LOD and Culling Systems That Scale

Mike Acton

福海

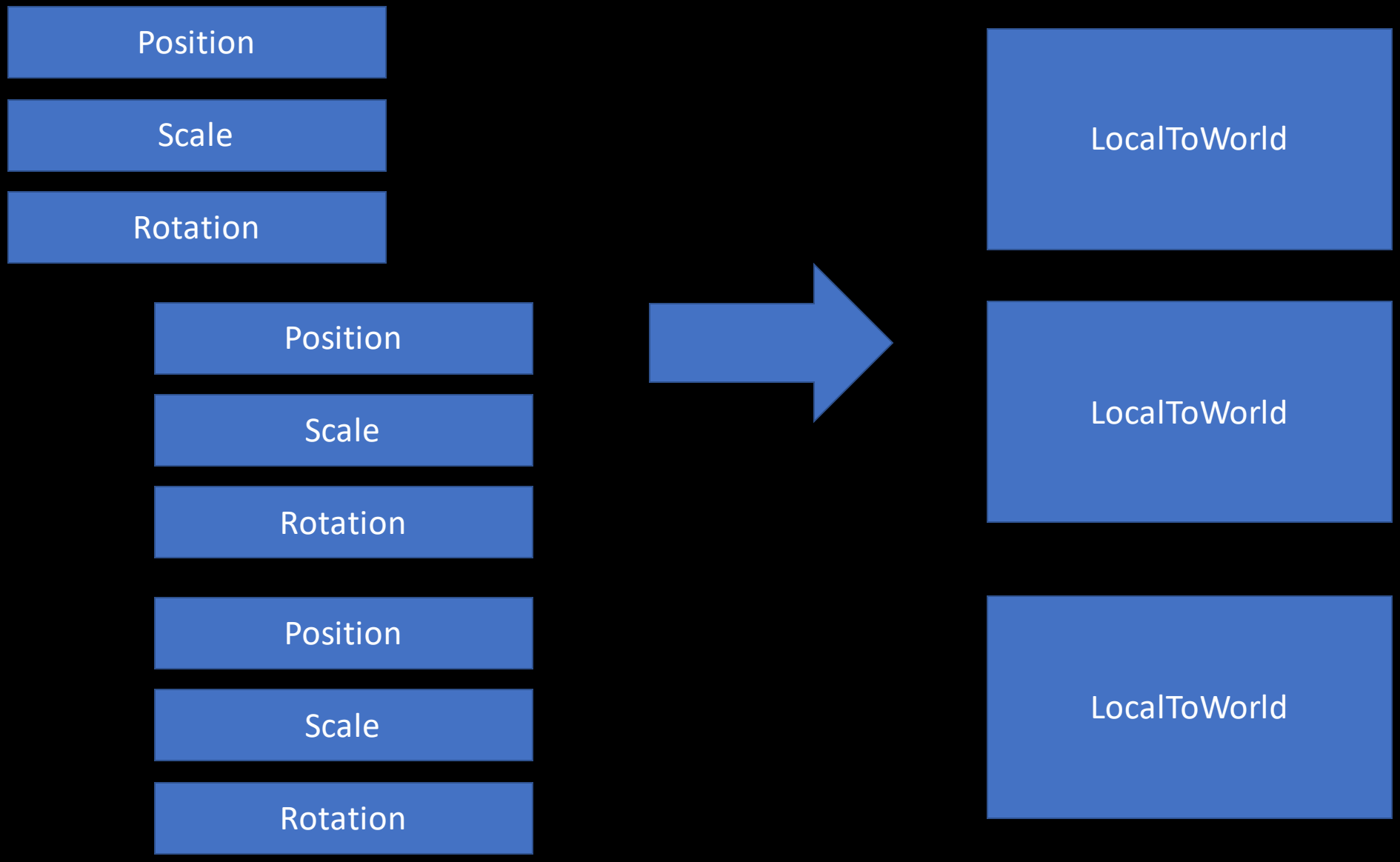
新匯豪
夜總會
卡拉OK

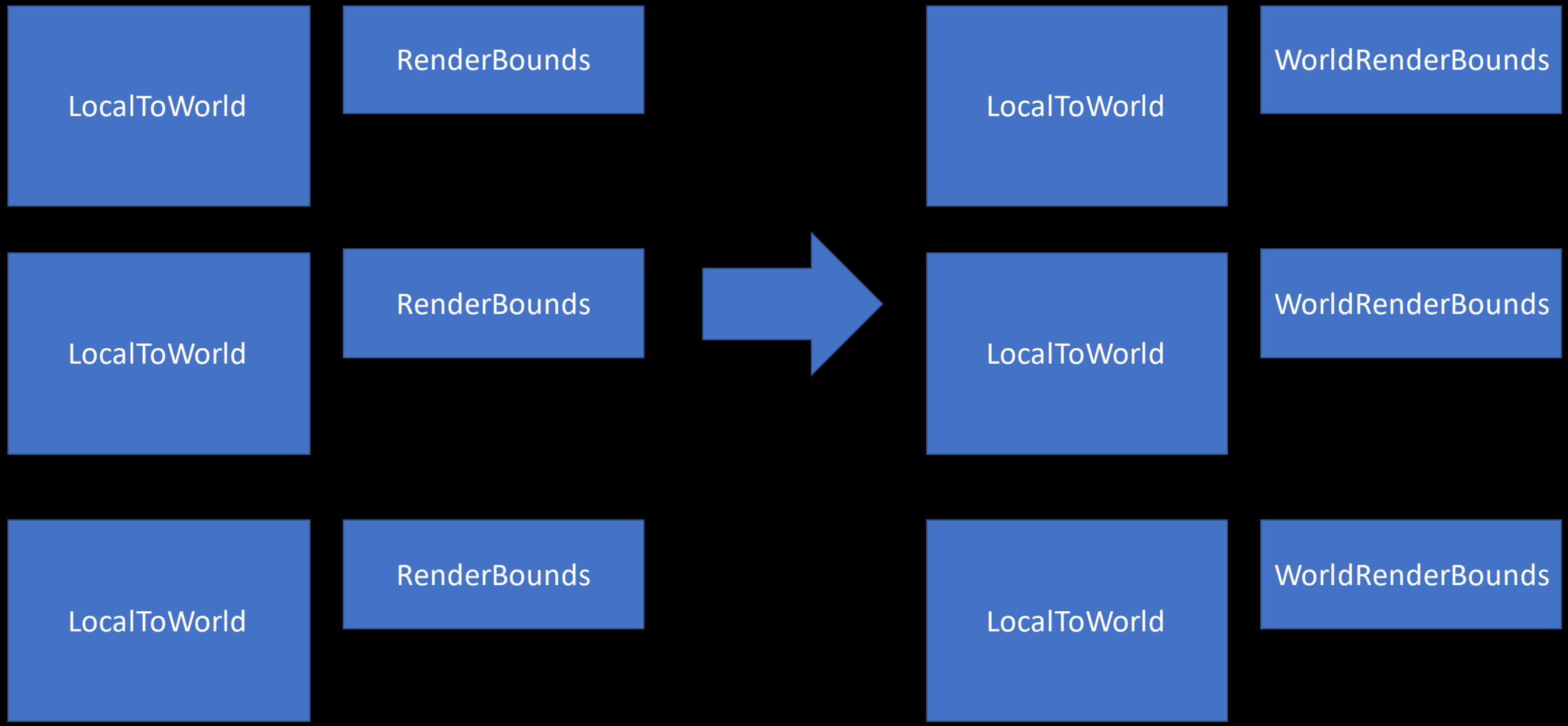
Culling

- Per view frustum (6-18 culling planes)
- 5 views in this sample (main camera + shadow cascades)
- Deep hierarchy (nested prefabs)
- ~30-200K instances per section (e.g. single building)
- ~5M instances in scene (whole scene in editor)

Culling

- What’s not surprising?
- World is static.
- Flatten hierarchy.


Position
Scale
Rotation
Position
Scale
Rotation
Position
Scale
Rotation
LocalToWorld
LocalToWorld
LocalToWorld


LocalToWorld
RenderBounds
LocalToWorld
RenderBounds
LocalToWorld
RenderBounds
LocalToWorld
WorldRenderBounds
LocalToWorld
WorldRenderBounds
LocalToWorld
WorldRenderBounds

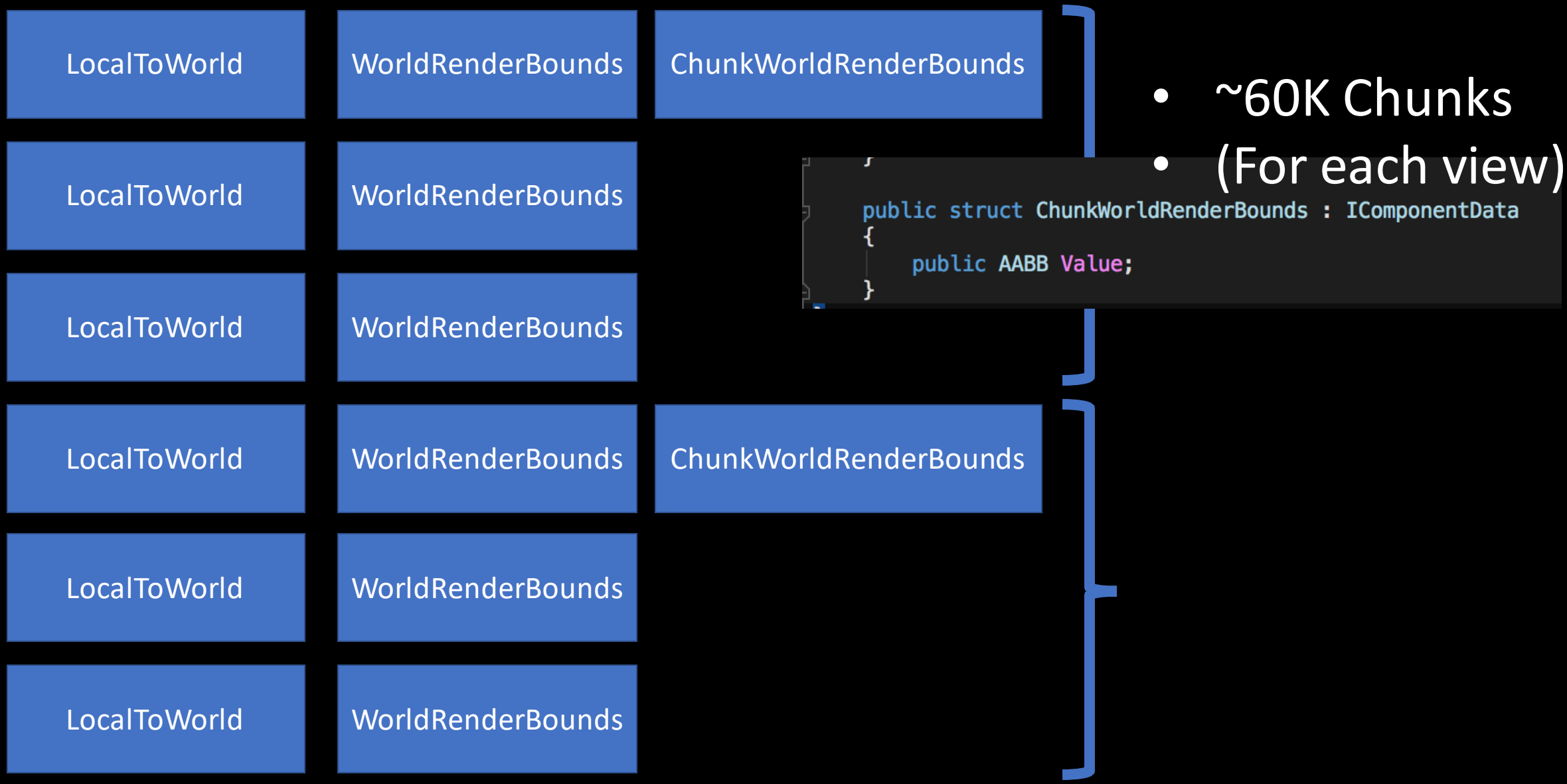

```
/// </summary>
[Serializable]
public struct LocalToWorld : IComponentData
{
    public float4x4 Value;
}
```

```
}

public struct WorldRenderBounds : IComponentData
{
    public AABB Value;
}
```

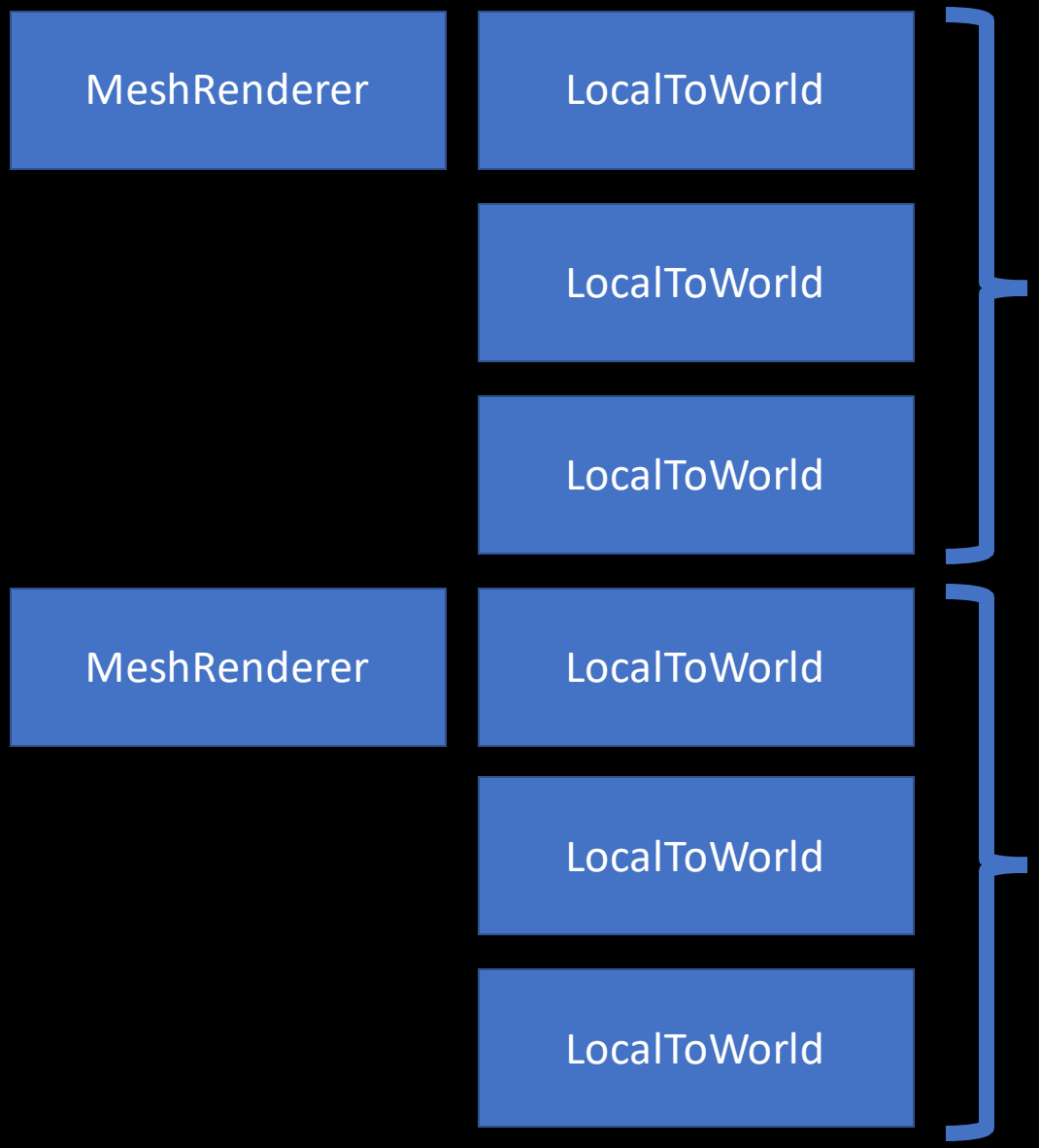
- Flat list of 5M instances
- Next step?
- Hierarchical spatial tree?
- What’s our data look like?

