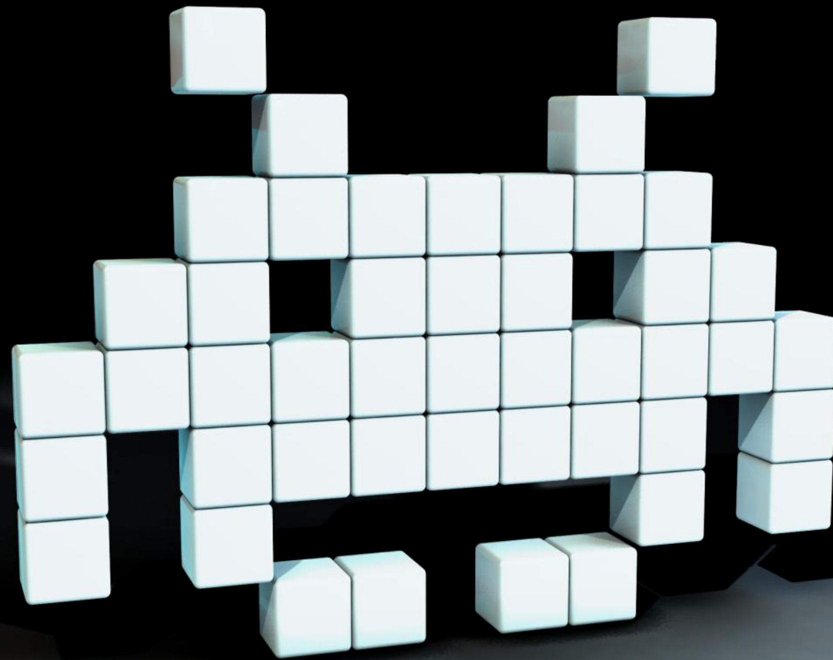
MFG

Innovationsagentur
für IT und Medien

HOCHSCHULE DER MEDIEN



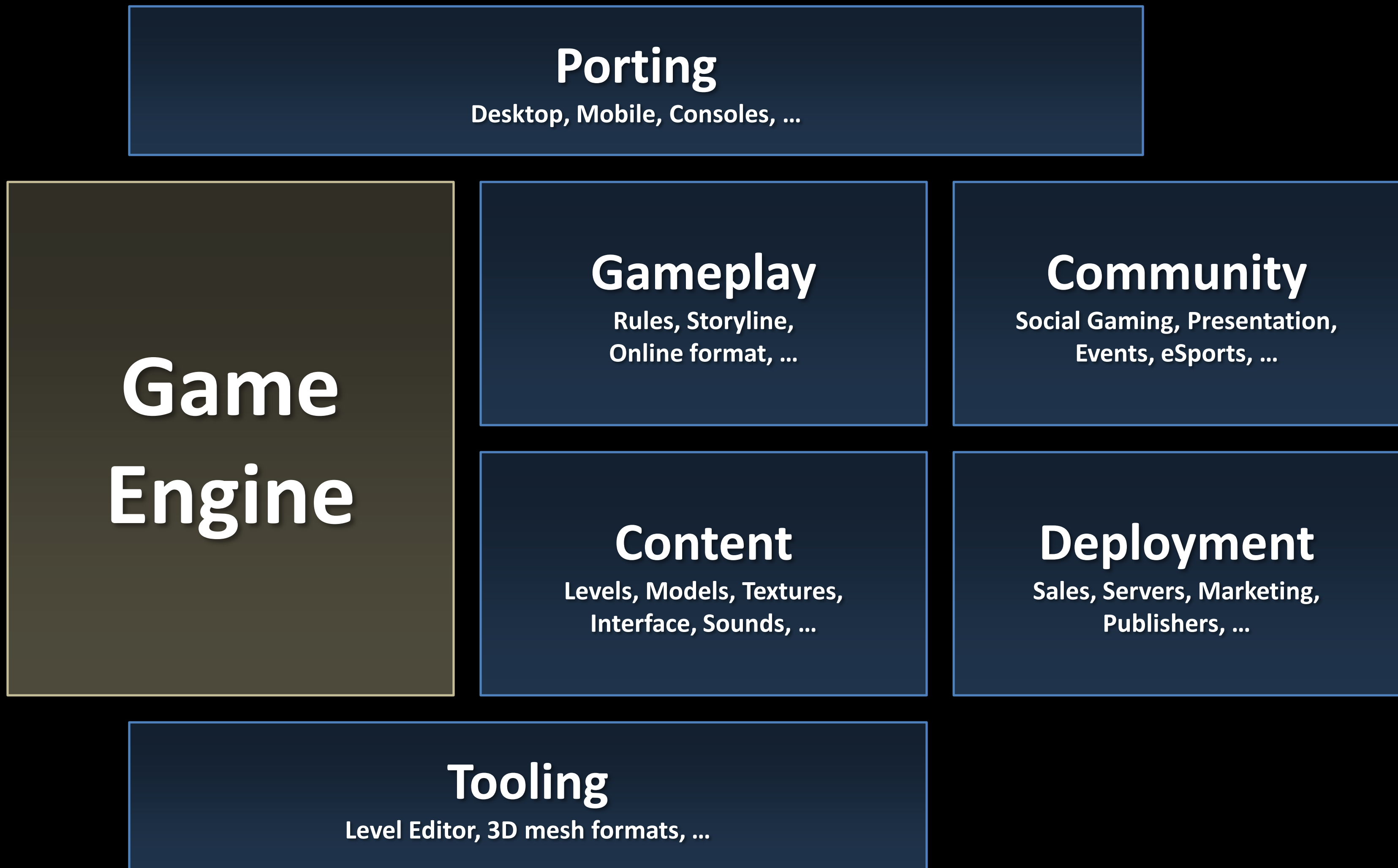
Summer Games University

Day 1: Engine Architecture

Real-time Game Development

Day 1 Day 2&3 Day 4

Game Engines Overview



Game Engines Overview

Engines are often even sold separately

Separating engine and content is not always wise or even possible (MMOs)

Image sources: "Mass Effect 2" BioWare, "Aion" NCsoft

Game Engine Dependencies

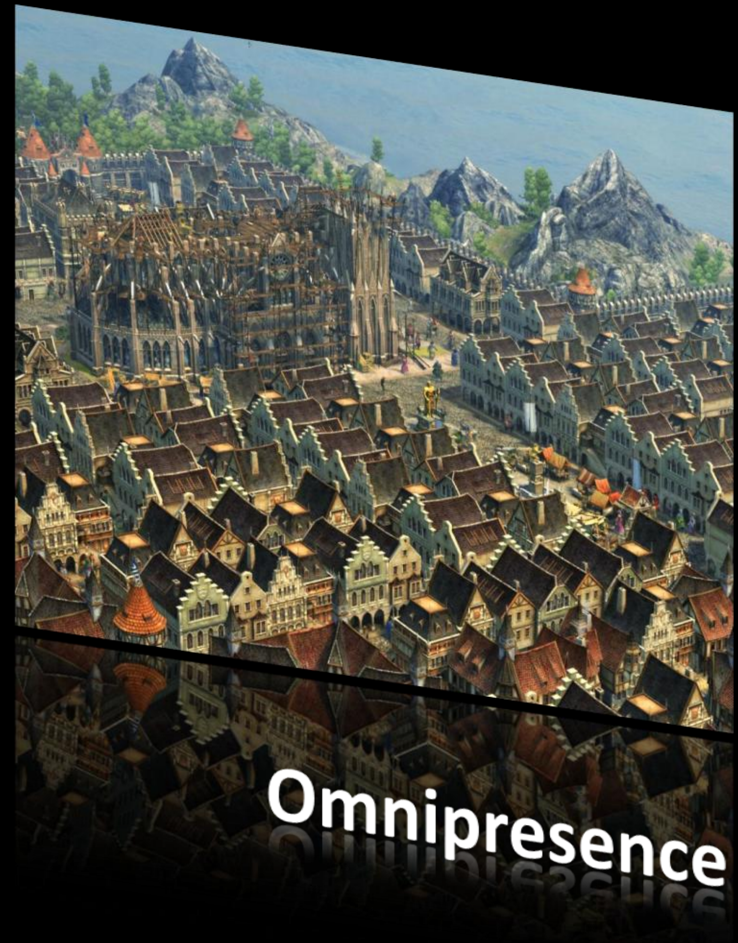
There are many factors that might have influence on the architecture

Two basic ways to handle these dependencies:

- Change the ***requirements*** to match what is possible
- Change the ***engine*** architecture to allow all requirements

Game Engine Dependencies

Genre dependencies



Imager sources: "Unreal Tournament Revolution" Epic Games, "Anno 1404" Related Designs

