Memory Management in Ogre 2.x

by Matías Nazareth Goldberg

This is a short overview on how the SoA memory manager works in Ogre 2.0. The code can be very intimidating at first, but once you know what's going on, it's easy to folow. I made this document for that reason

Memory Management in Ogre 2.x

NodeMemoryManager

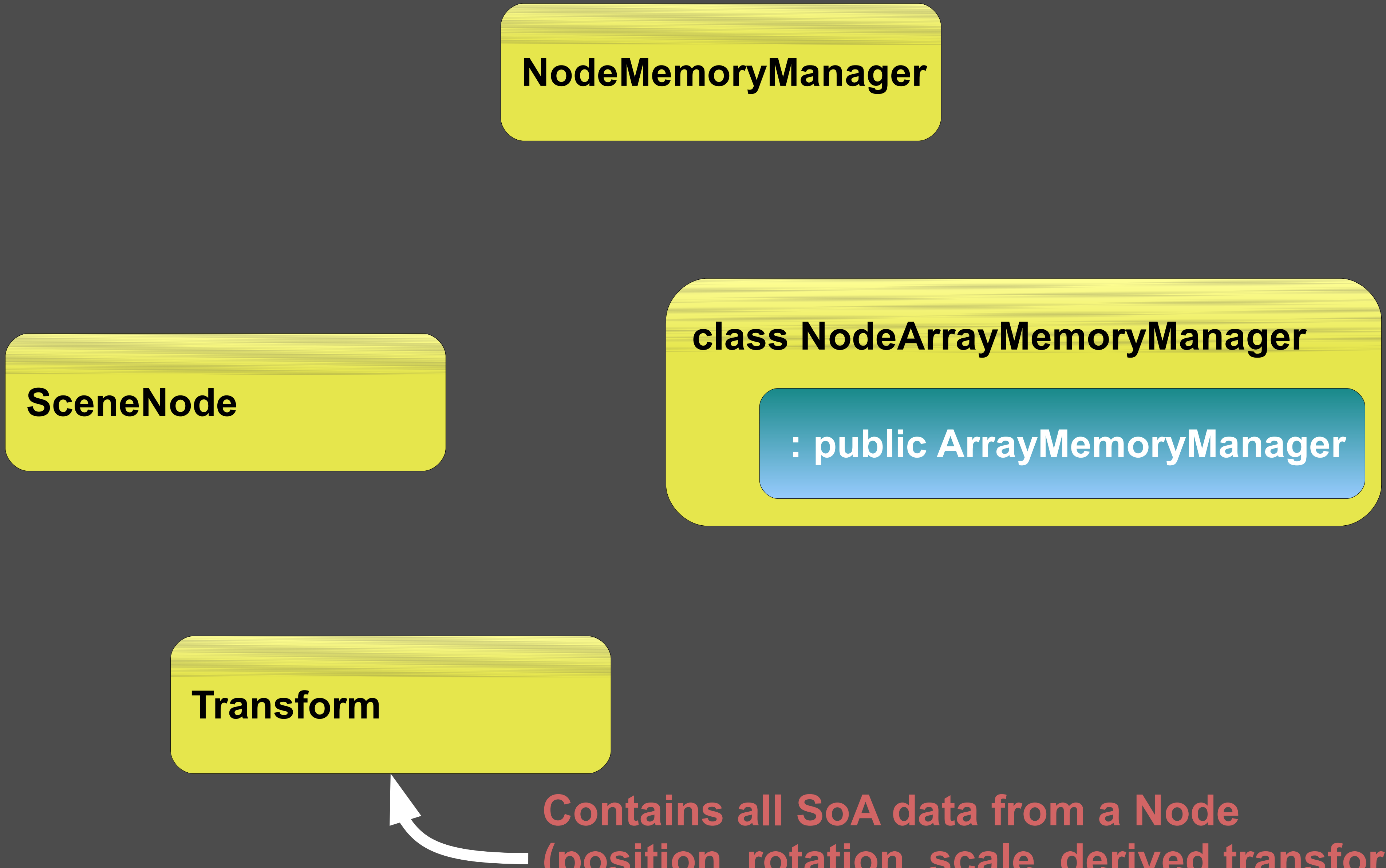
SceneNode

class NodeArrayMemoryManager

: public ArrayMemoryManager

Transform

Memory Management in Ogre 2.x



