



SCALABLE DYNAMIC SYNTHETIC ENVIRONMENTS USING A NEXT-GENERATION GAME ARCHITECTURE

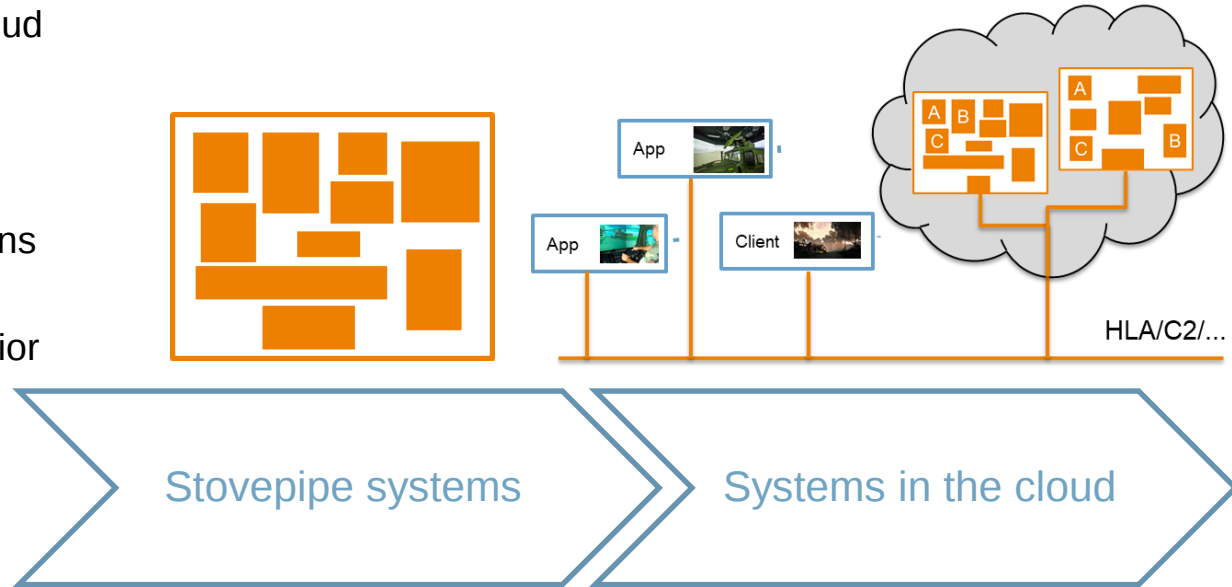
ROBBERT KRIJNEN, RUBEN SMELIK - TNO

Image: Crackdown 3 - Pre-alpha in-game footage

CURRENT STATE OP PLAY

Moving existing systems to the cloud

- + Centralized resources
- + “Reusable”
- Still large monolithic applications
- Not really (micro)services
- Hard to change specific behavior



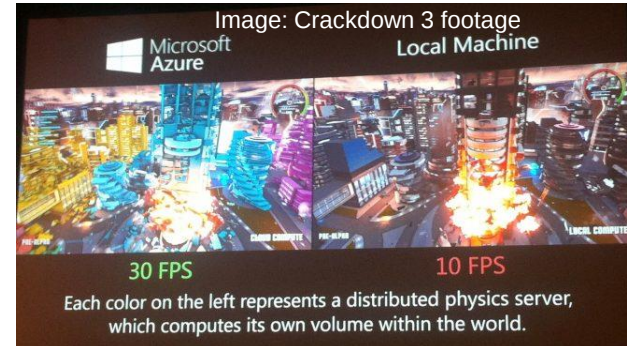
GAMING TRENDS

- › Agile development – daily releases
- › Cross-platform play
- › Cloud integration (game server, physics service, ...)
- › Massive (destructible) game worlds