Game Engine Dependencies

Genre dependencies

Realism is a difficult topic.
Either achieve it..
Or don’t even try

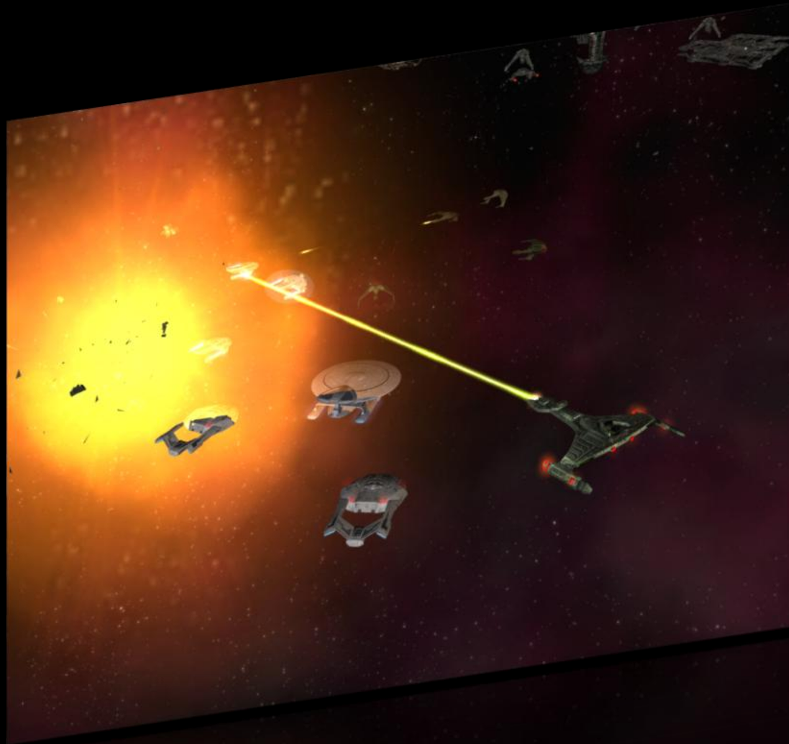
Character based games usually have a hard time to create a believable universe.

Image source: "Crysis" Crytek

Game Engine Dependencies

Genre dependencies

We will take a look at causality and predictability later on. Let's keep it short for now.



It should still be obvious, that games deploying an omnipresent player can't predict a players "Sphere of influence", which affects performance

Also has a large impact on network code!

Image source: "Fleet Operations" www.fleetops.net

Game Engine Dependencies

Setting dependencies

At first glance, the setting a game takes place in defines a lot of the game content, like models, levels, sounds and the like.

Physical destruction vs Character Animation

Image source: "Sins of a Solar Empire" Ironclad Games

Game Engine Dependencies

Platform dependencies

Well, it should be obvious that developing for consoles or multiple platforms changes a lot of the architecture.

Image source: "Nintendo Entertainment System" Nintendo

Game Engine Dependencies

Audience dependencies

Huu? Now what's that?

Service and socials functions might change the complete foundation of an engine.

A good example is the new Battle.Net 2.

Image source: BlizzCon 2009

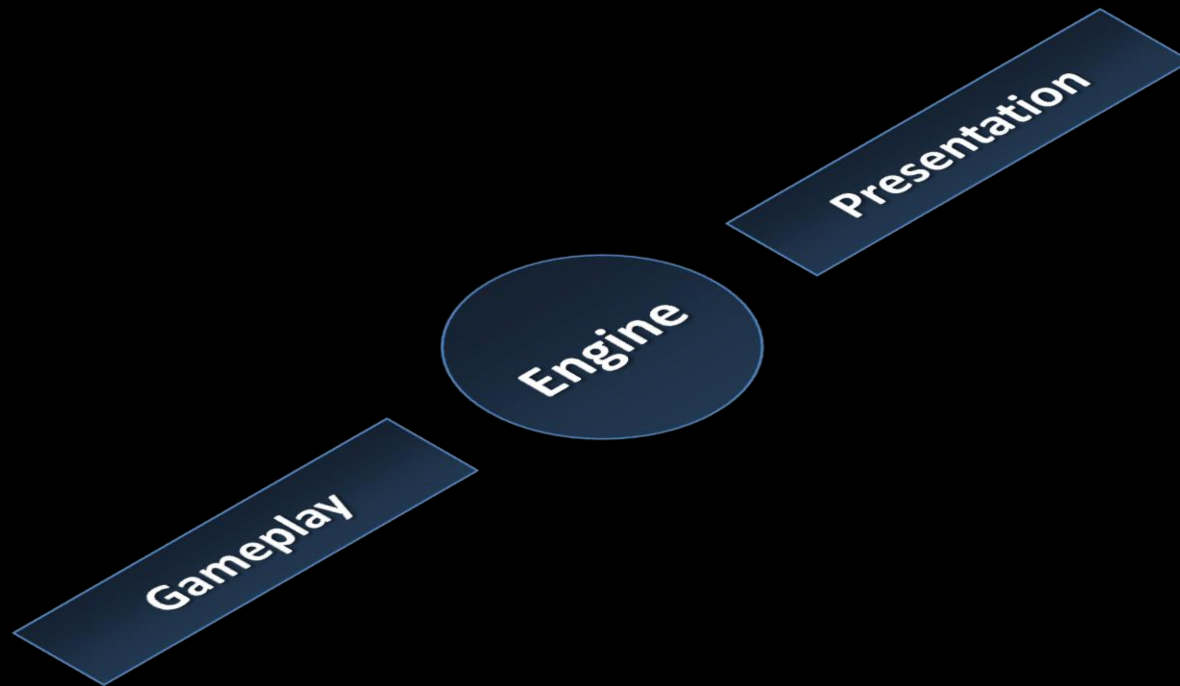
Game Engines Overview

Game Mechanics	Script Language	Update Routine
	Game Objects	Animation
Physics	Constraints	
	Physics Objects	Causality
Networking	Synchronization	
	Protocols	Cheating

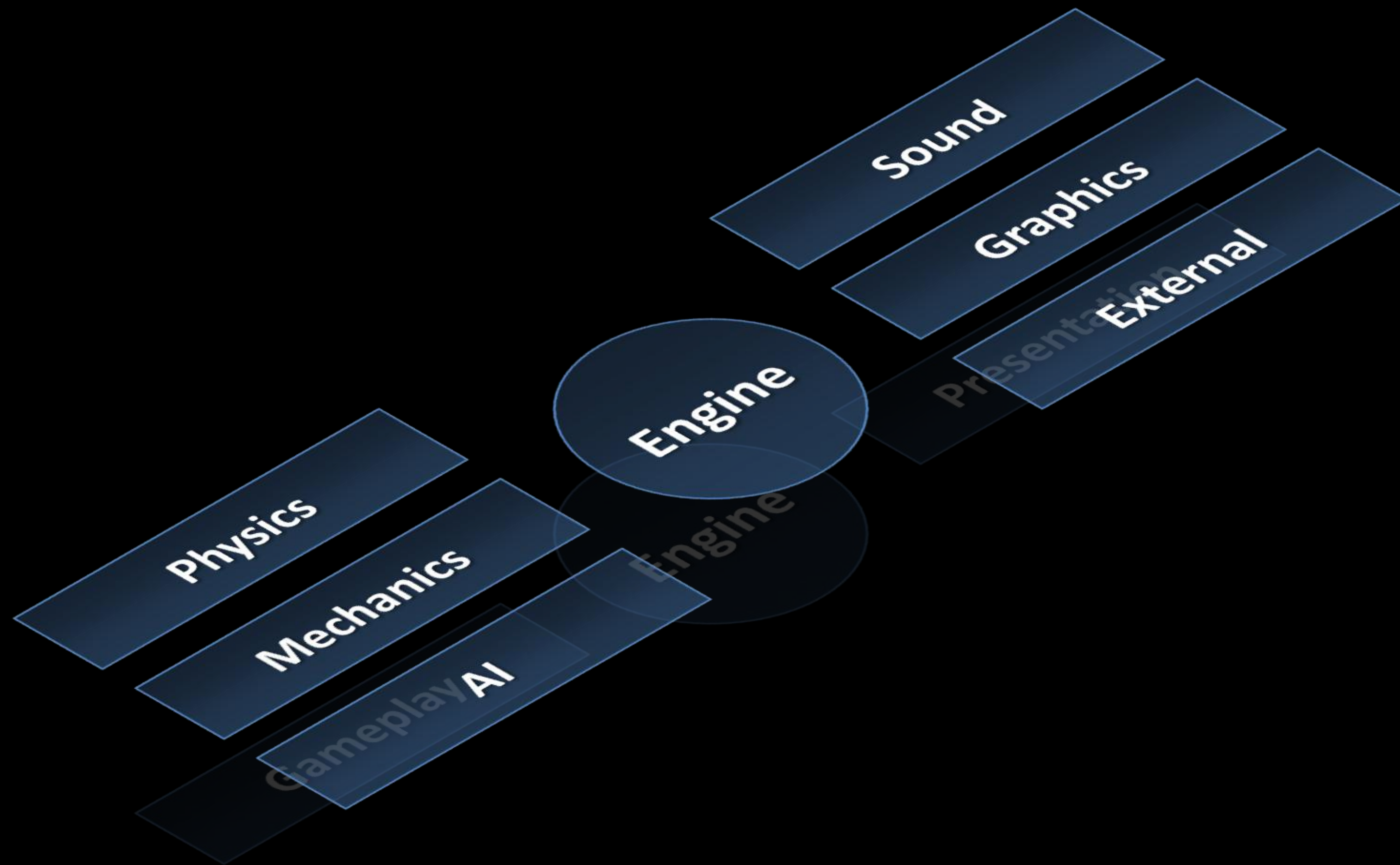
Game Engines Overview

Artificial Intelligence	Strategic	Memories
	Situational	AI World
Graphics	Renderer	Content Input
	Resources	
Interface	Framework	
Sound	3D Sound Objects	

Game Engines Overview



Game Engines Overview



Game Engines Overview

Game Engines Overview

A very basic engine prototype?

```
while(true)
  ProcessPlayerInput()
  Physics.Simulate()
  GameScripts.Simulate()
  Network.SyncState()
  SoundEngine.PlayEvents()
  GraphicsEngine.RenderState()
end
```

Time in Games

In the beginning…

there was ***time***

Image source: NASA

Time in Games

Why are we talking about time?