NodeMemoryManager

SceneNode

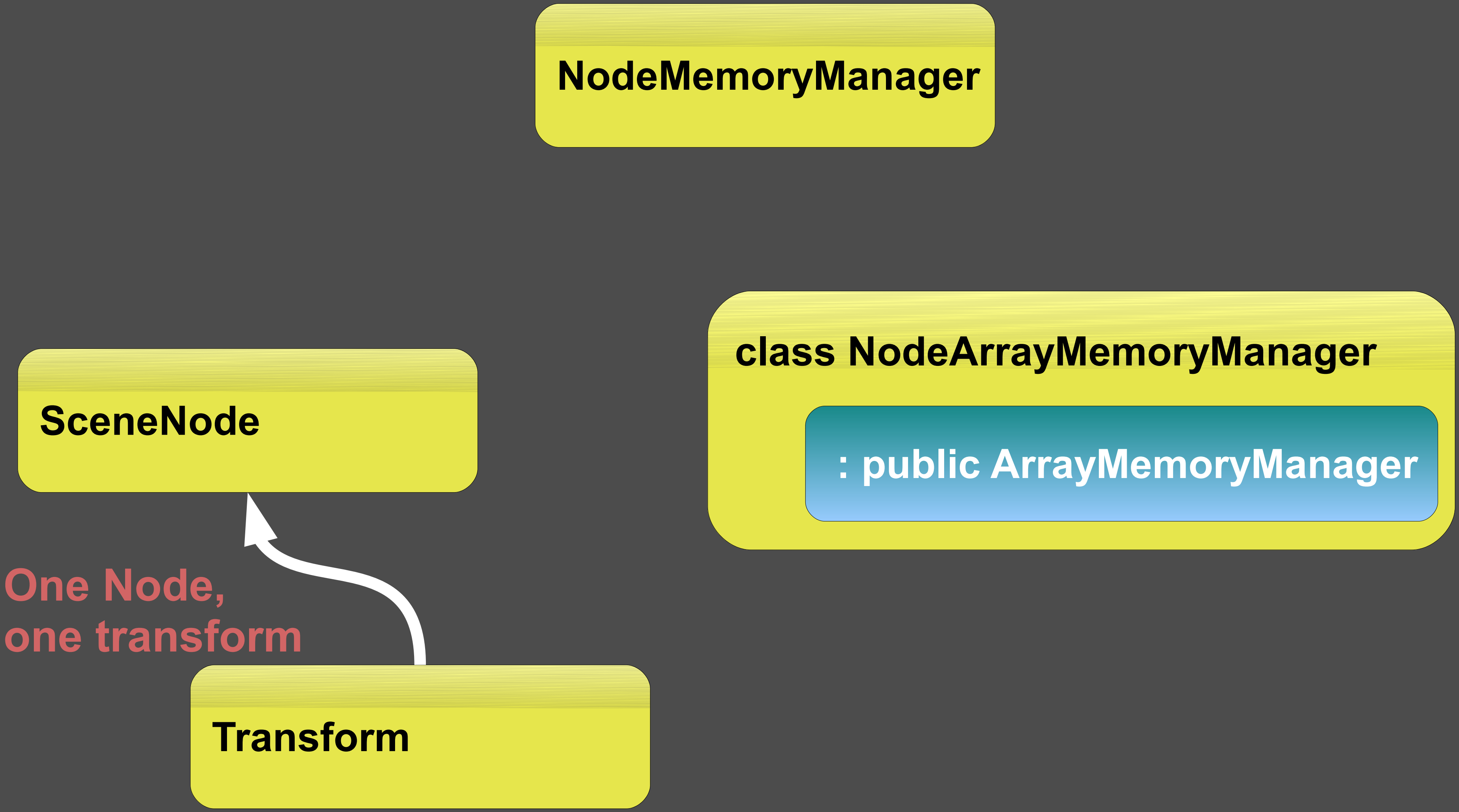
class NodeArrayMemoryManager

: public ArrayMemoryManager

Transform

Contains all SoA data from a Node (position, rotation, scale, derived transform matrix4x4, etc)

Memory Management in Ogre 2.x



NodeMemoryManager

SceneNode

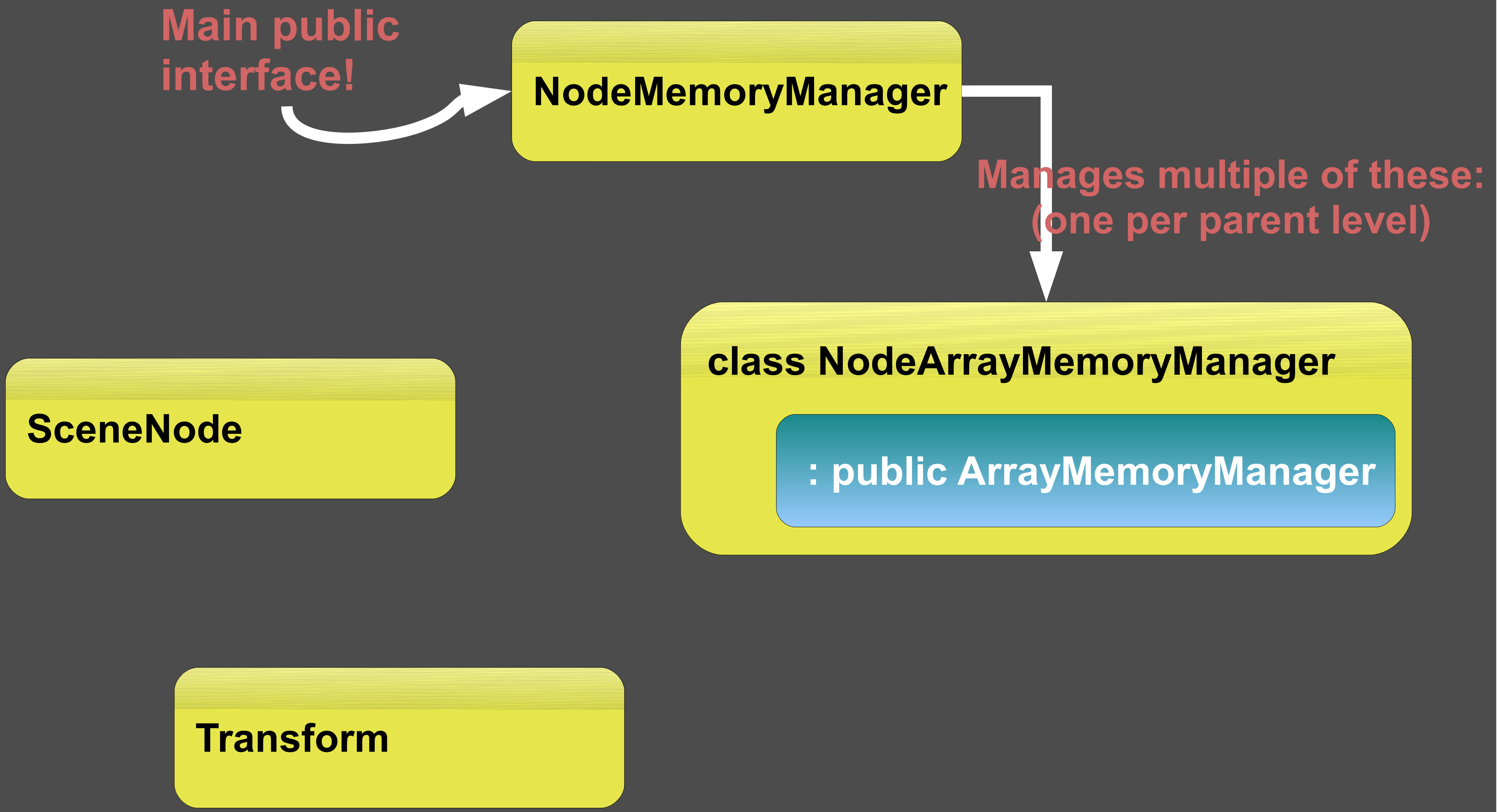
class NodeArrayMemoryManager

: public ArrayMemoryManager

One Node,
one transform

Transform

Memory Management in Ogre 2.x



Main public interface!

