An Entity System for Unity: 2015-1

Who am I?

- Former game developer + CTO
- Building Entity Systems since 2002
- Writing about Entity Systems since 2007
- Now teaching children to program (Ages 6-16), and training teachers

I

❤

Editor Customization

I

❤

[bitch on Twitter about]

Editor Customization

Unity Editor is still a

Black Art

WARNING!

WARNING!

“Here Be Generalizations”

WARNING!

“Here Be Generalizations”

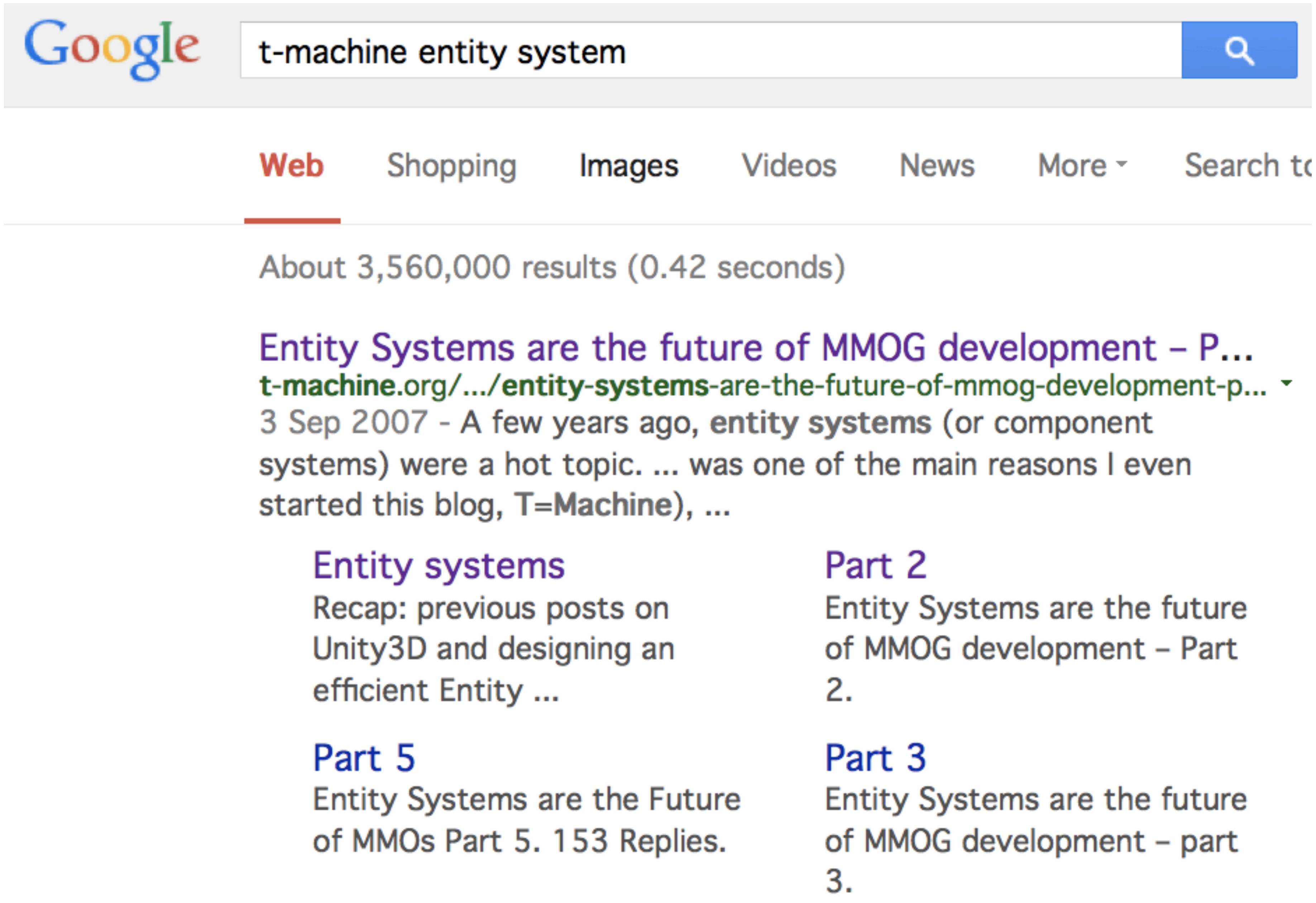
(only a short talk, lots to cover)

Today ...
... Entity Systems

Why use an Entity System?

- 1. Quicker prototyping (games, features)
- 2. Faster game (CPU, RAM, Mobile)
- 3. Easier code (debug less, design more)

What is an Entity System?



t-machine entity system

Web Shopping Images Videos News More Search tc

About 3,560,000 results (0.42 seconds)

Entity Systems are the future of MMOG development – P...
t-machine.org/.../entity-systems-are-the-future-of-mmog-development-p...
3 Sep 2007 - A few years ago, **entity systems** (or component systems) were a hot topic. ... was one of the main reasons I even started this blog, T=**Machine**), ...

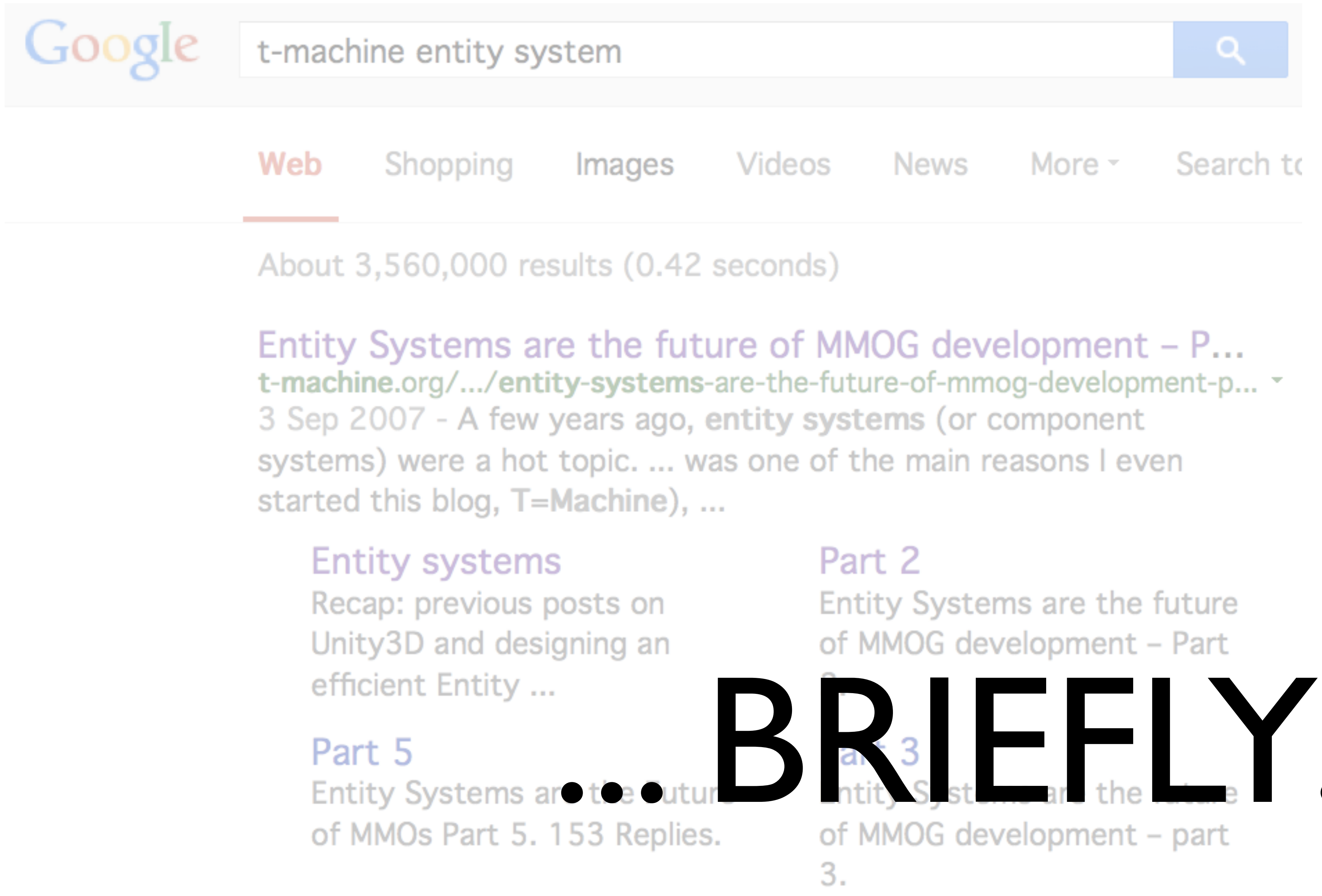
Entity systems
Recap: previous posts on Unity3D and designing an efficient Entity ...

Part 2
Entity Systems are the future of MMOG development – Part 2.

Part 5
Entity Systems are the Future of MMOs Part 5. 153 Replies.

Part 3
Entity Systems are the future of MMOG development – part 3.

What is an Entity System?



About 3,560,000 results (0.42 seconds)

Entity Systems are the future of MMOG development – P...
t-machine.org/.../entity-systems-are-the-future-of-mmog-development-p...
3 Sep 2007 - A few years ago, entity systems (or component systems) were a hot topic. ... was one of the main reasons I even started this blog, T=Machine), ...

Entity systems
Recap: previous posts on Unity3D and designing an efficient Entity ...

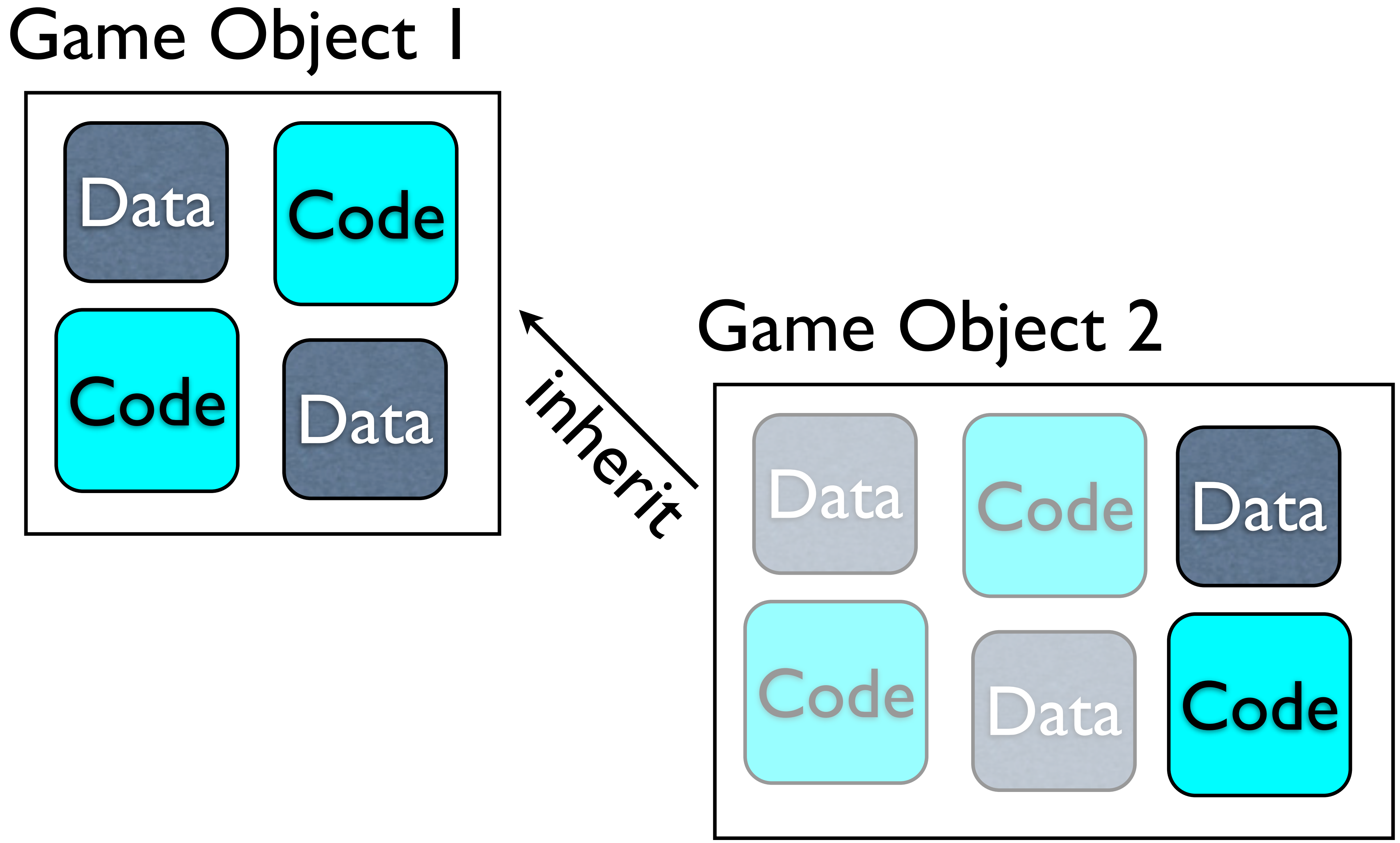
Part 2
Entity Systems are the future of MMOG development – Part 2.

Part 5
Entity Systems are the future of MMOs Part 5. 153 Replies.

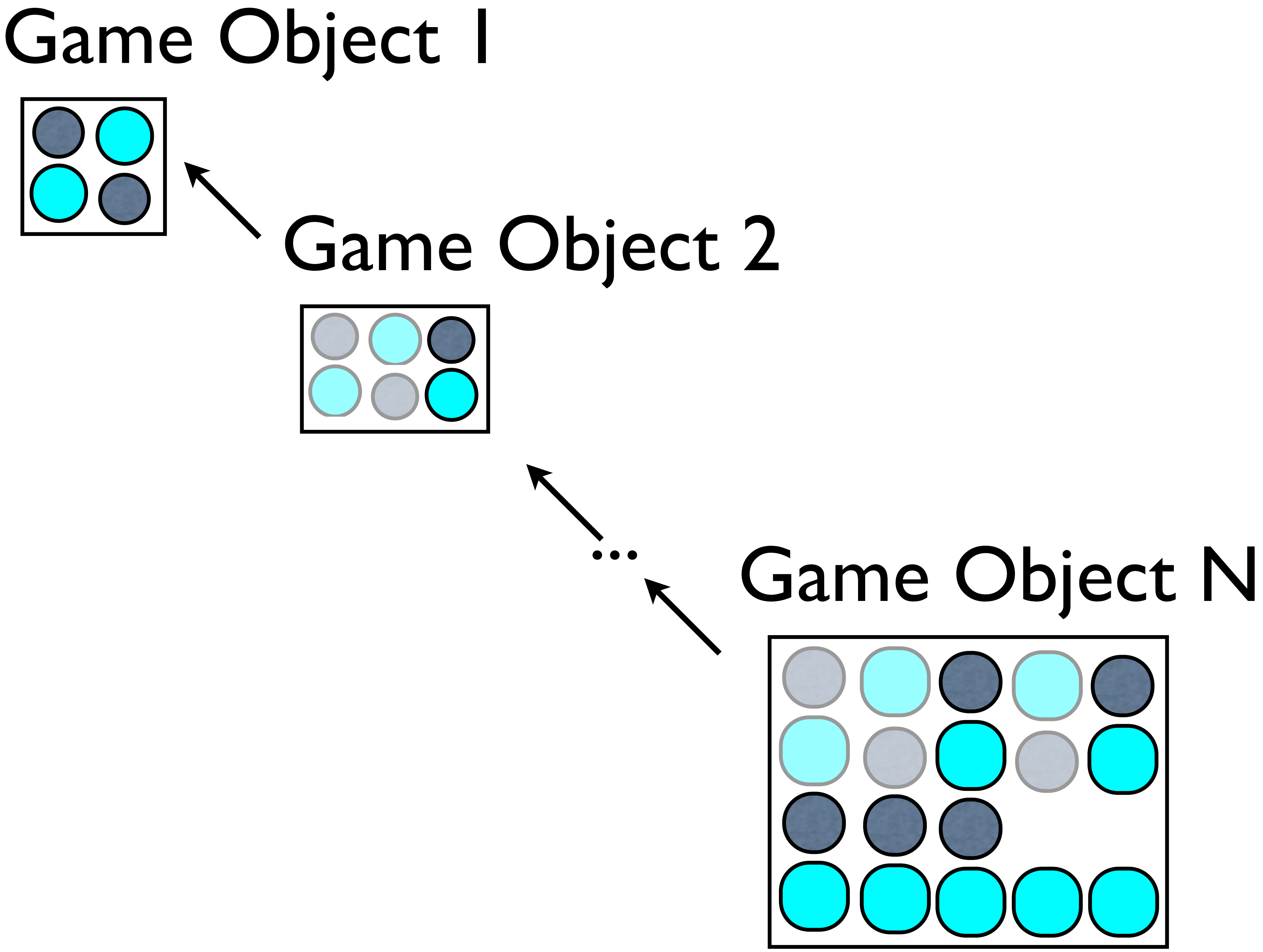
Part 3
Entity Systems are the future of MMOG development – part 3.

...BRIEFLY:

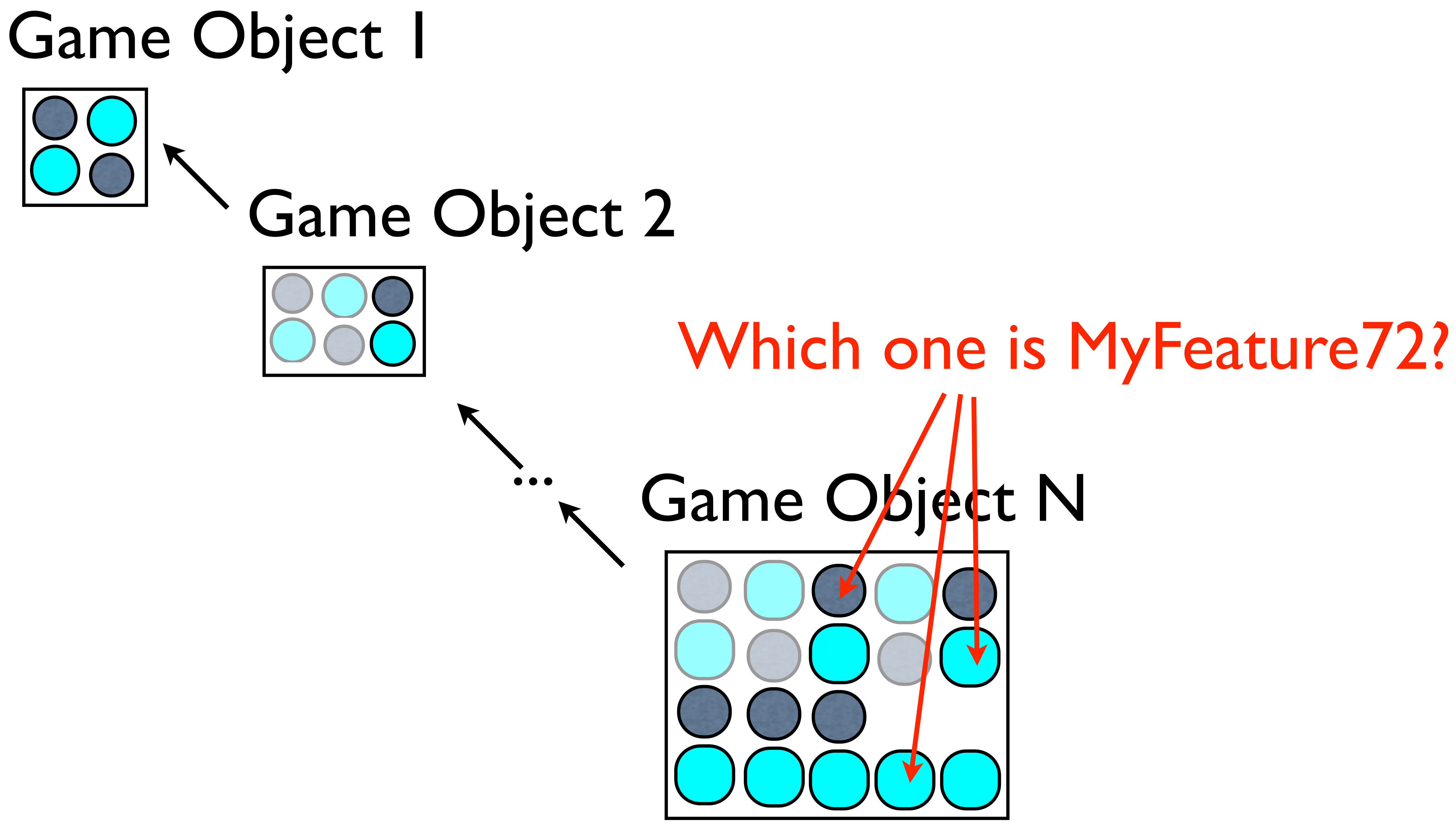
IT Industry attempts Computer Games



IT Industry attempts Computer Games



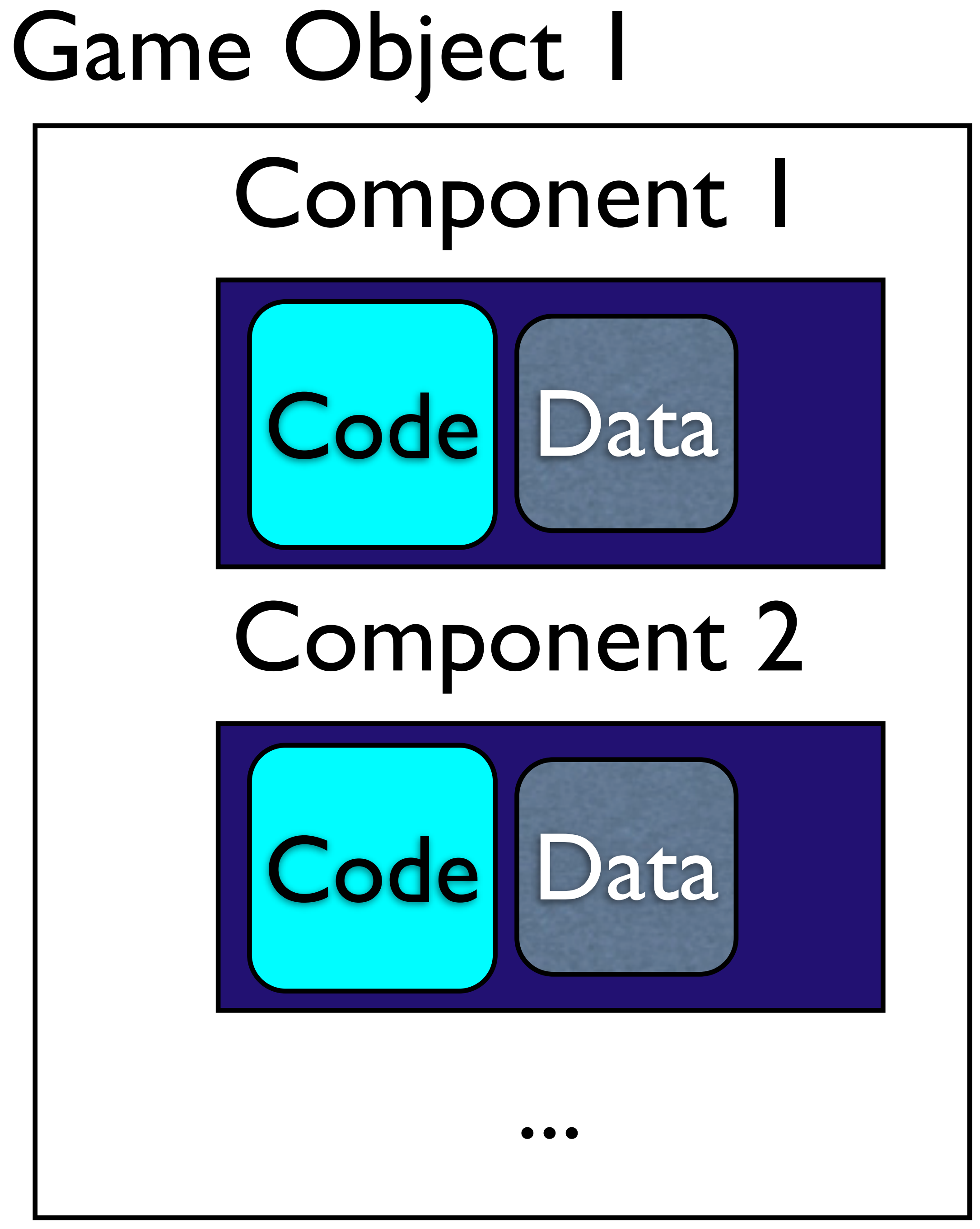
IT Industry attempts Computer Games



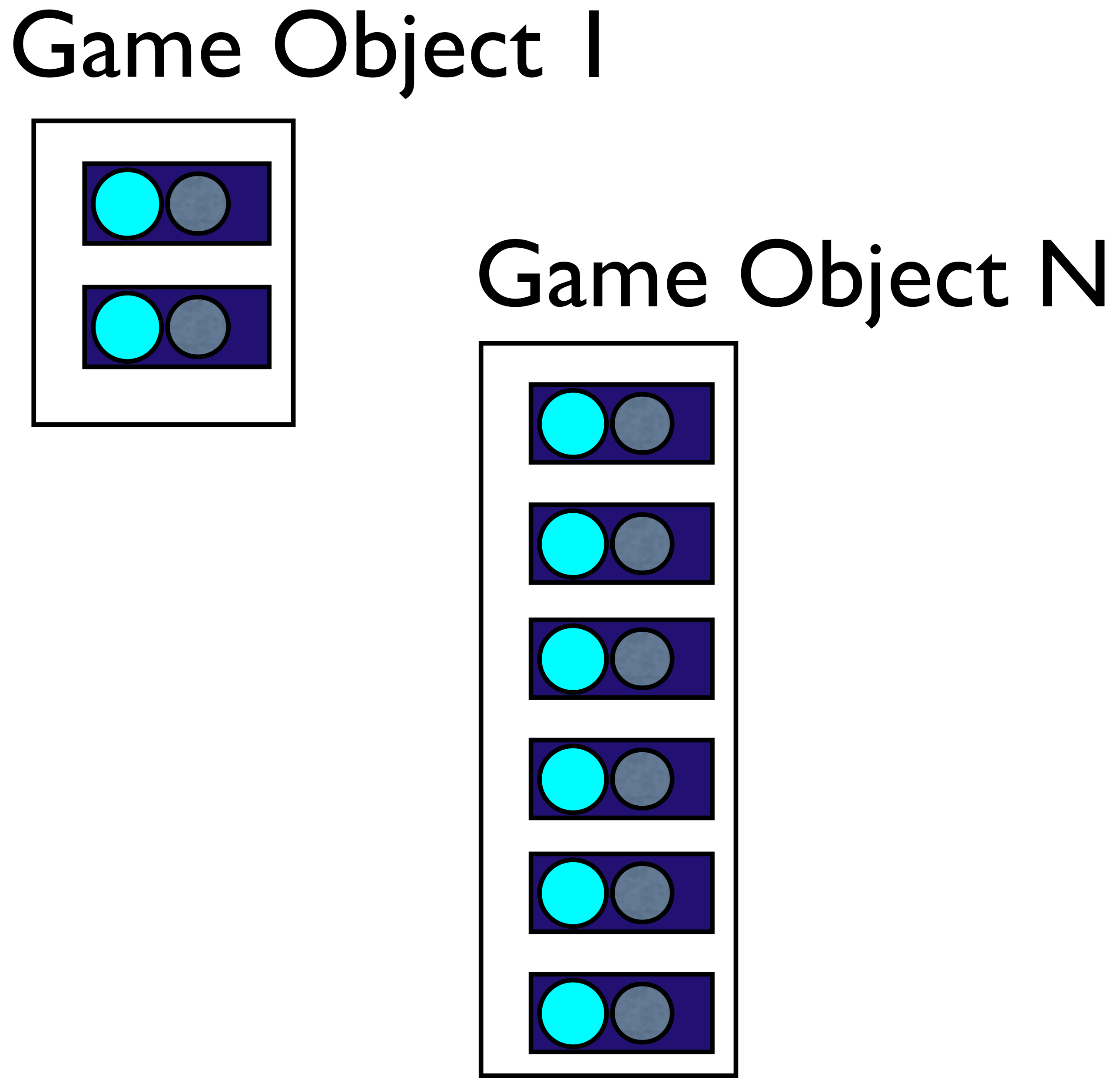
Computer hates it too

- Modern computers terribly slow at doing "one RANDOM little piece at a time"
- Memory-fragmentation is an Unsolved Problem (tm)
- Multithreaded code / processors detest OOP encapsulation

Unity does Computer Games

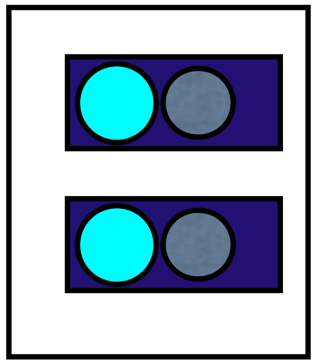


Unity does Computer Games

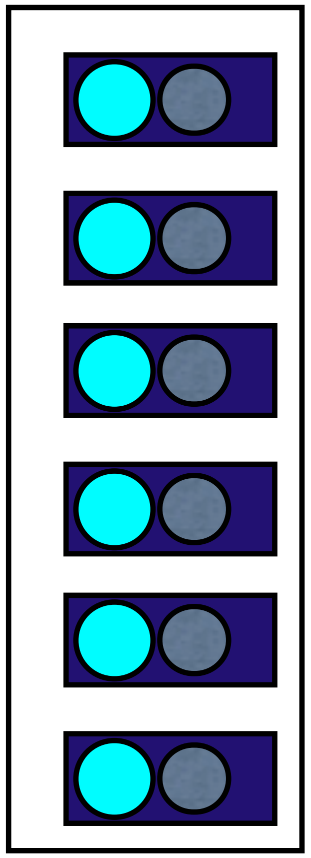


Unity does Computer Games

Game Object 1



Game Object N

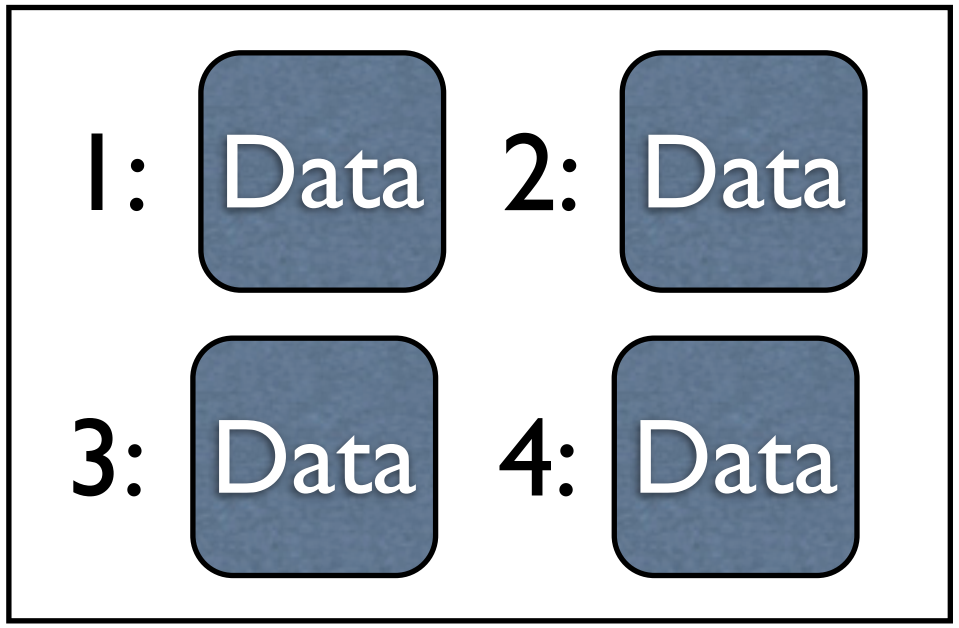


Improvement?

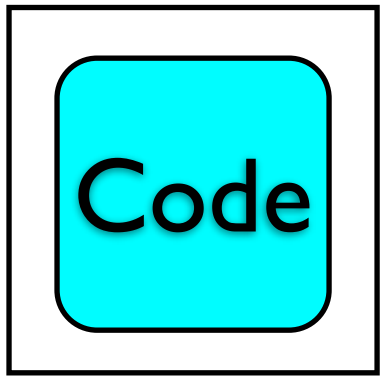
Hmm.

GameDevs make EntitySystems

Feature 1: all objects

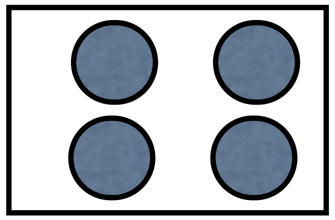


Feature 1: all code

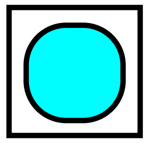


GameDevs make EntitySystems

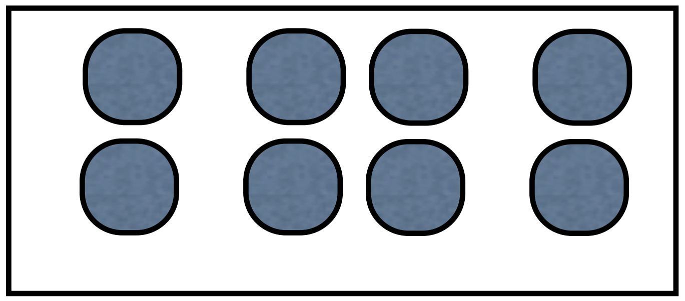
Feature 1: all objects



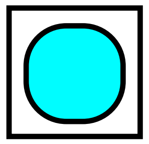
Feature 1: all code



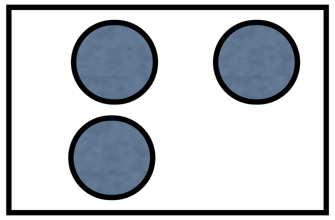
Feature 2: all objects



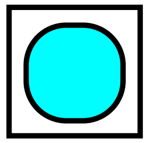
Feature 2: all code



Feature 3: all objects

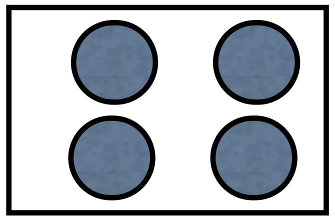


Feature 3: all code

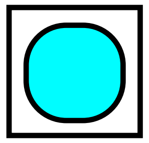


GameDevs make EntitySystems

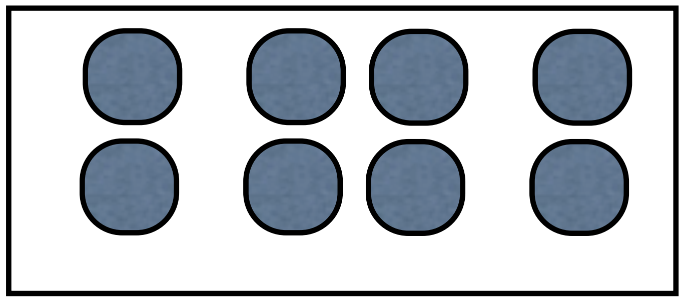
Feature 1: all objects



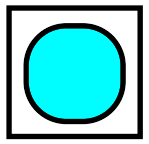
Feature 1: all code



Feature 2: all objects

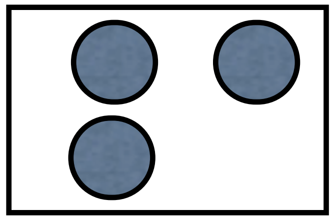


Feature 2: all code

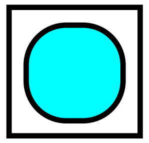


Which one is MyFeature2?

Feature 3: all objects

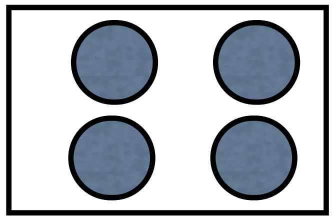


Feature 3: all code

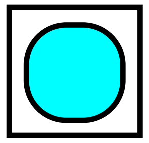


GameDevs make EntitySystems

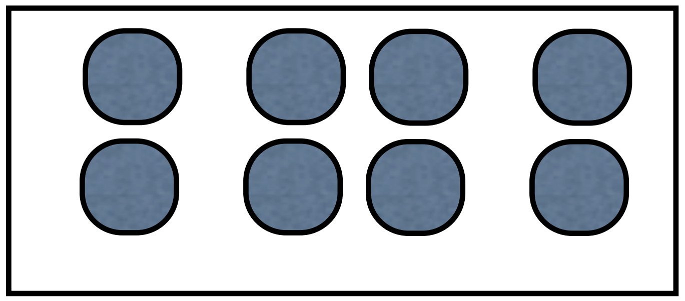
Feature 1: all objects



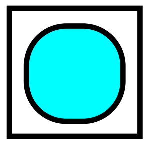
Feature 1: all code



Feature 2: all objects

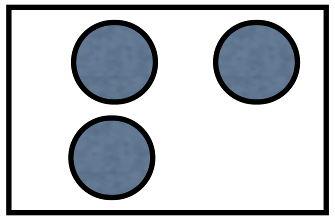


Feature 2: all code

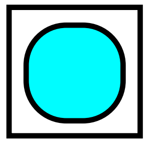


Which one is MyFeature2?

Feature 3: all objects



Feature 3: all code



Computer loves it too

- Can do "all the Feature N" objects at once
- Fragmentation much easier to cope with
- Multithreading is EASY! (almost: Free)

In practice, though...

Unity

Entity System

"which file did I write that in?" $\xrightarrow[\text{Coding}]{\text{vs.}}$

In practice, though...

Unity

Entity System

“which file did I write that in?” $\xrightarrow[\text{Coding}]{\text{vs.}}$ “Easy. This one.”

In practice, though...

Unity **Entity System**

“which file did I write that in?” $\xrightarrow[\text{Coding}]{\text{vs.}}$ “Easy. This one.”

“to add a number, must write a script” $\xrightarrow[\text{Prototyping}]{\text{vs.}}$

In practice, though...

Unity		Entity System
“which file did I write that in?”	$\xrightarrow[\text{Coding}]{\text{vs.}}$	“Easy. This one.”
“to add a number, must write a script”	$\xrightarrow[\text{Prototyping}]{\text{vs.}}$	“Add the number”

In practice, though...

Unity		Entity System
“which file did I write that in?”	$\xrightarrow[\text{Coding}]{\text{vs.}}$	“Easy. This one.”
“to add a number, must write a script”	$\xrightarrow[\text{Prototyping}]{\text{vs.}}$	“Add the number”
“Find() is slooooow”	$\xrightarrow[\text{Speed}]{\text{vs.}}$	

In practice, though...

Unity		Entity System
“which file did I write that in?”	$\xrightarrow[\text{Coding}]{\text{vs.}}$	“Easy. This one.”
“to add a number, must write a script”	$\xrightarrow[\text{Prototyping}]{\text{vs.}}$	“Add the number”
“Find() is slooooow”	$\xrightarrow[\text{Speed}]{\text{vs.}}$	“Entity.Find() is lightning fast”

Generally speaking

- Every “new game feature” is quicker to try
- Every algorithm requires less source code
- Bugs are easier to isolate
- Performance is improved across the board

Entity Systems
... inside Unity

3 x Key Areas

- Making stuff in the Editor
- Writing + editing scripts
- Running in the Player (Runtime)

(recap)

Why use an Entity System?

- 1. Quicker prototyping (games, features)
- 2. Faster game (CPU, RAM, Mobile)
- 3. Easier code (debug less, design more)

(a.k.a.)

Why use an Entity System?

- 1. Easier to add game-features
- 2. Runs faster
- 3. Less / easier Debugging

Effectiveness

	Debug	Game Features	Speed
In the Editor?	✓	✓	
Writing scripts?	✓	✓	
Running in the Player?		✓	✓

Editor, I choose you!

	Debug	Game Features	Speed
In the Editor?	✓	✓	
Writing scripts?	✓	✓	
Running in the Player?		✓	✓

In-Editor Goals

- Is it easy to use?
  - ... easier than plain Unity?
- Can this be implemented in C#?
  - ... without destroying performance?
  - ... without writing “bad code”?
  - ... without “bizarre coding practices”?

Challenges so far...

3 problems

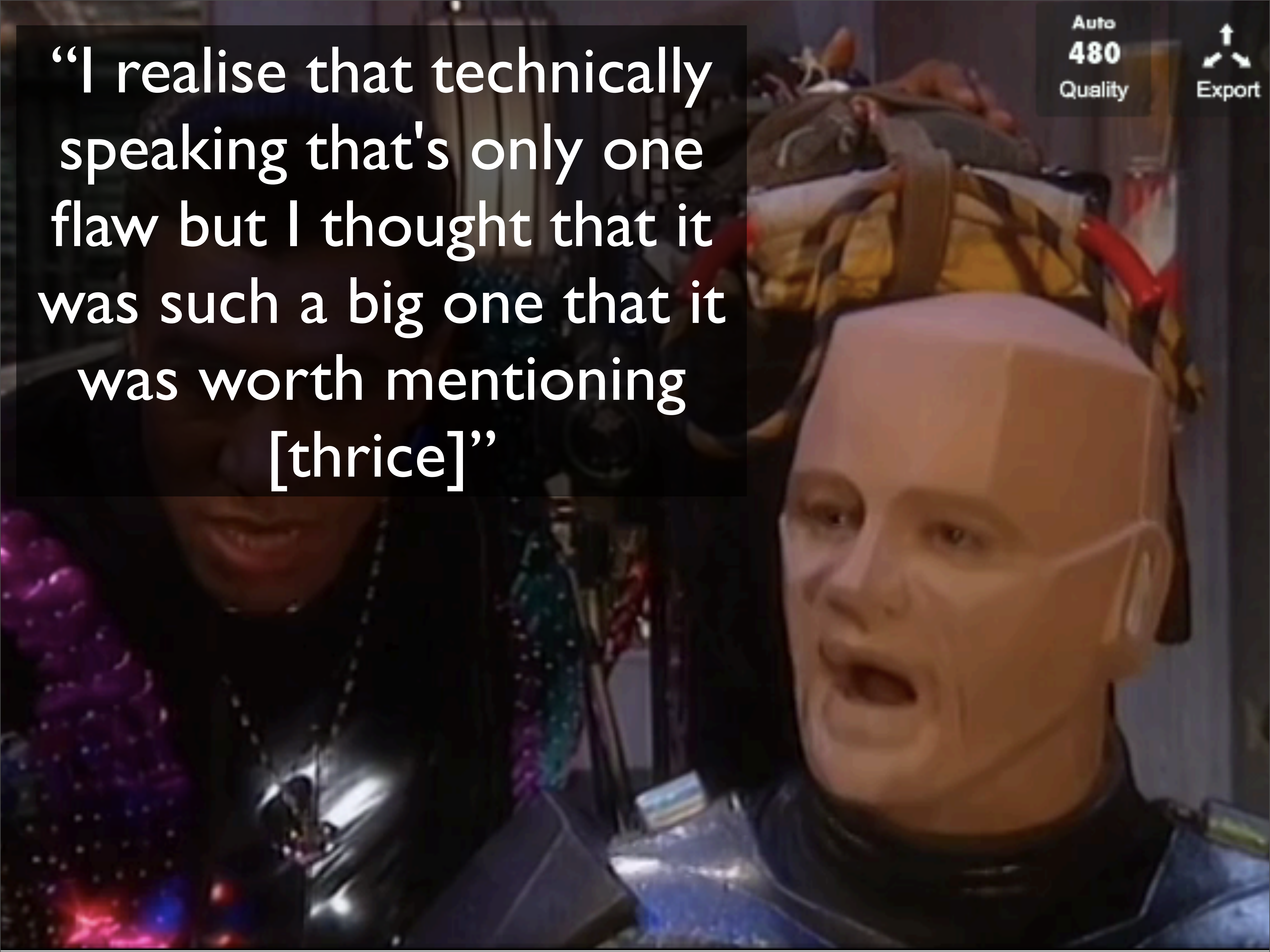
- 1. Unity Serialization

3 problems

- 1. Unity Serialization
- 2. Unity Serialization

3 problems

- 1. Unity Serialization
- 2. Unity Serialization
- 3. Unity Serialization



"I realise that technically speaking that's only one flaw but I thought that it was such a big one that it was worth mentioning [thrice]"

Unity ...“challenges”

- Unity has no API for save/load to a Scene
- ScriptableObject is half-implemented, half-broken
- ...Unity Serialization

Unity Serialization … “challenges”

