“THE COMPUTING SCIENTIST’S MAIN CHALLENGE IS NOT TO GET CONFUSED BY THE COMPLEXITIES OF THEIR OWN MAKING. ”

E. W. Dijkstra

ENTITY-COMPONENT SYSTEMS + YOU

(THEY'RE NOT JUST FOR GAMES ANYMORE)

HELLO!

- Paris (@parisba)
  - Game developer, PhD Computing, Author
- Tim (@The_McJones)
  - Game developer, PhD Computing, Author
- Mars (@TheMartianLife)
  - Software developer, Research & Teaching Assistant, Upcoming Author


Learning Swift
Practical Artificial Intelligence with Swift
Head First Swift
A Brain-Friendly Guide
THE KERBAL PLAYER'S GUIDE
Learning Cocoa with Objective-C
Unity Game Development Cookbook
iOS Swift Game Development Cookbook
Jonathon Manning & Paris Buttfield-Addison
Mobile Game Development with Unity
Swift Development with Cocoa
Swift Development for the Apple Watch
SECRET LAB
AUC
NIGHT IN THE WOODS
acs
UNIVERSITY of TASMANIA

ECS?

IF YOU GOOGLE ECS, YOU'LL PROBABLY GET UNITY, OR AMAZON.

OR SOMETHING ENTIRELY UNRELATED.

Sorry.

THIS IS NOT A GAMES TALK!

Mostly.

BEFORE WE START..

Is anyone using ECS, or working in games?

OVERVIEW

» What is ECS?

» History Lesson

» ECS Implementations

» What ECS is good for

» What ECS is not good for

» Summary

WHAT IS ECS?

ENTITY COMPONENT SYSTEMS

THERE'S THE SHORT ANSWER, AND THE LONG ANSWER...


ECS?!


STRICT SEPARATION
BETWEEN DATA AND
LOGIC

OH, ECS...

THE IMPETUS

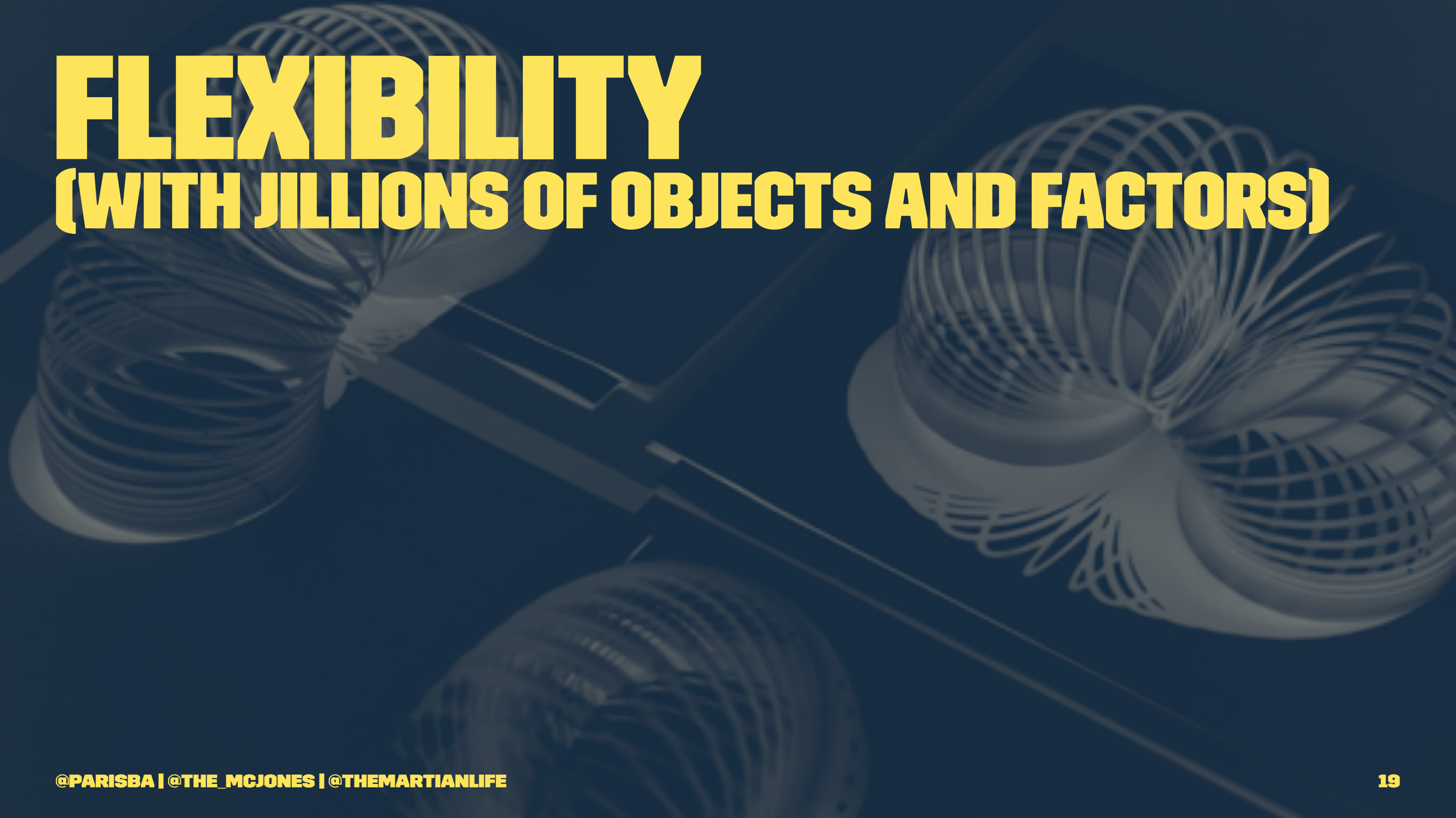
Games really only have two needs:

» performance

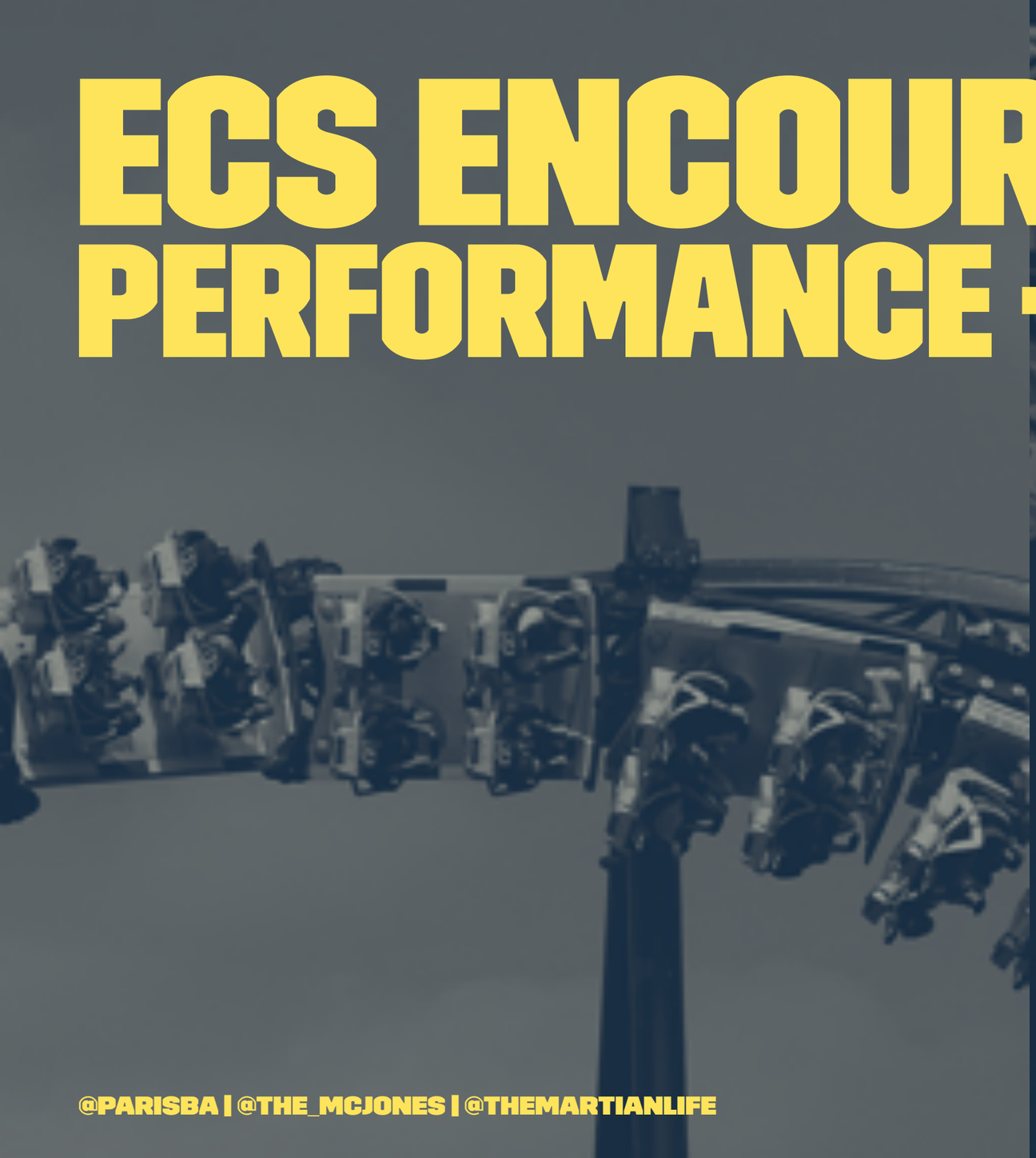
» flexibility

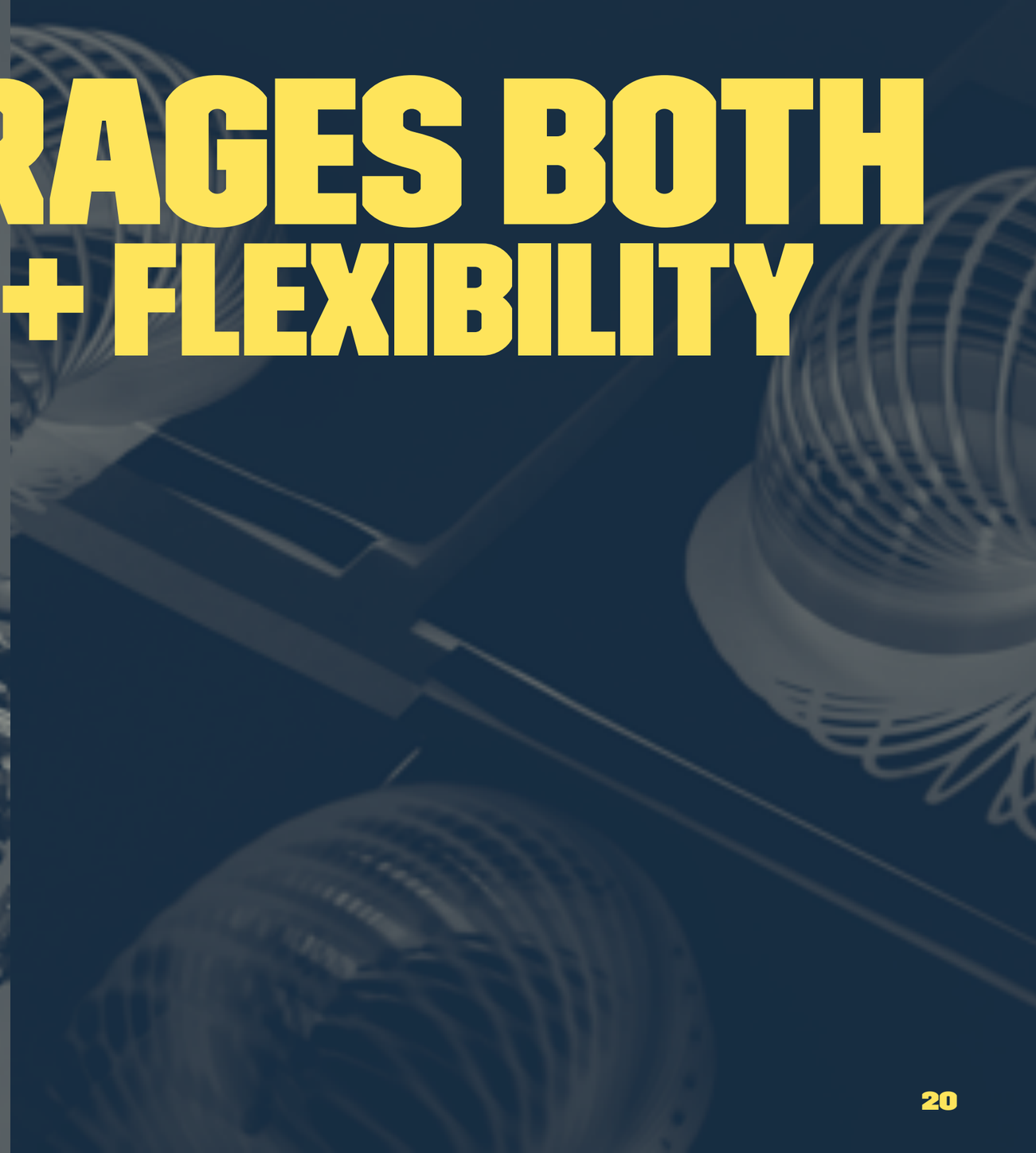
PERFORMANCE
(YOU'VE ONLY GOT ~16.6 MILLISECONDS)

FLEXIBILITY
(WITH JILLIONS OF OBJECTS AND FACTORS)



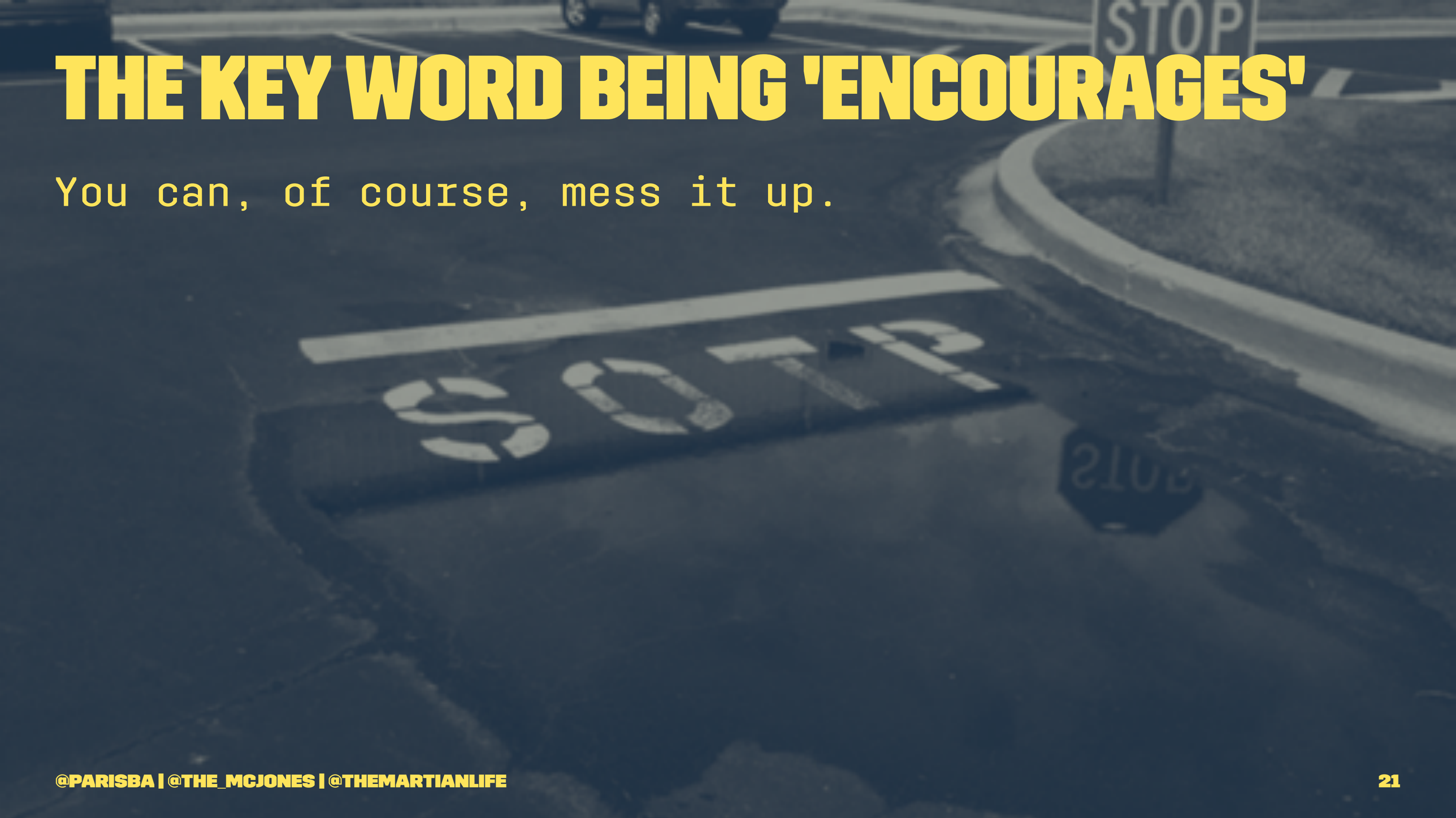
ECS ENCOURAGES BOTH PERFORMANCE + FLEXIBILITY





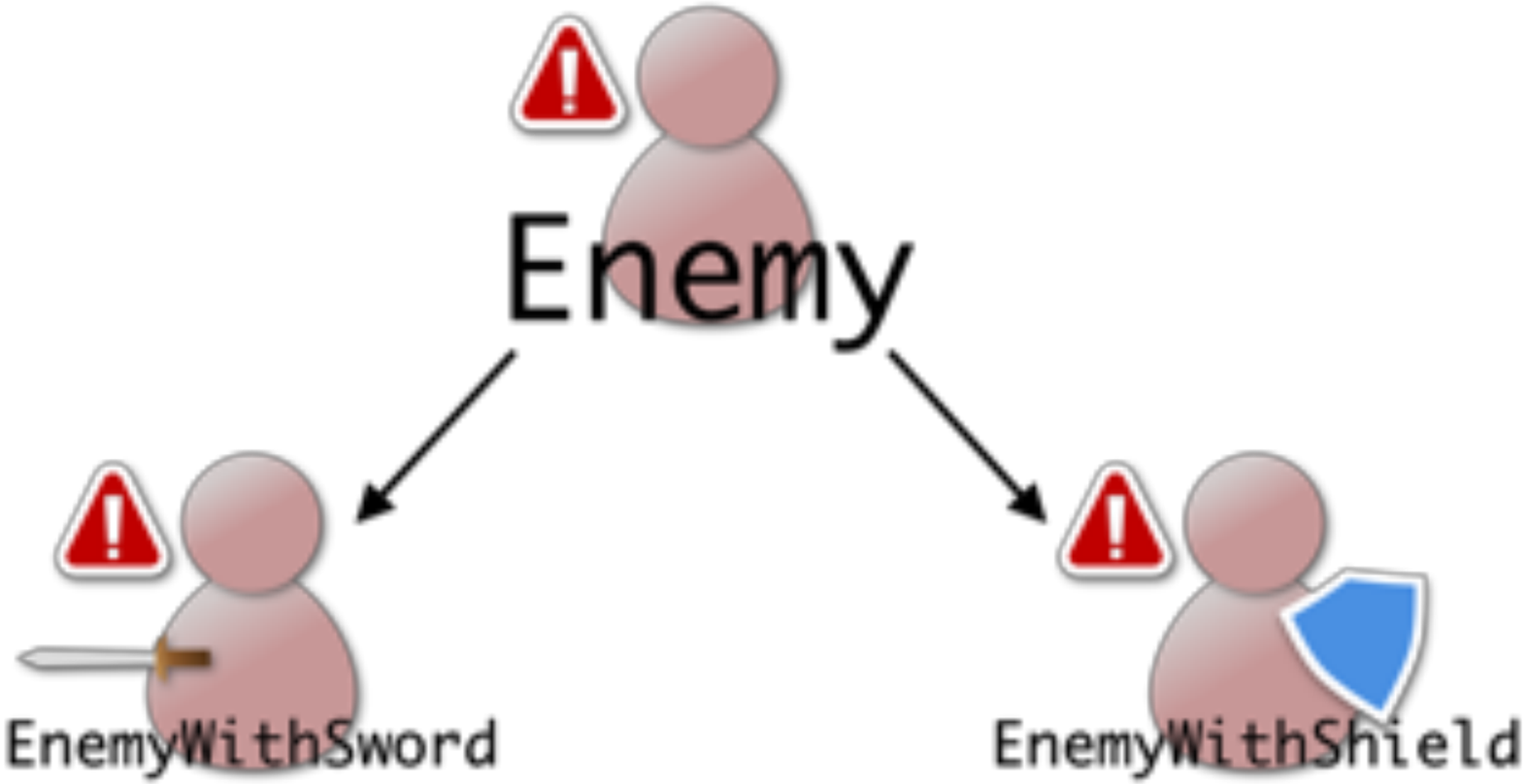
THE KEY WORD BEING 'ENCOURAGES'

