# ECS @ BBI

Yossarian (Yggy) King

UBC CPSC 427

March 2, 2021

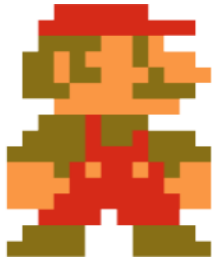
tinyurl.com/ecsatbbi
linkedin.com/in/yossarianking

BLACKBIRD INTERACTIVE

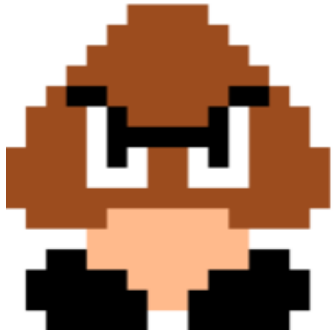

Good for programmers

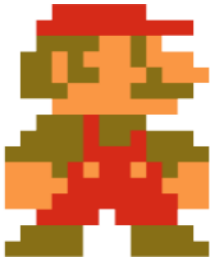
Good for performance

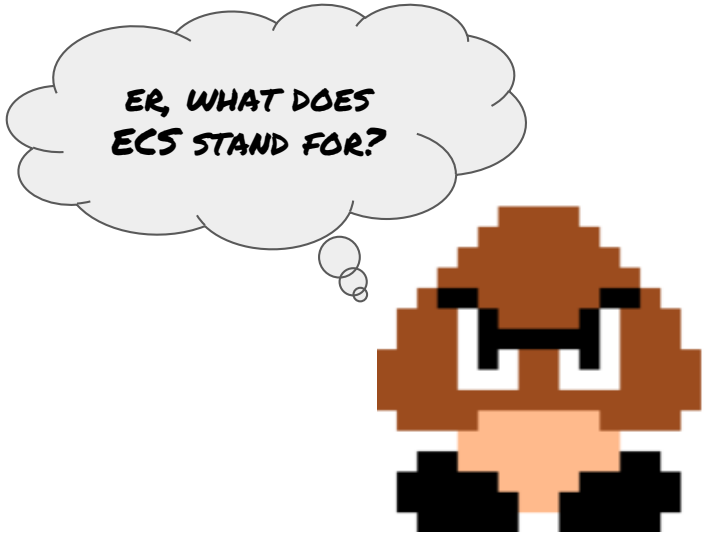
Good for designers



First, let's recap ...

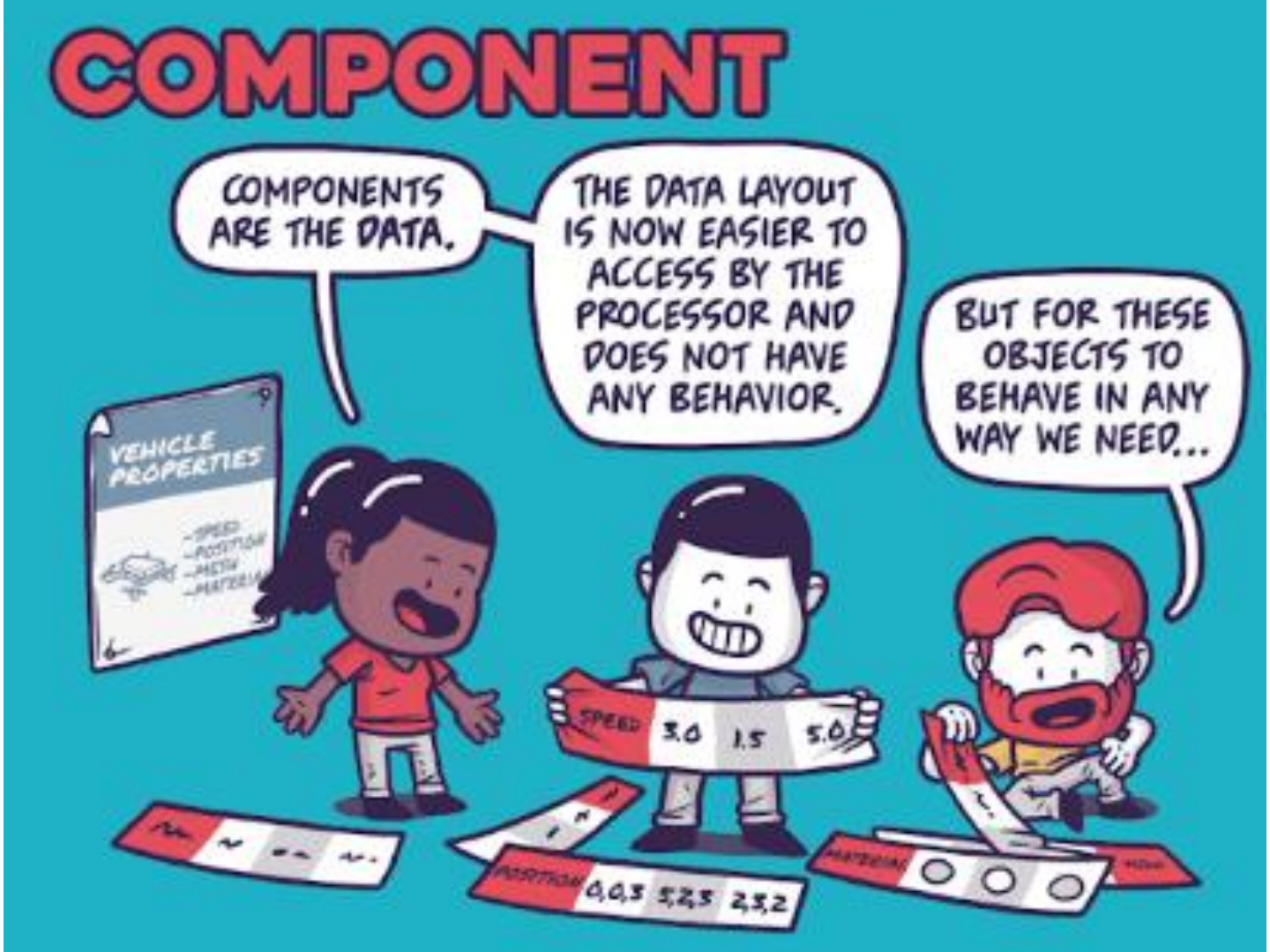




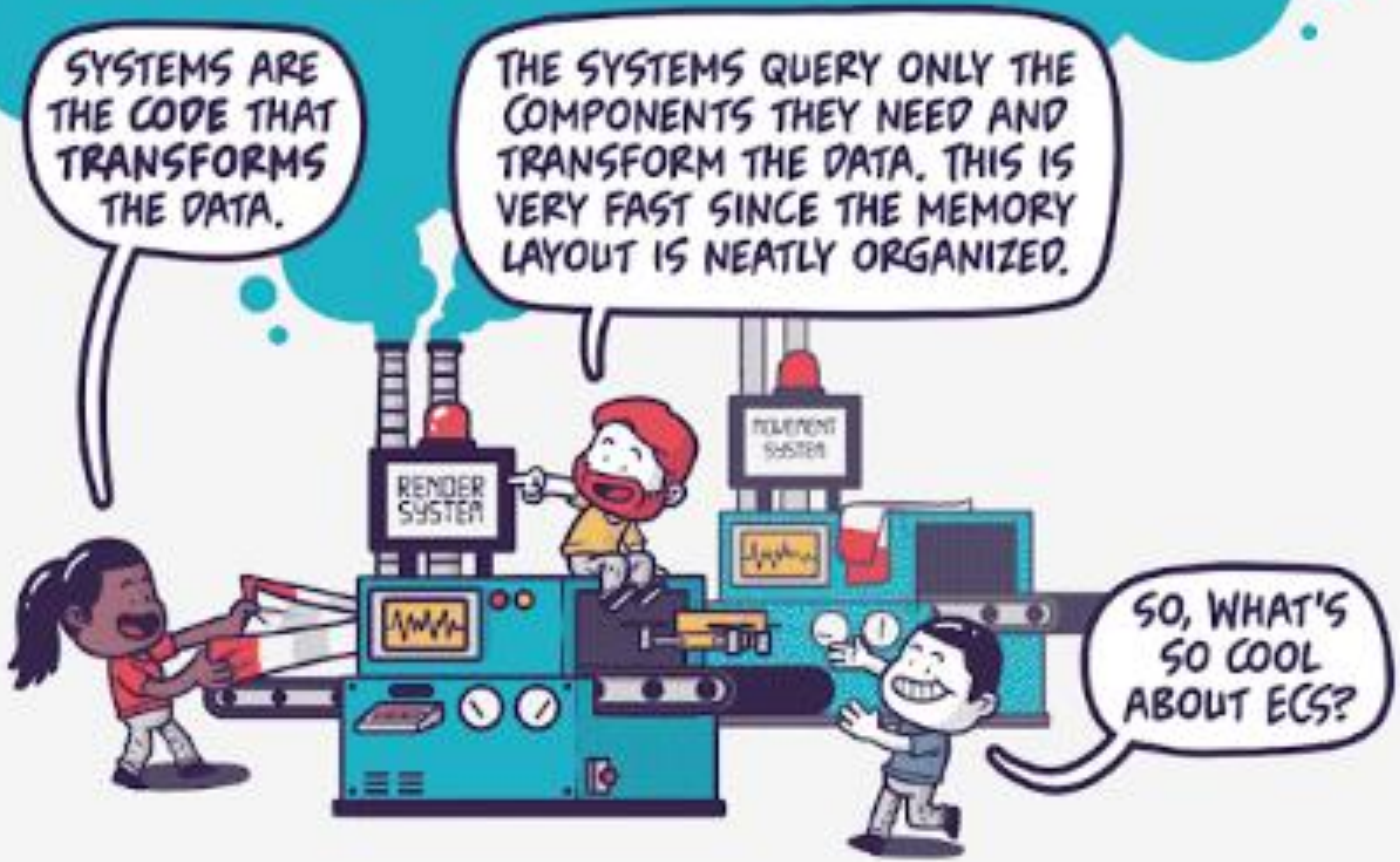

ER, WHAT DOES
ECS STAND FOR?


