Entity Component System

Ein datenorientiertes Architekturmuster für performante Spieleentwicklung

Agenda

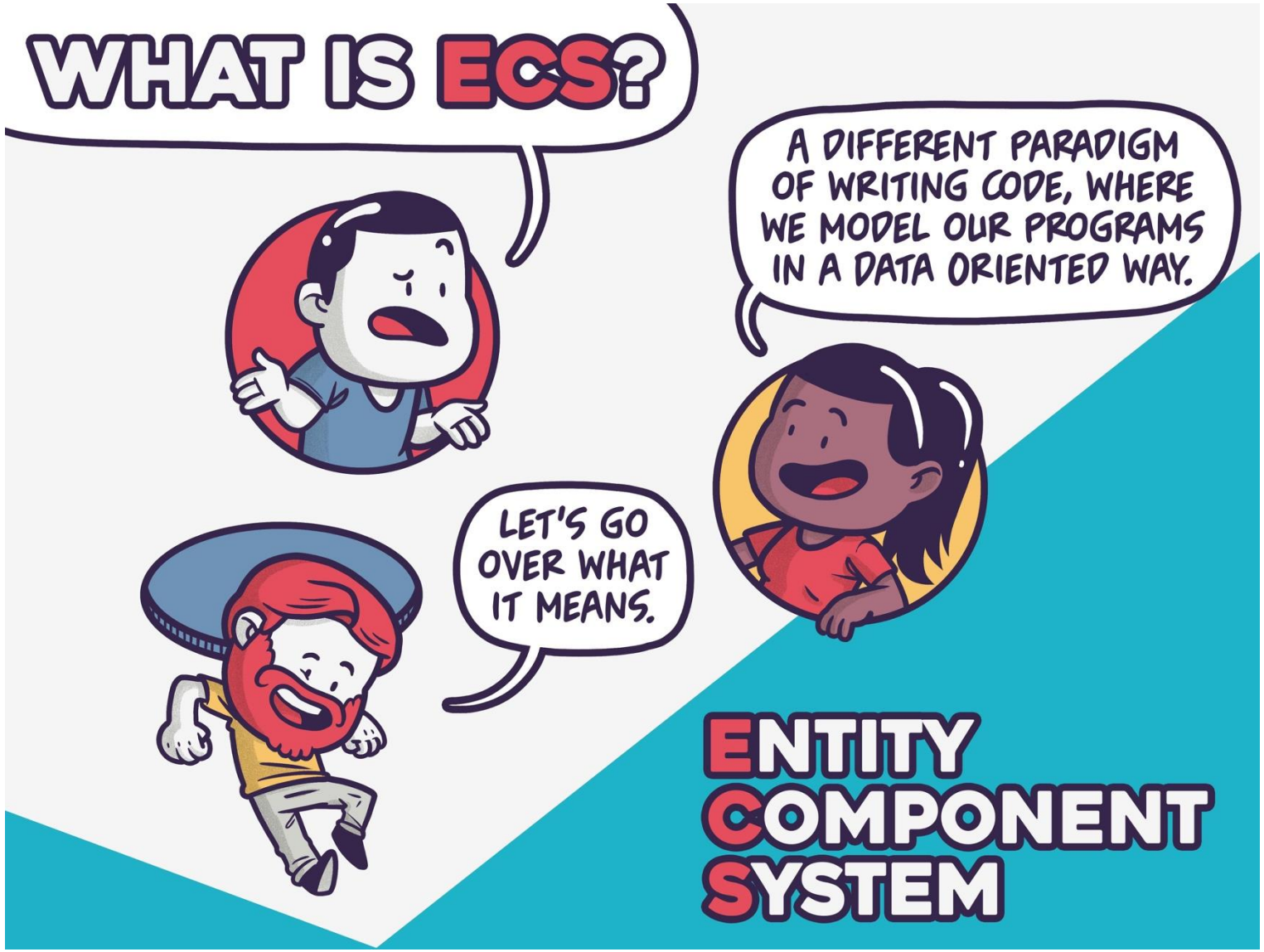
- Kennenlernen
- Was ist ein ECS
- Abgrenzung zu EC-Logik & OOP
- Warum ECS?
- Pro & Con
- Pure vs. hybrid ECS
- **Diskussion: Bedeutung & mögliche Anwendungen**
- **Hands-On: Einfaches ECS modellieren**
- Offene Diskussion und Fragen