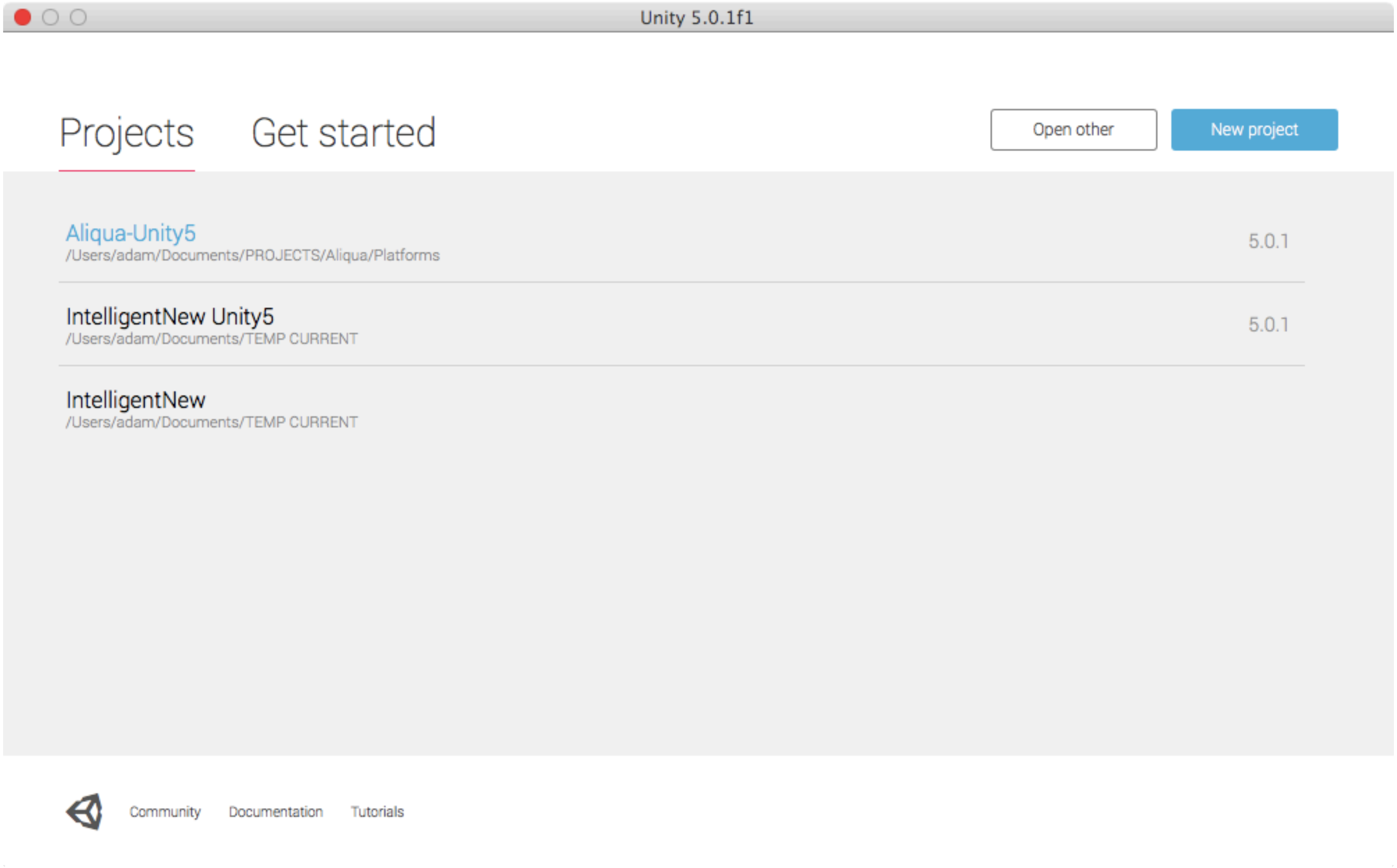
- ..requires SetDirty
- ..sometimes ignores SetDirty
  - SetDirty needs Find
    - Unity doesn’t fully support Find (yet)
- ..doesn’t support C# core classes
- ..restoring breaks “static” variables
- ..“change scene” code in Unity 5 is wrong
- ..C# Constructors go FUBAR
- ..THERE’S NO DEBUG INFORMATION
- ..zero Editor support from Unity Corp

Unity Serialization ... “challenges”

...and C#

- “return ref” isn’t possible (C#)

Progress so far...
... in Editor

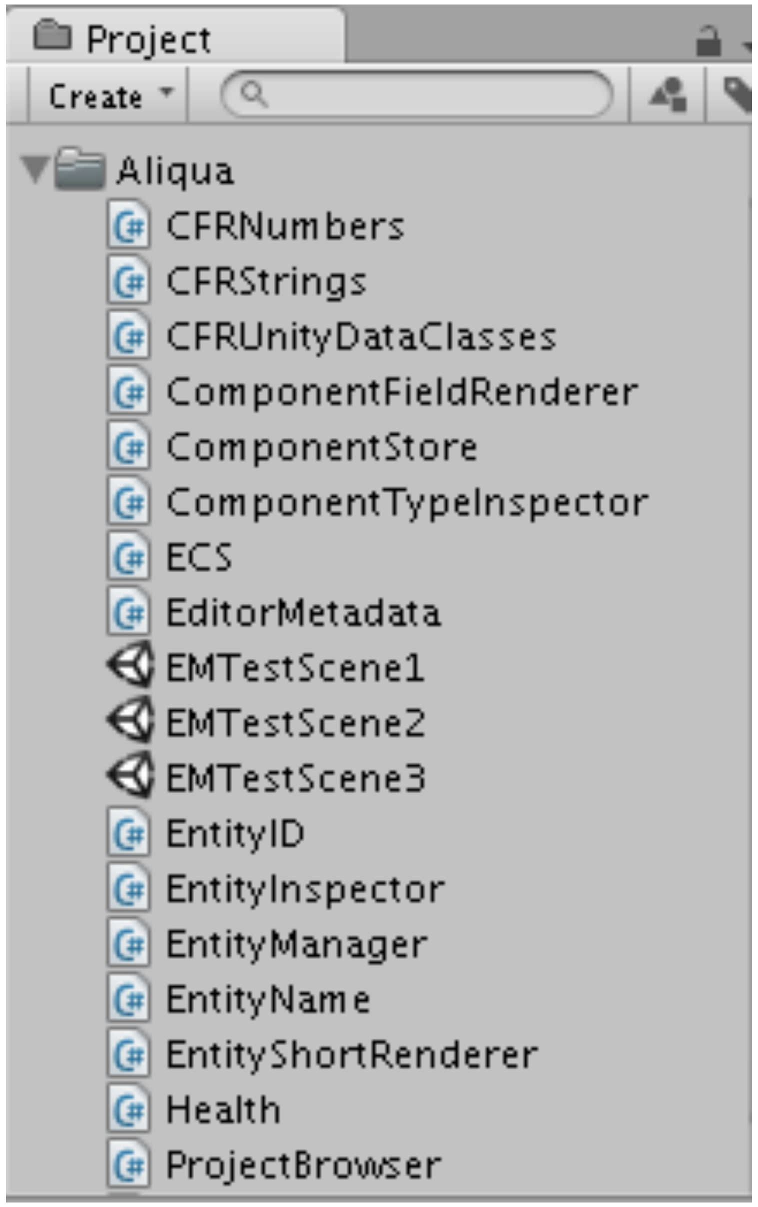


Aliqua-Unity5
/Users/adam/Documents/PROJECTS/Aliqua/Platforms
5.0.1

IntelligentNew Unity5
/Users/adam/Documents/TEMP CURRENT
5.0.1

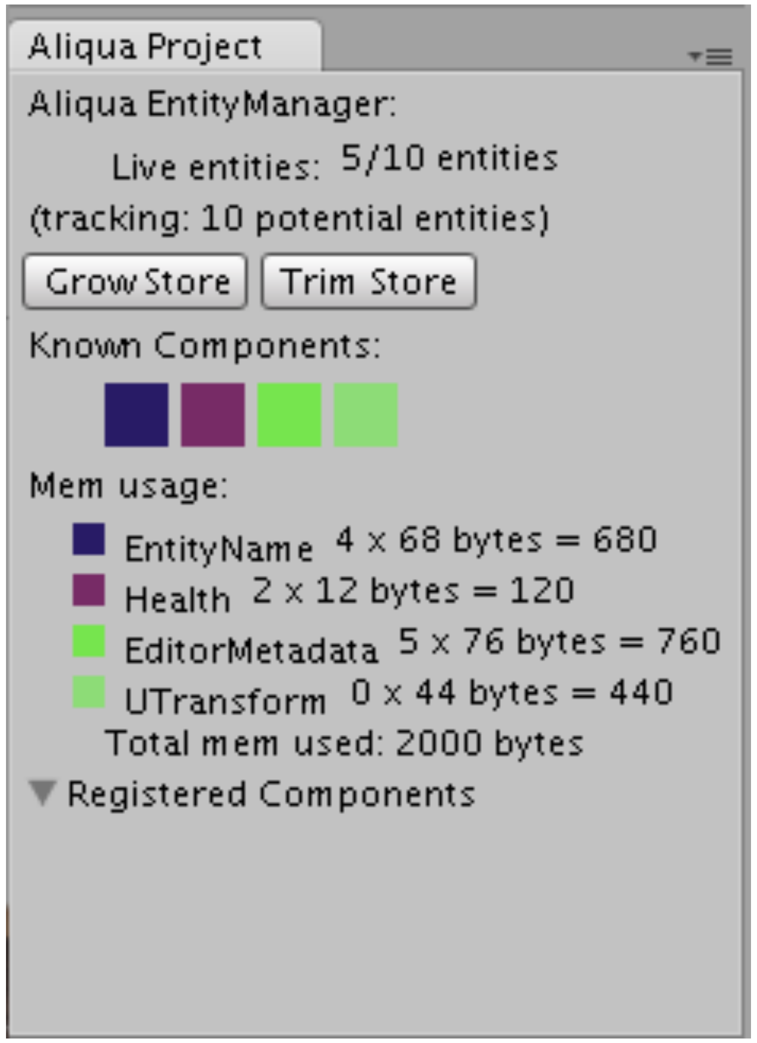
IntelligentNew
/Users/adam/Documents/TEMP CURRENT

Project Window



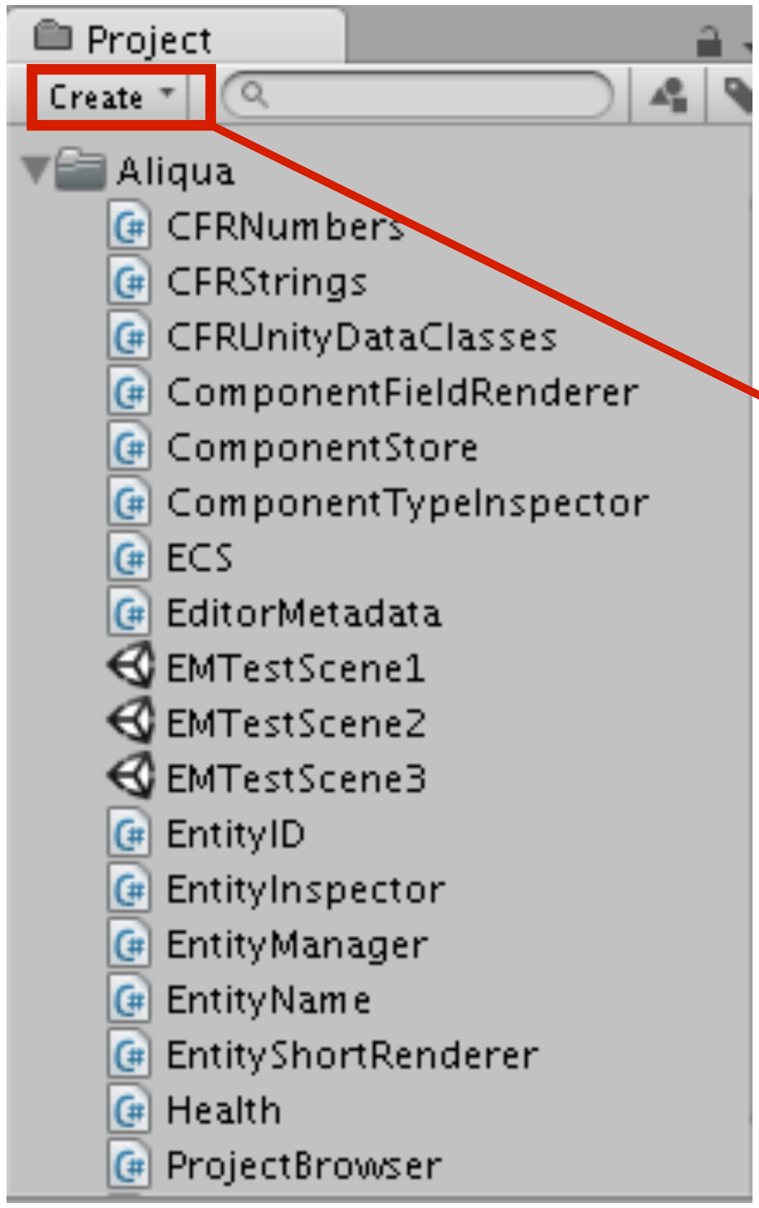
```
Aliqua
    CFRNumbers
    CFRStrings
    CFRUnityDataClasses
    ComponentFieldRenderer
    ComponentStore
    ComponentTypeInspector
    ECS
    EditorMetadata
    EMTestScene1
    EMTestScene2
    EMTestScene3
    EntityID
    EntityInspector
    EntityManager
    EntityName
    EntityShortRenderer
    Health
    ProjectBrowser
```

+



```
Aliqua EntityManager:
    Live entities:  5/10 entities
(tracking: 10 potential entities)
Known Components:
Mem usage:
    EntityName  4 x 68 bytes = 680
    Health  2 x 12 bytes = 120
    EditorMetadata  5 x 76 bytes = 760
    UTransform  0 x 44 bytes = 440
    Total mem used: 2000 bytes
```

Project Window

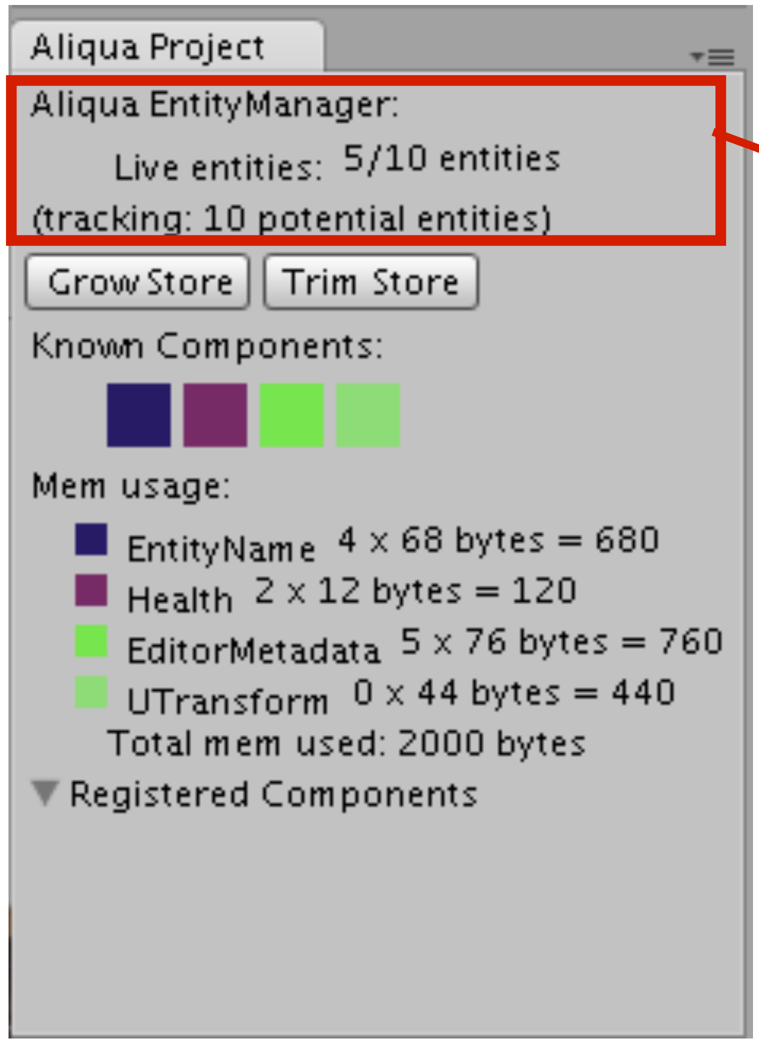


Aliqua
CFRNumbers
CFRStrings
CFRUnityDataClasses
ComponentFieldRenderer
ComponentStore
ComponentTypeInspector
ECS
EditorMetadata
EMTestScene1
EMTestScene2
EMTestScene3
EntityID
EntityInspector
EntityManager
EntityName
EntityShortRenderer
Health
ProjectBrowser

Use Unity:

- Create/Edit scripts

Project Window



Aliqua Project

Aliqua EntityManager:

Live entities: 5/10 entities

(tracking: 10 potential entities)

Known Components:

Mem usage:

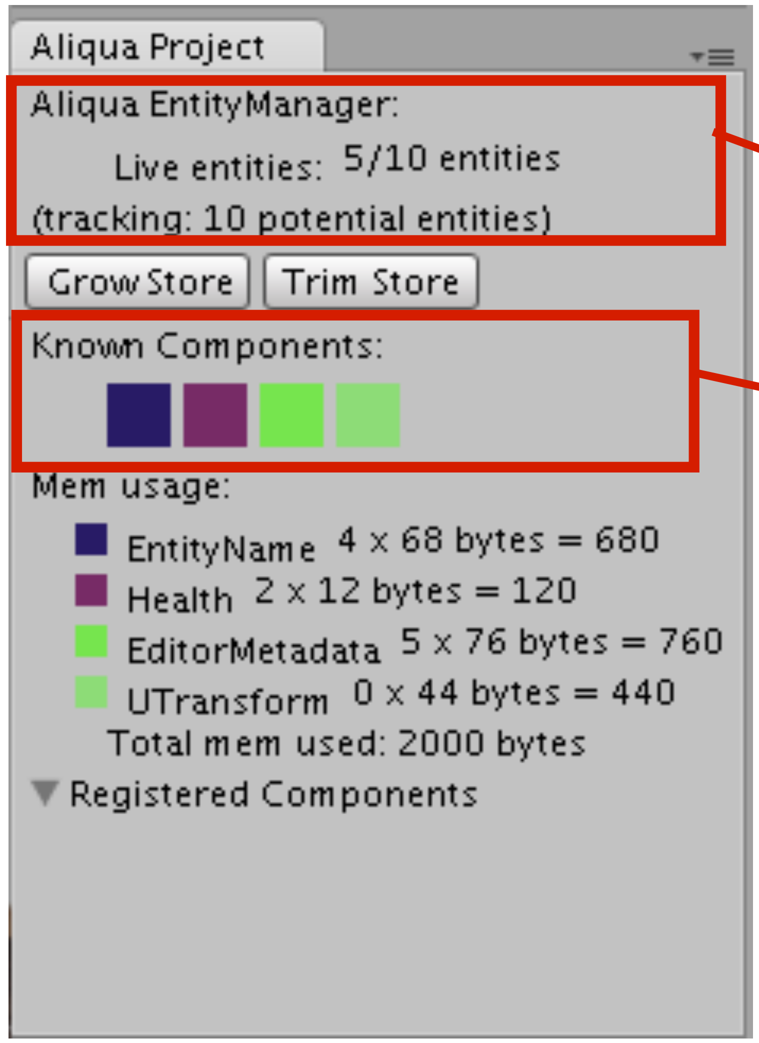
🟦	EntityName	4 x 68 bytes = 680
🟪	Health	2 x 12 bytes = 120
🟩	EditorMetadata	5 x 76 bytes = 760
🟩	UTransform	0 x 44 bytes = 440

Total mem used: 2000 bytes

Use Entity System:

- Debug "invisible stuff"

Project Window



Aliqua Project

Aliqua EntityManager:

Live entities: 5/10 entities

(tracking: 10 potential entities)

Known Components:

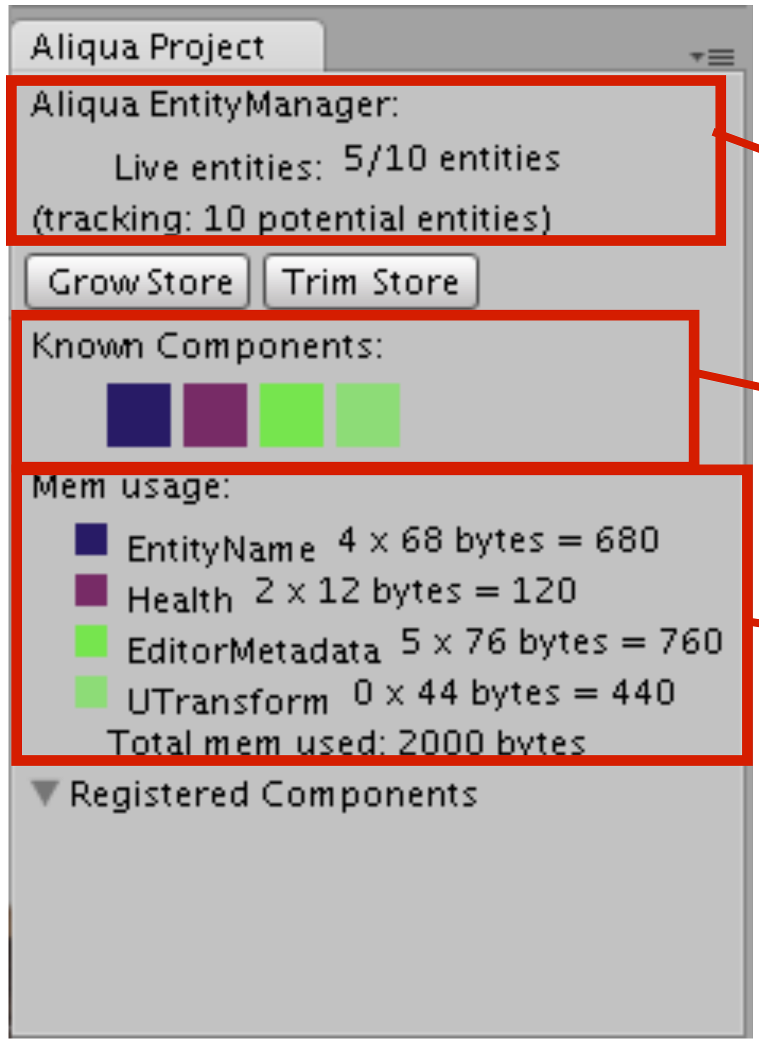
Mem usage:

```
🟦 EntityName     4 x 68 bytes = 680
🟪 Health         2 x 12 bytes = 120
🟩 EditorMetadata 5 x 76 bytes = 760
🟩 UTransform     0 x 44 bytes = 440
Total mem used: 2000 bytes
```

Use Entity System:

- Debug "invisible stuff"
- Register Components

Project Window



Aliqua Project

Aliqua EntityManager:

Live entities: 5/10 entities

(tracking: 10 potential entities)

Grow Store | Trim Store

Known Components:

Mem usage:

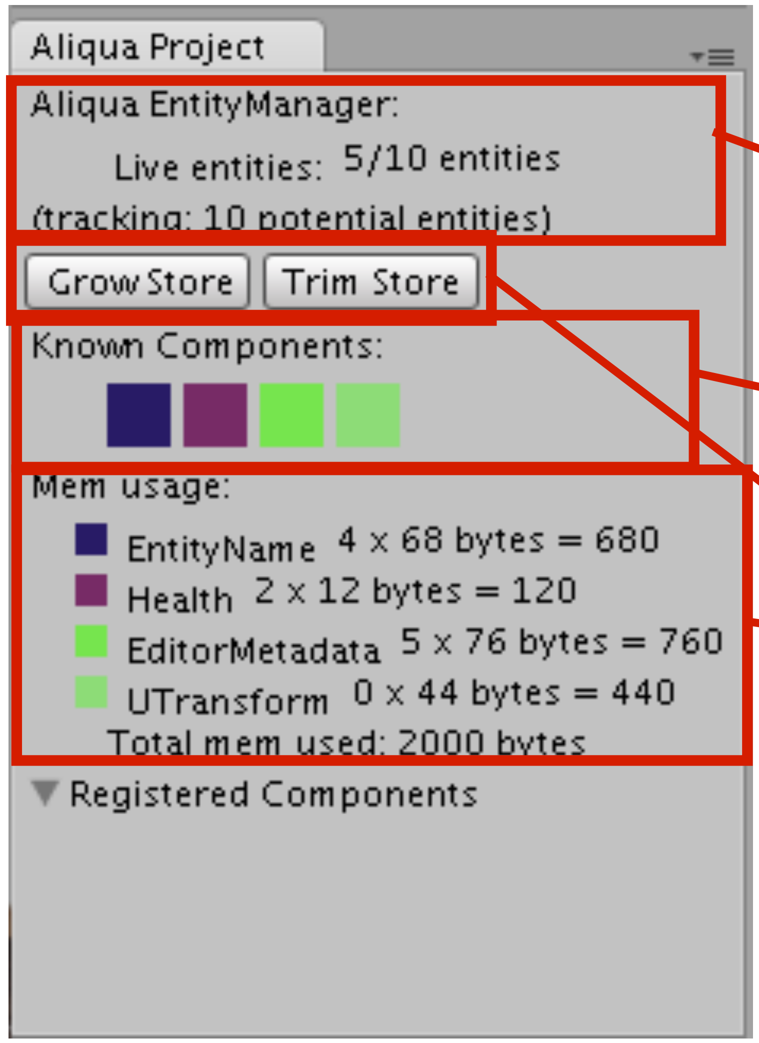
```
EntityName     4 x 68 bytes = 680
Health         2 x 12 bytes = 120
EditorMetadata 5 x 76 bytes = 760
UTransform     0 x 44 bytes = 440
Total mem used: 2000 bytes
```

▼ Registered Components

Use Entity System:

- Debug "invisible stuff"
- Register Components
- Watch Memory-usage

Project Window



Aliqua Project

Aliqua EntityManager:

Live entities: 5/10 entities

(tracking: 10 potential entities)

Known Components:

🟦 🟪 🟩 🟩

Mem usage:

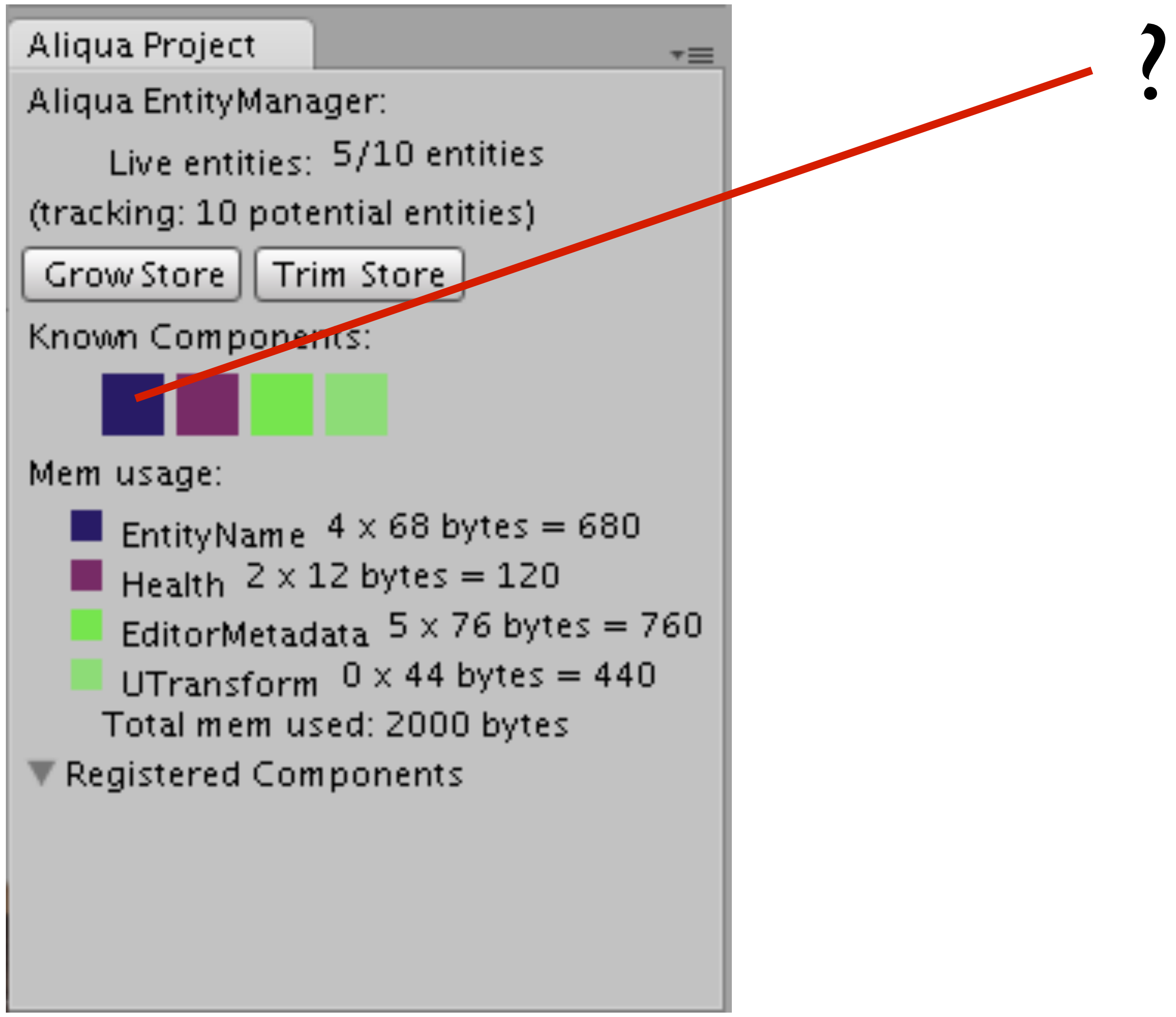
```
🟦 EntityName    4 x 68 bytes = 680
🟪 Health        2 x 12 bytes = 120
🟩 EditorMetadata  5 x 76 bytes = 760
🟩 UTransform    0 x 44 bytes = 440
  Total mem used: 2000 bytes
```

▼ Registered Components

Use Entity System:

- Debug "invisible stuff"
- Register Components
- Watch Memory-usage
- Test internals

Project Window



Aliqua Project

Aliqua EntityManager:

Live entities: 5/10 entities

(tracking: 10 potential entities)

Grow Store | Trim Store

Known Components:

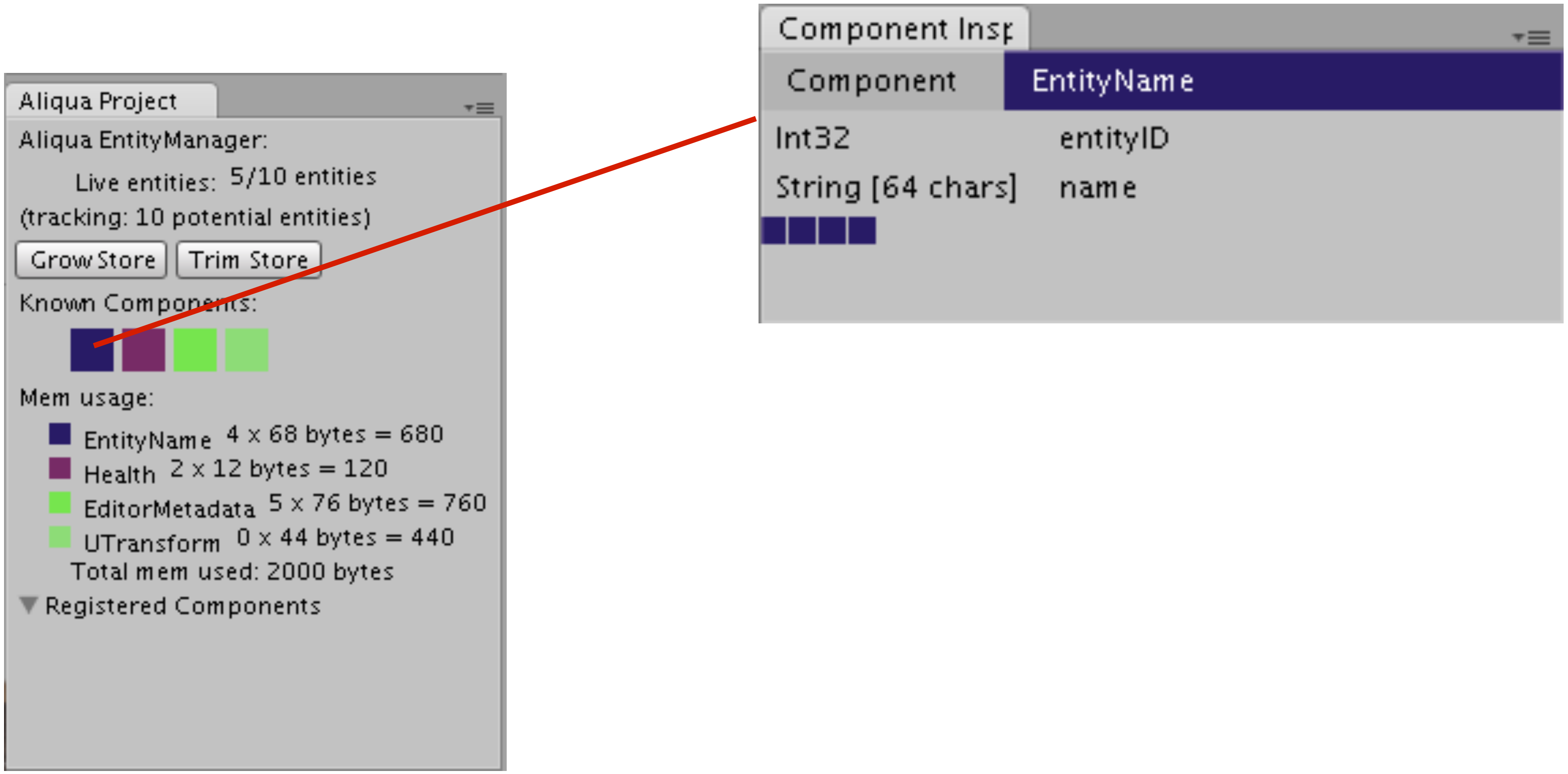
Mem usage:

🟪	EntityName	4 x 68 bytes = 680
🟪	Health	2 x 12 bytes = 120
🟩	EditorMetadata	5 x 76 bytes = 760
🟩	UTransform	0 x 44 bytes = 440

Total mem used: 2000 bytes

▼ Registered Components

Project Window



Aliqua Project

Aliqua EntityManager:

Live entities: 5/10 entities

(tracking: 10 potential entities)

Grow Store | Trim Store

Known Components:

🟦 🟪 🟩 🟩

Mem usage:

🟦	EntityName	4 x 68 bytes = 680
🟪	Health	2 x 12 bytes = 120
🟩	EditorMetadata	5 x 76 bytes = 760
🟩	UTransform	0 x 44 bytes = 440

Total mem used: 2000 bytes

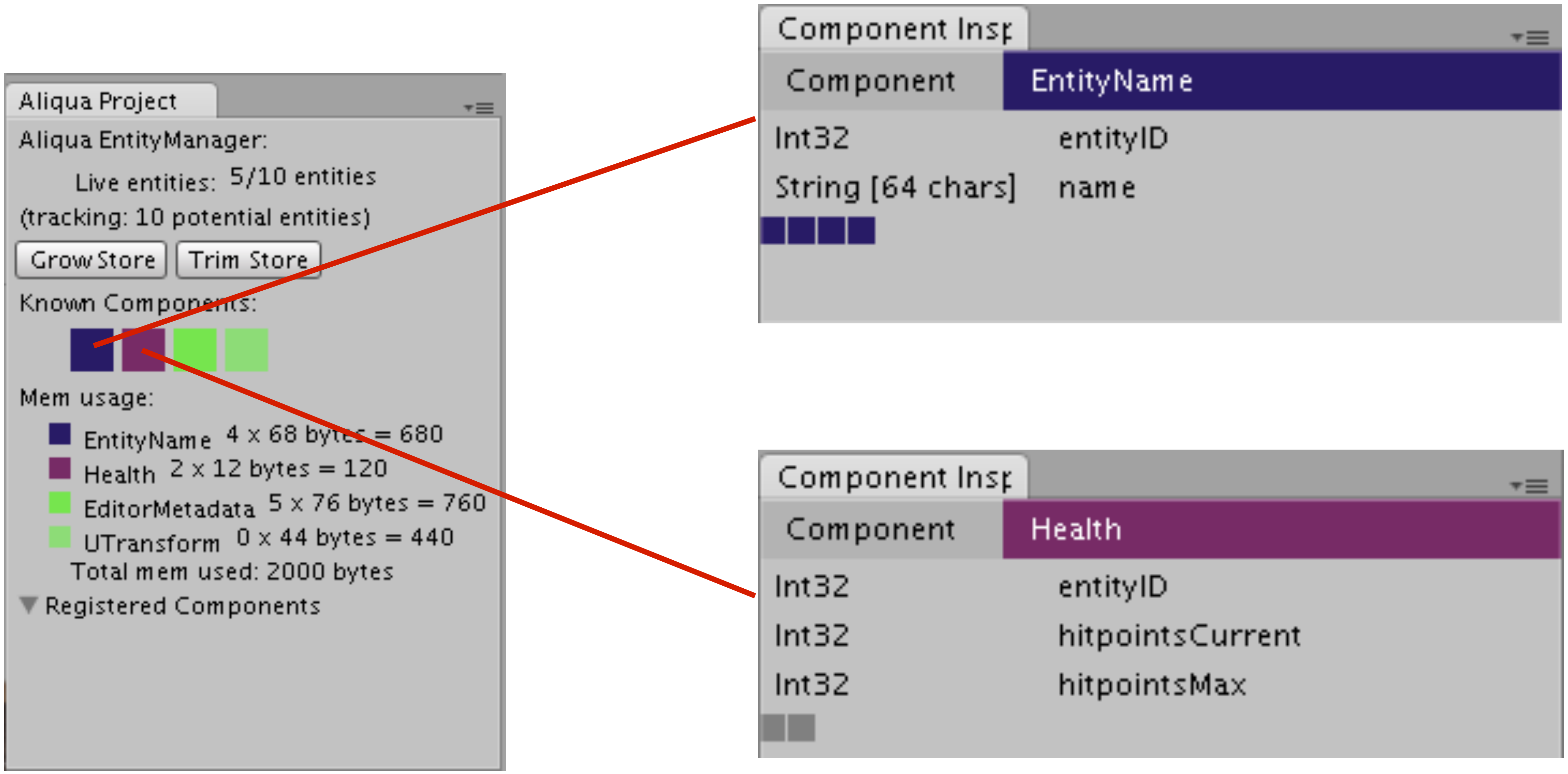
▼ Registered Components

Component Insp

Component	EntityName
Int32	entityID
String [64 chars]	name

🟦🟦🟦🟦

Project Window



Aliqua Project

Aliqua EntityManager:

Live entities: 5/10 entities

(tracking: 10 potential entities)

Grow Store | Trim Store

Known Components:

Mem usage:

■	EntityName	4 x 68 bytes = 680
■	Health	2 x 12 bytes = 120
■	EditorMetadata	5 x 76 bytes = 760
■	UTransform	0 x 44 bytes = 440

Total mem used: 2000 bytes

▼ Registered Components

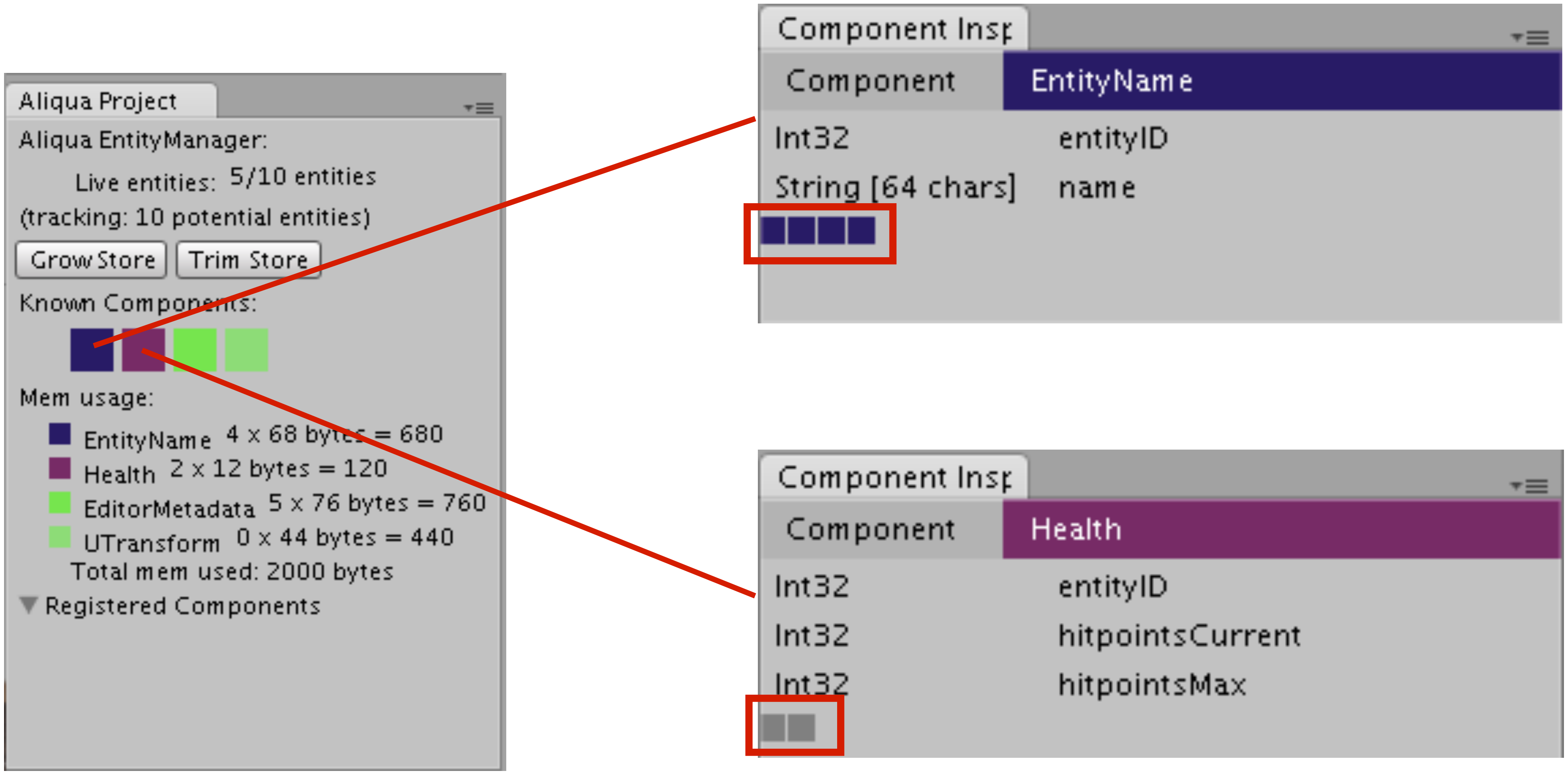
Component Insp

Component	EntityName
Int32	entityID
String [64 chars]	name

Component Insp

Component	Health
Int32	entityID
Int32	hitpointsCurrent
Int32	hitpointsMax

Project Window



Aliqua Project

Aliqua EntityManager:

Live entities: 5/10 entities

(tracking: 10 potential entities)

Grow Store | Trim Store

Known Components:

Mem usage:

■	EntityName	4 x 68 bytes = 680
■	Health	2 x 12 bytes = 120
■	EditorMetadata	5 x 76 bytes = 760
■	UTransform	0 x 44 bytes = 440

Total mem used: 2000 bytes

▼ Registered Components

Component Insp

Component	EntityName
Int32	entityID
String [64 chars]	name

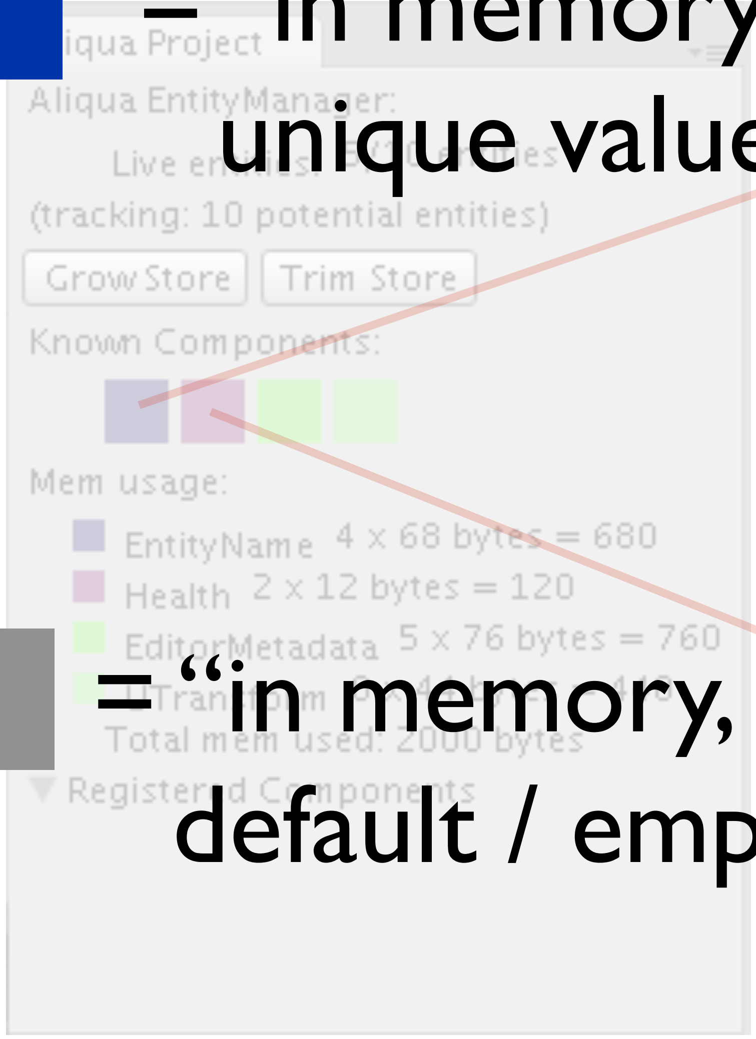
Component Insp

Component	Health
Int32	entityID
Int32	hitpointsCurrent
Int32	hitpointsMax

Project Window

🟦 = "in memory, has unique values"

