- Composing Objects
- Breaking apart Objects
- Swapping and Replacing
- Editing In Place
- Fork and Edit In Place

# Modular Construction Demo

# Maya: Meshes in Context

- Maya for Meshes
- Horizon for Level Design
- Edit meshes in the Context of a Zone
- Two-way synchronization
  - Horizon and Maya
- Initial Breakout to a Zone

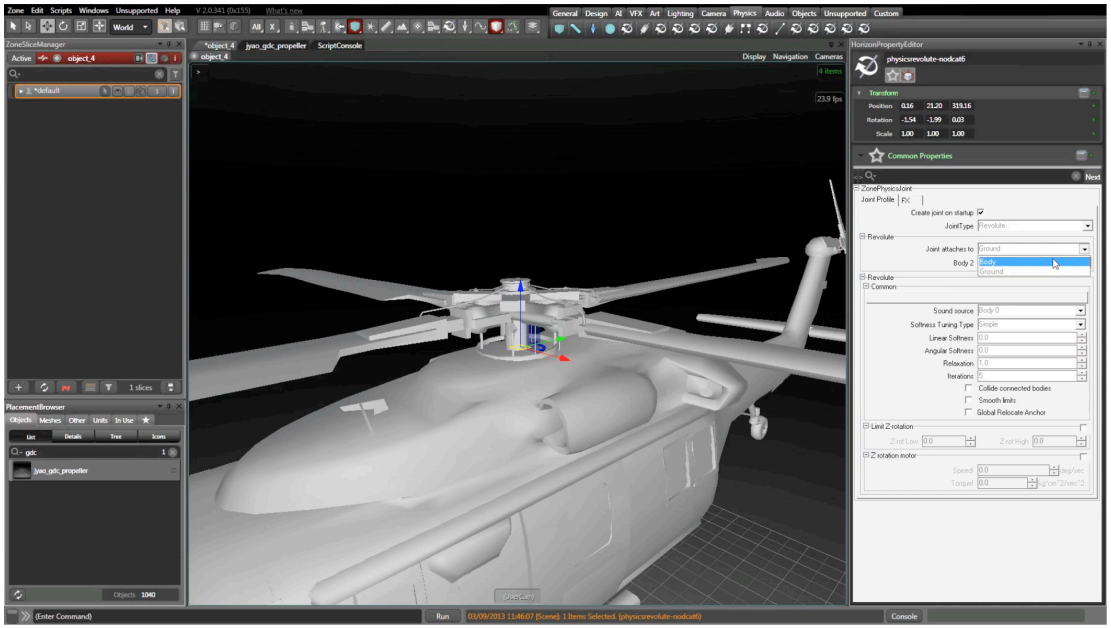
# Maya Mesh in Context Example



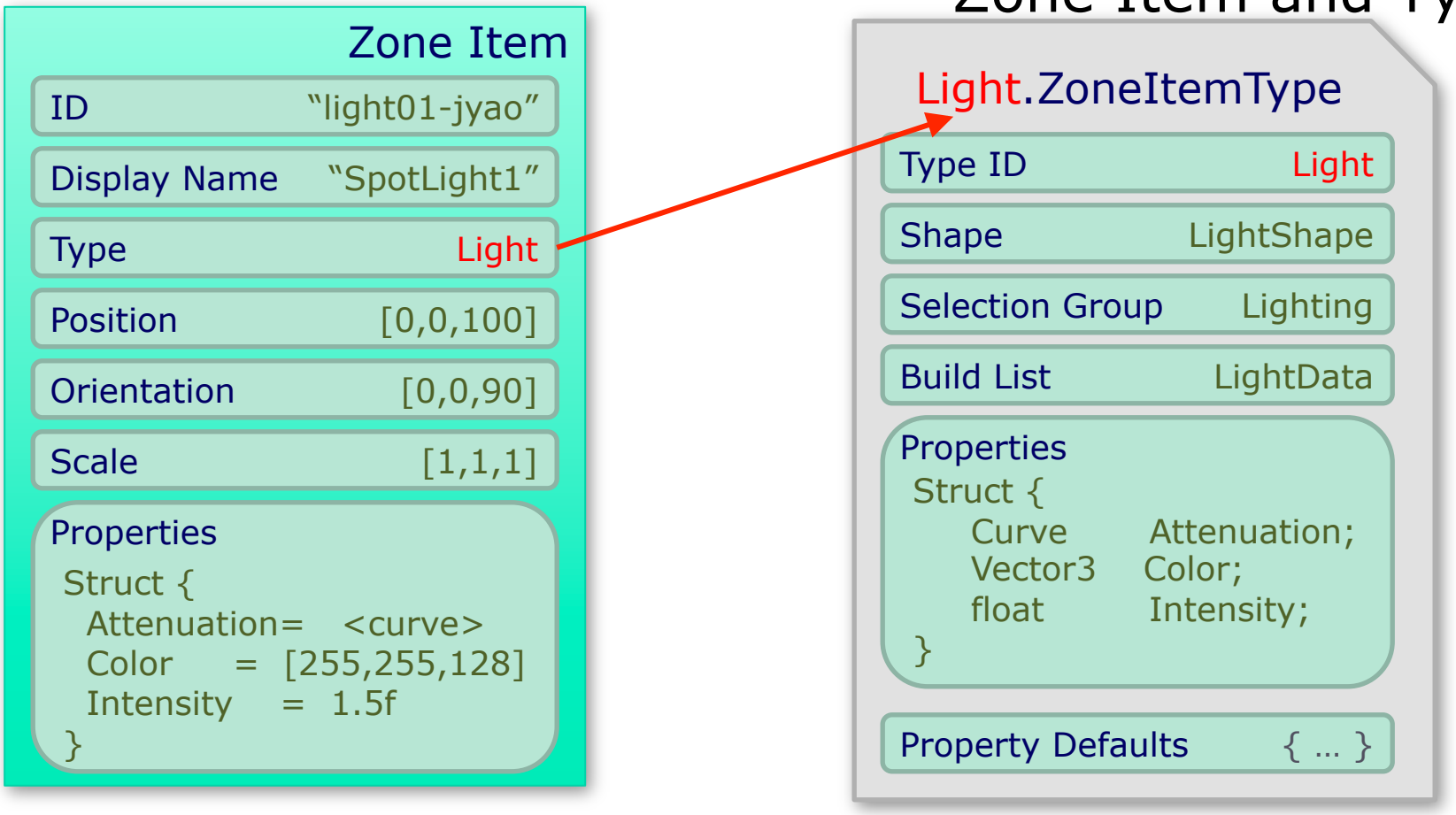
# Building an Object

- Same pattern as Level Creation
- Adding Behavior
- Adding Collision
- Adding Physics Joints
- Placing objects in a level

