



**GAME DEVELOPERS
CONFERENCE**

INSTANCING AND OIT IN TOTAL WAR:THREE KINGDOMS

Tamas Rabel
Rendering Technical Director
Creative Assembly



Campaign is turn-based



@IntelSoftware

@IntelGraphics

@totalwar

@CAGames





LÜ BU PERISHES AT XIAPI

22 1728

Spring

	40	147	158		152	153		160	4
	143	157			141	145		160	3

Battle is RTS

AGENDA

Instancing pipeline

Worker threads

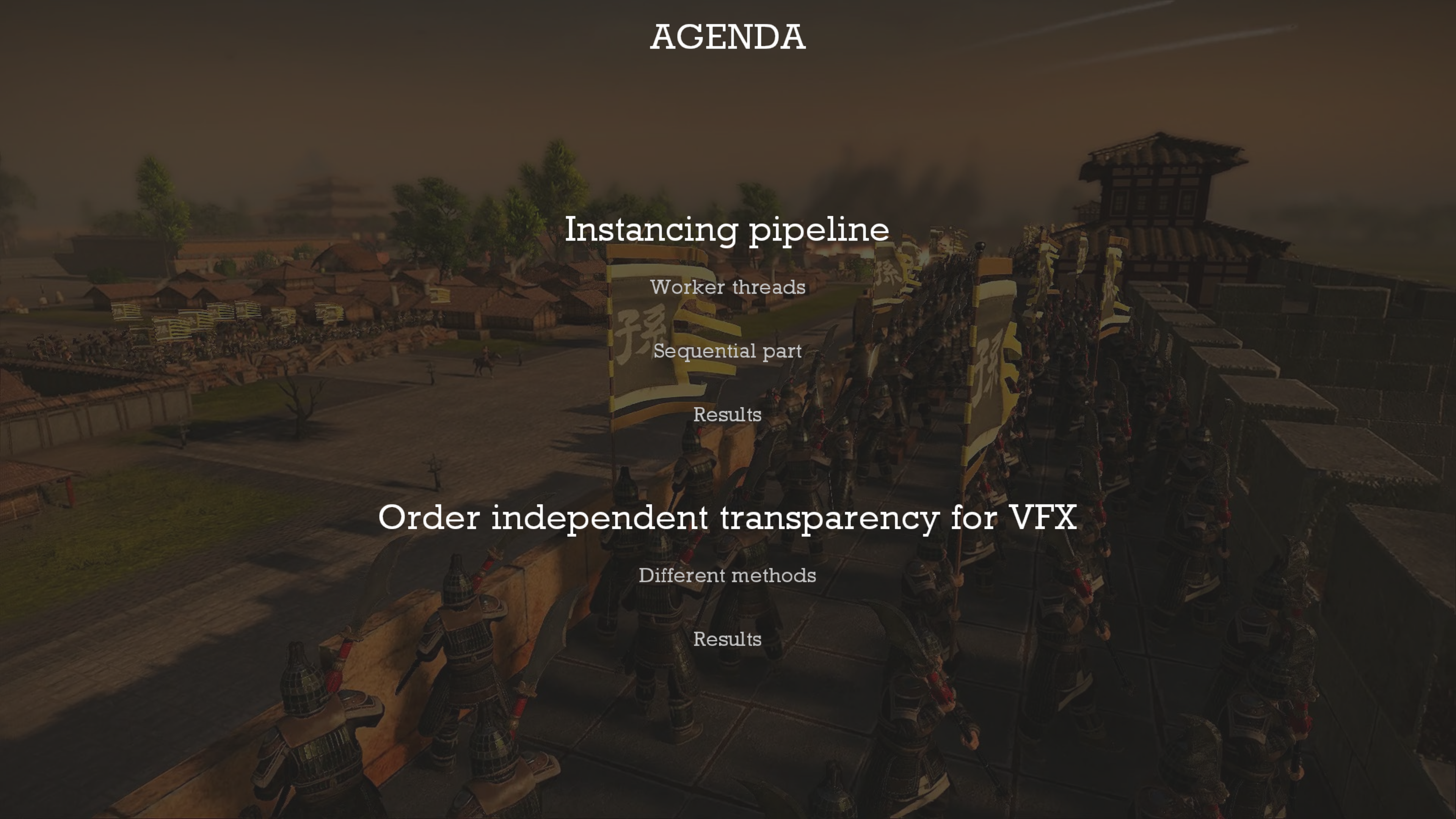
Sequential part

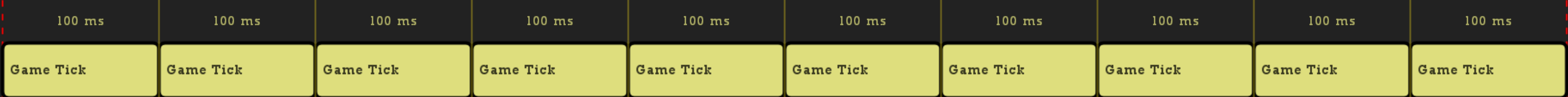
Results

Order independent transparency for VFX

Different methods

Results





Game Tick thread is running ~10 fps



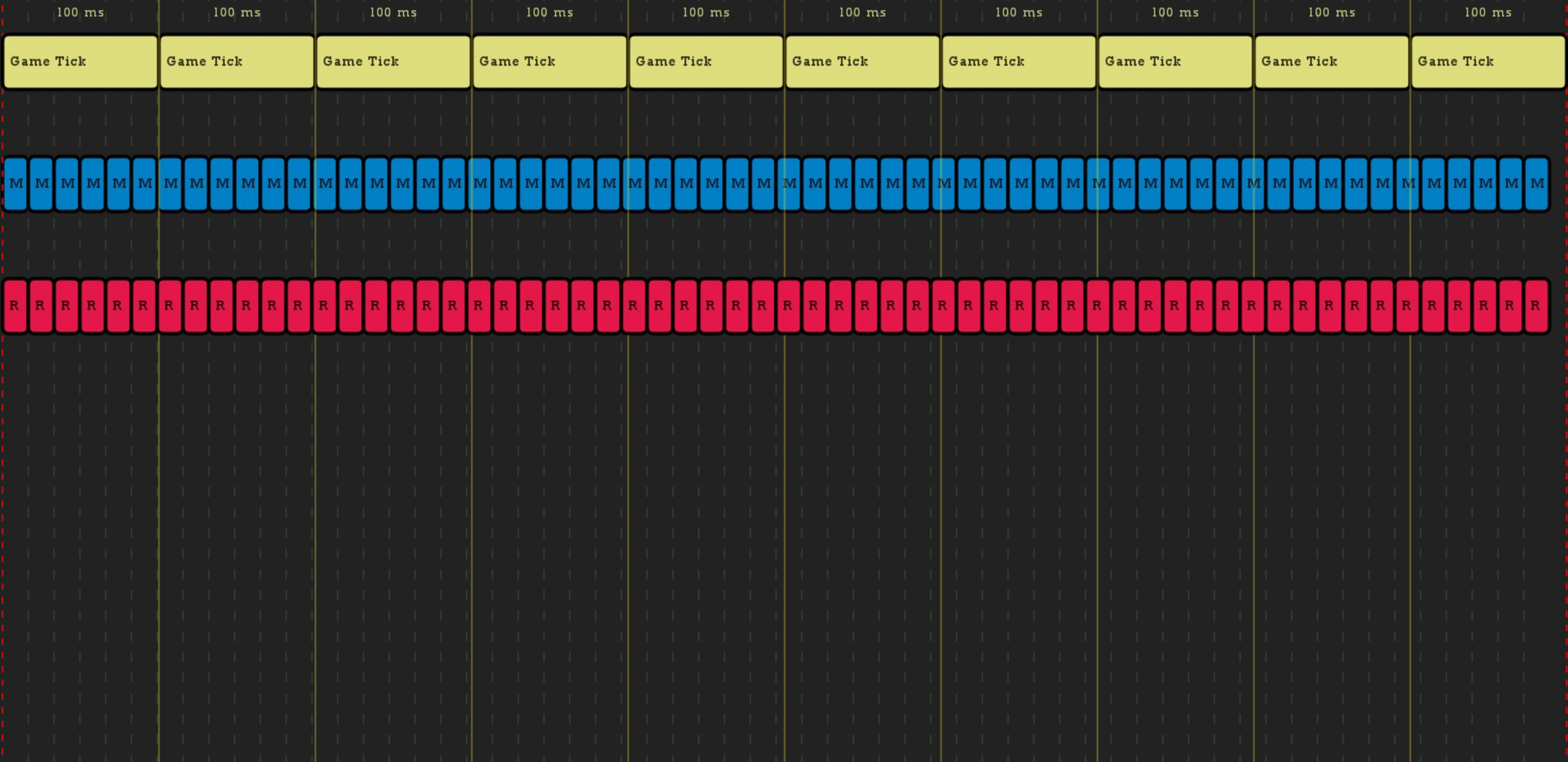
@IntelSoftware

@IntelGraphics

@totalwar

@CAGames



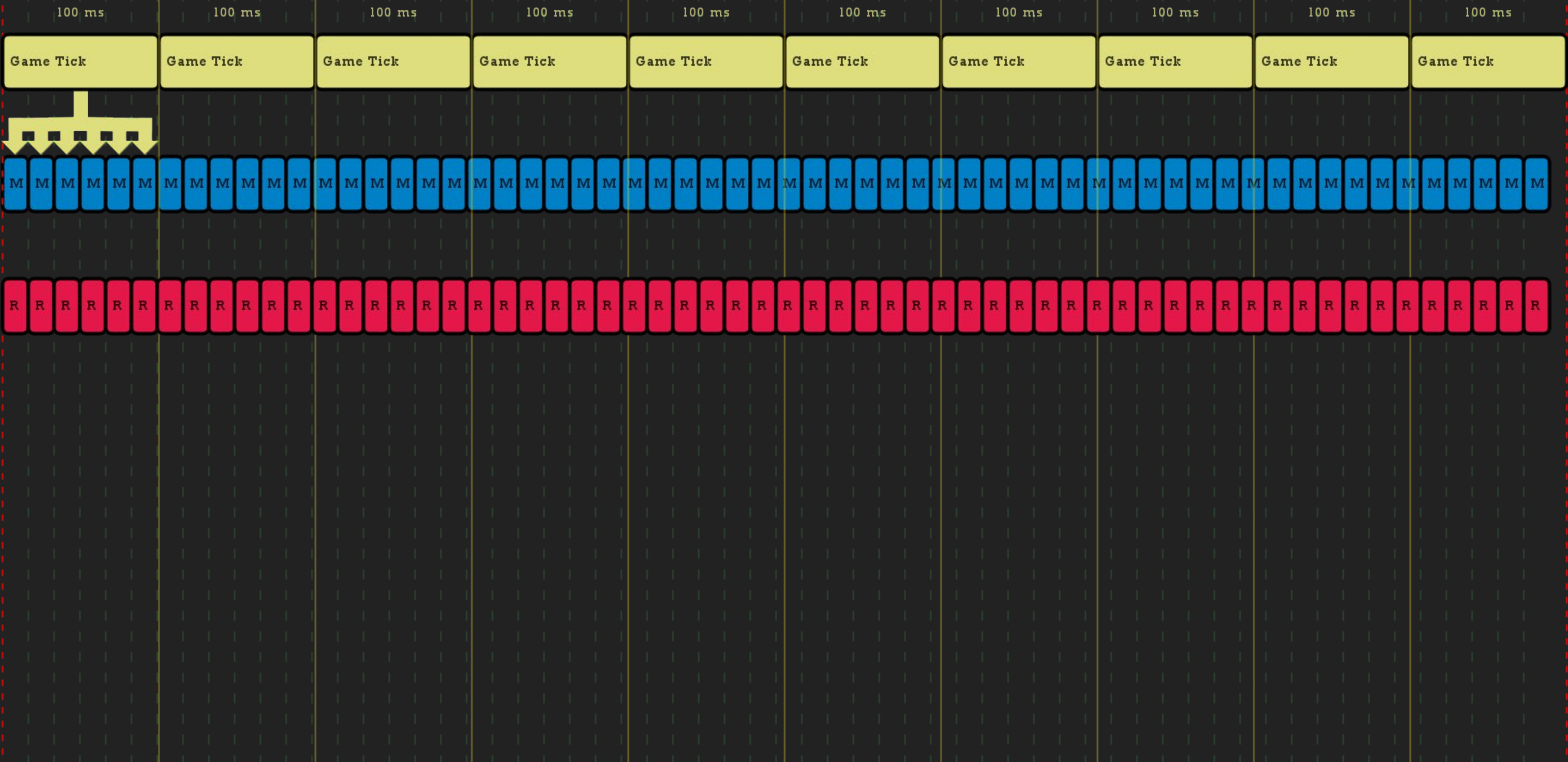


Main Thread and Render Thread are running in sync



@IntelSoftware @IntelGraphics @totalwar @CAGames





Game Tick builds game state and passes it to main thread





@IntelSoftware

@IntelGraphics

@totalwar

@CAGames

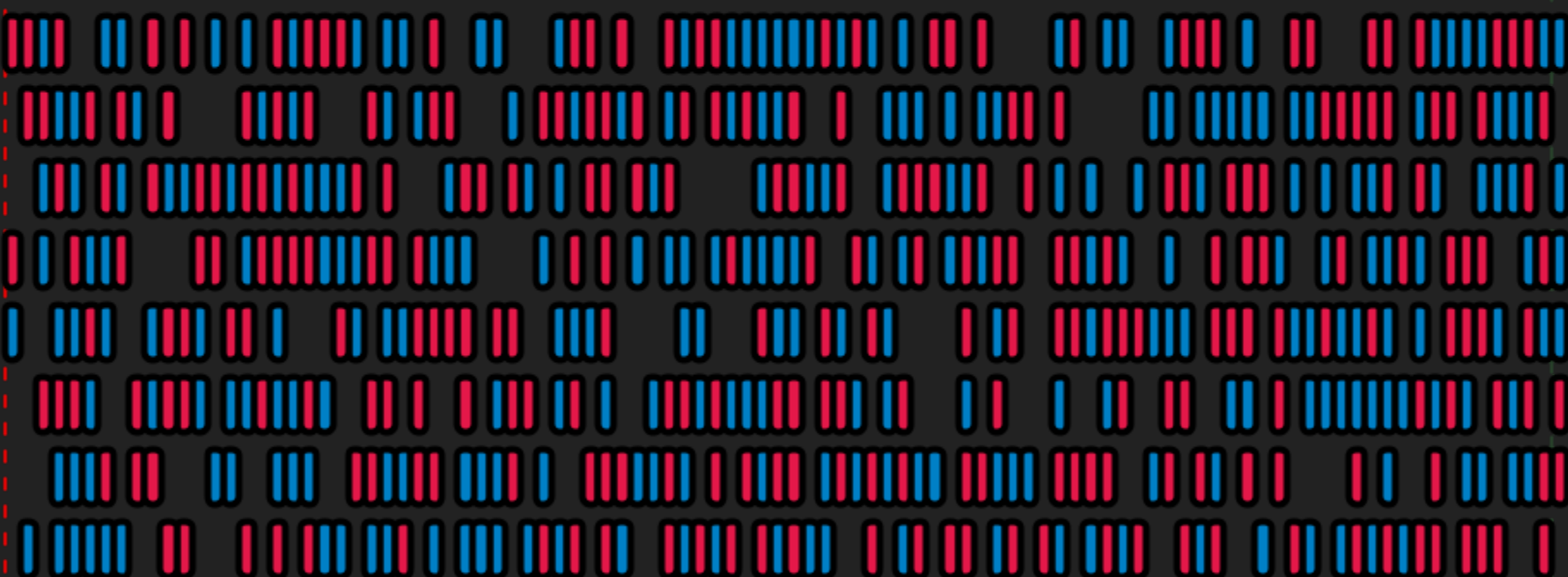


100 ms



16 ms

16 ms



One for each physical core

Worker Threads



10 ms

Game Tick

Main Thread

Main Thread

Main Thread

Render Thread

Render Thread

Render Thread

16 ms

16 ms



One for each physical core



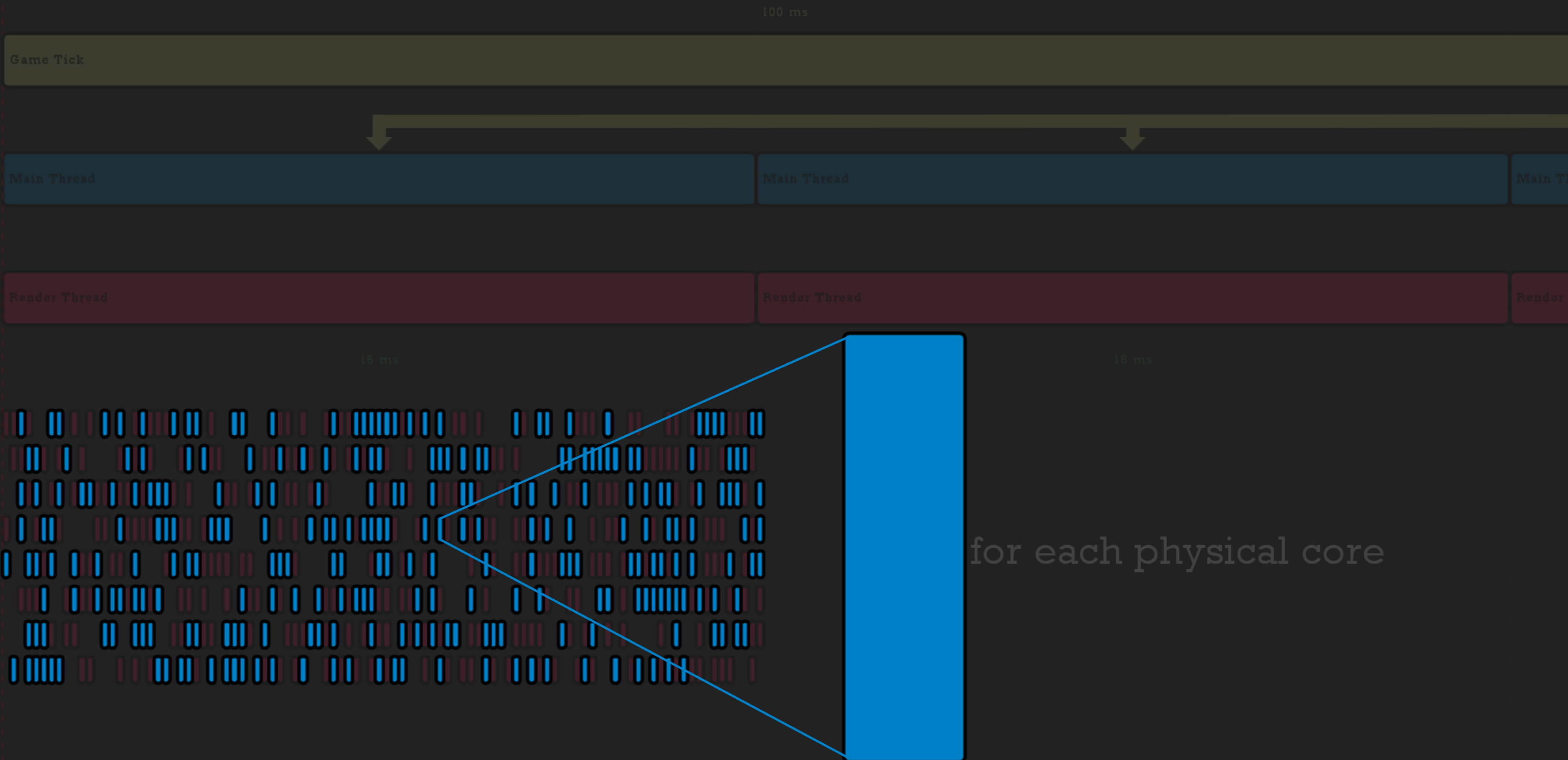
@IntelSoftware

@IntelGraphics

@totalwar

@CAGames





@IntelSoftware

@IntelGraphics

@totalwar

@CAGames



WORKER THREAD FOR MT



@IntelSoftware

@IntelGraphics

@totalwar

@CAGames





CULLING



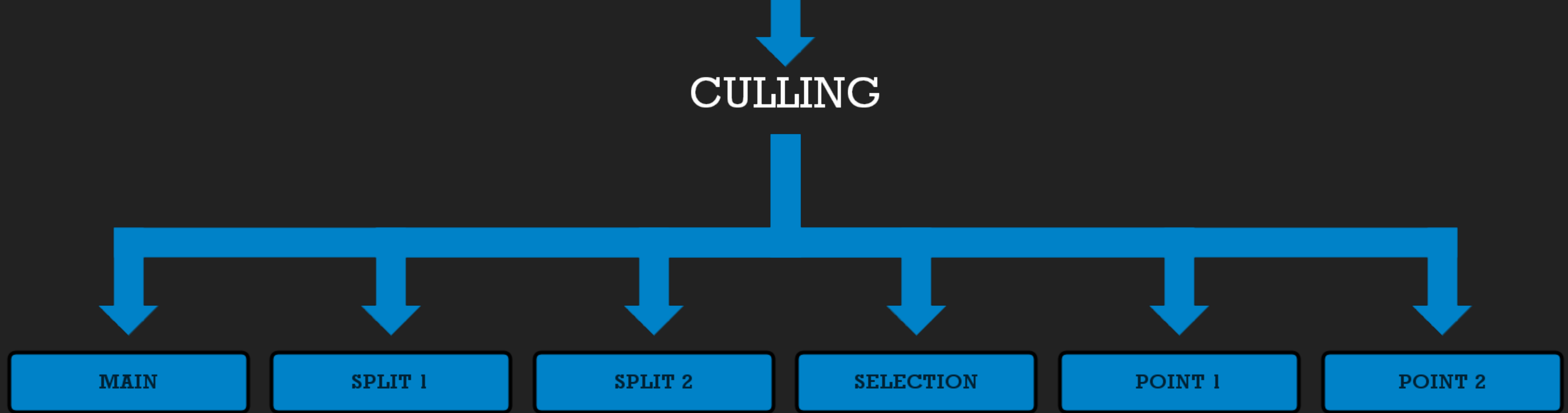
@IntelSoftware

@IntelGraphics

@totalwar

@CAGames





We calculate intersection with all the frustums at once



@IntelSoftware

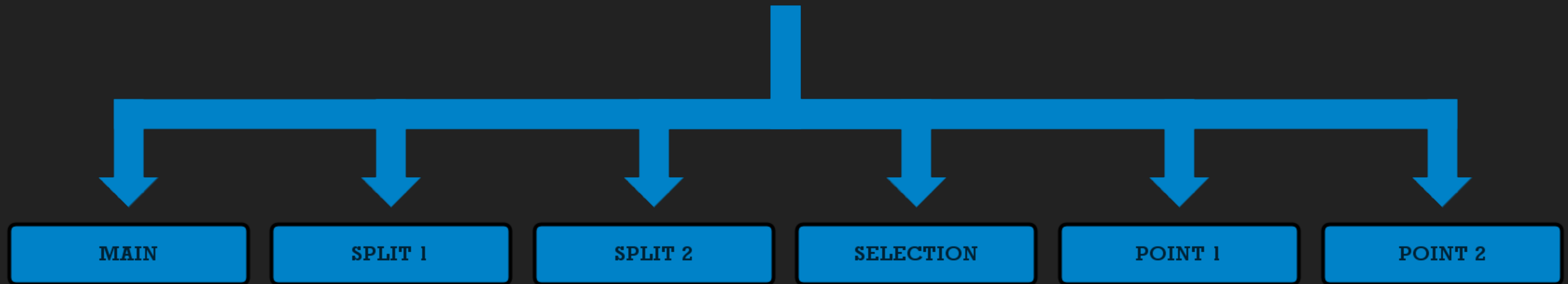
@IntelGraphics

@totalwar

@CAGames



CULLING



BITFLAG (64 BITS)

Intersection results are stored in a bitfield



@IntelSoftware

@IntelGraphics

@totalwar

@CAGames





LOD SELECTION

PROJECT BOUNDING BOX

Small Object Culling

We calculate the area of the projected bounding box and cull small models



@IntelSoftware

@IntelGraphics

@totalwar

@CAGames



LOD SELECTION

PROJECT BOUNDING BOX

Small Object Culling

AVERAGE TRIANGLE DENSITY

LOD 0

LOD 1

LOD 2

LOD 3

Then we select the LOD level based on the approximated average triangle size on screen



@IntelSoftware

@IntelGraphics

@totalwar

@CAGames



LOD SELECTION

PROJECT BOUNDING BOX

Small Object Culling

AVERAGE TRIANGLE DENSITY

LOD 0

LOD 1

LOD 2

LOD 3

LOD 1

Then we select the LOD level based on the approximated average triangle size on screen



@IntelSoftware

@IntelGraphics

@totalwar

@CAGames



l1:L1:P0:chinese_crossbow_2h_01_lod2:VariantMeshes/_VariantModels/props/chinese/rigid/chinese_crossbow_2h_01_rigid_model_v2 (VariantMeshes/_VariantModels/props/chinese/weighted/materials/chinese_crossbow_2h_01_rigid_alpha_off.xml.material)
l1:L1:P0:neckstump_01_lod2:VariantMeshes/_VariantModels/character/stumps/neckstump_01_rigid_model_v2 (shaders/system/rigid_standard.xml.shader_dx000002265ec697e8_0x000002265ec68c98)
l1:L1:P0:1h_sword_jian_upgrade_lod2:VariantMeshes/_VariantModels/props/chinese/rigid/1h_sword_jian_upgrade_rigid_model_v2 (VariantMeshes/_VariantModels/props/chinese/rigid/materials/1h_sword_jian_upgrade_alpha_off.xml.material)
l1:L1:P0:generic_head_07:VariantMeshes/_VariantModels/character/human/male/chinese/components/skin/generic_head_07_rigid_model_v2 (VariantMeshes/_VariantModels/character/human/male/chinese/components/skin/generic_head_07_weighted4_alpha_off.xml.material)