- 16K Chunks of component data
- Subscenes roughly spatial


LocalToWorld
WorldRenderBounds
ChunkWorldRenderBounds
LocalToWorld
WorldRenderBounds
LocalToWorld
WorldRenderBounds
LocalToWorld
WorldRenderBounds
ChunkWorldRenderBounds
LocalToWorld
WorldRenderBounds
LocalToWorld
WorldRenderBounds
public struct ChunkWorldRenderBounds : IComponentData
{
public AABB Value;
}
• ~60K Chunks
• (For each view)

Culling

- Another level of hierarchy?
- Take a look at the data we have again…

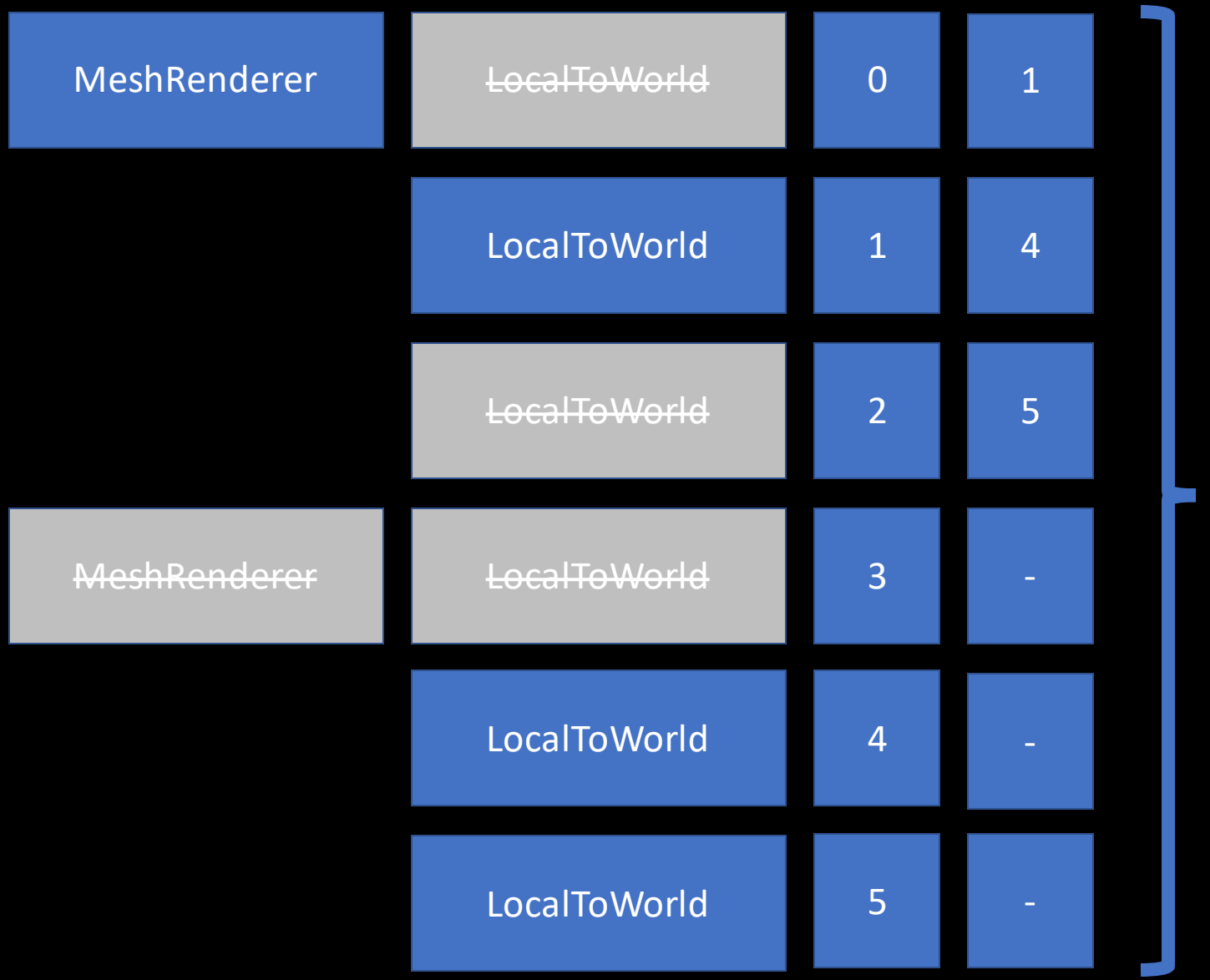


- Render Batch

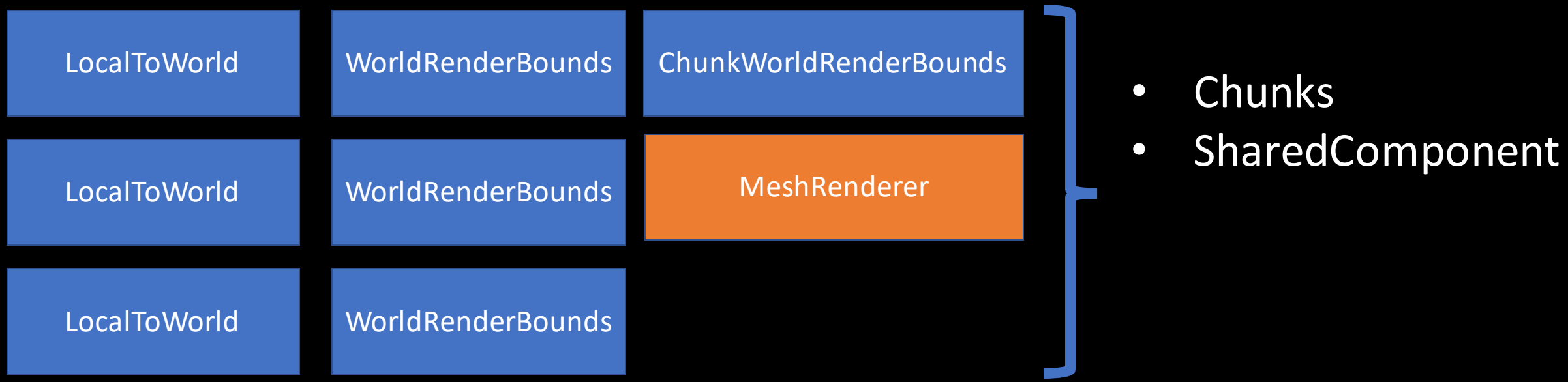

MeshRenderer
LocalToWorld
LocalToWorld
LocalToWorld
MeshRenderer
LocalToWorld
LocalToWorld
LocalToWorld

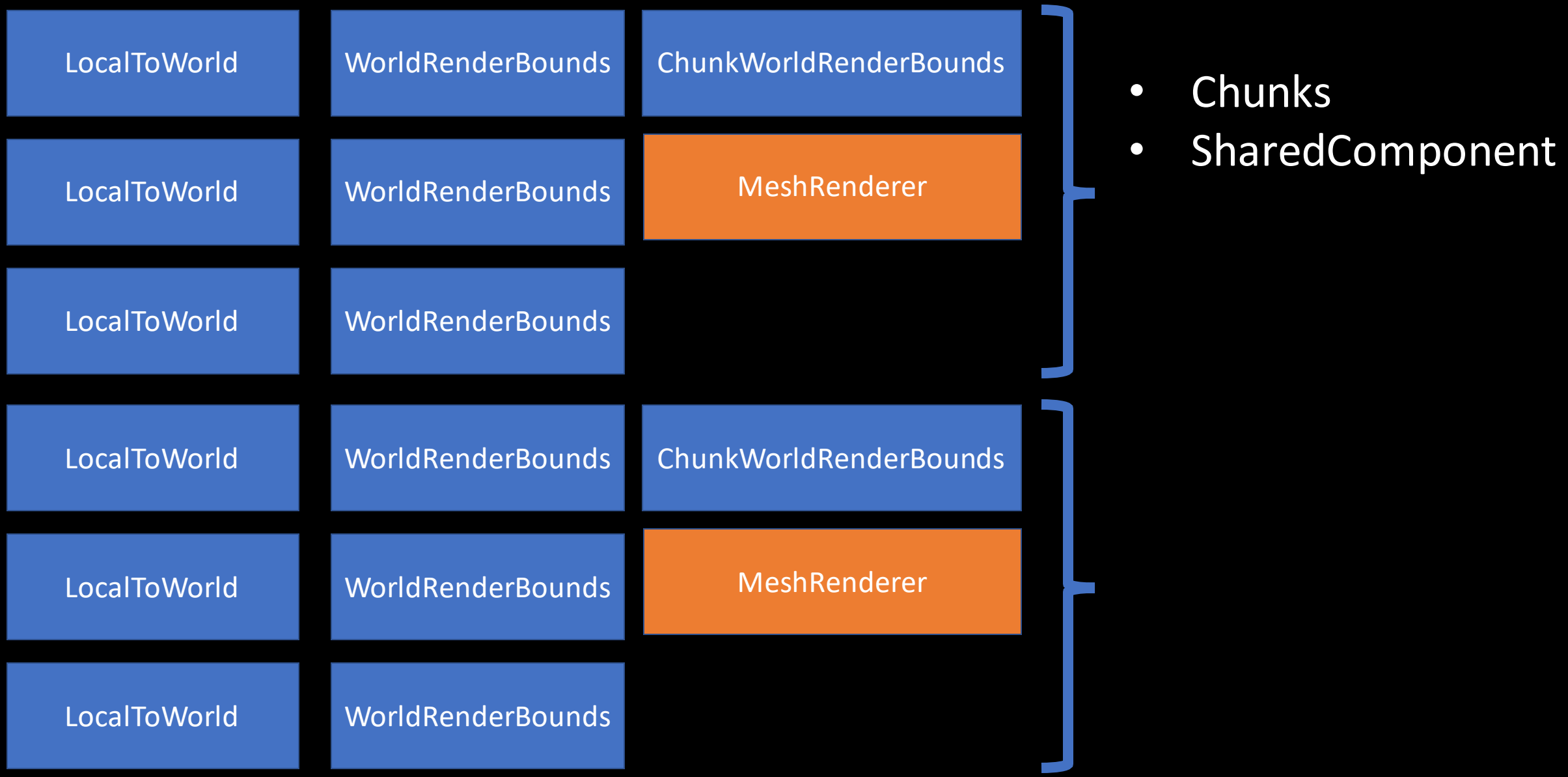

MeshRenderer
LocalToWorld
0
-
LocalToWorld
1
-
LocalToWorld
2
-
~~MeshRenderer~~
LocalToWorld
3
-
LocalToWorld
4
-
LocalToWorld
5
-

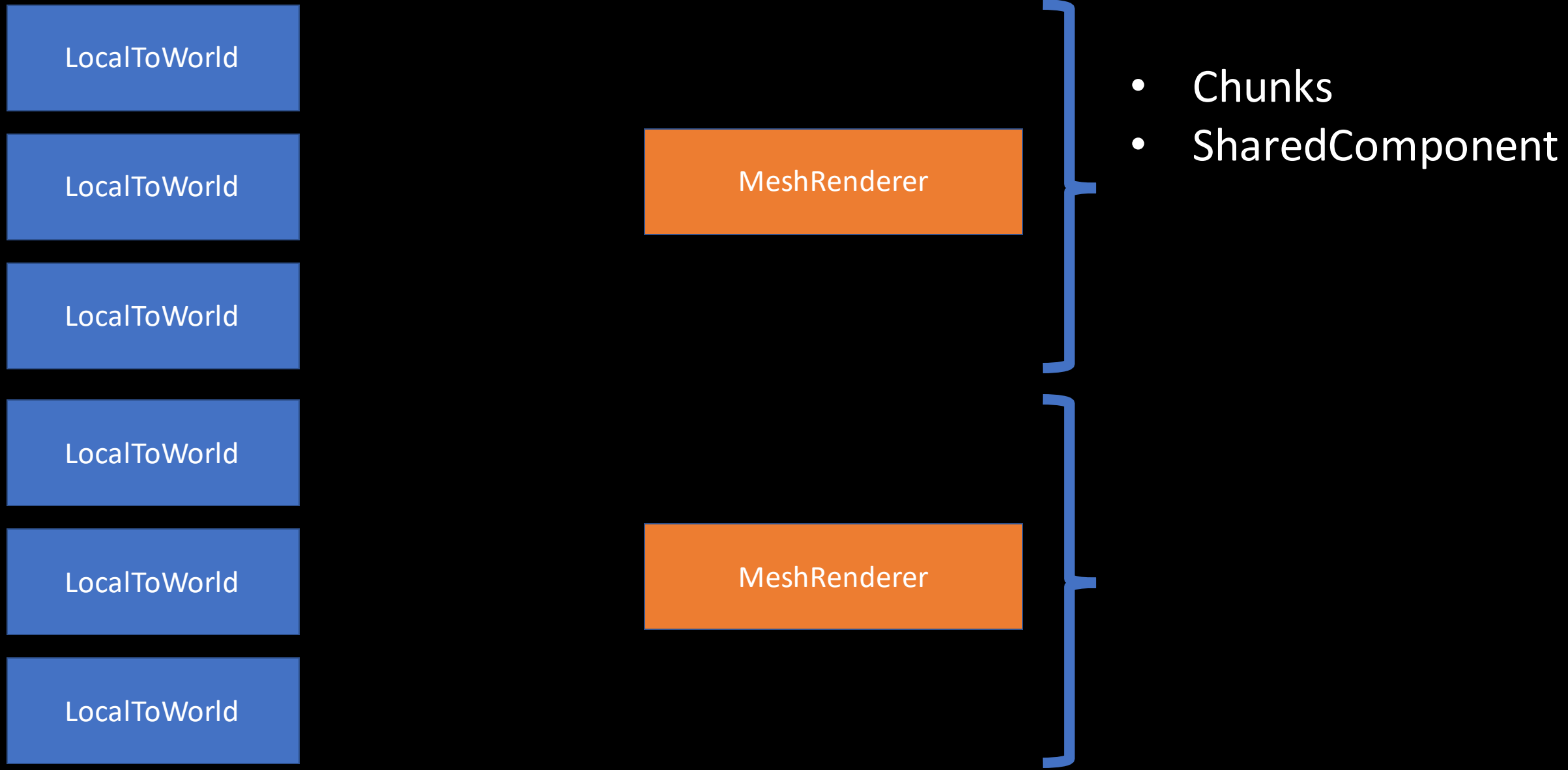
MeshRenderer	~~LocalToWorld~~	0	-
	LocalToWorld	1	-
	~~LocalToWorld~~	2	-
~~MeshRenderer~~	~~LocalToWorld~~	3	-
	LocalToWorld	4	-
	LocalToWorld	5	-