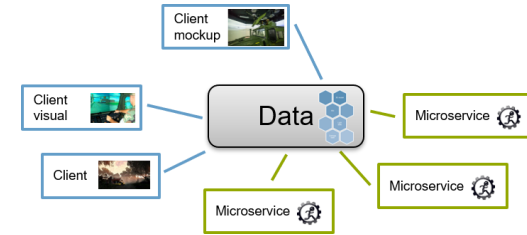
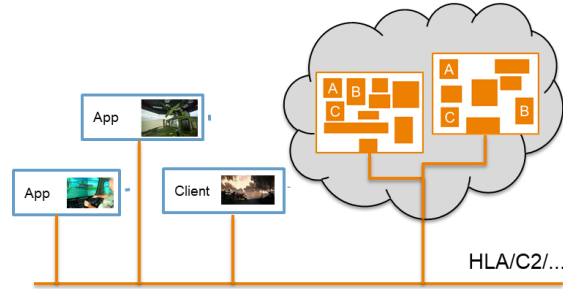
- › Cloud-based game development platforms
 - › Cloudgine: Crackdown 3 w. Microsoft Cloud
 - › SpatialOS
 - › Amazon Gamelift
 - › Coherence
 - › Unity – Connected Games services



SpatialOS



VISION



Stovepipe
systems

Systems in the
cloud

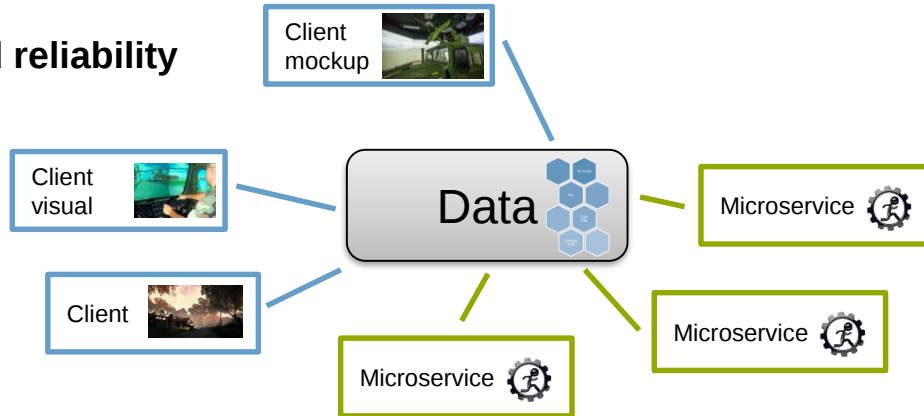
Component-
based systems
and services in
the cloud

Next-gen
architecture
using
microservices

MSaaS vision

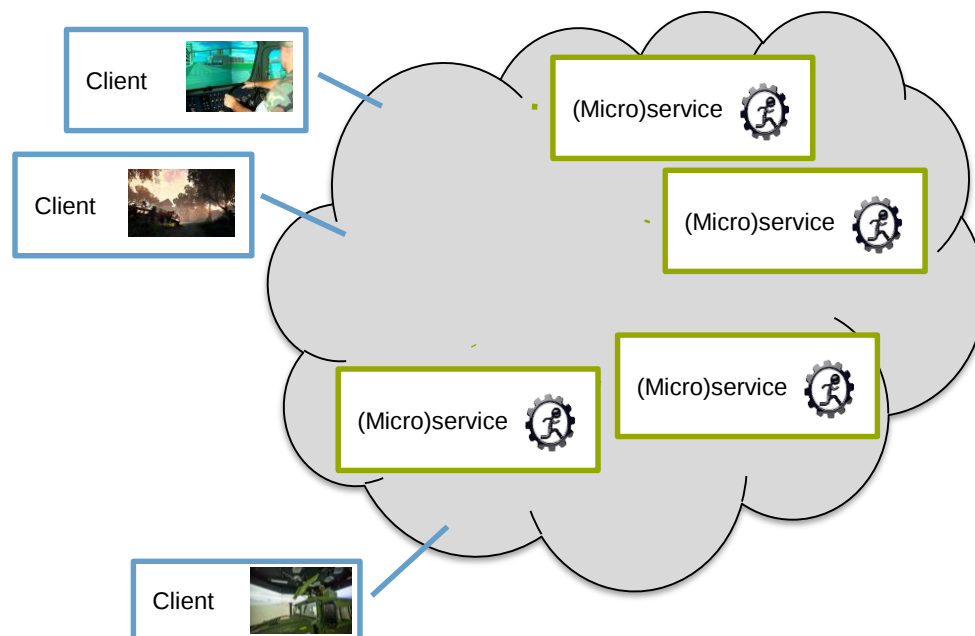
NEXT-GEN ARCHITECTURE USING MICROSERVICES