WHAT IS ECS?
A DIFFERENT PARADIGM OF WRITING CODE, WHERE WE MODEL OUR PROGRAMS IN A DATA ORIENTED WAY.
LET'S GO OVER WHAT IT MEANS.
ENTITY
COMPONENT
SYSTEM


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

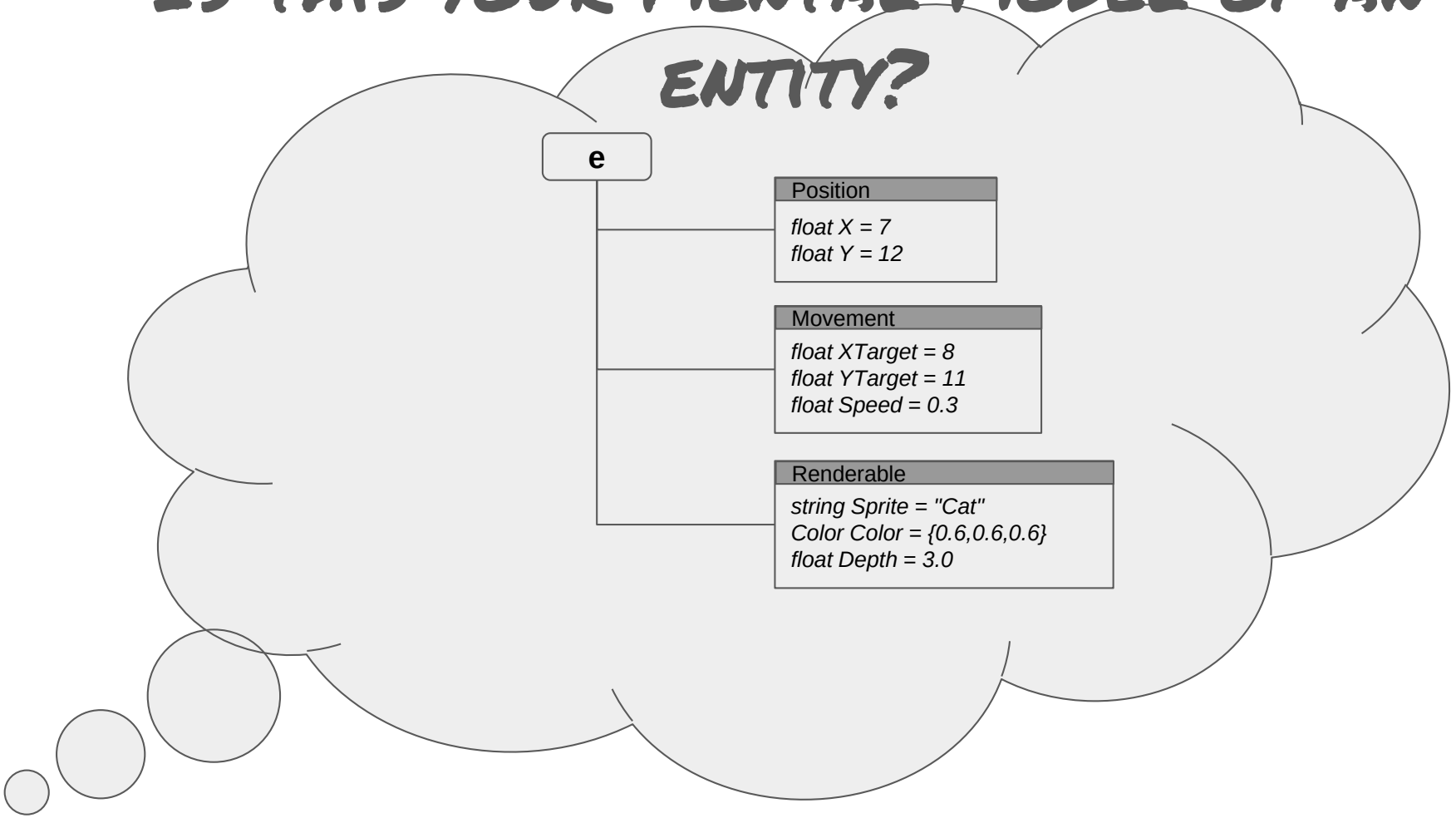

COMPONENT
COMPONENTS ARE THE DATA.
THE DATA LAYOUT IS NOW EASIER TO ACCESS BY THE PROCESSOR AND DOES NOT HAVE ANY BEHAVIOR.
BUT FOR THESE OBJECTS TO BEHAVE IN ANY WAY WE NEED...
VEHICLE PROPERTIES
SPEED 3.0 1.5 5.0
POSITION 0,0,3 5,2,3 2,3,2

# SYSTEM

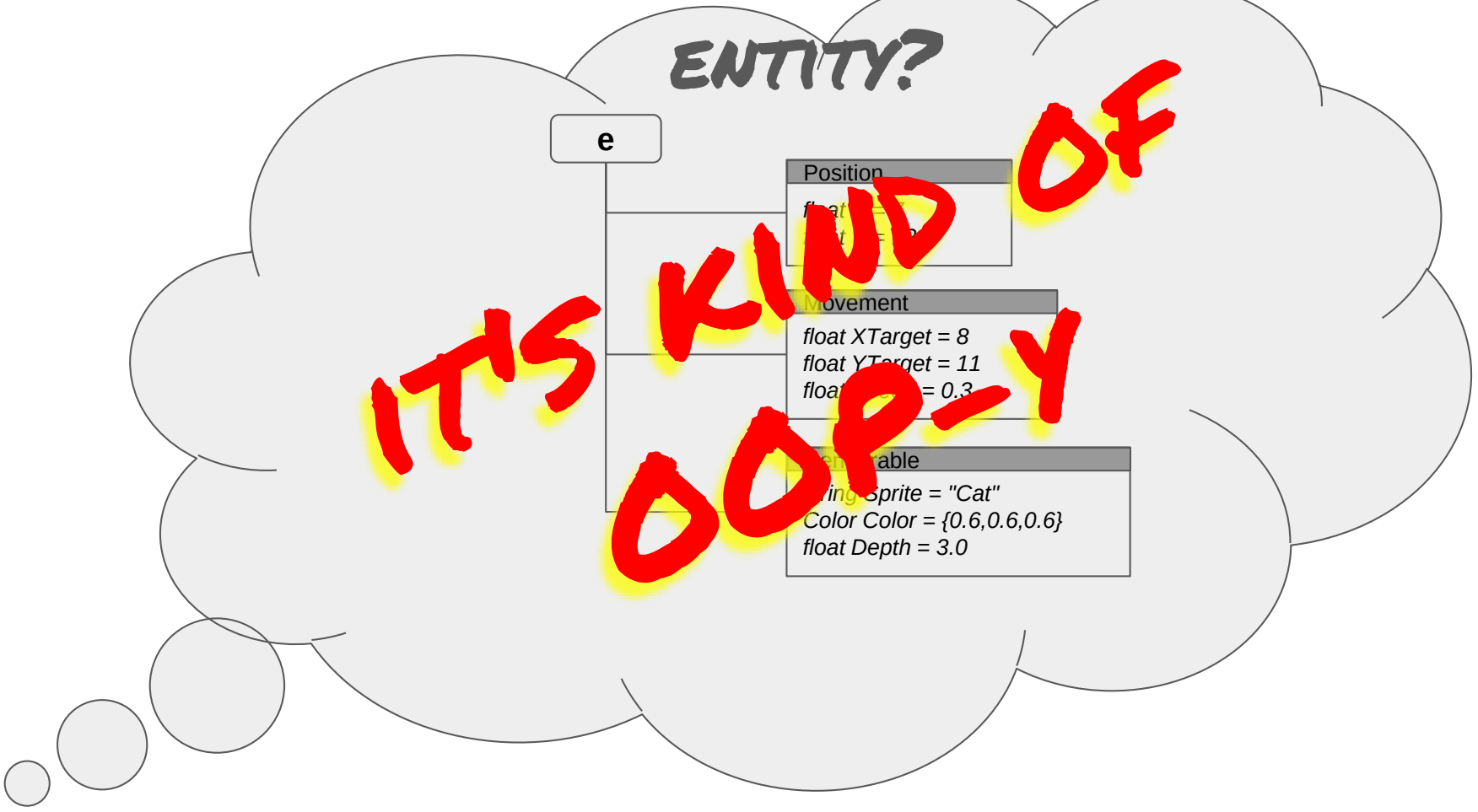

SYSTEMS ARE THE CODE THAT TRANSFORMS THE DATA.
THE SYSTEMS QUERY ONLY THE COMPONENTS THEY NEED AND TRANSFORM THE DATA. THIS IS VERY FAST SINCE THE MEMORY LAYOUT IS NEATLY ORGANIZED.
RENDER SYSTEM
MOVEMENT SYSTEM
SO, WHAT'S SO COOL ABOUT ECS?


CONTENT: ARTURO NEREU GRAPHICS: GERARDO ALBA

# Is this your mental model of an entity?



# Is this your mental model of an entity?



# Get tabular!

# Entities are Keys, Components are tables

| Position | | |
|---|---|---|
| *Key* | *X* | *Y* |
| | | |
| 10 | 7 | 12 |
| | | |
| | | |
| | | |
| | | |

| Movement | | | |
|---|---|---|---|
| *Key* | *XTarget* | *YTarget* | *Speed* |
| | | | |
| | | | |
| | | | |
| 10 | 8 | 11 | 0.3 |

| Renderable | | | |
|---|---|---|---|
| *Key* | *Sprite* | *Color* | *Depth* |
| | | | |
| | | | |
| 10 | "Cat" | White | 3 |
| | | | |
| | | | |

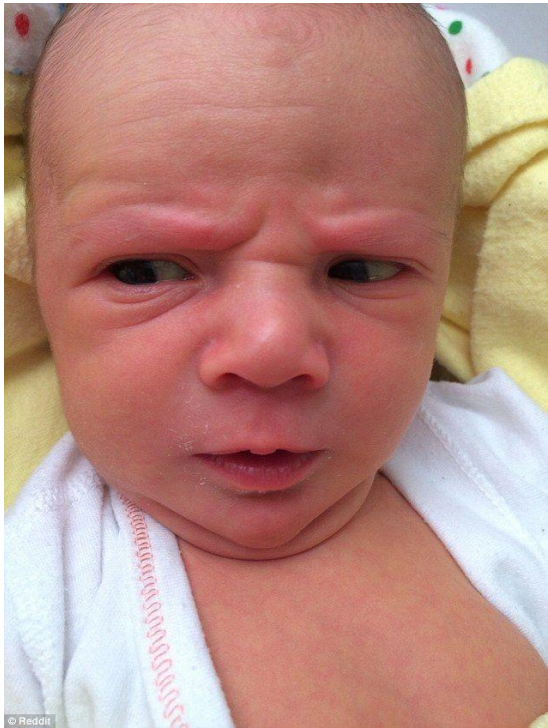

OK, WHY IS
THAT BETTER?

# Reason #1

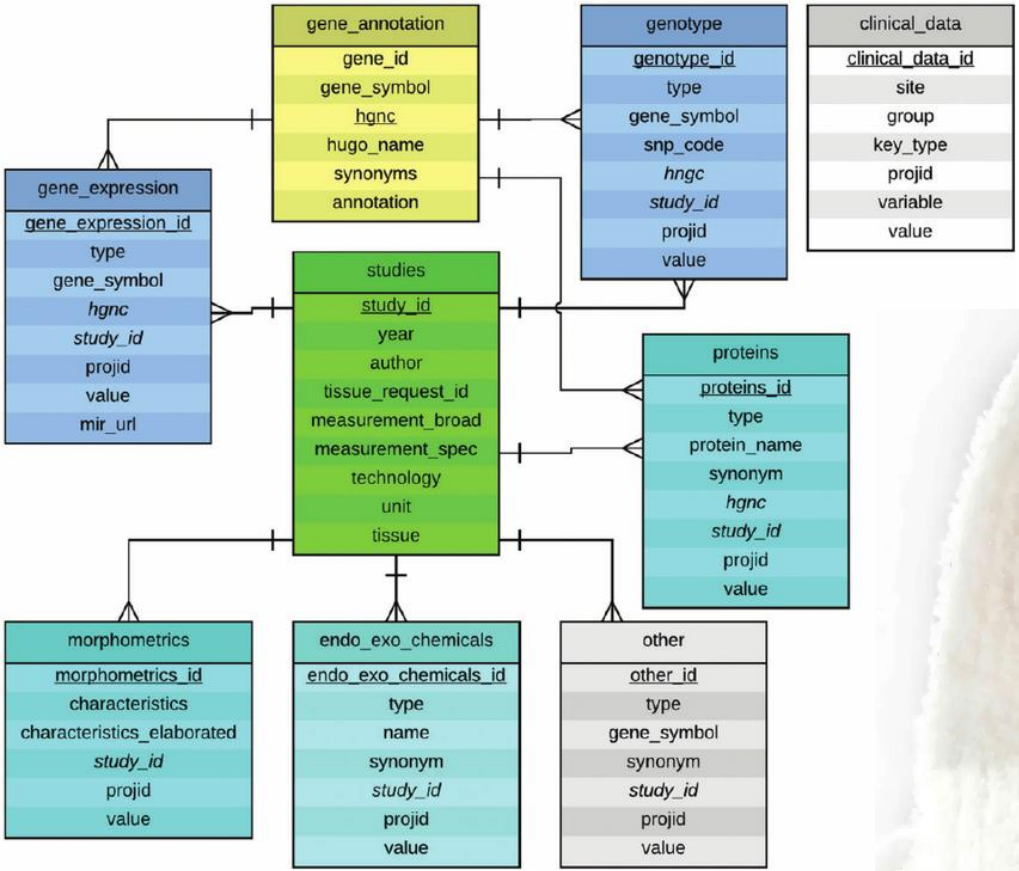
## Puts focus on the data, not the entity

## End the tyranny of the instance!

# Object-oriented memory layout



# Data-oriented memory layout




# Reason #2

# Keeps behaviour out of components

# Reason #3

# Emphasizes batch processing

# Data are plural!

# Reason #4

## Linear layout for a happy cache


REASON #5
WHOLE NEW WORLDS
OF DATA ANALYSIS + OPTIMIZATION
(MAYBE)

So if we're talking tables, what about SQL?