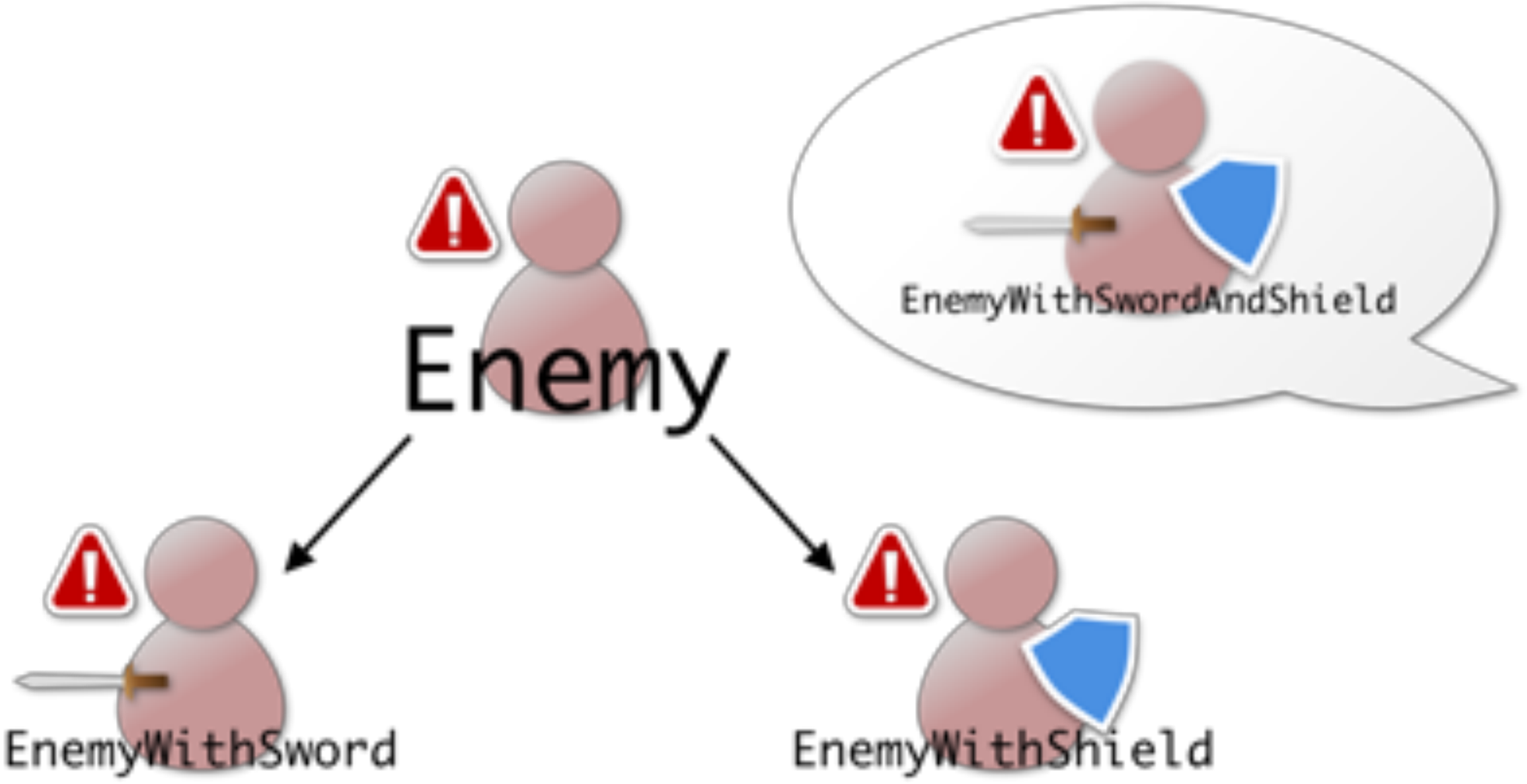
You can, of course, mess it up.

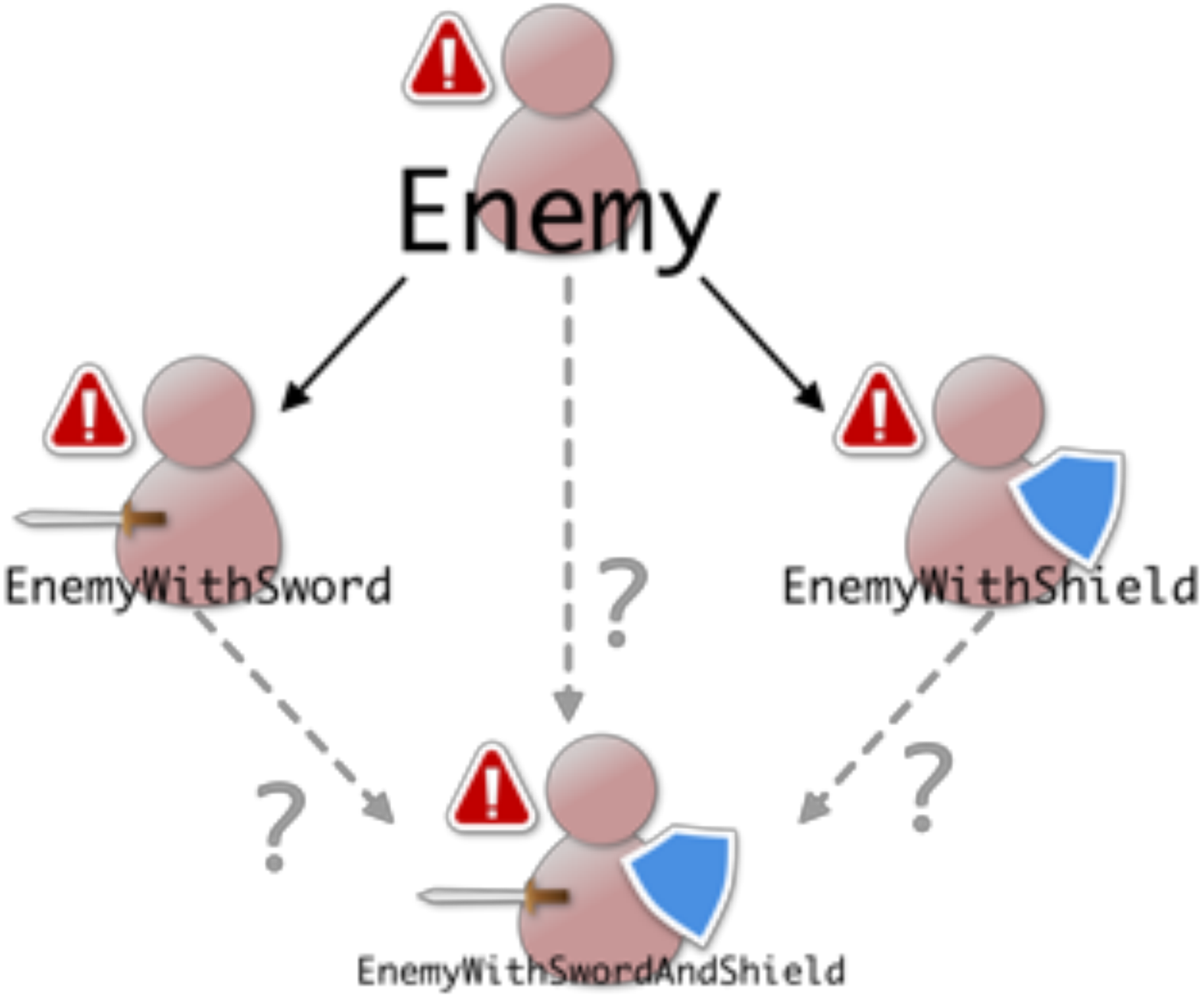


SO WHAT?

Everything is fine already!


Enemy
EnemyWithSword
EnemyWithShield


Enemy
EnemyWithSwordAndShield
EnemyWithSword
EnemyWithShield


Enemy
EnemyWithSword
EnemyWithShield
?
?
?
EnemyWithSwordAndShield


IT'S A BIT MESSY.