- › Everything is a **entity**
- › An entity is composed of reusable **components** (e.g. position, damage state)
- › Entities live in the cloud (persistent!)
- › **Microservices** modify components (movement model, physics simulation, ...)
- › Microservices interoperate via a **data model**
- › “Data everywhere” abstraction
- › Platform provides **load balancing, scalability and reliability**
- › **Clients** can connect to the cloud anywhere and anytime (late joining)
- › Local client (visual) is generated based on entities in view



MICROSERVICES - EXAMPLES

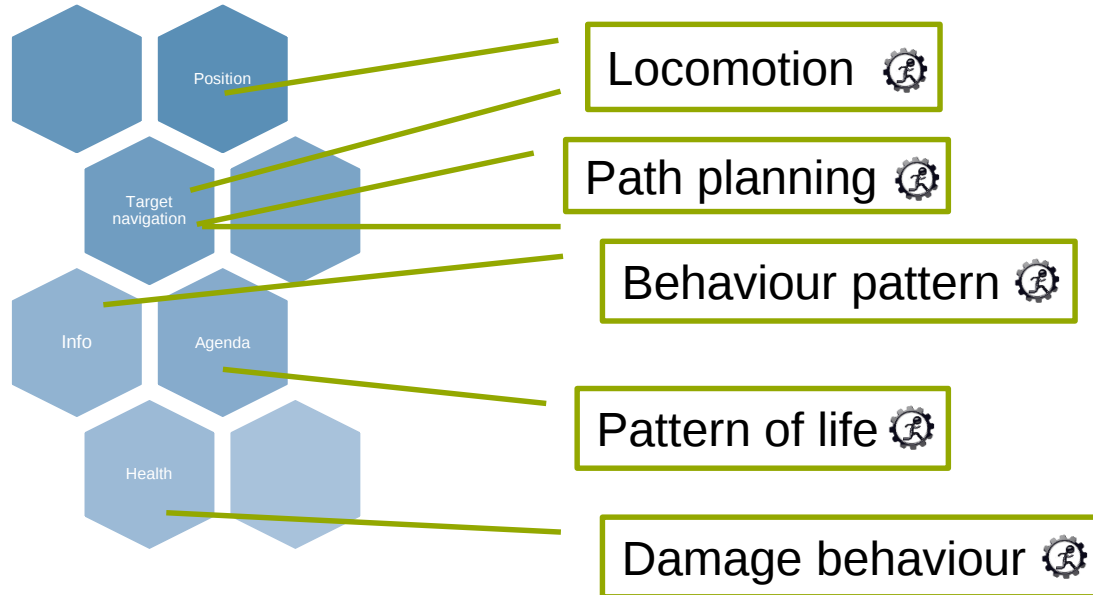
- › Weapon service (missile trajectory, ...)
- › Weapon effect service (damage)
- › Dynamic terrain service
- › Route planning service
- › Weather service
- › Data recording service
- › Performance evaluation service
- › Mediation services (HLA-C2, DIS-HLA, ...)
- › ...