Kennenlernen

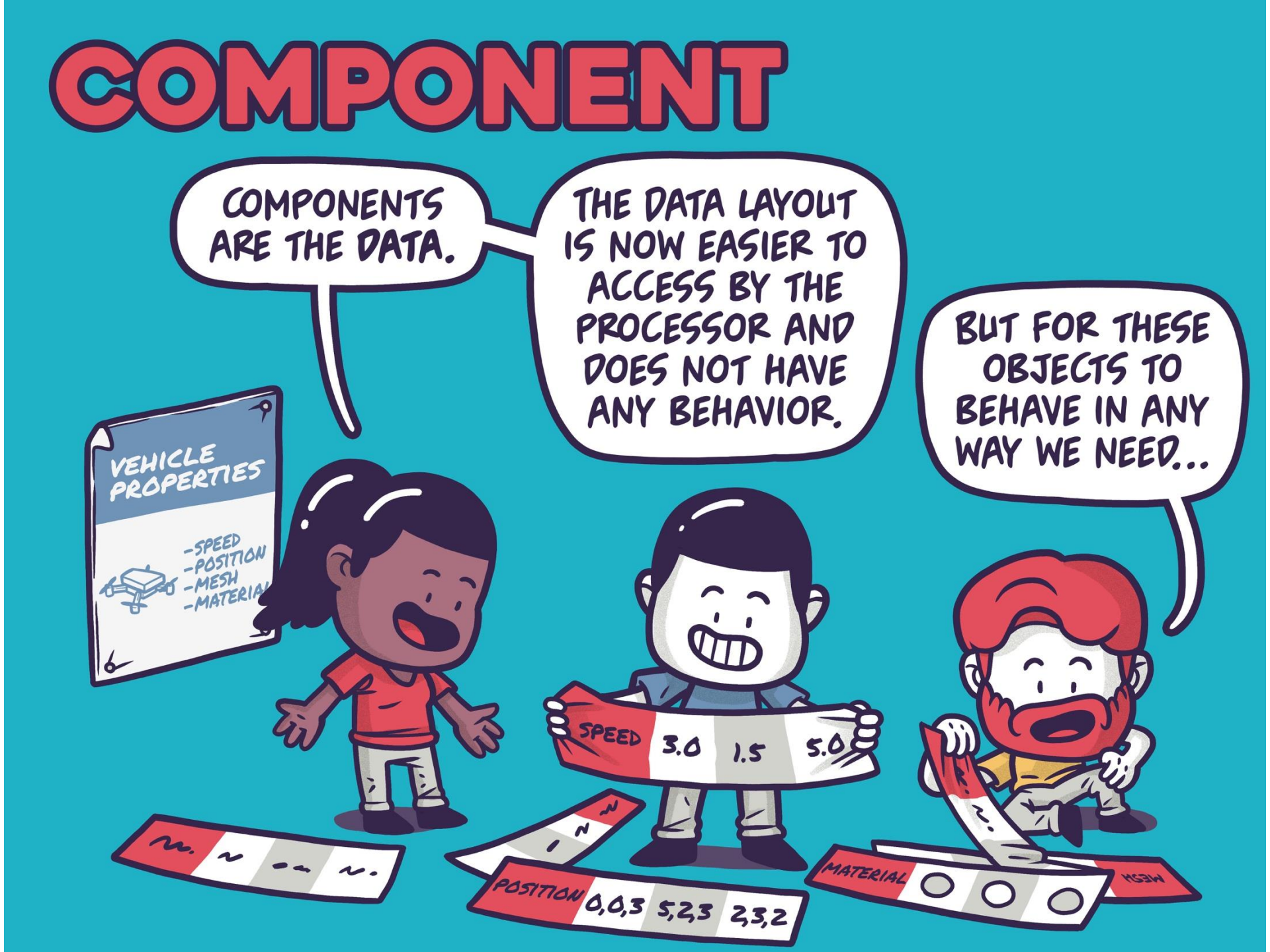
(0. Wie heißt du)

1. Was weißt du über Entity Component Systems?
2. Was weißt du über Datenorientierung?
3. Was weißt du über Architekturmuster?
4. Was weißt du über Spieleentwicklung?
5. Was führt dich in diesen Workshop?

