Killzone Shadow Fall: Threading the Entity Update on PS4

Jorrit Rouwé
Lead Game Tech, Guerrilla Games

GDC 'Eu

GAME DEVELOPERS CONFERENCE™ EUROPE
CONGRESS-CENTRUM OST KOELNMESSE · COLOGNE, GERMANY
AUGUST 11–13, 2014 · EXPO: AUGUST 11–12, 2014

Introduction

- Killzone Shadow Fall is a First Person Shooter
- PlayStation 4 launch title
- In SP up to 60 characters @ 30 FPS
- In MP up to 24 players @ 60 FPS
- Gameplay logic has lots of
  - Branches
  - Virtual functions
  - Cache misses
- Not suitable for PS3 SPU’s but PS4 has 6 x86 cores

What do we cover?

- What is an Entity?
- What we did on PS3
- Multi threading on PS4
- Balancing Entities across frames
- Performance issues found
- Performance results achieved
- Debug tools

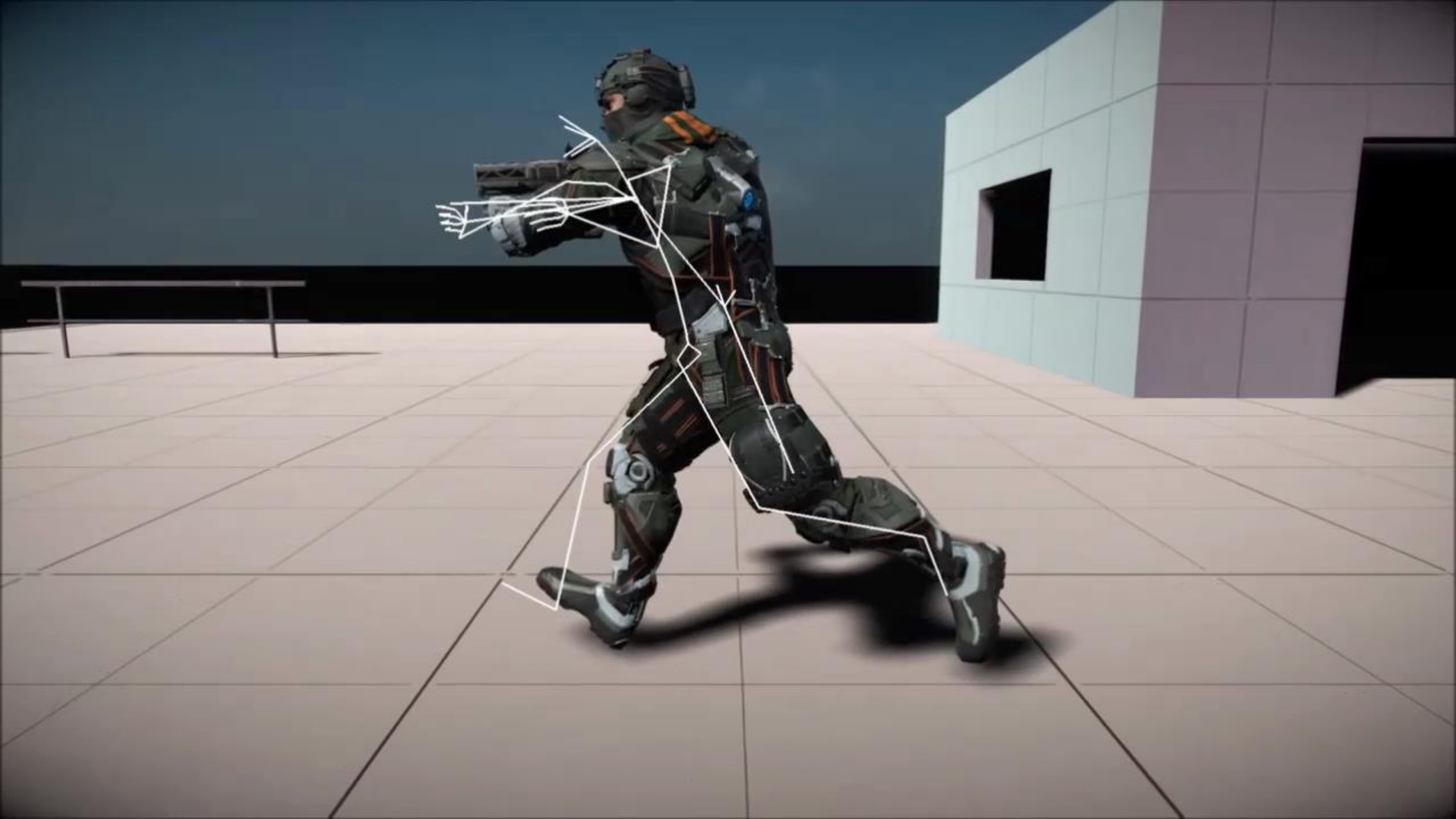
What is an Entity?

What is an Entity?

- Base class for most game objects
  - E.g. Player, Enemy, Weapon, Door
  - Not used for static world
- Has Components
  - E.g. Model, Mover, Destructibility
- Update at a fixed frequency
  - 15, 30, 60 Hz
  - Almost everything at 15 Hz
  - Player updated at higher frequency to avoid lag

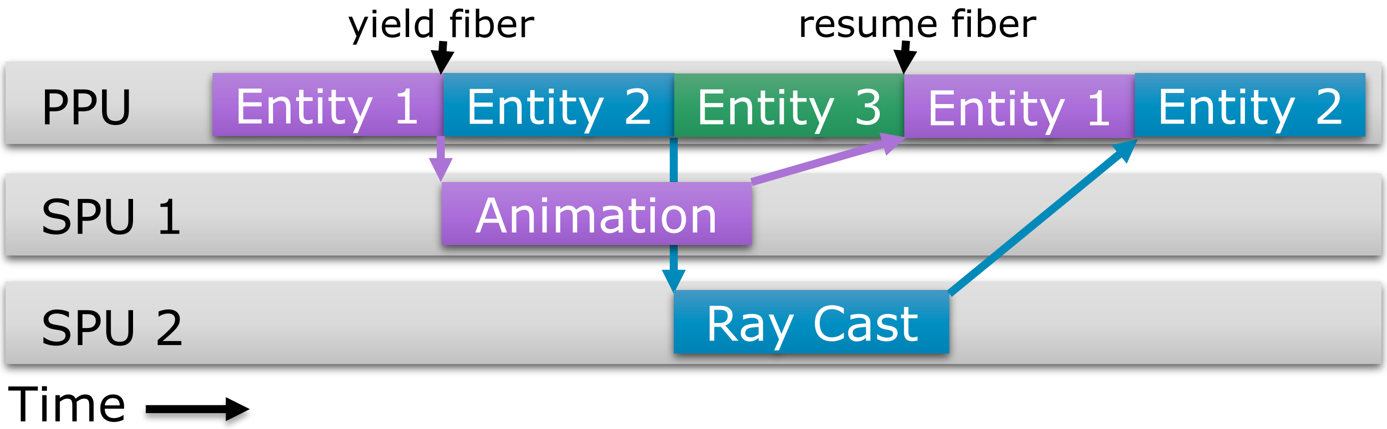
What is a Representation?