A scientific approach:

Let’s talk about causality!

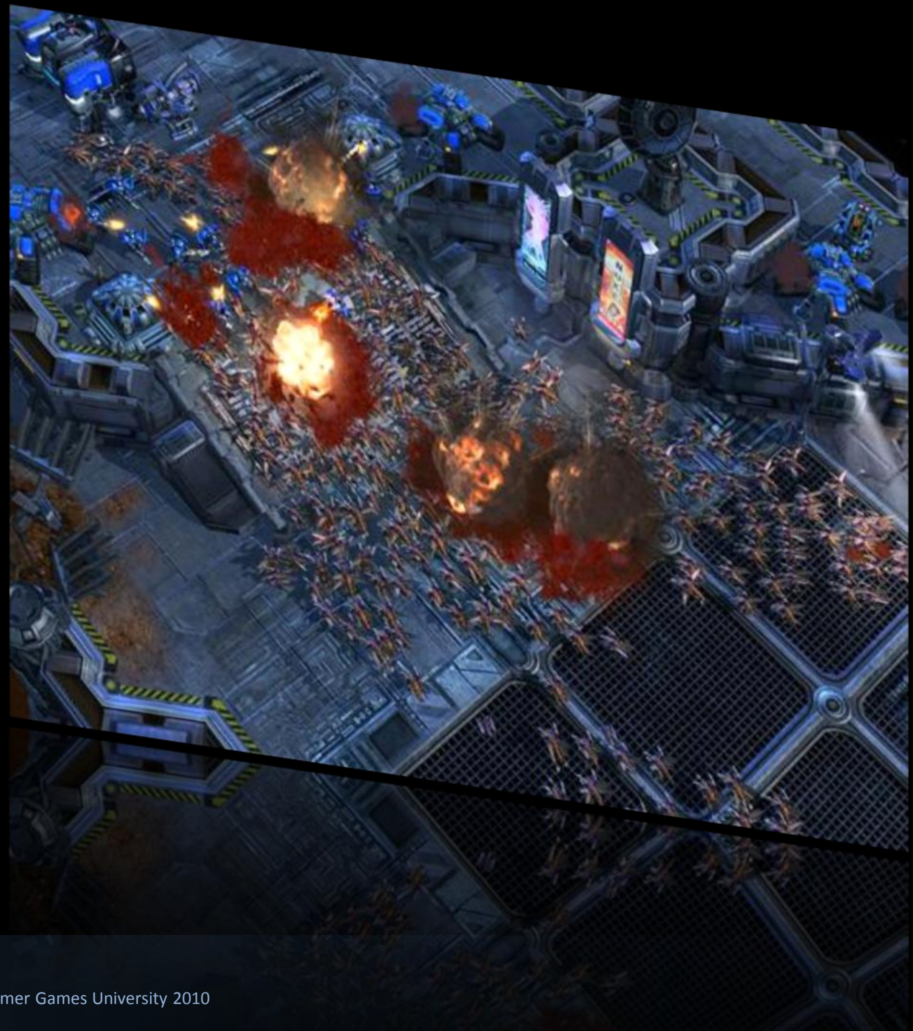


Image source: "StarCraft 2" Blizzard Entertainment

Time in Games

What's so difficult about real-time games?

Time itself is not the problem. But the way we recognize reality involves causality (something happening). And causality (as we recognize it) can be expressed as a function over time. Sadly, our real time is (almost) continuous.

But in order to resolve causality, we have to do calculations, to make something happen in our private universe. And the calculations involved are causal events in the "real universe" – therefore we will never get real-time.

Time in Games

We can't do real-time. We can't do calculations faster then god does! (not yet!)

Solution: Separate real-time and ingame-time

A cycle maps a certain fraction of real time to give us time for calculations.

Time in Games

We can't do real-time. We can't do calculations faster then god does! (not yet!)

Solution: Separate real-time and ingame-time

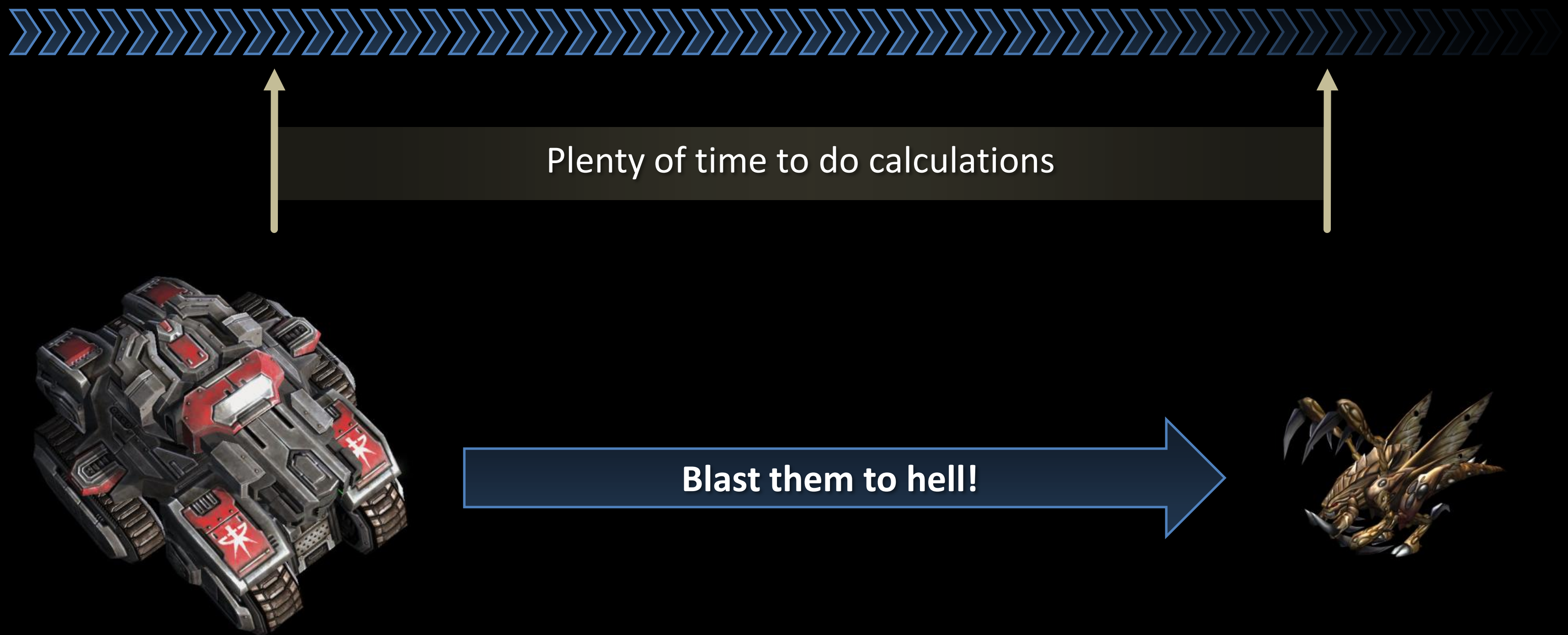


Image source: "StarCraft 2" Blizzard Entertainment

Time in Games

But what happens if events are triggered that ***between*** two cycles?