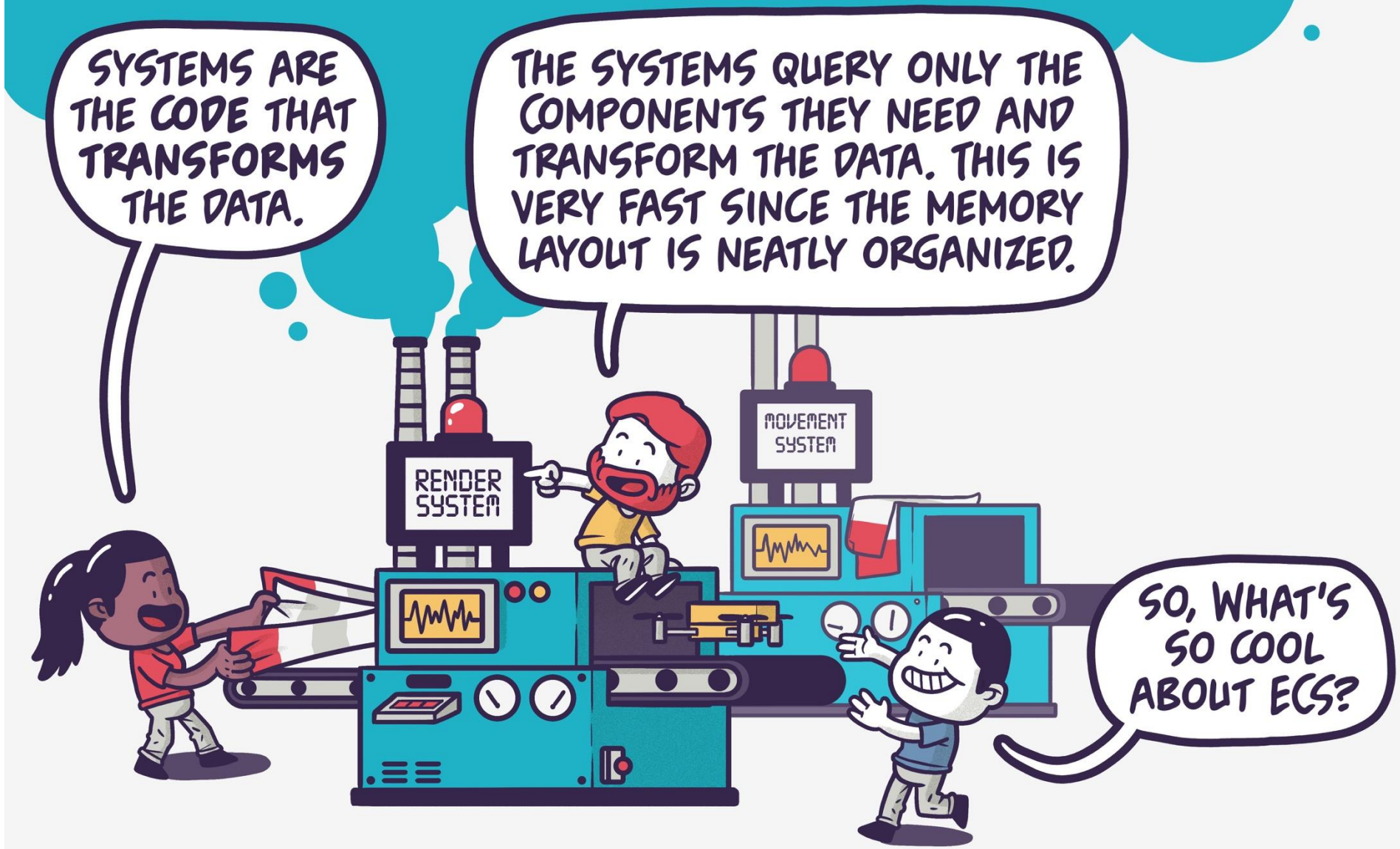
Was ist ein **E**ntity **C**omponent **S**ystem?