- Entities and Components have Representation
- Controls rendering, audio and VFX
- State is interpolated in frames where entity not updated
  - Cheaper to interpolate than to update
- Introduces latency
  - Always blend towards last update



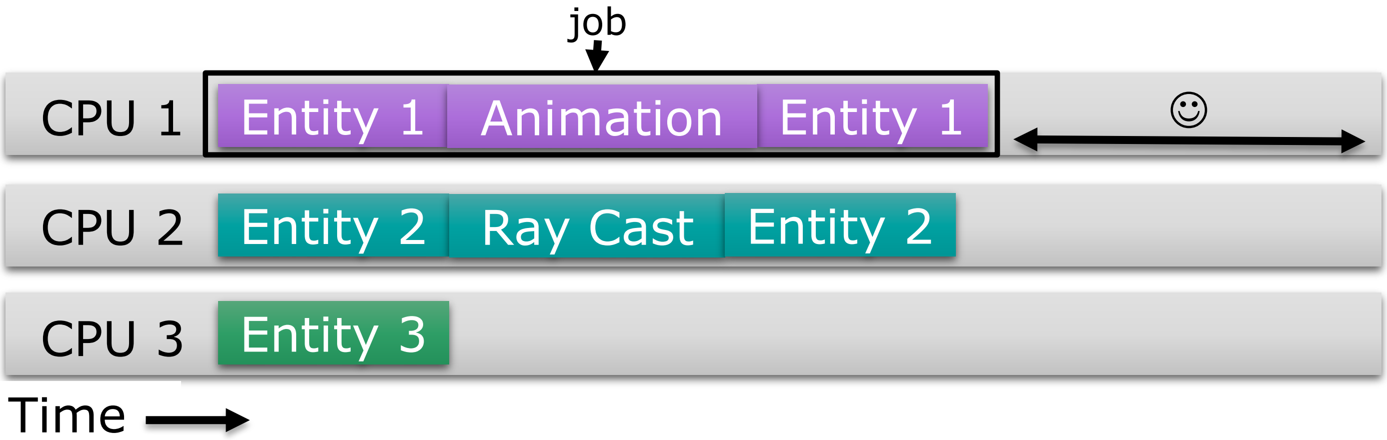
Multi Threading Approach

PS3: 1 Entity = 1 Fiber



- Most time spent on PPU
- No clear concurrency model
  - Read partial updated state
  - Entities deadlock waiting for each other

PS4: 1 Entity = 1 Job



job

CPU 1 Entity 1 Animation Entity 1 ☺

CPU 2 Entity 2 Ray Cast Entity 2

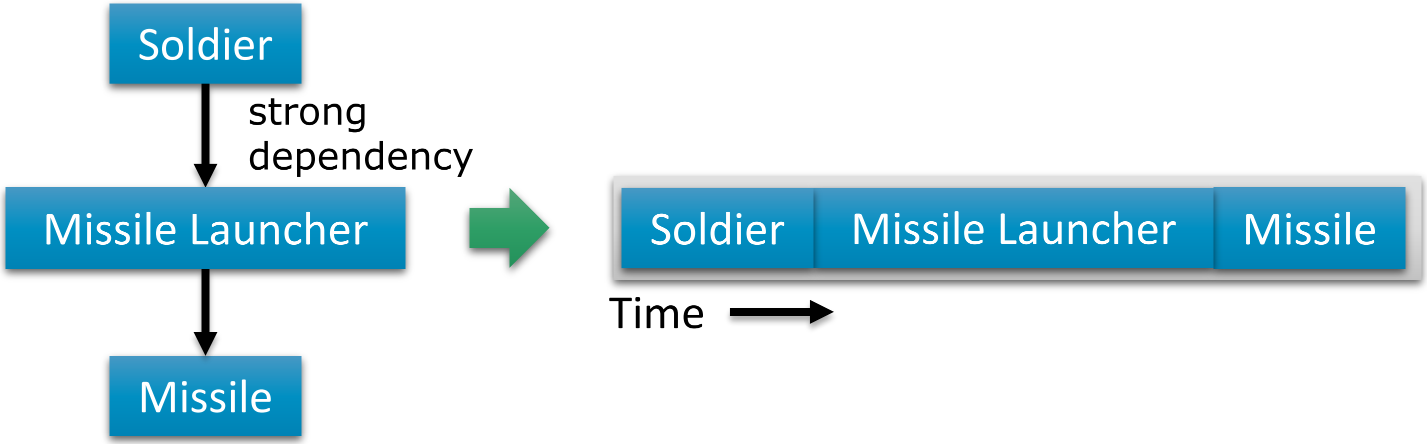
CPU 3 Entity 3

Time

- No fibers
- Entity updates as a whole
- How to solve race conditions?

Strong Dependencies

- Make update order explicit:



Soldier

strong dependency

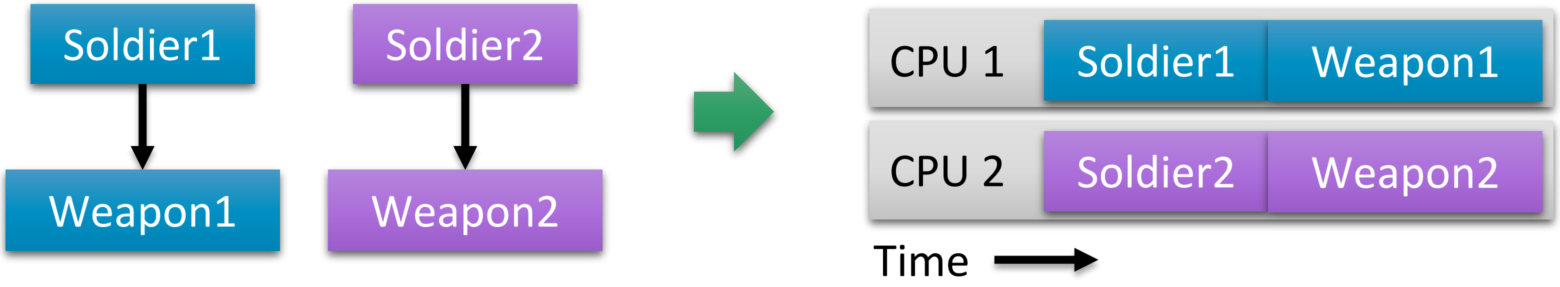
Missile Launcher

Missile

Soldier | Missile Launcher | Missile

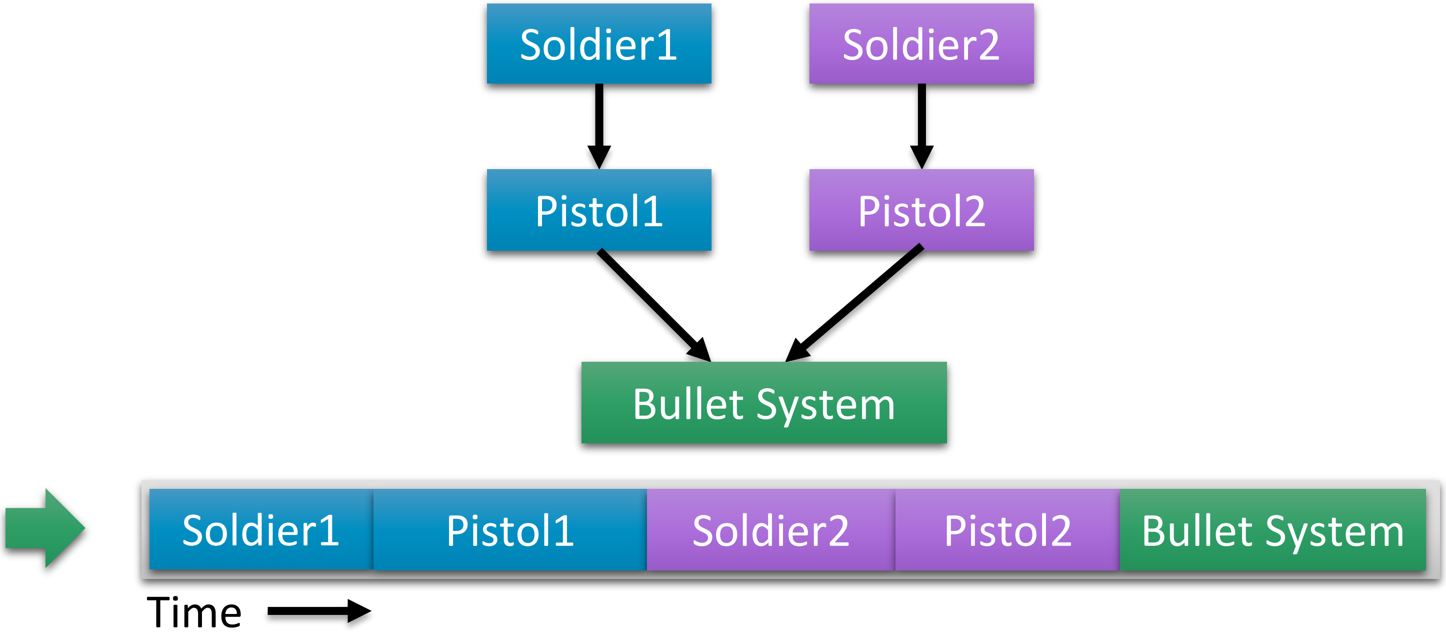
Time

Non-dependent Entities can Execute Concurrently



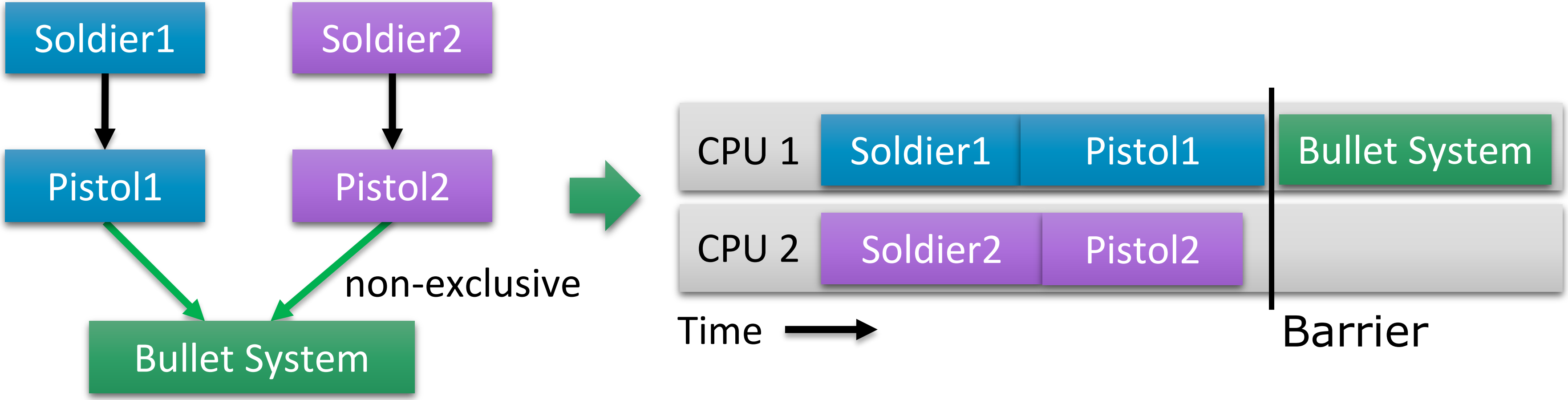
- No (indirect) dependency = no access
- Works two ways: Weapon can access Soldier too
- Create dependency has 1 frame latency
- Global systems need locks

What about this?



- A few entities cause huge bottleneck

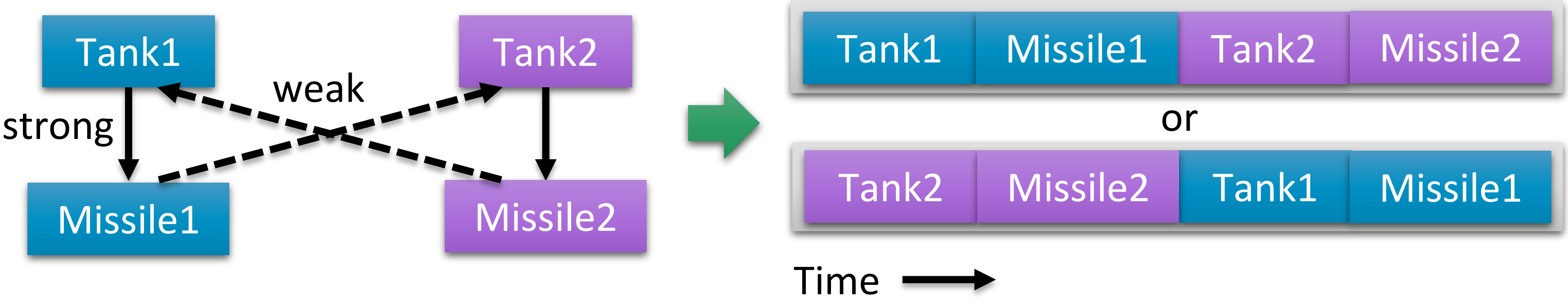
Non-exclusive Dependencies



- Access to 'Bullet System' must be lock protected

Weak Dependencies

- 2 tanks fire at each other:



Tank1

Tank2

weak

strong

Missile1

Missile2

Tank1 Missile1 Tank2 Missile2

or

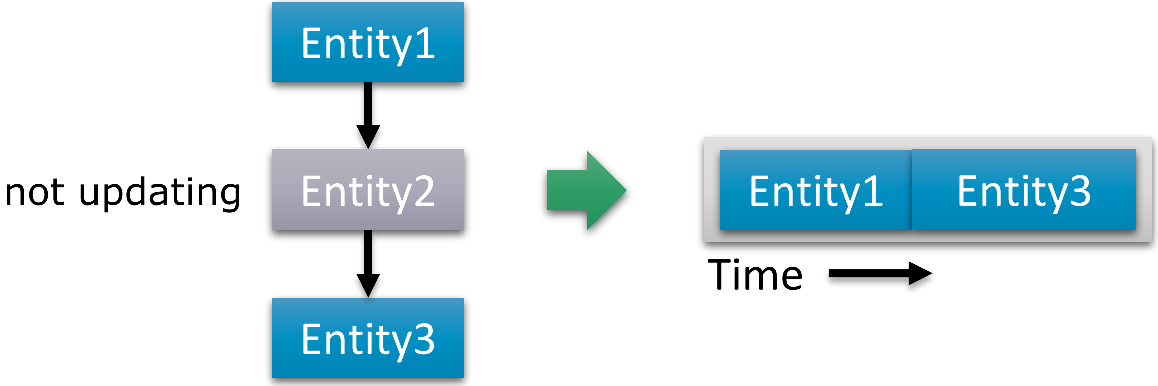
Tank2 Missile2 Tank1 Missile1

Time

- Update order reversed when circular dependency occurs
- Not used very often (< 10 per frame)

Non-updating Entities

- Entity can skip updates (LOD)
- Entity can update in other frame



- Do normal scheduling!