We can’t do real-time,
we can’t do real-causality,
…

Image source: "Supreme Commander" Gas Powered Games

Cycles

What our world is built of

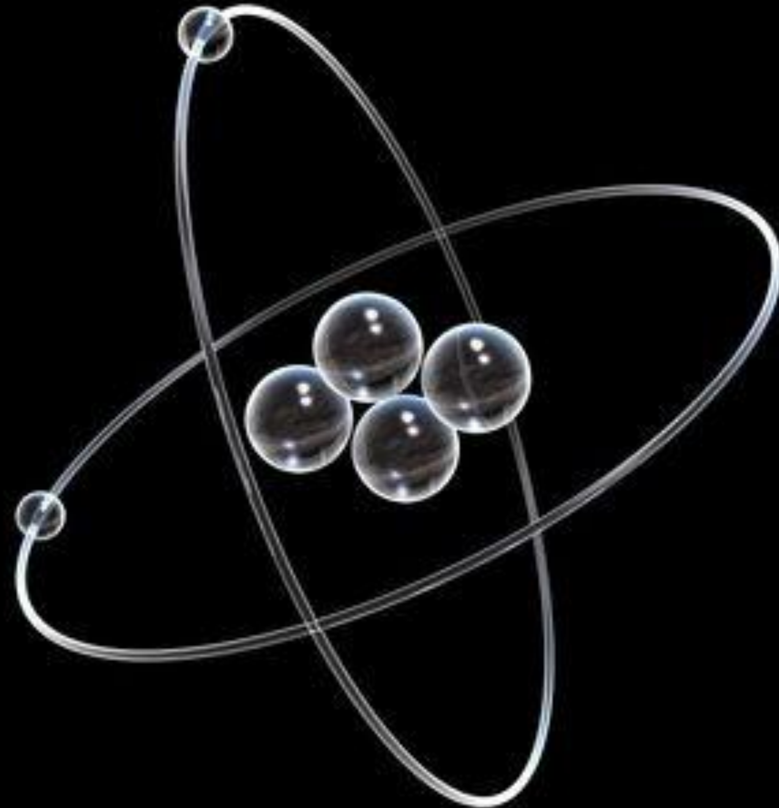


Image source: Strange Daze

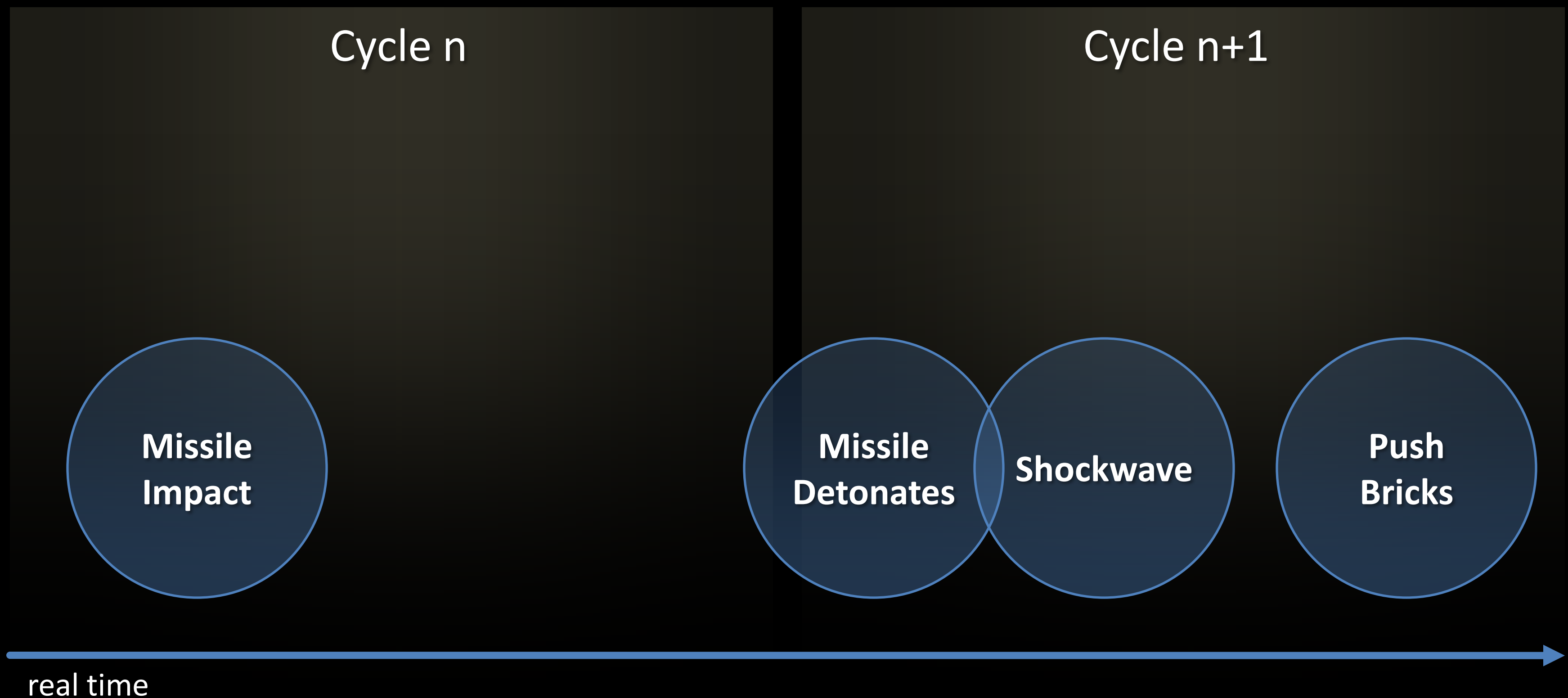
Cycles

The precise way cycles are handled is strongly dependant on the infrastructure and architecture an engine deploys. Let’s take a look at a few simple examples to get a feeling for the differences!

```
Function CycleUpdate
   elapsed = Clock.Elapsed
   Input.Apply()
   Physics.Simulate(elapsed)
   GameMechanics.Simulate(elapsed)
End
```

Teleporting Cycles

The most straight forward way is to “teleport” one cycle’s state into the next. This method is also often deployed by game mechanics engines.



Teleporting Cycles

We can’t simulate real-world causality!

Just like we did with time, we will define our own derivate of causality.

Causality is defined as the flow of information from one node to another.

Causal relations are obvious: A sender triggered an event and a recipient resolves it.

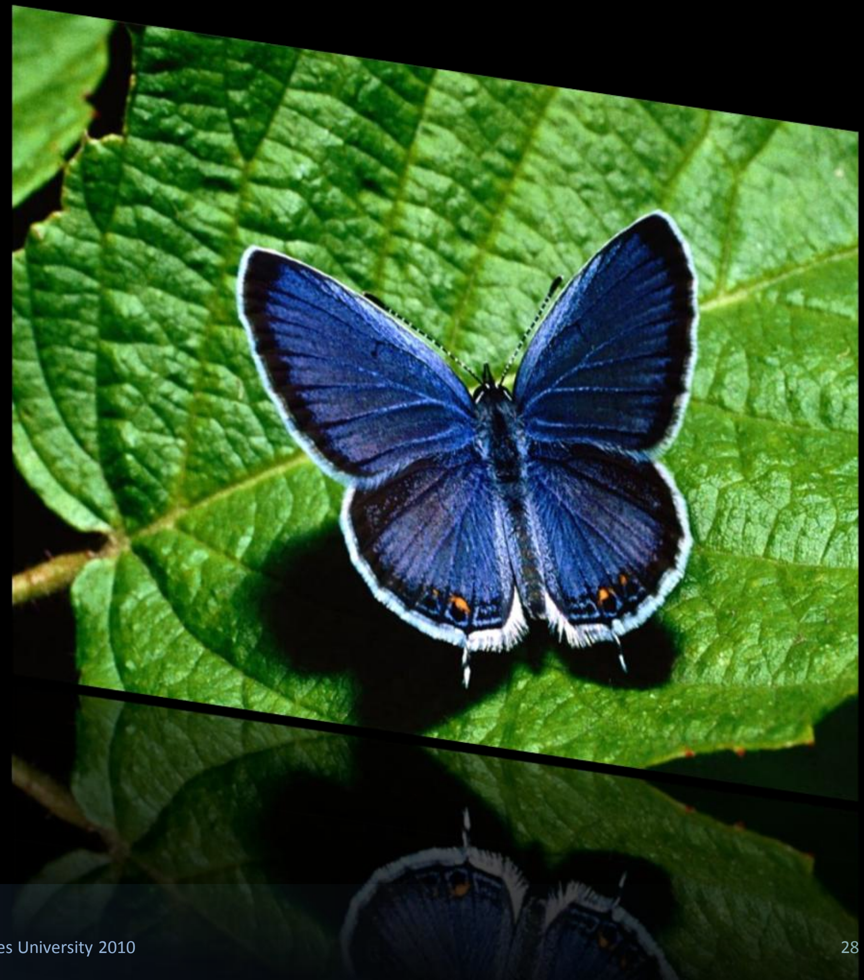
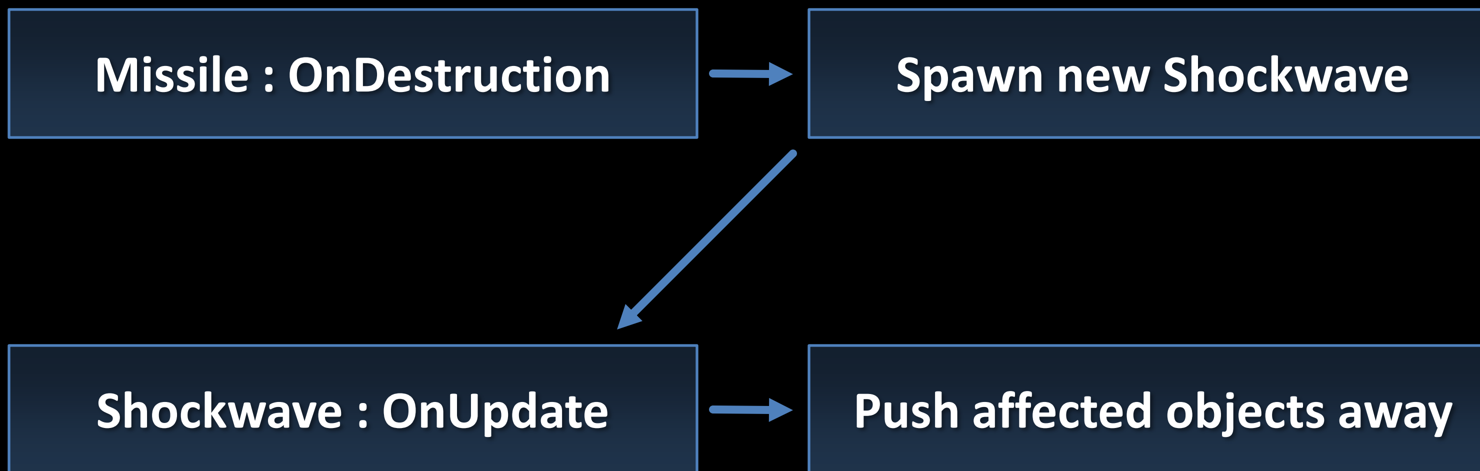