NodeMemoryManager

Manages multiple of these:
(one per parent level)

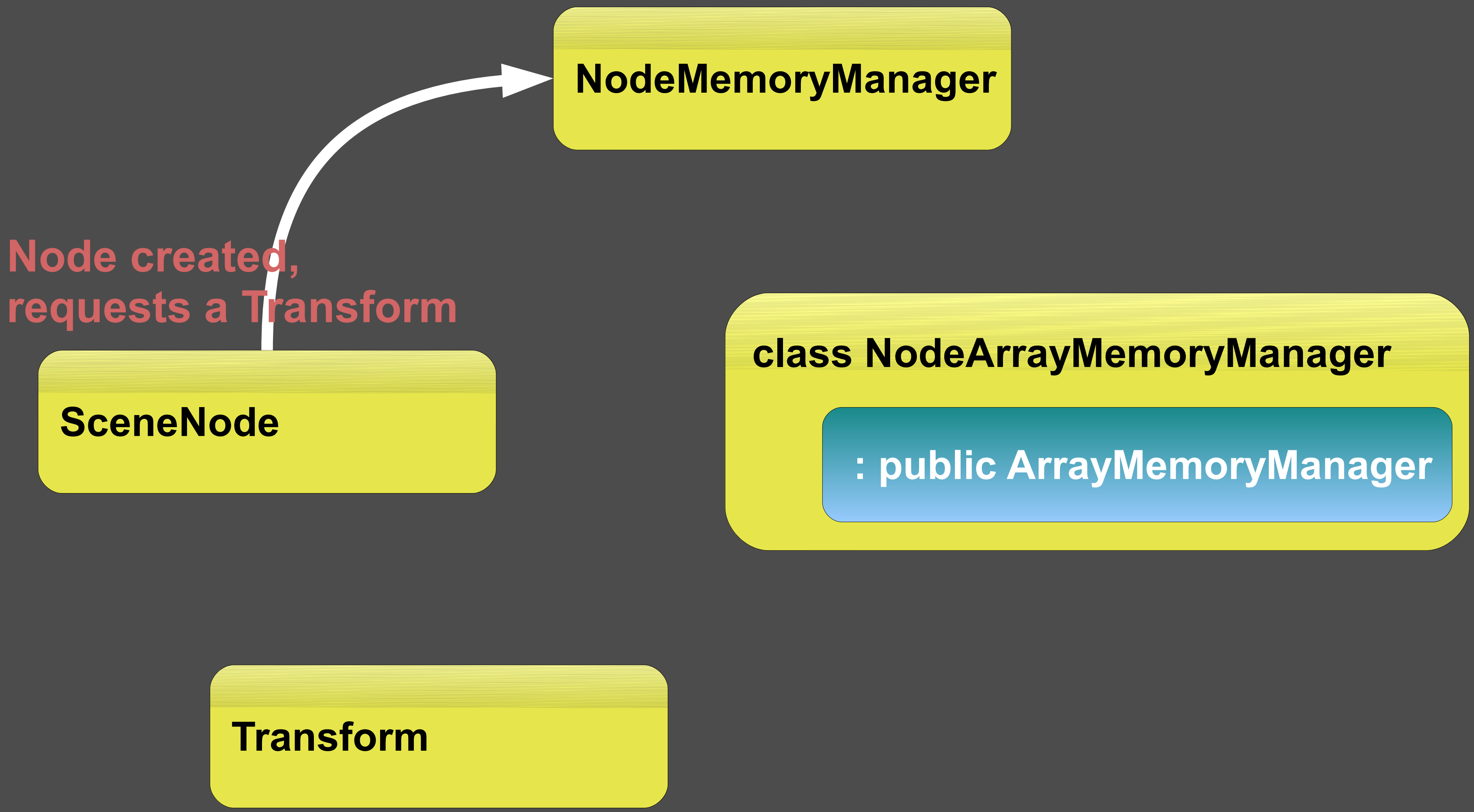
class NodeArrayMemoryManager

: public ArrayMemoryManager

SceneNode

Transform

Memory Management in Ogre 2.x



NodeMemoryManager

Node created,
requests a Transform

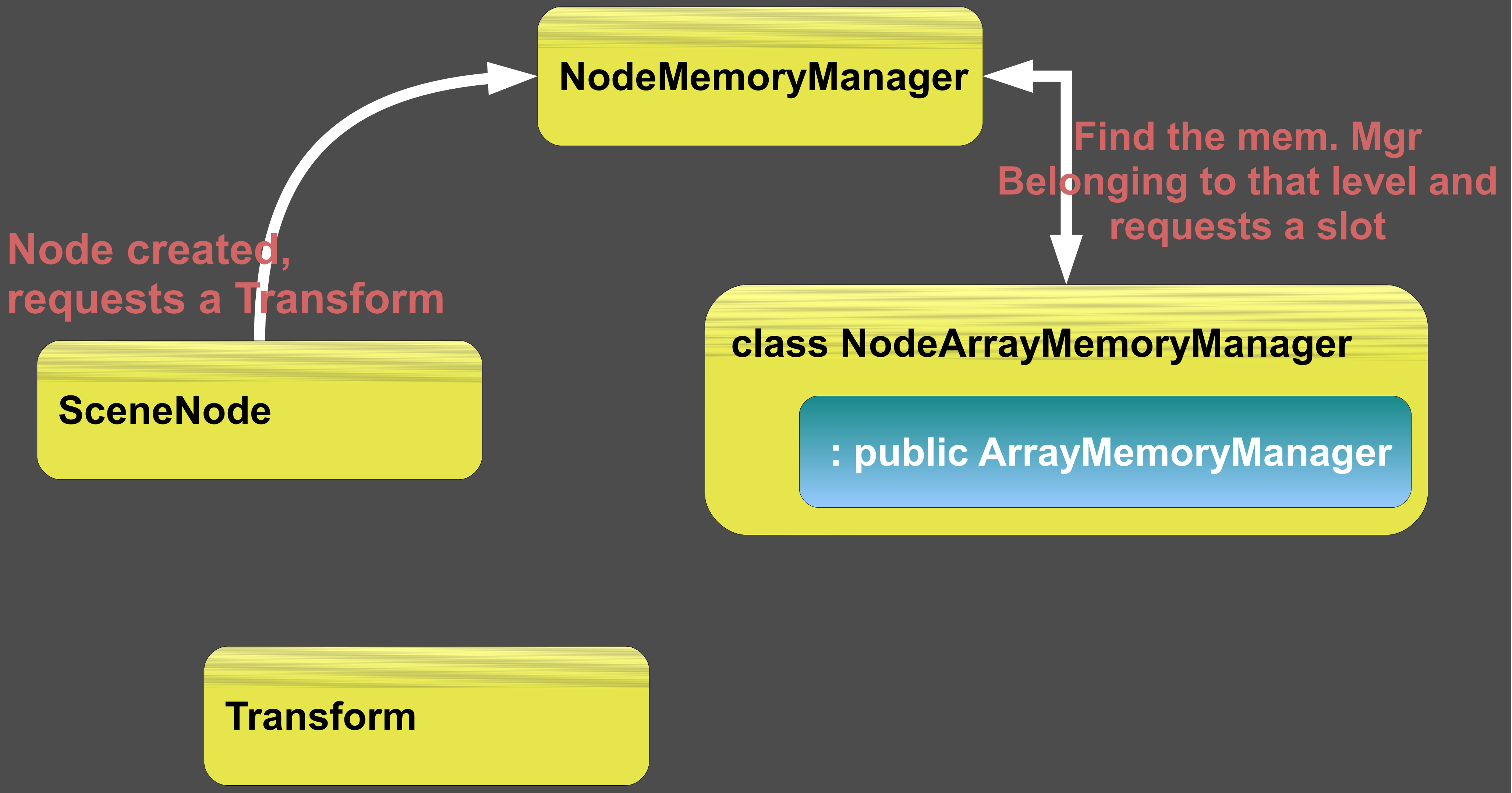
SceneNode

class NodeArrayMemoryManager

: public ArrayMemoryManager