ENTITY
ENTITIES ARE INDICES.
YOU CAN ALSO THINK OF THEM AS ID'S.
THE ENTITIES THEMSELVES DO NOT CONTAIN ANY DATA OR DO ANYTHING.
DRONE 1
DRONE 2
DRONE N


COMPONENT
COMPONENTS ARE THE DATA.
THE DATA LAYOUT IS NOW EASIER TO ACCESS BY THE PROCESSOR AND DOES NOT HAVE ANY BEHAVIOR.
BUT FOR THESE OBJECTS TO BEHAVE IN ANY WAY WE NEED...
VEHICLE PROPERTIES
-SPEED
-POSITION
-MESH
-MATERIAL
SPEED 3.0 1.5 5.0
POSITION 0,0,3 5,2,3 2,3,2
MATERIAL
MESH

SYSTEM


SYSTEMS ARE THE CODE THAT TRANSFORMS THE DATA.
THE SYSTEMS QUERY ONLY THE COMPONENTS THEY NEED AND TRANSFORM THE DATA. THIS IS VERY FAST SINCE THE MEMORY LAYOUT IS NEATLY ORGANIZED.
RENDER SYSTEM
MOVEMENT SYSTEM
SO, WHAT'S SO COOL ABOUT ECS?

CONTENT: ARTURO NERELI GRAPHICS: GERARDO ALBA

Unity®

Überblick

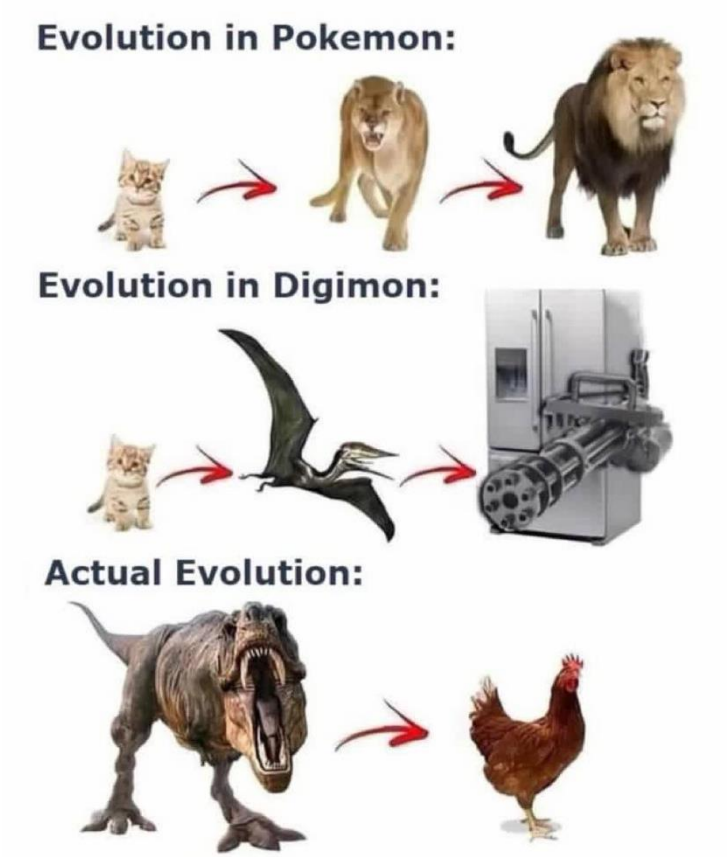
- **Entity:** Eindeutige ID, Container für Components
- **Component:** NUR Daten, KEINE Logik (struct/record)
- **System:** Reine Logik, operiert auf Components

