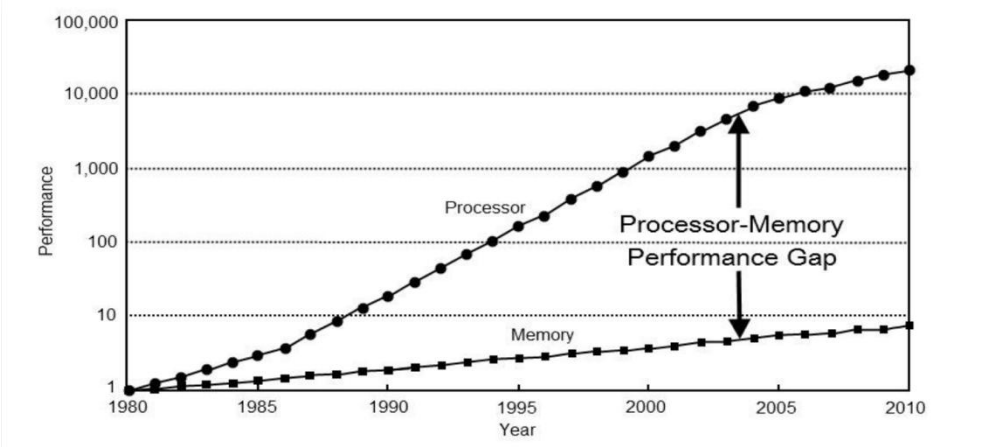
Introduction to Data-Oriented Design

So what is this Data-Oriented Design?

It’s about on shifting focus to how data is read and written

Why should we care?

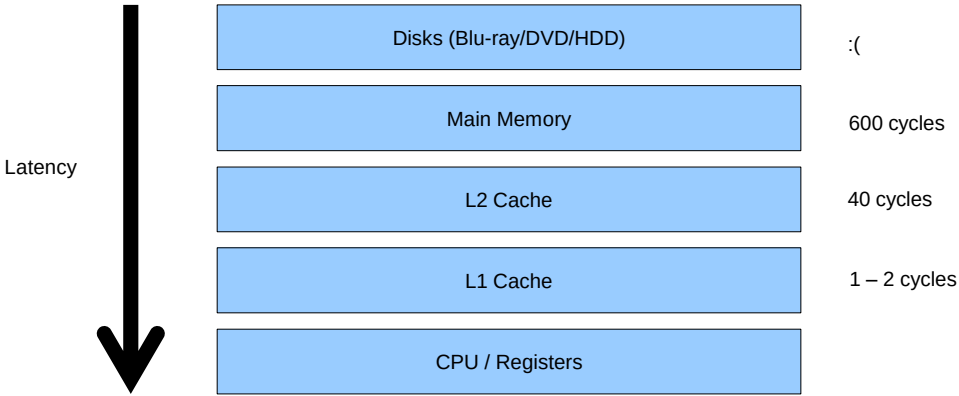
Performance



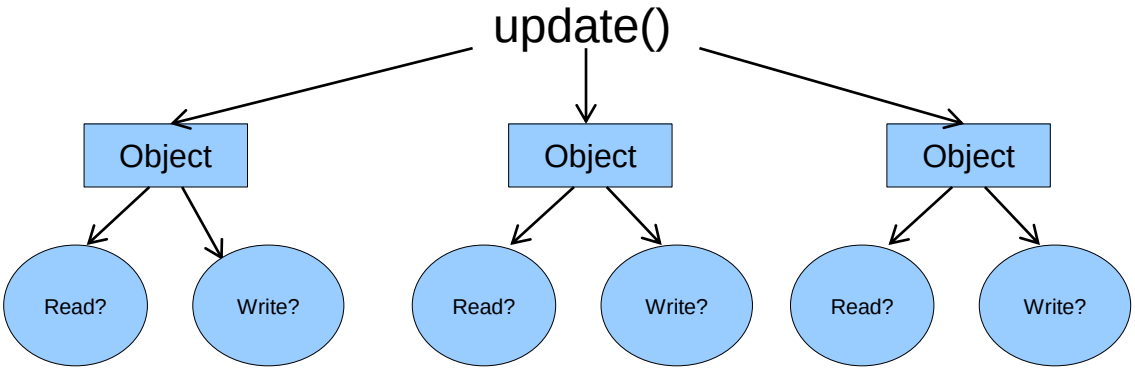
A read from memory takes ~600 cycles at 3.2 GHz