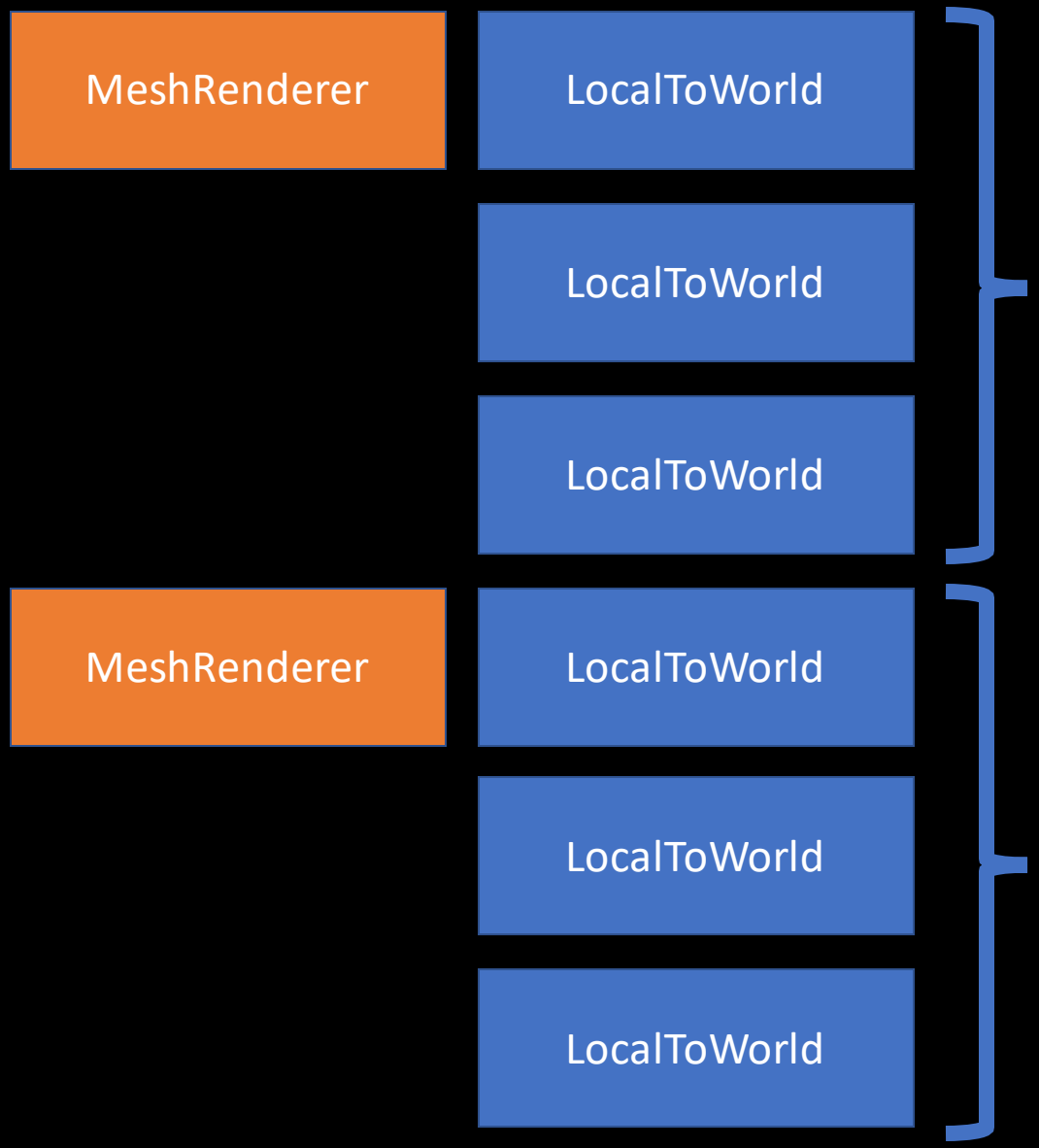


- Render Batch Index List
- Same Mesh Renderer
- Max 1023 instances

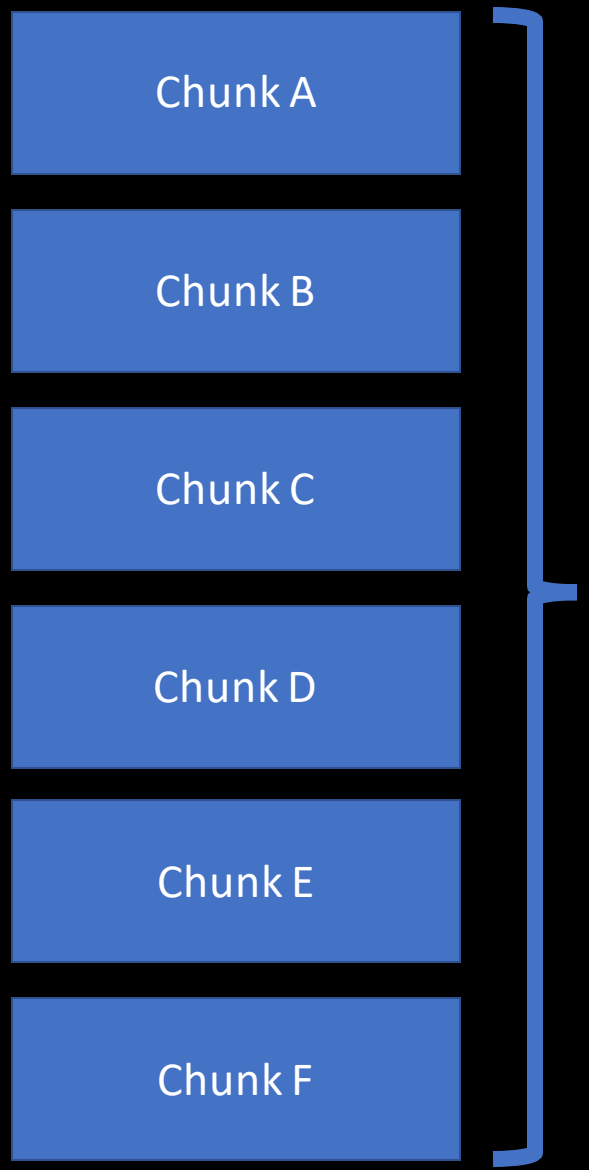

LocalToWorld
WorldRenderBounds
ChunkWorldRenderBounds
LocalToWorld
WorldRenderBounds
MeshRenderer
LocalToWorld
WorldRenderBounds
• Chunks
• SharedComponent


LocalToWorld
WorldRenderBounds
ChunkWorldRenderBounds
LocalToWorld
WorldRenderBounds
MeshRenderer
LocalToWorld
WorldRenderBounds
LocalToWorld
WorldRenderBounds
ChunkWorldRenderBounds
LocalToWorld
WorldRenderBounds
MeshRenderer
LocalToWorld
WorldRenderBounds
• Chunks
• SharedComponent

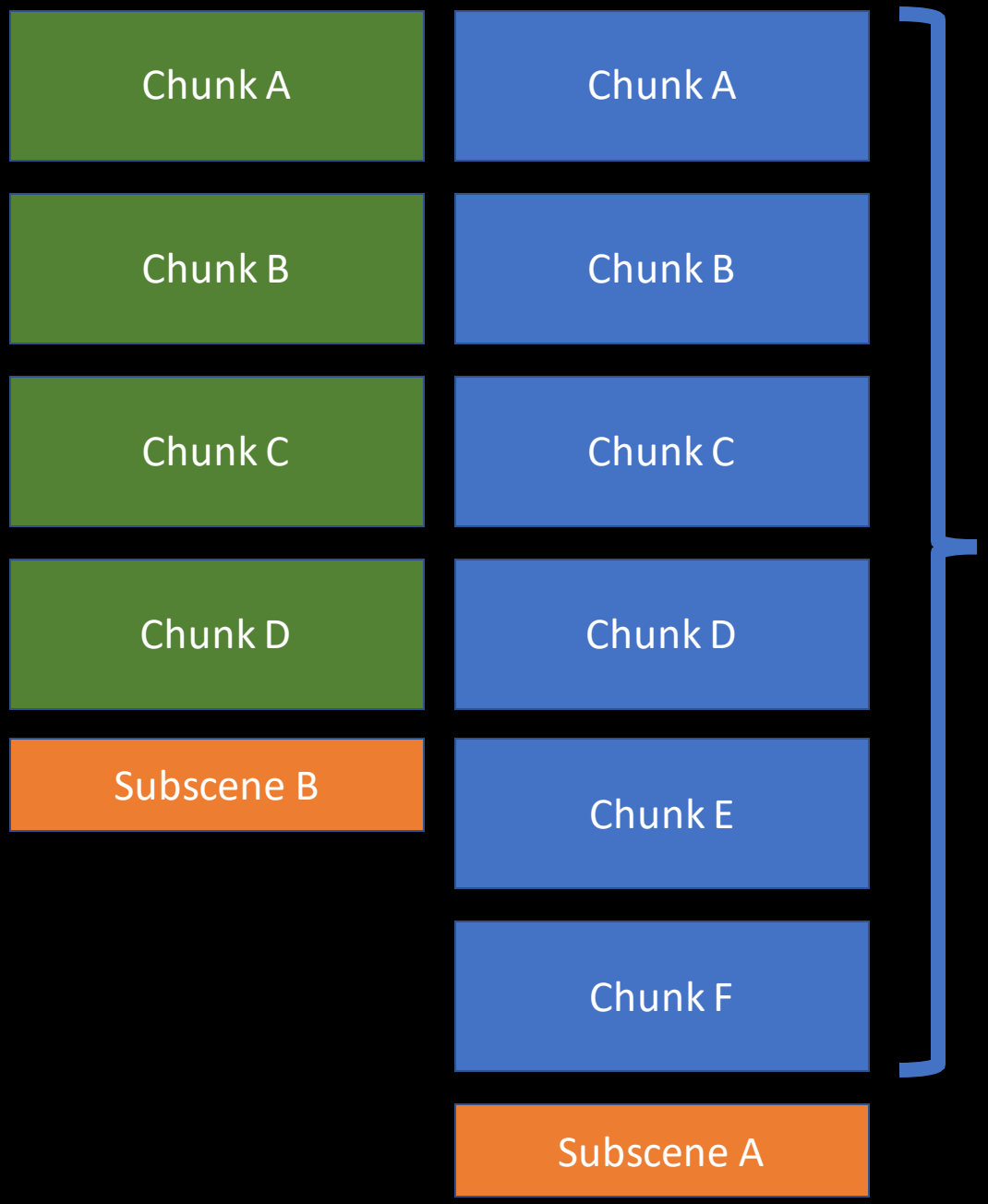

LocalToWorld
LocalToWorld
LocalToWorld
LocalToWorld
LocalToWorld
LocalToWorld
MeshRenderer
MeshRenderer
• Chunks
• SharedComponent



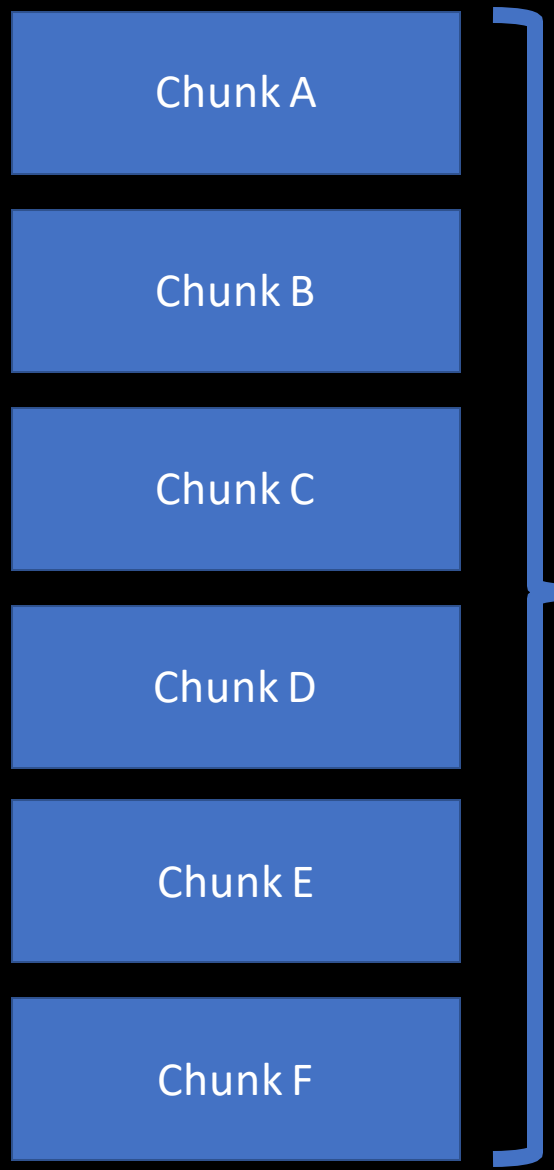
- Looks a like like what we need for render batch



- Render Batch = NativeArray of Chunks
- …Gives us second-level hierarchy



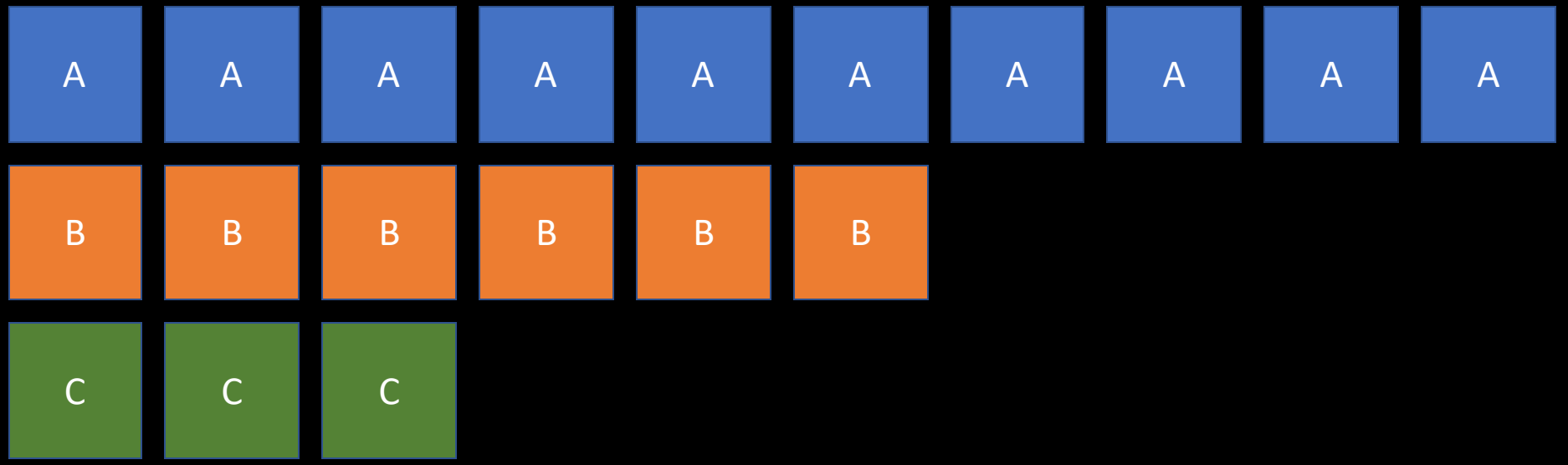
- Different streaming subcenes?