*"Entities **are**, Components **describe**, Systems **do**"*

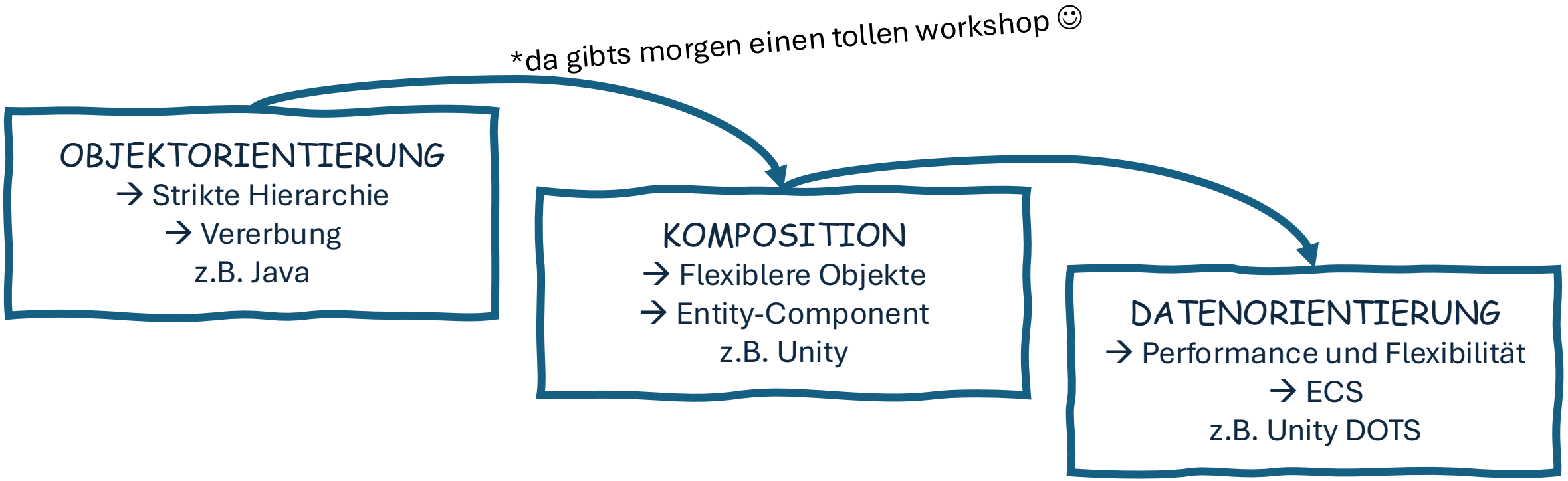
Was für Beipspiele für eine Entity-Komponenten-System-Kombi fallen DIR ein?

Abgrenzung zu EC-Logik & OOP

Wo kommen wir her?



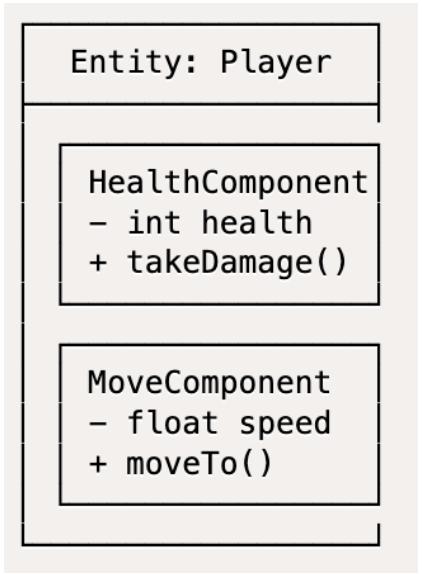
Wo kommen wir her?



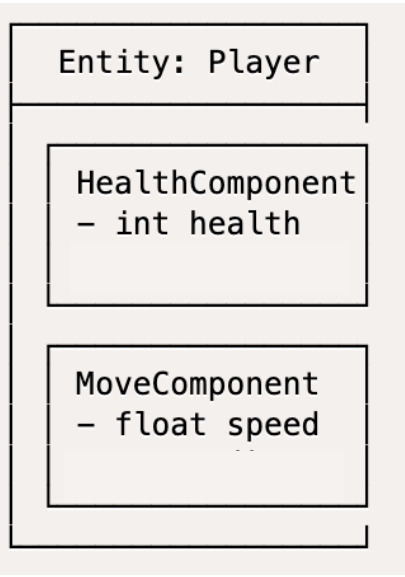
Enitity Component vs. Entity Component System

⚠️ **Wichtige Unterscheidung!**

	Entity Component	Entity Component System
Entity	Container für Komponenten	Container für Komponenten (Index, dem Komponenten zugewiesen werden)
Component	Enthält Daten **und** Logik	Enthält Daten
System	Kein direkter Teil aber oft einige größere Systeme im Hintergrung (z.B. Schwerkraft, Physics-Engine..)	Enthält Logik, die auf Kompoonenten operiert
Beispiel	Unity MonoBehaviour	Unity DOTS, Bevy, Flecs



vs




```
MovementSystem
- for each entity with
  (Position, Velocity):
  pos += vel * deltaTime
```

Warum ECS?