A read from memory takes 40 cycles at 300 MHz

Performance

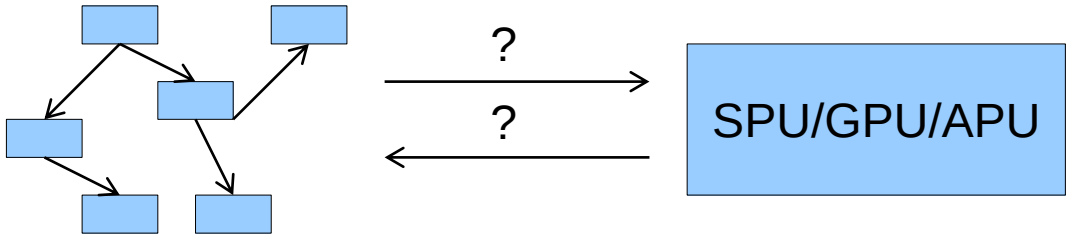


Multithreading



- **Cannot** multithread without knowing how data is touched
- Adding locks always protects **data** not **code**

Offloading to co-unit



- If data is unknown **hard/impossible** to run on co-unit

Better design

- Data focus *can* lead to isolated, self-contained, interchangeable pieces of data and code
- This *can* make it easier to test data and code in isolation

Example - OOD

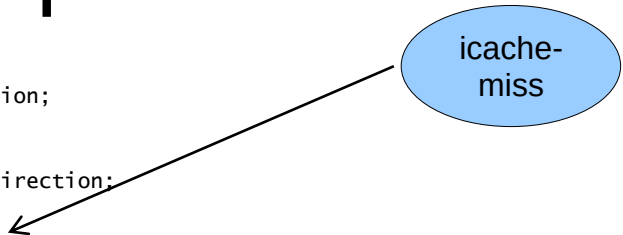
```
class Bot
{
   ...
   Vec3 m_position;
   ...
   float m_mod;
   ...
   float m_aimDirection;
   ...

   void updateAim(Vec3 target)
   {
     m_aimDirection = dot3(m_position, target) * m_mod;
   }
}
```

Example - OOD

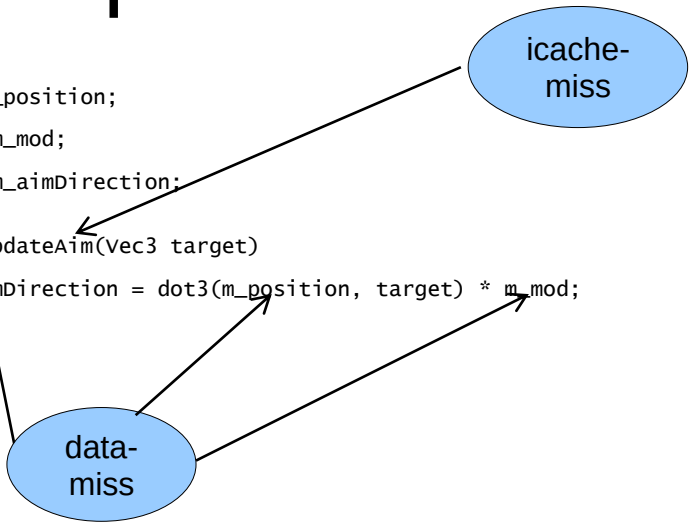
```
class Bot
{
   ...
   Vec3 m_position;
   ...
   float m_mod;
   ...
   float m_aimDirection;
   ...

   void updateAim(Vec3 target)
   {
     m_aimDirection = dot3(m_position, target) * m_mod;
   }
}
```