l1:L1:P0:merged_helmet_plates_01_common:VariantMeshes/_VariantModels/character/human/male/chinese/merged_vmd/unit_medium_leather_1/unit_medium_leather_1_3_rigid_model_v2 (VariantMeshes/_VariantModels/character/human/male/chinese/merged_vmd/unit_medium_leather_1/materials/unit_medium_leather_1_weighted4_alpha_off.xml.material)
l1:L1:P1:merged_trousers_fabric_01:VariantMeshes/_VariantModels/character/human/male/chinese/merged_vmd/unit_medium_leather_1/unit_medium_leather_1_3_rigid_model_v2 (VariantMeshes/_VariantModels/character/human/male/chinese/merged_vmd/tonics_trousers/materials/tonics_trousers_weighted4_alpha_off.xml.material)

LOD PARTS



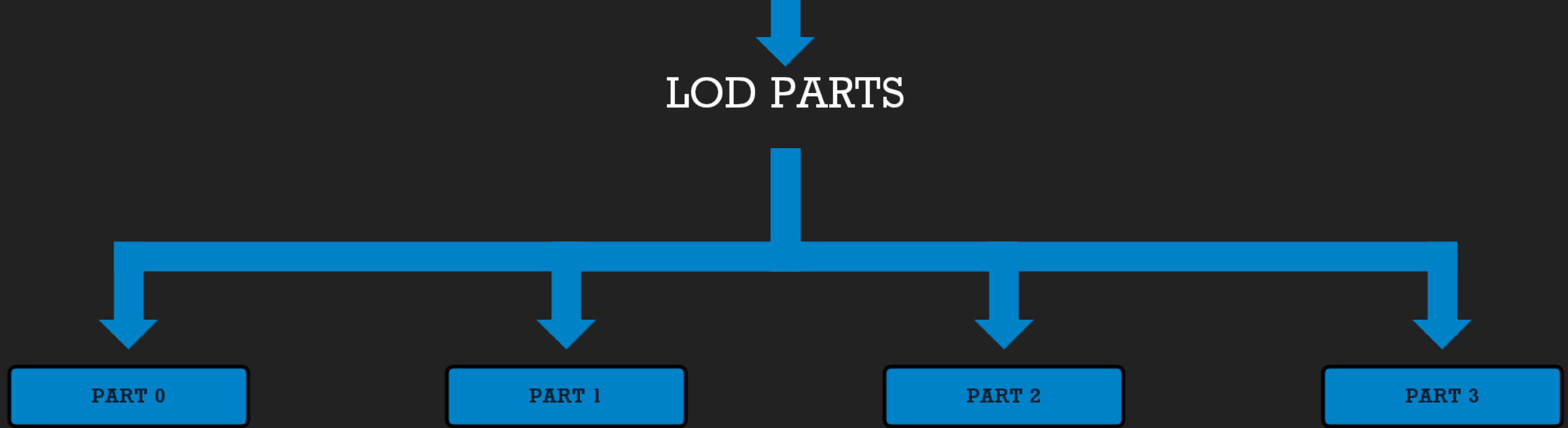
@IntelSoftware

@IntelGraphics

@totalwar

@CAGames





Triangles with different materials are different parts

Part is what can be rendered with a single draw call

In our terminology a part is a MESH



@IntelSoftware

@IntelGraphics

@totalwar

@CAGames





LOD PARTS

3-5 parts on average plus weapons/shield

3 different variations on average

$5 * 3 = 15$ draw calls per unit

(Unit is a group of soldiers, players control units)



@IntelSoftware

@IntelGraphics

@totalwar

@CAGames





LOD PARTS

3-5 parts on average plus weapons/shield

3 different variations on average

Heads are using more variations

$$5 * 3 + 5 = 20 \text{ draw calls per unit}$$

(Unit is a group of soldiers, players control units)



@IntelSoftware

@IntelGraphics

@totalwar

@CAGames





LOD PARTS

3-5 parts on average plus weapons/shield

3 different variations on average

Heads are using more variations

Each unit is using 2 lods on average

$$(5*3 + 5) * 2 = 40 \text{ draw calls per unit}$$

(Unit is a group of soldiers, players control units)



@IntelSoftware

@IntelGraphics

@totalwar

@CAGames





LOD PARTS

3-5 parts on average plus weapons/shield

3 different variations on average

Heads are using more variations

Each unit is using 2 lods on average

Everything is rendered into the shadow map too

$$(5*3 + 5) * 2 * 2 = 80 \text{ draw calls per unit}$$

(Unit is a group of soldiers, players control units)



@IntelSoftware

@IntelGraphics

@totalwar

@CAGames





LOD PARTS

3-5 parts on average plus weapons/shield

3 different variations on average

Heads are using more variations

Each unit is using 2 lods on average

Everything is rendered into the shadow map too

$(5*3 + 5) * 2 * 2 = 80$ draw calls per unit

$(5*3 + 5) * 2 * 2 * 42 = 3360$ draw calls (for ~5k soldiers)

(Unit is a group of soldiers, players control units)




@IntelSoftware

@IntelGraphics

@totalwar

@CAGames





LOD PARTS

Imposters are always 1 draw call

Imposters are super low poly (<200 tris)



@IntelSoftware

@IntelGraphics

@totalwar

@CAGames





LOD PARTS

Imposters are always 1 draw call

Imposters are super low poly (<200 tris)

Imposters also have two lod levels (<200 tris and <50 tris)



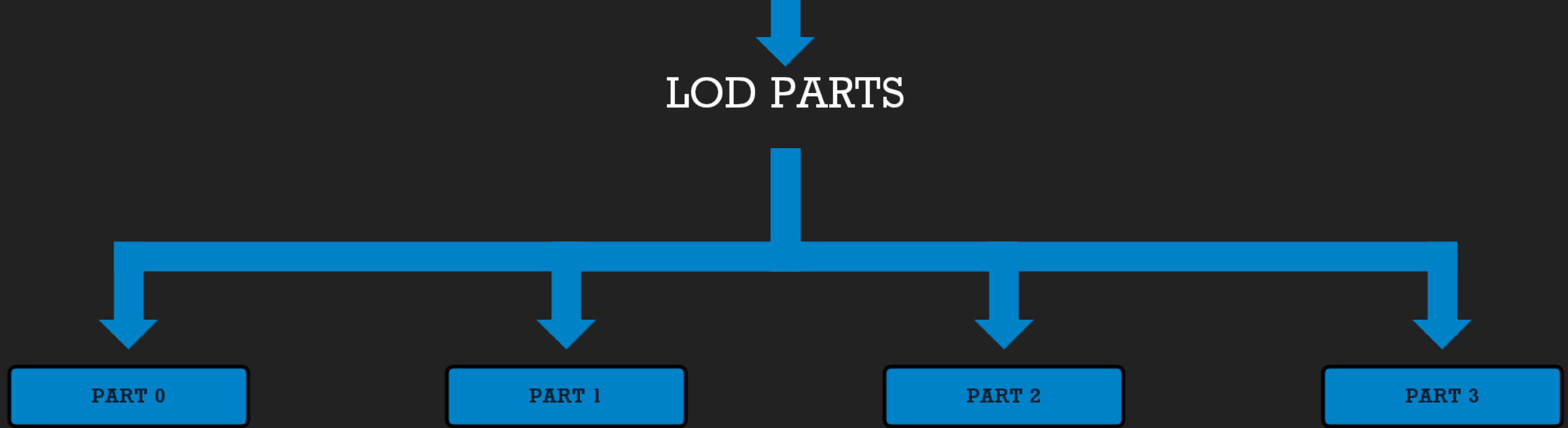
@IntelSoftware

@IntelGraphics

@totalwar

@CAGames





Triangles with different materials are different parts

Part is what can be rendered with a single draw call

In our terminology a parts is a MESH



@IntelSoftware

@IntelGraphics

@totalwar

@CAGames



PART 1

Next step is building instance data for the individual parts



@IntelSoftware

@IntelGraphics

@totalwar

@CAGames





INSTANCE DATA

Every mesh type has its own instance data format

Different materials use the same instance data

This is what will be uploaded to the GPU



@IntelSoftware

@IntelGraphics

@totalwar

@CAGames



PIPELINE STAGES



Based on the mesh and instance data and the material we figure out which parts of the pipeline the mesh needs to be rendered to.