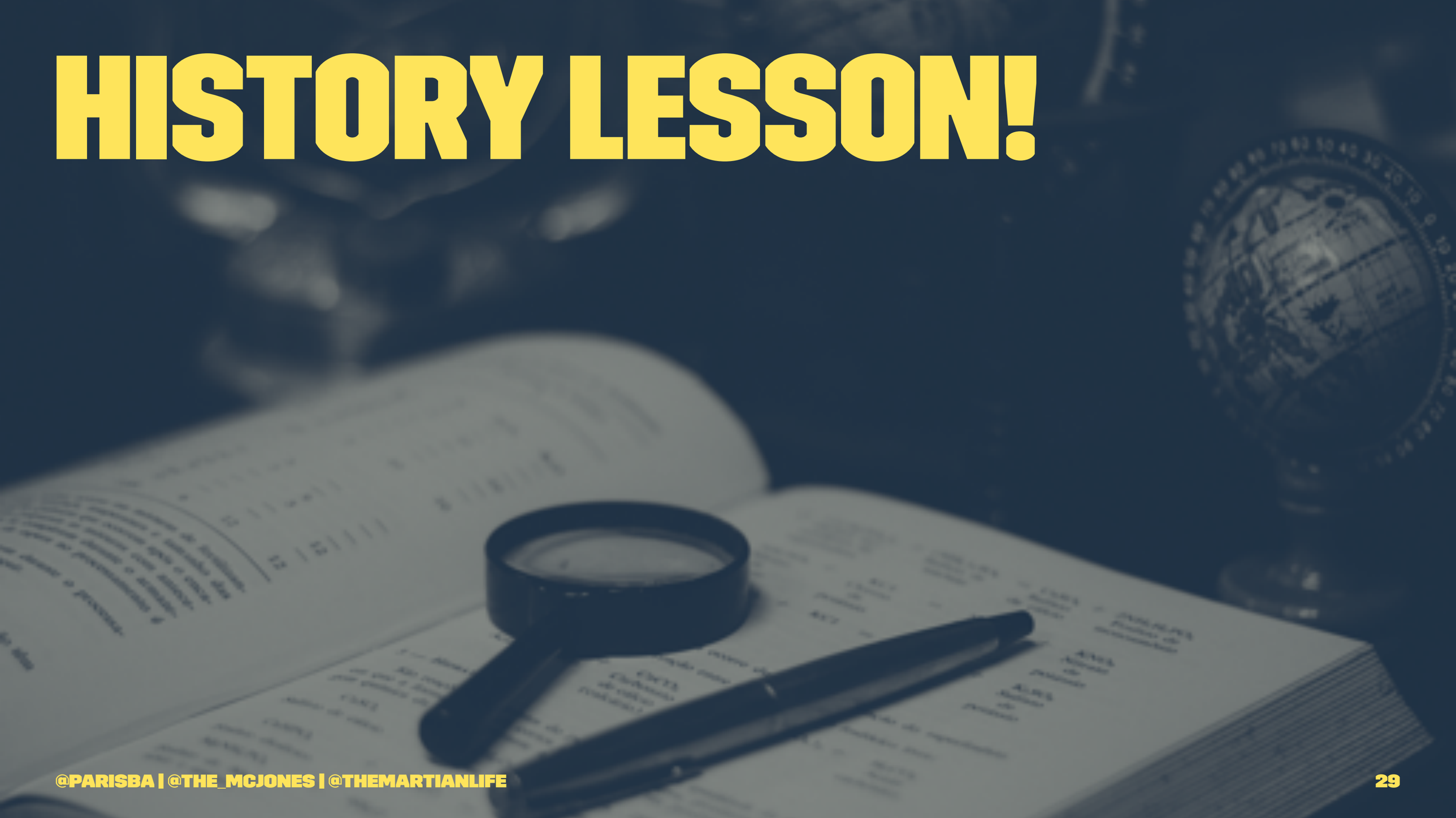
GOD OBJECTS

2018 RustConf

How many accessors could you possibly need?

```
class Player :
  public virtual EntityEntity,
  public virtual LivingEntity,
  public virtual CharacterEntity,
  public virtual DamageableEntity,
  public virtual MonsterEntity,
  public virtual HumanEntity,
  public virtual MigratedEntity,
  public virtual RewindEntity {

public:
  Player(PlayerConfigPtr config, Void void = Void());
  Player(PlayerConfigPtr config, Json const& diskStore);
  Player(PlayerConfigPtr config, ByteArray const& netStore);

  ClientContextPtr clientContext() const;
  void setClientContext(ClientContextPtr clientContext);

  StationsPtr stations() const;
  void setStations(StationsPtr stations);

  QuestManagerPtr questManager() const;

  Json diskStore();
  ByteArray netStore();

  EntityType entityType() const override;

  void init(World* world, EntityId entityId, EntityMode mode) override;
  void uninit() override;

  Vec2F position() const override;
  Vec2F velocity() const override;

  Vec2F mouthPosition() const override;
  Vec2F feetOffset() const;
  Vec2F headArmorOffset() const;
  Vec2F chestArmorOffset() const;
  Vec2F legsArmorOffset() const;
  Vec2F backArmorOffset() const;
```

HISTORY LESSON!



EARLY 'ECS'


OPERATION
FLASHPOINT
DRAGON RISING
18

ADAM MARTIN'S ECS

» Entities have IDs

» Components have data

» Systems have logic

LET'S LOOK AT THESE ONE BY ONE...

ENTITY

» an object in your game, uh, software

» anything, really, a player, an enemy, anything

» an identifier, no logic

ENTITY

Two excellent descriptions of an entity (in games):

» everything but the terrain

» assemblages

ENTITIES ARE BORING.

Just a thing to reference.

ISN'T THIS JUST A PRIMARY KEY?

YES, YES IT IS.

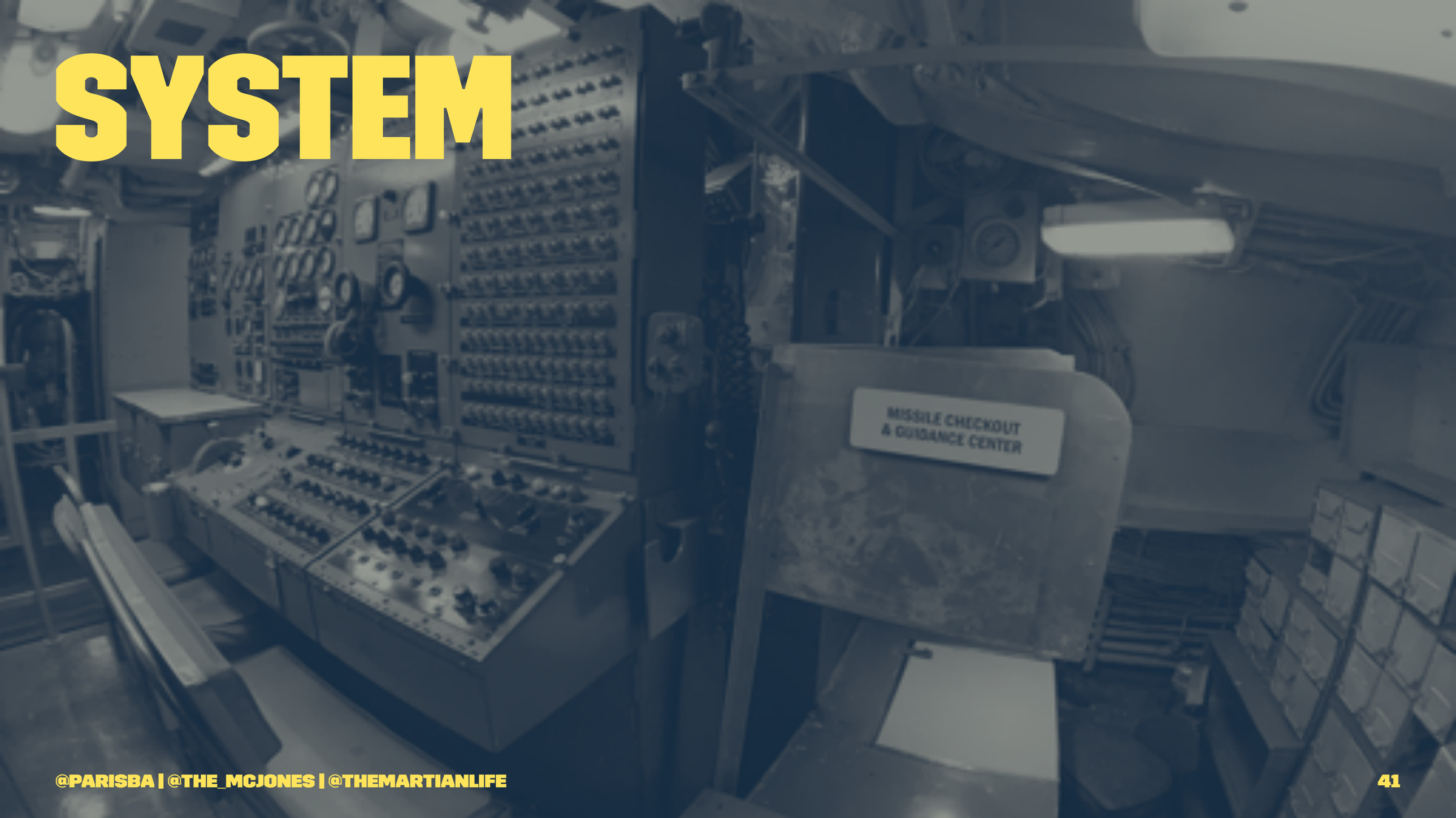
COMPONENTS

» it has some data

COMPONENTS
ALSO PRETTY BORING.

A flag that may also hold raw data
for one aspect of a thing.

SYSTEM



SYSTEMS ARE THE INTERESTING BIT.

Logic that runs continuously and performs actions on every Entity that has a Component of the aspect the System is designed to work with.

ECS OR EC?

ECS IN USE TODAY

IT'S VERY POPULAR. IN GAMES.

unity

TO DO THIS...

WE'RE GONNA TALK ABOUT GAMES. SORRY.

GAMES ARE VERY DYNAMIC

GAME WORLDS ARE BUILT OUT OF ENTITIES

A TREE IS DIFFERENT FROM A HUMAN WHO IS DIFFERENT FROM AN ALIEN WHO IS DIFFERENT FROM...


GAMES ARE IN
CONSTANT
DEVELOPMENT FLUX

The Strange Log
@TheStrangeLog

Eyespiders can squeeze through doors like vermin can.

12:14 PM · Sep 19, 2018 · TweetDeck

The Strange Log
@TheStrangeLog

Warm laundry should no longer kill your Sims.

10:33 AM · Mar 9, 2018 · TweetDeck

GAMES ARE MADE UP OF MOVING PARTS

BUT...

THERE IS COMMON POOL OF ASPECTS

» Position data

» Visuals data

» Animation data

» Sound data

» Gameplay data

WHAT A GAME SEEMS LIKE

1. Player presses walk forward
2. Move the player forward
3. Check if enemy can see the player avatar
4. Move towards the player
5. If in range swing sword
6. Deal damage to the player