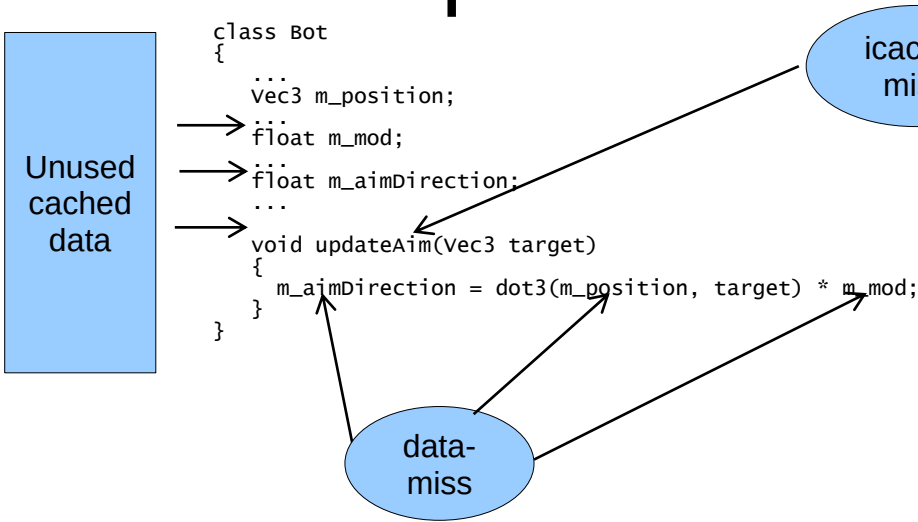


Example - OOD

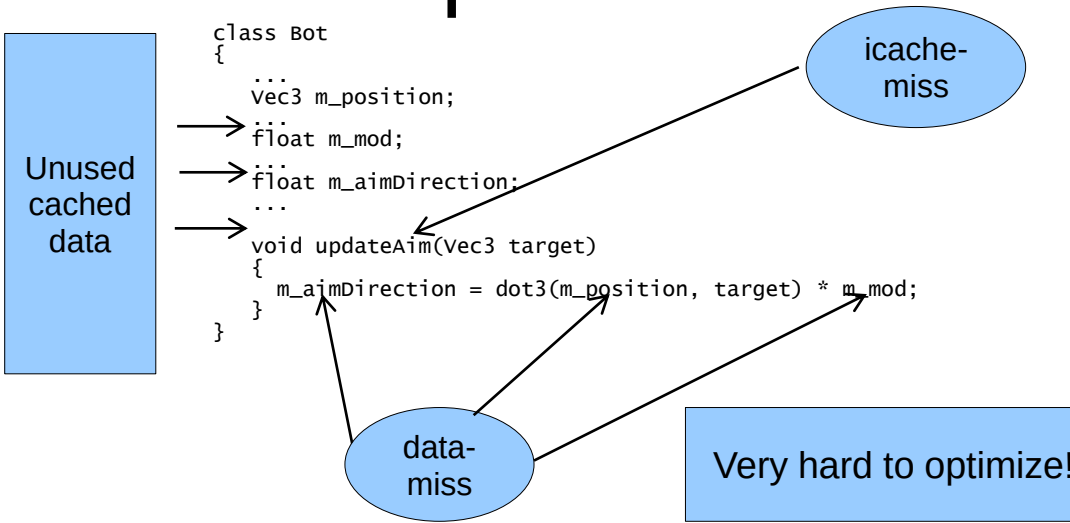
```
class Bot
{
   ...
   Vec3 m_position;
   ...
   float m_mod;
   ...
   float m_aimDirection;
   ...
   void updateAim(Vec3 target)
   {
     m_aimDirection = dot3(m_position, target) * m_mod;
   }
}
```