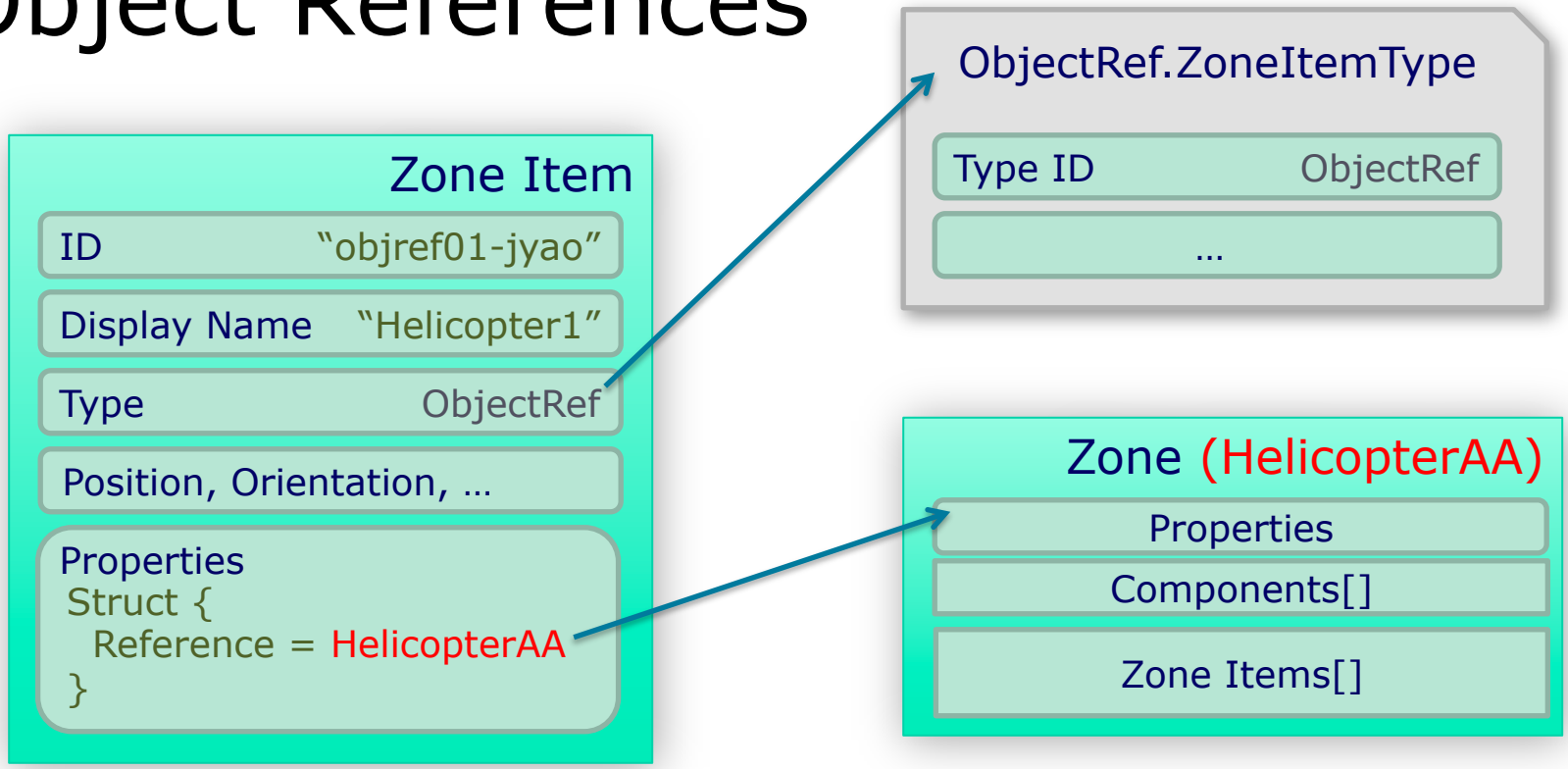
# Building an Object Demo



# Zone Item and Type



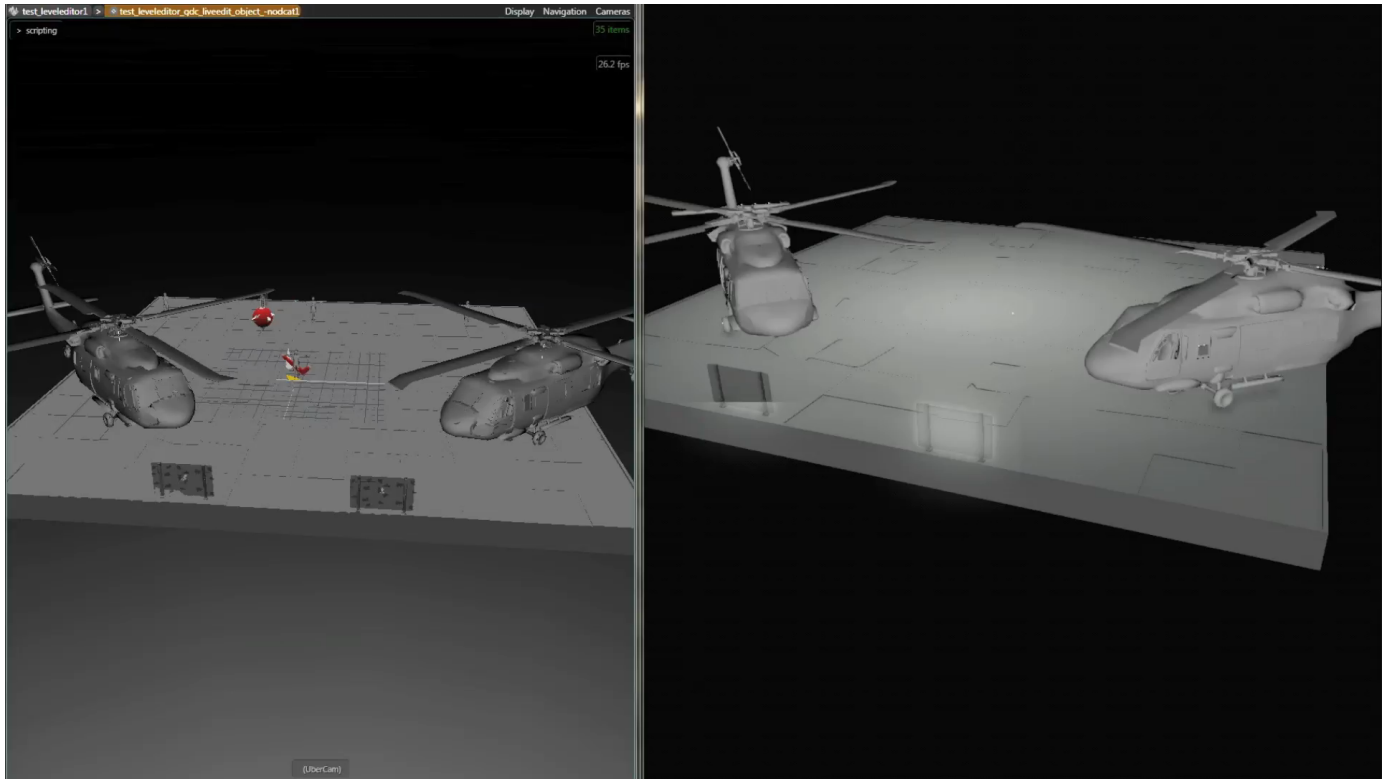
# Object References



# Live Editing in Game

- For Placements and Properties
- Our Fastest Iteration Loop

# Live Edit Horizon/Game Demo



# Other Features

- Integrated Backup
  - Never lose more than 5 seconds
- Script Console
- Dynamic Lighting Workflow

# Agenda

1. Introduction
2. Horizon Features
3. **Technologies and Process**
4. Pros and Cons
5. Summary

# Strategy

- Focus on Quality over Quantity
- Minimize jumping between authoring tools
- Solve for a single game team
- Balance long and short term architecture goals
- Keep what works

# Technical Design Requirements

- Assert Often and Crash Early
- Minimal, Well defined interface between C++, C#
- Data is unique, statically identifiable
- No arbitrary maximum size for input data
- Exceed game runtime capabilities
- Preserve position, rotation, scale and skew with transform wrapper

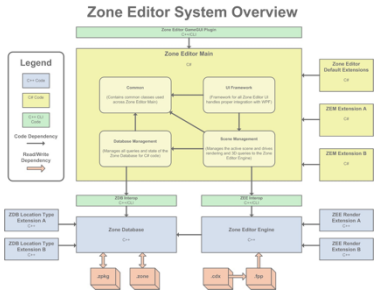
# How did we do?

- ✓ Assert Often and Crash Early
- ✓ Minimal, Well defined interface between C++, C#
- ✓ Data is unique, statically identifiable
- ○ No arbitrary maximum size for input data [PARTIALLY]
- ❖ Exceed game runtime capabilities [NOT REALLY]
- ✓ Preserve position, rotation, scale and skew with transform wrapper

# Focus on User Experience

- User Experience Director
- Reduce mouse clicks
- Minimize learning curve
- Be familiar and consistent
- Intuitive design, less documentation
- Gain grassroots support
- Improve perception and expectations
- Rapid “polish” feature turnaround

# High-level Architecture




- Collaborative, simple UML Design
- Assign Owner to each system.