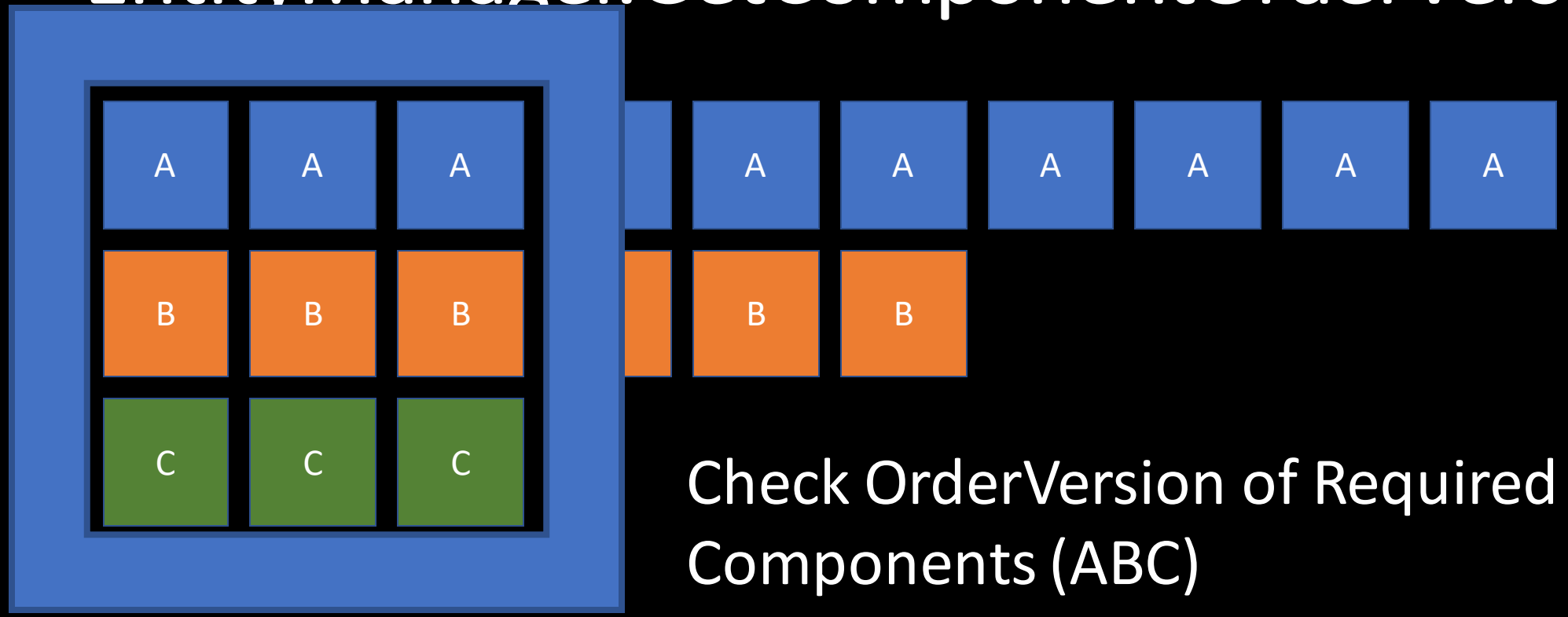
[Aside] Let’s go back to…
Render Batch = NativeArray of Chunks

Is this safe?

EntityManager.GetComponentOrderVersion

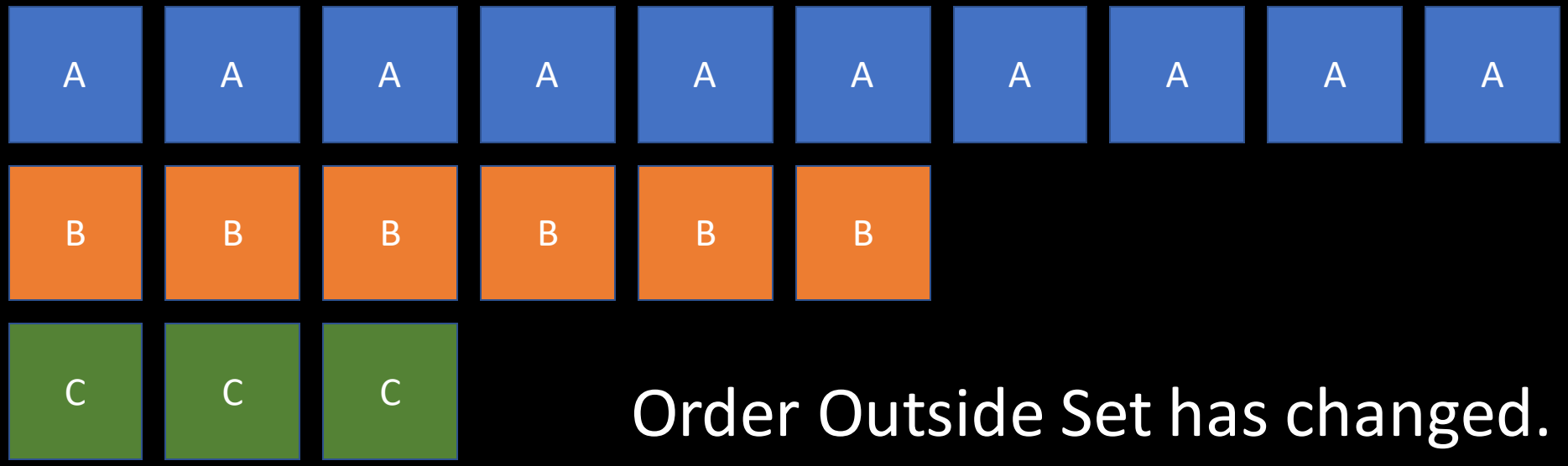


EntityManager.GetComponentOrderVersion



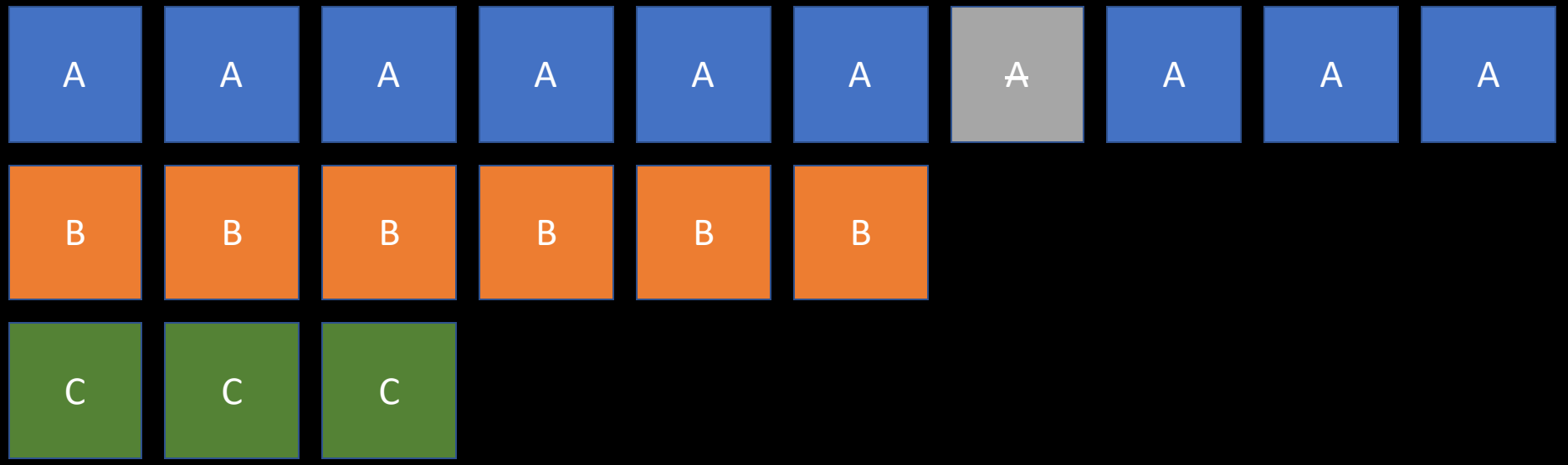
Check OrderVersion of Required Components (ABC)

EntityManager.GetComponentOrderVersion

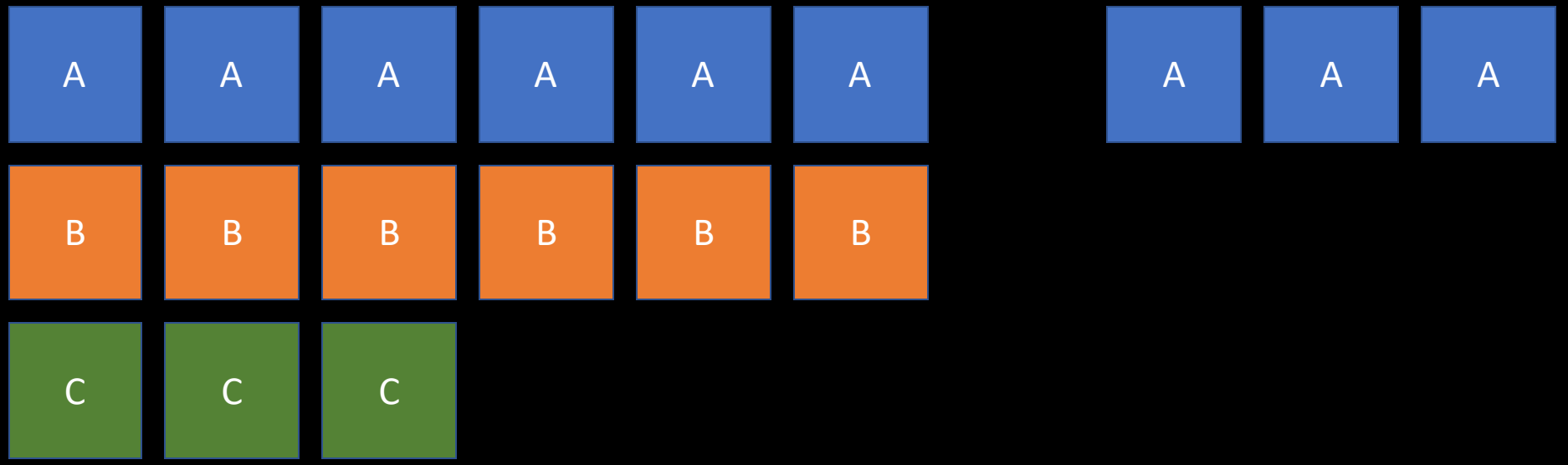


Order Outside Set has changed.