▨ = "in memory, null / default / empty"



Aliqua EntityManager:

(tracking: 10 potential entities)

Known Components:

Mem usage:

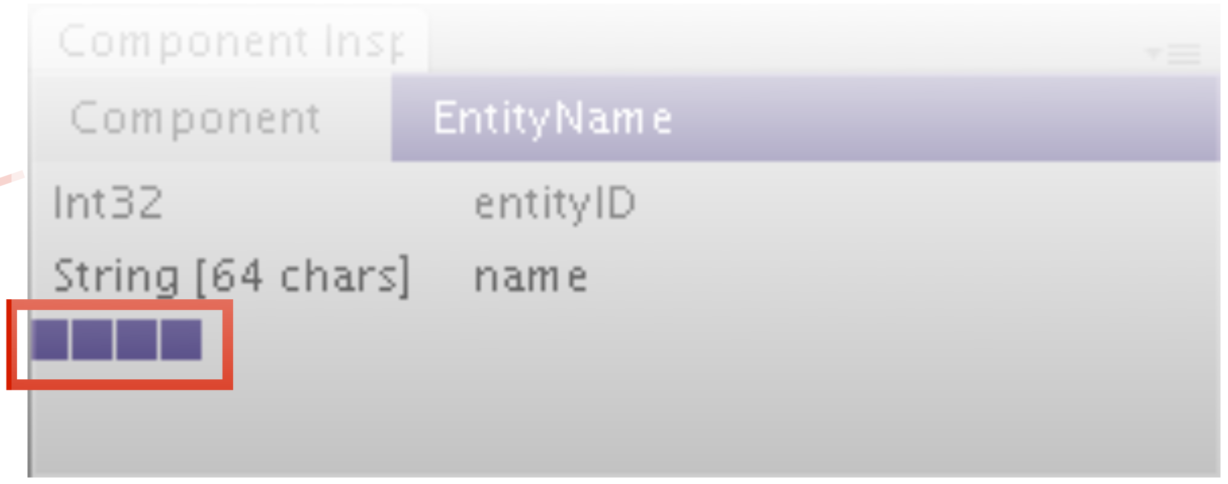
EntityName 4 x 68 bytes = 680

Health 2 x 12 bytes = 120

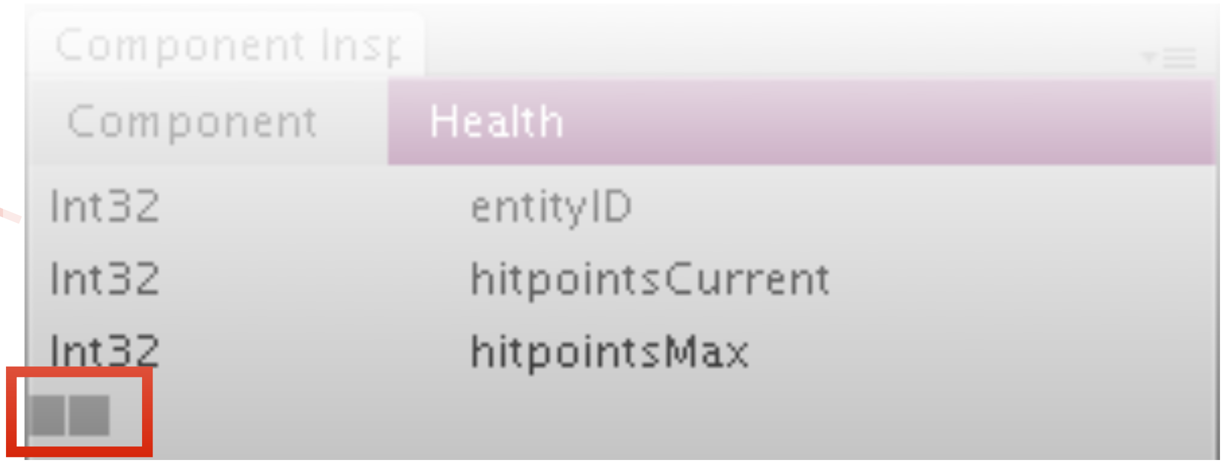
EditorMetadata 5 x 76 bytes = 760

Total mem used: 2000 bytes

Registered Components

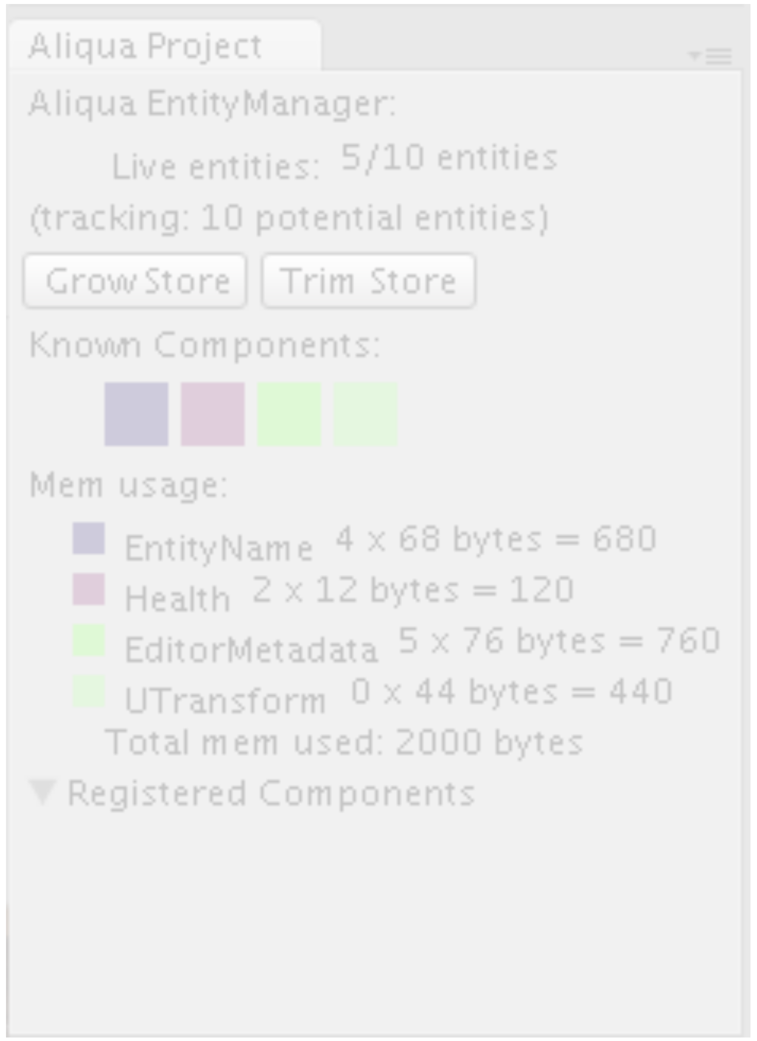


Component	EntityName
Int32	entityID
String [64 chars]	name



Component	Health
Int32	entityID
Int32	hitpointsCurrent
Int32	hitpointsMax

Project Window



Aliqua Project

Aliqua EntityManager:

Live entities: 5/10 entities

(tracking: 10 potential entities)

Known Components:

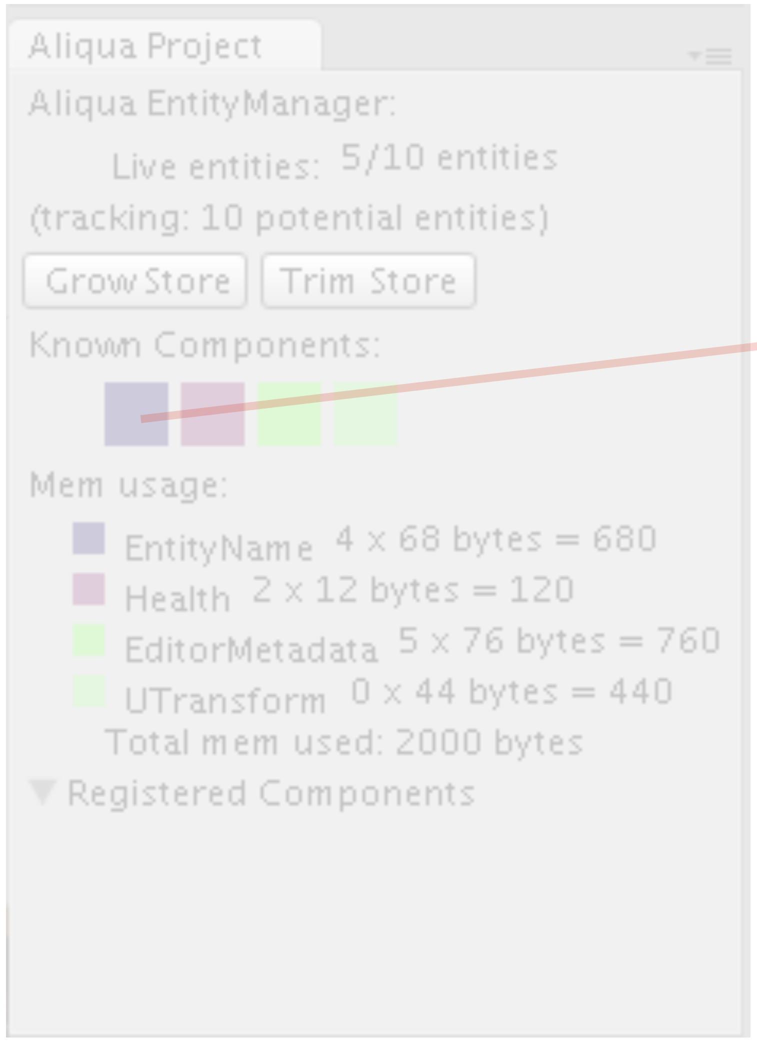
Mem usage:

🟪	EntityName	4 x 68 bytes = 680
🟪	Health	2 x 12 bytes = 120
🟩	EditorMetadata	5 x 76 bytes = 760
🟩	UTransform	0 x 44 bytes = 440

Total mem used: 2000 bytes

▼ Registered Components

Project Window



Aliqua Project

Aliqua EntityManager:

Live entities: 5/10 entities

(tracking: 10 potential entities)

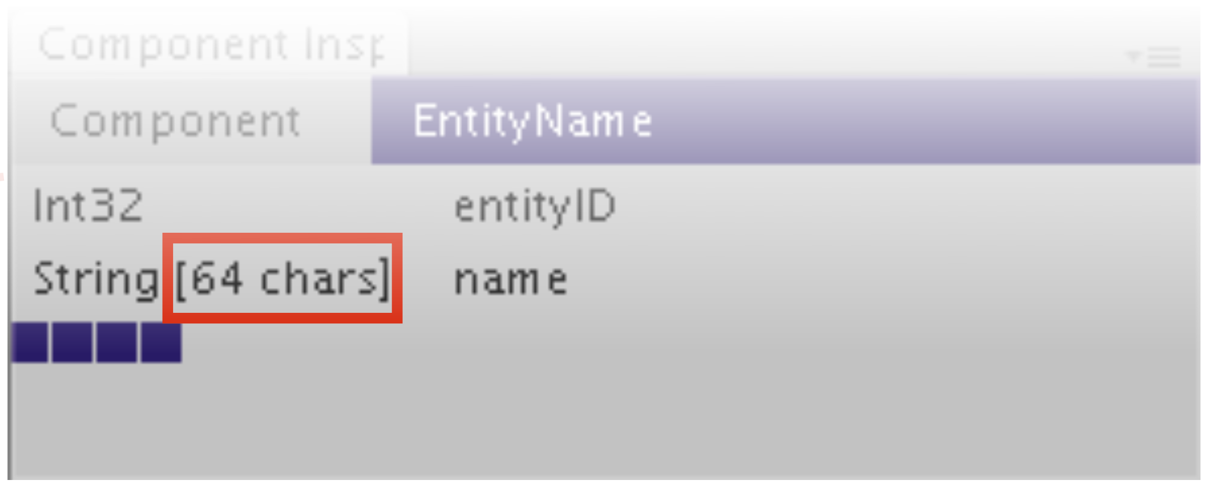
Known Components:

Mem usage:

🟪	EntityName	4 x 68 bytes = 680
🟪	Health	2 x 12 bytes = 120
🟩	EditorMetadata	5 x 76 bytes = 760
🟩	UTransform	0 x 44 bytes = 440

Total mem used: 2000 bytes

▼ Registered Components

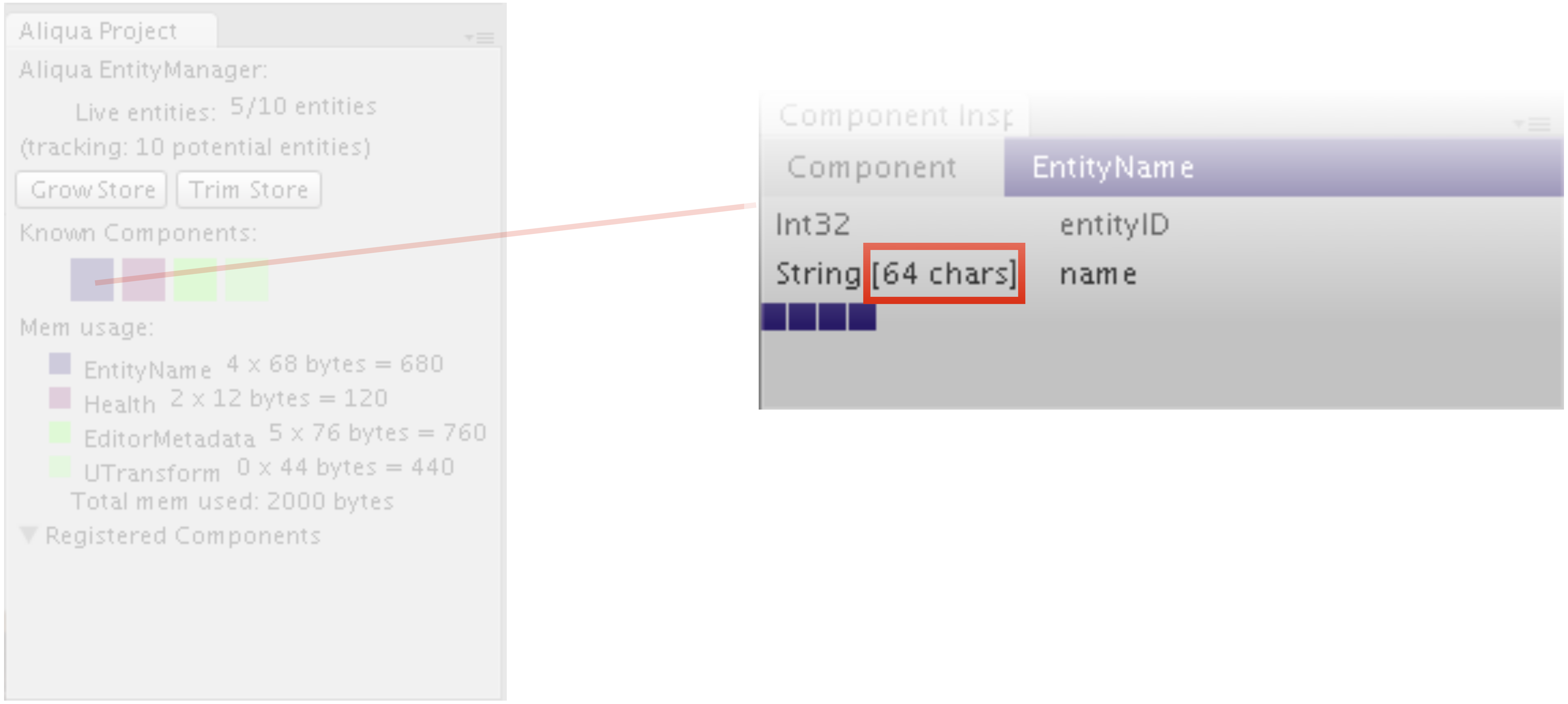


Component Insp

Component	EntityName
Int32	entityID
String [64 chars]	name

Project Window

```
[MarshalAs(UnmanagedType.ByValTStr, SizeConst = 64)]
public string name;
```



Aliqua Project

Aliqua EntityManager:

Live entities: 5/10 entities

(tracking: 10 potential entities)

Known Components:

Mem usage:

	EntityName	4 x 68 bytes = 680
	Health	2 x 12 bytes = 120
	EditorMetadata	5 x 76 bytes = 760
	UTransform	0 x 44 bytes = 440

Total mem used: 2000 bytes

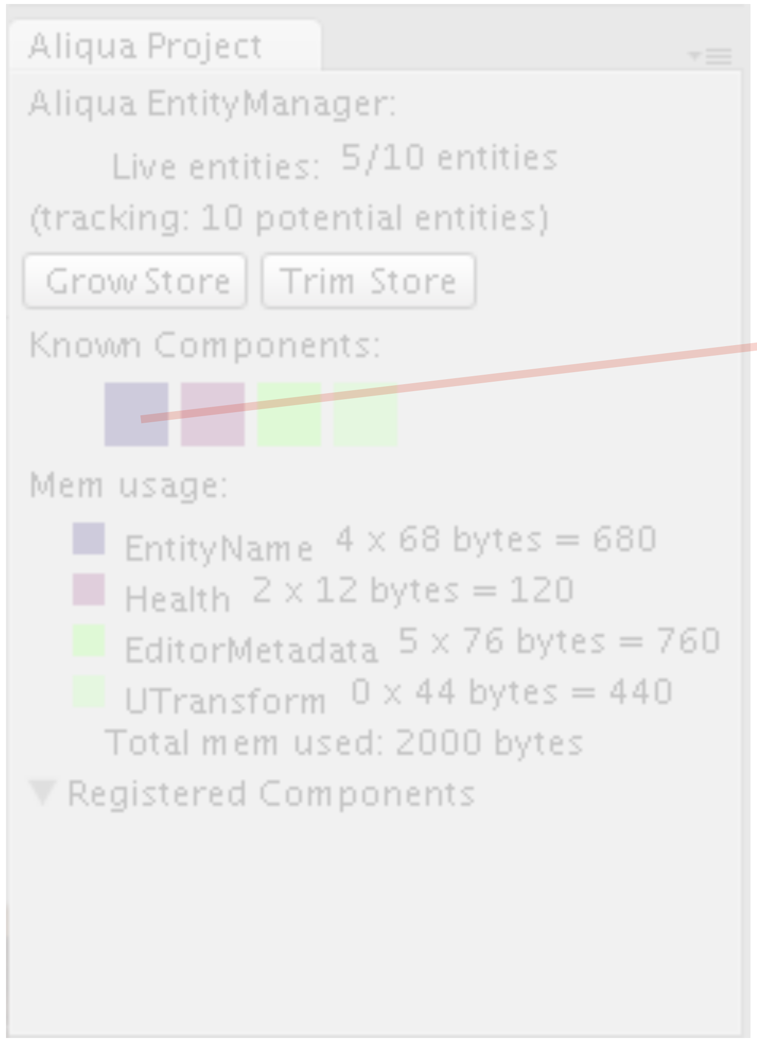
Registered Components

Component Insp

Component	EntityName
Int32	entityID
String [64 chars]	name

Project Window

[MarshalAs(UnmanagedType.ByValTStr, SizeConst = 64)]
public string name;



Aliqua Project

Aliqua EntityManager:

Live entities: 5/10 entities

(tracking: 10 potential entities)

Grow Store | Trim Store

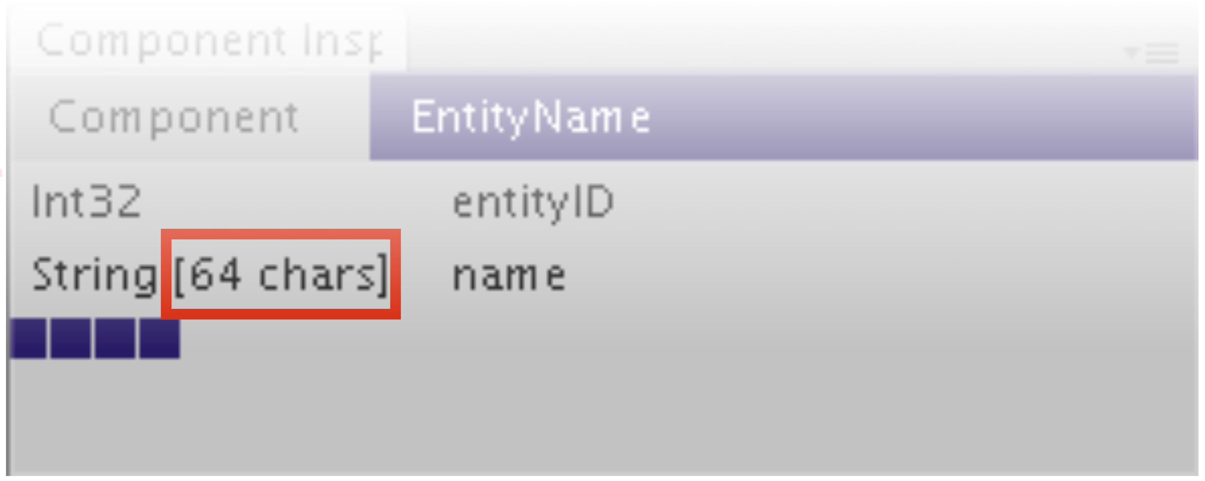
Known Components:

Mem usage:

▨	EntityName	4 x 68 bytes = 680
▨	Health	2 x 12 bytes = 120
▨	EditorMetadata	5 x 76 bytes = 760
▨	UTransform	0 x 44 bytes = 440