Horizon (Front End)

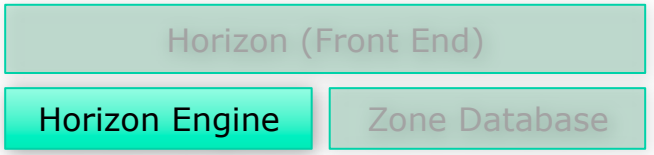
Horizon Engine

Zone Database

# Horizon (Front End)

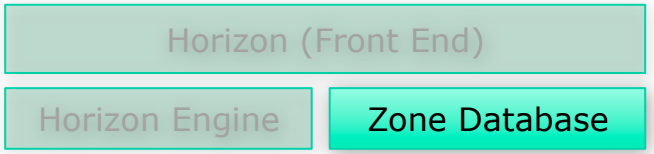

- User Interface Management
- Data Editing
- Scene Navigation and Viewport
- Pluggable architecture using reflection

# Horizon Engine



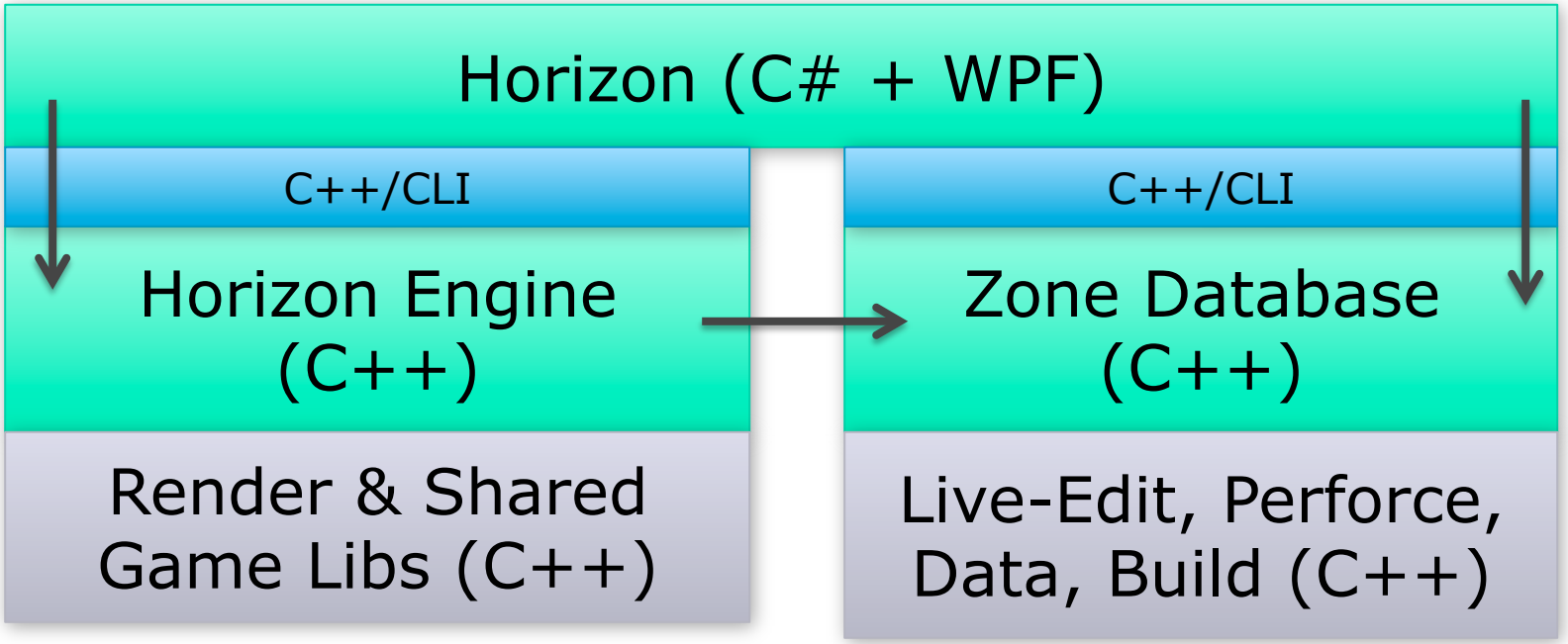

- Powers the 3D Viewport
- Manages Rendering Scenes
- Manages Collision Scenes
- Draw List Interface
- Texture, Material and Mesh Cooking
- Access to Shared Runtime Libraries

# Zone Database




- Centric Asset Interface and File System
- Management of ZDB Objects
- Metadata system
- Placements & Assets are uniquely identifiable by Database ID
- Live-Edit Connection
- Enables concurrent authoring