ENTITY-COMPONENT-SERVICES EXAMPLES

NPC entity with components

Microservices



DEMONSTRATOR - DYNAMIC DESTRUCTION

- › Use case: fire support
- › Features:
 - › Geo-specific military training village (Altmark, Germany)
 - › Terrain deformation (craters)
 - › Object destruction
 1. Traditional model-switching
 2. Dynamic (physics) based destruction using NVIDIA Blast



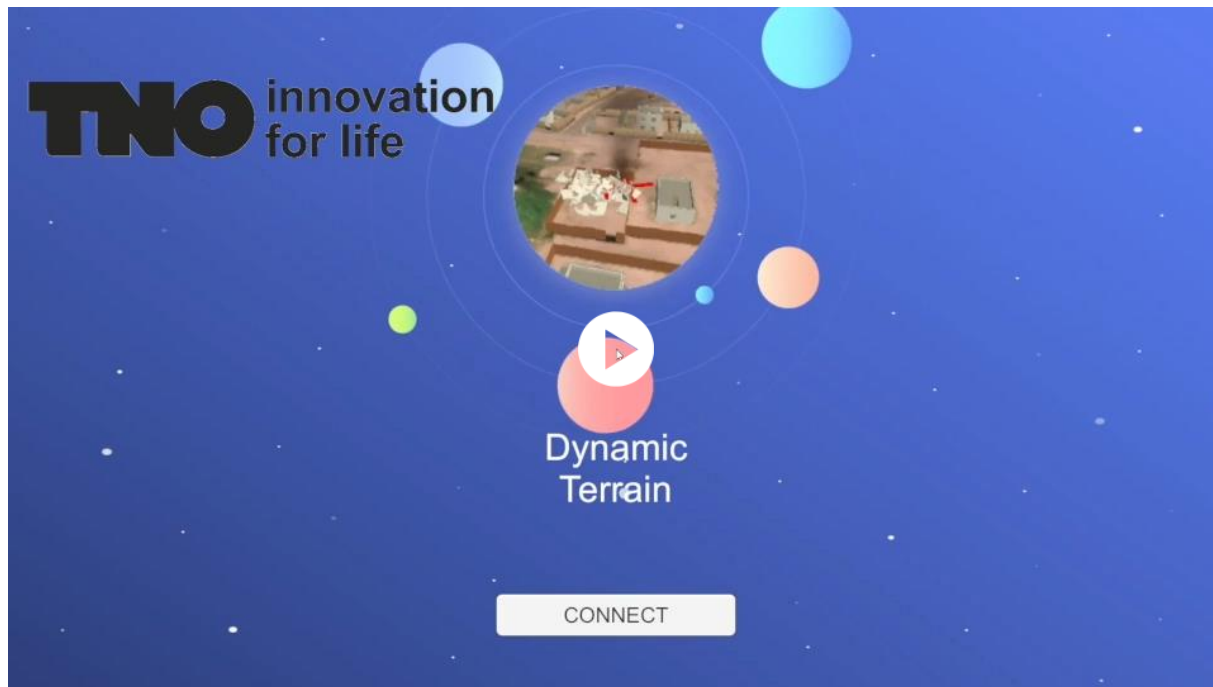
SpatialOS



DEMONSTRATOR - DYNAMIC DESTRUCTION

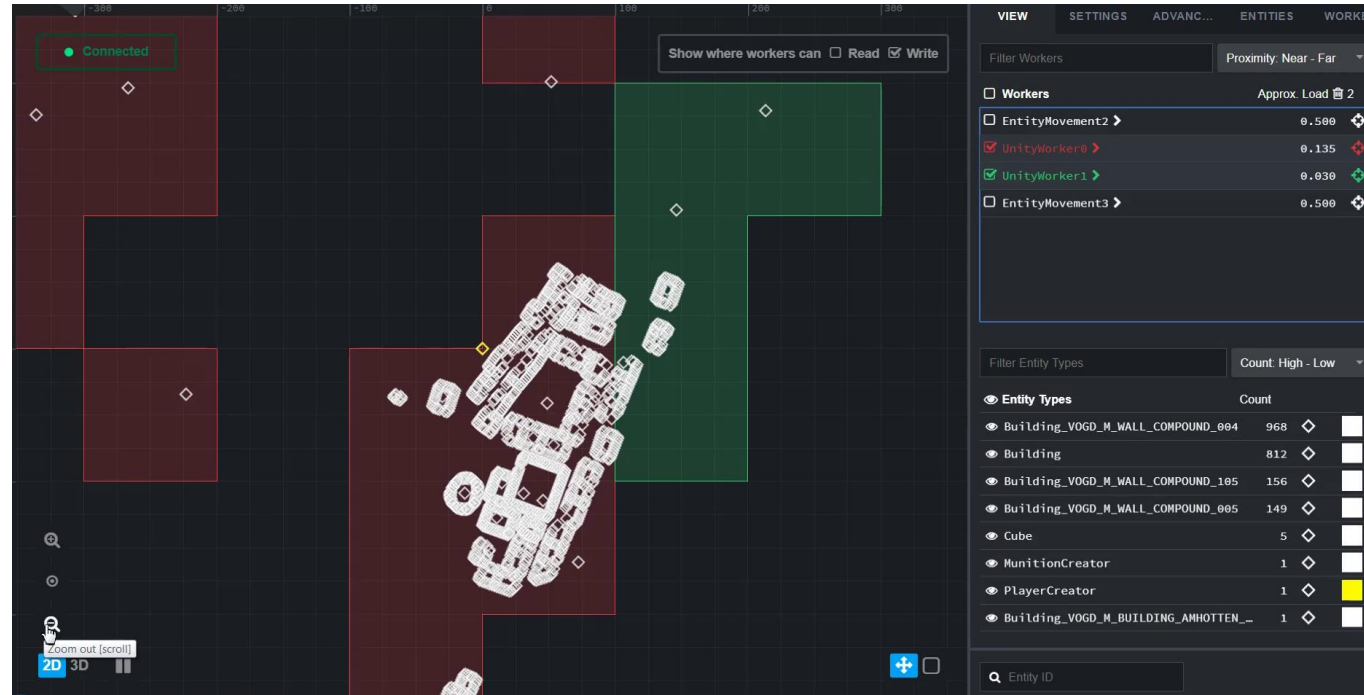


SpatialOS

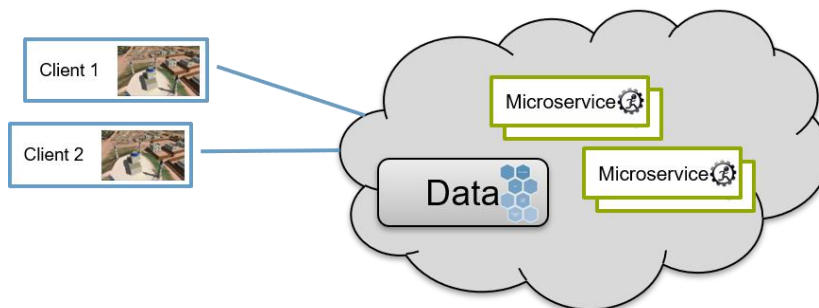


DEMONSTRATOR - DYNAMIC DESTRUCTION

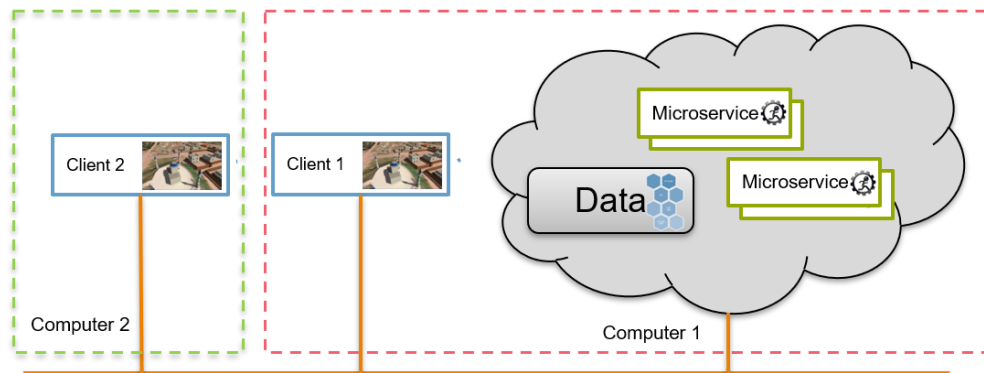
- › 2100+ entities
 - › 1273 wall segments
 - › 813 buildings
- › 215 unique 3D models