How would you find all entities with a position and a movement component?

```
Entities.ForEach(
(ref Position pos,
ref Movement move)
=> {DoMove(pos,move)});
```

```
SELECT *
FROM Position
JOIN Movement
ON Position.Key=Movement.Key
```

| Position | | |
|---|---|---|
| *Key* | *X* | *Y* |
| 3 | 1 | 2 |
| 10 | 7 | 12 |
| 11 | 3 | 4 |
| 17 | 5 | 6 |
| 21 | 7 | 8 |
| 30 | 3 | 3 |

| Movement | | | |
|---|---|---|---|
| *Key* | *XTarget* | *YTarget* | *Speed* |
| 5 | 2 | 2 | 0.1 |
| 10 | 8 | 11 | 0.3 |
| 15 | 0 | 0 | 0.1 |
| 30 | 5 | 3 | 0.5 |

```
SELECT *
FROM Position
JOIN Movement
WHERE Position.Key=Movement.Key
```

| *Key* | *X* | *Y* | *XTarget* | *YTarget* | *Speed* |
|---|---|---|---|---|---|
| 10 | 7 | 12 | 8 | 11 | 0.3 |
| 30 | 3 | 3 | 5 | 3 | 0.5 |

# Let's talk about JOINs

| Position | | |
|---|---|---|
| *Key* | *X* | *Y* |
| 3 | 1 | 2 |
| 10 | 7 | 12 |
| 11 | 3 | 4 |
| 17 | 5 | 6 |
| 21 | 7 | 8 |
| 30 | 3 | 3 |

| Movement | | | |
|---|---|---|---|
| *Key* | *XTarget* | *YTarget* | *Speed* |
| 5 | 2 | 2 | 0.1 |
| 10 | 8 | 11 | 0.3 |
| 15 | 0 | 0 | 0.1 |
| 30 | 5 | 3 | 0.5 |

```
SELECT *
FROM Position
JOIN Movement
WHERE Position.Key=Movement.Key
```

| *Key* | *X* | *Y* | *XTarget* | *YTarget* | *Speed* |
|---|---|---|---|---|---|
| 10 | 7 | 12 | 8 | 11 | 0.3 |
| 30 | 3 | 3 | 5 | 3 | 0.5 |

Position

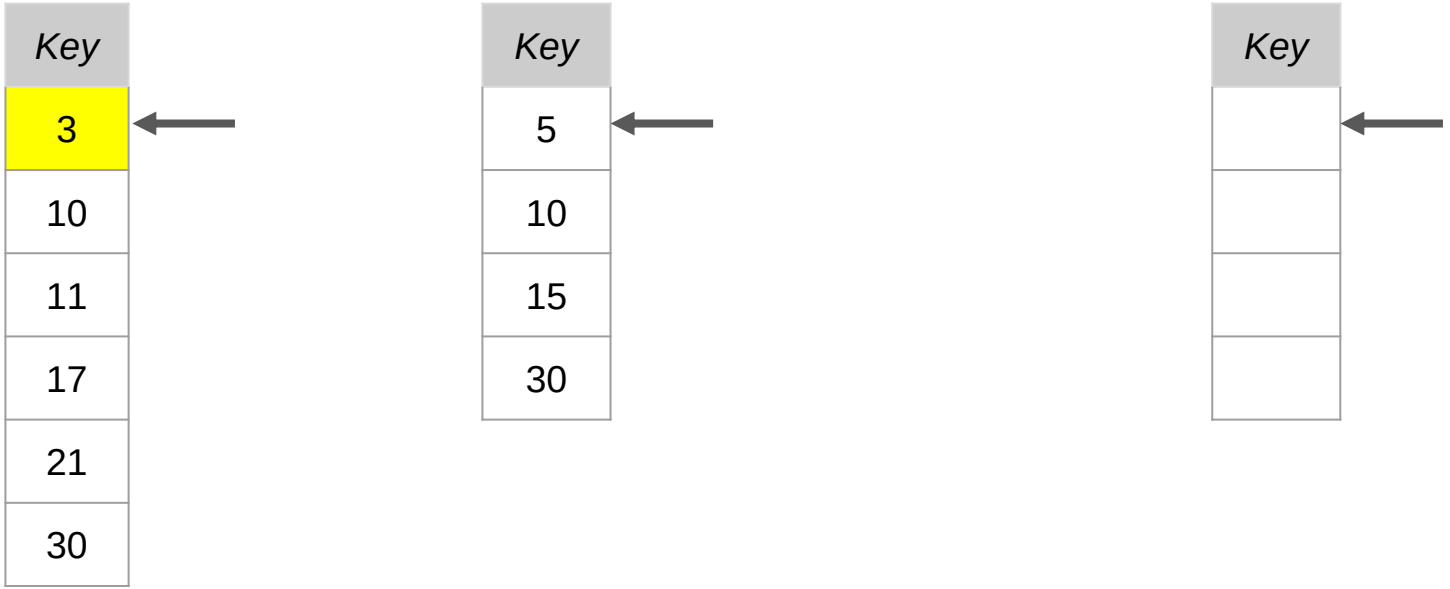
| *Key* | *X* | *Y* |
|---|---|---|
| 3 | 1 | 2 |
| 10 | 7 | 12 |
| 11 | 3 | 4 |
| 17 | 5 | 6 |
| 21 | 7 | 8 |
| 30 | 3 | 3 |

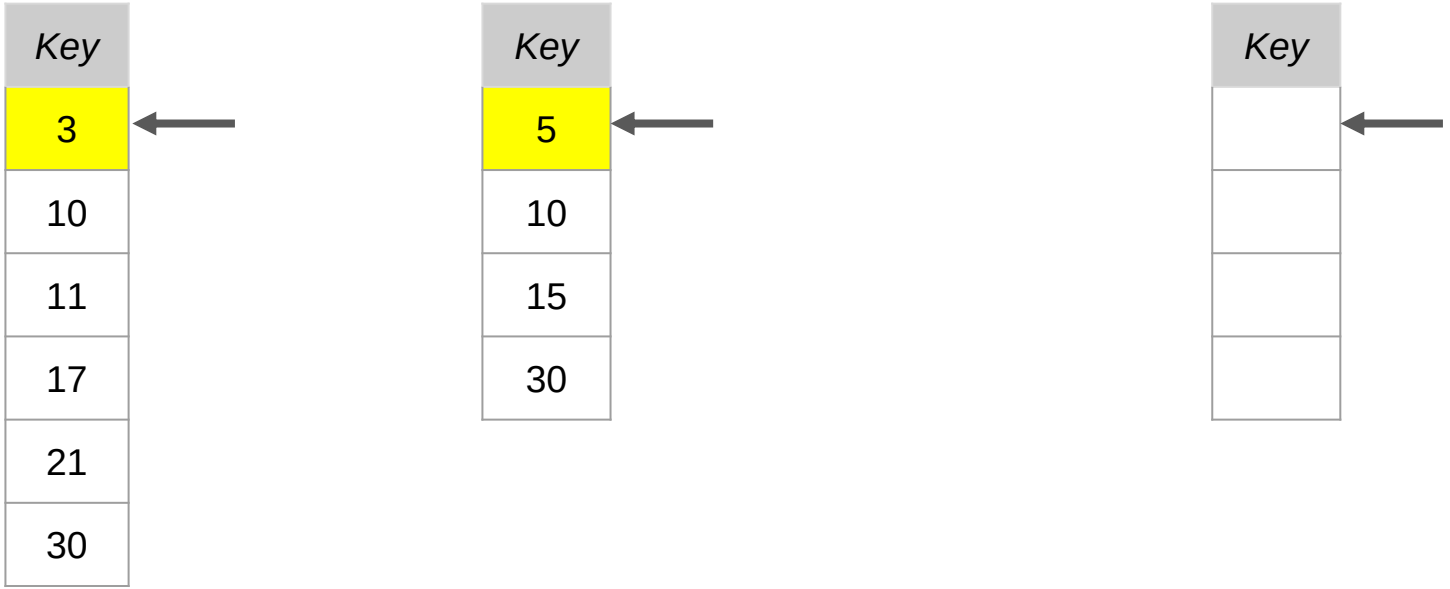
Movement

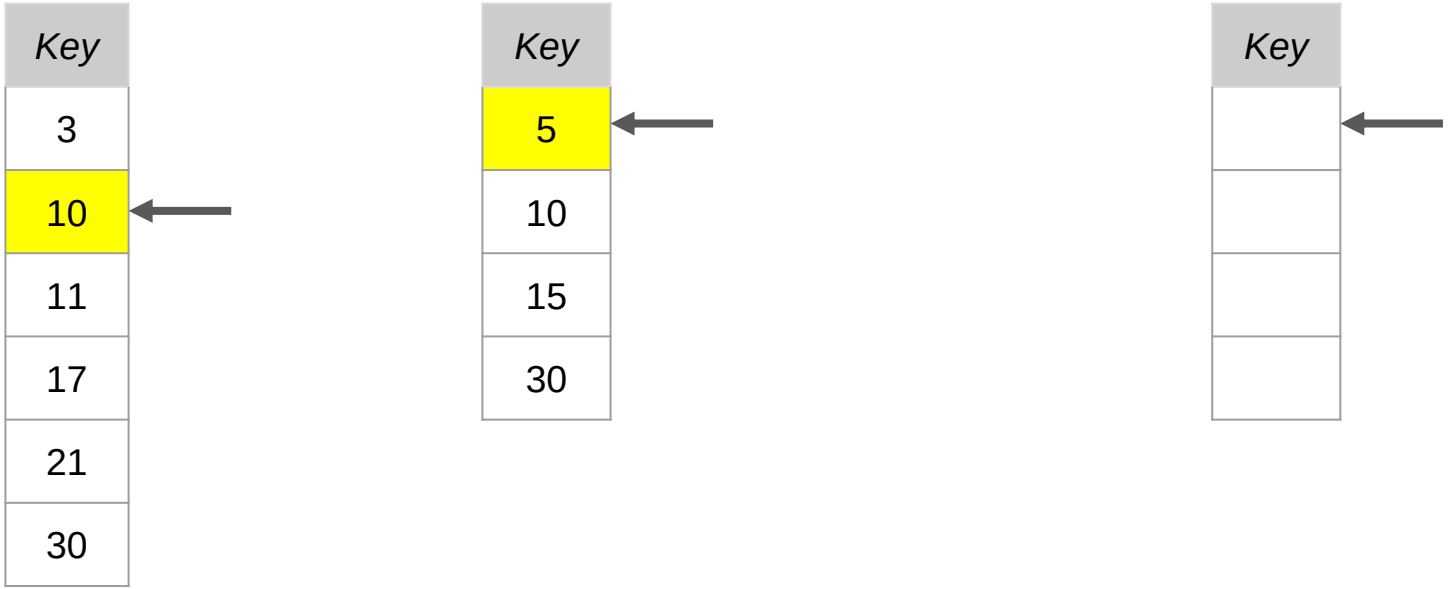
| *Key* | *XTarget* | *YTarget* | *Speed* |
|---|---|---|---|
| 5 | 2 | 2 | 0.1 |
| 10 | 8 | 11 | 0.3 |
| 15 | 0 | 0 | 0.1 |
| 30 | 5 | 3 | 0.5 |