Summarizing Dependencies

	Strong Exclusive	Weak Exclusive	Strong Non-excl.	Weak Non-excl.
Symbol	→	⇢	→	⇢
Two way access	✓	✓	✓	✓
Order guaranteed	✓		✓	
Allow concurrency	+	+	++	++
Require lock			✓	✓

Referencing Entities

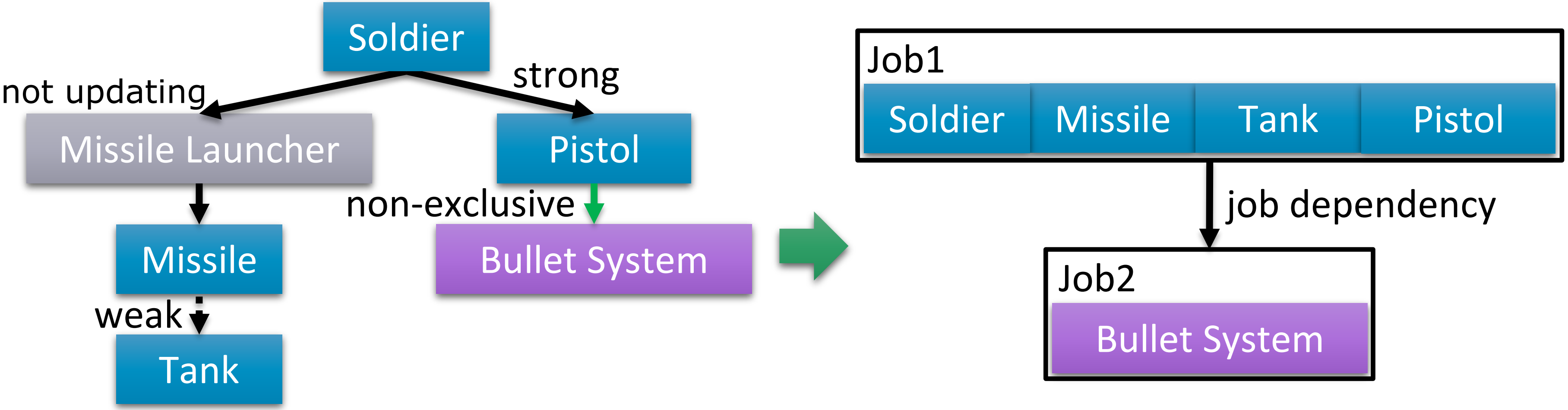
- Dev build: `CheckedPtr<Entity>`
  - Acts as normal pointer
  - Check dependency on dereference
- Retail build: `Entity *`
  - No overhead!
- Doesn’t catch everything
  - Can use pointers to members
  - Bugs were easy to find

Working with Entities without dependency

- `ThreadSafeEntityInterface`
  - Mostly read only
  - Often used state (name, type, position, …)
  - Mutex per Entity
- Send message (expensive)
  - Processed single threaded when no dependency
- Schedule single threaded callback (expensive)
  - Everything can be accessed

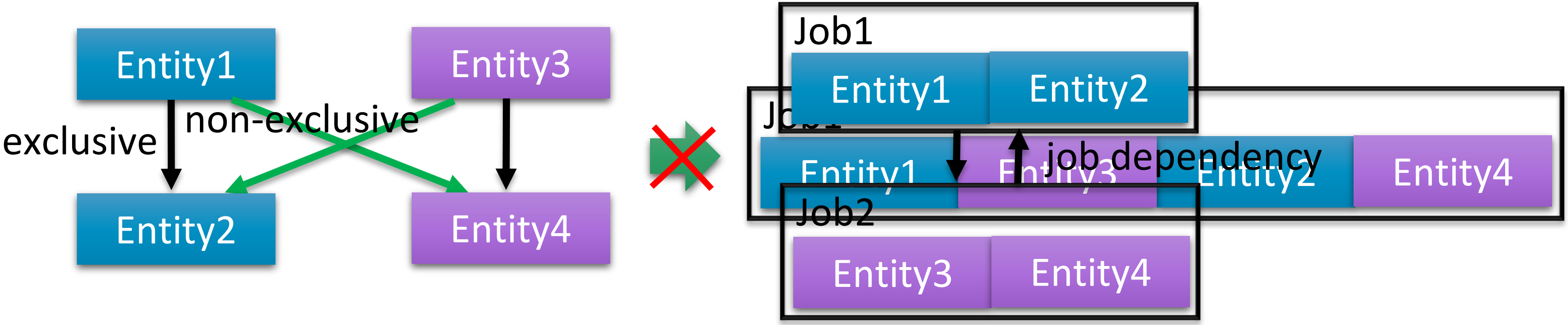
Scheduling Algorithm

Scheduling Algorithm



- Entities with exclusive dependencies merged to 1 job
  - Dependencies determine sorting
- Non-exclusive dependencies become job dependencies
- **Expensive jobs kicked first!**

Scheduling Algorithm – Edge Case



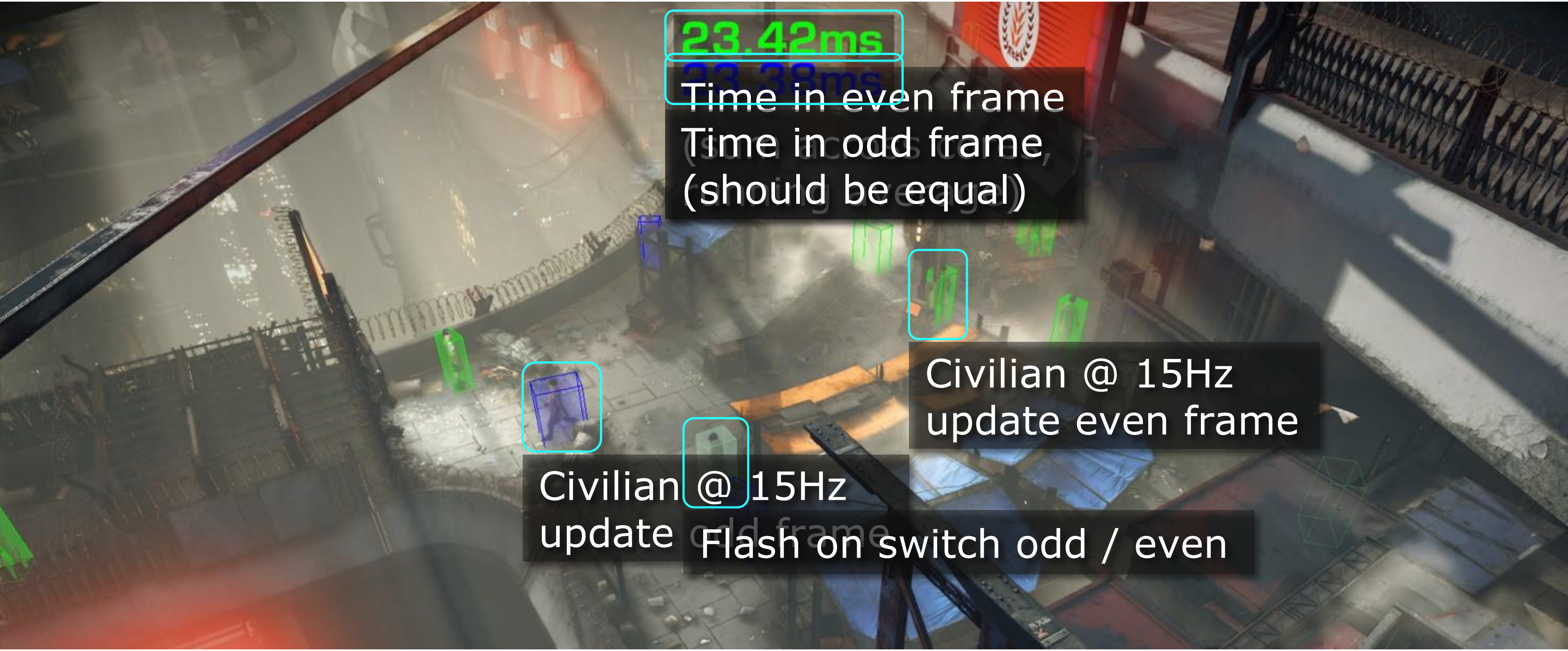
- Non cyclic dependency becomes cyclic job dependency
- Job1 and Job2 need to be merged