WHAT A GAME IS ACTUALLY DOING

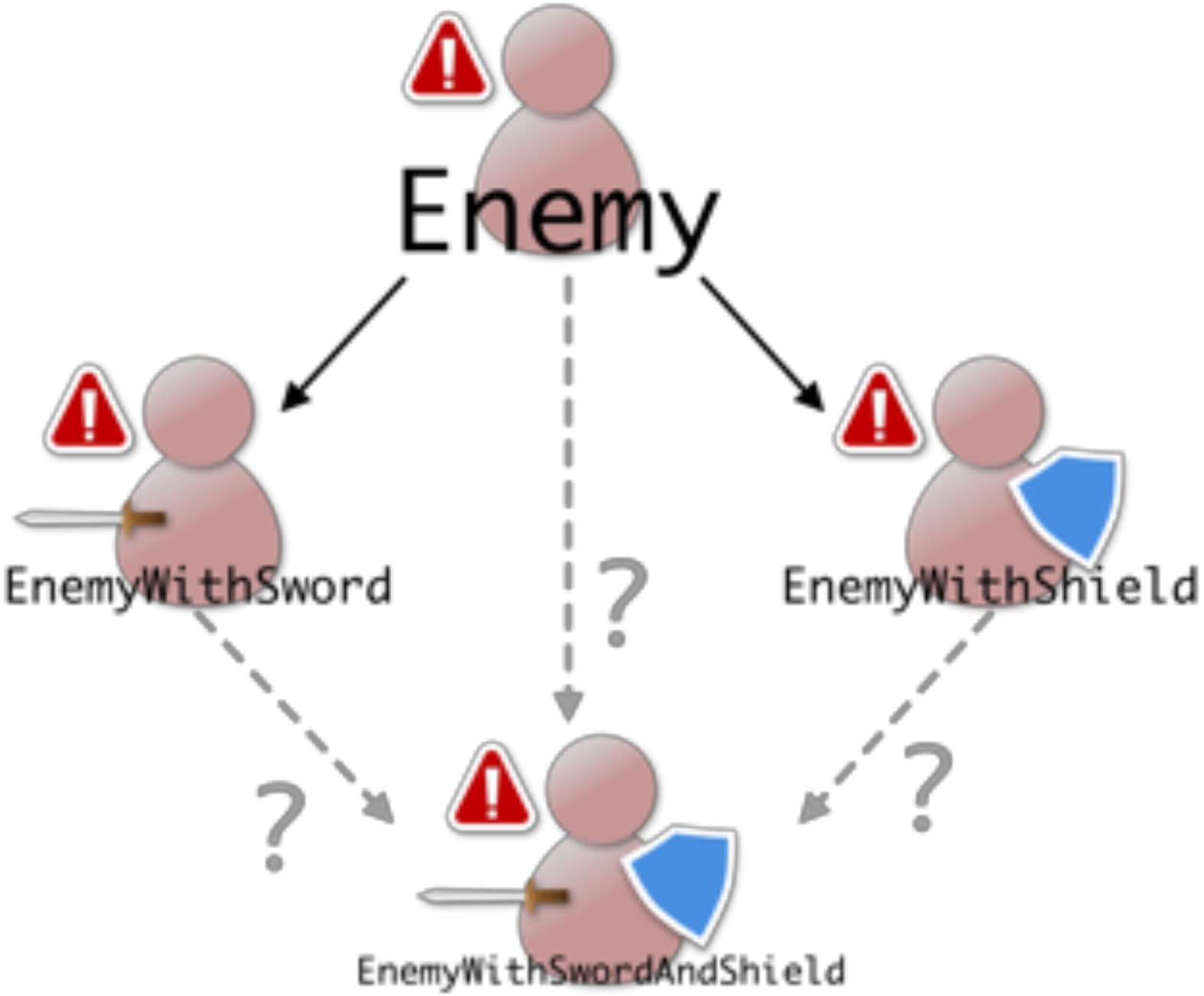
1. Receive user input
2. Raycast from enemy to player avatar
3. Update avatar and enemy position
4. Deal damage
5. Update animations
6. Render frame

SEPARATING DATA FROM BEHAVIOUR

OBJECTS?


OBJECTS AND (RIGID,
MOSTLY) CLASS
HIERARCHIES

COMPLEX HIERARCHIES


Enemy
EnemyWithSword
EnemyWithShield
?
?
?
EnemyWithSwordAndShield

COMPOSITION, NOT INHERITANCE

COMPOSING ENTITIES

COMPOSING ENTITIES

Benefits:

1. Easy to add new entities.
2. Easy to change entities.
3. Performant!

EnemyWithSword

- <enemy_ behaviours>
- sword
- …

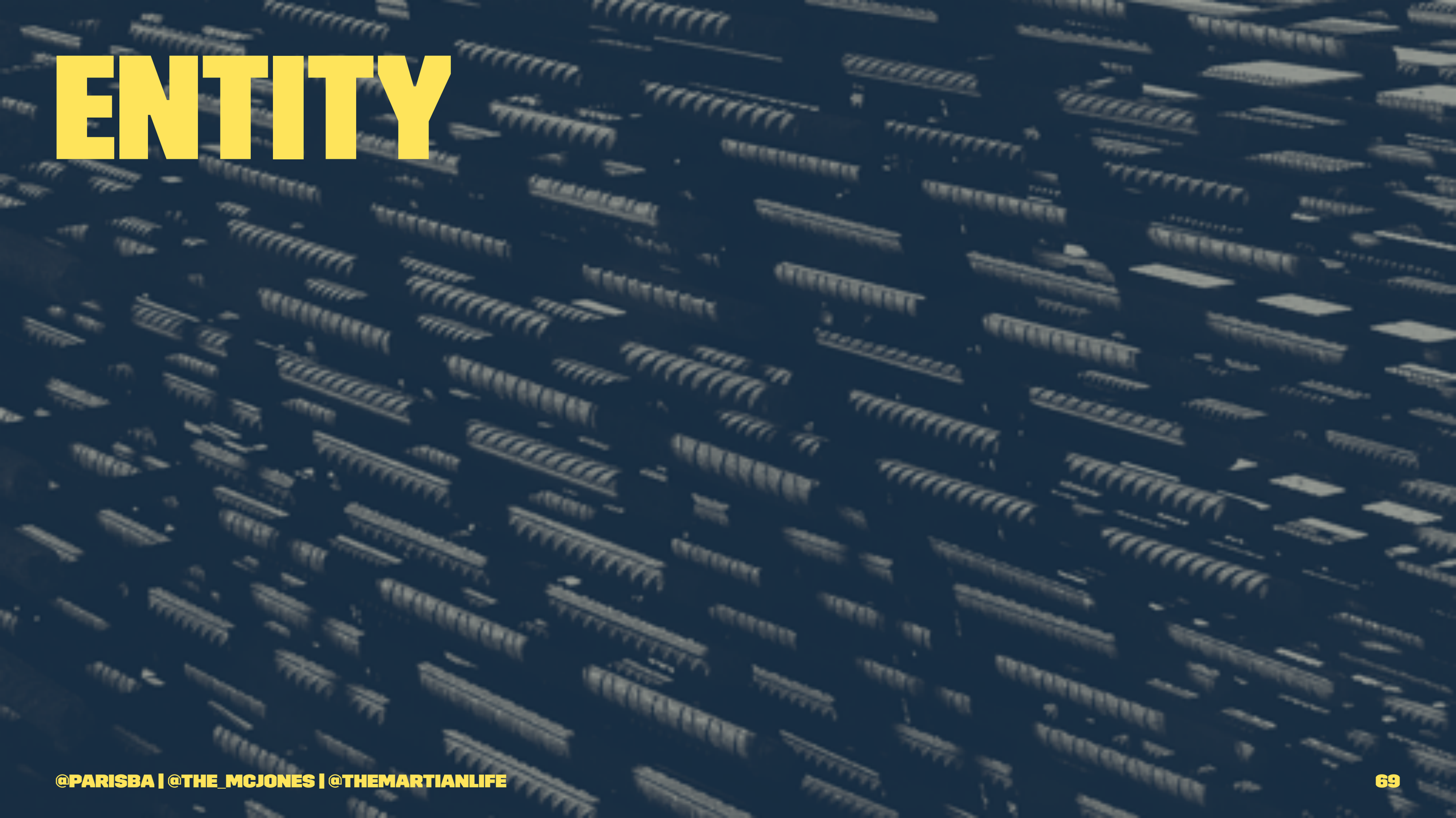
EnemyWithShield

- <enemy_ behaviours>
- shield
- …

EnemyWithSwordAndShield

- <enemy_ behaviours>
- sword
- shield
- …

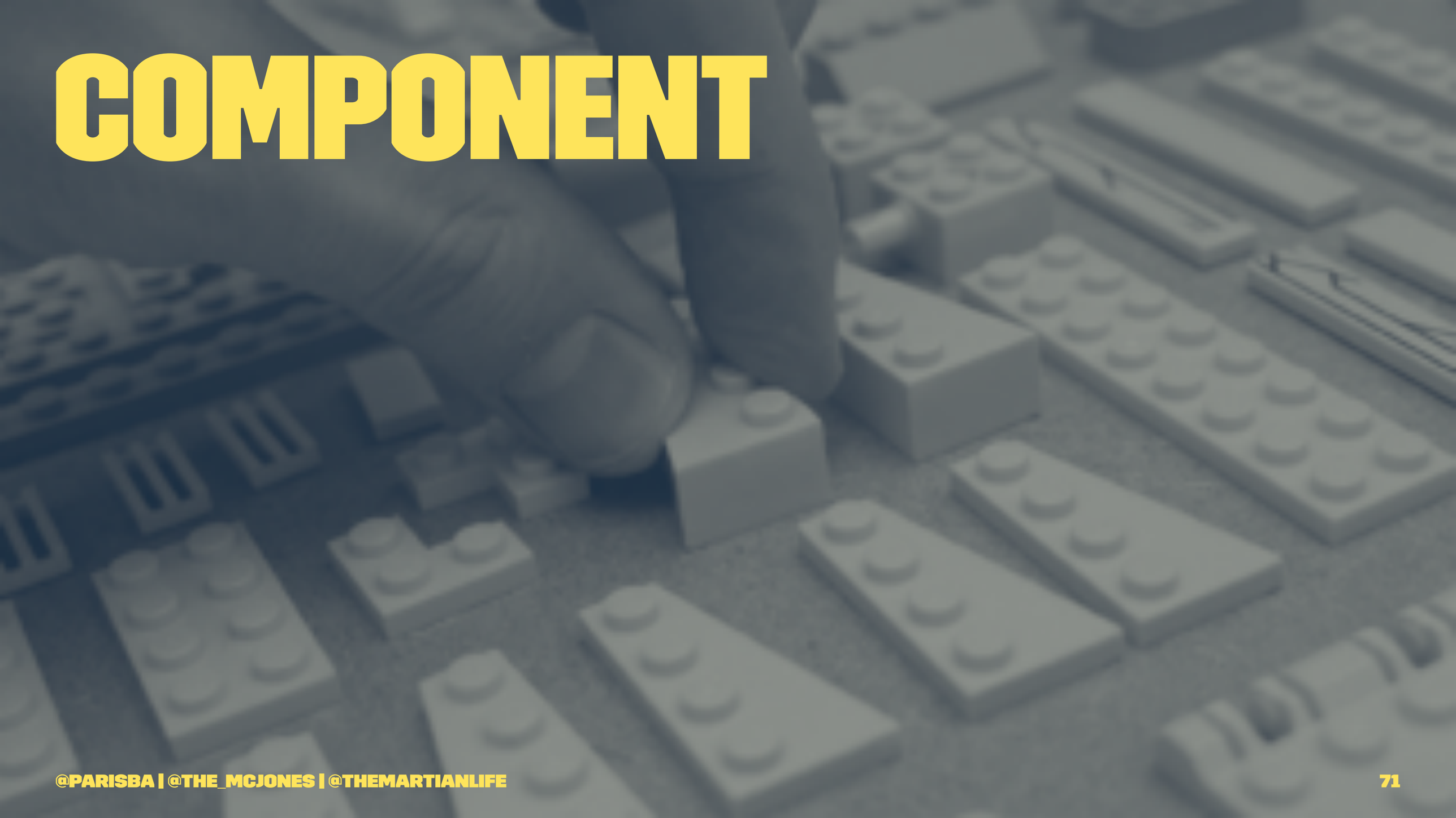
ECS AND GAMES, NOW


ENTITY

ENTITY

Examples:

- Crate
- Ball
- Enemy
- Player
- Traffic light


COMPONENT

COMPONENT EXAMPLES

» position (an x and a y)

» velocity

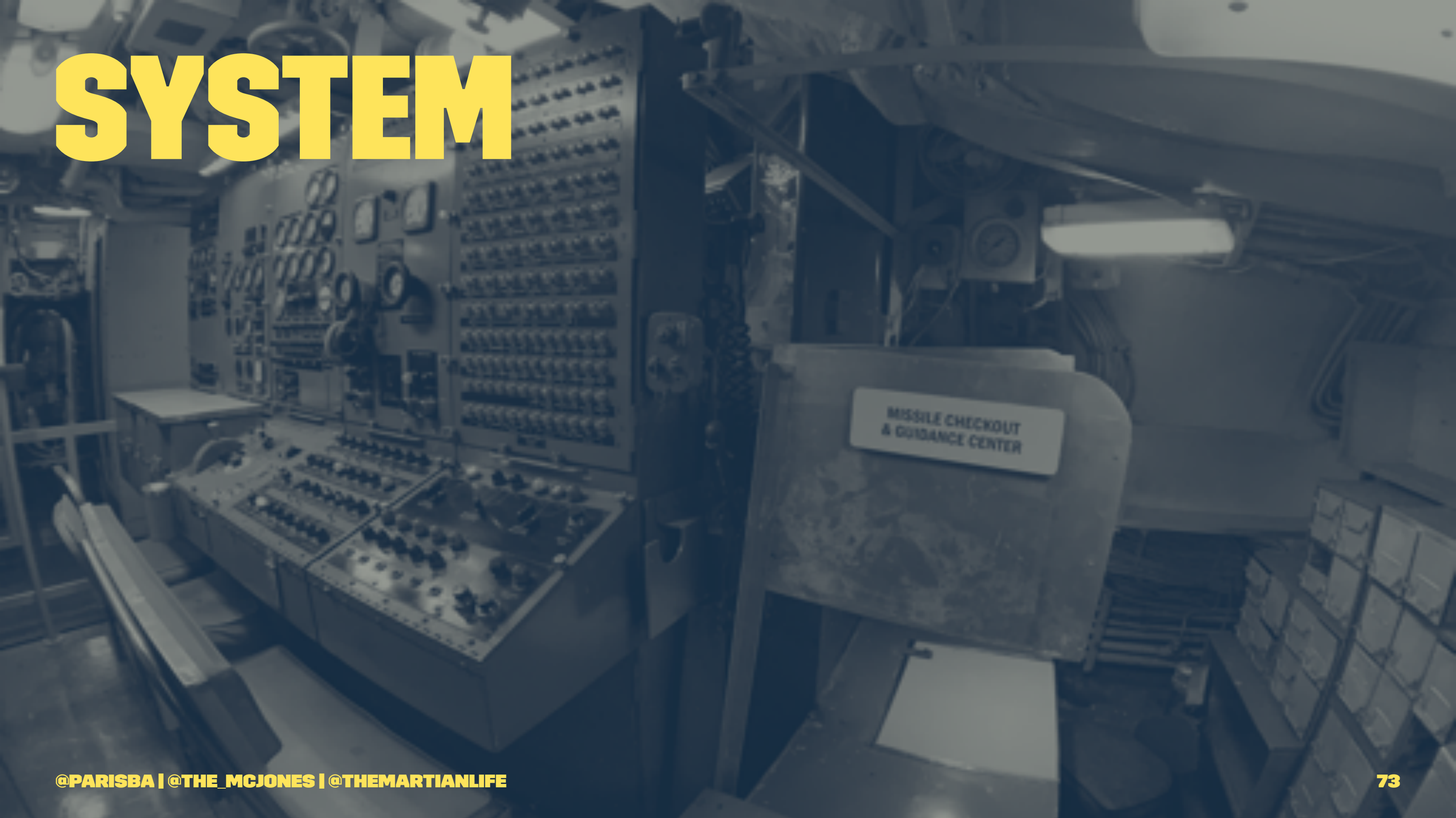
» sprite(s)

» health value

» character name

» player (tag)

SYSTEM



SYSTEM EXAMPLES

» player control

» render

» gravity

» movement

» AI control

GAMES, GAMES, GAMES

ECS DOESN'T NECESSARILY MEAN LESS CODE.

Theoretically it means better code, though.

WHAT ECS IS GOOD FOR

PERFORMANCE?

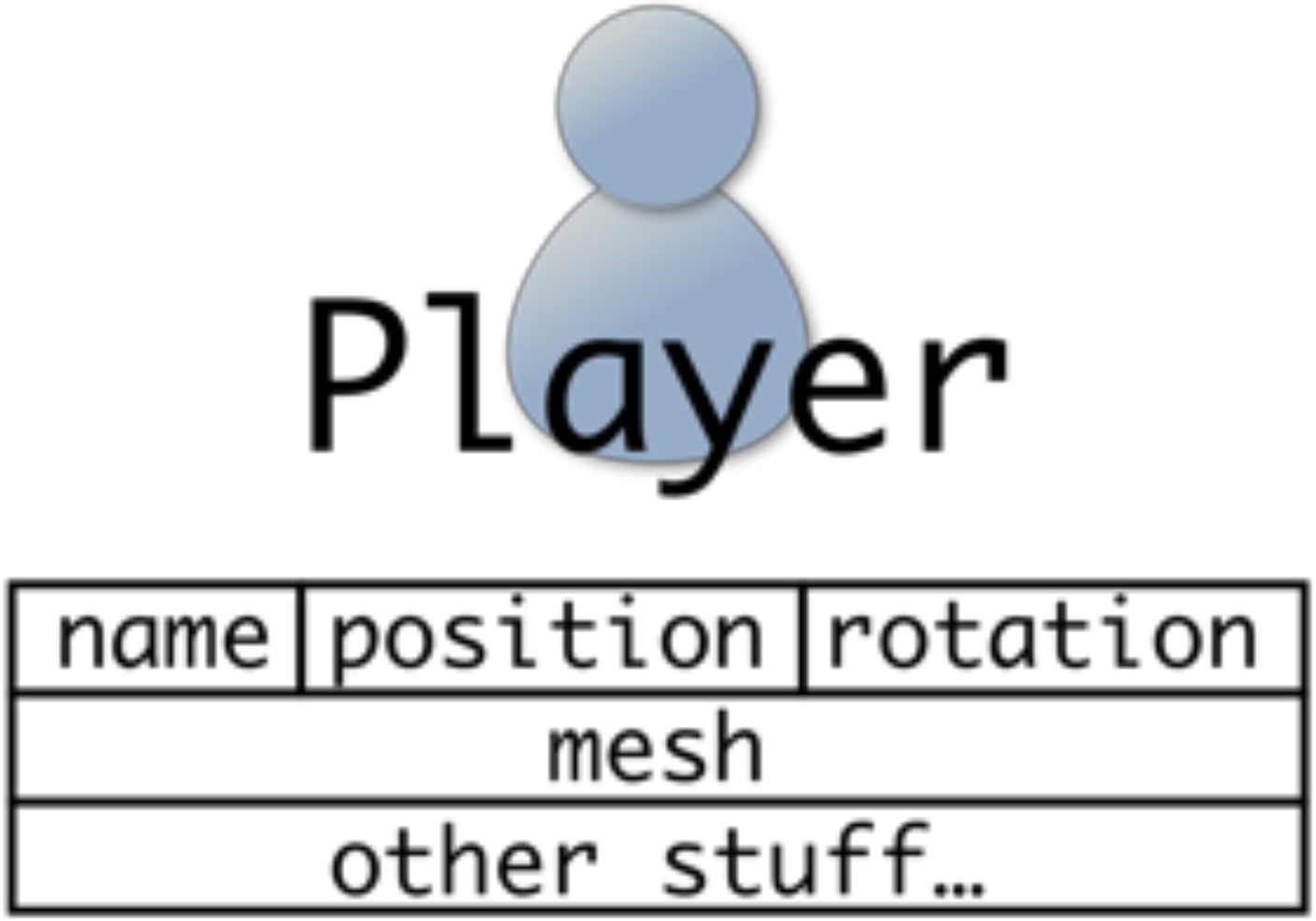
“THE PURPOSE OF ALL PROGRAMS, AND ALL PARTS OF THOSE PROGRAMS, IS TO TRANSFORM DATA FROM ONE FORM TO ANOTHER.”