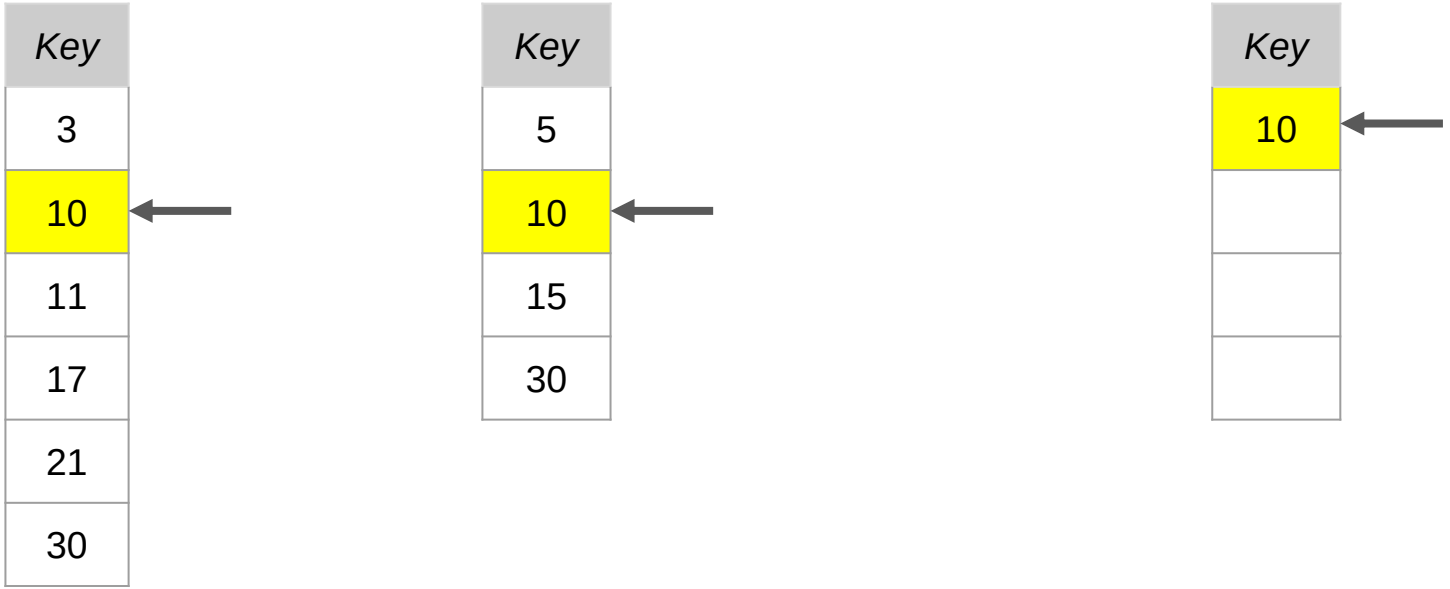
```
SELECT *
FROM Position
JOIN Movement
WHERE Position.Key=Movement.Key
```

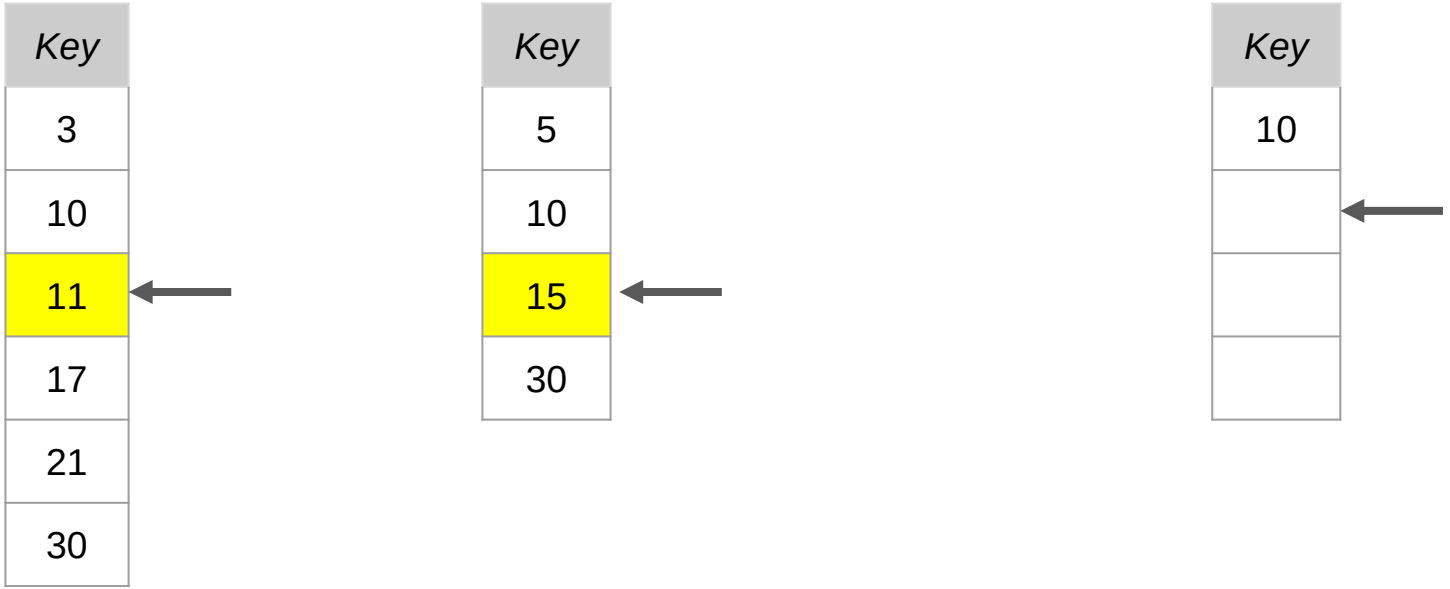
| *Key* | *X* | *Y* | *XTarget* | *YTarget* | *Speed* |
|---|---|---|---|---|---|
| 10 | 7 | 12 | 8 | 11 | 0.3 |
| 30 | 3 | 3 | 5 | 3 | 0.5 |