Image source: all best wallpapers

Teleporting Cycles

Solve causal dependencies as call-chains



Problems:

- Multiple updates per object
- Or multiple update passes

Both lead to strong object-object links

Teleporting Cycles

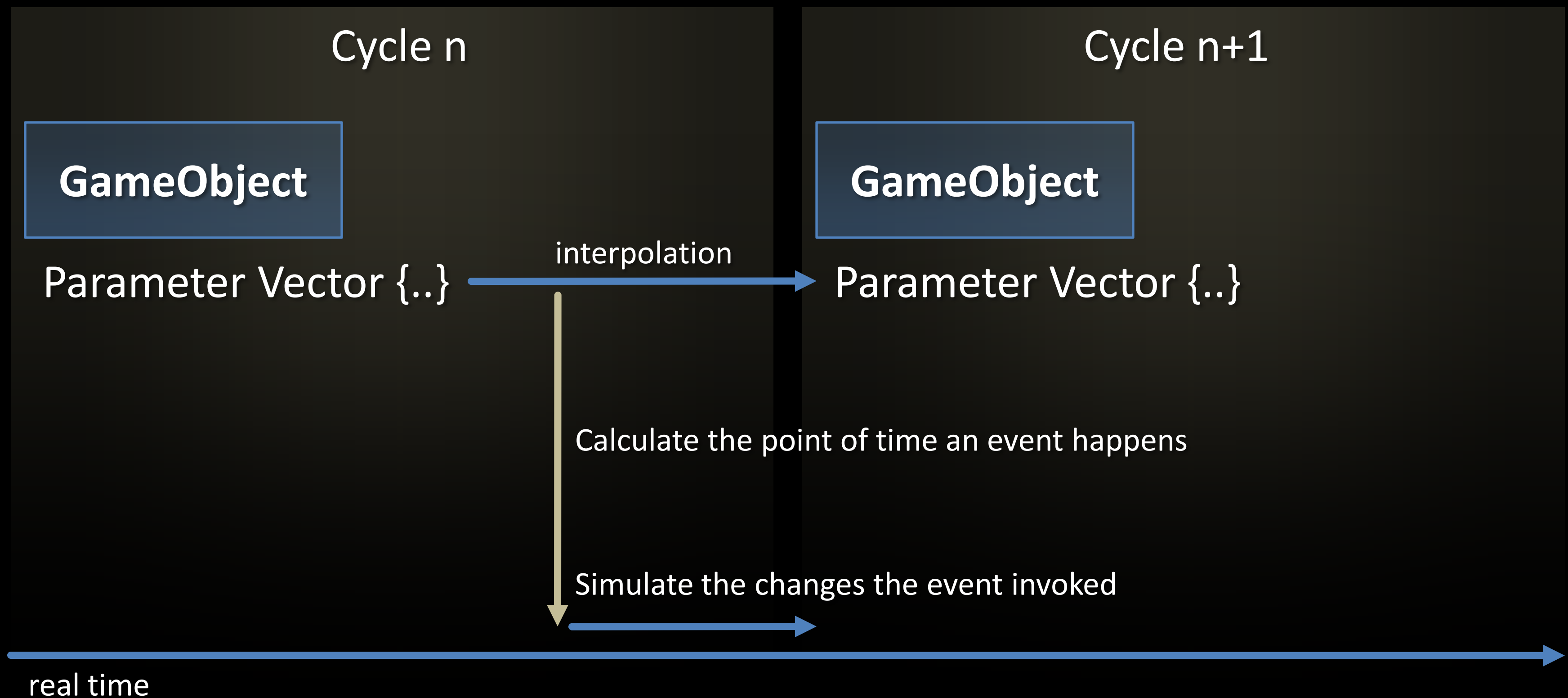
Solve causal dependencies with messages

Problems:

- Causal events might be blurred through time

Continuous Cycles

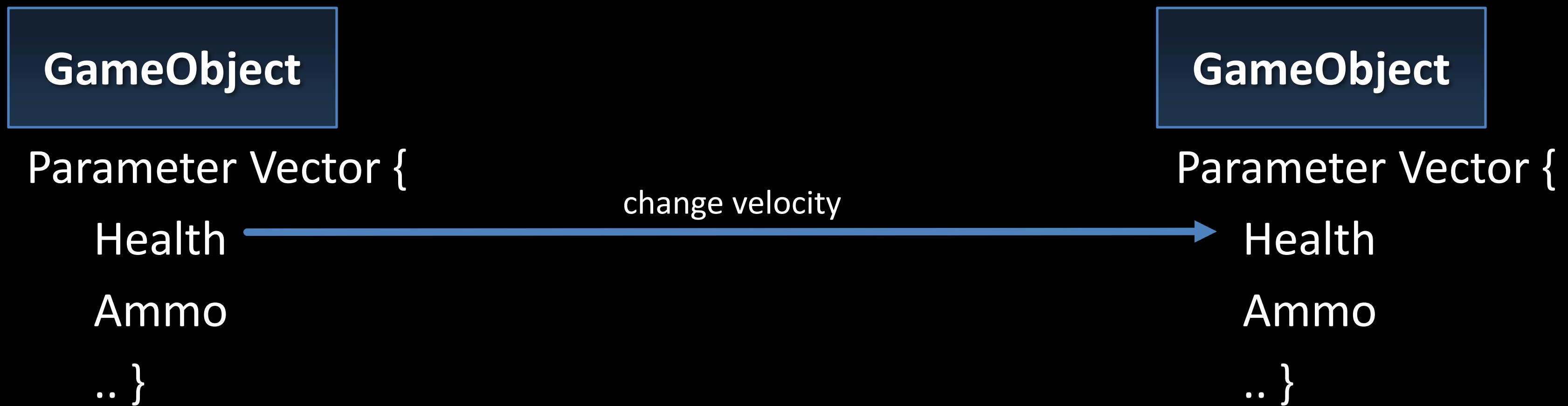
Try to reconstruct the precise time an event happens



Continuous Cycles

Many redundant calculations due to events

Think in velocity rather then attribute fields. This speeds up calculation.



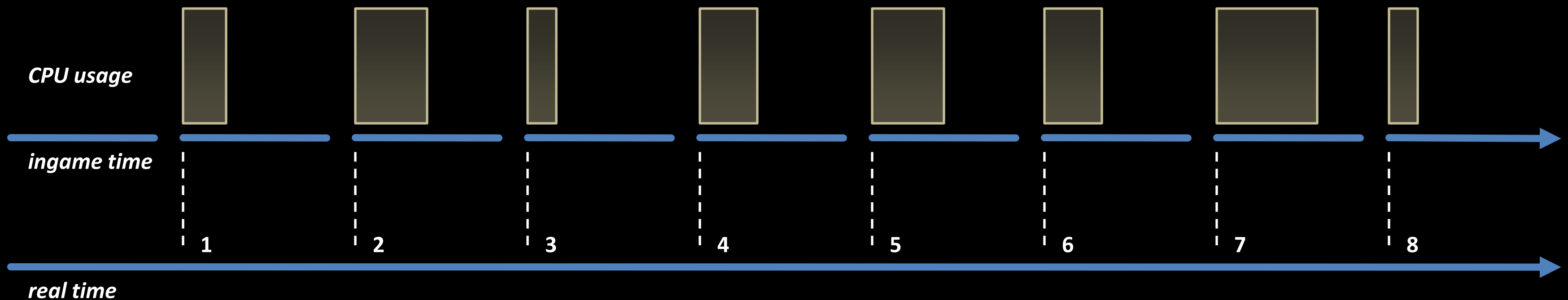

```
Health = Health +
  HealthVelocity * elapsed
```

Maping time on cycles

As our game universe changes, it is probable that its complexity will also change, involving more calculations per real time second. How do we deal with that?