@IntelSoftware

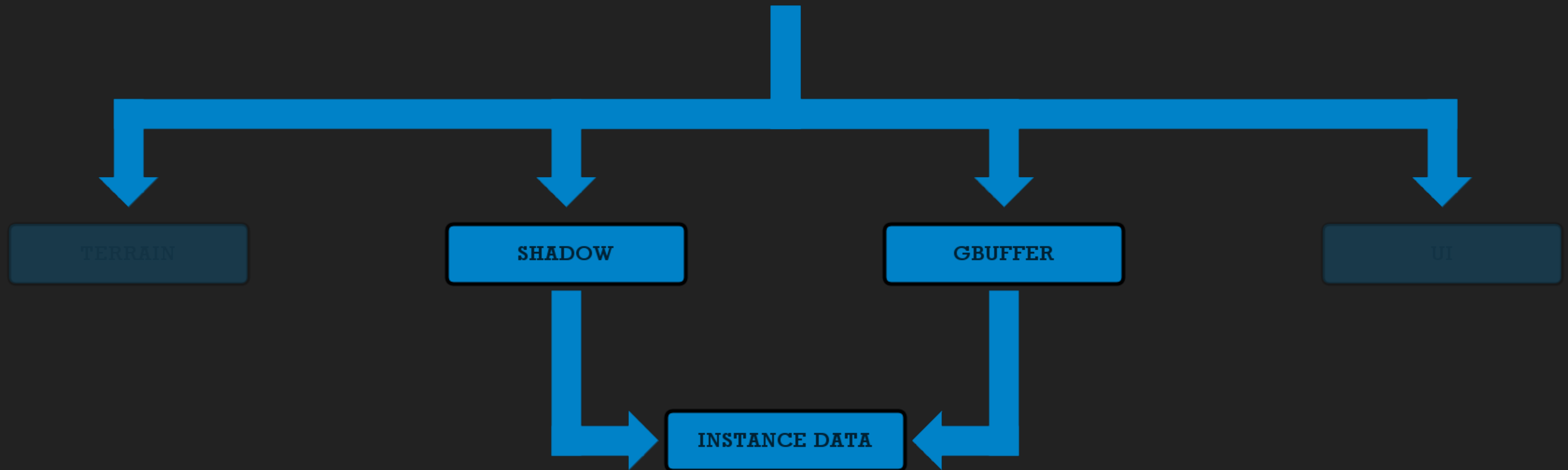
@IntelGraphics

@totalwar

@CAGames



PIPELINE STAGES



Each pipeline stage uses the same instance data.



@IntelSoftware

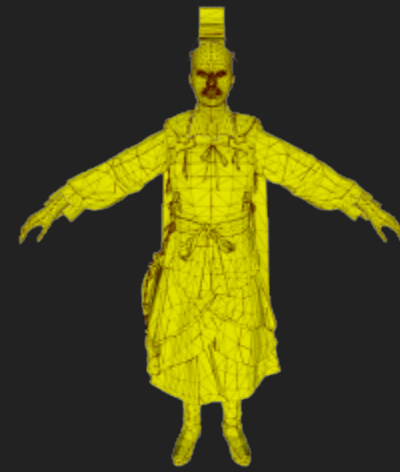
@IntelGraphics

@totalwar

@CAGames

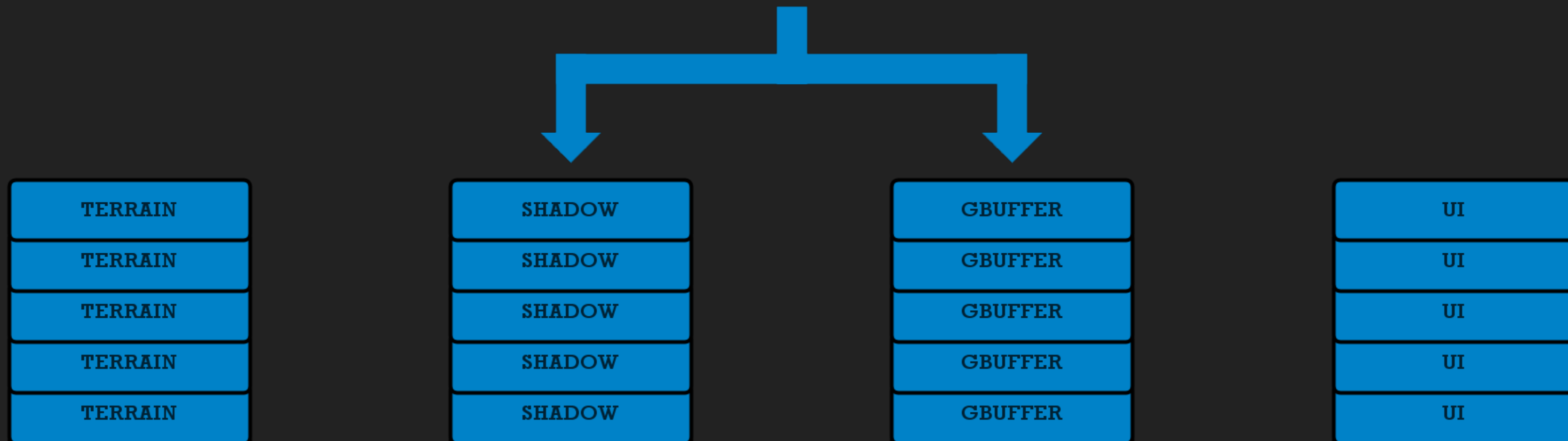


INSTANCE LISTS



Every single mesh has a number of instance lists.
One instance list per pipeline stage.
The instance is added to all the relevant instance lists.

INSTANCE LISTS



The instance lists are duplicated to every worker thread to ensure lockless access.



@IntelSoftware

@IntelGraphics

@totalwar

@CAGames



INSTANCE LISTS



TERRAIN	TERRAIN
TERRAIN	TERRAIN
TERRAIN	TERRAIN
TERRAIN	TERRAIN
TERRAIN	TERRAIN
TERRAIN	TERRAIN
TERRAIN	TERRAIN
TERRAIN	TERRAIN

SHADOW	SHADOW
SHADOW	SHADOW
SHADOW	SHADOW
SHADOW	SHADOW
SHADOW	SHADOW
SHADOW	SHADOW
SHADOW	SHADOW
SHADOW	SHADOW

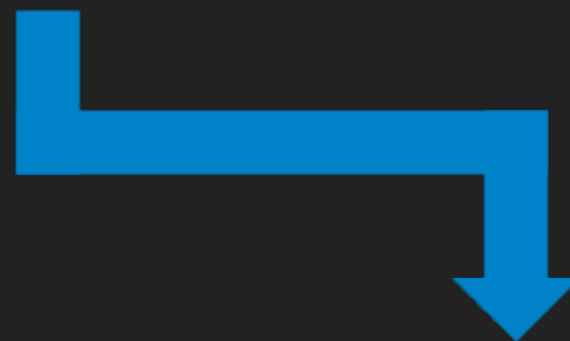
GBUFFER	GBUFFER
GBUFFER	GBUFFER
GBUFFER	GBUFFER
GBUFFER	GBUFFER
GBUFFER	GBUFFER
GBUFFER	GBUFFER
GBUFFER	GBUFFER
GBUFFER	GBUFFER

UI	UI
UI	UI
UI	UI
UI	UI
UI	UI
UI	UI
UI	UI
UI	UI

And double buffered because while the render thread is rendering the current frame, the main thread (and workers) is adding instances to the next frame's instance lists.



MESH LISTS



TERRAIN

RIGID

WEIGHTED

BILLBOARD

We have one mesh list per mesh type.
Every mesh with at least a single instance in the frame
will get added to exactly one mesh list matching its mesh type.



@IntelSoftware

@IntelGraphics

@totalwar

@CAGames



MESH LISTS



Just like the instance lists we have one mesh list for every single worker thread.

TERRAIN

RIGID

WEIGHTED

BILLBOARD



@IntelSoftware

@IntelGraphics

@totalwar

@CAGames



MESH LISTS



TERRAIN	TERRAIN
TERRAIN	TERRAIN
TERRAIN	TERRAIN
TERRAIN	TERRAIN
TERRAIN	TERRAIN
TERRAIN	TERRAIN
TERRAIN	TERRAIN

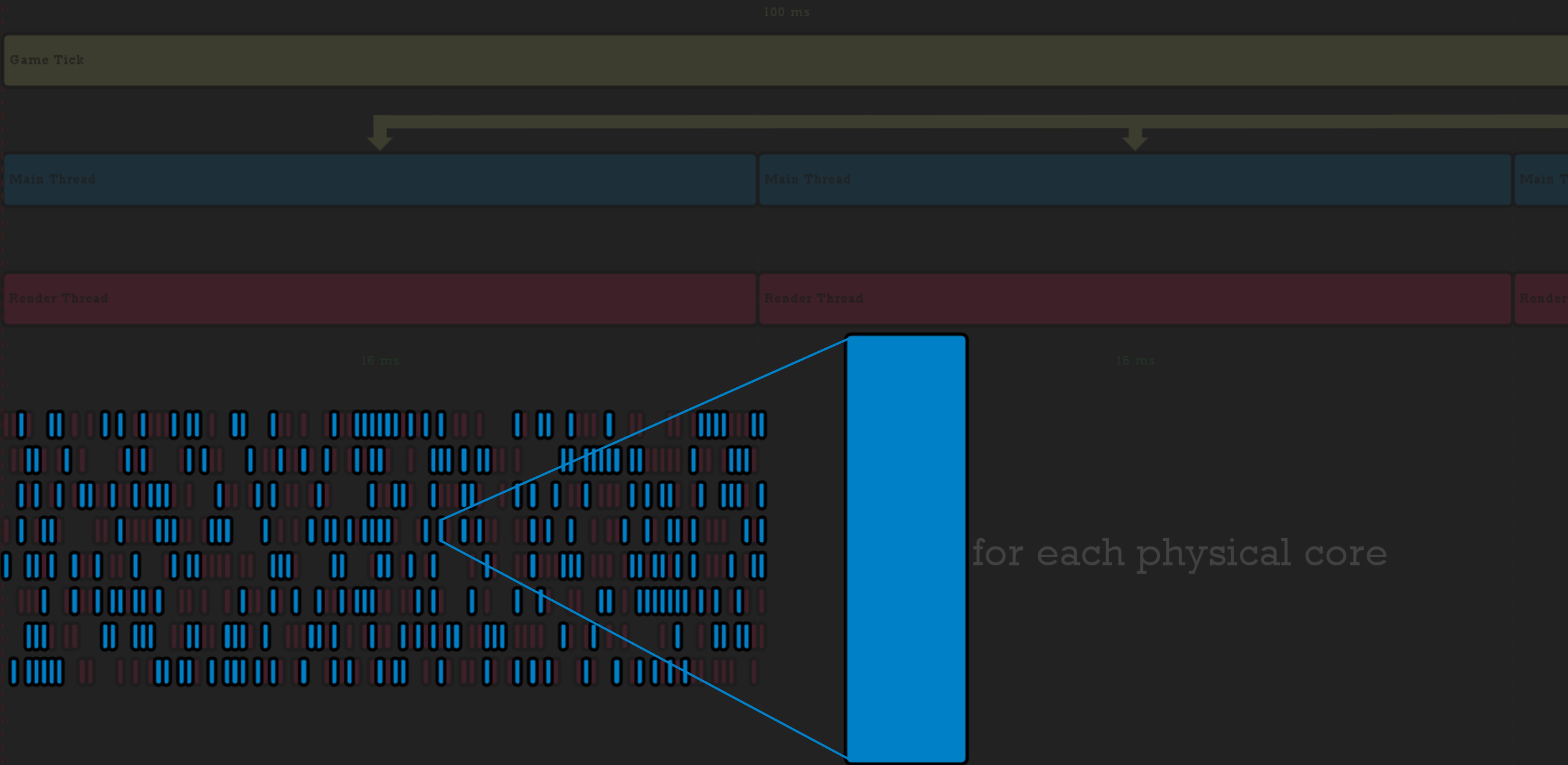
RIGID	RIGID
RIGID	RIGID
RIGID	RIGID
RIGID	RIGID
RIGID	RIGID
RIGID	RIGID
RIGID	RIGID

WEIGHTED	WEIGHTED
WEIGHTED	WEIGHTED
WEIGHTED	WEIGHTED
WEIGHTED	WEIGHTED
WEIGHTED	WEIGHTED
WEIGHTED	WEIGHTED
WEIGHTED	WEIGHTED

BILLBOARD	BILLBOARD
BILLBOARD	BILLBOARD
BILLBOARD	BILLBOARD
BILLBOARD	BILLBOARD
BILLBOARD	BILLBOARD
BILLBOARD	BILLBOARD
BILLBOARD	BILLBOARD

Double buffered too.





@IntelSoftware

@IntelGraphics

@totalwar

@CAGames



100 ms



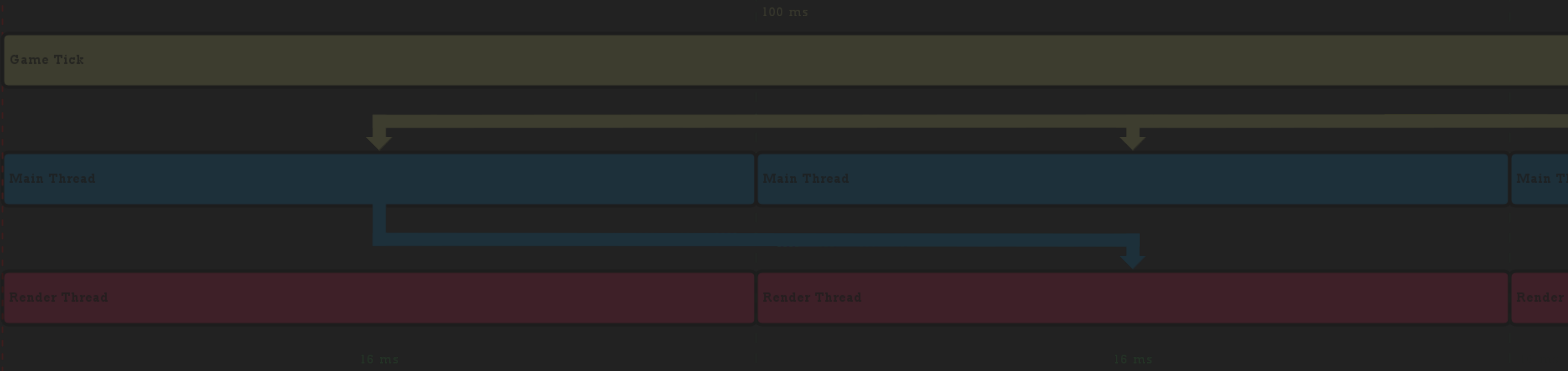
16 ms

16 ms



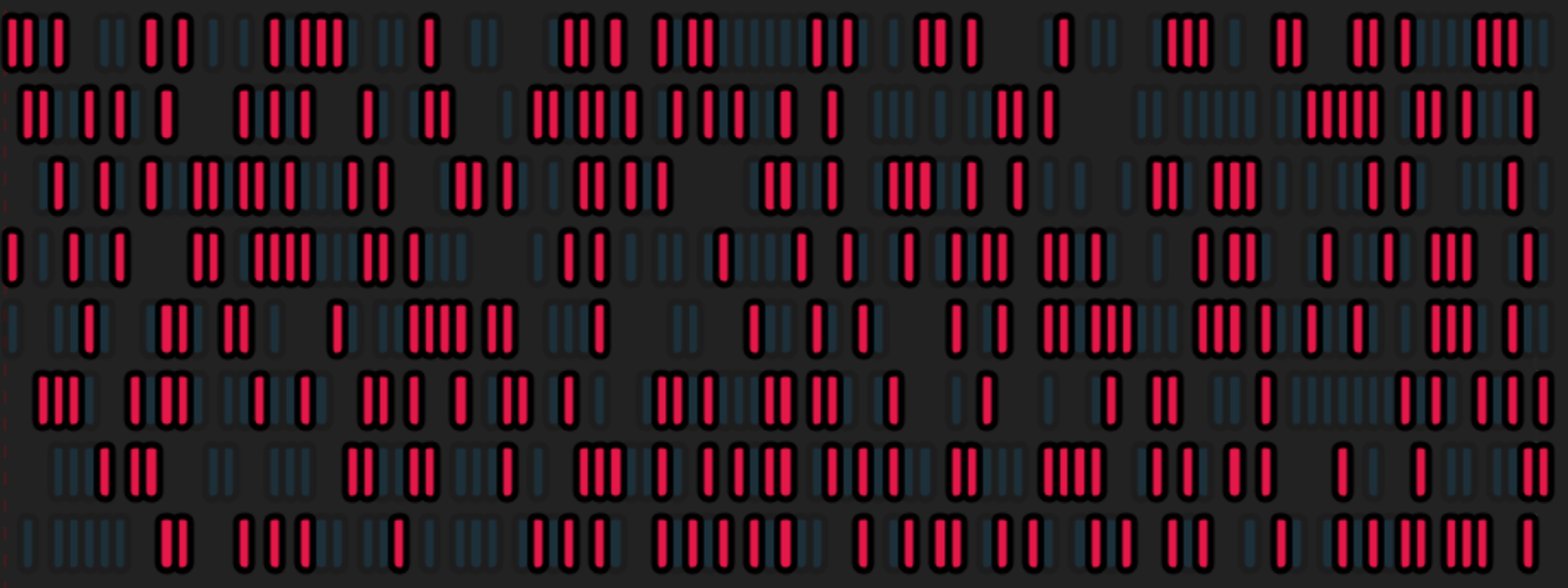
@IntelSoftware @IntelGraphics @totalwar @CAGames