Transform

Memory Management in Ogre 2.x



NodeMemoryManager

Find the mem. Mgr
Belonging to that level and
requests a slot

Node created,
requests a Transform

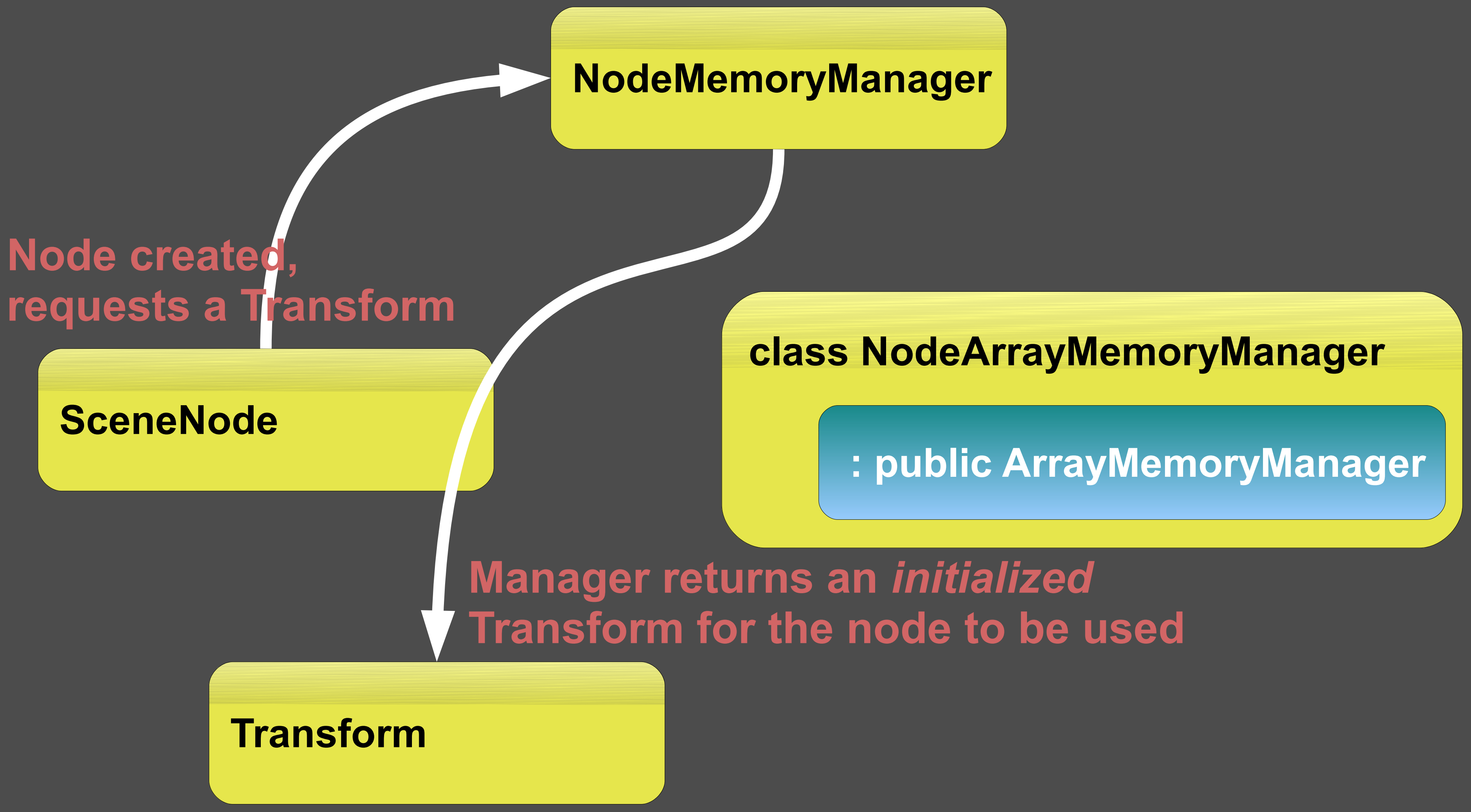
SceneNode

class NodeArrayMemoryManager

: public ArrayMemoryManager

Transform

Memory Management in Ogre 2.x



NodeMemoryManager

Node created, requests a Transform