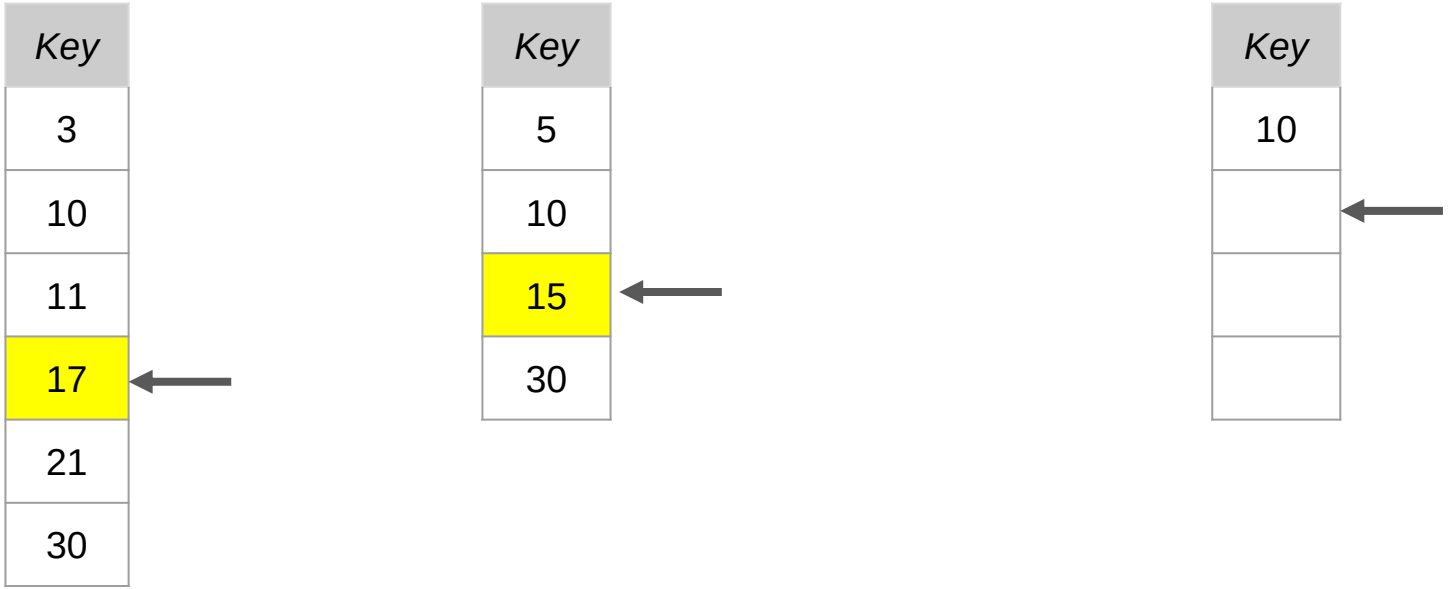

Key
3
10
11
17
21
30
Key
5
10
15
30
Key

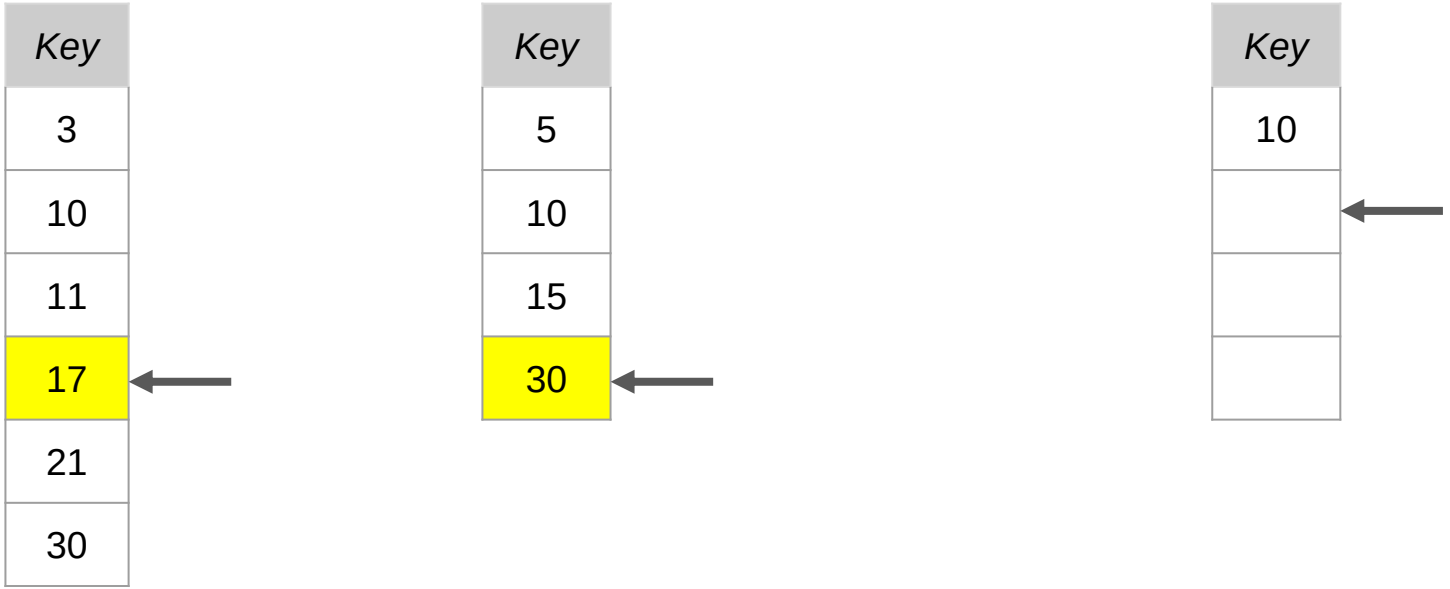

Key
3
10
11
17
21
30
Key
5
10
15
30
Key

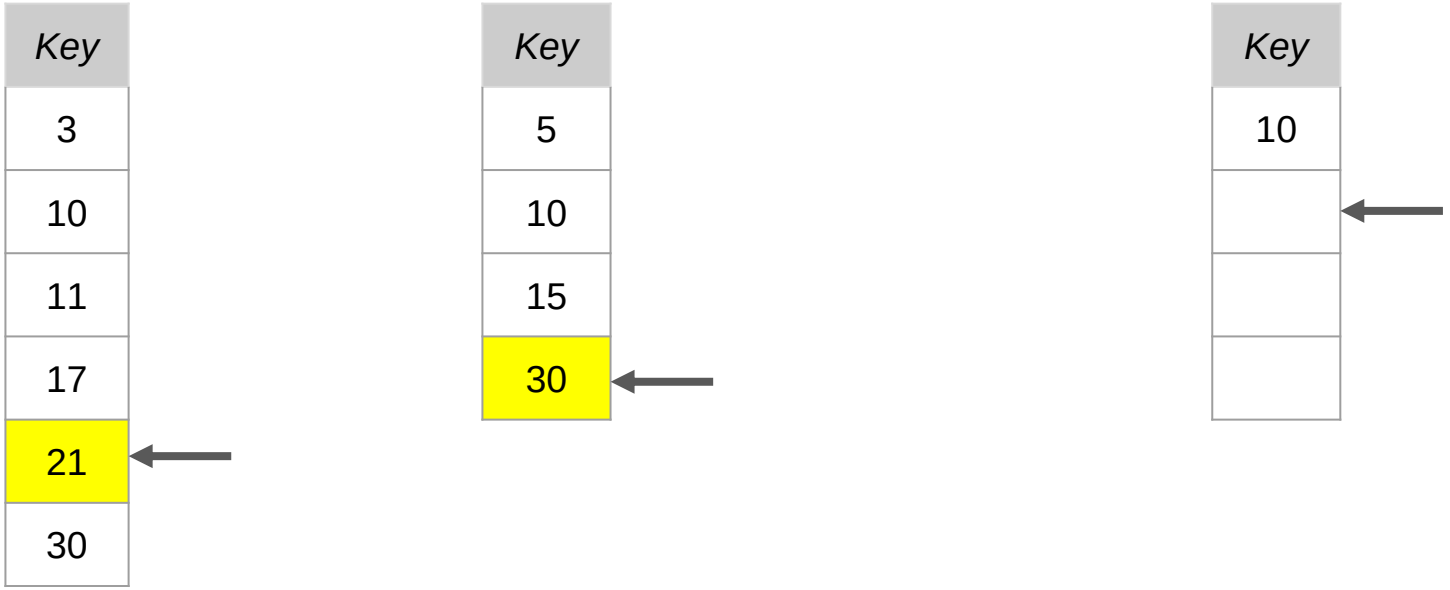

Key
3
10
11
17
21
30
Key
5
10
15
30
Key

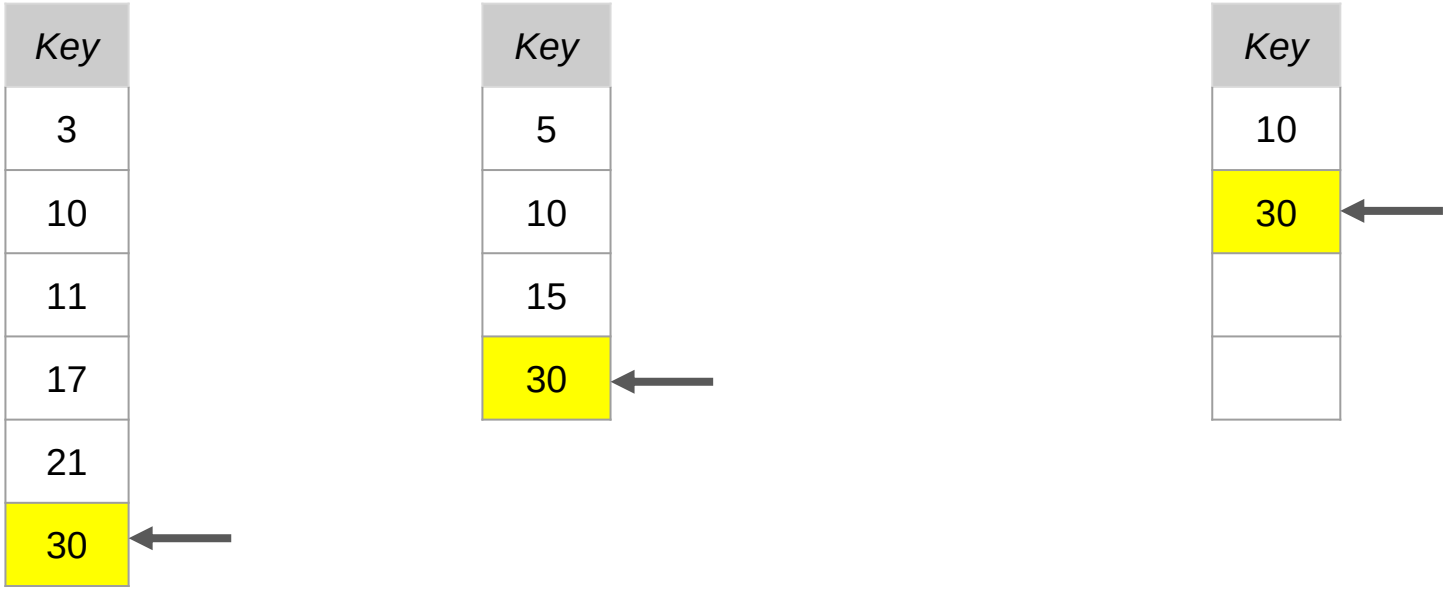

Key
3
10
11
17
21
30
Key
5
10
15
30
Key
10


Key
3
10
11
17
21
30
Key
5
10
15
30
Key
10


Key
3
10
11
17
21
30
Key
5
10
15
30
Key
10


Key
3
10
11
17
21
30
Key
5
10
15
30
Key
10


Key
3
10
11
17
21
30
Key
5
10
15
30
Key
10

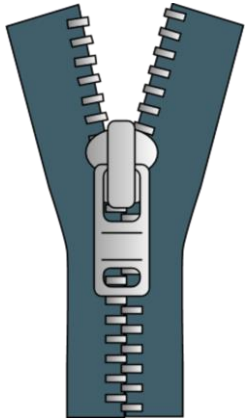

Key
3
10
11
17
21
30
Key
5
10
15
30
Key
10
30

| Key |
| --- |
| 3 |
| 10 |
| 11 |
| 17 |
| 21 |
| 30 |

| Key |
| --- |
| 5 |
| 10 |
| 15 |
| 30 |

| Key |
| --- |
| 10 |
| 30 |
| |
| |

WIKIPEDIA.ORG/SORT-MERGE_JOIN



# However ...

| Position | | |
|---|---|---|
| *Key* | *X* | *Y* |
| 3 | 1 | 2 |
| 10 | 7 | 12 |
| 11 | 3 | 4 |
| 17 | 5 | 6 |
| 21 | 7 | 8 |
| 30 | 3 | 3 |

| Movement | | | |
|---|---|---|---|
| *Key* | *XTarget* | *YTarget* | *Speed* |
| 5 | 2 | 2 | 0.1 |
| 10 | 8 | 11 | 0.3 |
| 15 | 0 | 0 | 0.1 |
| 30 | 5 | 3 | 0.5 |

# Remember that data we wanted to join?

```
SELECT *
FROM Position
JOIN Movement
WHERE Position.Key=Movement.Key
```

| *Key* | *X* | *Y* | *XTarget* | *YTarget* | *Speed* |
|---|---|---|---|---|---|
| 10 | 7 | 12 | 8 | 11 | 0.3 |
| 30 | 3 | 3 | 5 | 3 | 0.5 |

| Position | | |
|---|---|---|
| *Key* | *X* | *Y* |
| 3 | 1 | 2 |
| 10 | 7 | 12 |
| 11 | 3 | 4 |
| 17 | 5 | 6 |
| 21 | 7 | 8 |
| 30 | 3 | 3 |

| Movement | | | |
|---|---|---|---|
| *Key* | *XTarget* | *YTarget* | *Speed* |
| 5 | 2 | 2 | 0.1 |
| 10 | 8 | 11 | 0.3 |
| 15 | 0 | 0 | 0.1 |
| 30 | 5 | 3 | 0.5 |

```
SELECT *
FROM Position
JOIN Movement
WHERE Position.Key=Movement.Key
```

| *Key* | *X* | *Y* | *XTarget* | *YTarget* | *Speed* |
|---|---|---|---|---|---|
| 10 | 7 | 12 | 8 | 11 | 0.3 |
| 30 | 3 | 3 | 5 | 3 | 0.5 |

OUR RESULTS :-)

| Position | | |
|---|---|---|
| *Key* | *X* | *Y* |
| 3 | 1 | 2 |
| 10 | 7 | 12 |
| 11 | 3 | 4 |
| 17 | 5 | 6 |
| 21 | 7 | 8 |
| 30 | 3 | 3 |