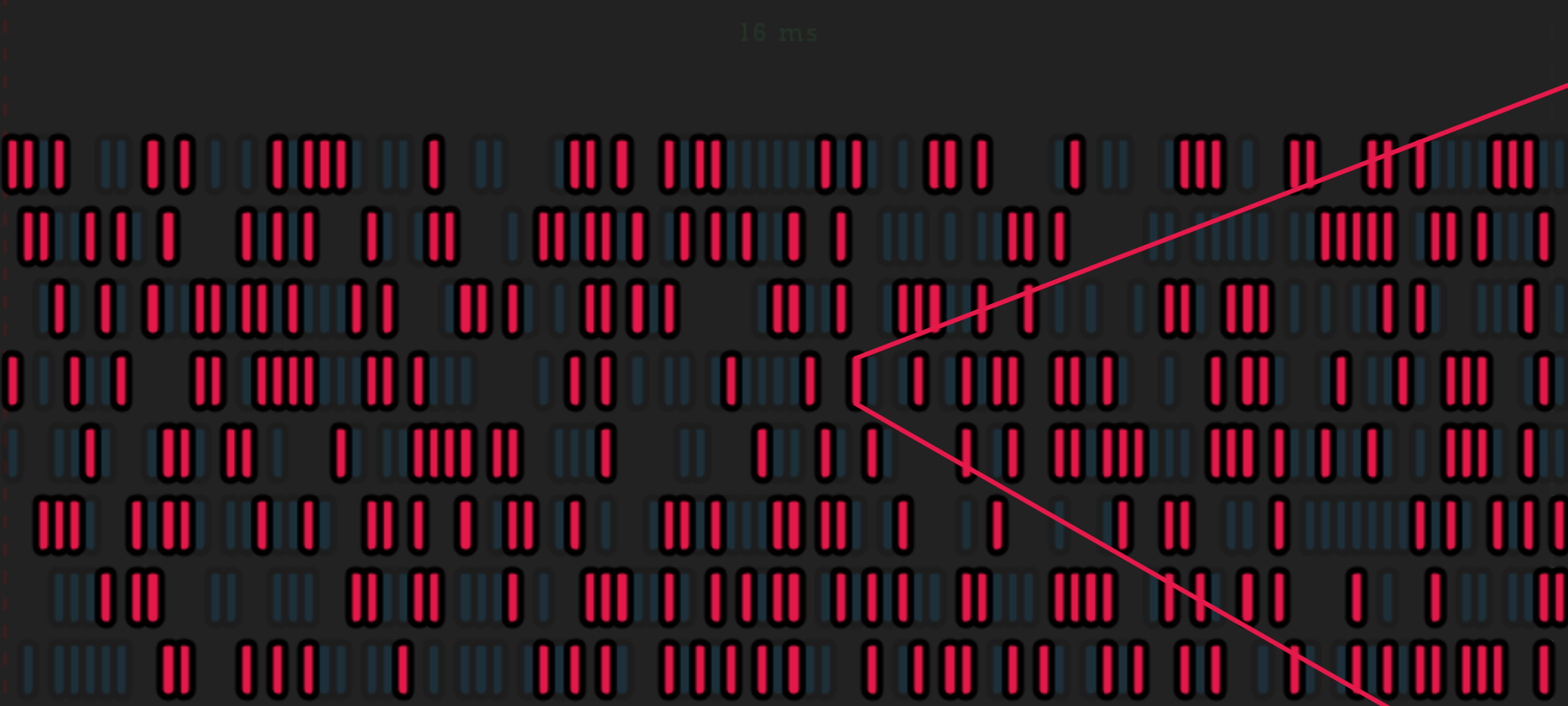
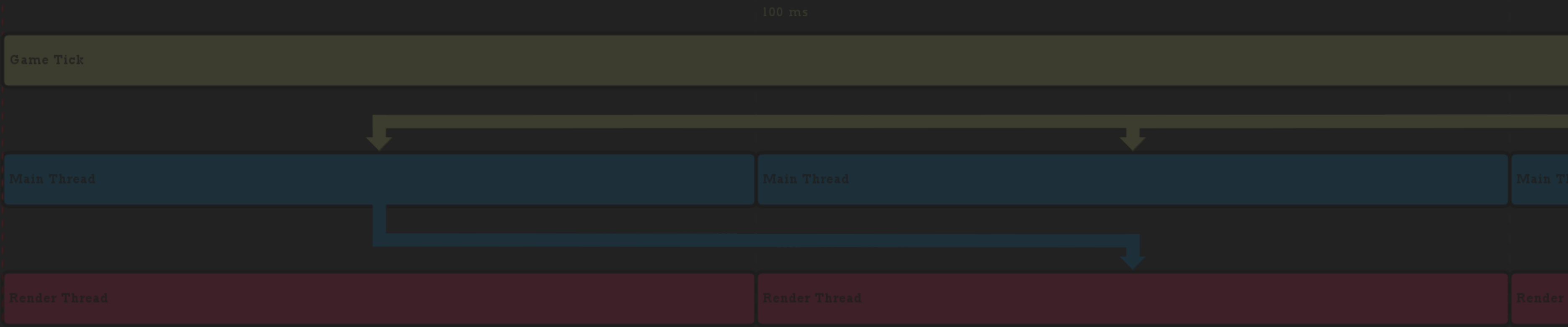




16 ms

16 ms





WORKER THREAD FOR RT



@IntelSoftware

@IntelGraphics

@totalwar

@CAGames





MESH LIST OF ONE MESH TYPE

TERRAIN	TERRAIN
TERRAIN	TERRAIN
TERRAIN	TERRAIN
TERRAIN	TERRAIN
TERRAIN	TERRAIN
TERRAIN	TERRAIN
TERRAIN	TERRAIN
TERRAIN	TERRAIN
TERRAIN	TERRAIN

RIGID	RIGID
RIGID	RIGID
RIGID	RIGID
RIGID	RIGID
RIGID	RIGID
RIGID	RIGID
RIGID	RIGID
RIGID	RIGID
RIGID	RIGID

WEIGHTED	WEIGHTED
WEIGHTED	WEIGHTED
WEIGHTED	WEIGHTED
WEIGHTED	WEIGHTED
WEIGHTED	WEIGHTED
WEIGHTED	WEIGHTED
WEIGHTED	WEIGHTED
WEIGHTED	WEIGHTED
WEIGHTED	WEIGHTED

BILLBOARD	BILLBOARD
BILLBOARD	BILLBOARD
BILLBOARD	BILLBOARD
BILLBOARD	BILLBOARD
BILLBOARD	BILLBOARD
BILLBOARD	BILLBOARD
BILLBOARD	BILLBOARD
BILLBOARD	BILLBOARD
BILLBOARD	BILLBOARD

We start by processing the mesh lists
 We run one task per mesh type





MESH LIST OF ONE MESH TYPE



Per-thread mesh lists are combined to a single list of meshes



@IntelSoftware

@IntelGraphics

@totalwar

@CAGames





MESH LIST OF ONE MESH TYPE



Collated meshes



Per-thread mesh lists are combined to a single list of meshes

MESH LIST OF ONE MESH TYPE

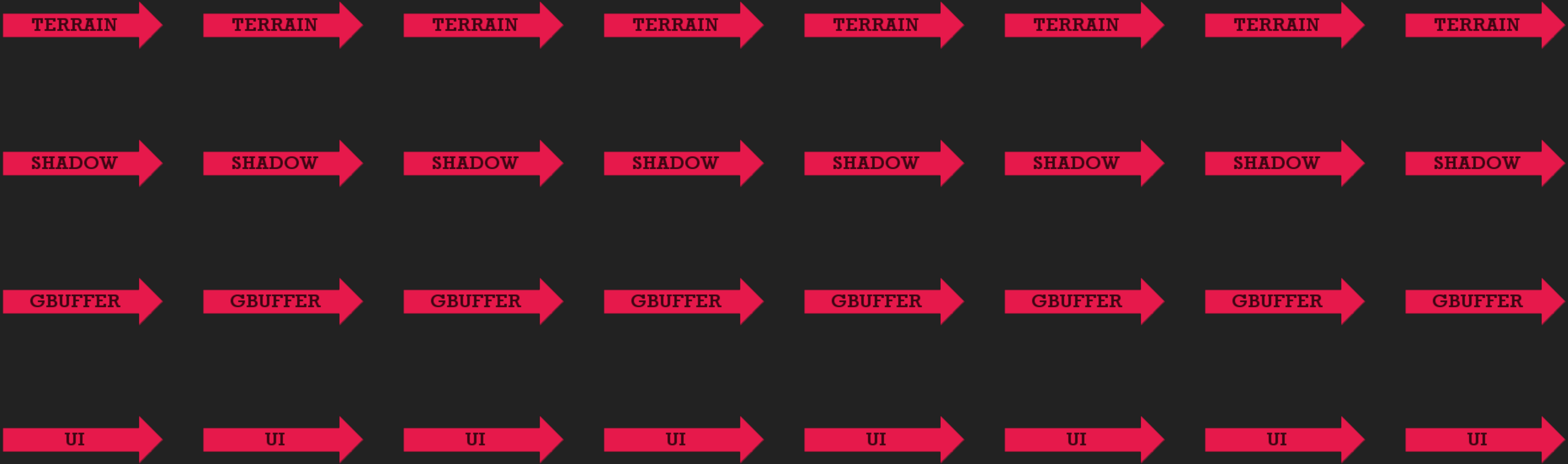
Collated meshes



Then we process all the meshes in the list one by one



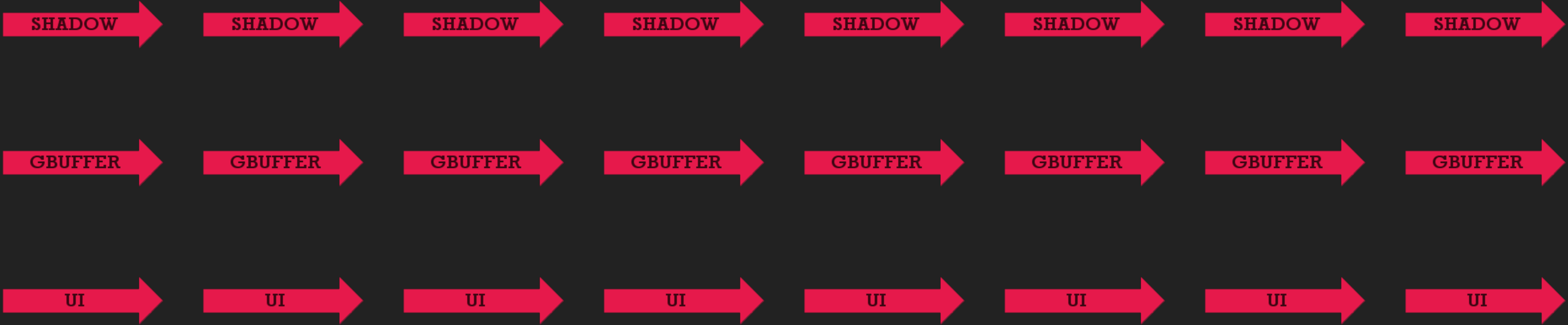
INSTANCE LISTS



Each mesh has instance lists per pipeline stage

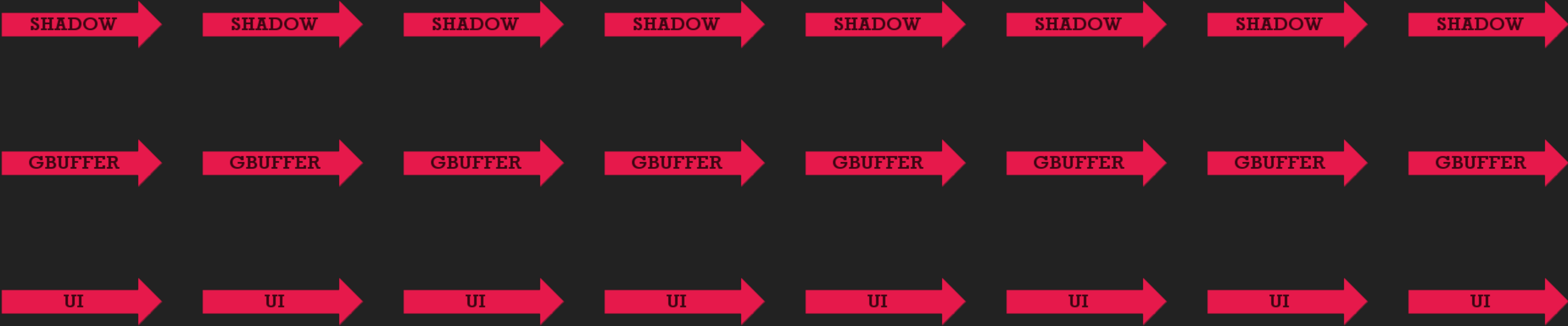
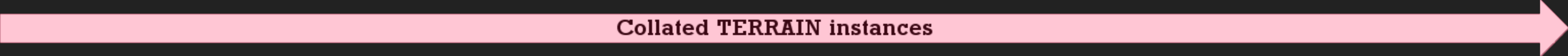


INSTANCE LISTS



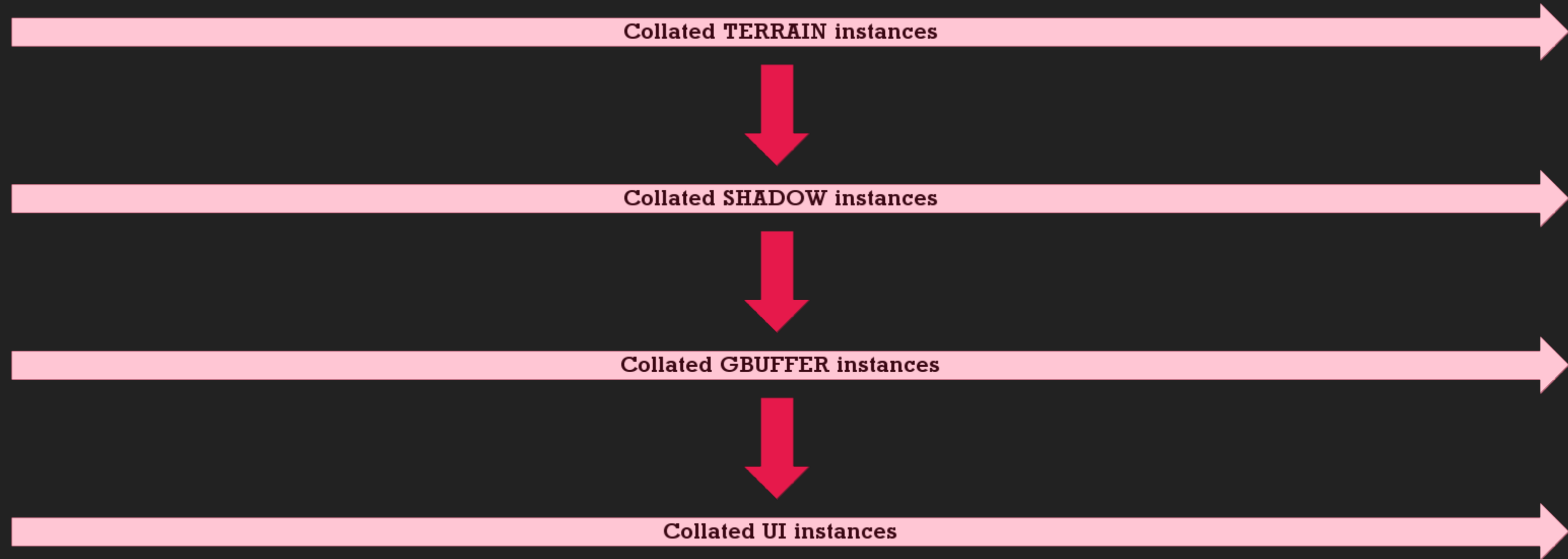
We start with the first pipeline stage and combine the instances there into a single list

INSTANCE LISTS



Then we process the instances

INSTANCE LISTS



Then process all subsequent stages sequentially



@IntelSoftware

@IntelGraphics

@totalwar

@CAGames



INSTANCE LIST PROCESSING



Collated TERRAIN instances



@IntelSoftware

@IntelGraphics

@totalwar

@CAGames



INSTANCE LIST PROCESSING



Collated TERRAIN instances

Every instance has a 64 bit sort key

Includes bits from distance from camera

Pieces of 'instance' data map to bits in the sort key

Material flags (blood, burning, dead), LUT index, etc.



@IntelSoftware

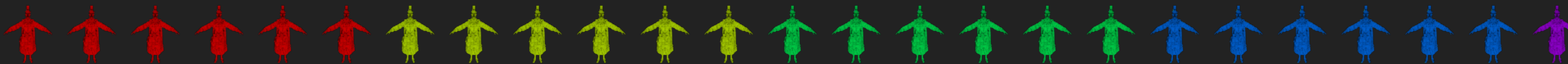
@IntelGraphics

@totalwar

@CAGames



INSTANCE LIST PROCESSING



Collated TERRAIN instances

Every instance has a 64 bit sort key

Includes bits from distance from camera

Pieces of 'instance' data map to bits in the sort key

Material flags (blood, burning, dead), LUT index, etc.