class NodeArrayMemoryManager

: public ArrayMemoryManager

SceneNode

Manager returns an *initialized* Transform for the node to be used

Transform

Memory Management in Ogre 2.x

NodeMemoryManager

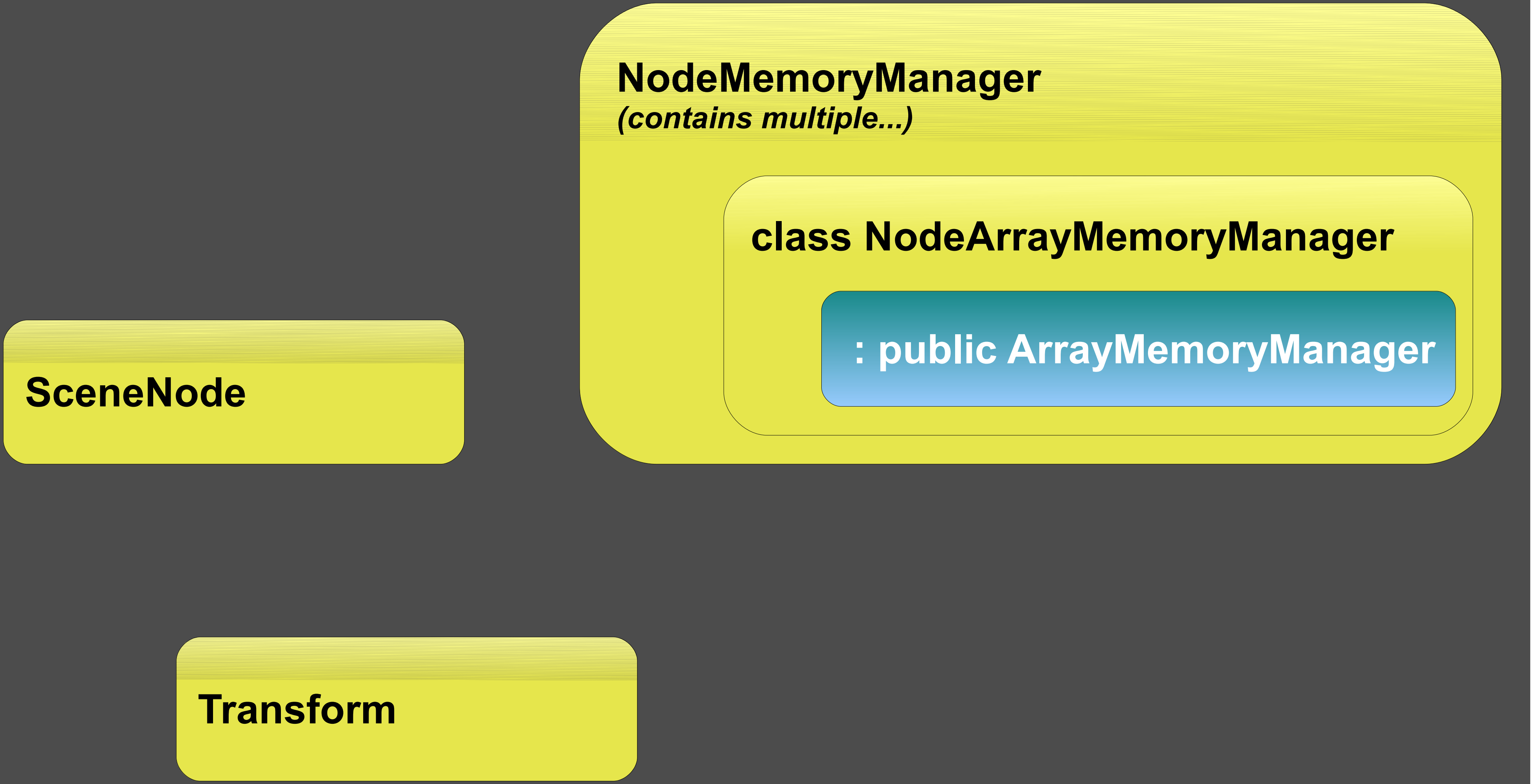
SceneNode

class NodeArrayMemoryManager

: public ArrayMemoryManager

Transform

Memory Management in Ogre 2.x



NodeMemoryManager
(contains multiple...)