# High-level Architecture 2



# Why WPF?

- Faster iteration
- Advanced and pretty UI creation behavior
- Data model management separation
- MS Expression Blends Editor
- Larger building blocks (WPF3 and .NET 3.5)

# Why not a true Database?

- Originally, built on a File-based engine
- Need to solve database version control
- Heading towards a True Database

# Ready for some Code?

# Static Interface Pattern (C++)

```
class IZone : IZDBObject
{
  // Static access interface over Singleton pattern
  static IZone*         Get(ZoneID zid);
  static bool           Exists(ZoneID zid);
  …
  static bool           Create(ZoneID zid);
  static bool           Copy(ZoneID old, ZoneID new);
  …
  // Still contains Member variables
  IStructField*         Add/GetComponent(ComponentID cid);
  IStructField*         GetProperties();
};
```

# Changing a Mesh Property (C#)

```
ZoneItemID id = new ZoneItemID("treemesh01-jyao");
ZoneItem zi = ZoneItem.Get(id);

string path = zi.Properties.
                GetStringValue("meshref", null /*default*/);
// RESULT: path == "smalltree.mesh"

zi.Properties.SetStringValue("meshref", "bigTree.mesh");
// Changed to "bigTree.mesh"
```

# Notification By ID (C#)

```
ZIEvents ziEvents = ZoneItem.Events(zItemID);
// Register for data (property) changes
// Zone Item does not have to exist or be loaded.
ziEvents.PostModify += OnPostModifiedZoneItem;
…
// Listen for Event
void OnPostModifiedZoneItem(object sender, ModifyArgs args)
{
  if (IsPathAffected(args, "m_lightData.attenuation"))
       … // do processing.
}
```

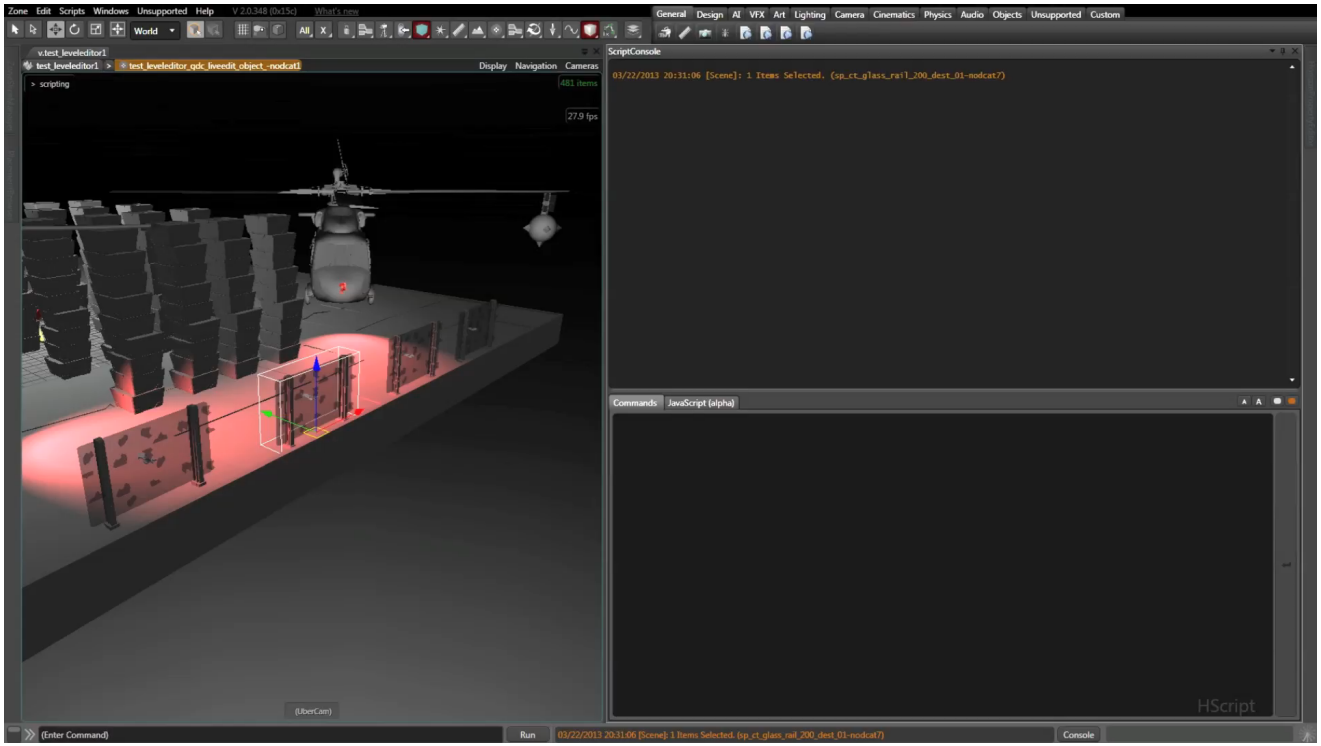
# Using Drawing Contexts (C#)

```
using(var shapeList =new ShapeContext(hemi, compileWhenDone))
{ // Draw a Snow Cone!!
  shapeList.IsCollidable = true;
  shapeList.CollisionID = CollideID("MySnowCone");
  shapeList.SetColor( rgba: [1,1,1,0.5] );
  shapeList.AddSphere( center: [0,0,0],  radius: 10 );

  shapeList.SetColor( rgba: [0,1,0,1] );
  shapeList.AddCone(tip:[10,10,0], base:[0,0,0], radius: 20);
} // Draw list compiled and posted to Horizon Engine
```