Mike Acton

DATA ORIENTED DESIGN

DATA-ORIENTED DESIGN IS TRICKY

YOU NEVER DO JUST ONE THING


Player
name
position
rotation
mesh
other stuff…

Enemy

name	position	rotation
mesh		
other stuff…		

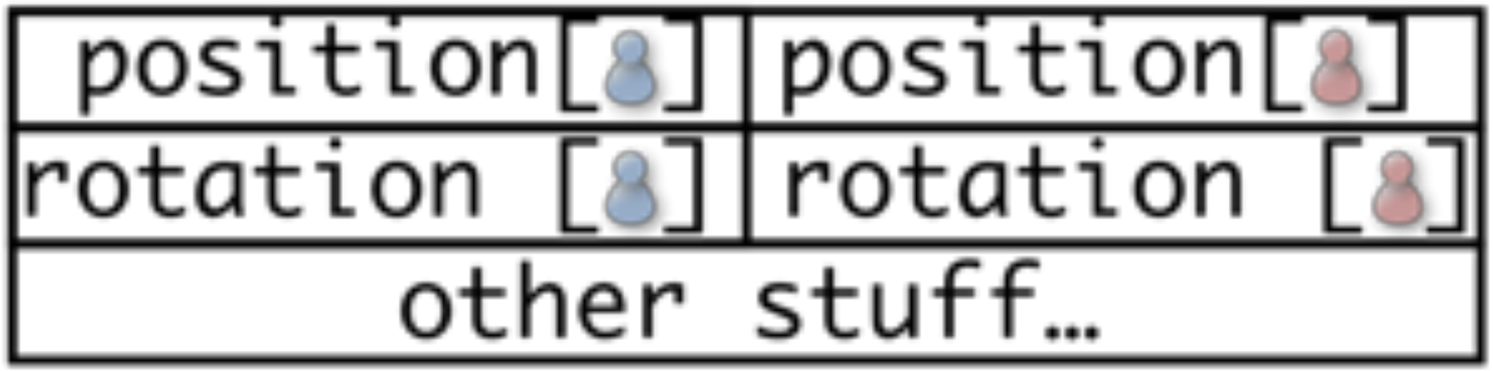
CACHE MISSES

CPUS HAVE A HIERARCHICAL CACHE SYSTEM

One cycle on a 3 GHz processor	1	ns			
L1 cache reference	0.5	ns			
Branch mispredict	5	ns			
L2 cache reference	7	ns			14x L1 cache
Mutex lock/unlock	25	ns			
Main memory reference	100	ns			20x L2, 200x L1
Compress 1K bytes with Snappy	3,000	ns			
Send 1K bytes over 1 Gbps network	10,000	ns	0.01	ms	
Read 4K randomly from SSD*	150,000	ns	0.15	ms	
Read 1 MB sequentially from memory	250,000	ns	0.25	ms	
Round trip within same datacenter	500,000	ns	0.5	ms	
Read 1 MB sequentially from SSD*	1,000,000	ns	1	ms	4X memory
Disk seek	10,000,000	ns	10	ms	20x datacenter RT
Read 1 MB sequentially from disk	20,000,000	ns	20	ms	80x memory, 20X SSD
Send packet CA->Netherlands->CA	150,000,000	ns	150	ms	

CACHE MISSES SUCK

transforms



DATA-ORIENTED DESIGN = PROGRAMMING FOR GOOD MEMORY ACCESS

ECS ENCOURAGES DATA-ORIENTED DESIGN

CACHES, MEMORY SPEED, OH MY!

PARALLELISATION

LOTS OF TASKS.
LARGE SET OF DATA.


STRUCTURE AROUND
CONSUMING DATA AS A
STREAM.

COMPLEX, INTERLOCKING SYSTEMS.

COMPOSABILITY
THE BIGGEST ADVANTAGE

ECS IS GREAT FOR...

» GUI Programming

» Editors

» Audio systems

» VFX systems

» Simulations

» A service registry

» ... almost anything that's event-driven

» ... anything you want to easily unit test

GUI PROGRAMMING

A NEAT SIDE-EFFECT OF ECS!

IN PRACTICE, IT'S NOT THAT EASY...

WHAT ECS IS NOT GOOD FOR

» Disk heavy operations

» Unchanging data and procedures

» Heavily hierarchical data

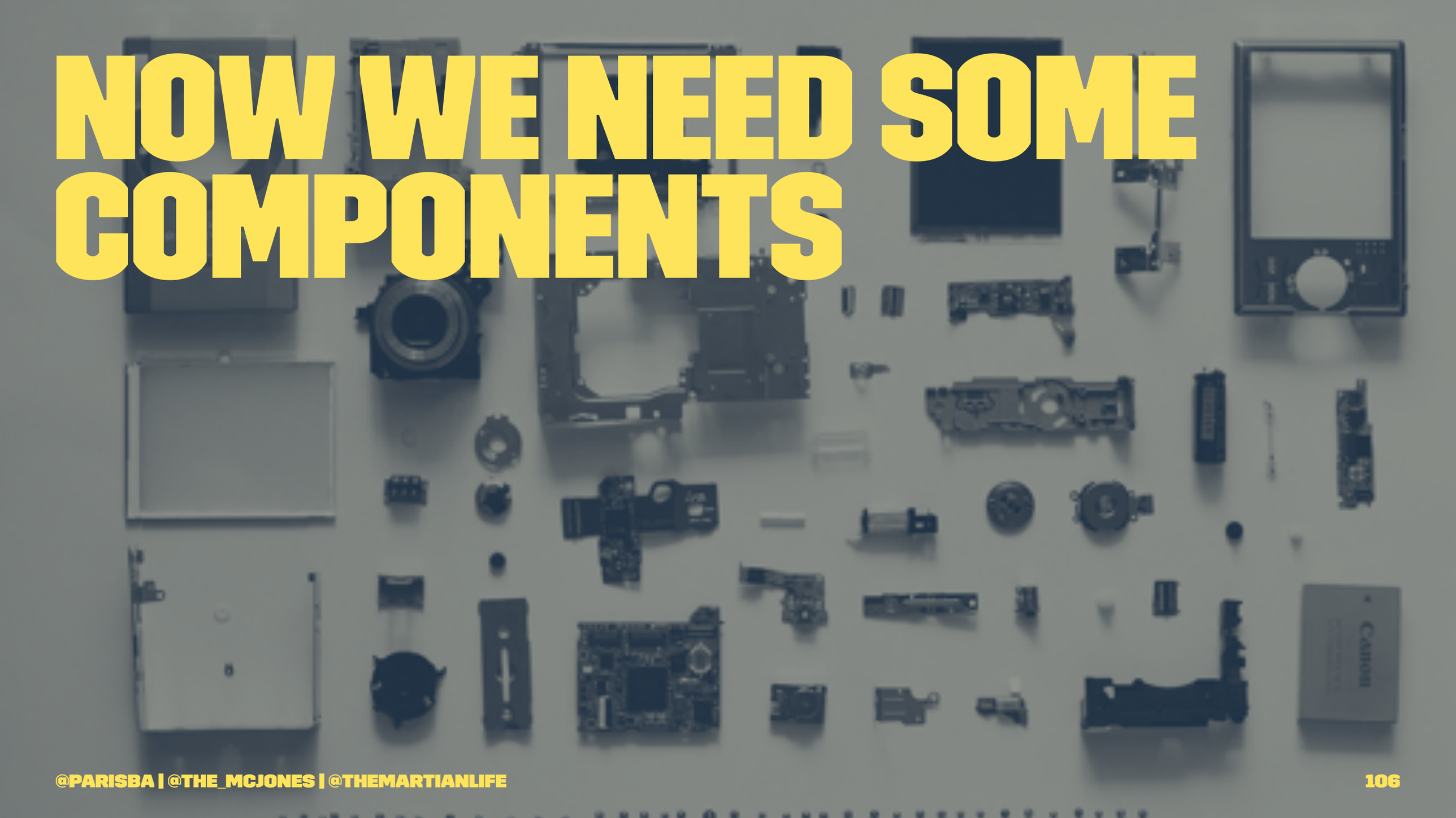
HOW WOULD YOU GO ABOUT MAKING AN ECS?

QUICK AND DIRTY WAY

START WITH OUR ENTITIES

```
entities int[] = [0,1,2,3,4,5...]

/*
player
skeleton with sword
skeleton with shield
camera
etc
etc
*/
```

NOW WE NEED SOME COMPONENTS



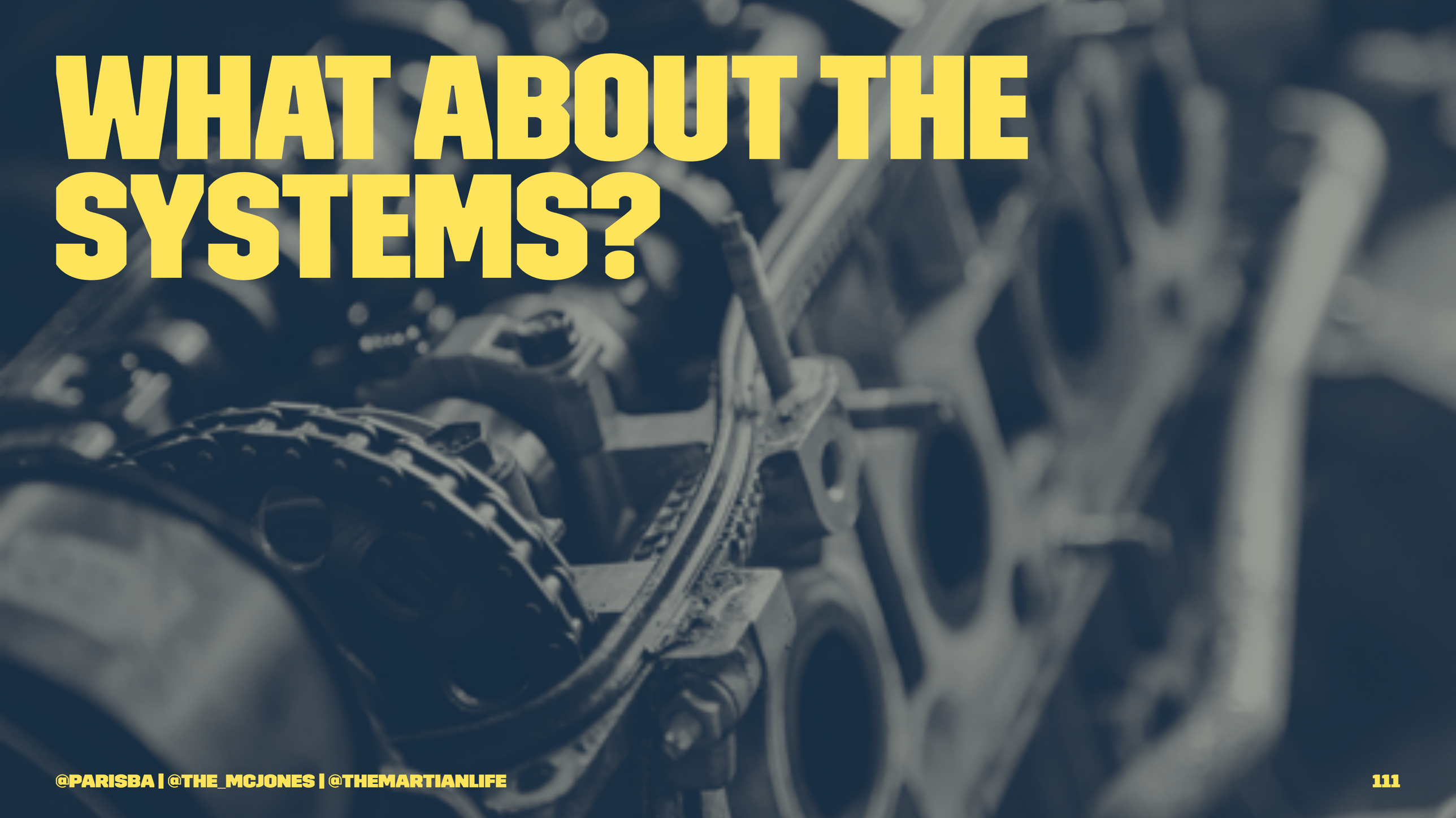
```
TransformComponent
{
 position
 rotation
 scale
}

HealthComponent
{
 life
}

DamageComponent
{
 damage
 target
}

PlayerComponent {}
```

```
MovementComponent
{
 speed
 direction
}

ShieldComponent
{
 reduction
}

RenderComponent
{
 mesh
 textures
}

CameraComponent {}
```