DEMONSTRATOR - DYNAMIC DESTRUCTION



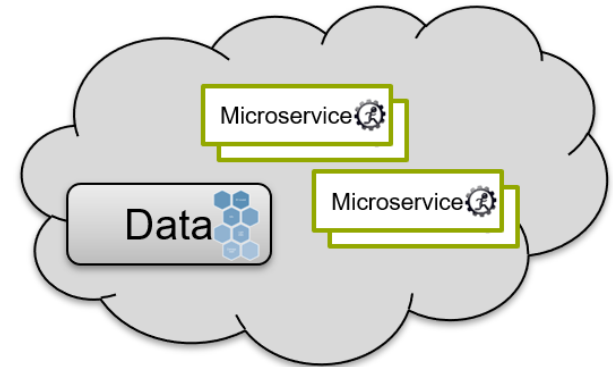
Application topology



Network topology

DEMONSTRATOR: DESTRUCTIBLE TERRAIN

- › Data model
 - › Now: used ad-hoc schema
 - › Todo: use concepts from the RPR-FOM
- › Microservices:
 - › ArtilleryFireService
 - › Munition creator (artillery)
 - › Munition detonation handler
 - › Detonation effect handler
 - › Damage assessment (physics-based destruction)
 - › Damage assessment (model switching)
 - › Entity movement (simple motion model)
 - › Player movement (stealth viewer)



DEMONSTRATOR - LEASONS LEARNED

- › SpatialOS
 - › Steep learning curve
 - › Beta (regular version updates that break the API)
 - › Game focus
 - › e.g. data read/write permissions (security / anti-cheat)
 - › performance
 - › User-based access
 - › ...
 - › Still need prediction (Dead Reckoning/ interpolation)
 - › Easy to create gateway to integrate existing HLA applications
 - › Hosting (public cloud, enterprise cloud)
 - › Not an open architecture
 - › Licensing

NEXT-GEN ARCHITECTURE USING MICROSERVICES

Advantages:

1. Centralized approach to ensure data correlation and *fair play*
2. Interoperability via a data model
3. Scalability / off-load computational work to the cloud (overcome limitations of local client)
4. Separation of concerns (experts work on specialized microservices)
5. (Visual) representation generated by client (Stealth view, Dismounted view, Flightsim, C2 view, ...)
6. Multi-resolution (different microservices, data abstraction, ...)
7. Centralized configuration, management and monitoring