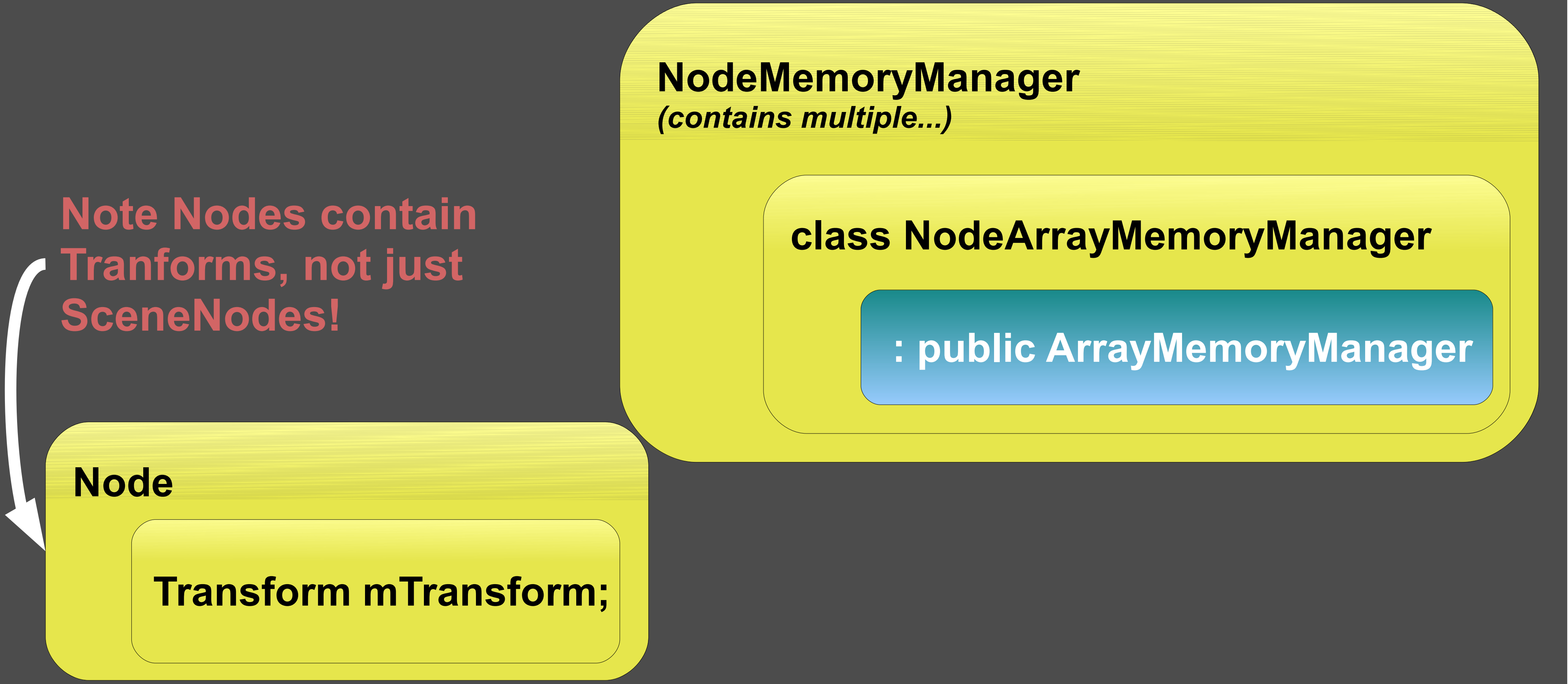
class NodeArrayMemoryManager

: public ArrayMemoryManager

SceneNode

Transform

Memory Management in Ogre 2.x



NodeMemoryManager
(contains multiple...)

class NodeArrayMemoryManager

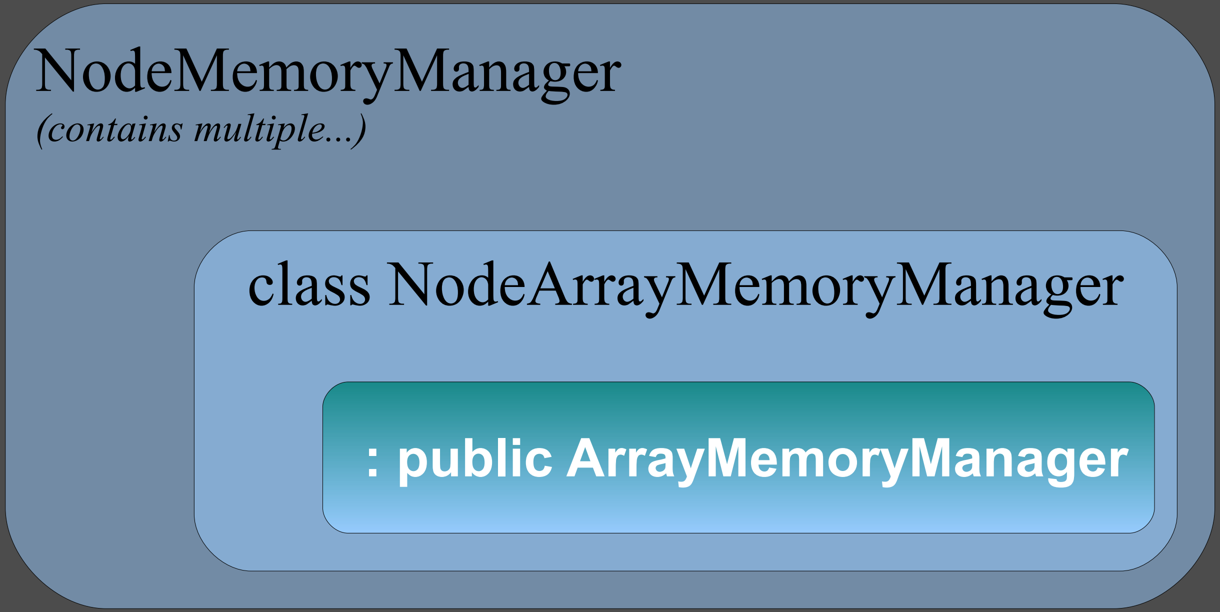
: public ArrayMemoryManager

Note Nodes contain Tranforms, not just SceneNodes!

Node

Transform mTransform;

ArrayMemoryManager



NodeMemoryManager

(contains multiple...)