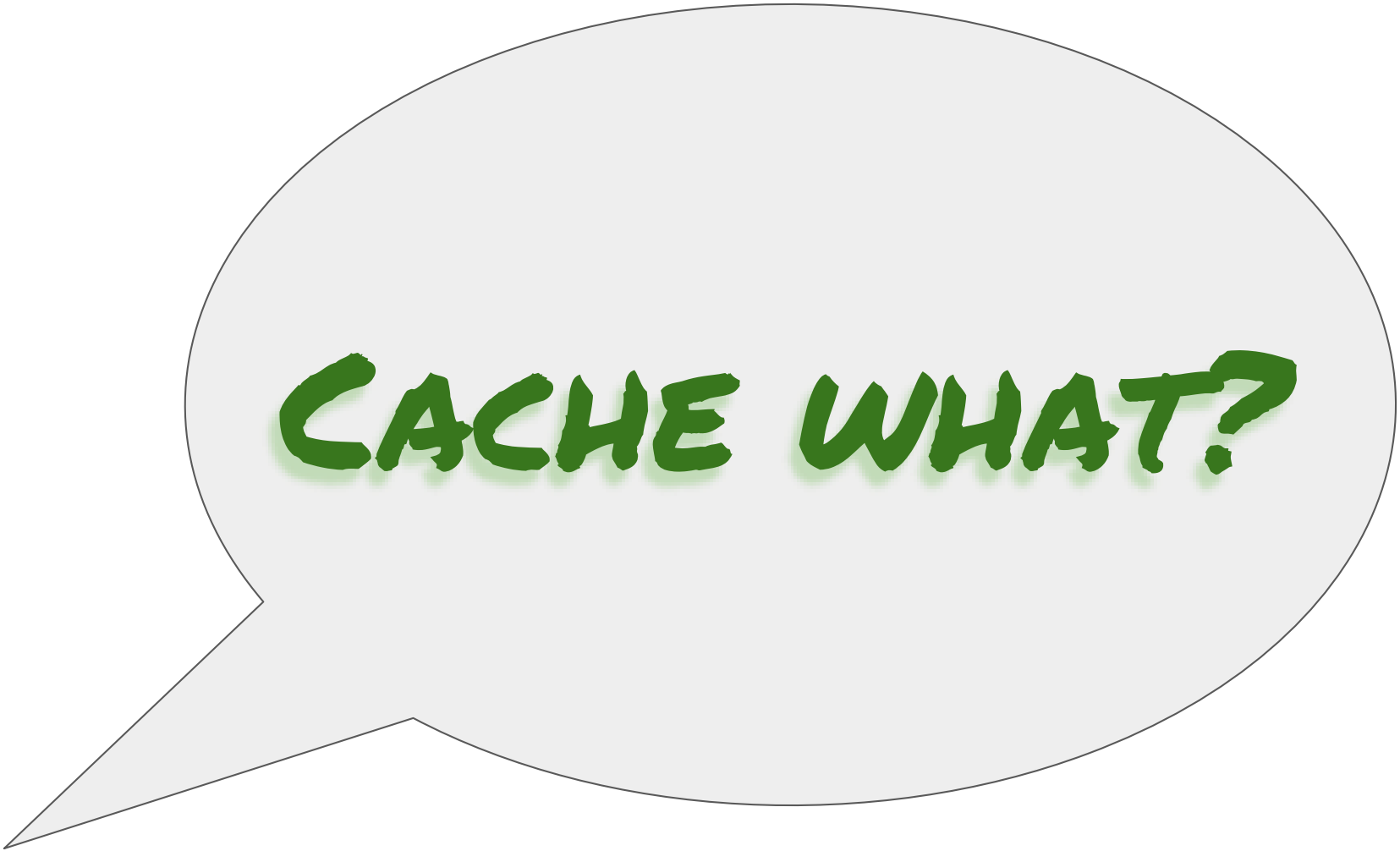
| Movement | | | |
|---|---|---|---|
| *Key* | *XTarget* | *YTarget* | *Speed* |
| 5 | 2 | 2 | 0.1 |
| 10 | 8 | 11 | 0.3 |
| 15 | 0 | 0 | 0.1 |
| 30 | 5 | 3 | 0.5 |

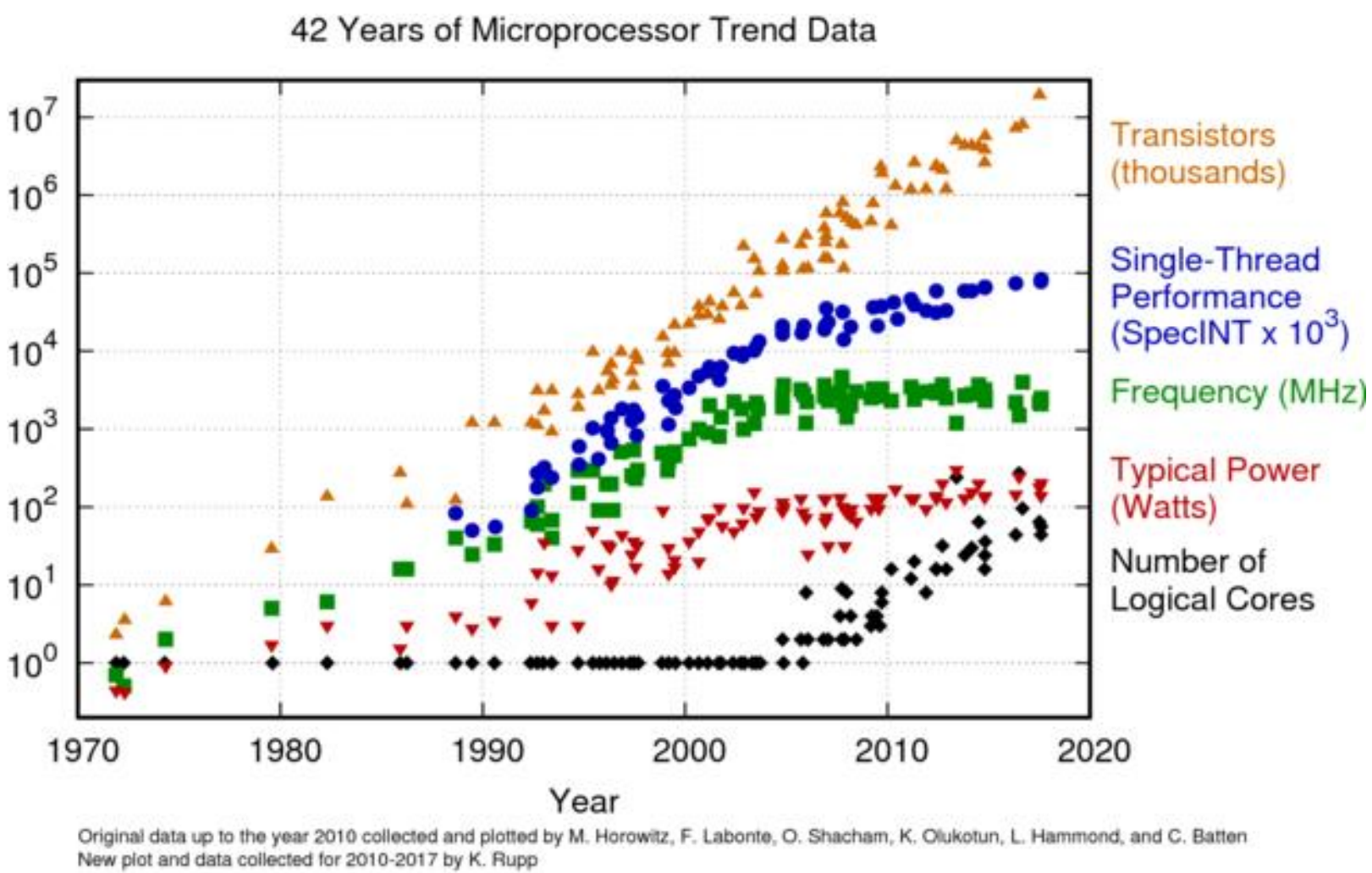
```
SELECT *
FROM Position
JOIN Movement
WHERE Position.Key=Movement.Key
```

| *Key* | *X* | *Y* | *XTarget* | *YTarget* | *Speed* |
|---|---|---|---|---|---|
| 10 | 7 | 12 | 8 | 11 | 0.3 |
| 30 | 3 | 3 | 5 | 3 | 0.5 |