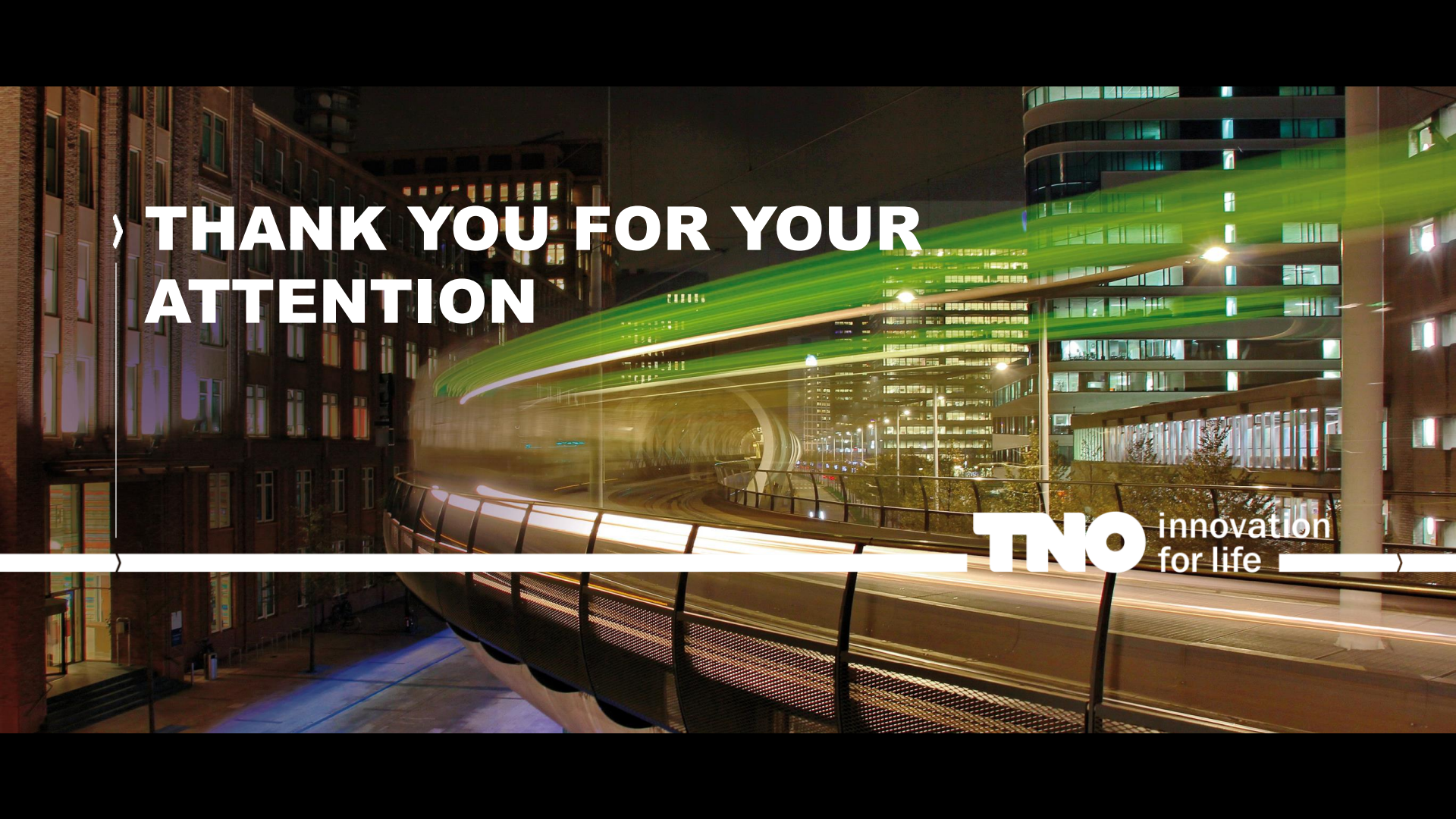
NEXT-GEN ARCHITECTURE USING MICROSERVICES

Concerns:

- › Microservice interoperability - Data model (components) needs to be standardized for reuse
- › Microservice interoperability - Need for clear description of functionality and behavior (app store)
- › Timing issues (risk of asynchronous updates)
- › How to scope a microservice (granularity)
 - › 100 Lines of Code? 1000 Lines of Code?
 - › Single concern, testable behavior

DISCUSSION

- › For our dynamic SE use case, promising architecture:
 - › Centralized data model allows for dynamic terrain correlation
 - › Performance and scalability of platform allows for complex dynamic effects (building destruction)
- › Is this the next-generation simulation architecture implementing the MSaaS vision?
- › To make this work in practice:
 - › Standardize microservice interoperability (data model, API)
 - › Support transition phase and legacy systems
 - › M&S system vendor business model has to change

A nighttime photograph of a city street featuring a modern, curved pedestrian bridge with a glass railing. The bridge is illuminated with warm lights, and there are long, horizontal light trails from vehicles and streetlights. In the background, there are multi-story buildings with lit windows. The overall scene is urban and modern.

› **THANK YOU FOR YOUR
ATTENTION**

TNO innovation
for life