Example - OOD


Unused cached data
class Bot
{
...
Vec3 m_position;
...
float m_mod;
...
float m_aimDirection;
...
void updateAim(Vec3 target)
{
m_aimDirection = dot3(m_position, target) * m_mod;
}
}
icache-miss
data-miss

Example - OOD


Unused cached data
class Bot
{
...
Vec3 m_position;
...
float m_mod;
...
float m_aimDirection;
...
void updateAim(Vec3 target)
{
m_aimDirection = dot3(m_position, target) * m_mod;
}
}
icache-miss
data-miss
Very hard to optimize!

Example - OOD

Lets say we call this code 4 times (4 diffrent Bots)

```
void updateAim(Vec3 target)
{
  m_aimDirection = dot3(m_position, target) * m_mod;
}
```

Example - OOD

Lets say we call this code 4 times (4 diffrent Bots)

```
void updateAim(Vec3 target)
{
  m_aimDirection = dot3(m_position, target) * m_mod;
}
```

iCache – 600

Example - OOD

Lets say we call this code 4 times (4 diffrent Bots)

```
void updateAim(Vec3 target)
{
  m_aimDirection = dot3(m_position, target) * m_mod;
}
```

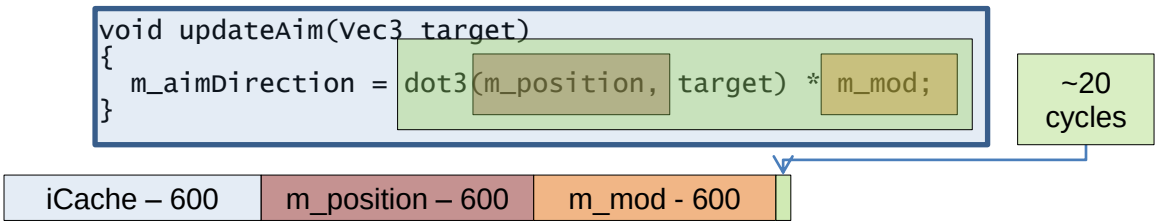
Example - OOD

Lets say we call this code 4 times (4 diffrent Bots)

```
void updateAim(Vec3 target)
{
  m_aimDirection = dot3(m_position, target) * m_mod;
}
```

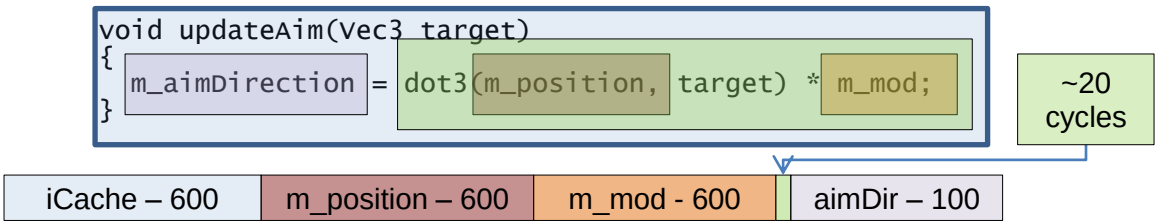
Example - OOD

Lets say we call this code 4 times (4 diffrent Bots)



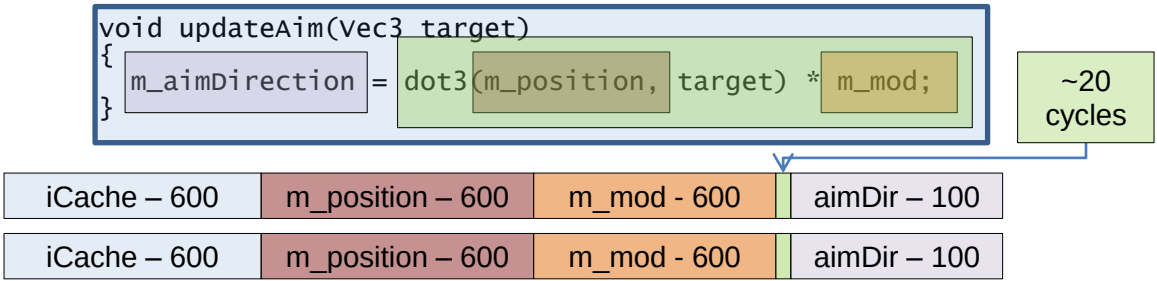
Example - OOD

Lets say we call this code 4 times (4 diffrent Bots)