class NodeArrayMemoryManager

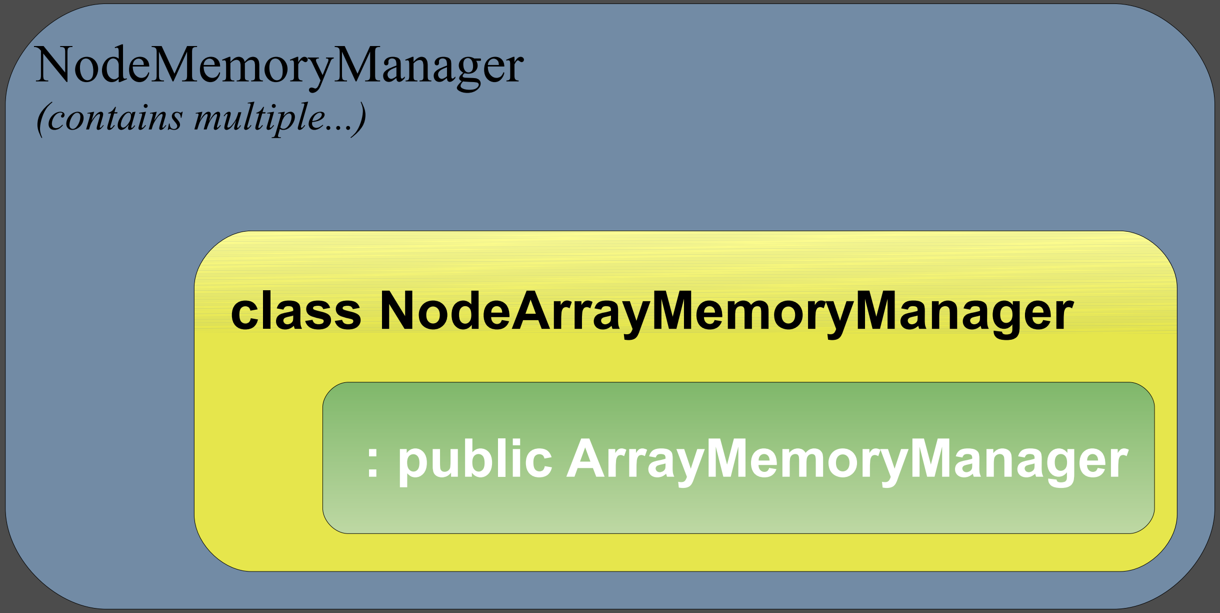
: public ArrayMemoryManager

Abstract system to generically create SIMD-friendly SoA memory. It's used by Node and MovableObject's managers.

Works by requesting a 'slot' and then releasing it when not needed. ArrayMemoryManager can handle cleanups and out of memory situations. However requires a Listener for advanced handling (eg. NodeArrayMemoryManager provides a Listener implementation) otherwise it can only raise exceptions when out of memory.

A 'slot' can be anything: A MovableObject, a Node, etc. and its memory is initialized to all zeroes

NodeArrayMemoryManager



NodeMemoryManager
(contains multiple...)

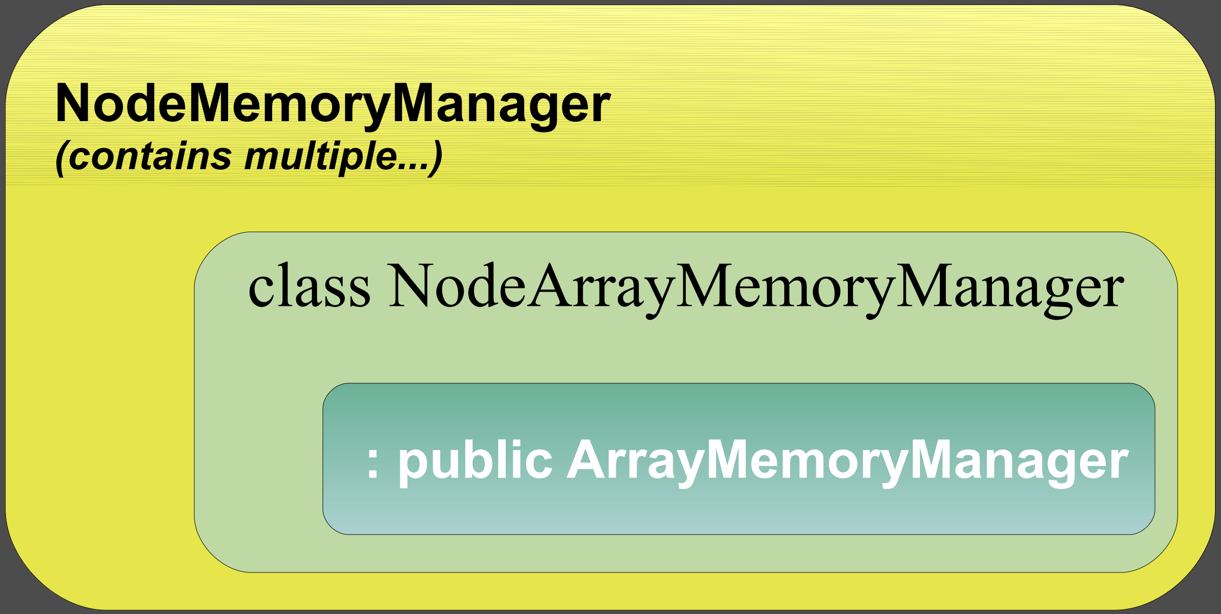
class NodeArrayMemoryManager

: public ArrayMemoryManager

**Specializes to make each 'slot' a Transform container for Nodes.
Initializes all slots/Transforms to default values (not just zeroes).
When a Transform is destroyed/released, some of its data needs to be reset to valid values to mantain *SIMD coherence***

Handles out of memory situations the generic ArrayMemoryManager can't.

NodeMemoryManager



NodeMemoryManager
(contains multiple...)