Total mem used: 2000 bytes

▼ Registered Components

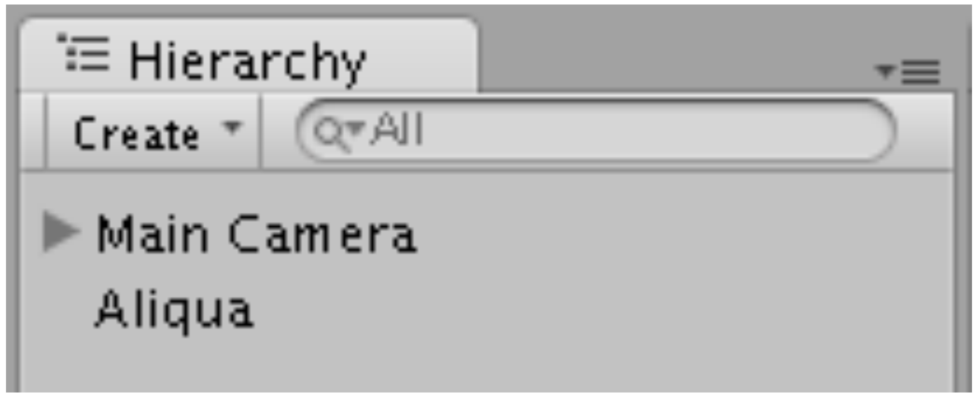


Component Insp

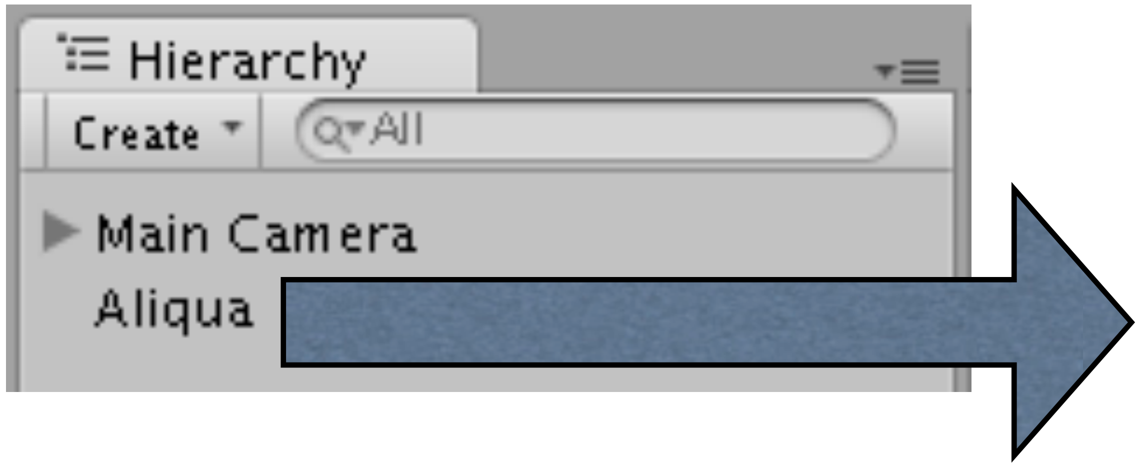
Component	EntityName
Int32	entityID
String [64 chars]	name

= "fixed size struct, only the first 64 chars will save"

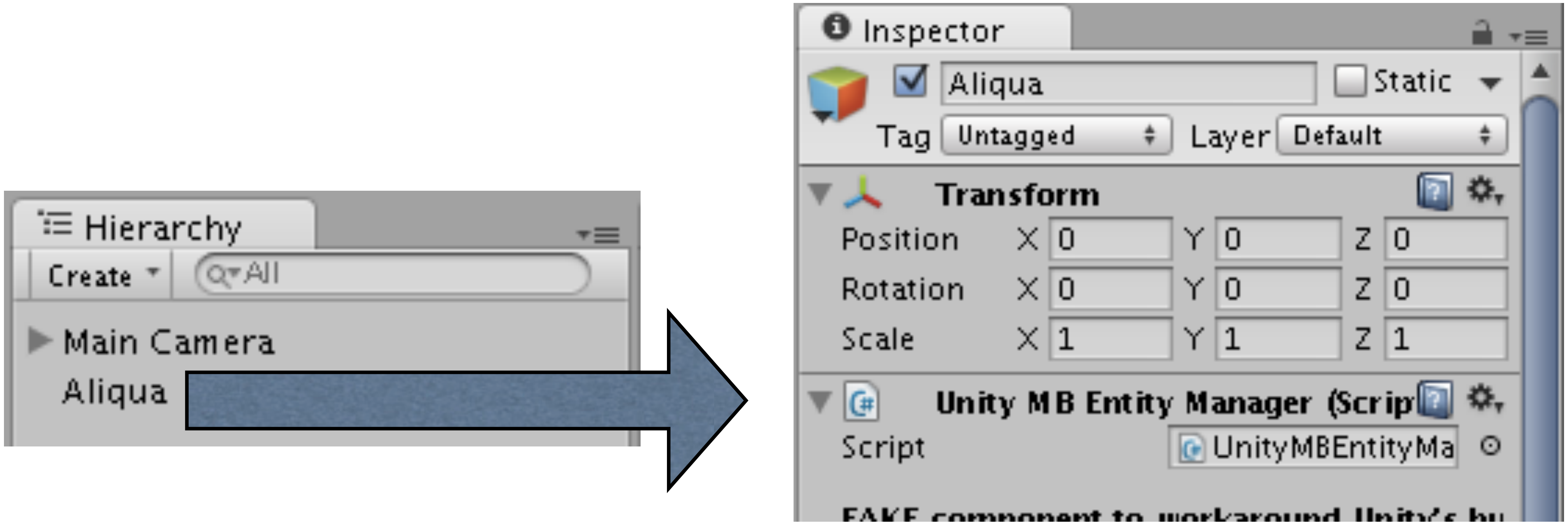
Hierarchy Window



Hierarchy Window



Hierarchy Window



Main Camera
Aliqua

Aliqua
Static
Tag: Untagged
Layer: Default

Transform

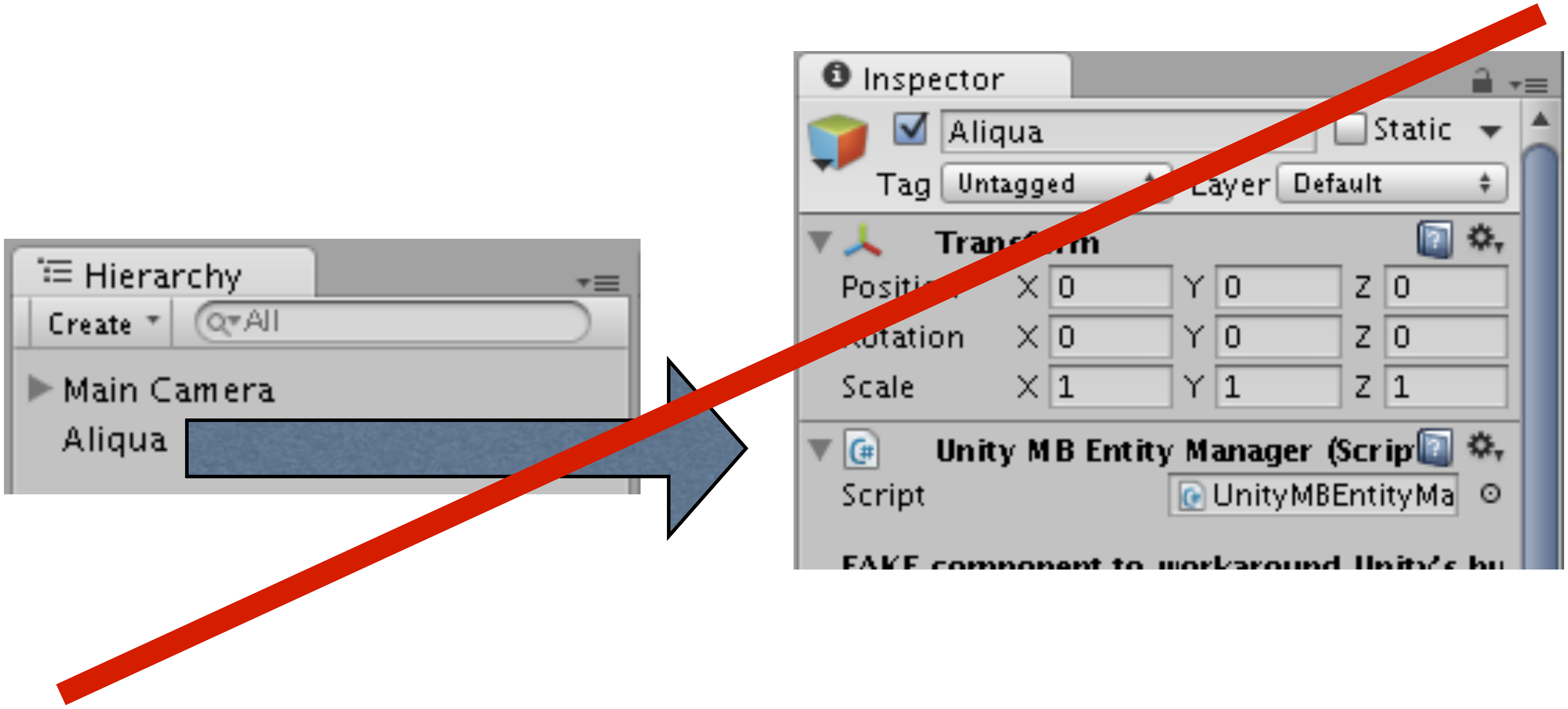
Position	X	0	Y	0	Z	0
Rotation	X	0	Y	0	Z	0
Scale	X	1	Y	1	Z	1

Unity MB Entity Manager (Scrip

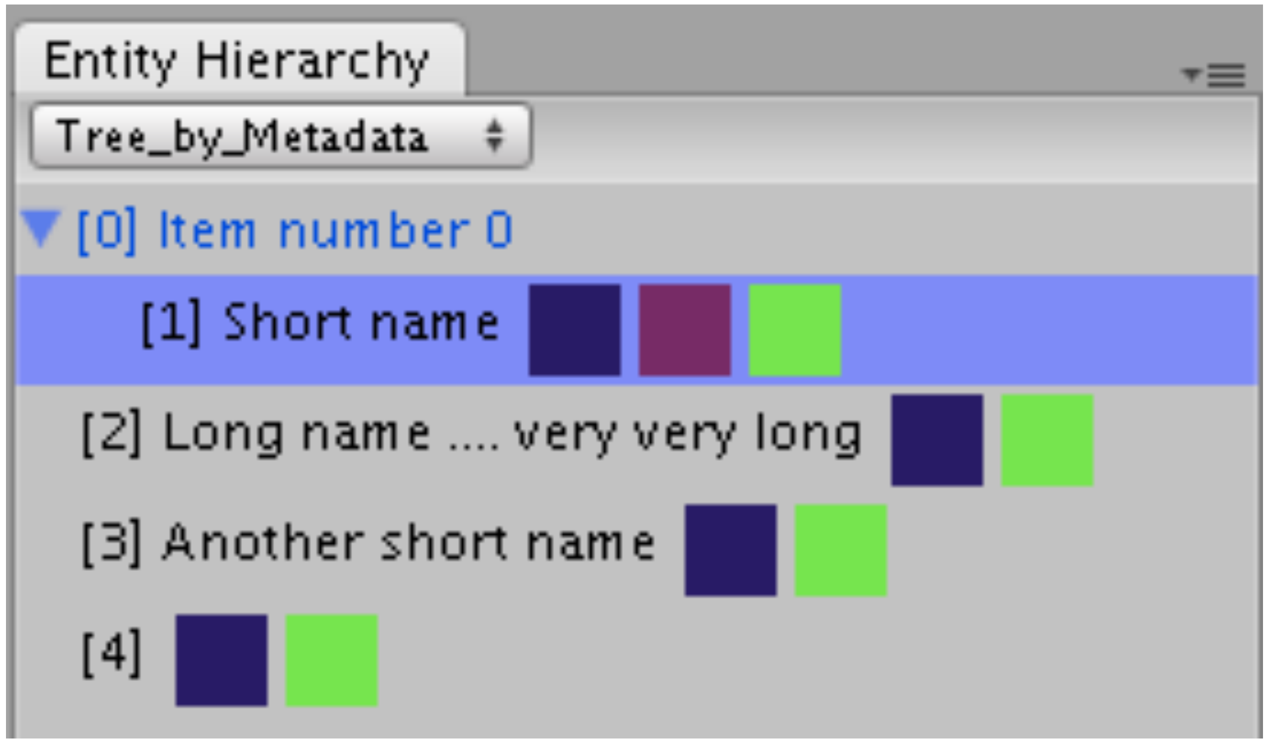
Script: UnityMBEntityMa

FAKE component to workaround Unity's bu

Hierarchy Window



Hierarchy Window



Entity Hierarchy

Tree_by_Metadata

[0] Item number 0

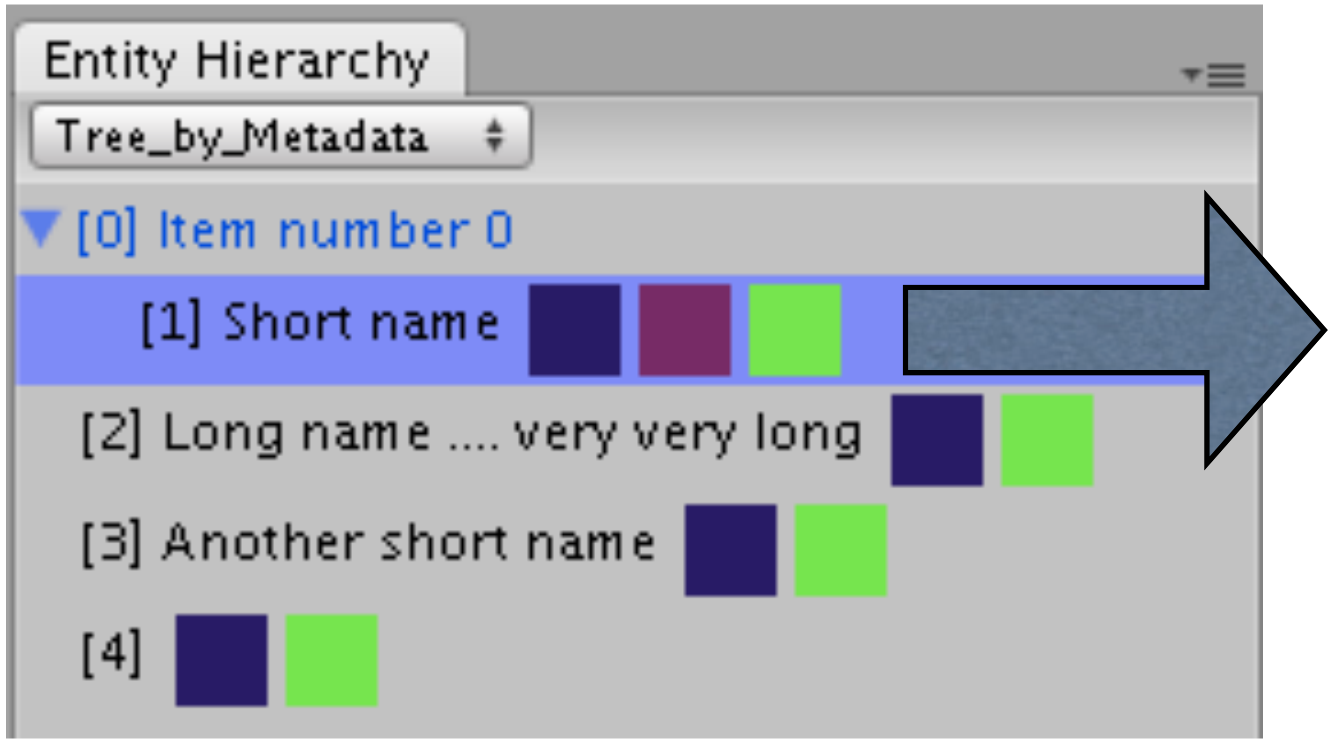
[1] Short name

[2] Long name .... very very long

[3] Another short name

[4]

Hierarchy Window



Entity Hierarchy

Tree_by_Metadata

[0] Item number 0

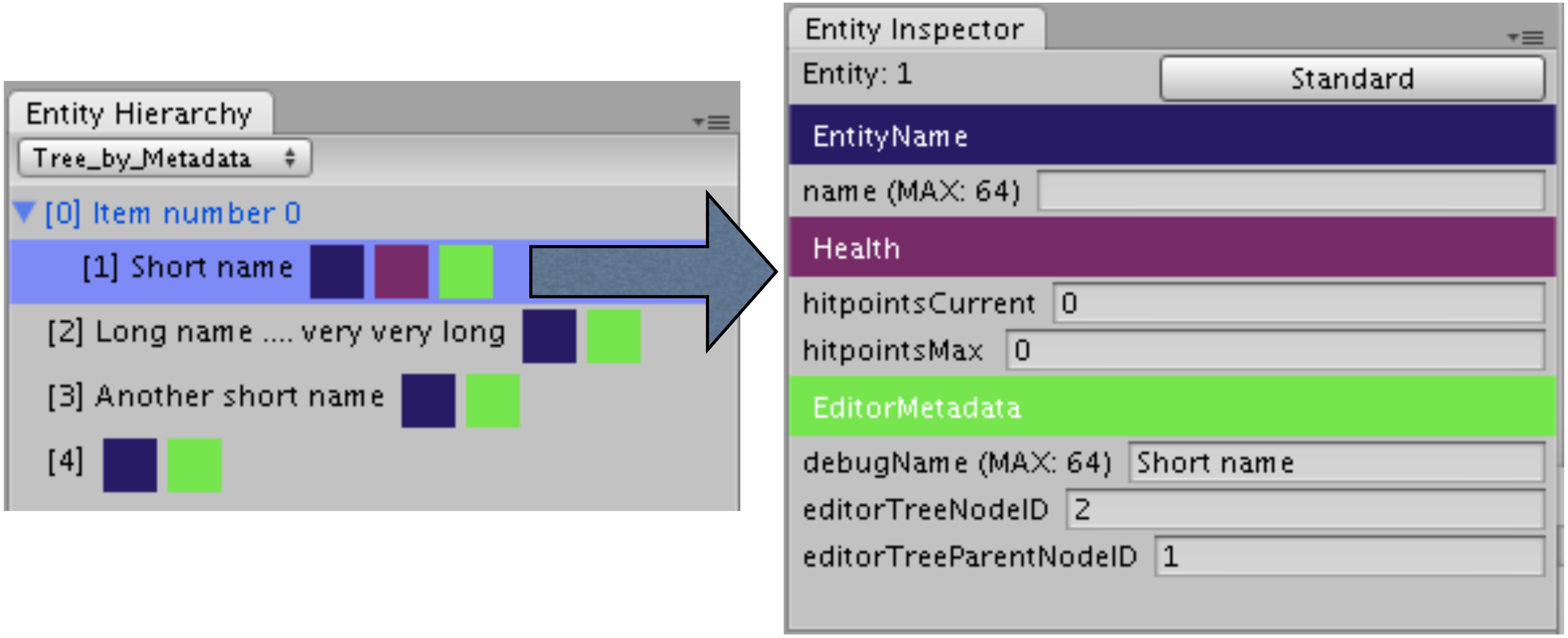
[1] Short name

[2] Long name .... very very long

[3] Another short name

[4]

Hierarchy Window



Entity Hierarchy

Tree_by_Metadata

- [0] Item number 0
  - [1] Short name 🟦 🟪 🟩
- [2] Long name .... very very long 🟦 🟩
- [3] Another short name 🟦 🟩
- [4] 🟦 🟩

Entity Inspector

Entity: 1 — Standard

EntityName

name (MAX: 64)	

Health

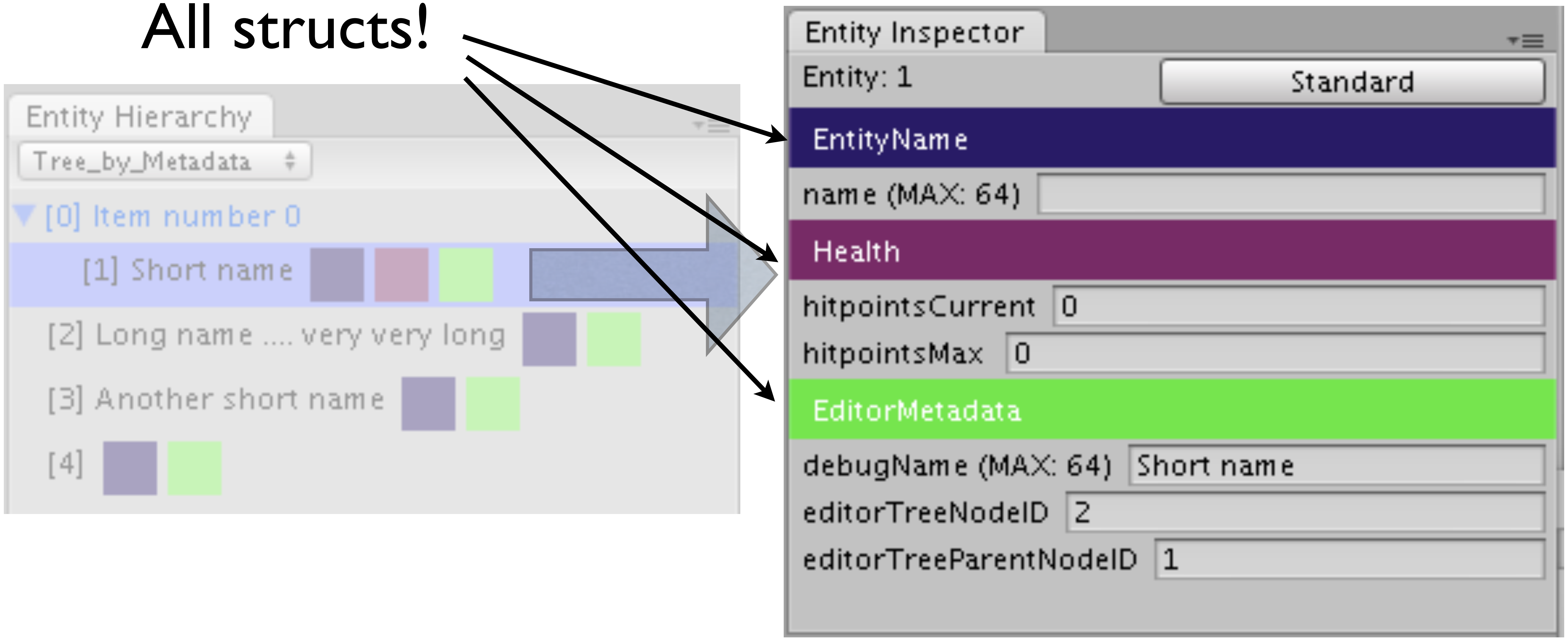
hitpointsCurrent	0
hitpointsMax	0

EditorMetadata

debugName (MAX: 64)	Short name
editorTreeNodeID	2
editorTreeParentNodeID	1

Hierarchy Window

All structs!