Balancing Entities Across Frames

Balancing Entities Across Frames

- Prevent all 15 Hz entities from updating in same frame
- Entity can move to other frame
  - Smaller delta time for 1 update
- Keep parent-child linked entities together
  - Weapon of soldier
  - Soldier on mounted gun
  - Locked close combat

Balancing Entities – In Action

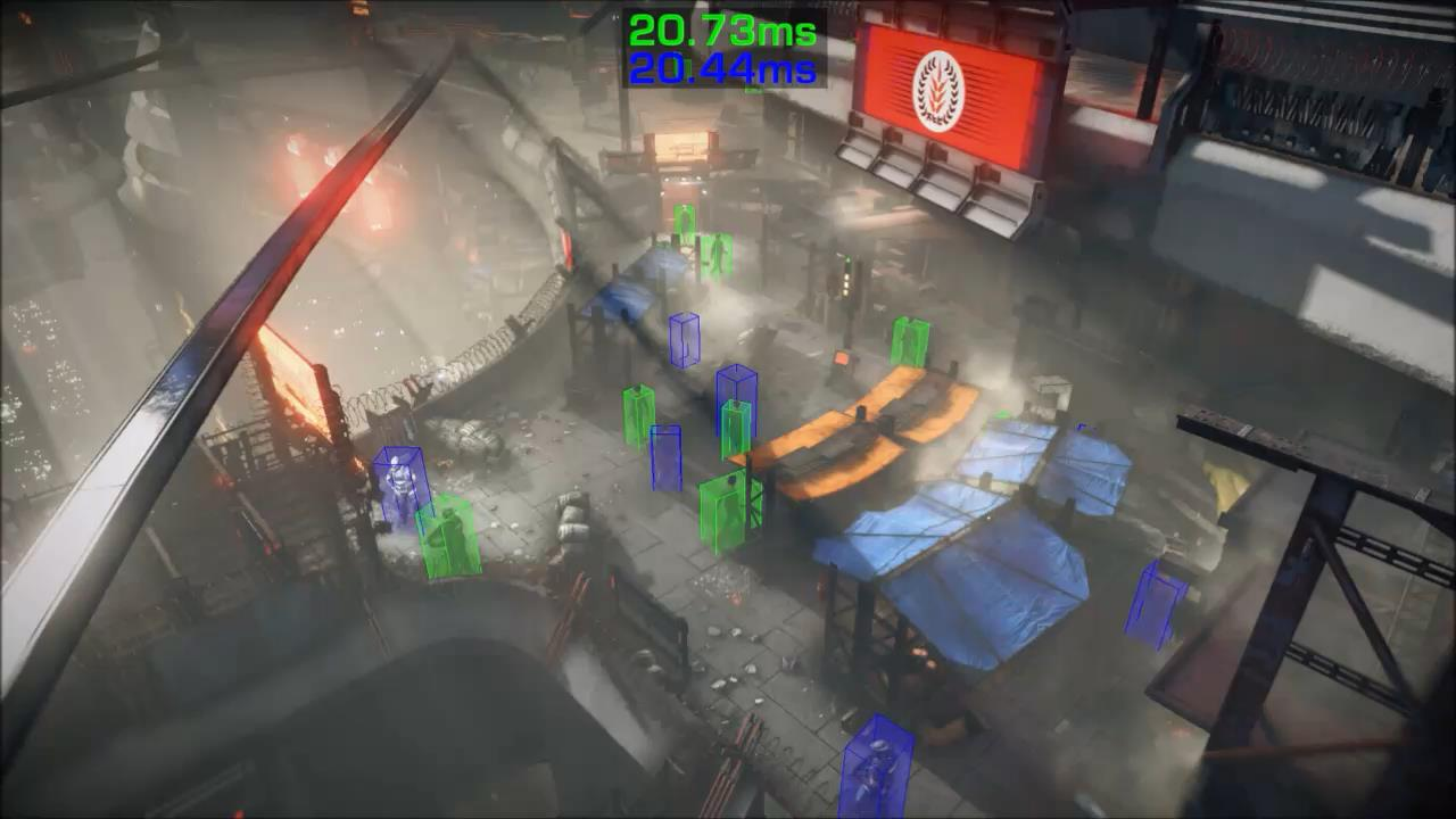


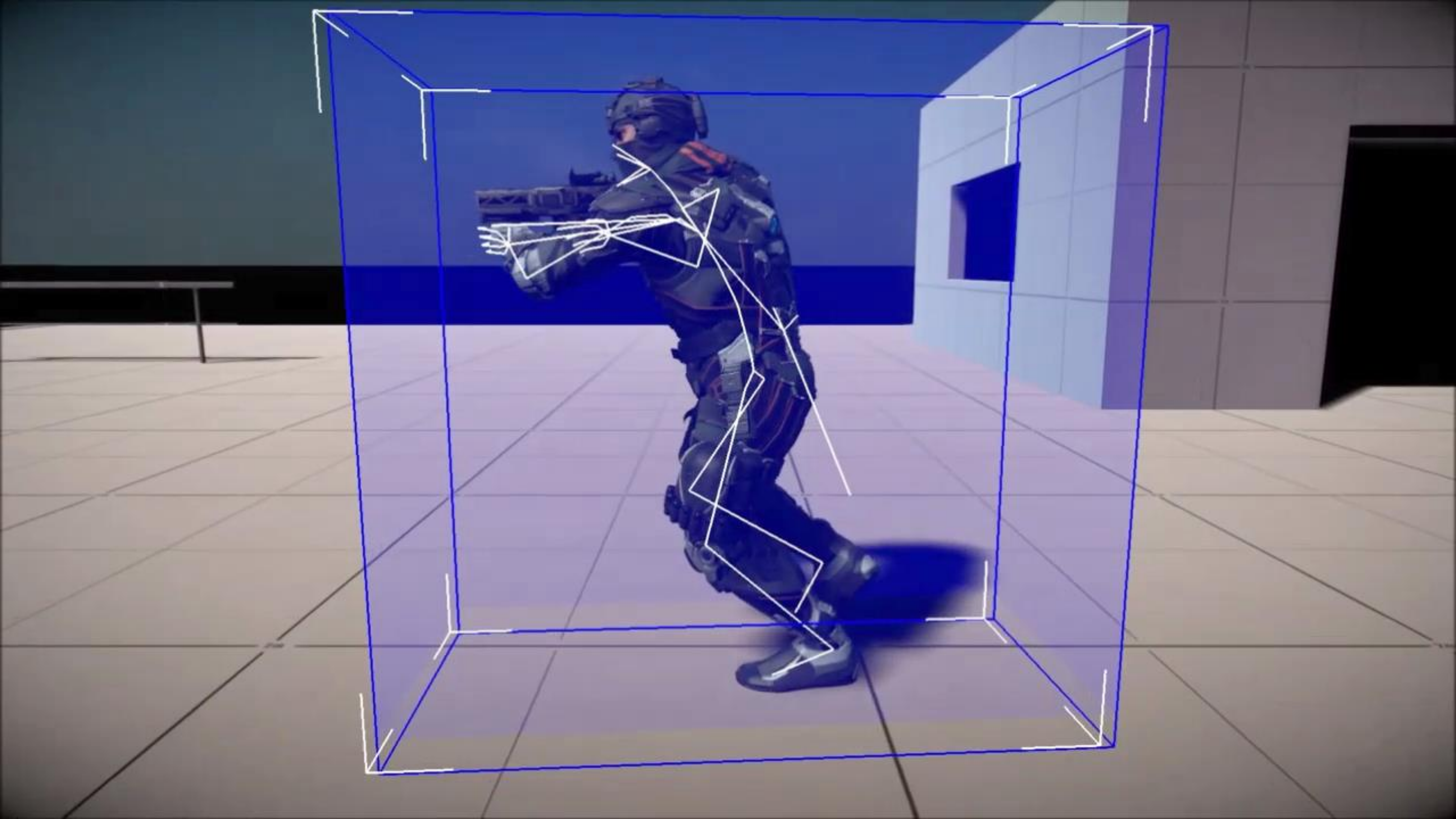
Time in even frame
Time in odd frame
(should be equal)