Sorting moves similar instances next to each other

Similar instances can be rendered with a single draw call



@IntelSoftware

@IntelGraphics

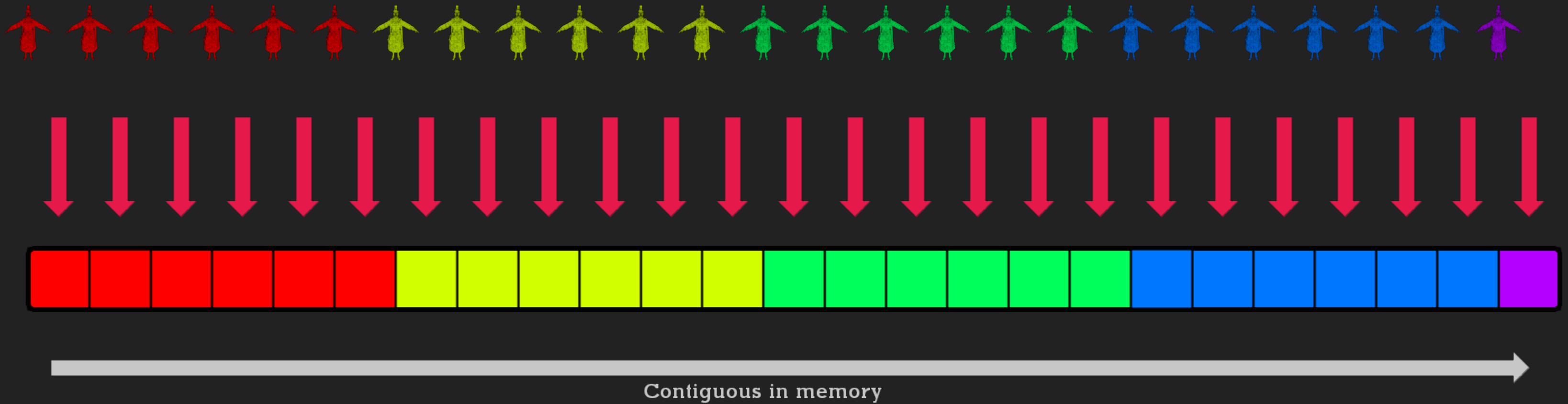
@totalwar

@CAGames



INSTANCE LIST PROCESSING

Collated TERRAIN instances



Instance data is now uploaded to the GPU and prepared for batching



@IntelSoftware

@IntelGraphics

@totalwar

@CAGames





INSTANCE LISTS




Collated TERRAIN instances



Collated SHADOW instances



Collated GBUFFER instances



Collated UI instances



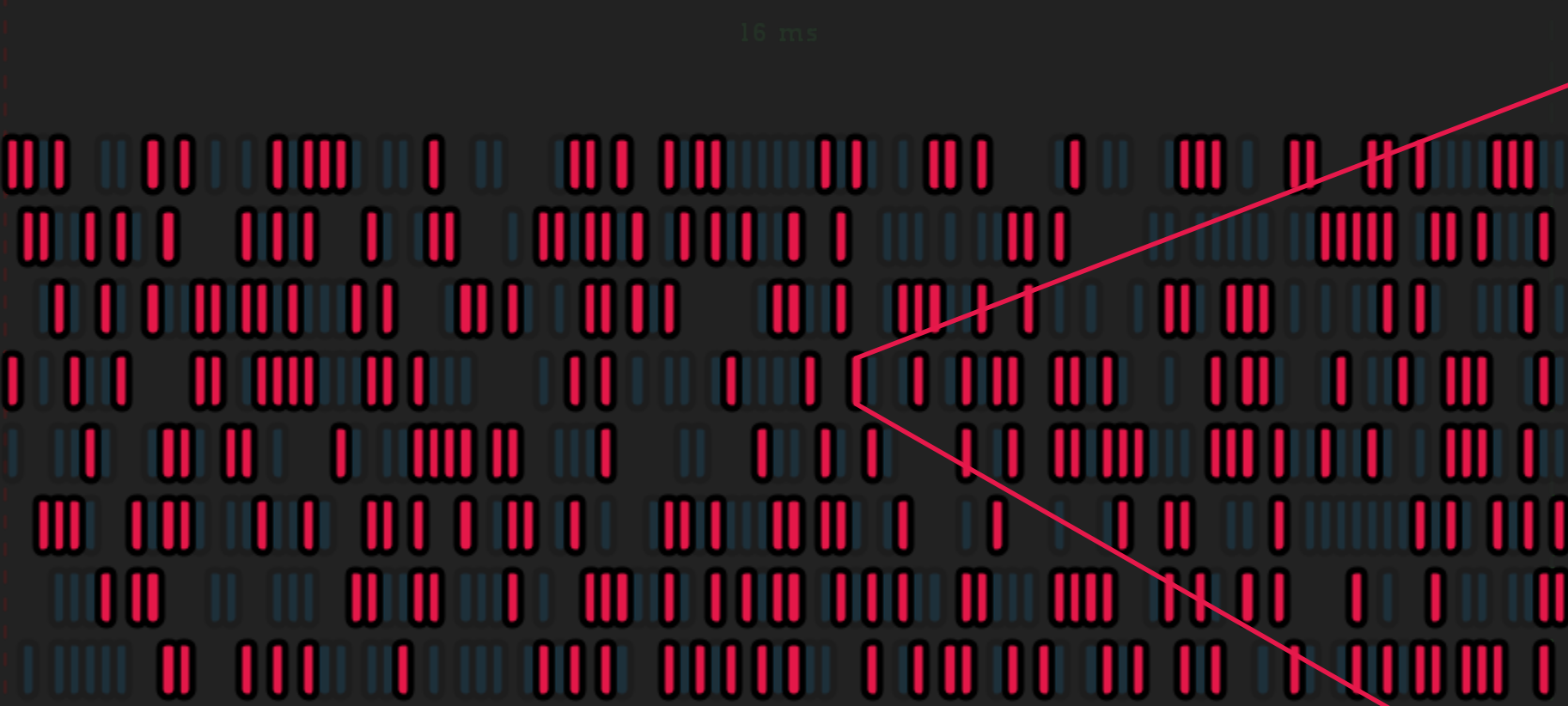
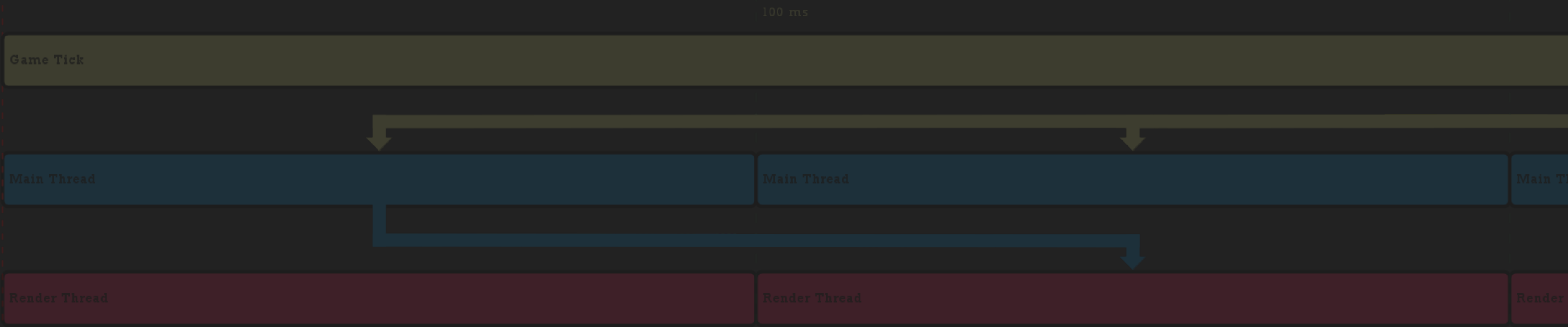
@IntelSoftware