HOW DO WE CONNECT ENTITIES AND COMPONENTS?

```
Game
{
    // all our entities...
    entities int[] = [0,1,2,3,4,5...]

    // all of these are of size entities long
    damages DamageComponent[]
    transforms TransformComponent[]
    renders RenderComponent[]
    movers MovementComponent[]
    healths HealthComponent[]
    shields ShieldComponent[]
    players PlayerComponent[]
    cameras CameraComponent[]
```

```
damages = [nil, 1, 1, 3, 3, 1, nil]
healths = [100, 5, 5, 5, 5, 5, nil]
```



WHAT ABOUT THE SYSTEMS?

```
updateFrame(deltaTime)
{
    // first we get user input and use that to update player position
    input = getInput()
    updatePlayerSpeed(input, players)

    // then move everything that can be moved
    // including the player
    moveEntities(movers, transforms, deltaTime)

    // then we deal damage
    damageEntities(damages, healths)

    // then we reduce damage for everyone with shields
    reduceDamage(shields, healths)

    // then we render the frame
    renderFrame(renders, transforms, cameras)
}
```

```
moveEntities(movers, transforms, deltaTime)
{
    for mover, transform in movers, transforms
    {
        transform.position += mover.velocity * deltaTime
    }
}
```

Player

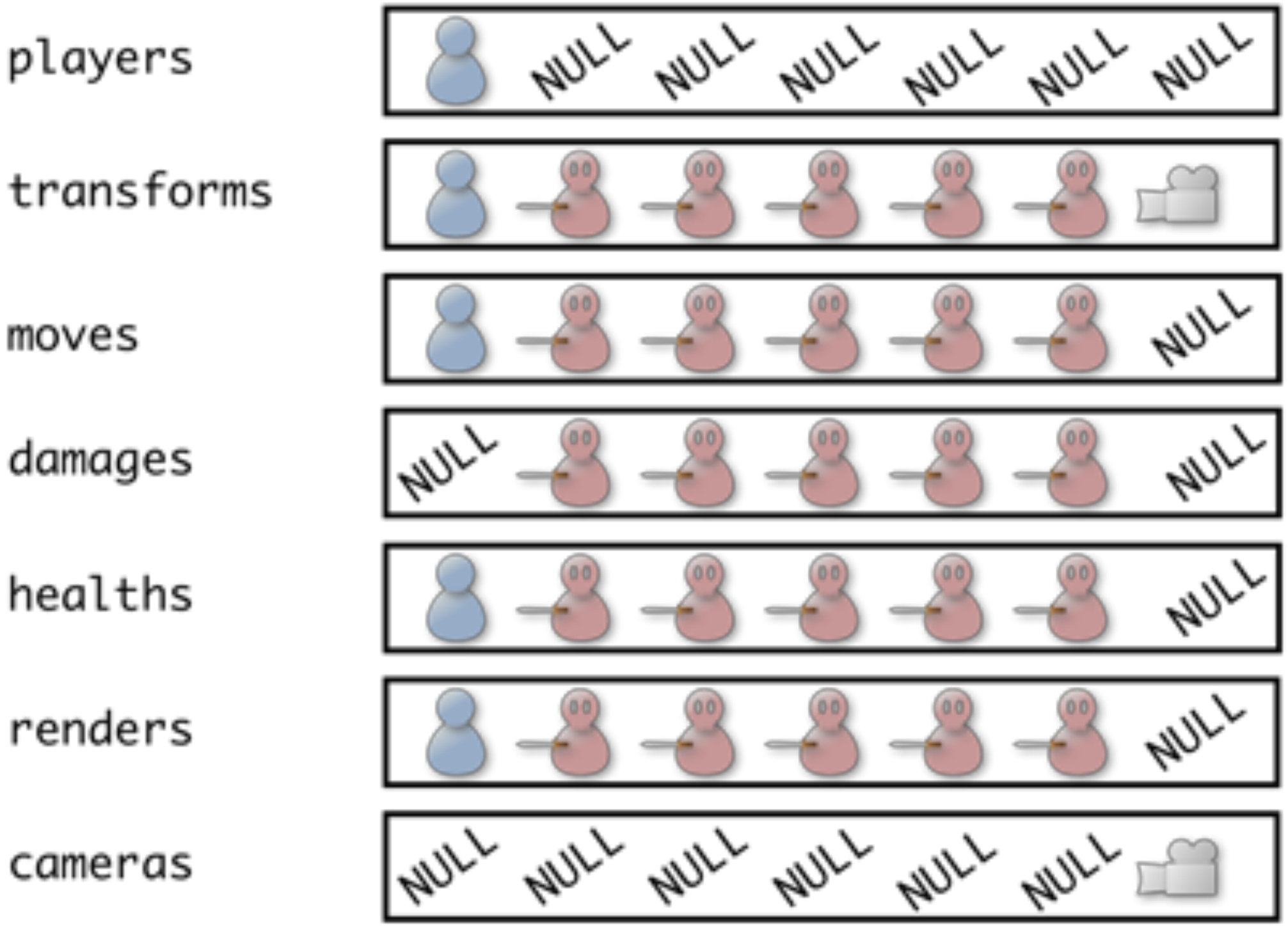
- transform
- player
- render
- move
- health
- …

SkeletonWithSword

- transform
- render
- move
- health
- damage
- …

Camera

- transform
- camera
- …


players
NULL
NULL
NULL
NULL
NULL
NULL
transforms
moves
NULL
damages
NULL
NULL
healths
NULL
renders
NULL
cameras
NULL
NULL
NULL
NULL
NULL
NULL

IT GETS BETTER, OF COURSE

This approach isn't the best for large scale, but its the simplest way to get started.

EXISTING IMPLEMENTATIONS

ECS IMPLEMENTATIONS

» Rust (Specs): https://github.com/slide-rs/specs

» C++ (EnTT): https://github.com/skypjack/entt

» Elixir: https://github.com/yosriady/ecs

» C# (Entitas): https://github.com/sschmid/Entitas-CSharp

» Java (Artemis): http://gamadu.com/artemis/

» C# (EgoCS): https://github.com/andoowhy/EgoCS

» C++ (EntityX): https://github.com/alecthomas/entityx

» C++ (Anax): https://github.com/miguelmartin75/anax

» ...and many, many more!

NONE OF THIS IS NEW.

SUMMARY

ENTITIES
COMPONENTS
SYSTEMS

ECS = COMPOSABLE

ECS IS COMPOSITION ON STEROIDS

DATA-ORIENTED DESIGN

STRENGTHS

» Performance (hopefully)

» Flexibility

» No heirarchy to remember

» Systems have many of the advantages of microservices/functional programming

WEAKNESSES

» More code upfront

» Harder to keep everything in your head

» No clear starting point

PICK AND CHOOSE THE IDEAS YOU LIKE

There is no one 'right way'.

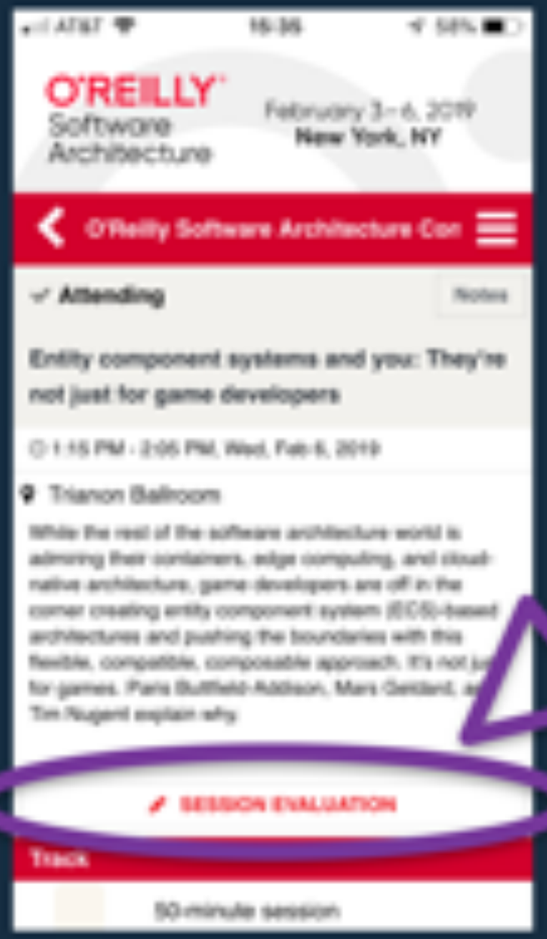
RESOURCES

» Slides will go up on the SACon website shortly

» Talk resources can be found at https://lab.to/SAConNYC19Resources

REVIEW US!


Entity component systems and you: They're not just for game developers
Rate This Session
presentation for?
Software architects, programmers, and the curious
Level
Beginner
Prerequisite knowledge
A basic understanding of programming or software architecture
What you'll learn
Learn what an ECS is and how it's related to existing similar architectures
Understand why ECS is important even if you're not a game developer
Description


O'REILLY
Software Architecture
February 3–6, 2019
New York, NY
O'Reilly Software Architecture Con
Attending
Notes
Entity component systems and you: They're not just for game developers
1:15 PM - 2:05 PM, Wed, Feb 6, 2019
Trianon Ballroom
SESSION EVALUATION
Track
50-minute session

THANK YOU!

» Find us online:

  » @parisba

  » @The_McJones

  » @TheMartianLife

» Chat with us at 'Meet the Experts'!
3:05PM today! Sponsor Pavilion! Come and tell us we're wrong!

» Stay warm + enjoy the rest of your #OReillySACon <3