Entity: 1

EntityName

name (MAX: 64)

Health

hitpointsCurrent 0

hitpointsMax 0

EditorMetadata

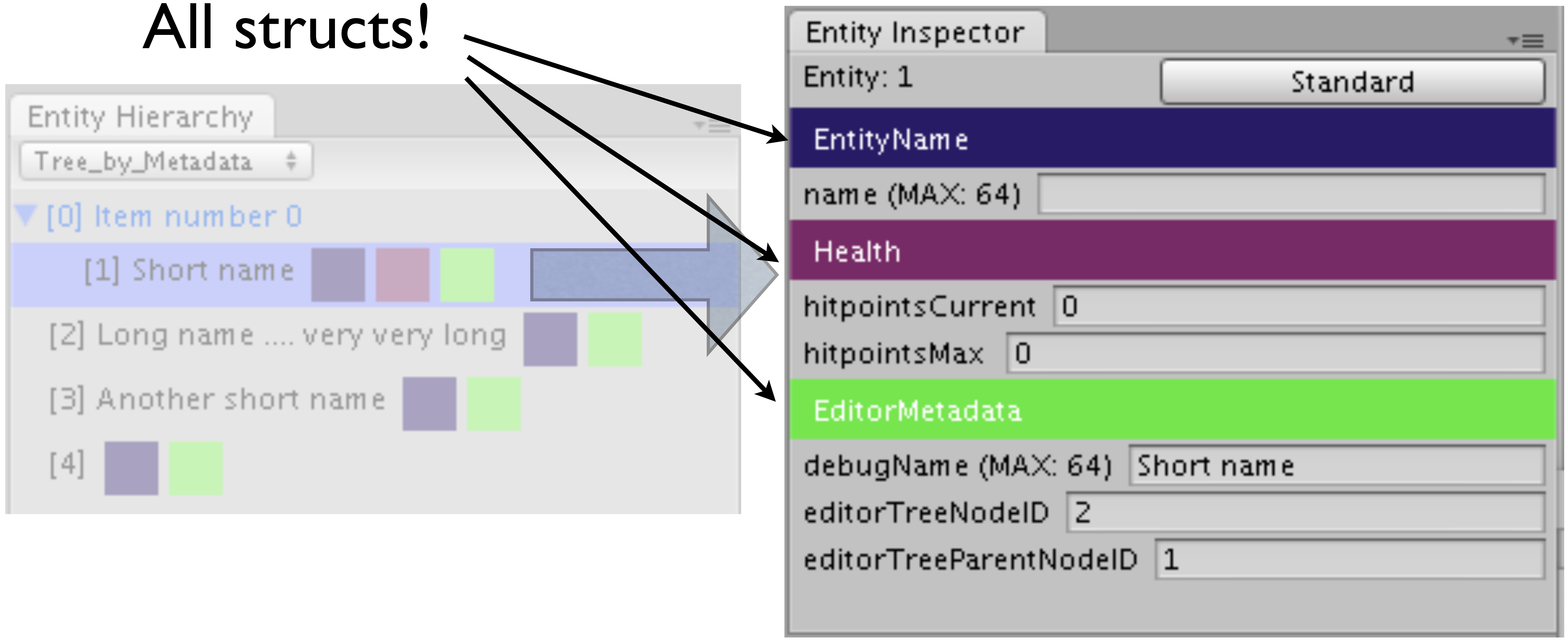
debugName (MAX: 64) Short name

editorTreeNodeID 2

editorTreeParentNodeID 1

Hierarchy Window

All structs!



[0] Item number 0
[1] Short name
[2] Long name .... very very long
[3] Another short name
[4]

Entity: 1

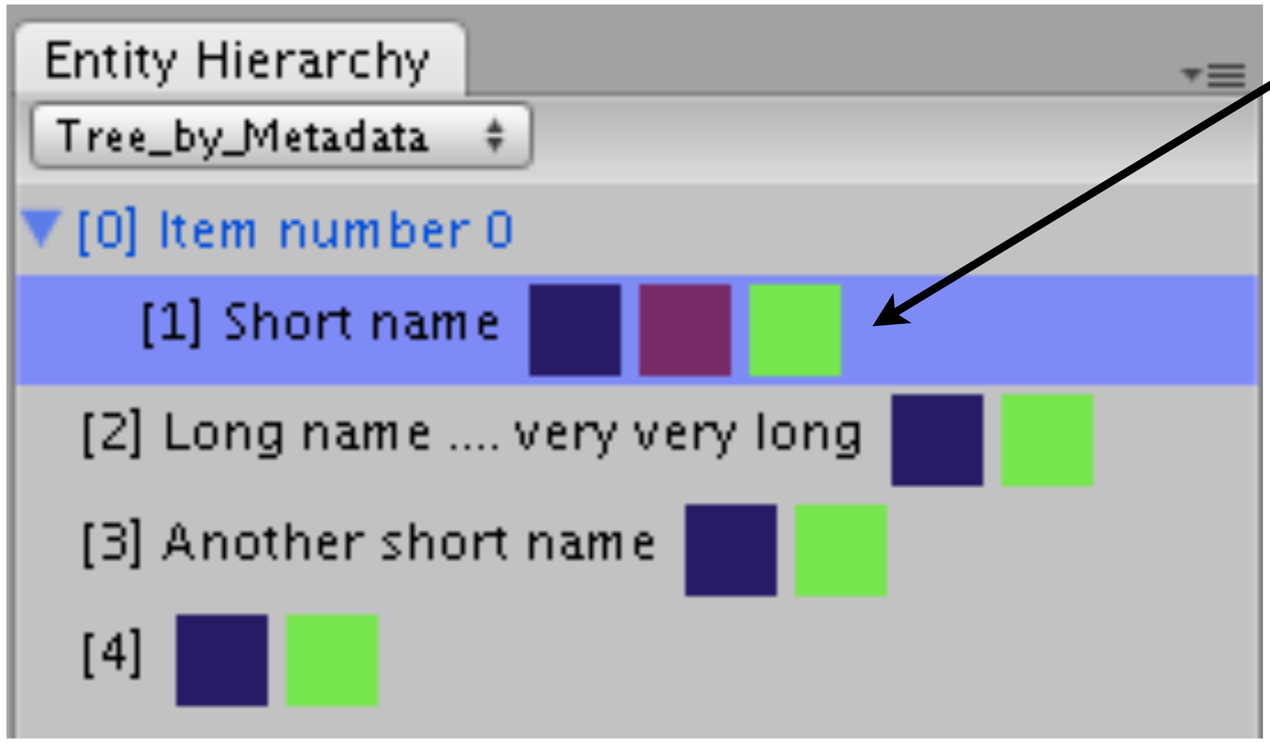
EntityName
name (MAX: 64)

Health
hitpointsCurrent 0
hitpointsMax 0

EditorMetadata
debugName (MAX: 64) Short name
editorTreeNodeID 2
editorTreeParentNodeID 1

...no source code allowed here!

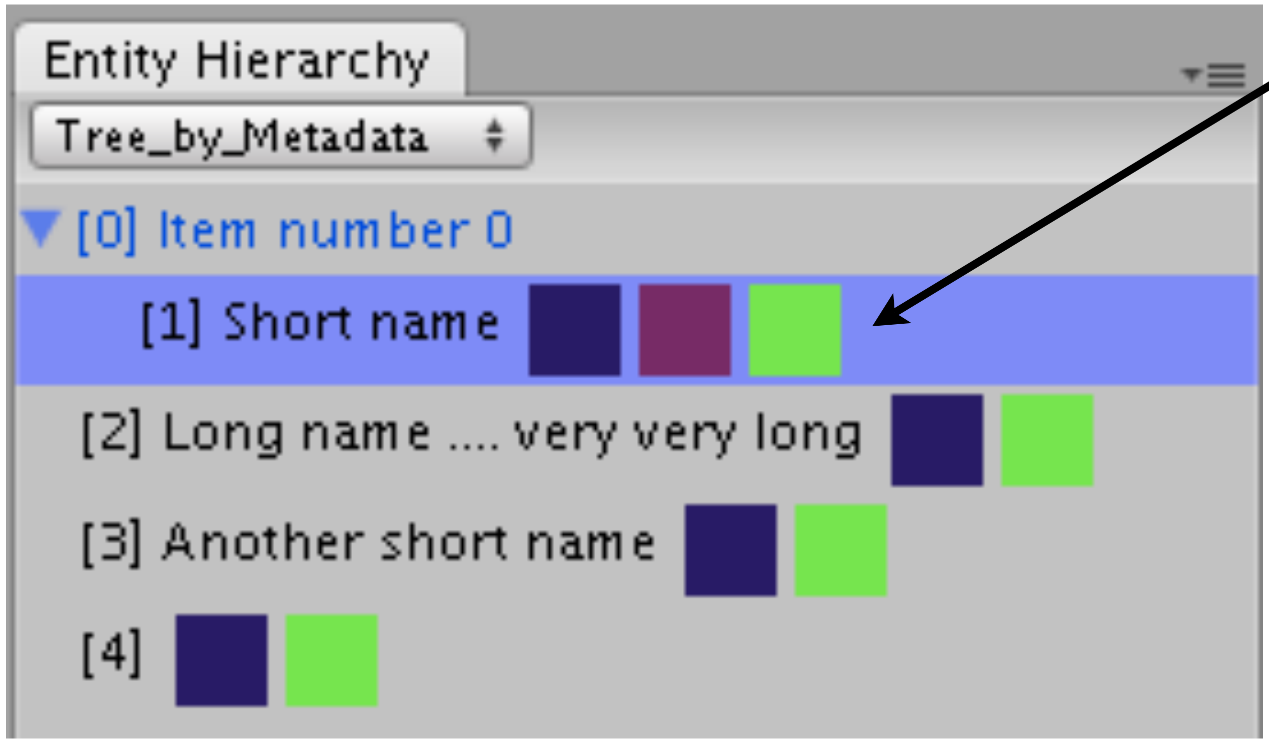
Hierarchy Window



[0] Item number 0
[1] Short name
[2] Long name .... very very long
[3] Another short name
[4]

Tree without Transforms!

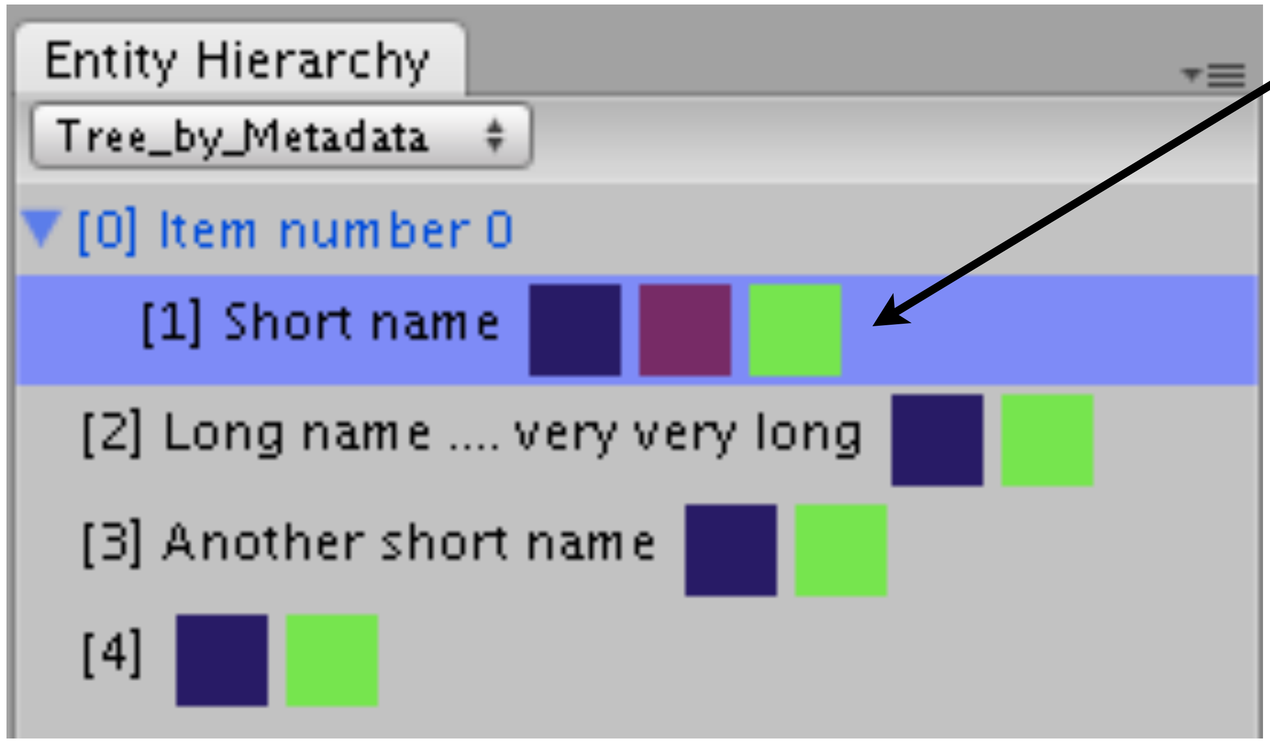
Hierarchy Window



Tree without Transforms!

... arrange however you like,
DOES NOT BREAK
Physics/Render/AI/Logic/etc

Hierarchy Window



Entity Hierarchy

Tree_by_Metadata

- [0] Item number 0
  - [1] Short name
- [2] Long name .... very very long
- [3] Another short name
- [4]

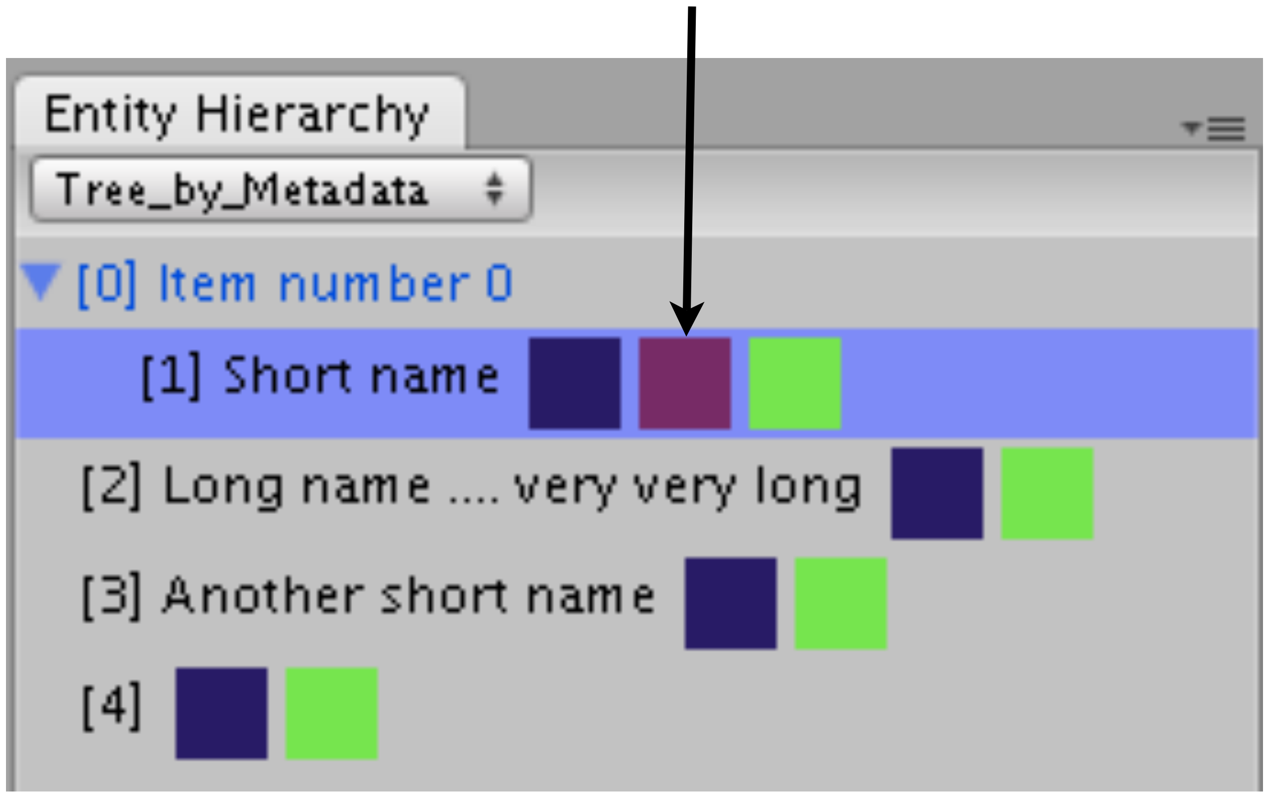
Tree without Transforms!

... arrange however you like,
DOES NOT BREAK
Physics/Render/AI/Logic/etc

... don't need a fake invisible
GameObject to group things!

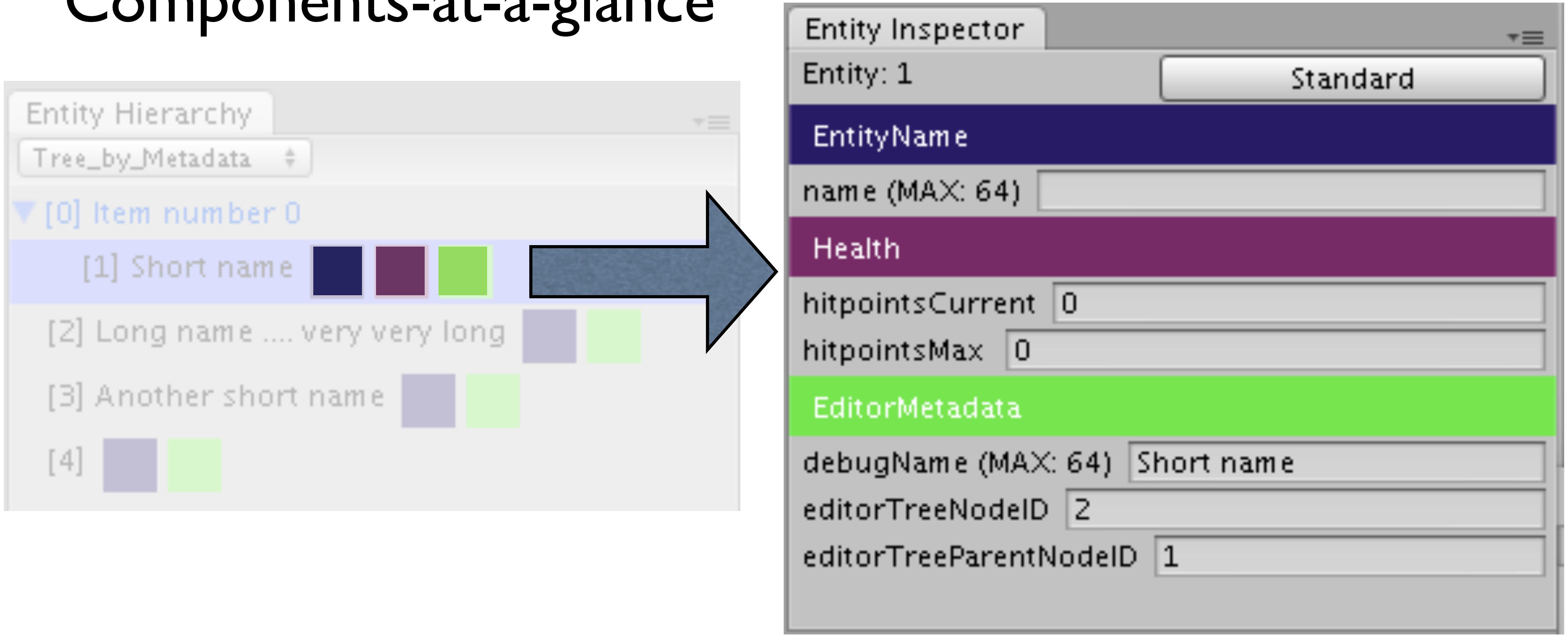
Hierarchy Window

Components-at-a-glance



Hierarchy Window

Components-at-a-glance



Entity Hierarchy

Tree_by_Metadata

[0] Item number 0

[1] Short name 🟦 🟪 🟩

[2] Long name .... very very long 🟦 🟩

[3] Another short name 🟦 🟩

[4] 🟦 🟩

Entity Inspector

Entity: 1 — Standard

EntityName

name (MAX: 64)

Health

hitpointsCurrent 0

hitpointsMax 0

EditorMetadata

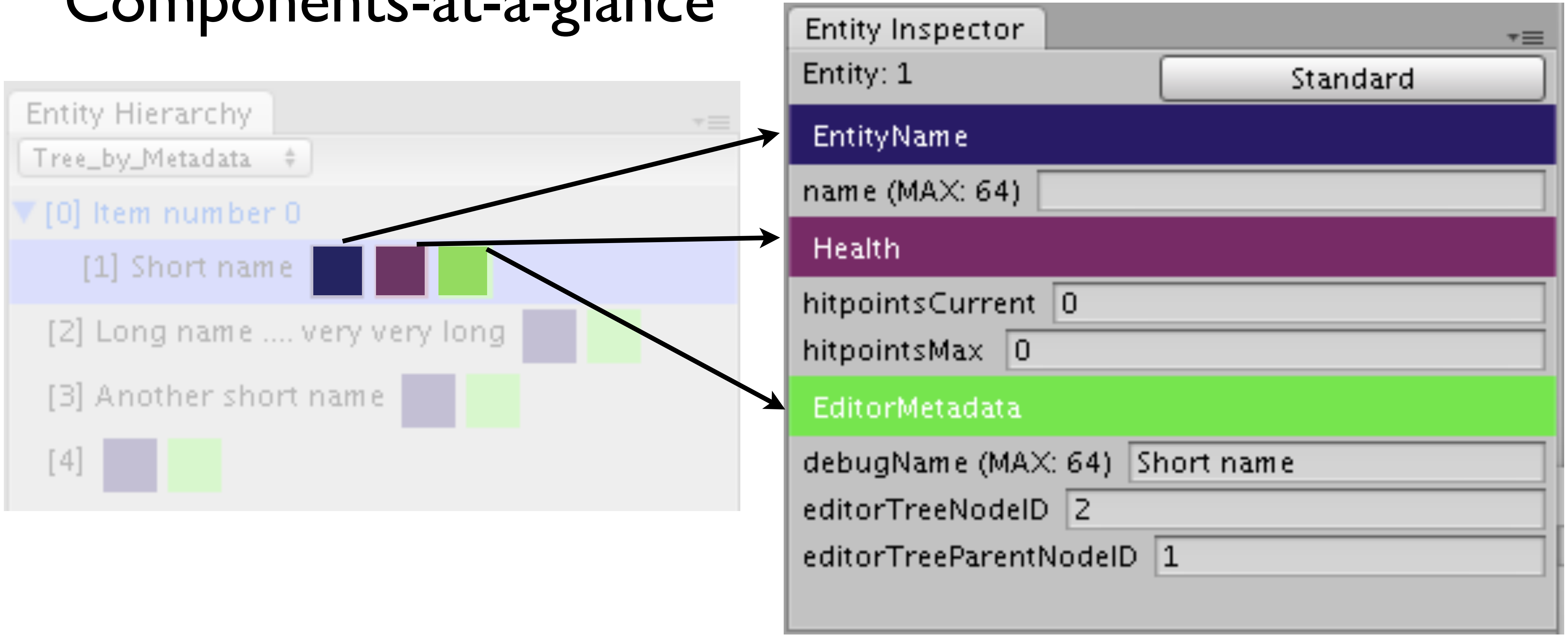
debugName (MAX: 64) Short name

editorTreeNodeID 2

editorTreeParentNodeID 1

Hierarchy Window

Components-at-a-glance



Entity Hierarchy

Tree_by_Metadata

[0] Item number 0

[1] Short name

[2] Long name .... very very long

[3] Another short name

[4]