Constant cycle length

Each cycle represents a defined fraction of real time, for example 100 µs



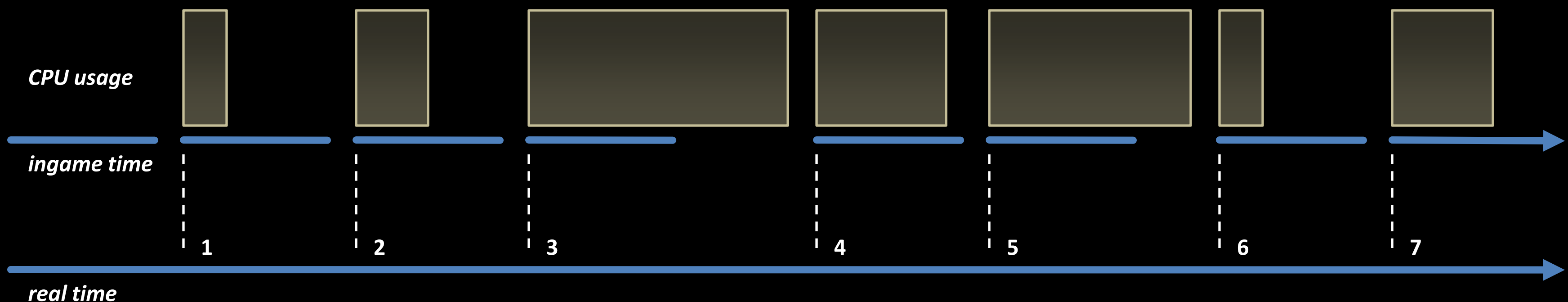
Maping time on cycles

As our game universe changes, it is probable that its complexity will also change, involving more calculations per real time second. How do we deal with that?

Constant cycle length

Each cycle represents a defined fraction of real time, for example 100 µs

Our simulation slows down!



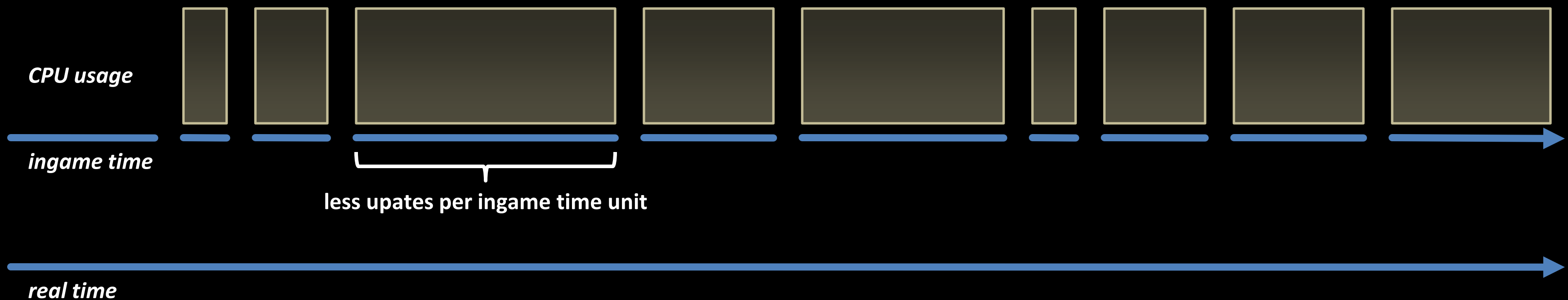
Maping time on cycles

As our game universe changes, it is probable that its complexity will also change, involving more calculations per real time second. How do we deal with that?

Variable cycle length

Each cycle represents a fraction of real time, equivalent to the real time it took to calculate it

Our simulation-resolution slows down, which will produce more calculations



Game Objects

What else does our universe need?

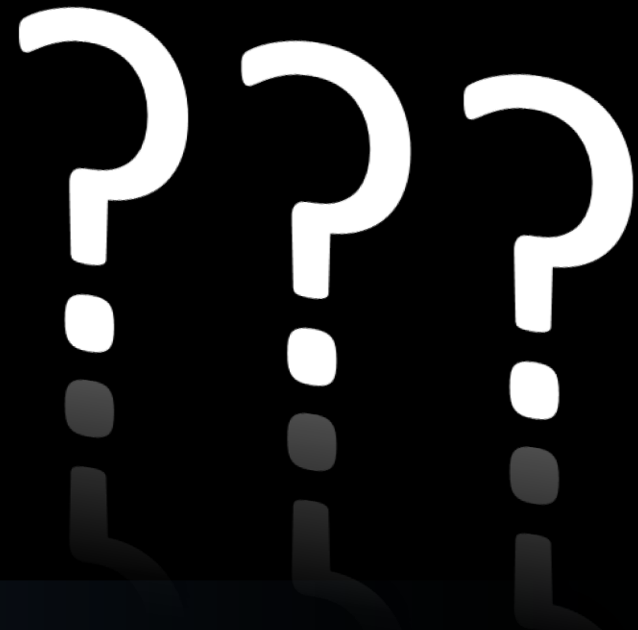
mass! ***Well, and information!***

Image source: NASA

Game Objects

What do Game Objects do?

- Represent an entity of some sort in the “ingame universe”
- Represent an accumulation of information
- Represent ports to produce, receive and process events
- Represent nodes in logical causal hierarchies
- Link GameMechanic entities and physics entities
- ..



Game Objects

Let's start with Game Objects representing entities

Don't necessarily have to be spatial entities. Quests for example.

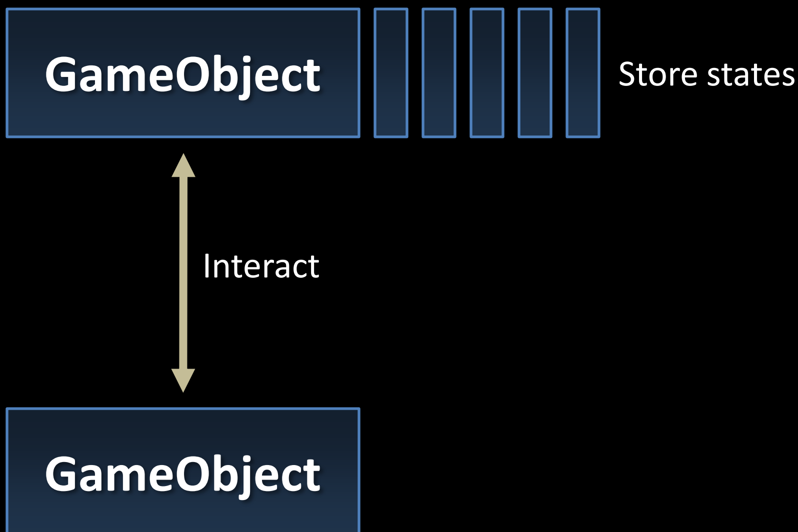
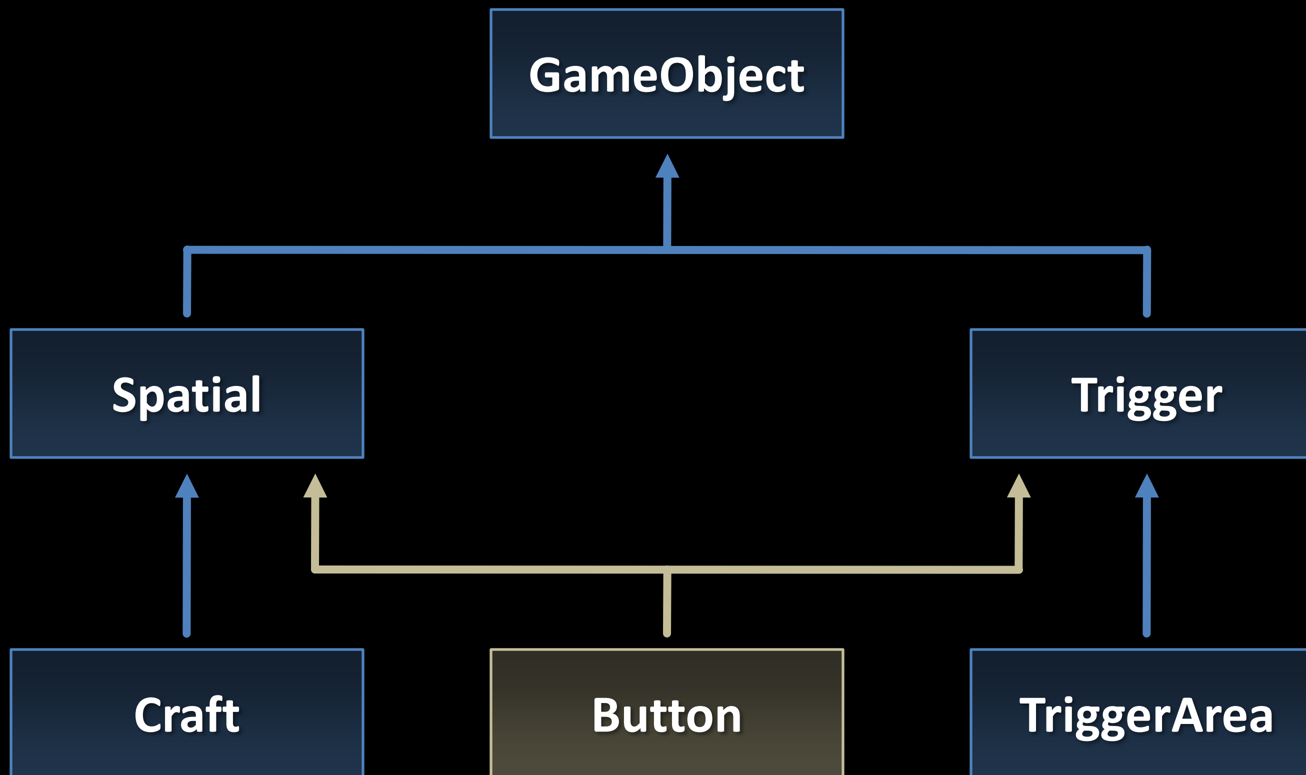