Einsatzzwecke

- 🎮 **Spiele mit vielen Objekten** (Tausende NPCs, Bullets, Partikel)
- 🌊 **Simulationen** (Physik, Flüssigkeiten, Menschenmengen)
- ✨ **Partikelsysteme** (Effekte, Feuerwerk, Rauch)
- 🏙️ **Städtebausimulationen** (z.B. Cities: Skylines II nutzt Unity DOTS)

Performance-Vorteil

Verarbeitung in Component-Batches

```
// 🚀 1000 Enemies updaten = 3 System-Runs über Arrays
movementSystem.update(positions[], velocities[]);  // 1x für ALLE
physicsSystem.update(positions[], colliders[]);    // 1x für ALLE
renderSystem.update(positions[], sprites[]);       // 1x für ALLE
```

Metrik	Entity-Component	ECS	Grund
Cache Hits	~30%	~95%	Sequentielle vs. Random Access
Function Calls	3000 virtual	3 direct	Batch vs. Individual
CPU Prediction	❌ Unmöglich	✅ Trivial	Klare Zugriffsmuster
Parallelisierung	⚠️ Schwierig	✅ Einfach	Systems sind unabhängig

Pro & Con


steinbergdrawscartoons
SHOULD I MAKE A PROS AND CONS LIST?
PROS
CONS
Steinberg

Vor- und Nachteile gegenüber ECs

Vorteile	Nachteile
Performance: Cache-Effizienz durch Data Locality **Parallelisierung:** Systems können parallel laufen **Flexibilität:** Neue Kombinationen zur Laufzeit **Testbarkeit:** Systems isoliert testbar	**Komplexität:** Höhere initiale Lernkurve **Indirektion:** Entity-Beziehungen schwerer nachvollziehbar **Overhead:** Für kleine Projekte oft überdimensioniert **Debugging:** Schwieriger zu debuggen als OOP

Wann ECS verwenden?

- ✓ Viele gleichartige Objekte (>1000)
- ✓ Performance-kritisch
- ✓ Häufige Komposition zur Laufzeit
- ✗ Kleine Prototypen
- ✗ Wenige, komplexe Objekte

In der Praxis: Pure vs Hybrid

Pure:

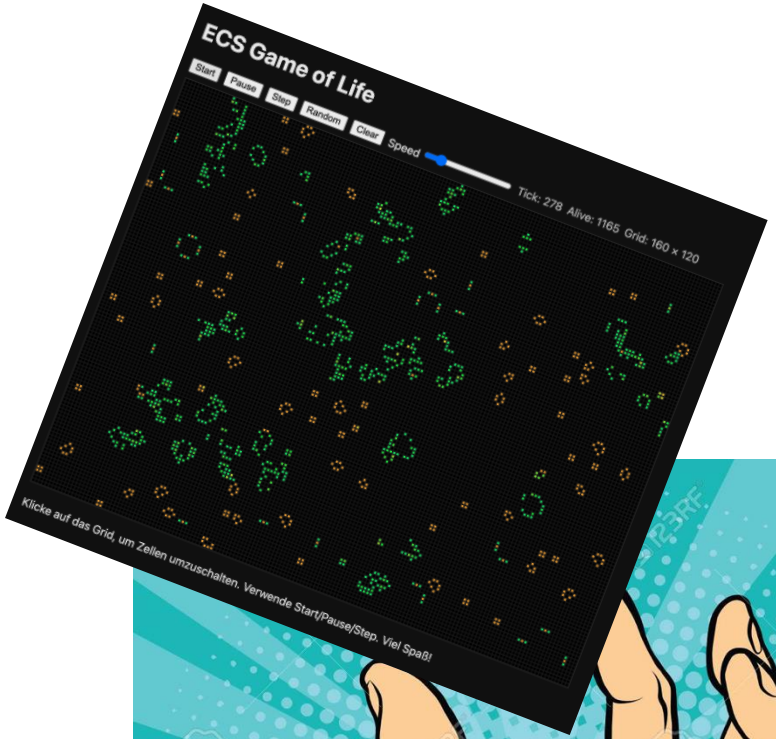
- Strikte Trennung von Daten und Logik
- Entitäten sind nur IDs
- Komponenten sind reine Daten
- Beispiele: EnTT (C++), Bevy ECS (Rust), Unity DOTS

Hybrid:

- Oft eine Mischung. Komponenten können doch etwas Logik enthalten oder Entitäten sind richtige Objekte.
- Oft einfacher für den Einstieg und für Prototyping. Performance ist nicht immer das Hauptziel.

Was für Einsatszwecke außerhalb der Spieleentwicklung fallen DIR ein?

https://github.com/huntredtimes/ECS_GameOfLife



Hands on

Develop Convoys Game of Life with an ECS!

War das ein pures ECS?

Was nimmst DU mit?


Thank you!