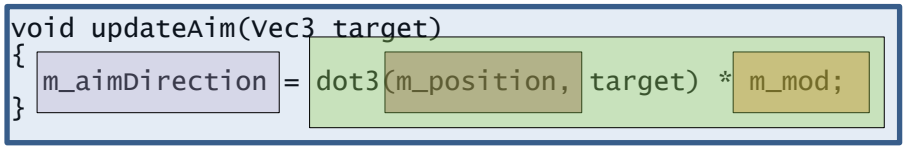
Example - OOD

Lets say we call this code 4 times (4 diffrent Bots)



Example - OOD

Lets say we call this code 4 times (4 diffrent Bots)

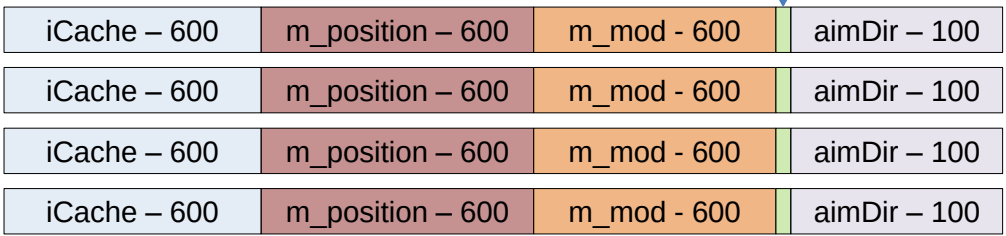


Example - OOD

Lets say we call this code 4 times (4 diffrent Bots)

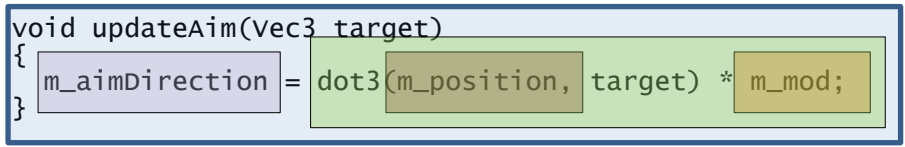
```
void updateAim(Vec3 target)
{
  m_aimDirection = dot3(m_position, target) * m_mod;
}
```

~20 cycles



Example - OOD

Lets say we call this code 4 times (4 diffrent Bots)



~20 cycles

iCache – 600	m_position – 600	m_mod - 600	aimDir – 100
iCache – 600	m_position – 600	m_mod - 600	aimDir – 100
iCache – 600	m_position – 600	m_mod - 600	aimDir – 100
iCache – 600	m_position – 600	m_mod - 600	aimDir – 100

7680

Example - DOD

Example - DOD

- Design ”back to front” and focus on the output data

Example - DOD

- Design ”back to front” and focus on the output data
- Then add the *minimal* amount of data needed to do the transform to create the correct output

Example - DOD

```
void updateAims(float* aimDir,const AimingData* aim,
                Vec3 target, uint count)
{
  for (uint i = 0; i < count; ++i)
  {
    aimDir[i] = dot3(aim->positions[i],target) * aim->mod[i];
  }
}
```

Example - DOD

```
void updateAims(float* aimDir,const AimingData* aim,
                Vec3 target, uint count)
{
  for (uint i = 0; i < count; ++i)
  {
    aimDir[i] = dot3(aim->positions[i],target) * aim->mod[i];
  }
}
```

What has changed?