Civilian @ 15Hz
update even frame

Civilian @ 15Hz
update odd frame

Flash on switch odd / even


20.73ms
20.44ms

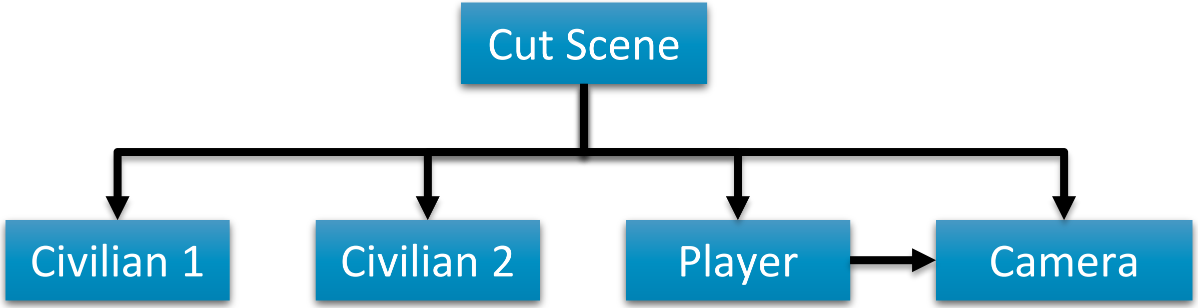


Performance Issues

Performance Issues

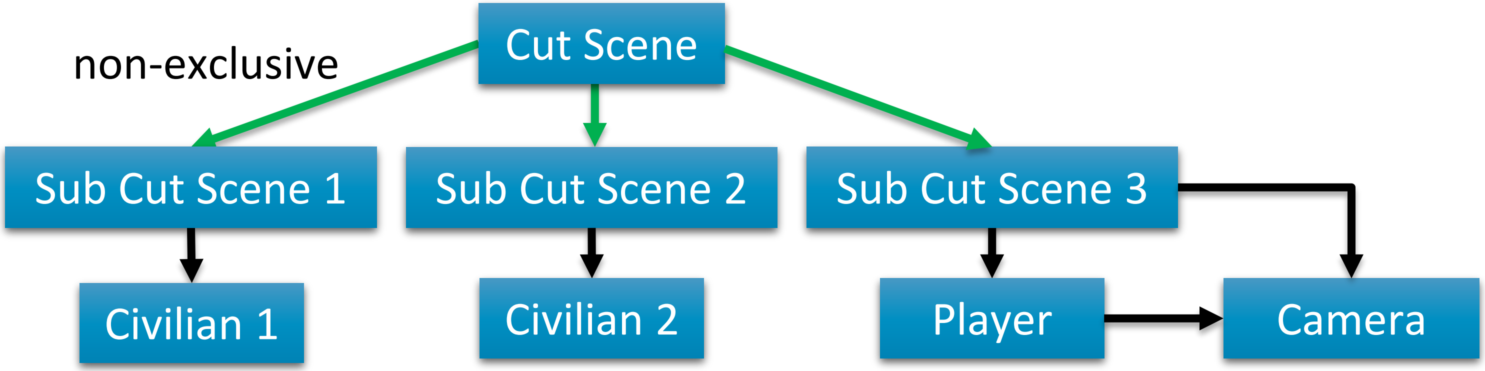
- Memory allocation mutex
  - Eliminated many dynamic allocations
  - Use stack allocator
- Locking physics world
  - R/W mutex for main simulation world
  - Second ‘bullet collision’ broadphase + lock
- Large groups of dependent entities
- Player update very expensive

Cut Scene - Problem



- Cut scene **entity** requires dependencies
- 10+ characters in cut scene creates huge job!

Cut Scene - Solution



- Create sub cut scenes for non-interacting entities
- Master cut scene determines time and flow
- Scan 1 frame ahead in timeline to create dependency

Using an Object

- Dependencies on useable objects not possible (too many)
- Get list of usable objects
  - Global system protected by lock
- ‘Use’ icon appears on screen
- Player selects
  - Create dependency
  - Start ‘use’ animation
- Start interaction 1 frame later (dependency valid)
- Hides 1 frame delay!

Grenade

- Explosion damages many entities
- Creating dependencies not option (too many)
- `ThreadSafeEntityInterface` not an option
    - Need knowledge of parts
- Run line of sight checks inside update
- Uses scheduled callback to apply damage

Performance Results

Performance Results - Synthetic

Level	Counts			Dependencies		Max Entities in Job	Speedup
	Number Entities	Updating Entities	Number Humans	Strong Excl	Strong Non-Excl		
5000 Crates (20 μs each)	5019	5008	1	12	4	13	2.8X
100 Soldiers (700 μs each)	326	214	105	212	204	19	4.2X
500 Flags (160 μs each)	519	508	1	12	4	13	5.2X

6 cores!