# C#: Commands via Reflection

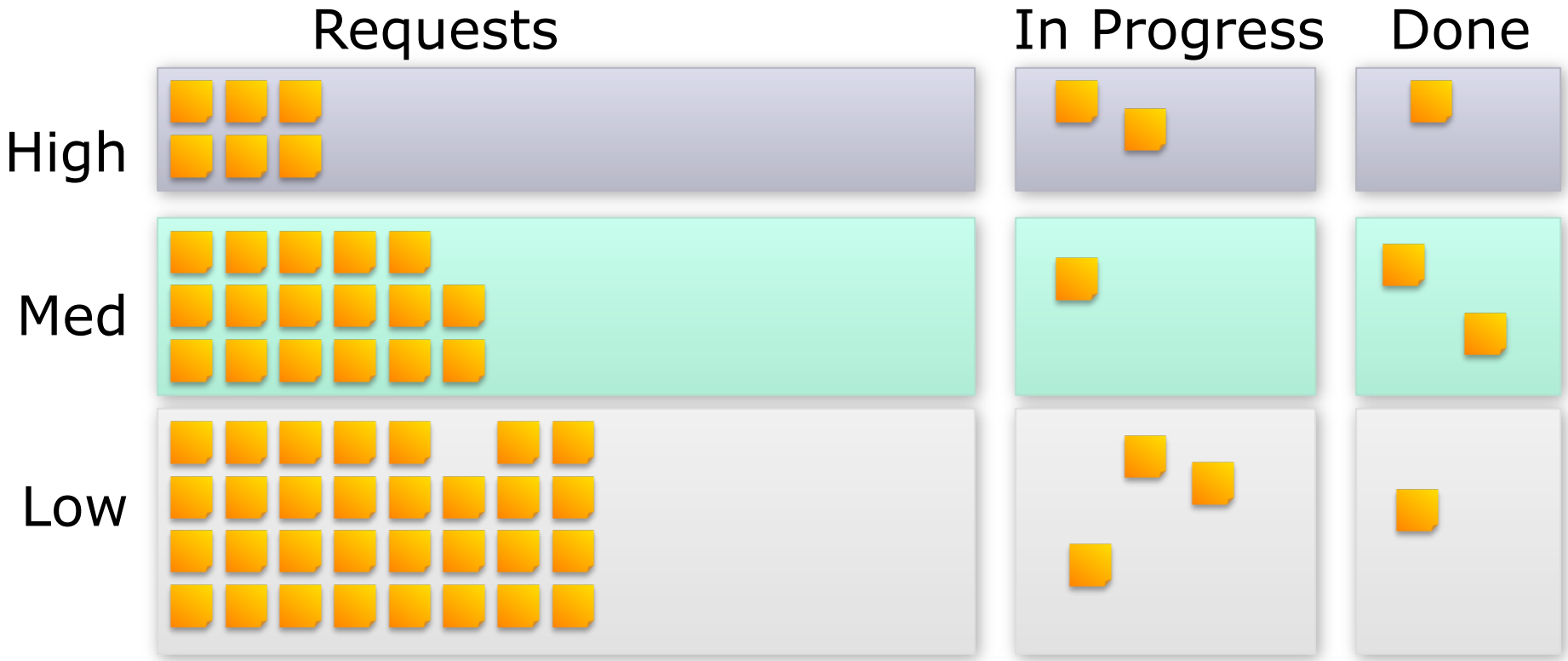
```
// Easy Command Declaration
[Command("Change the position of an item",
         Param0="Item to change",
         Param1="Vector Position")]
public static void SetPosition(ZoneItemID id, Vector3 pos)  {…}


[Command("Duplicates an item", Param0="Item to change")]
public static void Duplicate(ZoneItemID id)  {…}
```