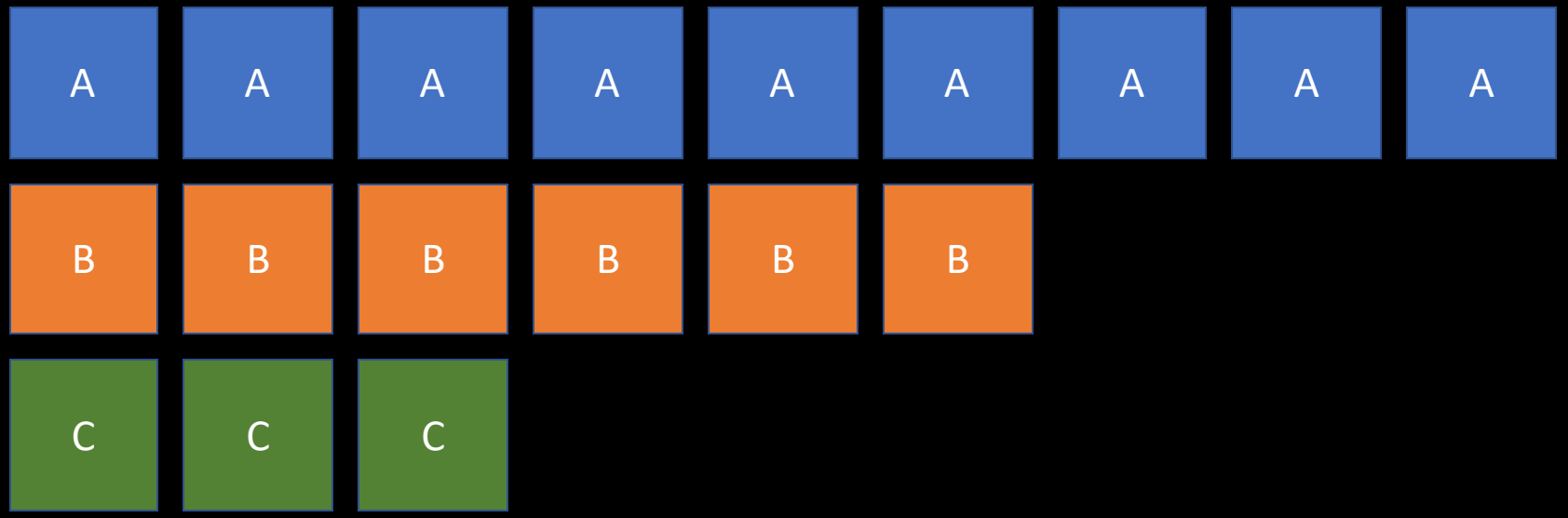
EntityManager.GetComponentOrderVersion



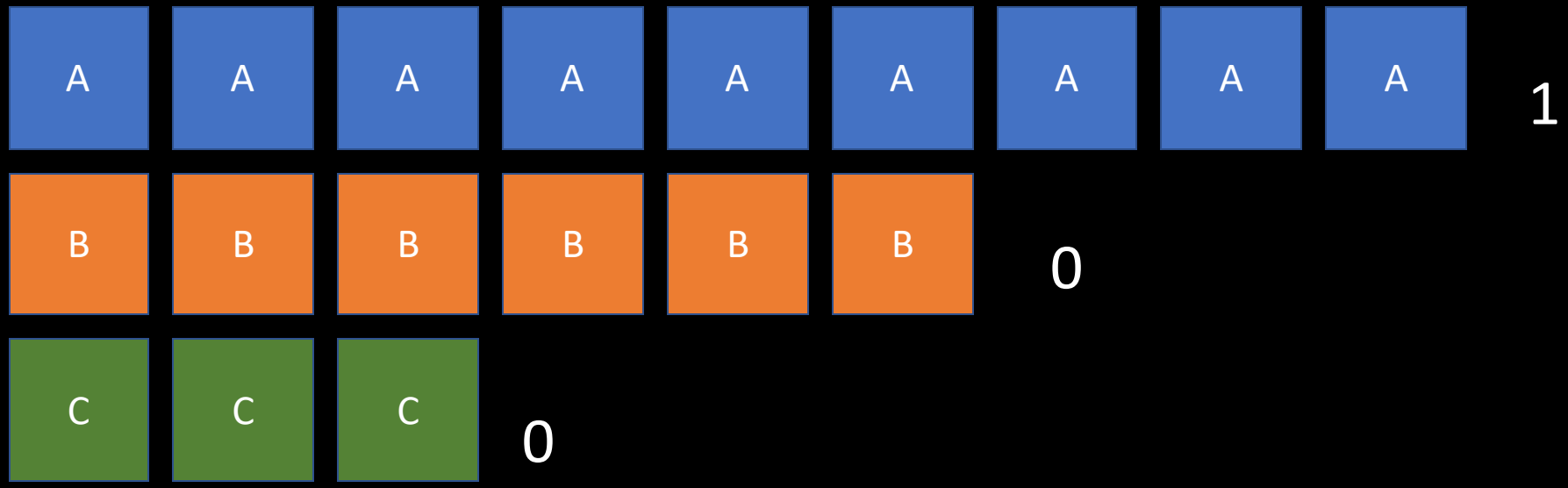
EntityManager.GetComponentOrderVersion



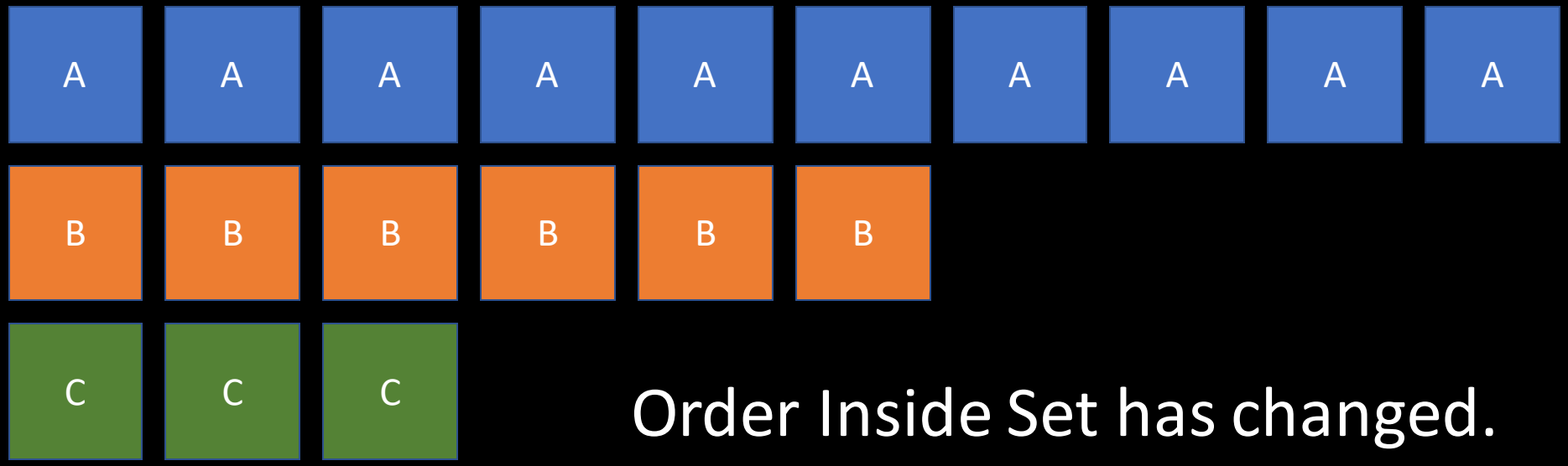
EntityManager.GetComponentOrderVersion



EntityManager.GetComponentOrderVersion

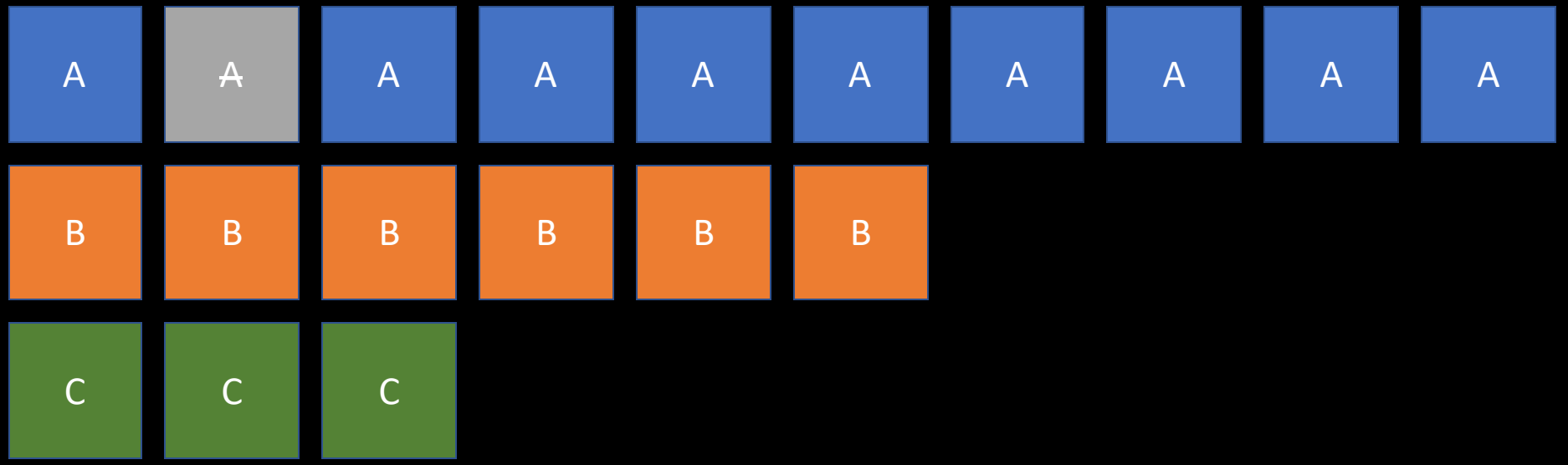


EntityManager.GetComponentOrderVersion

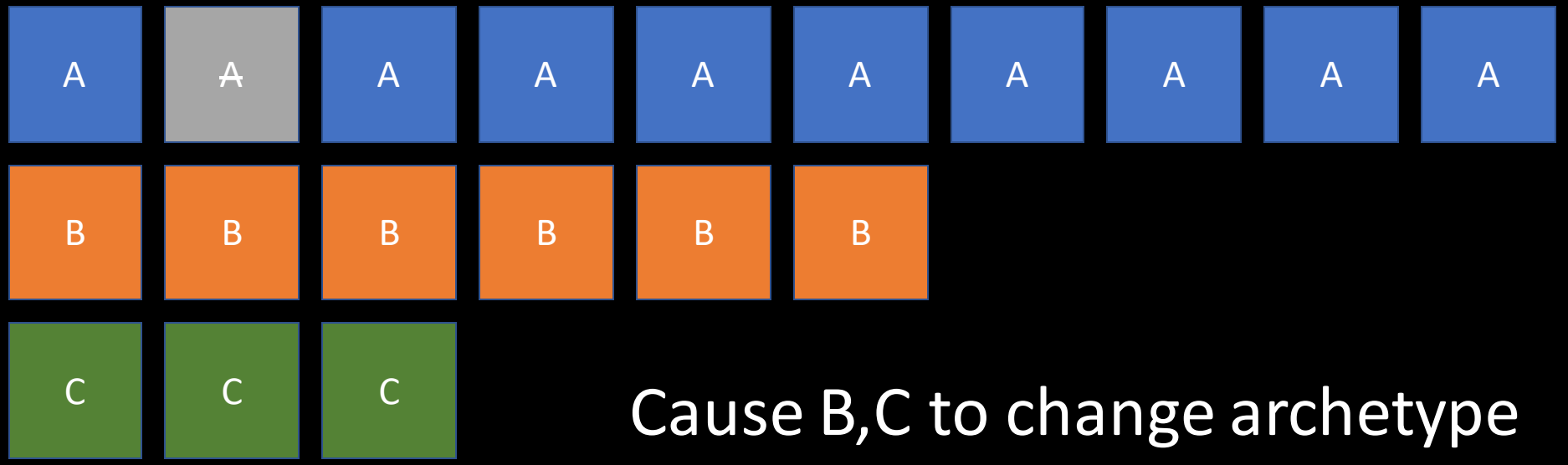


Order Inside Set has changed.