Image source: "World of Warcraft" Blizzard Entertainment

Game Object Representations

Let's take a small break and think about Game Object representations

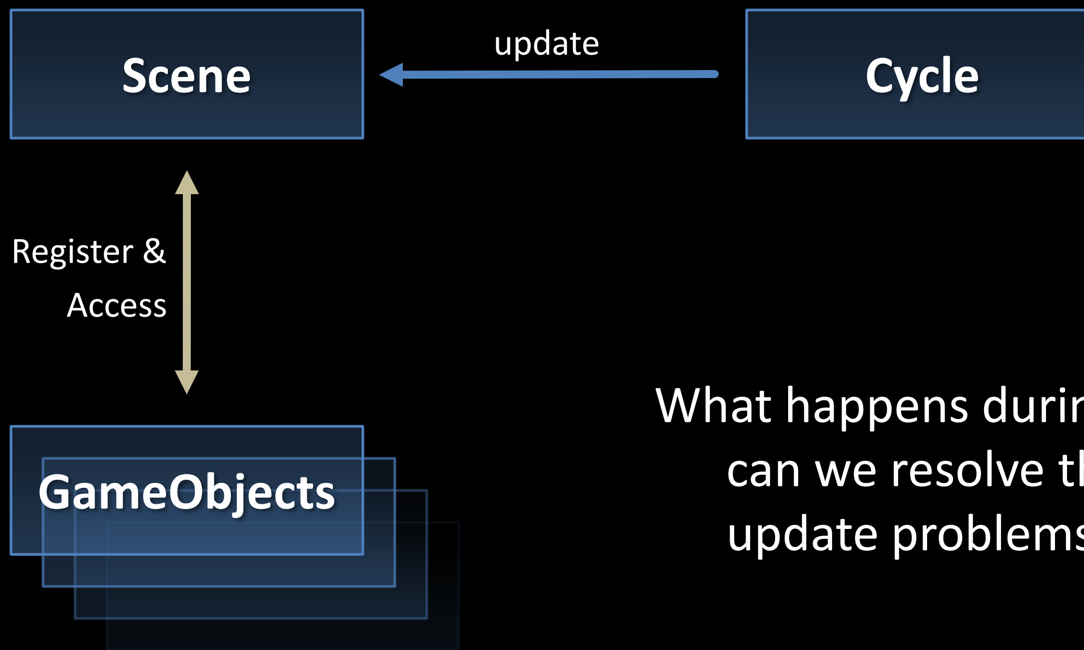


Game Object Representations

Let’s try “has-a” relations. The “nested objects” might be stored in individual scenes to use a good scene graph (quadtree for PhysicsObjects, ..)

Game Objects and the Universe – Part 1

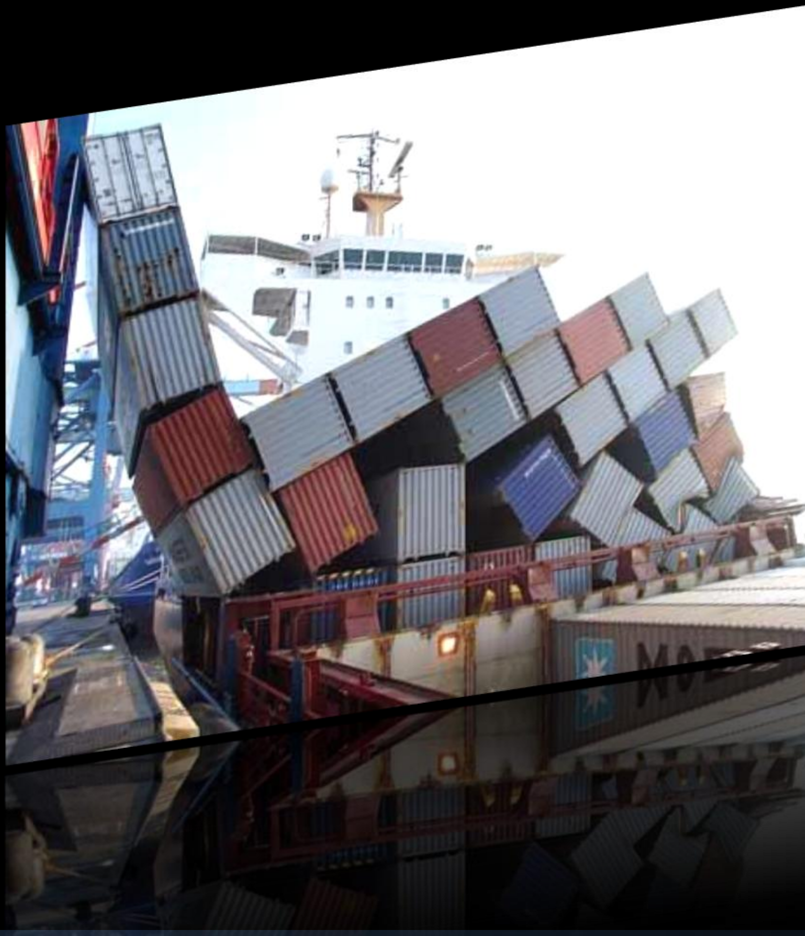
Game Objects have to be linked to the universe (the cycle runtime) in some way. They usually register upon spawning.



What happens during a scene update and how can we resolve the various GameObject update problems?

Game Objects and the Universe – Part 1

Game Object containers have a relevant impact on a games performance!



Some buzzwords and headlines for discussion?

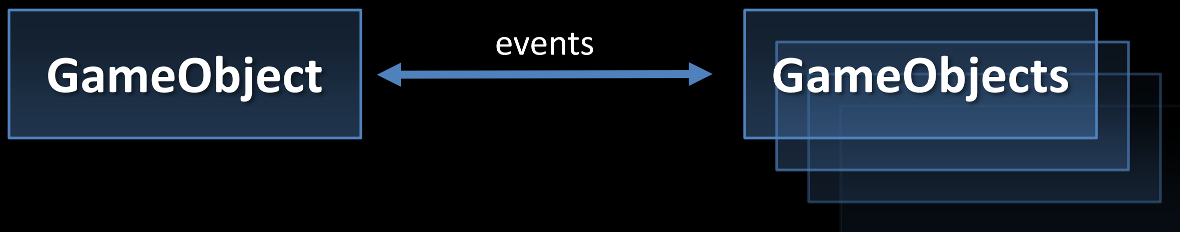
- Array
- Dictionary
- QuadTree
- Logic Hierarchy
- Sorting? When?
- Game Object IDs vs References
- Keeping Game Objects alive