class NodeArrayMemoryManager

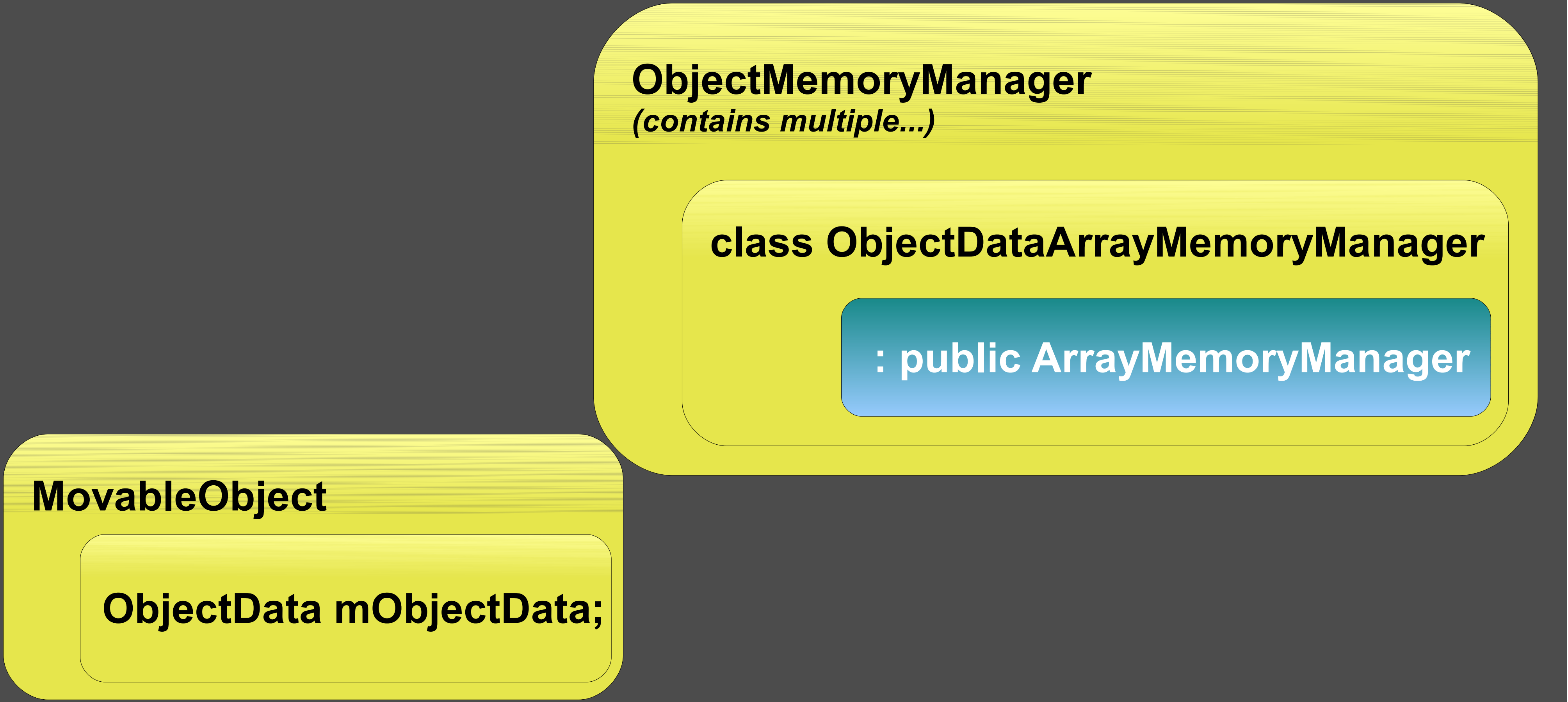
: public ArrayMemoryManager

Main public interface. Manages one NodeArrayMemoryManager per parent level (eg. Root → [child1, child2] → [child2_child3] means 3 levels)

Responsible for handling when a node gets reparented (its Transform migrates to another NodeArrayMemoryManager)

Default SceneManager creates one: mNodeMemoryManager. Other Scene Managers may want to have more (eg. One NodeMemoryManager per Octant or per Portal)

MovableObjects?



ObjectMemoryManager
(contains multiple...)

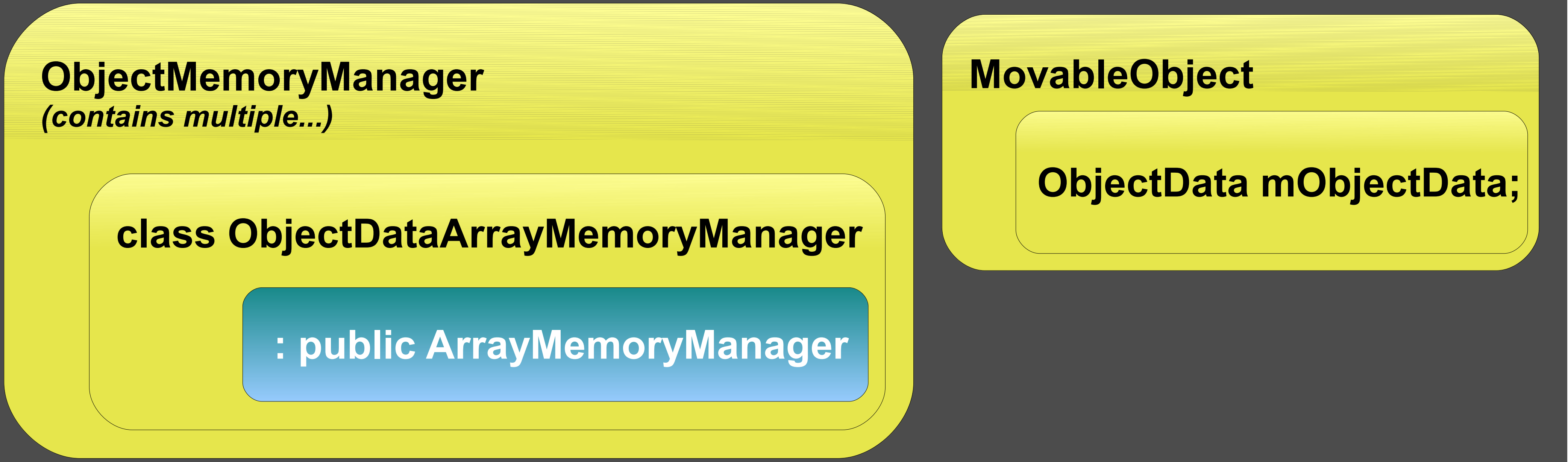
class ObjectDataArrayMemoryManager

: public ArrayMemoryManager

MovableObject

ObjectData mObjectData;

MovableObjects?



ObjectMemoryManager
(contains multiple...)

class ObjectDataArrayMemoryManager

: public ArrayMemoryManager

MovableObject

ObjectData mObjectData;

It's the same thing, except we have ObjectData instead of Transform, and MovableObjects are grouped by RenderQueue ID (starting from 0) instead of parent level depth.

SIMD Coherence?

Using SSE2 single precision, OGRE usually processes 4 nodes/entities at a time. However, if there is only 3 Nodes; we need to allocate memory for 4 of them (ArrayMemoryManager already handles this) and initialize the values to sane defaults even if they aren't used (eg. Set quaternion & matrices to identity)

Another issue is that both Transform and ObjectData store the parent node, like this:

```
    Node *mParents[4];
```

Even though mParents[3] is not in use, we still may need to access it. We could set the pointer to NULL, but that's bad DOD practice (check for null ptrs on all nodes even though only a few are actually null)

Instead, we set mParents[3] to a very basic, dummy pointer with valid data so that the engine doesn't crash and we avoid to check for null values during throughput-intensive operations.

SIMD Coherence is very important for both stability and performance, and is 99% of the time responsability of the Memory Managers

Matías Nazareth Goldberg
@matiasgoldberg

Google Summer of Code 2013
O-OGRE 3D Project