EntityManager.GetComponentOrderVersion

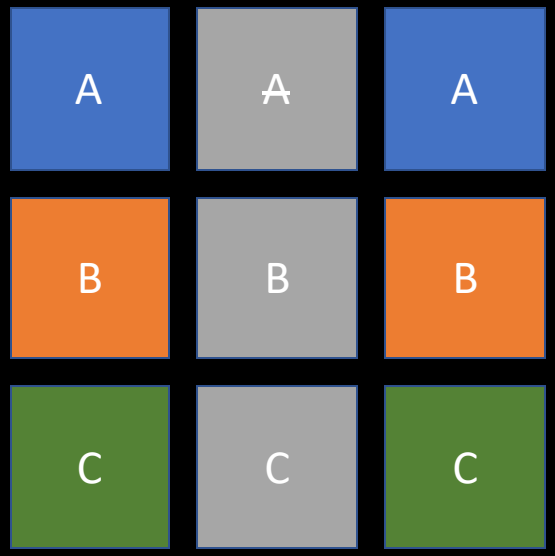


EntityManager.GetComponentOrderVersion



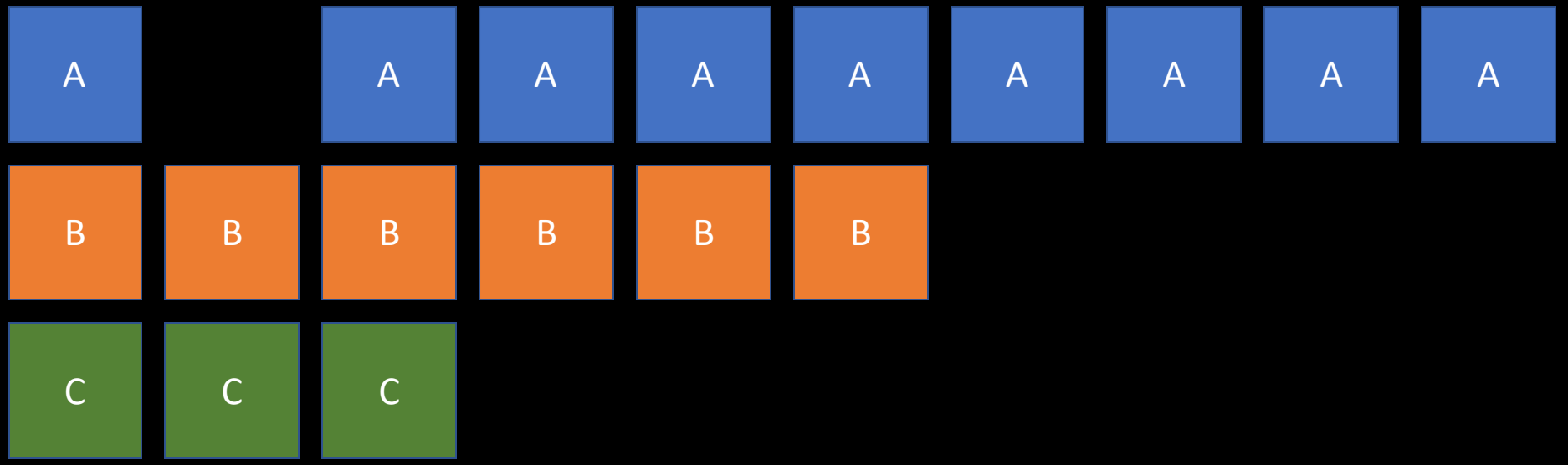
Cause B,C to change archetype

EntityManager.GetComponentOrderVersion

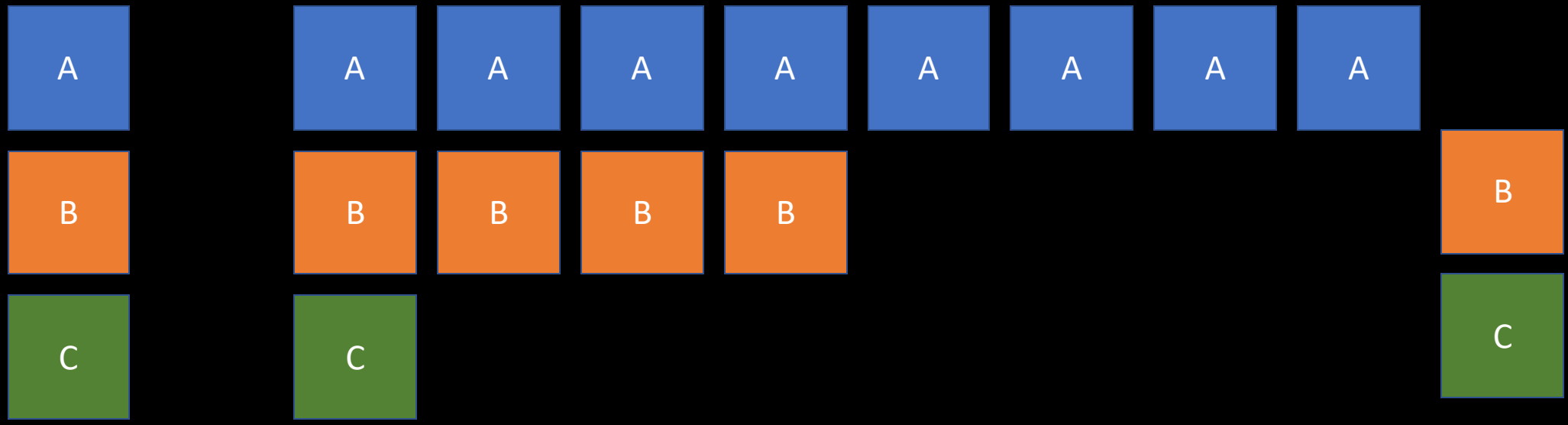


Cause B,C to change archetype

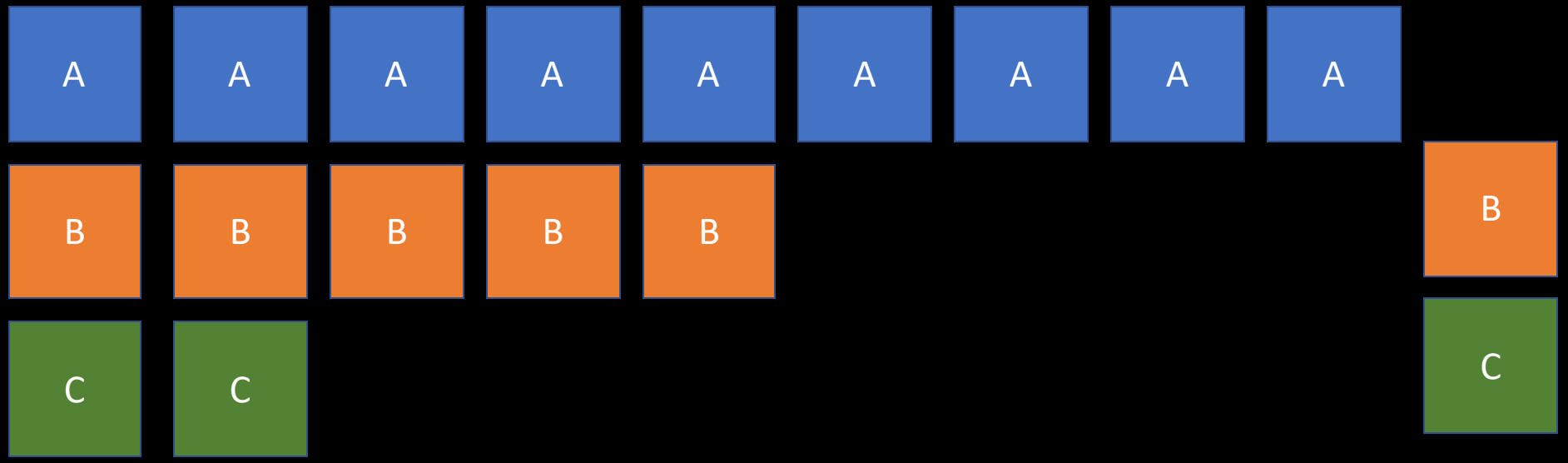
EntityManager.GetComponentOrderVersion



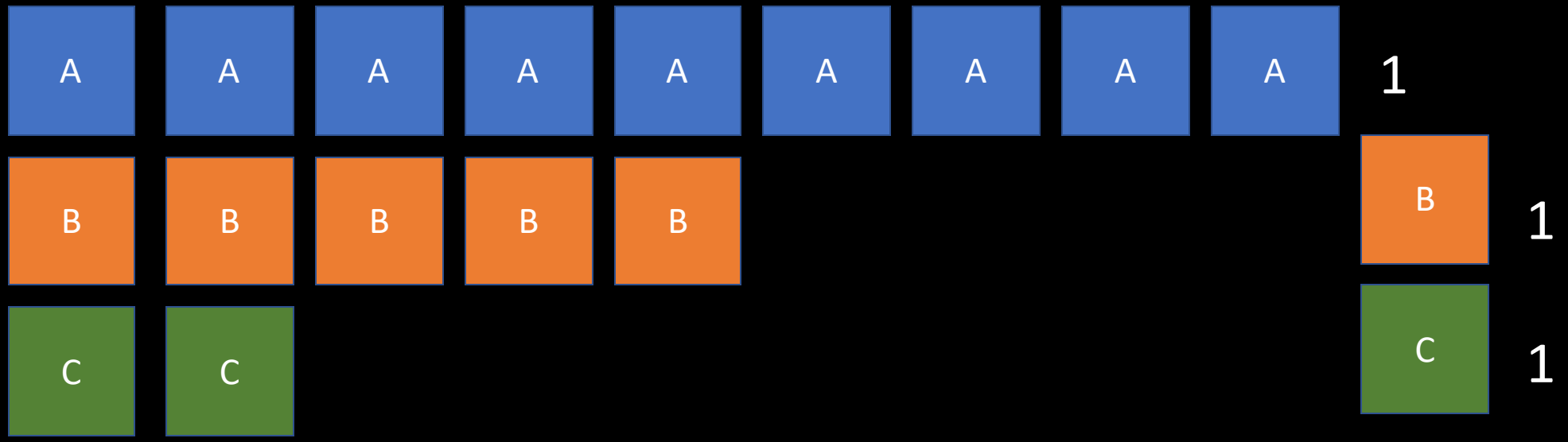
EntityManager.GetComponentOrderVersion



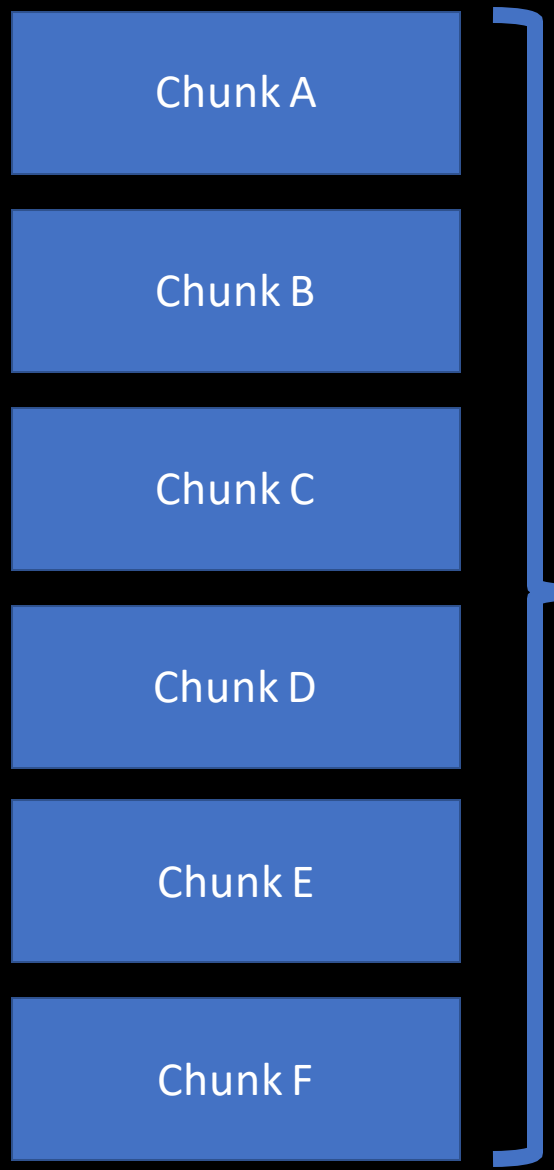
EntityManager.GetComponentOrderVersion



EntityManager.GetComponentOrderVersion



All Required Component Order Versions changed
= Chunks invalid

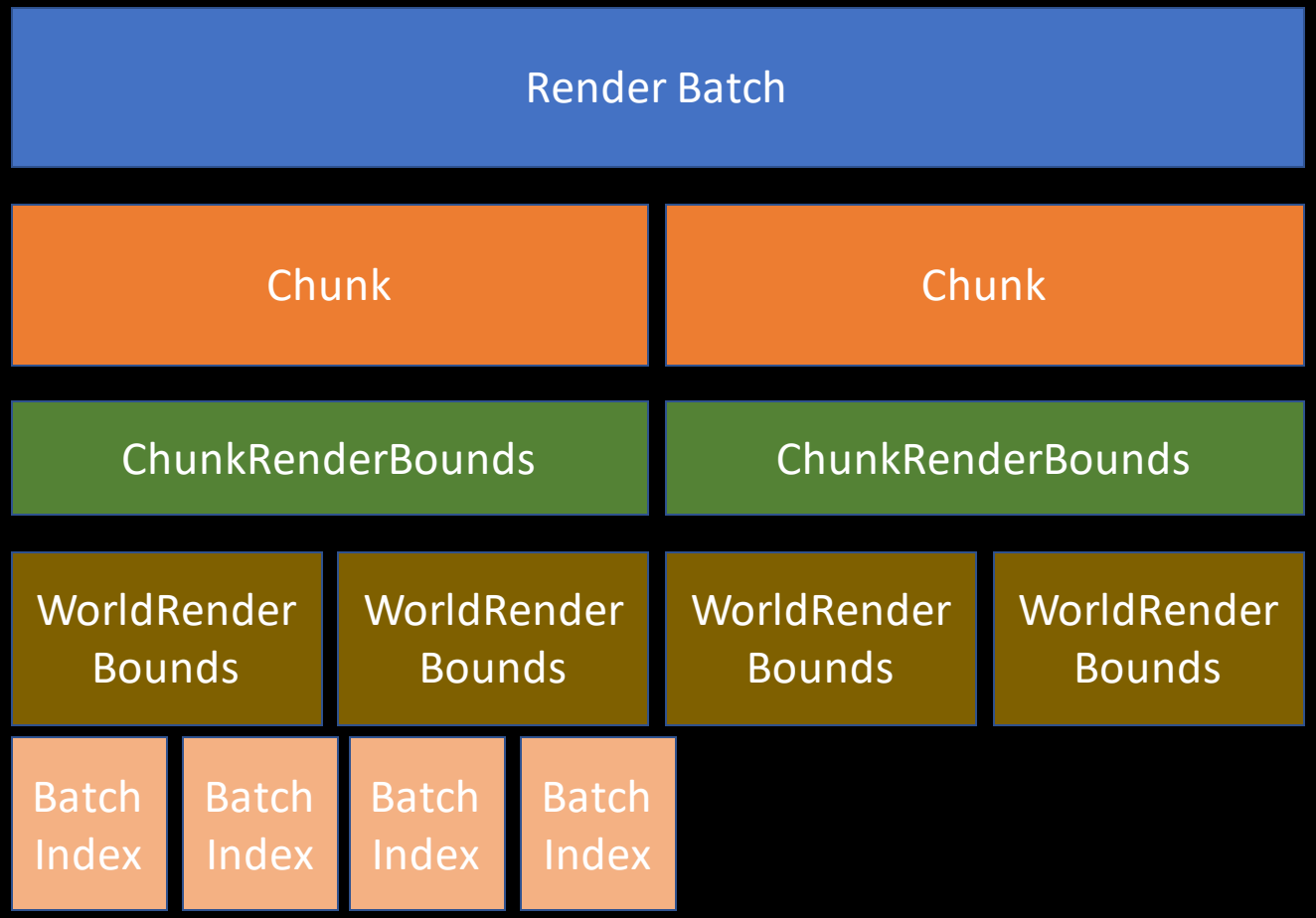


[Aside] Let's go back to…
Render Batch = NativeArray of Chunks

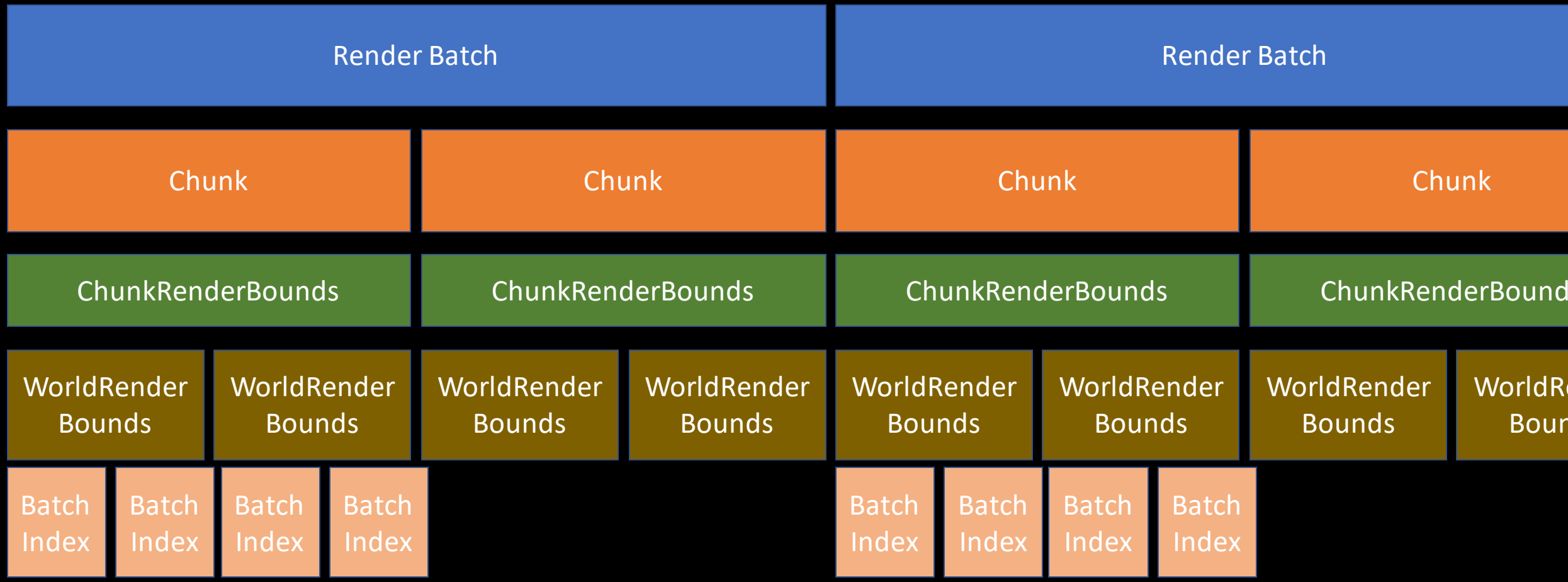
Is this safe?

Yes.

Culling loop is Simple.