@IntelGraphics

@totalwar

@CAGames



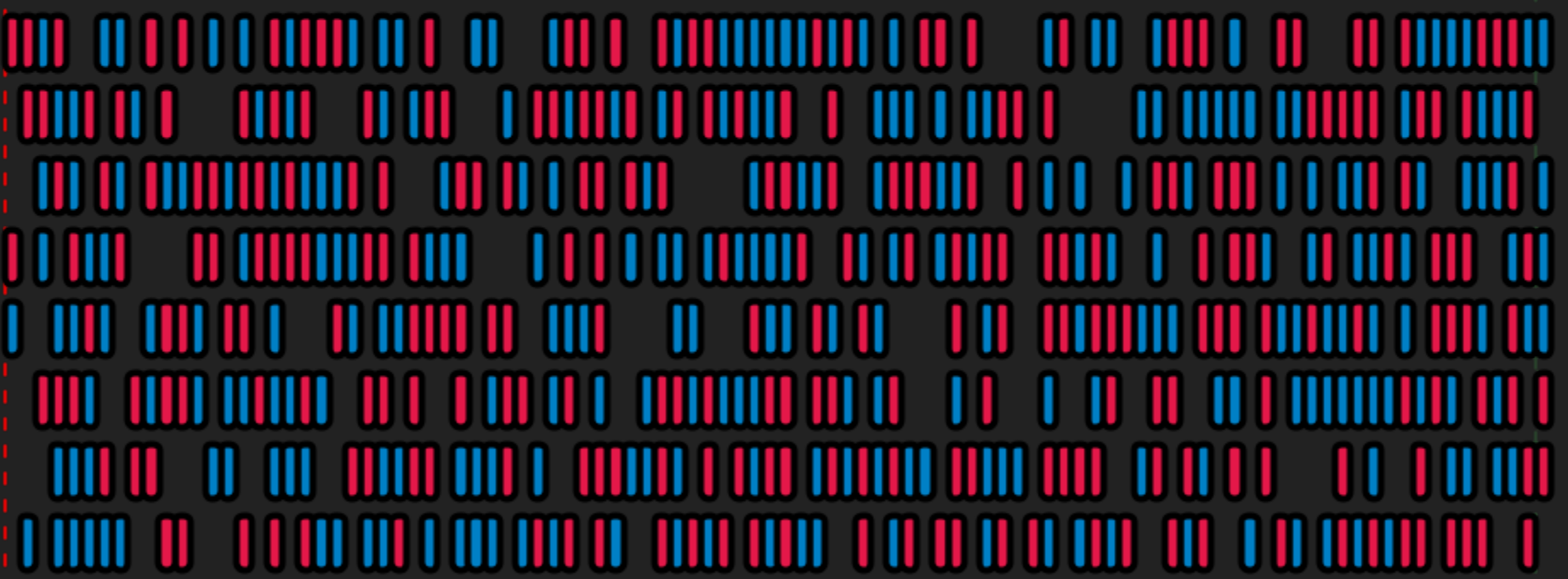


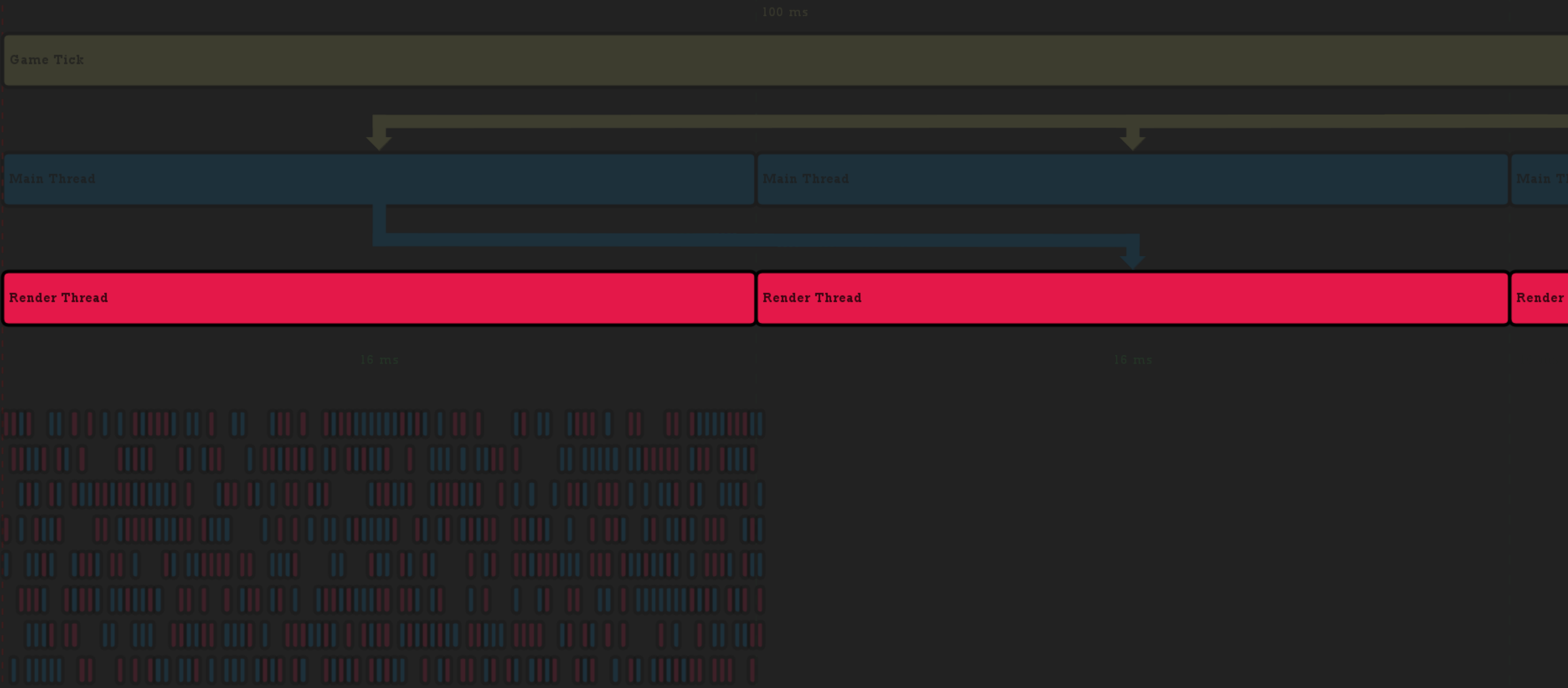
100 ms



16 ms

16 ms





@IntelSoftware

@IntelGraphics

@totalwar

@CAGames



RENDERING

Render Thread



The actual rendering follows the different pipeline stages



@IntelSoftware

@IntelGraphics

@totalwar

@CAGames



RENDERING

Render Thread



Each stage renders a selection of mesh types



@IntelSoftware

@IntelGraphics

@totalwar

@CAGames



RENDERING



Render Thread



The meshes are prepared by the render thread workers



@IntelSoftware

@IntelGraphics

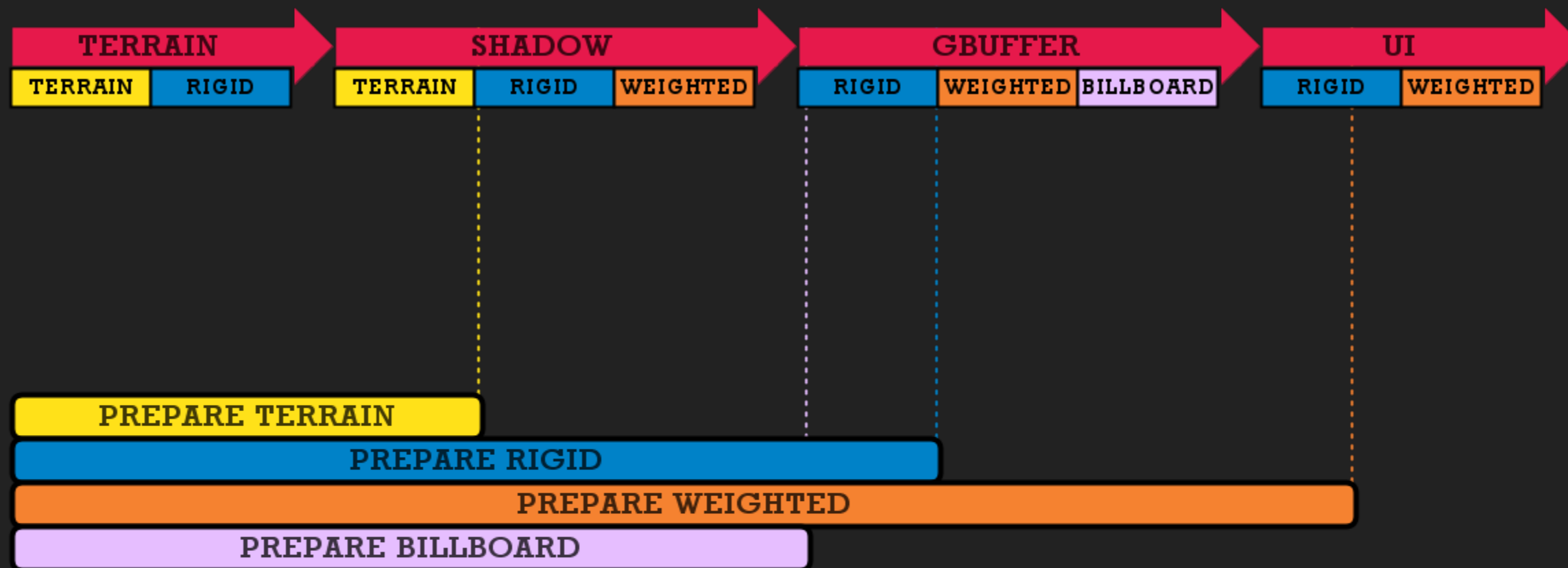
@totalwar

@CAGames



RENDERING

Render Thread



We have to start by waiting for the preparation tasks to be finished



@IntelSoftware

@IntelGraphics

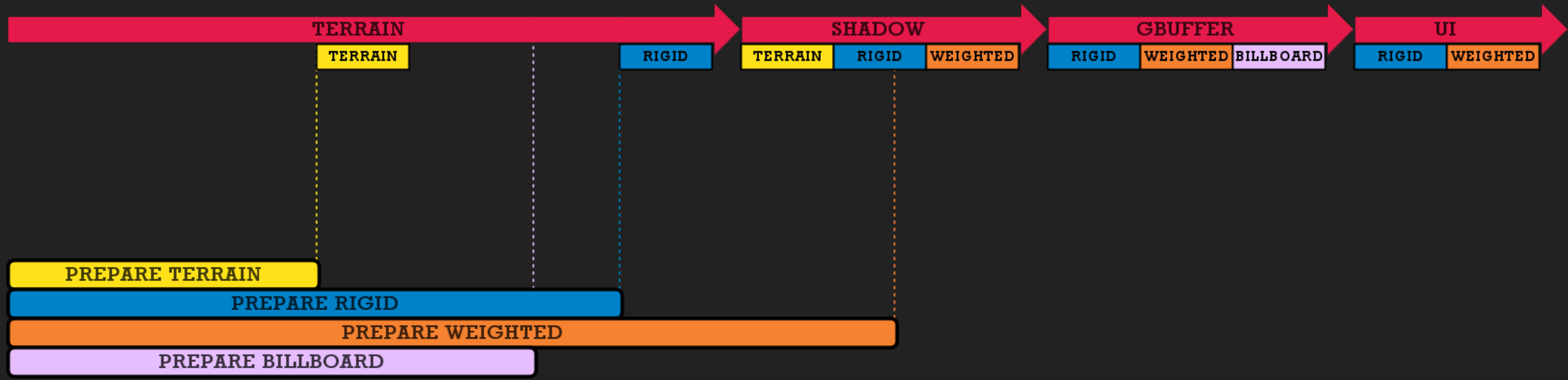
@totalwar

@CAGames



RENDERING

Render Thread



We have to start by waiting for the preparation tasks to be finished



@IntelSoftware

@IntelGraphics

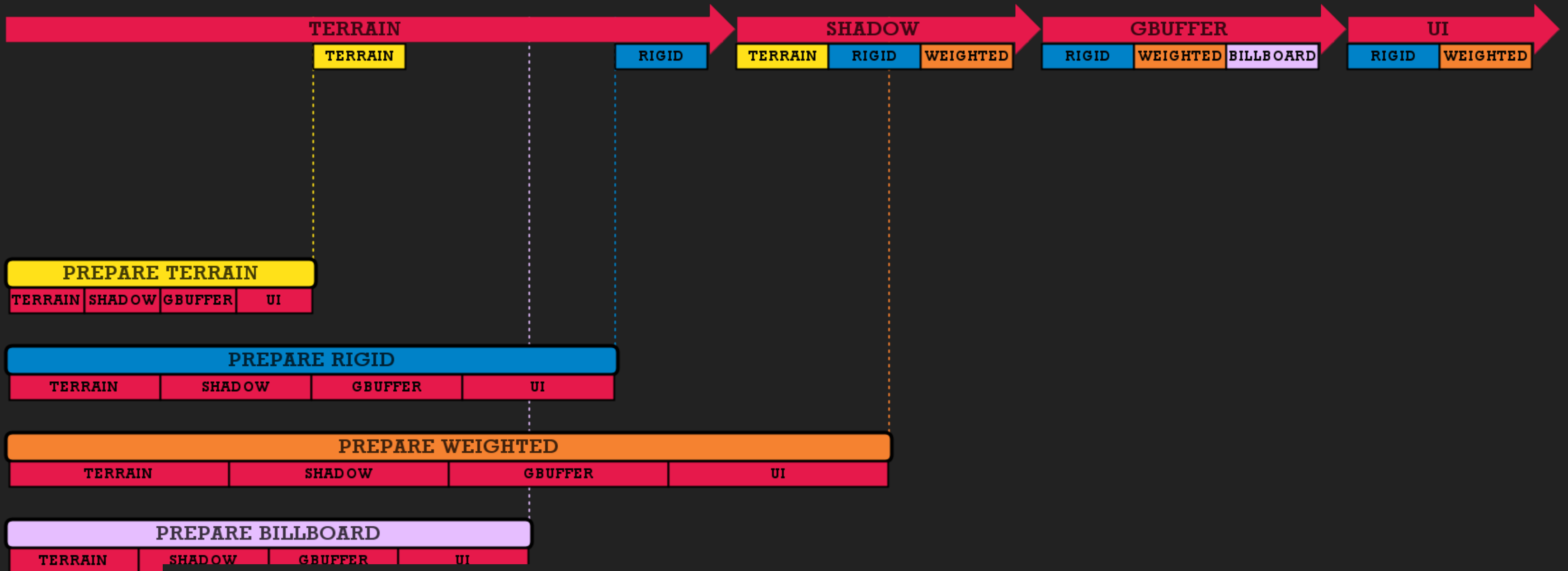
@totalwar

@CAGames



RENDERING

Render Thread



The tasks process all meshes/instances for all pipeline stages
We can use an atomic counter per pipeline stage