Entity Inspector

Entity: 1 | Standard

EntityName

name (MAX: 64)

Health

hitpointsCurrent 0

hitpointsMax 0

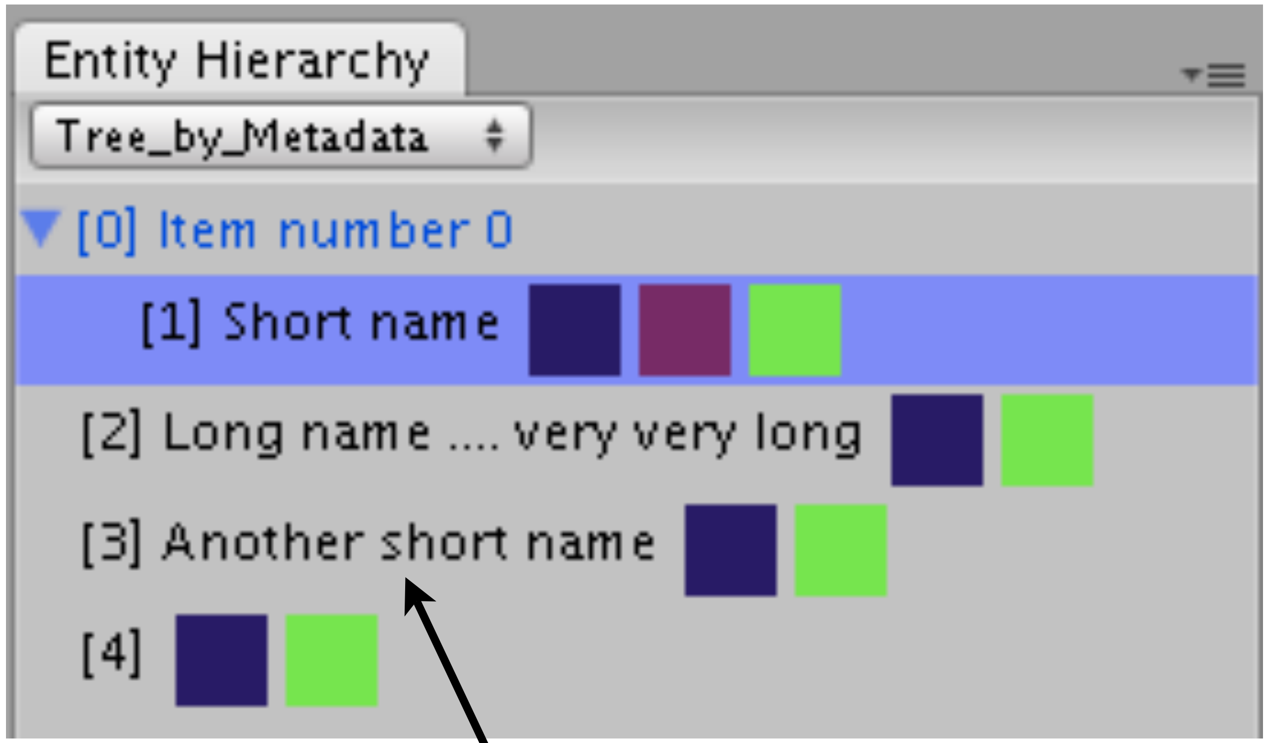
EditorMetadata

debugName (MAX: 64) Short name

editorTreeNodeID 2

editorTreeParentNodeID 1

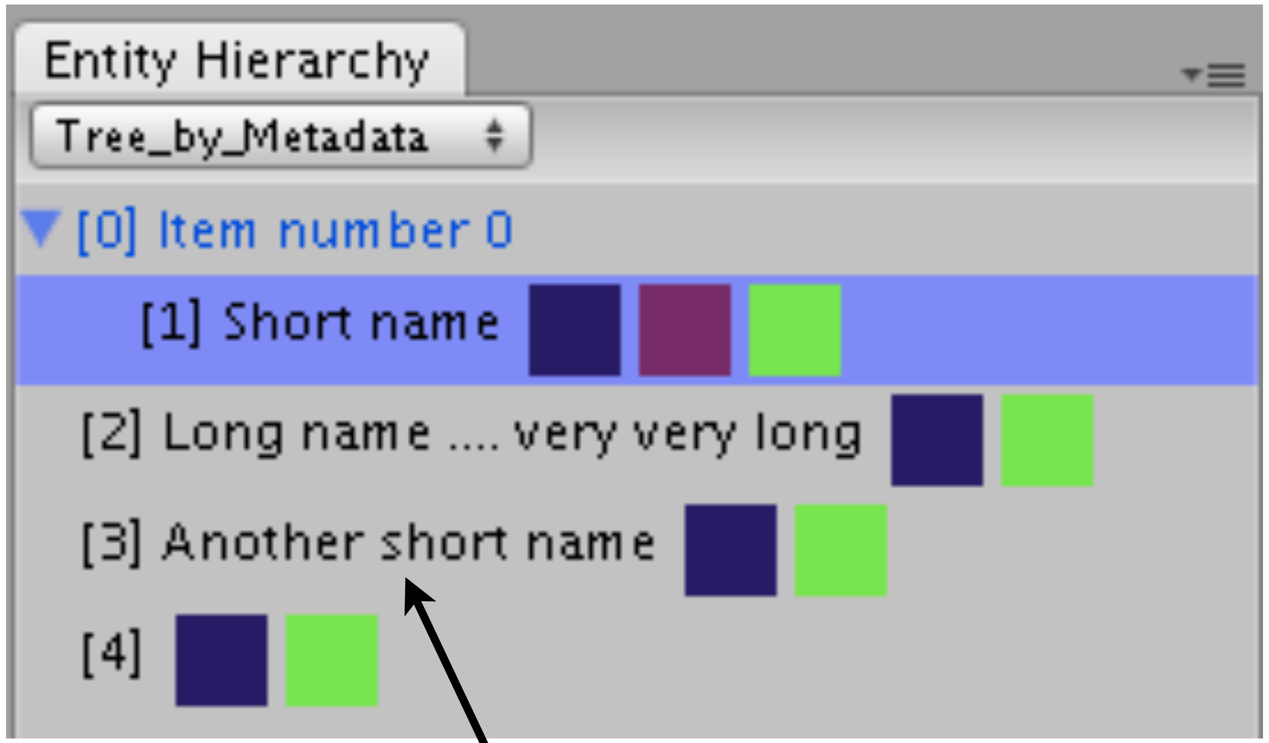
Hierarchy Window



[0] Item number 0
[1] Short name
[2] Long name .... very very long
[3] Another short name
[4]

Use anything as “name”

Hierarchy Window

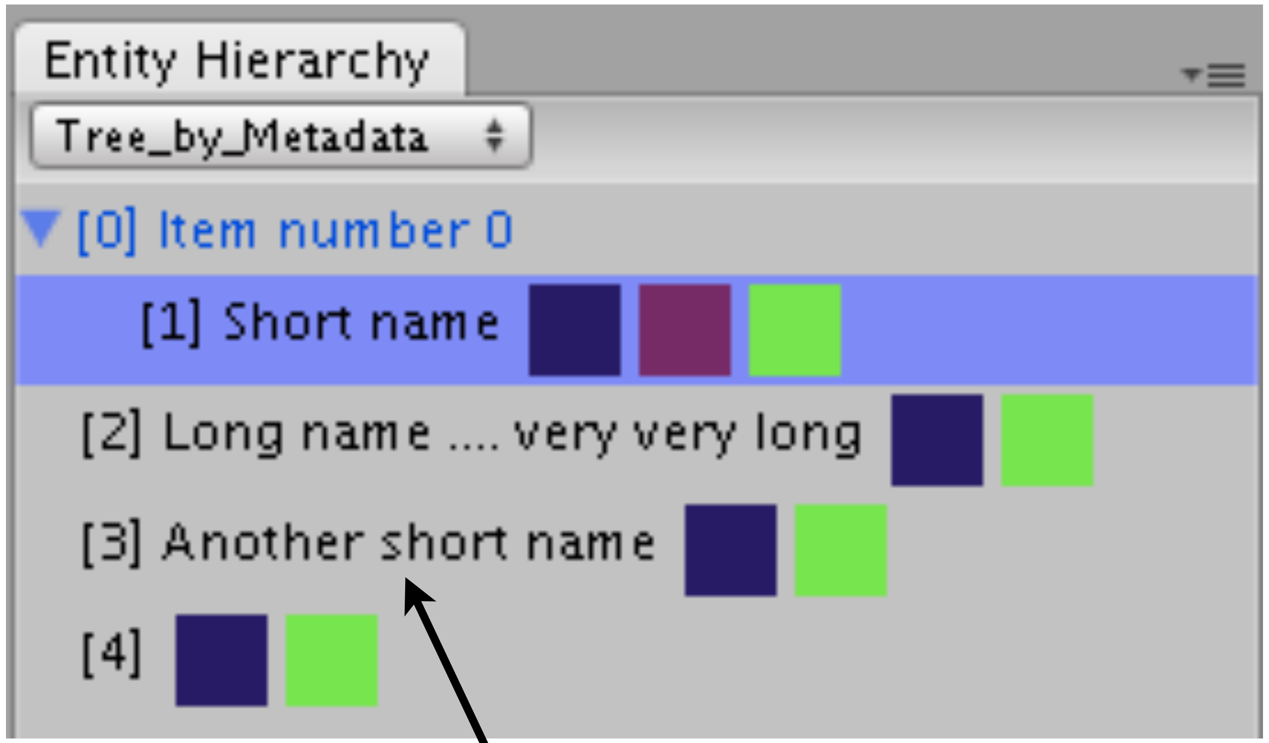


[0] Item number 0
[1] Short name
[2] Long name .... very very long
[3] Another short name
[4]

Have more than one name per object (Entity)!

Use anything as “name”

Hierarchy Window



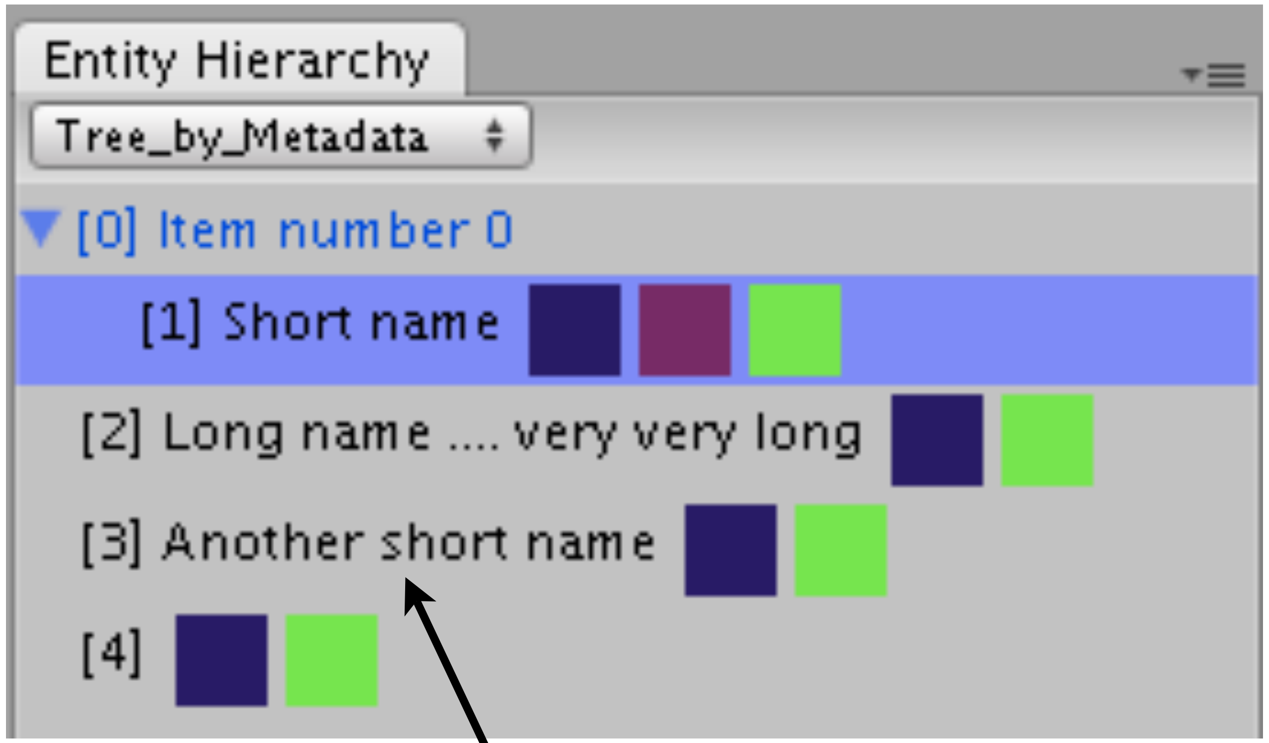
[0] Item number 0
[1] Short name
[2] Long name .... very very long
[3] Another short name
[4]

Have more than one name per object (Entity)!

... DEBUG name

Use anything as “name”

Hierarchy Window



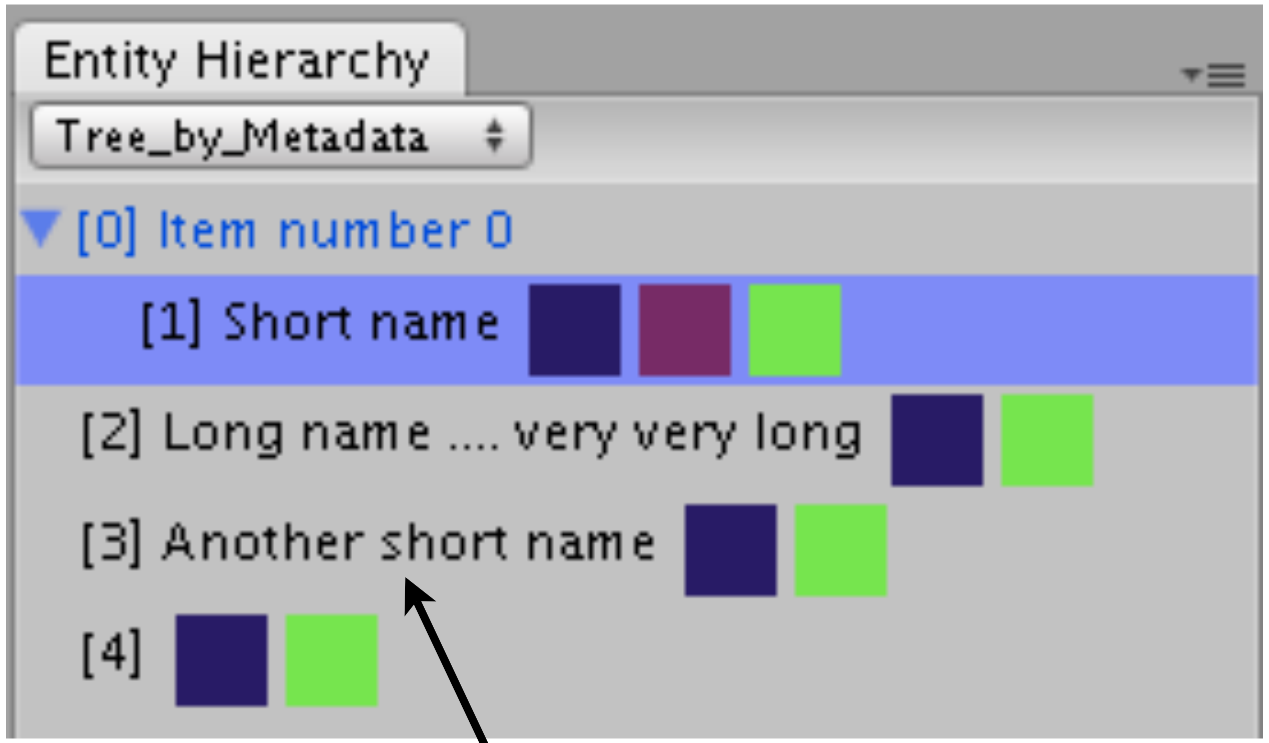
Use anything as "name"

Have more than one name per object (Entity)!

... DEBUG name

... + player-GUI name

Hierarchy Window



Have more than one name per object (Entity)!

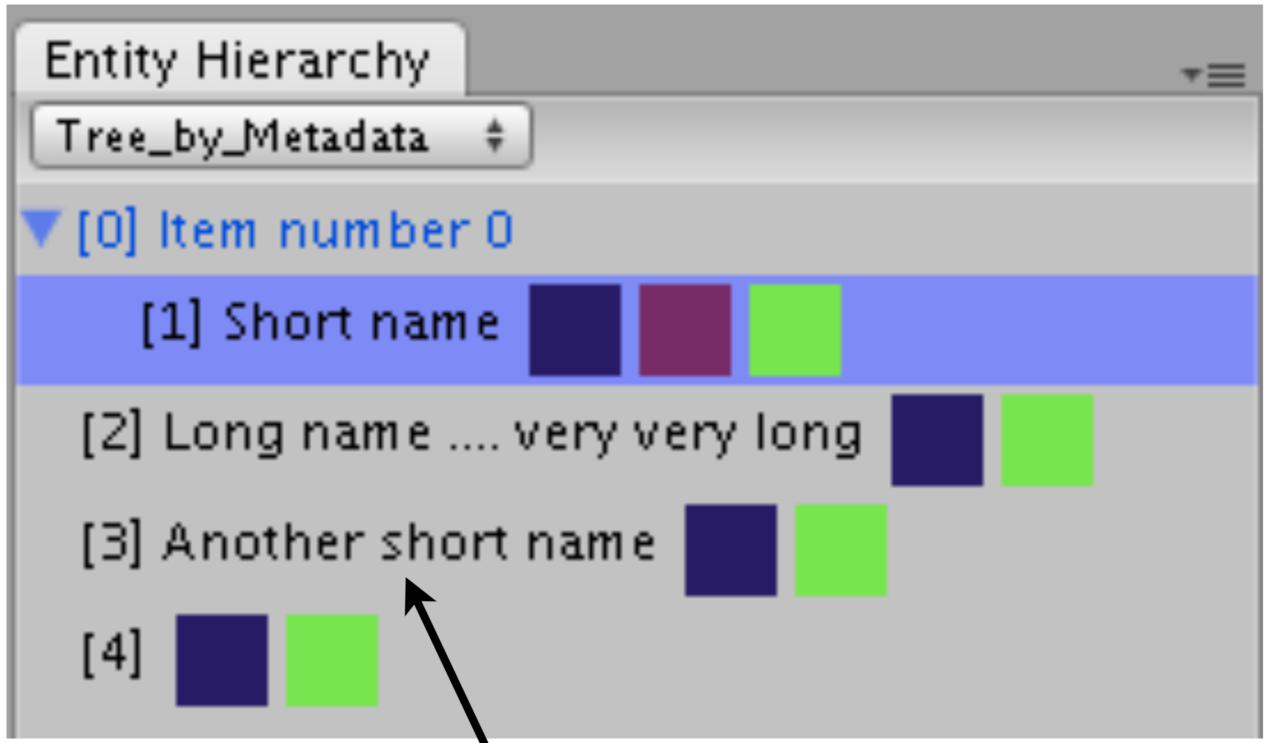
... DEBUG name

... + player-GUI name

... + script-hardcoded name

Use anything as "name"

Hierarchy Window



[0] Item number 0
[1] Short name
[2] Long name .... very very long
[3] Another short name
[4]

Use anything as "name"

Have more than one name per object (Entity)!

... DEBUG name

... + player-GUI name

... + script-hardcoded name

... etc

Progress so far...
... in Source Code

Source Code

- All Components are C# struct
- structs stored in massive raw [arrays]
- Generics guarantee safe + fast access
- Components can be fast-copied by C#
- Everything saved to Unity Serialization

Problem: no "return ref"

```
struct Position;

private Position[1000] allPositions;

public *Position GetPosition( ID identifier );
```

Problem: no “return ref”

struct Position;

private Position[1000] allPositions;

public ~~*Position~~ GetPosition(ID identifier);

Workarounds

- C# delegate ?
- private -> public ?
- C# Enumerable ?
- C# 7 <--- hope and pray

Unity problem: Integrate Serialization

(very quick overview; needs a whole talk in itself)

Unity Serialization: ISerializationCallbackReceiver

- Use C# Marshal to convert your game-data to/from raw byte[]
- Save byte[] as: [SerializeField] private byte[]
- Unity calls incorrectly in 10% of cases
  - => monitor + fix on case-by-case basis
- In some edge-cases, Unity doesn’t serialize
  - DEAL WITH IT. (i.e. “not-solved-yet”)

Summary...

1. Performance improves on Unity

With 100,000 GameObjects ... 10-30x faster

2. Replacement for Inspector works/usable

But nothing special (yet)

3. C# code is pretty clean, simple to write

EASIER to write if/when C# v7 is added to Unity

Could run MUCH faster if/when C# v7 is added to Unity

4. Many kludges, hacky source code

But ... IT WORKS!
(proves the concept fairly well)

Next steps...

Future features

- Re-implement prefabs BUT PROPERLY!
  - Drag/drop "prefab editor"
- Visual scripting
  - Maybe debugging only; Maybe coding too
- Database-backed entity store
  - SQL FTW (...maybe)

Features (cont'd)

- AWESOME FUNKTASTIC VISUALIZR!
  - (not entirely sure how this'll look yet)
- Very-High-speed version
  - Complex mem-management algorithms
- Auto-integrate with MonoBehaviour
  - Control Unity's Renderer; Physics; etc

Towards the Asset Store...

- Start using it in my own (hobby) games
- Early-access via Kickstarter, Patreon, or similar
- Work towards: Production build (good enough for shipping commercial games)

@t_machine_org
adam.m.s.martin@gmail.com

http://aliqua.org

More info coming soon ... subscribe for updates

* indicates required

Email Address *

(temporary holding page for following the project. Average of 1-2 emails per month)