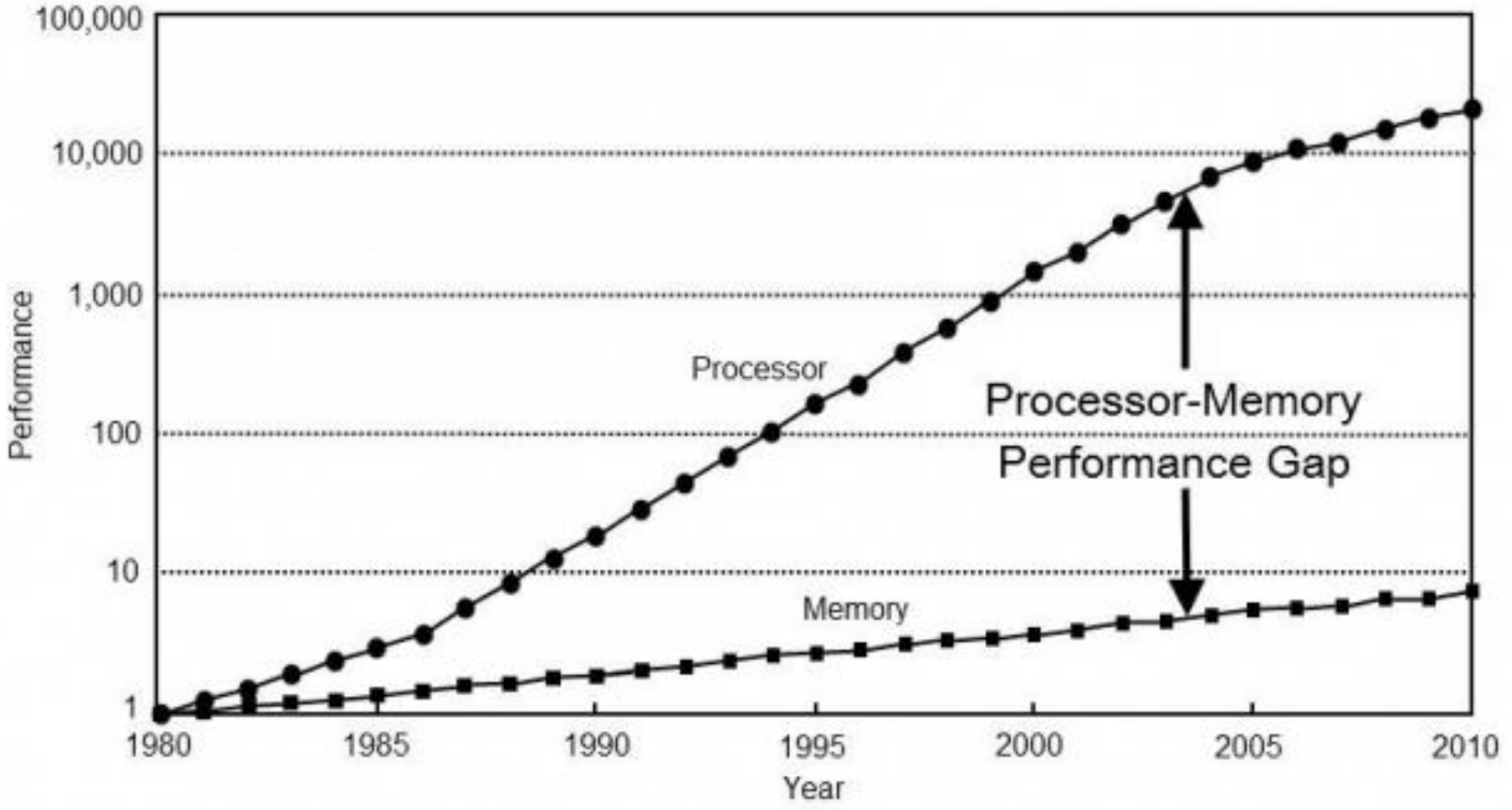
OUR CACHE WASTAGE :-(


# Let's talk about cache coherence

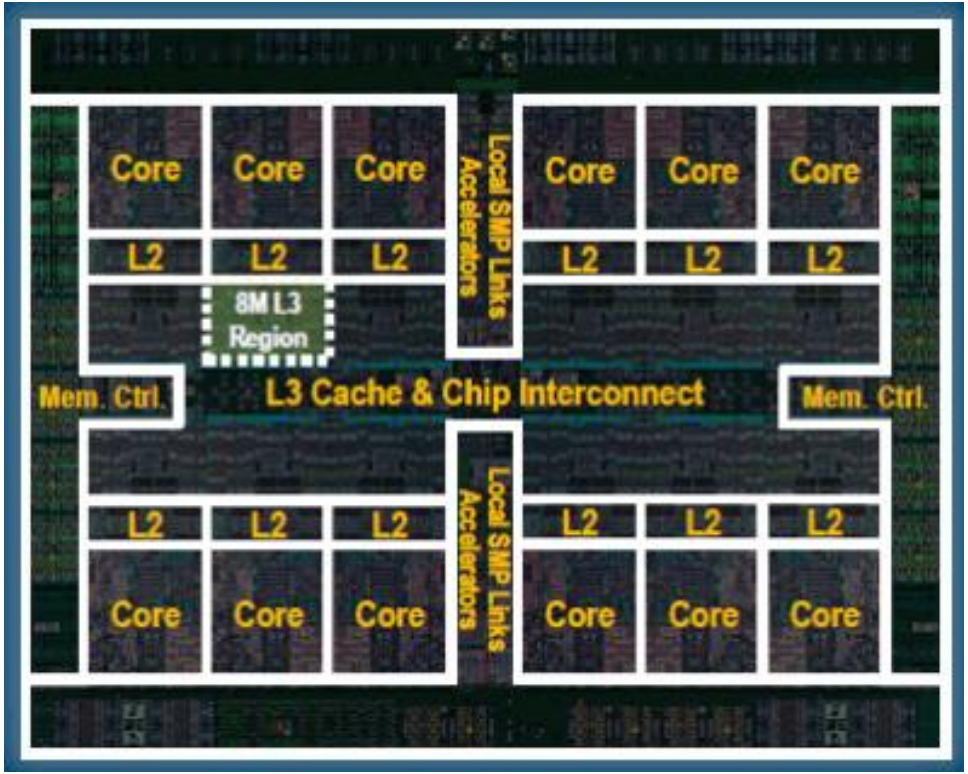

Cache what?




https://www.karlrupp.net/2018/02/42-years-of-microprocessor-trend-data/




https://www.extremetech.com/computing/261792-what-is-speculative-execution


Core
Core
Core
Local SMP Links Accelerators
Core
Core
Core
L2
L2
L2
L2
L2
L2
8M L3 Region
Mem. Ctrl.
L3 Cache & Chip Interconnect
Mem. Ctrl.
L2
L2
L2
Local SMP Links Accelerators
L2
L2
L2
Core
Core
Core
Core
Core
Core

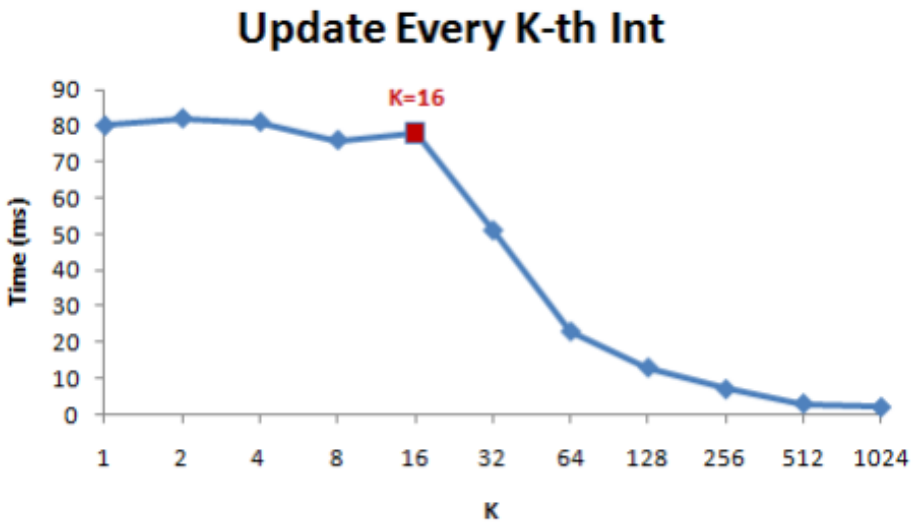
# Which is faster, loop 1 or 2?

```
int[] arr = new int[64 * 1024 * 1024];

// Loop 1
for (int i = 0; i < arr.Length; i++)
{
    arr[i] *= 3;
}

// Loop 2
for (int i = 0; i < arr.Length; i += 16)
{
    arr[i] *= 3;
}
```

http://igoro.com/archive/gallery-of-processor-cache-effects/

# Trick question, they're the same!

```
int[] arr = new int[64 * 1024 * 1024];

// Loop 1
for (int i = 0; i < arr.Length; i++)
{
    arr[i] *= 3;
}

// Loop 2
for (int i = 0; i < arr.Length; i += 16)
{
    arr[i] *= 3;
}
```




sizeof(int) is 4

16 x int is 64 bytes

64 bytes is one cache line

http://igoro.com/archive/gallery-of-processor-cache-effects/

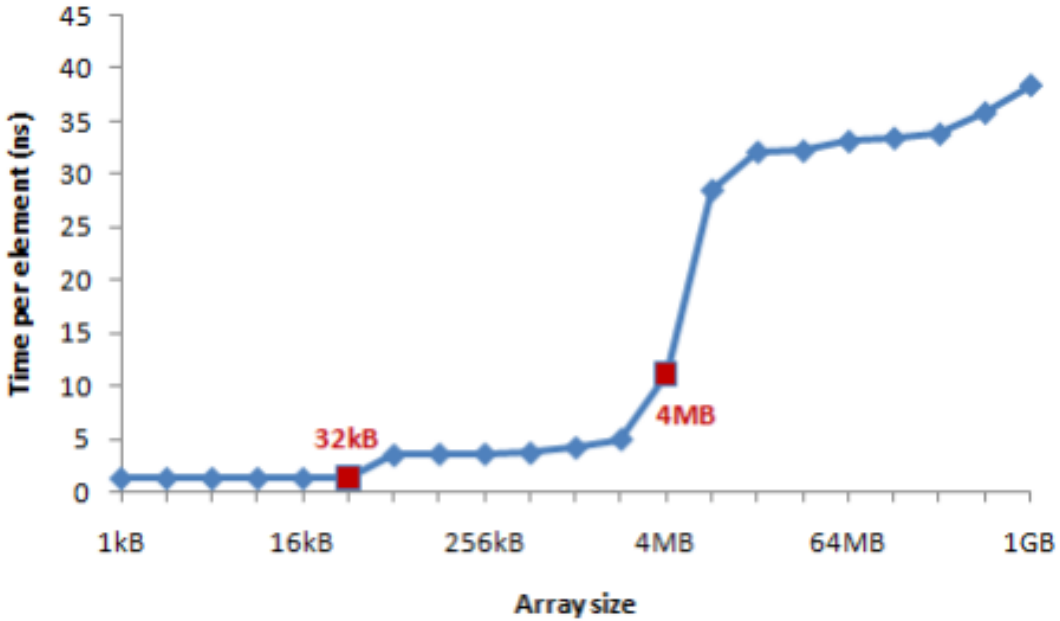
# How does array size affect speed?

```
int[] arr = new int[size];

for (int i = 0; i < 100000000; i++)
{
        // Touch each cache line
    arr[(i * 16) % size]++;
}
```

http://igoro.com/archive/gallery-of-processor-cache-effects/

# Well Hello L2 cache!

```
int[] arr = new int[size];

for (int i = 0; i < 100000000; i++)
{
        // Touch each cache line
    arr[(i * 16) % size]++;
}
```




http://igoro.com/archive/gallery-of-processor-cache-effects/

**Your hands are your registers.** It takes you 1 second to drop chocolate chips into the dough.

**The kitchen counter is your L1 cache, twelve times slower than registers.** It takes 12 x 1 = 12 seconds to step to the counter, pick up the bag of walnuts, and empty some into your hand.

**The fridge is your L2 cache, four times slower than L1.** It takes 4 x 12 = 48 seconds to walk to the fridge, open it, move last night's leftovers out of the way, take out a carton of eggs, open the carton, put 3 eggs on the counter, and put the carton back in the fridge.

**The cupboard is your L3 cache, three times slower than L2.** It takes 3 x 48 = 2 minutes and 24 seconds to take three steps to the cupboard, bend down, open the door, root around to find the baking supply tin, extract it from the cupboard, open it, dig to find the baking powder, put it on the counter and sweep up the mess you spilled on the floor.

**And main memory? That's the corner store, 5 times slower than L3.**

# Stop sending the CPU out for groceries!

https://stackoverflow.com/questions/1126529/what-is-the-cost-of-an-l1-cache-miss/29188516#29188516


TL;DR Multithreading and cache coherence are crucial to performance

# Unity ECS Archetypes

*Position-Movement Archetype*

| | Position | | Movement | | |
|---|---|---|---|---|---|
| *Index* | *X* | *Y* | *XTarget* | *YTarget* | *Speed* |
| *0* | 7 | 12 | 8 | 11 | 0.3 |
| *1* | 3 | 3 | 5 | 3 | 0.5 |

*Position Archetype*

| | Position | |
|---|---|---|
| *Index* | *X* | *Y* |
| *0* | 1 | 2 |
| *1* | 3 | 4 |
| *2* | 5 | 6 |
| *3* | 7 | 8 |

*Movement Archetype*

| | Movement | | |
|---|---|---|---|
| *Index* | *XTarget* | *YTarget* | *Speed* |
| *0* | 2 | 2 | 0.1 |
| *1* | 0 | 0 | 0.1 |

# Unity ECS Archetypes

No cache wastage! :-)
(also no keys)

*Position-Movement Archetype*

| | Position | | Movement | | |
|---|---|---|---|---|---|
| *Index* | *X* | *Y* | *XTarget* | *YTarget* | *Speed* |
| *0* | 7 | 12 | 8 | 11 | 0.3 |
| *1* | 3 | 3 | 5 | 3 | 0.5 |

*Position Archetype*

| | Position | |
|---|---|---|
| *Index* | *X* | *Y* |
| *0* | 1 | 2 |
| *1* | 3 | 4 |
| *2* | 5 | 6 |
| *3* | 7 | 8 |

*Movement Archetype*

| | Movement | | |
|---|---|---|---|
| *Index* | *XTarget* | *YTarget* | *Speed* |
| *0* | 2 | 2 | 0.1 |
| *1* | 0 | 0 | 0.1 |

*Position-Movement Archetype*

| | Position | | Movement | | |
|---|---|---|---|---|---|
| *Index* | *X* | *Y* | *XTarget* | *YTarget* | *Speed* |
| *0* | 7 | 12 | 8 | 11 | 0.3 |
| *1* | 3 | 3 | 5 | 3 | 0.5 |