@IntelSoftware

@IntelGraphics

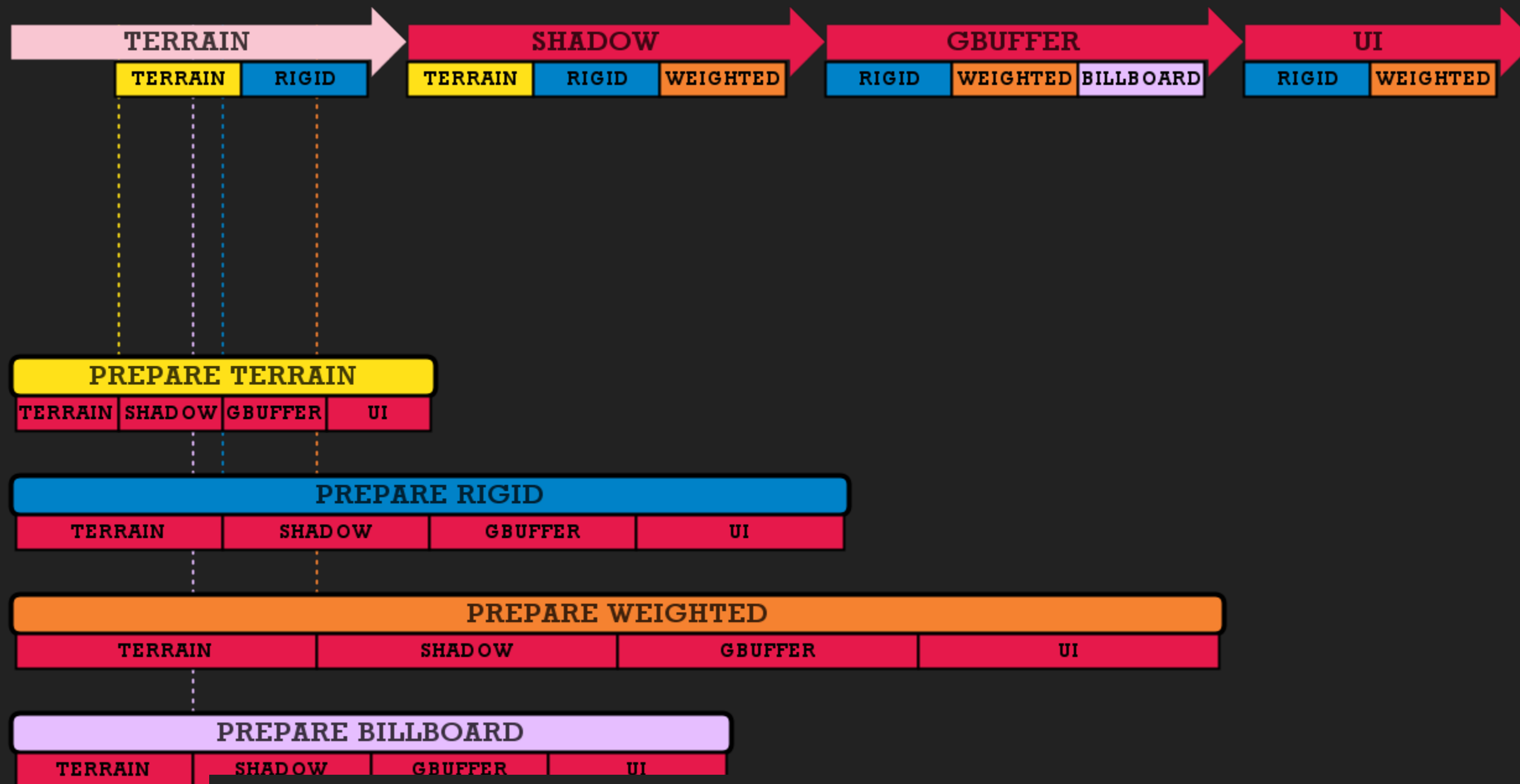
@totalwar

@CAGames



RENDERING

Render Thread



No need to further split the tasks

We just increased the granularity of waiting for other sub-tasks



@IntelSoftware

@IntelGraphics

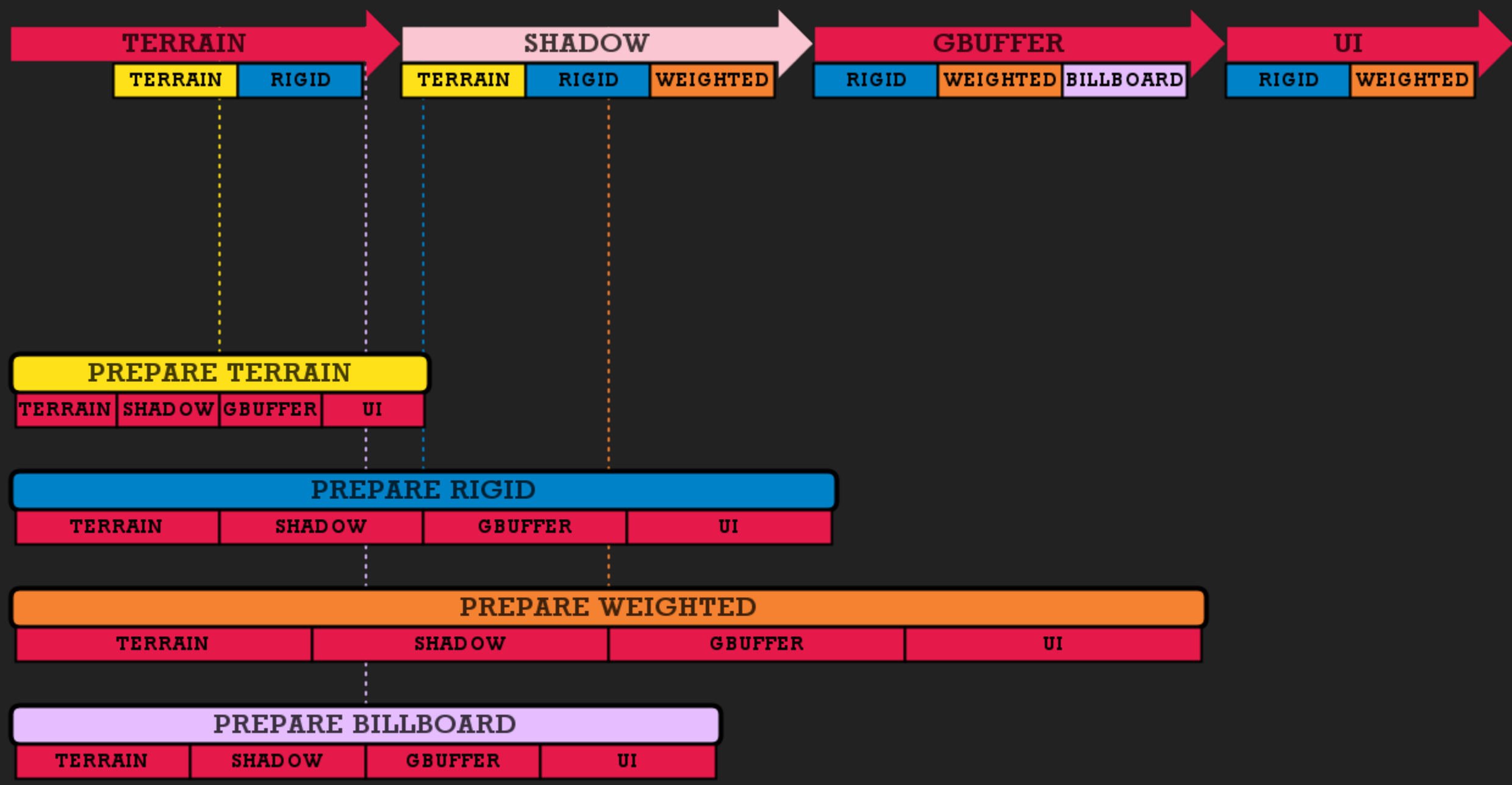
@totalwar

@CAGames



RENDERING

Render Thread



Usually everything is ready after the short initial wait



@IntelSoftware

@IntelGraphics

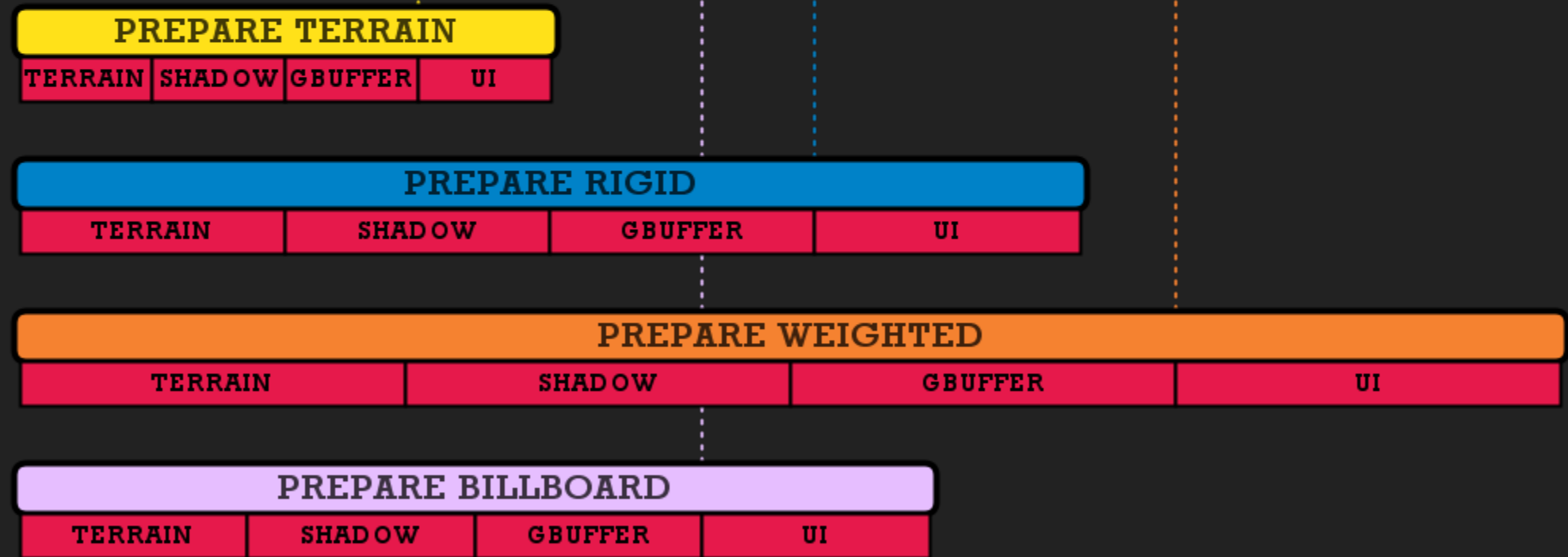
@totalwar

@CAGames



RENDERING

Render Thread

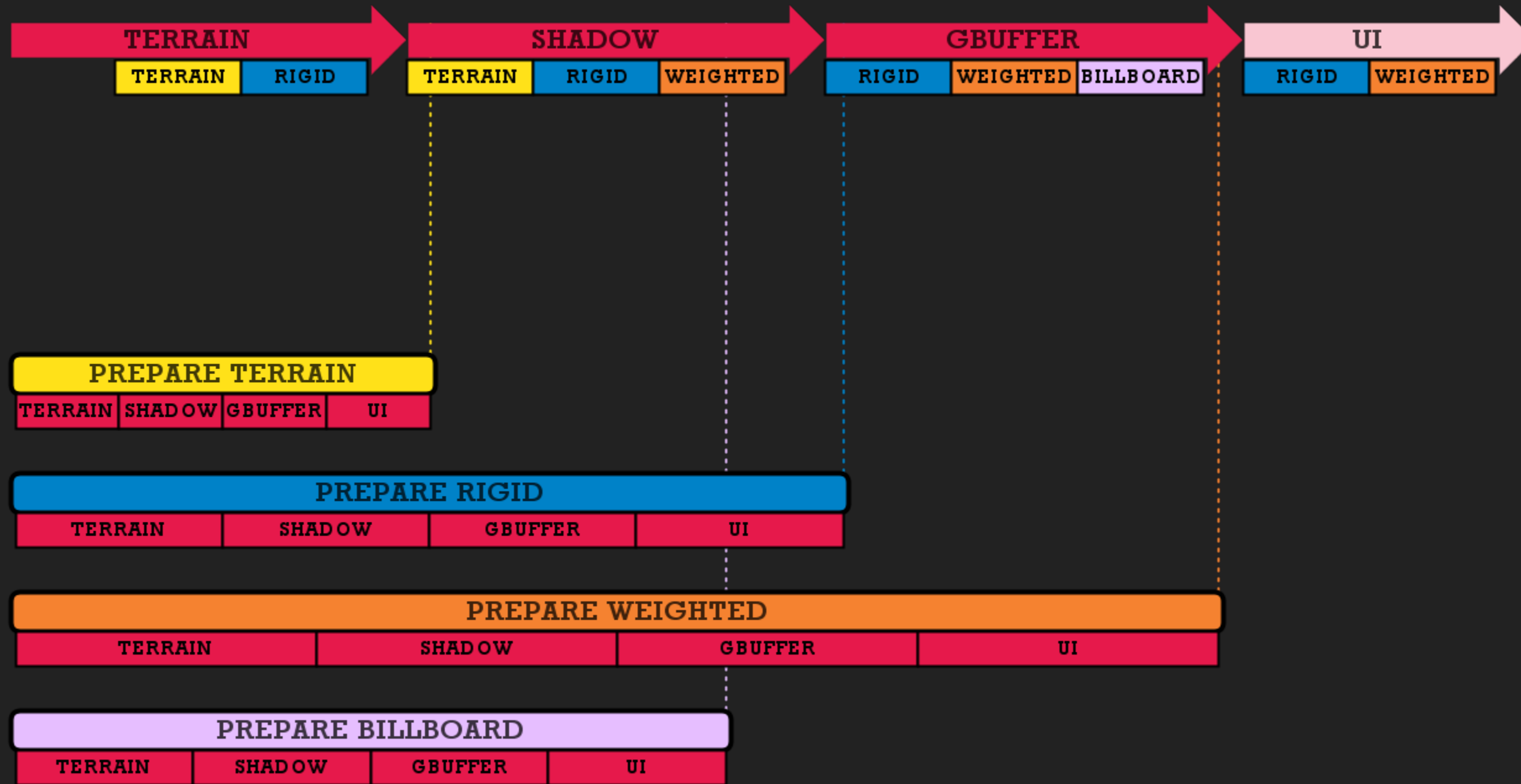


Usually everything is ready after the short initial wait



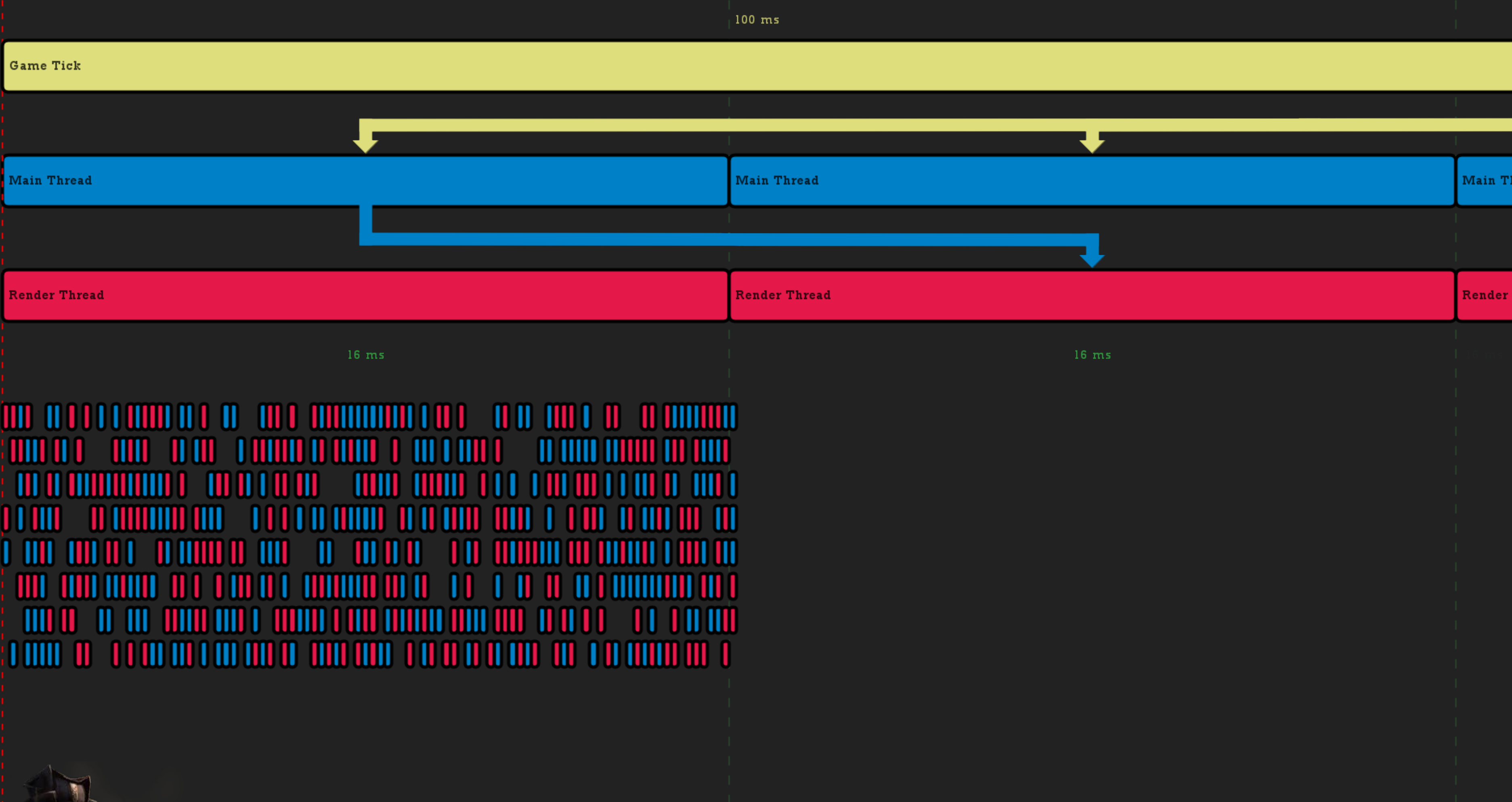
RENDERING

Render Thread



Usually everything is ready after the short initial wait





@IntelSoftware

@IntelGraphics

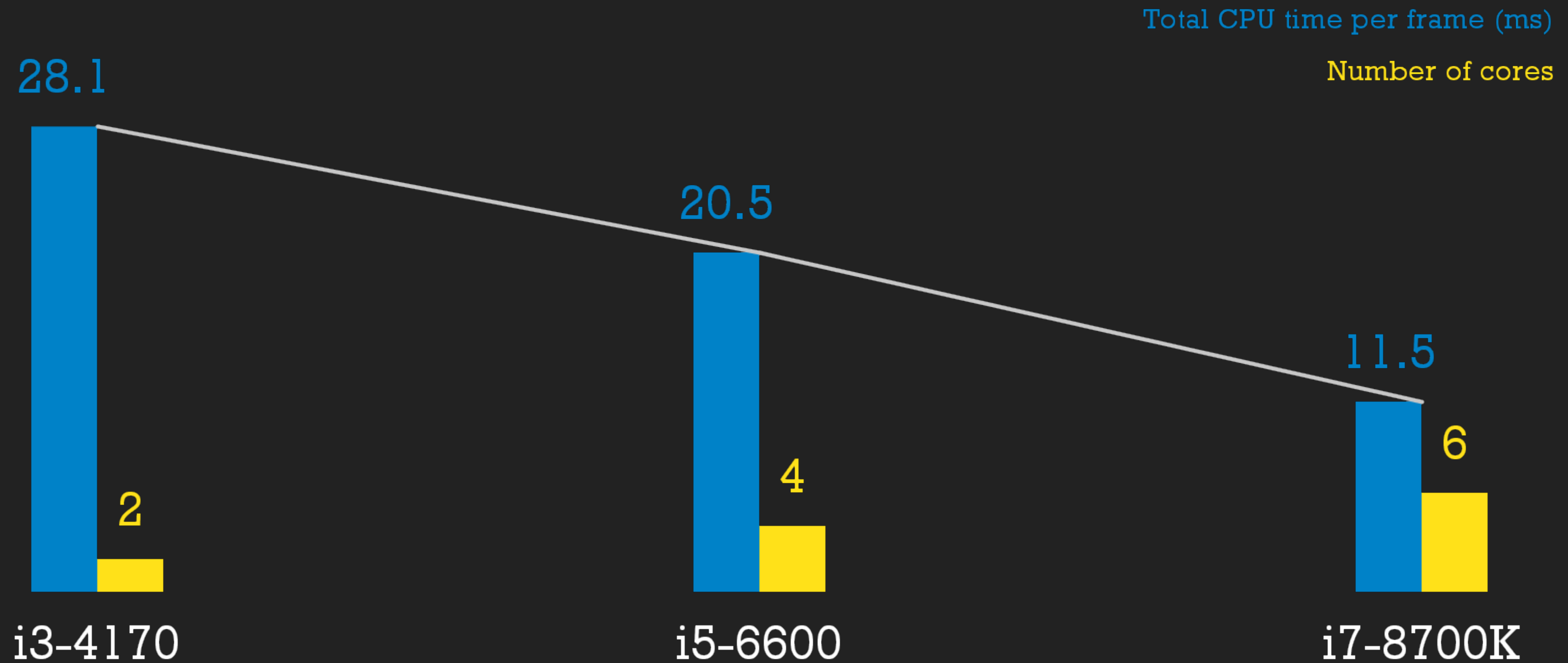
@totalwar

@CAGames



RESULTS

70-80% scaling



@IntelSoftware

@IntelGraphics

@totalwar

@CAGames



INSIGHTS

Per-thread lists are still our go-to way doing batch processing

Try splitting batch processing tasks to $2 * \text{\#cores}$ parts

Memory layout can highly influence performance

You can't do better than your longest serial part

You need to plan your systems with multithreading in mind from the start

More cores, stronger CPU can mean more detail and better visuals



@IntelSoftware

@IntelGraphics

@totalwar

@CAGames



INTEGRATED CARDS

Main test machine was Skull Canyon

Almost every change we made was to relieve pressure on memory

A quick early out in our DoF pass resulted in a 3.6ms uplift

Shader register usage can also cause problems

Reducing the number of raymarching steps in fog shader was a 3ms uplift

Copying resources can also take up significant time

Highest individual copy was 0.9ms



@IntelSoftware

@IntelGraphics

@totalwar

@CAGames



ORDER INDEPENDENT TRANSPARENCY FOR VFX



@IntelSoftware

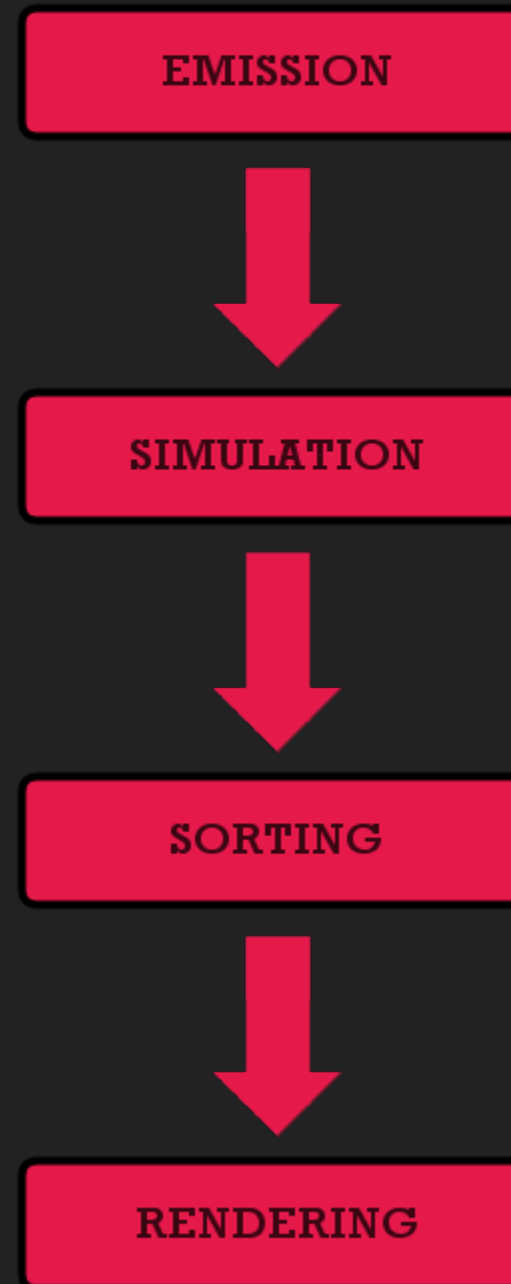
@IntelGraphics

@totalwar

@CAGames



ORDER INDEPENDENT TRANSPARENCY FOR VFX



@IntelSoftware

@IntelGraphics

@totalwar

@CAGames



ORDER INDEPENDENT TRANSPARENCY FOR VFX

Total War: ROME 2 (2013)

CPU

GPU

EMISSION

SIMULATION

SORTING

RENDERING



@IntelSoftware

@IntelGraphics

@totalwar

@CAGames



ORDER INDEPENDENT TRANSPARENCY FOR VFX

Total War: ATTILA (2015)

CPU

GPU (PS)

EMISSION

SIMULATION

SORTING

RENDERING

ORDER INDEPENDENT TRANSPARENCY FOR VFX

Total War: WARHAMMER (2016)

CPU

GPU (CS)