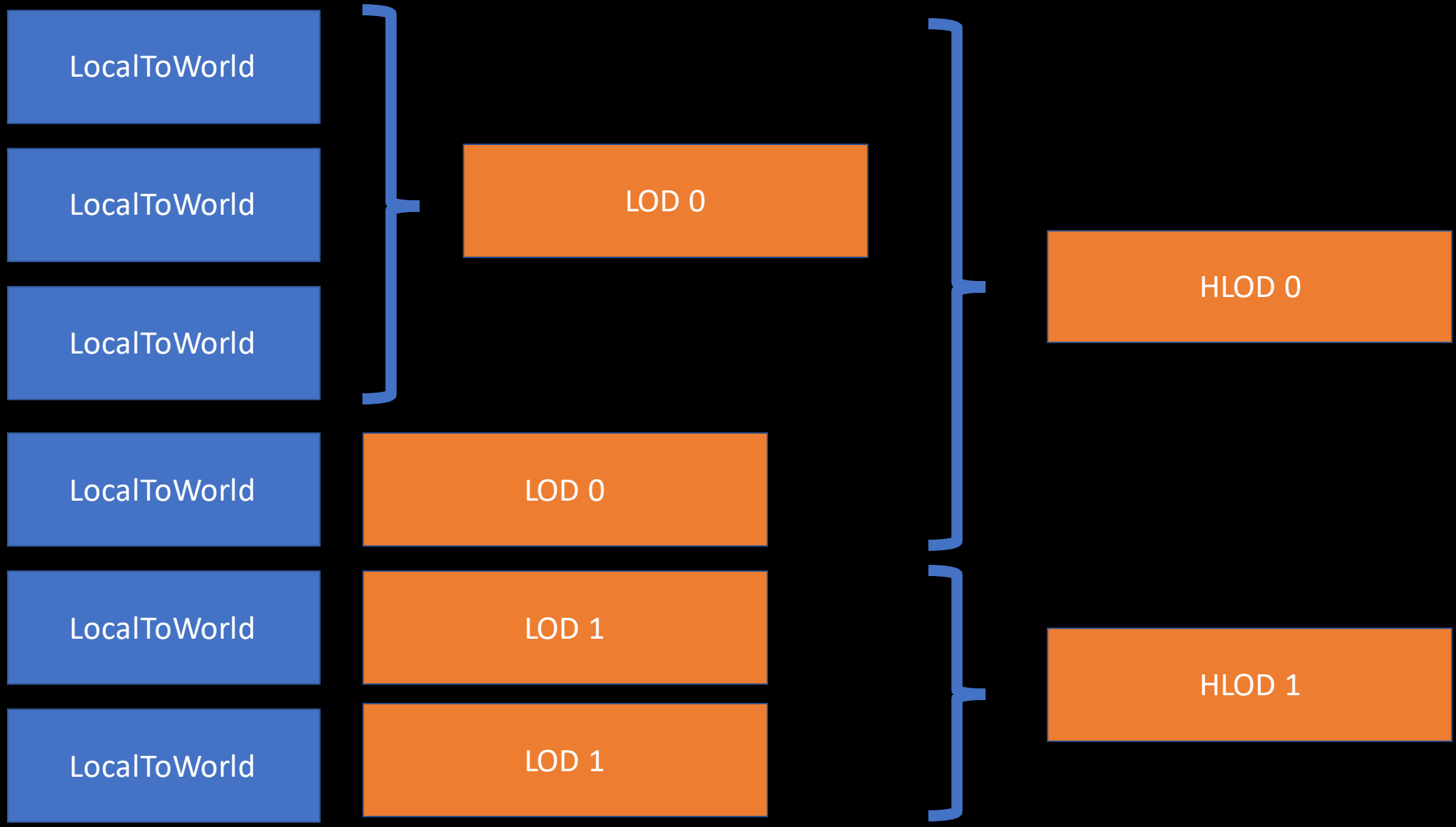


Culling loop is Simple. (Batches in parallel)



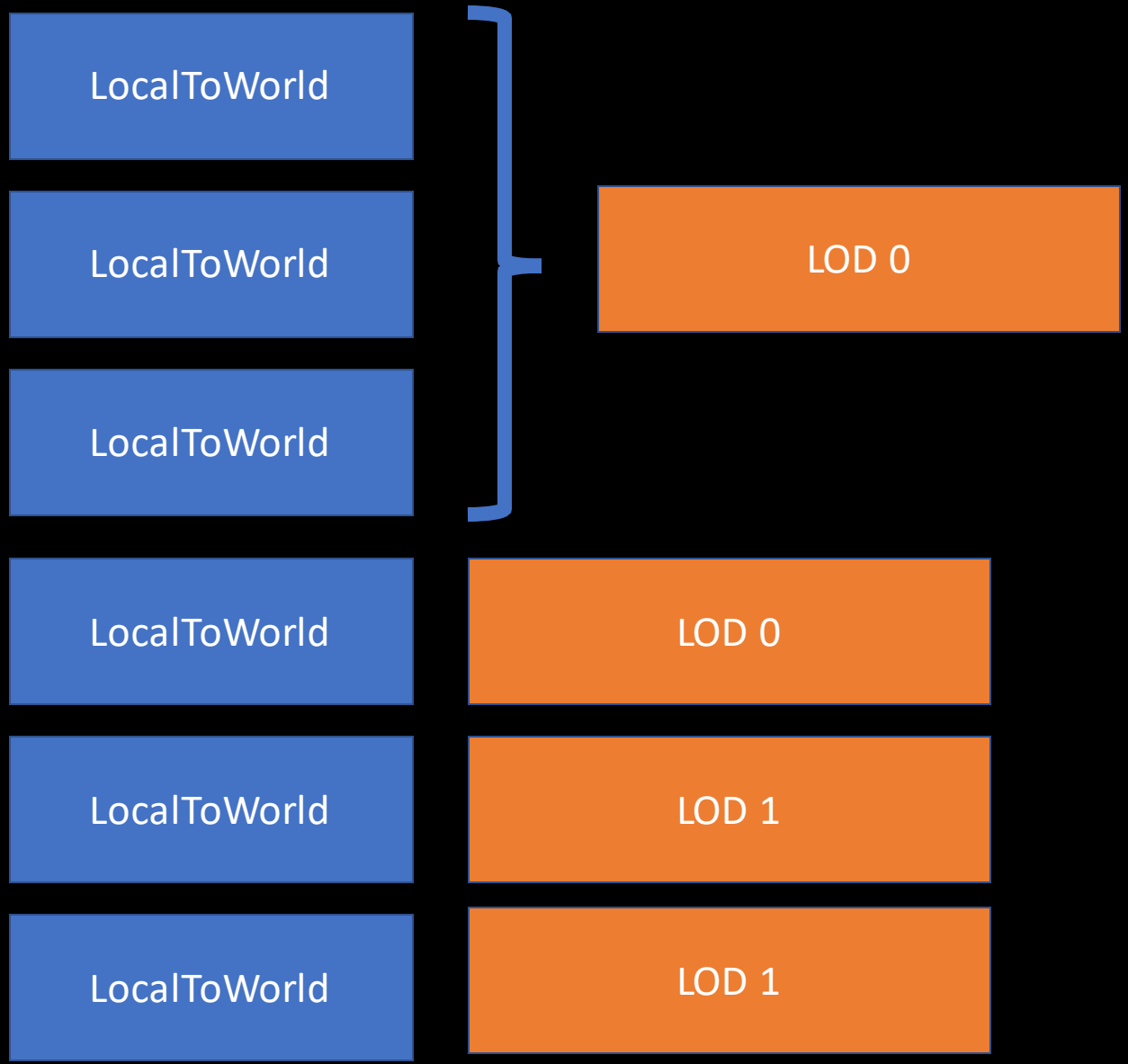
LOD

- What about LOD?
- Flatten similar to transform hierarchy
- But still LOD groupings…


LocalToWorld
LocalToWorld
LocalToWorld
LOD 0
LocalToWorld
LOD 0
LocalToWorld
LOD 1
LocalToWorld
LOD 1
HLOD 0
HLOD 1

LOD

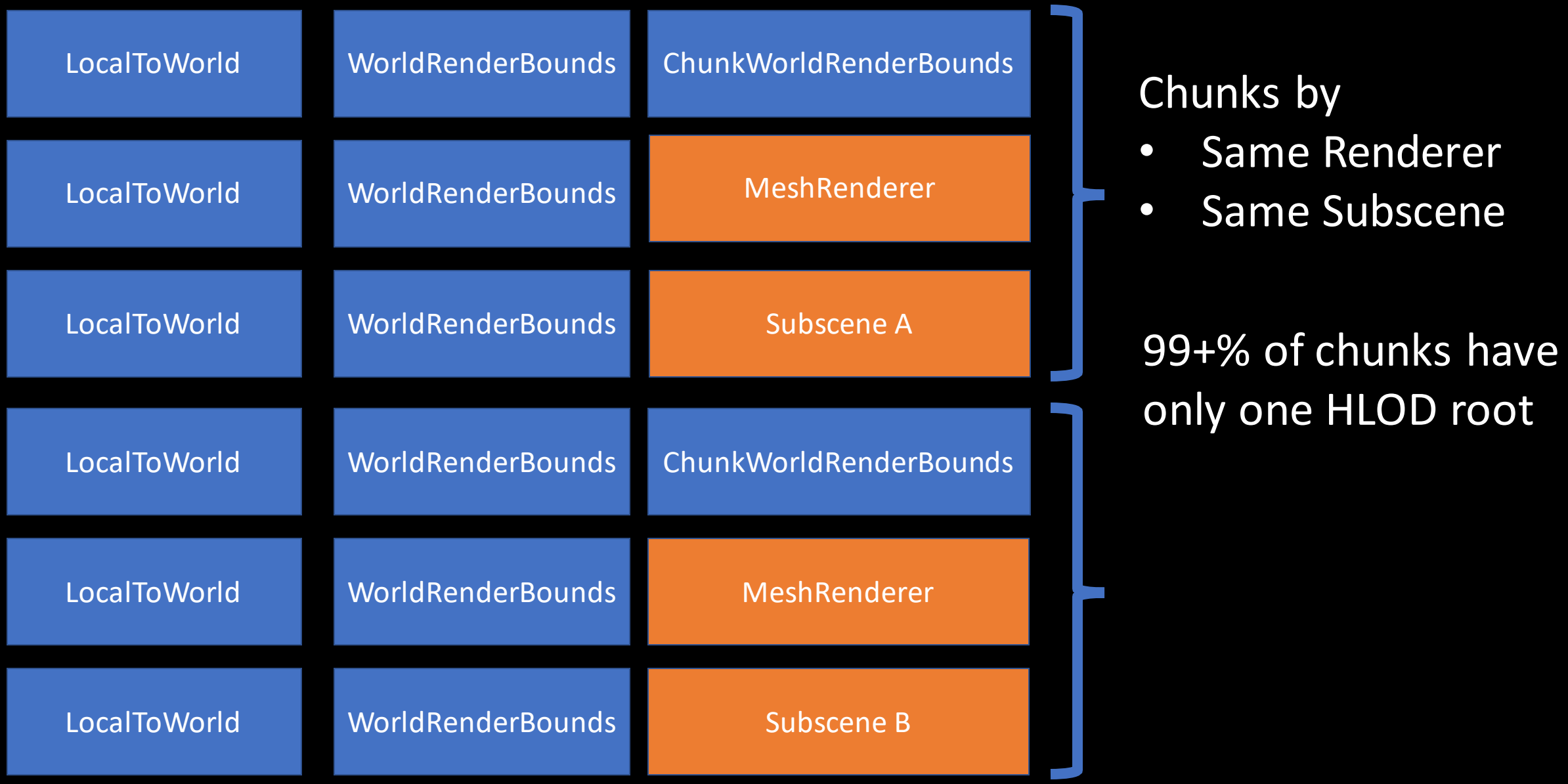
- What about LOD?
- Most instance LODs are *unique* (not mesh combinations)

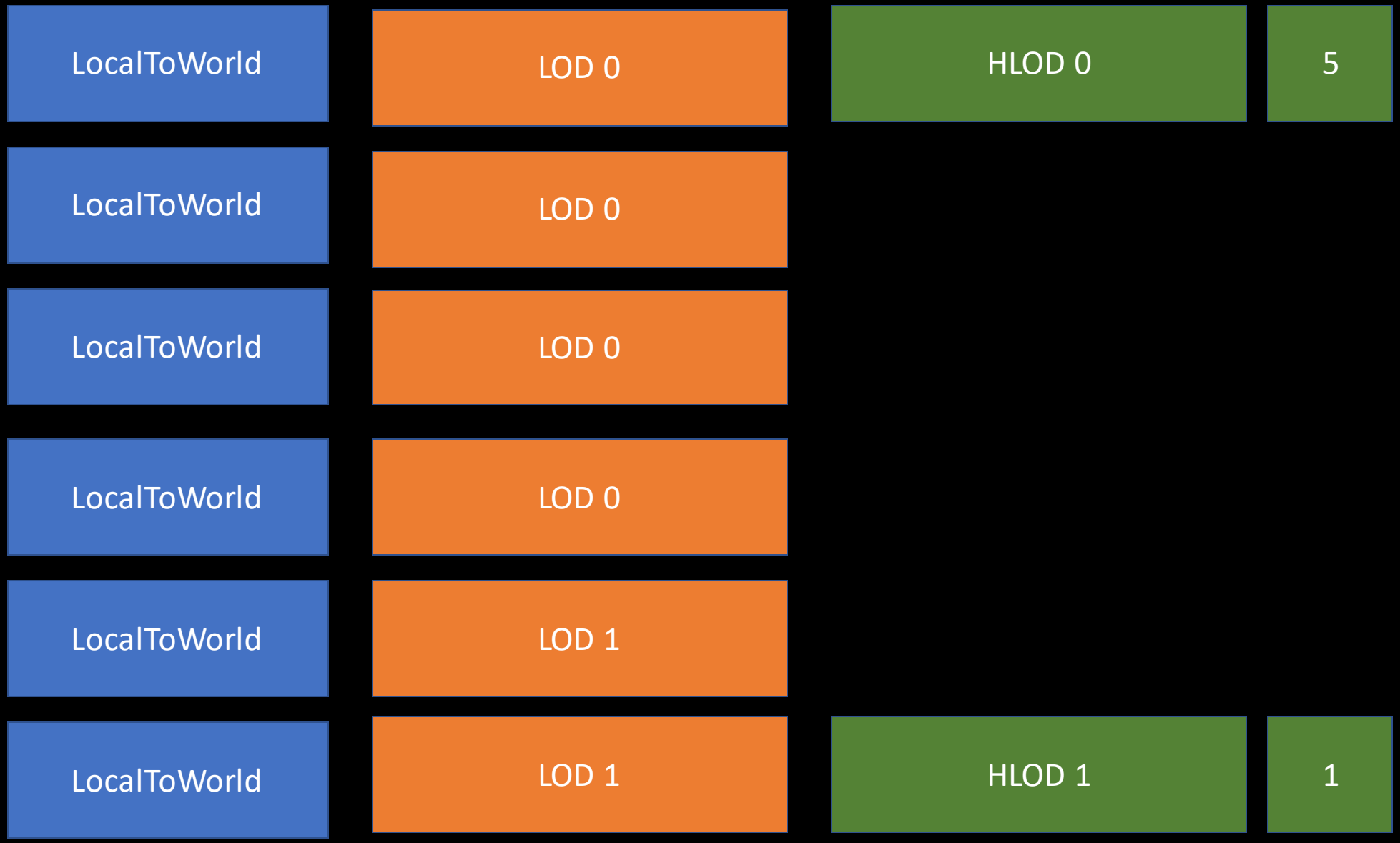

LocalToWorld
LocalToWorld
LocalToWorld
LOD 0
LocalToWorld
LOD 0
LocalToWorld
LOD 1
LocalToWorld
LOD 1


```
struct LodRequirement : IComponentData
{
    public float3 WorldReferencePosition;
    public float MinDist;
    public float MaxDist;
```