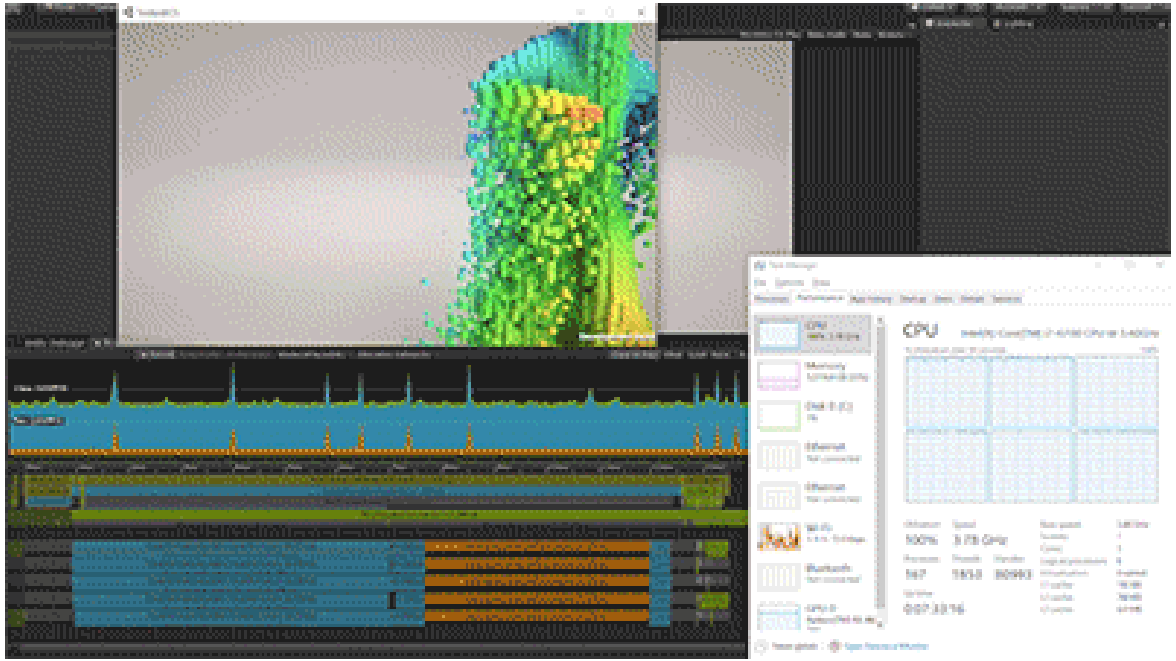
# ALTHOUGH ...

## CACHE WASTAGE :-(

*Position-Movement-Colour Archetype*

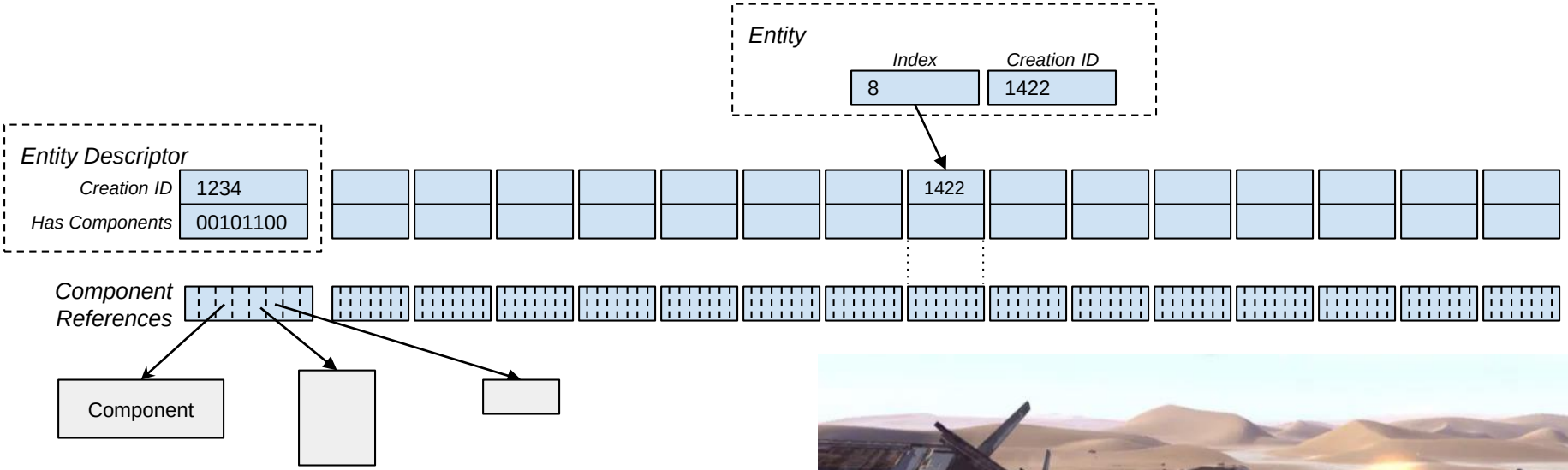
| | Position | | Movement | | | Colour | | |
|---|---|---|---|---|---|---|---|---|
| *Index* | *X* | *Y* | *XTarget* | *YTarget* | *Speed* | *Red* | *Green* | *Blue* |
| *0* | 5 | 5 | 8 | 8 | 0.5 | 0.2 | 0.9 | 0.1 |
| *1* | 42 | 24 | 24 | 42 | 0.42 | 0.9 | 0.0 | 0.0 |

# Unity C# Job System

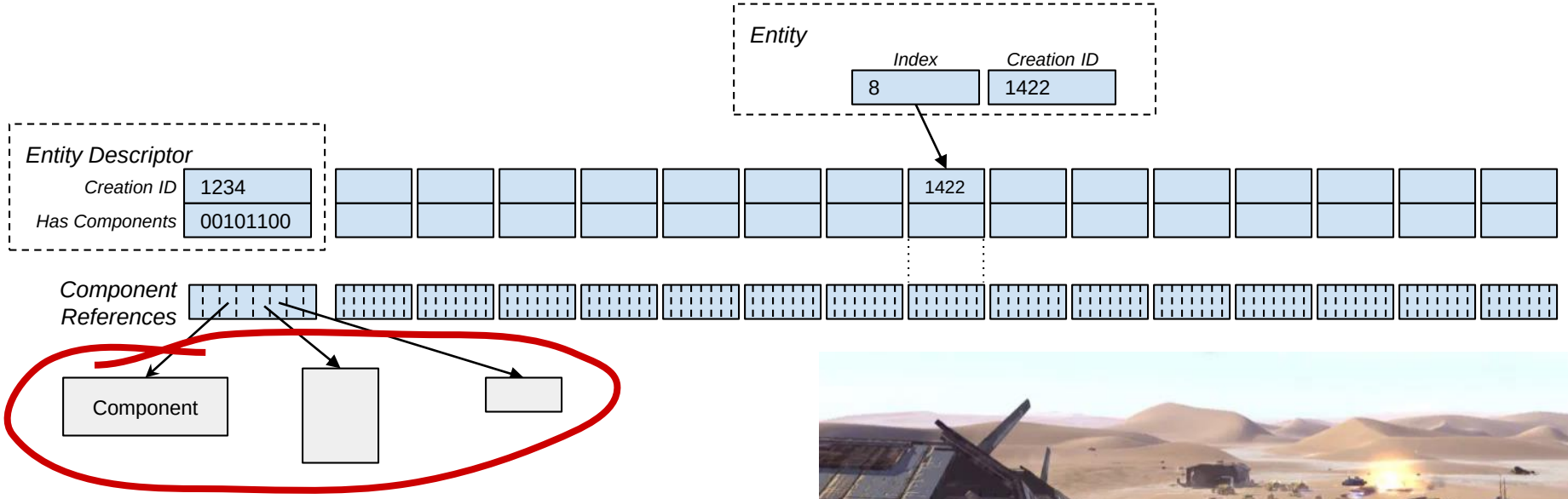


# ECS @ BBI

# Homeworld:Deserts of Kharak ECS




# Homeworld:Deserts of Kharak ECS




NOT CACHE FRIENDLY!

# Hardspace:Shipbreaker ECS

# Hardspace:Shipbreaker ECS

# Homeworld 3 ECS

# Homeworld 3 ECS

EnTT

GAMING MEETS MODERN C++

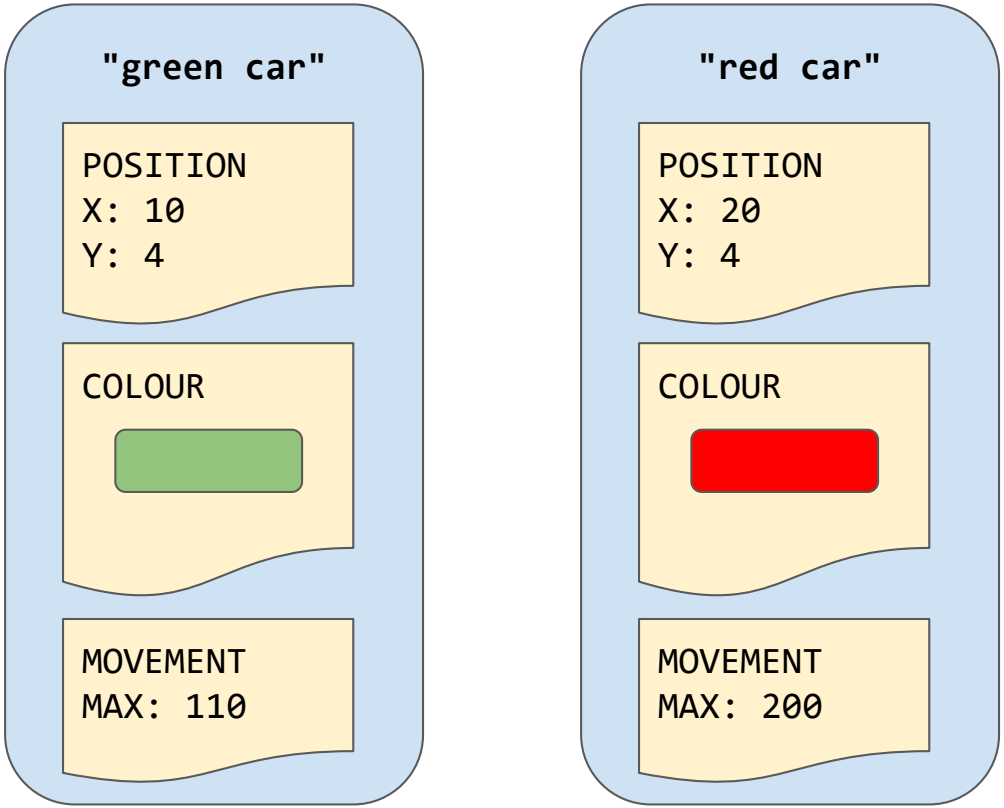
# ECS, NOT JUST FOR PROGRAMMERS!

- ASPECT-ORIENTED DESIGN
- GREAT WAY TO BREAK DOWN THE PROBLEM DOMAIN AND EVERYTHING IN IT
- HELPS YOU TO SEE THE COMMON AND UNIQUE CHARACTERISTICS OF THINGS
- ENCOURAGES DATA-DRIVEN DESIGN, EMPOWERING CREATORS
- ASPECTS CAN BE SHARED AND REUSED IN CREATIVE WAYS

Deserts of Kharak tornado

*image from gamespot.com/homeworld-deserts-of-kharak/user-reviews*

# Entity blueprints: designer-tunable specifications




WHAT IS ECS?


# Cache-friendly & multithreaded, that's what!


Good for programmers

Good for performance

Good for designers


tinyurl.com/ecsatbbi
linkedin.com/in/yossarianking

BLACKBIRD INTERACTIVE

# FURTHER READING

[Entity Systems are the future of MMOG development (2007)]

[EnTT - a fast and reliable C++ entity-component system] (and [skypjack blog])

[ECSQL - Vancouver Unity Meetup May 2019]

[A gallery of processor cache effects]

[github.com/Kobzol/hardware-effects]