# Scripting Demo



# A bit more on Process

# Polish. The Wall of Tasks



# Task Card Example

$$

Can I have discrete a rotation gizmo?

Request by Morris O.

# The Wall of Tasks Rules

- Assign a $, $$, $$$ cost
- Physical limit to each bucket
- The Team owns & self-prioritizes the cards
- 80/20 time split for Scheduled/Wall tasks
- Broadcast the completion of cards

# Agenda

1. Introduction
2. Horizon Features
3. Technologies and Process
4. **Pros and Cons**
5. Summary

# Modular Construction Takeaway

**Pros**

- Rapid construction and reuse
- Large set of ready pieces
- Fast designer blockout
- Good for outsourcing

**Cons**

- Effort to look “organic”
- Broad affect from content changes
- Wholistic optimization

# Modular Construction Verdict

- We will continue to use and improve
  - Asset locking by level
  - Tools that show how changes propagate

# Concurrent Authoring Takeaway

**Pros**

- Slices allow
  - flexibility
  - better coordination
- Whole team able to edit on a single level

**Cons**

- 100+ slice levels
- Slices became the central tool for level organization

# Concurrent Authoring Verdict

- Need a Layer System
  - Hierarchical organization
  - Independent from slices and file storage
- Need ability to “checkout” per placement
  - Prerequisite: full database backend

# Outsourcing Challenges Takeaway

**Pros**

- Allowed game team to focus on highest priorities
- We created a lot of content

**Cons**

- Manual merging was a nightmare
- Heroic efforts from Outsourcing TA
- We did not galvanize our workflows soon enough

# Outsourcing Challenges Verdict

- Outsourcing & Insourcing
  - Still very important for content generation
- We need more automation
- More dedicated outsourcing tools
- Clear definition of workflows

# Agenda

1. Introduction
2. Horizon Features
3. Technologies and Process
4. Pros and Cons
5. **Summary**

# Summary

- Understand your scope and constraints
- Game team driven features
- Gain grass roots support from game team
- Balance between generalization and specialization
- Optimize sooner than later
- Importance of branding

# Future Work

- Game viewport via TCP/IP
  - Game exposes draw list interface
- Usage analytics
- Authoring time cost system
- Timeline sequencing

# The Team

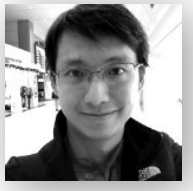
Lead Architect
- John Pursey

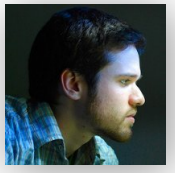
Senior Engineers
- Jason Yao
- Joel Hunter
- Ife Olowe
- Tim Pease

User Experience Director
- Joe Stinchcomb

Engine Director
- Gary Snethen





# Thanks! and Questions?

Special Thanks to:

Gary Snethen
Jason Lacroix
John Pursey
Joe Stinchcomb
Julien Merceron
The TR TEAM!!!!

**Other Crystal Talks**

Light Based Rendering in TR (Jason Lacroix)
Reinvention of Tomb Raider (Darrell Gallagher)
Croft of System Design (Jonathan Hamel)
Emotionally Engaging Cameras (Remi Lacoste)

Contact:
Jason Yao (jyao@crystald.com)

- Please fill out survey