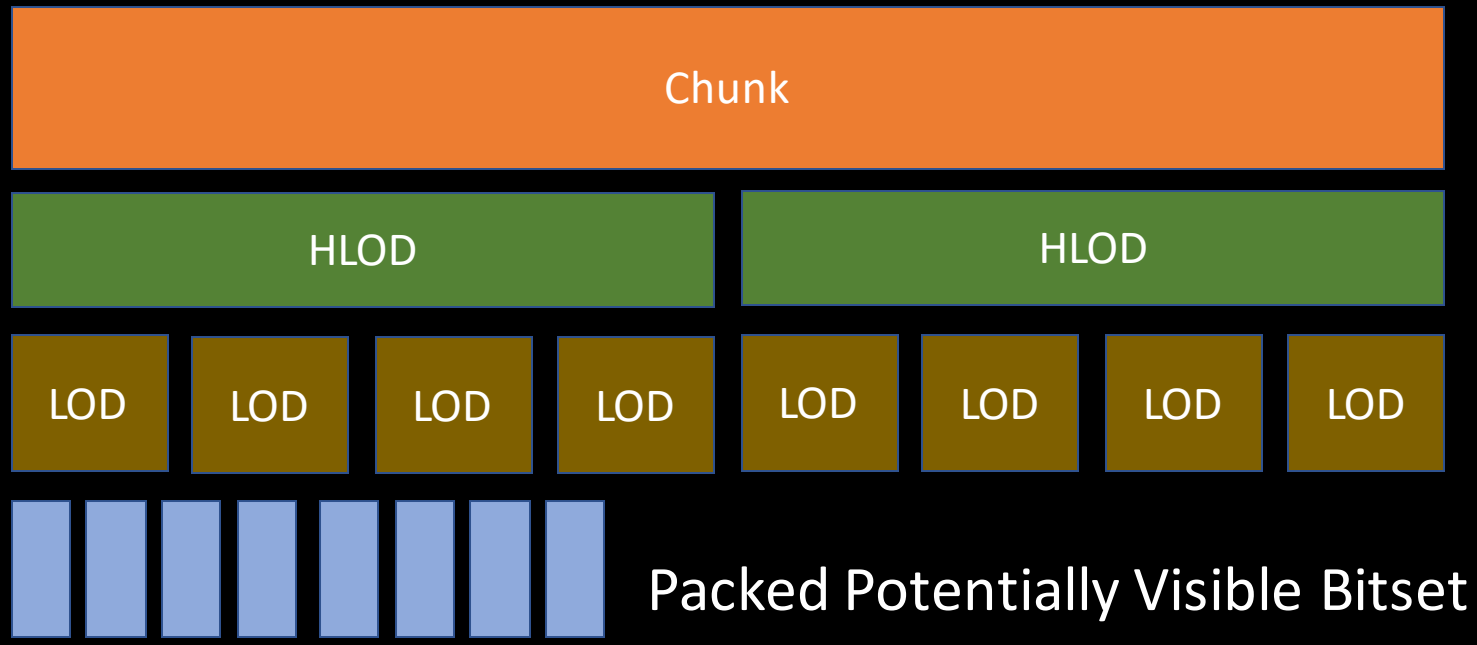
LOD

- What about LOD?
- Most instance LODs are *unique* (not mesh combinations)
- What about HLOD?

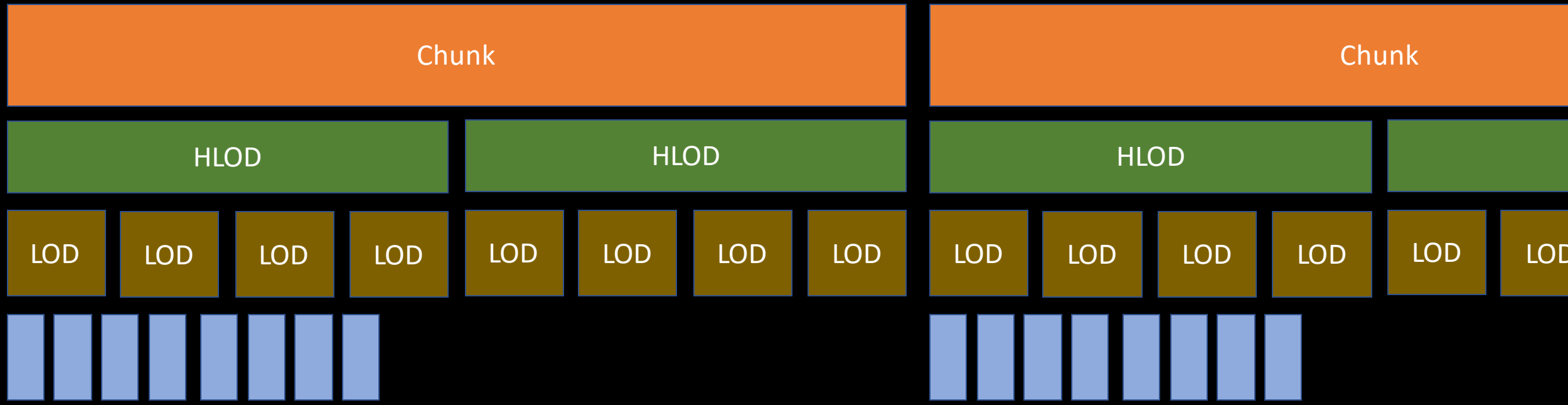

LocalToWorld
WorldRenderBounds
ChunkWorldRenderBounds
LocalToWorld
WorldRenderBounds
MeshRenderer
LocalToWorld
WorldRenderBounds
Subscene A
LocalToWorld
WorldRenderBounds
ChunkWorldRenderBounds
LocalToWorld
WorldRenderBounds
MeshRenderer
LocalToWorld
WorldRenderBounds
Subscene B
Chunks by
• Same Renderer
• Same Subscene
99+% of chunks have only one HLOD root


LocalToWorld
LOD 0
HLOD 0
5
LocalToWorld
LOD 0
LocalToWorld
LOD 0
LocalToWorld
LOD 0
LocalToWorld
LOD 1
LocalToWorld
LOD 1
HLOD 1
1

LOD loop is Simple.

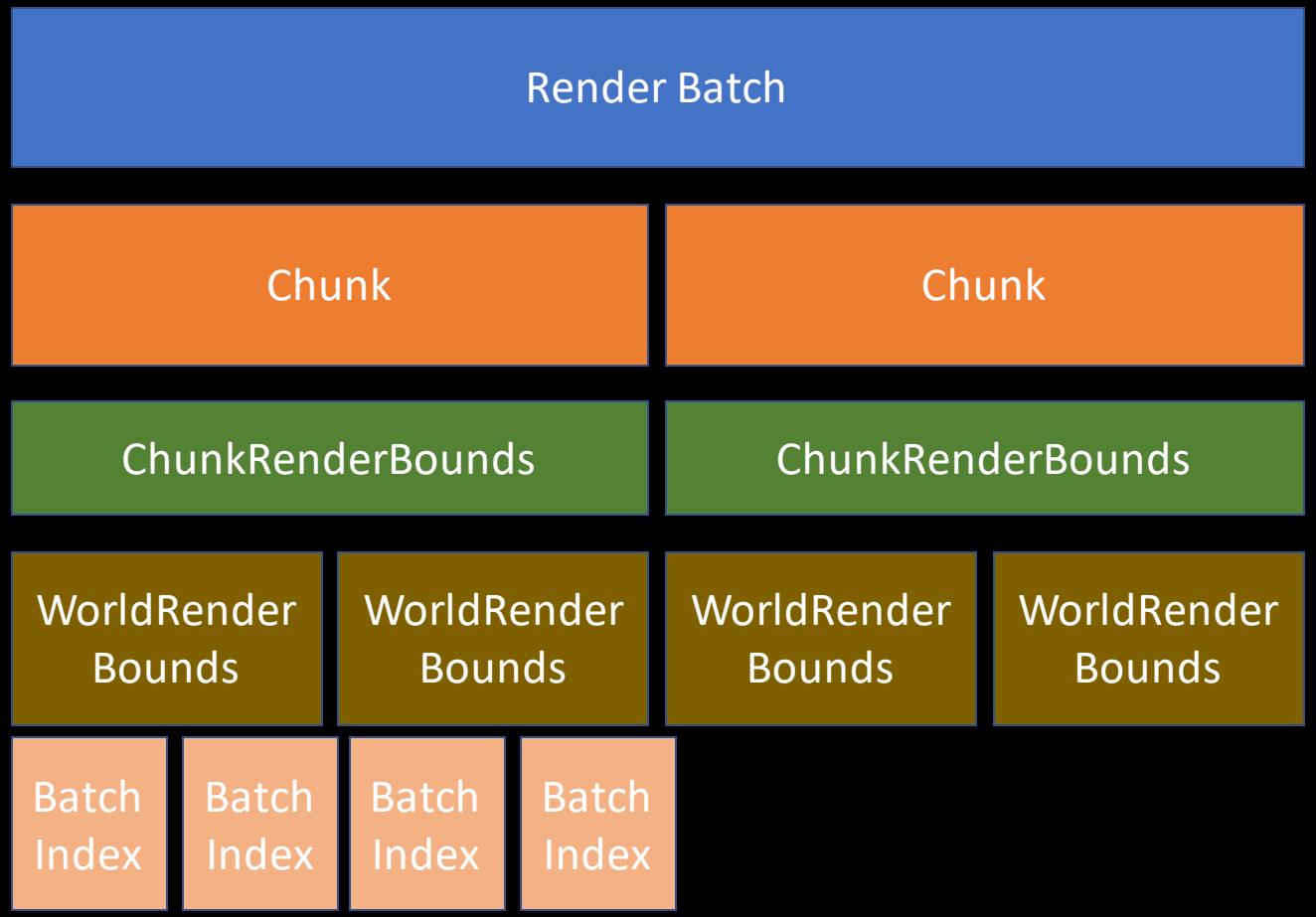


LOD loop is Simple. (Chunks in parallel.)

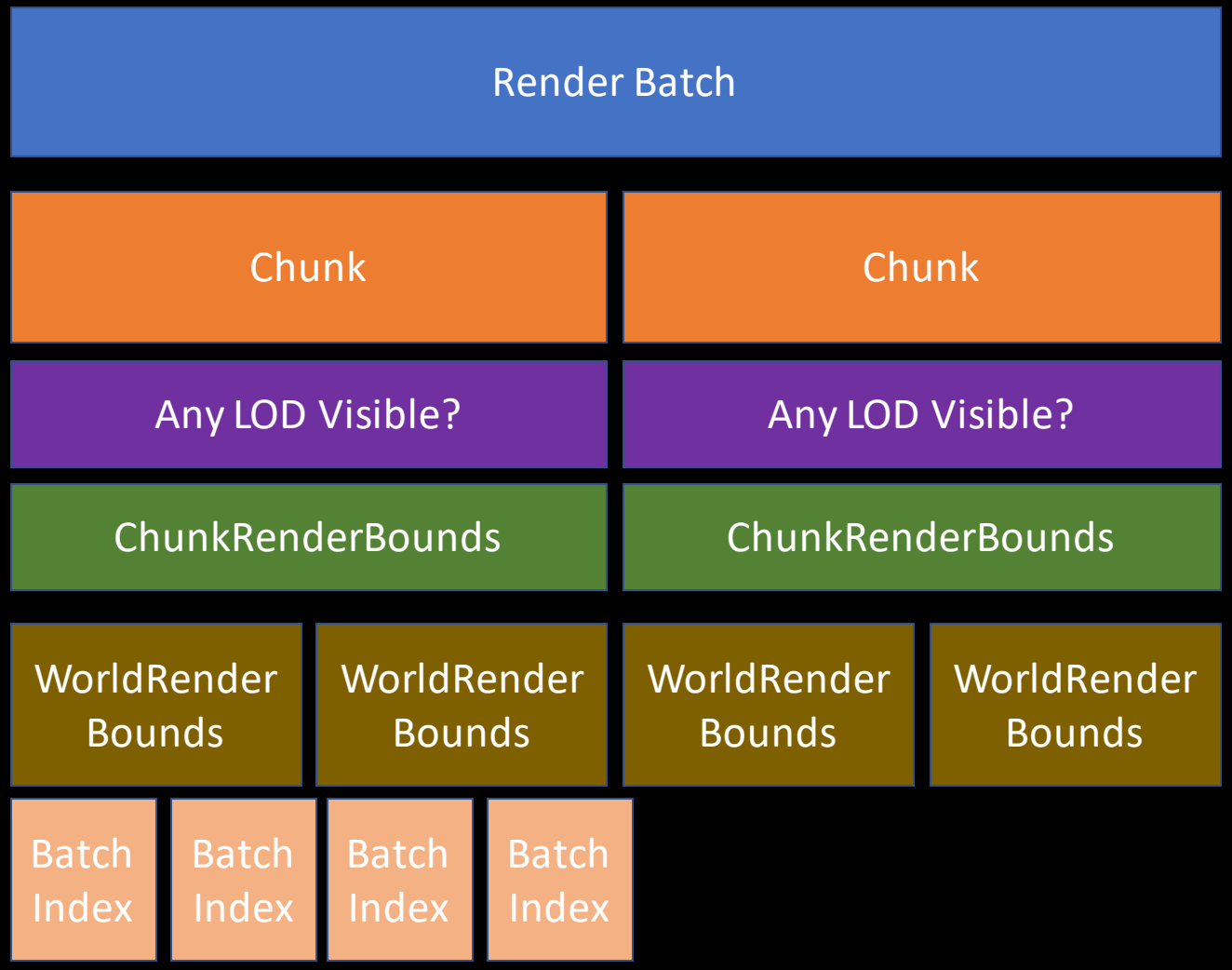


Putting them together

Look at culling loop again…



Early out based on LOD visibility



Recap

- Simple culling loop
- Simple LOD loop
- Because we could reason about how data was stored…
- …and some basic statistics about that data.

Key to Data-Oriented Practice = Data Transparency