Image source: Bild.de

Active Objects

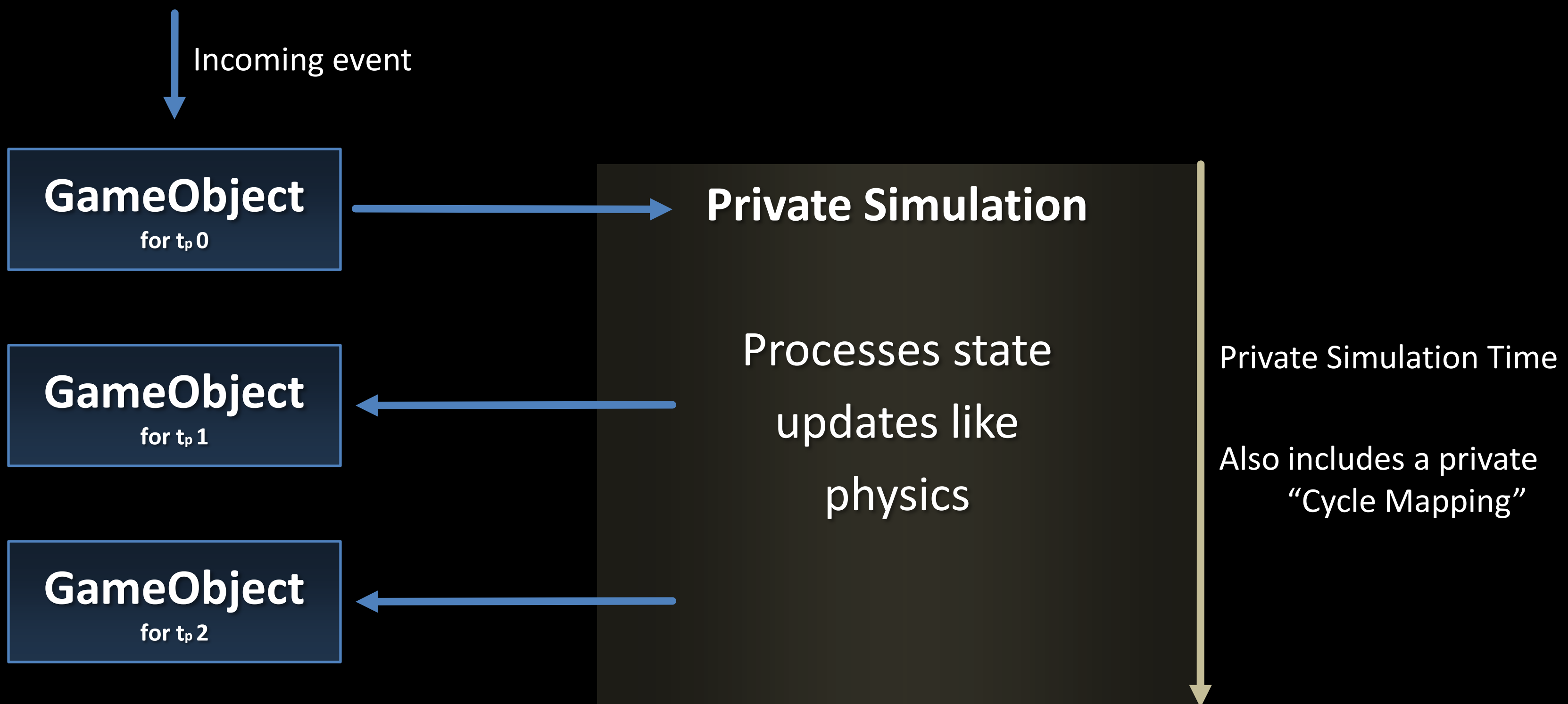
There are many different approaches on how to manage and update game objects.

One might pop into your head: Let the objects manage themselves!
Also known as “Active Objects”



Active Objects

The goal of Active Objects are to reduce unnecessary calculation to a minimum, by allowing objects to author themselves and therefore precisely control their state updates



Active Objects

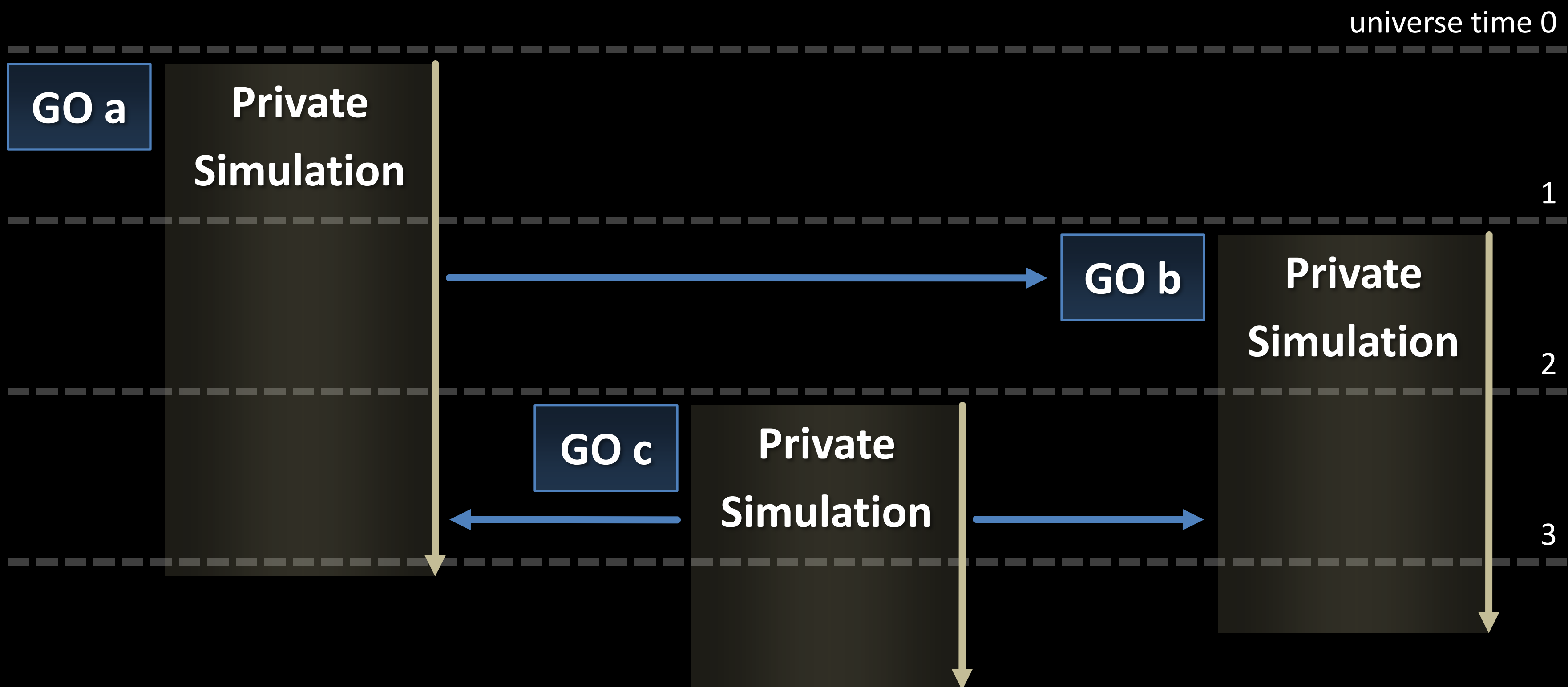
What happens if objects interact with each other?

The private simulation of object a throws an event at b

Active Objects

All is fine.. Until Game Object c joins the fun!

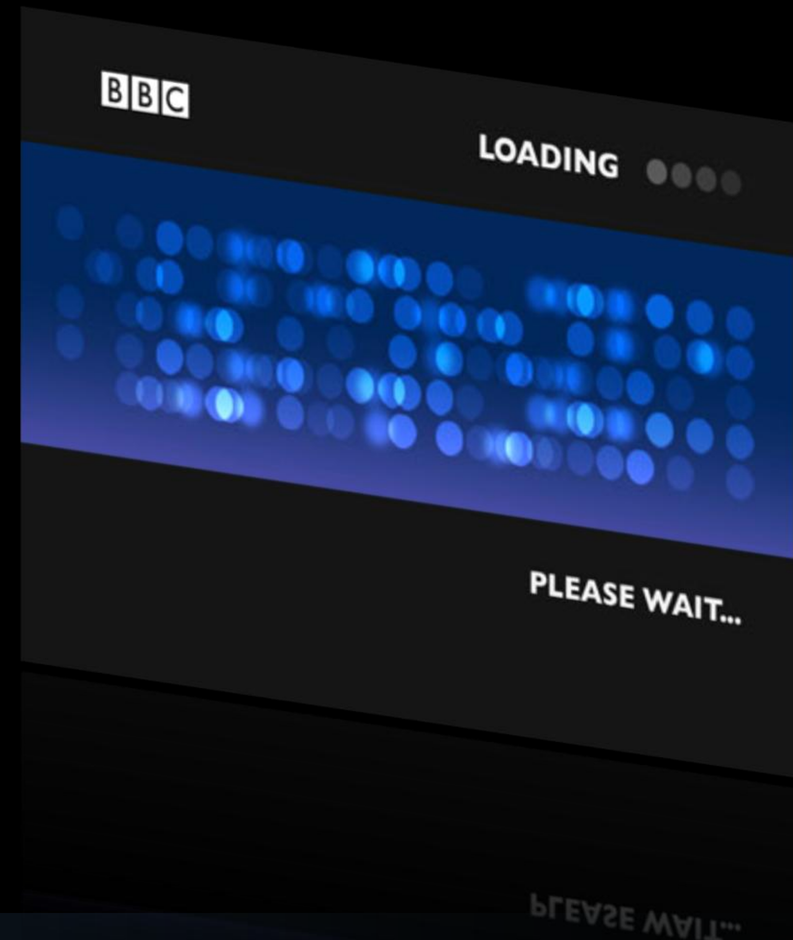
The instances of a and b do not necessarily share the same private time!



Active Objects

Active Objects lead to Game Objects loosing their temporal relation, as they exist in their very own time

This might lead to..



The issue is often solved by introducing a central clock which updates the Game Objects at given intervals..

Congrats! That's a cycle

Image source: BBC

More on game objects

We will continue this topic once we learned a bit more about synchrony tomorrow