EMISSION

SIMULATION

SORTING

RENDERING



@IntelSoftware

@IntelGraphics

@totalwar

@CAGames



SORTING



@IntelSoftware

@IntelGraphics

@totalwar

@CAGames



SORTING



The basic building blocks of the system are emitters



@IntelSoftware

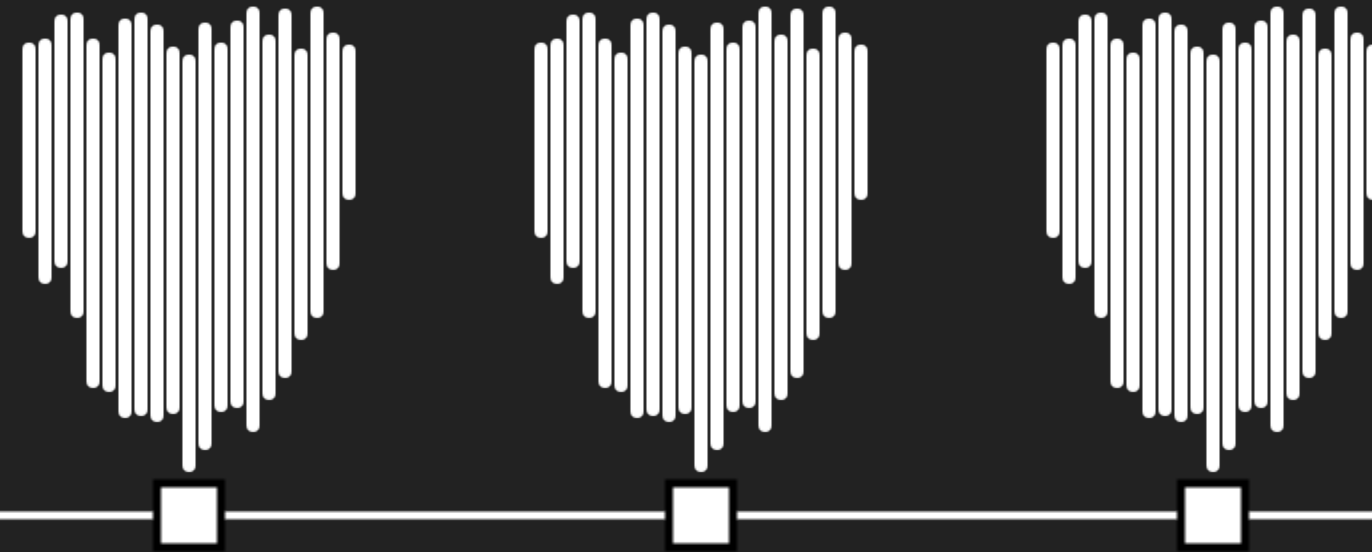
@IntelGraphics

@totalwar

@CAGames



SORTING



Emitters are emitting a number of particles over time



@IntelSoftware

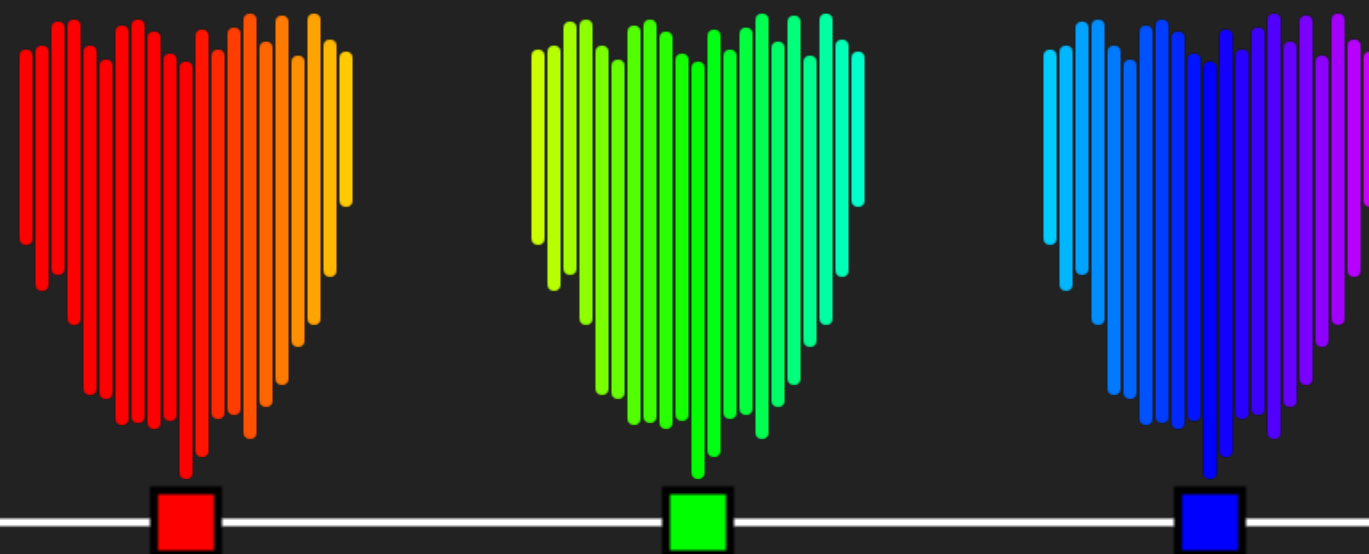
@IntelGraphics

@totalwar

@CAGames



SORTING



Particles are often just sorted per emitter



@IntelSoftware

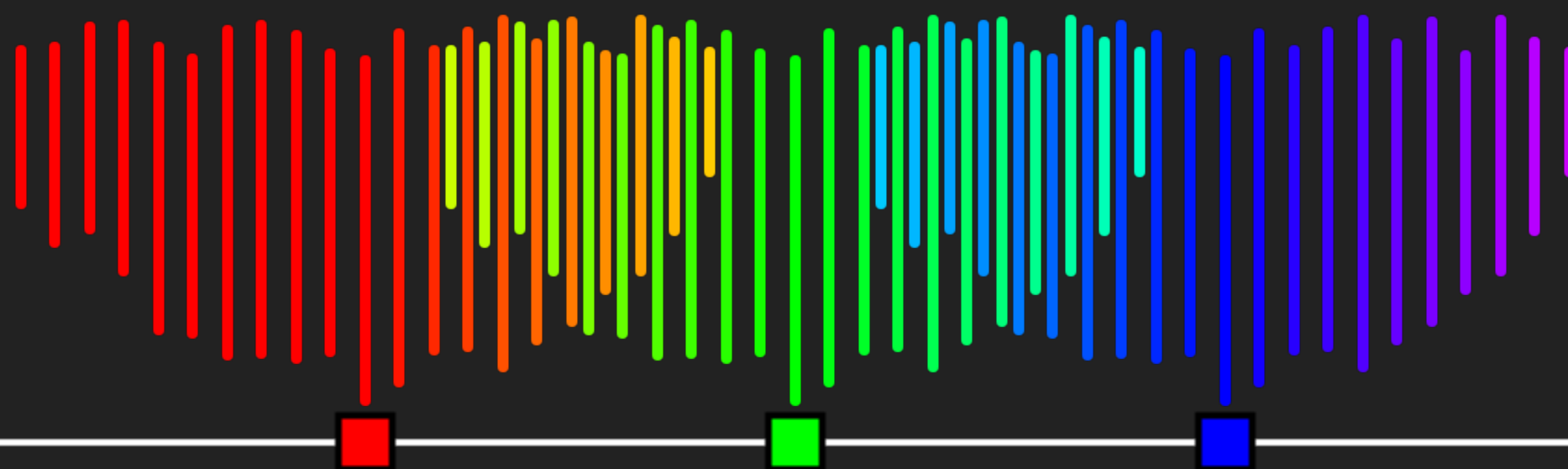
@IntelGraphics

@totalwar

@CAGames



SORTING



Problems start happening when the emitters overlap



@IntelSoftware

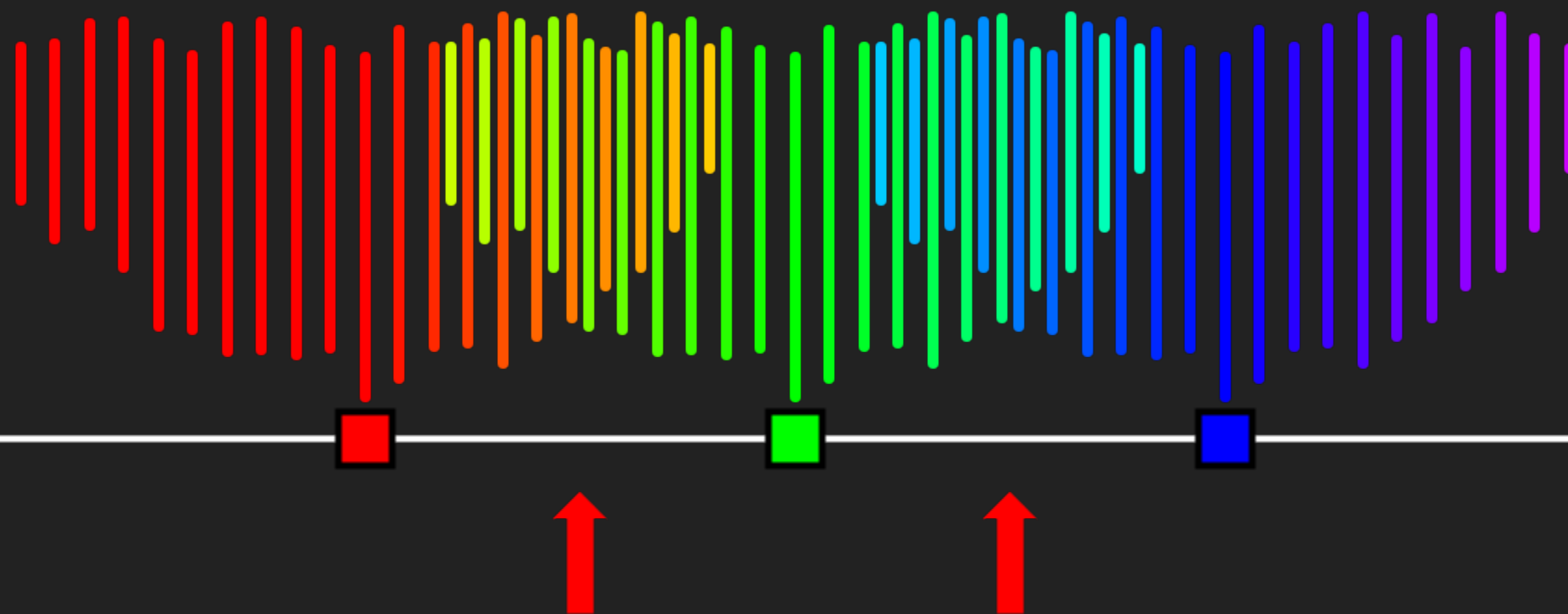
@IntelGraphics

@totalwar

@CAGames



SORTING



Problems start happening when the emitters overlap



@IntelSoftware

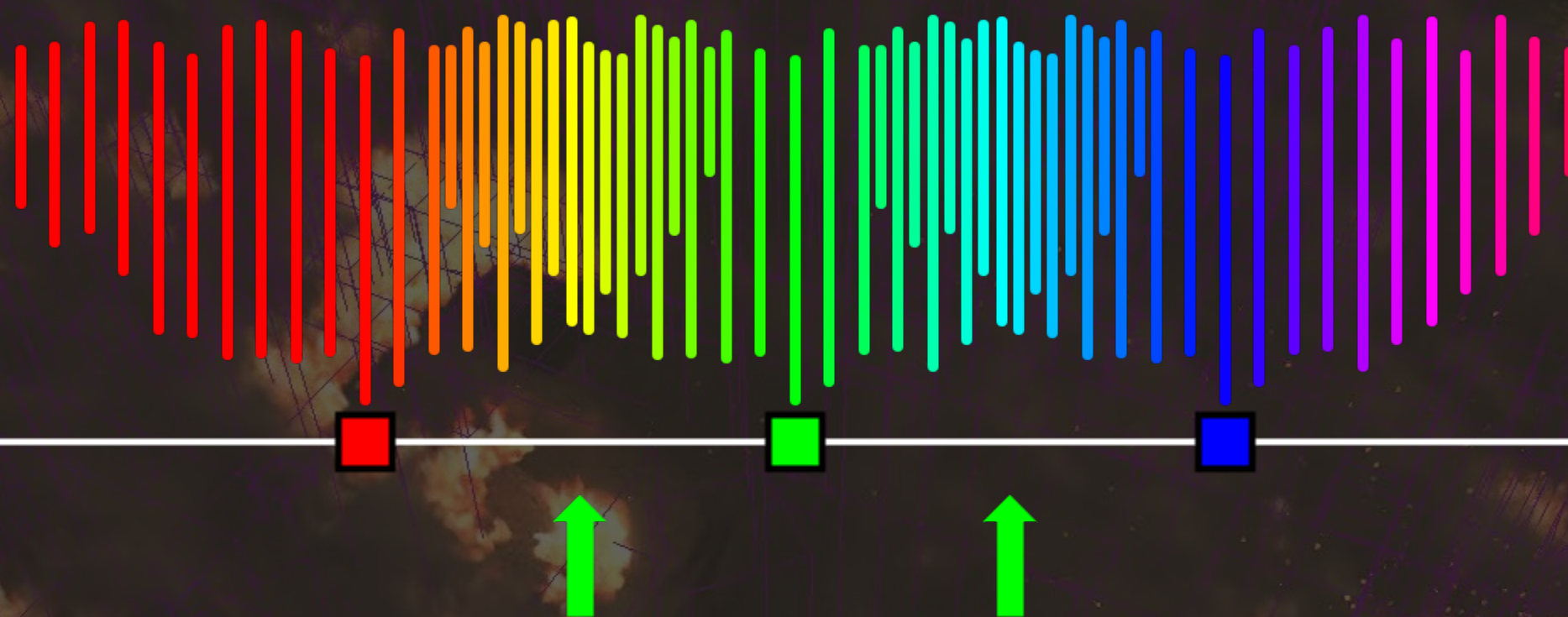
@IntelGraphics

@totalwar

@CAGames



SORTING



No artifacts when emitters are overlapping

Have to use übershader

Sorting time increases exponentially with number of particles

We sort all the particles together



@IntelSoftware

@IntelGraphics

@totalwar

@CAGames



EMISSION



@IntelSoftware

@IntelGraphics

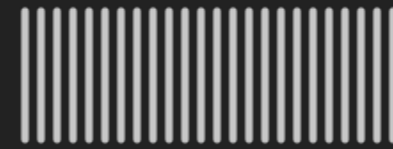
@totalwar

@CAGames



EMISSION

NEW PARTICLES (CPU)



Emission takes place on the CPU



@IntelSoftware

@IntelGraphics

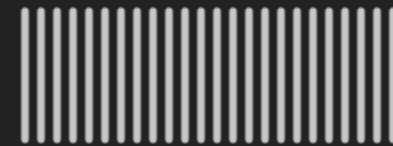
@totalwar

@CAGames

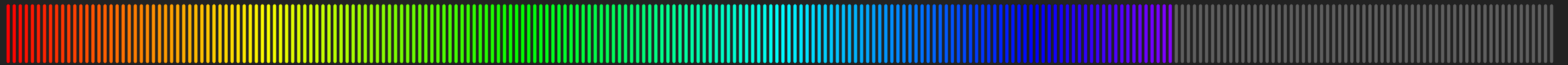
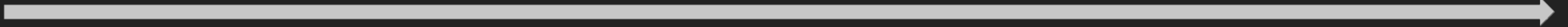


EMISSION

NEW PARTICLES (CPU)



DISTANCE FROM CAMERA



ALIVE PARTICLES

DEAD PARTICLES

PARTICLE BUFFER (GPU)

Sorting is responsible for moving dead particles to one end of the GPU buffer



@IntelSoftware

@IntelGraphics

@totalwar

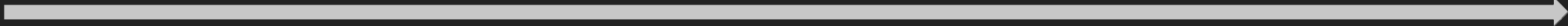
@CAGames



EMISSION

NEW PARTICLES (CPU)

DISTANCE FROM CAMERA



ALIVE PARTICLES

DEAD PARTICLES

PARTICLE BUFFER (GPU)

This make uploading new particles trivial



@IntelSoftware

@IntelGraphics

@totalwar

@CAGames

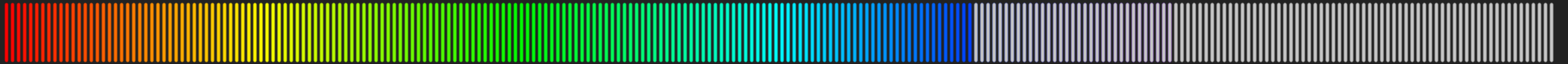


EMISSION

NEW PARTICLES (CPU)

If overrun, new particles stomp over the farthest live particles

DISTANCE FROM CAMERA



ALIVE PARTICLES

DEAD PARTICLES

PARTICLE BUFFER (GPU)

Running out of particle space stomps over particles farthest from camera



@IntelSoftware

@IntelGraphics

@totalwar

@CAGames



GOALS

(for Total War: THREE KINGDOMS)

Freely mix half and full resolution particles

Get rid of (some of) the sorting cost

Get rid of übershader and unlock artist driven shaders for particles

We were confident that order-independent transparency is the solution



@IntelSoftware

@IntelGraphics

@totalwar

@CAGames



PHENOMENOLOGICAL TRANSPARENCY (WBOIT) [McGuire17]

Lightweight

Already used in production
(Rendering Technology in 'Agents of Mayhem' [Kircher18])

Doesn't handle additive blending

Lots of magic numbers

Doesn't work well with close to opaque particles

First try was WBOIT



@IntelSoftware

@IntelGraphics

@totalwar

@CAGames



MOMENT BASED TRANSPARENCY (MBOIT) [Muenstermann2018]

Supports additive and multiplicative blending

Works well with near opaque particles

Only a few parameters to tweak

More expensive (extra pass and more render targets)

Second try was MBOIT



@IntelSoftware

@IntelGraphics

@totalwar

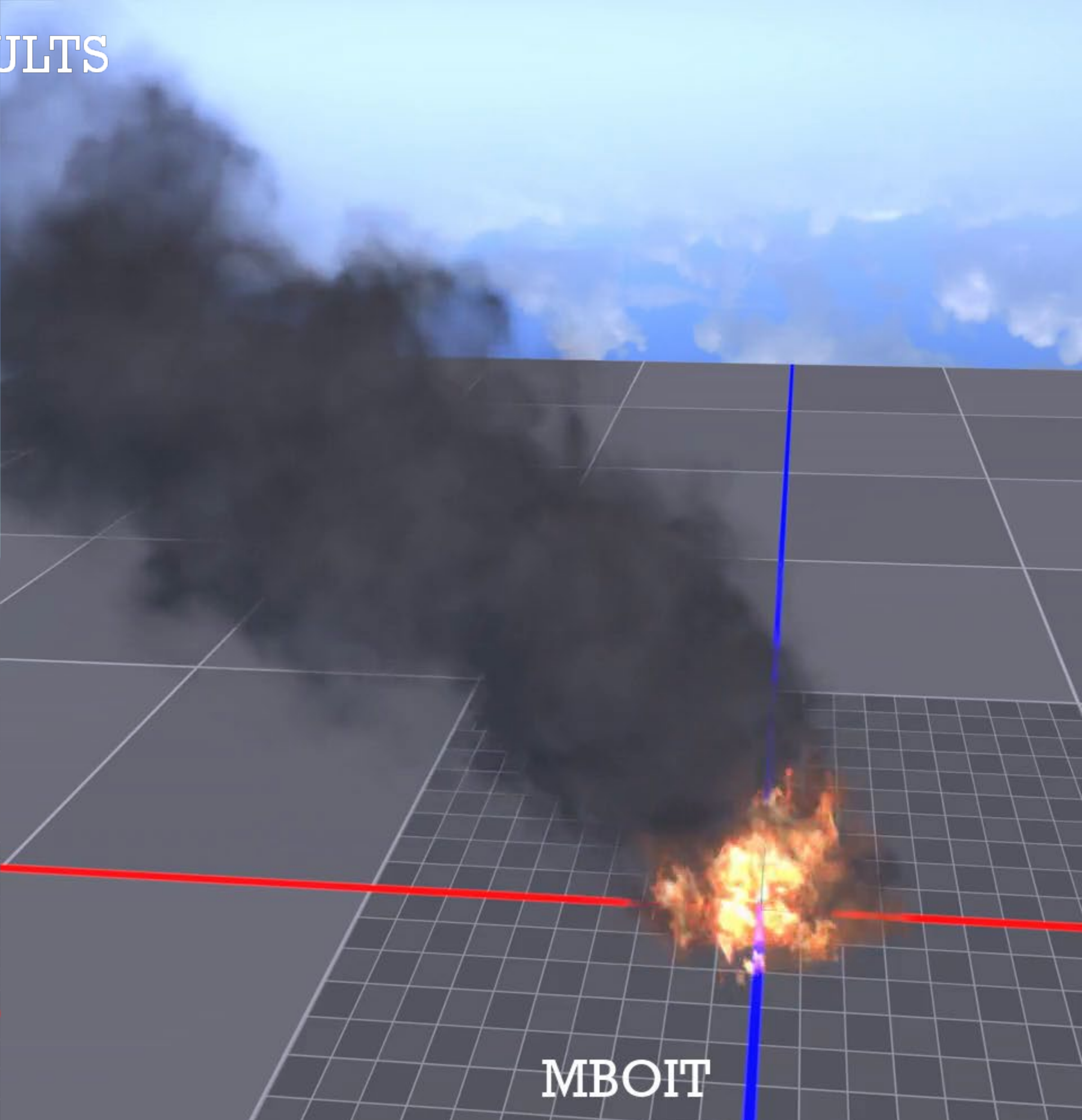
@CAGames



RESULTS



WBOIT



MBOIT



@IntelSoftware

@IntelGraphics

@totalwar

@CAGames



RESULTS

No sorting, only bucketing for dead particles

Same performance as before, less fixed cost

Mix and match shaders

Mix and match resolutions



ALIVE PARTICLES

DEAD PARTICLES

NEW PARTICLES

PARTICLE BUFFER (GPU)

TW:3K was released with MBOIT



@IntelSoftware

@IntelGraphics

@totalwar

@CAGames



THANKS

Wittawat Keawcharoen (Aik) for his OIT work

Steve Hughes for his support from Intel

Rendering Team @ CA

We are hiring!

Intel team



@IntelSoftware

@IntelGraphics

@totalwar

@CAGames





QUESTIONS?



intel®