5000 Crates

100 Soldiers

500 Flags

Performance Results - Levels

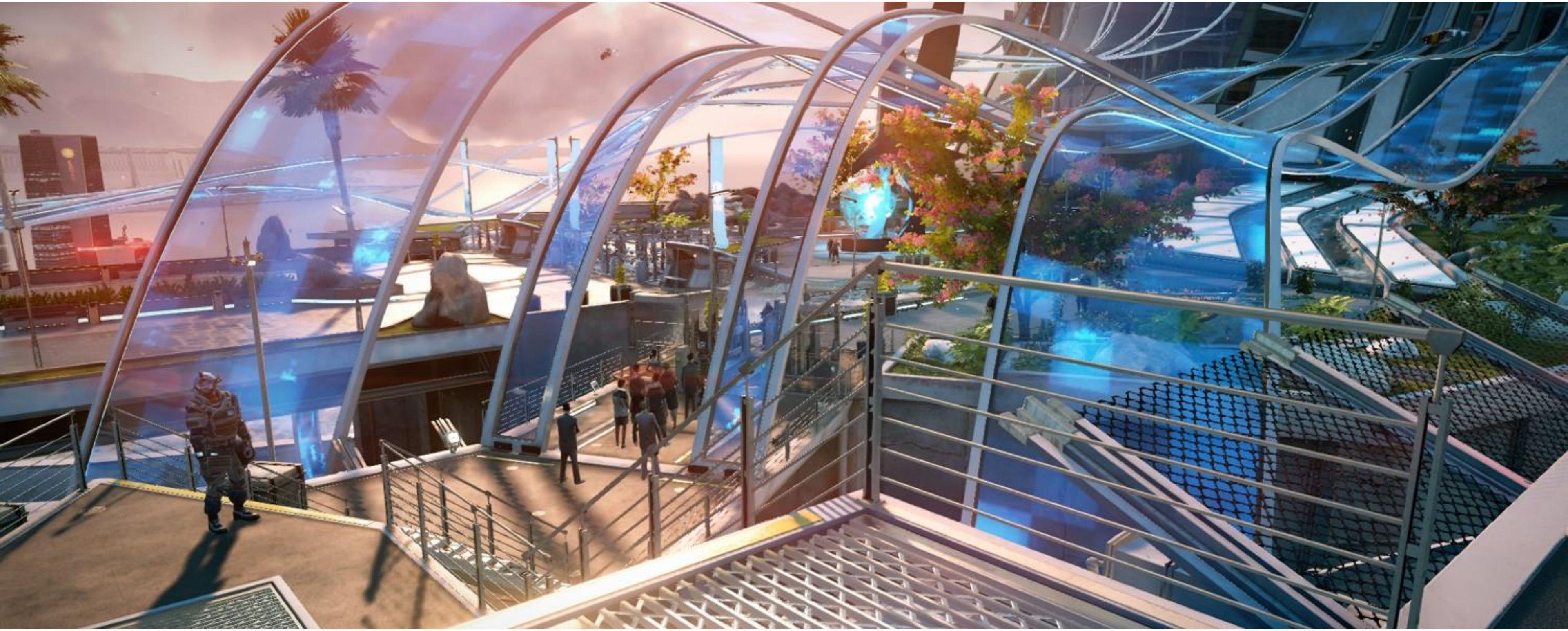
Level	Counts			Dependencies		Max Entities in Job	Speedup
	Number Entities	Updating Entities	Number Humans	Strong Excl	Strong Non-Excl		
The Helghast (You Owe Me)	1141	206	32	71	23	20	4.1X
The Patriot (On Vectan Soil)	435	257	44	199	107	15	4.3X
The Remains (12p Botzone)	450	128	14	97	44	18	3.7X

The Helghast

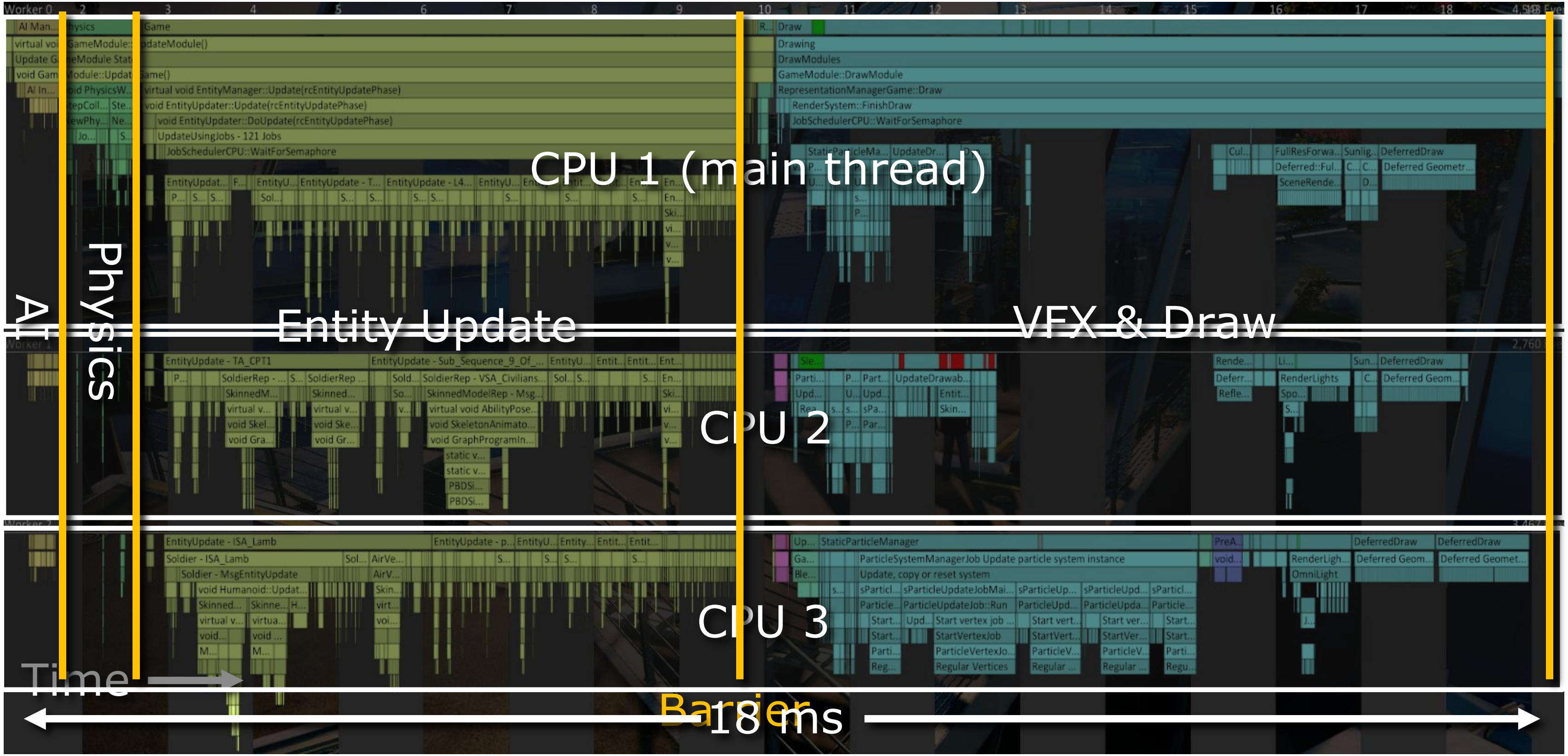
The Patriot

The Remains

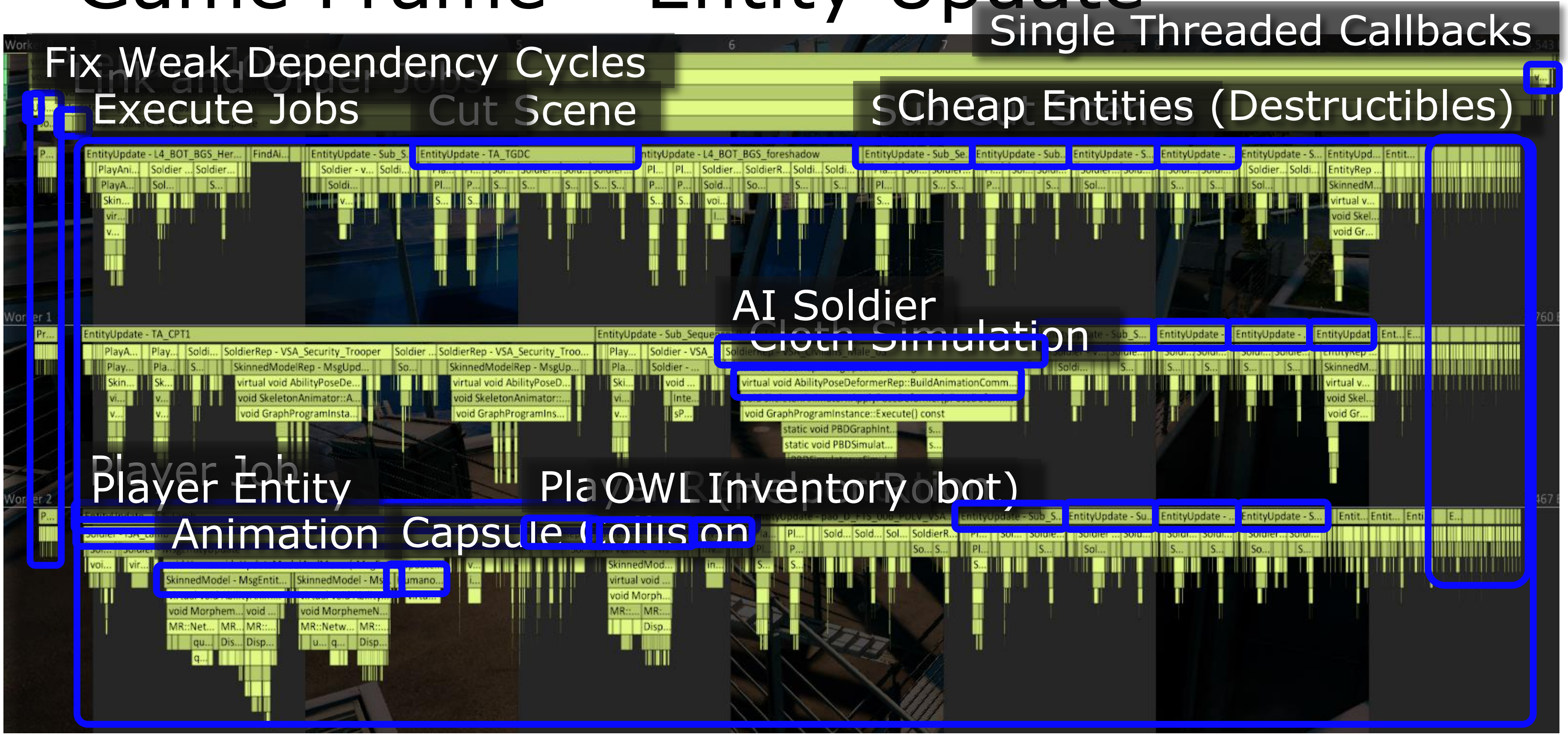
Game Frame - The Patriot



Game Frame - Global



Game Frame – Entity Update



Single Threaded Callbacks

Link and Order Jobs

Fix Weak Dependency Cycles

Execute Jobs

Cut Scene

Sub Cut Scenes

Cheap Entities (Destructibles)

AI Soldier

Cloth Simulation

Player Job

Player Entity

Player (Human Robot)

OWL Inventory

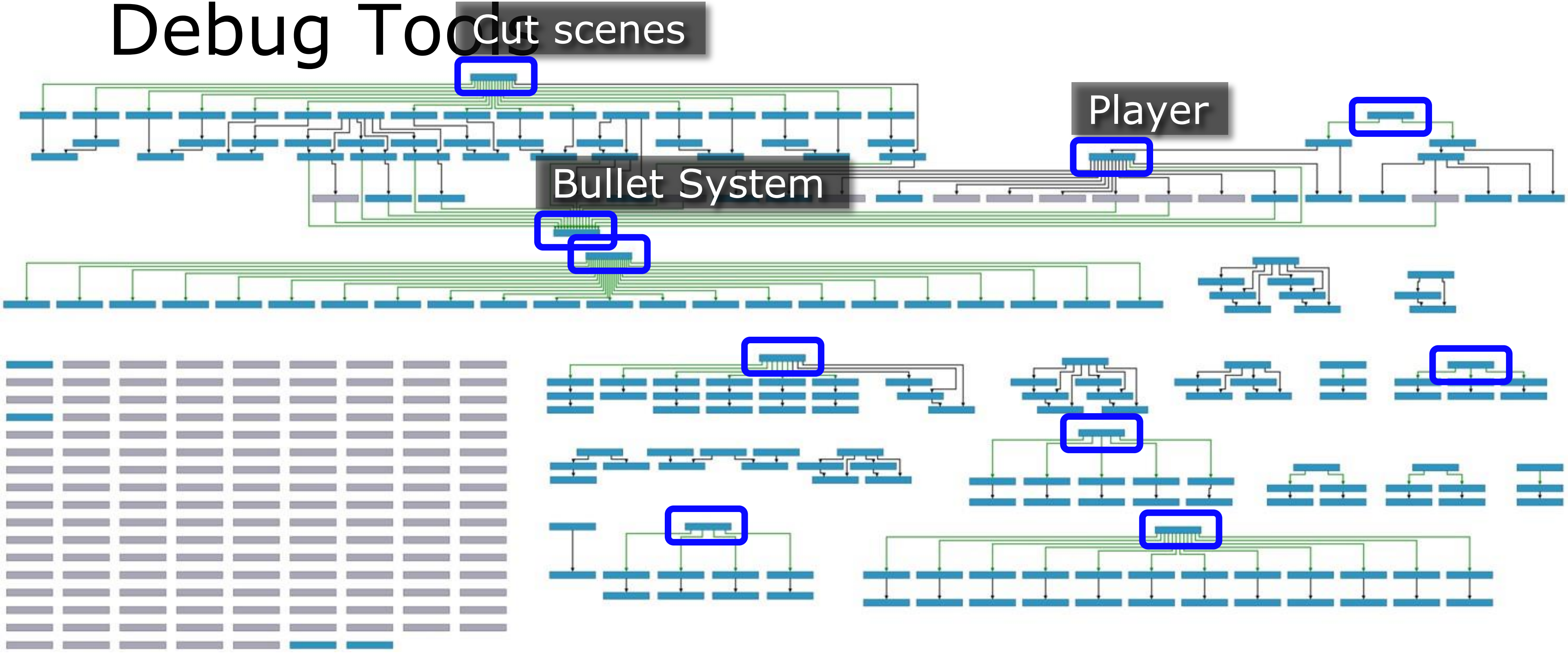
Animation

Capsule Collision

Worker
Worker 1
Worker 2
EntityUpdate - L4_BOT_BGS_Her...
FindAi...
EntityUpdate - Sub_S...
EntityUpdate - TA_TGDC
EntityUpdate - L4_BOT_BGS_foreshadow
EntityUpdate - Sub_Se...
EntityUpdate - Sub...
EntityUpdate - S...
EntityUpdate - ...
EntityUpd...
EntityRep ...
SkinnedM...
EntityUpdate - TA_CPT1
EntityUpdate - Sub_Seque...
SoldierRep - VSA_Security_Trooper
SoldierRep - VSA_Security_Troo...
SkinnedModelRep - MsgUpd...
SkinnedModelRep - MsgUp...
virtual void AbilityPoseDe...
void SkeletonAnimator::A...
void GraphProgramInsta...
virtual void AbilityPoseDeformerRep::BuildAnimationComm
void GraphProgramInstance::Execute() const
static void PBDGraphInt...
static void PBDSimulat...
SkinnedModel - MsgEntit...
SkinnedModel - Ms...
void MorphemeN...
void Morphem...
MR::Net...
MR::Netw...
virtual void ...
void Morph...
SkinnedMod...
6
7

Debug Tools

Debug Tools



- Dependencies get complex, we use yEd to visualize!

Conclusions

- Easy to implement in existing engine
- Game programmers can program as if single threaded
- Very few multithreading issues

Questions?

jorrit@guerrilla-games.com