Example - DOD

```
void updateAims(float* aimDir,const AimingData* aim,
                Vec3 target, uint count)
{
  for (uint i = 0; i < count; ++i)
  {
    aimDir[i] = dot3(aim->positions[i],target) * aim->mod[i];
  }
}
```

What has changed?

Only read needed inputs

Example - DOD

```
void updateAims(float* aimDir,const AimingData* aim,
                Vec3 target, uint count)
{
  for (uint i = 0; i < count; ++i)
  {
    aimDir[i] = dot3(aim->positions[i],target) * aim->mod[i];
  }
}
```

What has changed?

Only read needed inputs	Write to linear array

Example - DOD

```
void updateAims(float* aimDir,const AimingData* aim,
                Vec3 target, uint count)
{
  for (uint i = 0; i < count; ++i)
  {
    aimDir[i] = dot3(aim->positions[i],target) * aim->mod[i];
  }
}
```

What has changed?

Only read needed inputs	Write to linear array
Loop over all the data	

Example - DOD

```
void updateAims(float* aimDir,const AimingData* aim,
                Vec3 target, uint count)
{
  for (uint i = 0; i < count; ++i)
  {
    aimDir[i] = dot3(aim->positions[i],target) * aim->mod[i];
  }
}
```

What has changed?

Only read needed inputs	Write to linear array
Loop over all the data	Actual code unchanged

Example - DOD

```
void updateAims(float* aimDir,const AimingData* aim,
                Vec3 target, uint count)
{
  for (uint i = 0; i < count; ++i)
  {
    aimDir[i] = dot3(aim->positions[i],target) * aim->mod[i];
  }
}
```

What has changed?

Only read needed inputs	Write to linear array
Loop over all the data	Actual code unchanged
Code separated	

Example - DOD

```
void updateAims(float* aimDir, const AimingData* aim,
                Vec3 target, uint count)
{
  for (uint i = 0; i < count; ++i)
  {
    aimDir[i] = dot3(aim->positions[i], target) * aim->mod[i];
  }
}
```

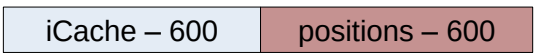
Example - DOD

```
void updateAims(float* aimDir, const AimingData* aim,
                Vec3 target, uint count)
{
  for (uint i = 0; i < count; ++i)
  {
    aimDir[i] = dot3(aim->positions[i], target) * aim->mod[i];
  }
}
```

iCache – 600

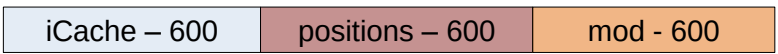
Example - DOD

```
void updateAims(float* aimDir, const AimingData* aim,
                Vec3 target, uint count)
{
  for (uint i = 0; i < count; ++i)
  {
    aimDir[i] = dot3(aim->positions[i], target) * aim->mod[i];
  }
}
```



Example - DOD

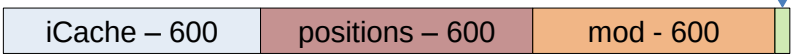
```
void updateAims(float* aimDir, const AimingData* aim,
                Vec3 target, uint count)
{
  for (uint i = 0; i < count; ++i)
  {
    aimDir[i] = dot3(aim->positions[i], target) * aim->mod[i];
  }
}
```



Example - DOD

```
void updateAims(float* aimDir, const AimingData* aim,
                Vec3 target, uint count)
{
  for (uint i = 0; i < count; ++i)
  {
    aimDir[i] = dot3(aim->positions[i], target) * aim->mod[i];
  }
}
```

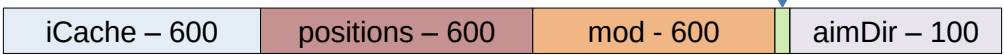
~20 cycles



Example - DOD

```
void updateAims(float* aimDir, const AimingData* aim,
                Vec3 target, uint count)
{
  for (uint i = 0; i < count; ++i)
  {
    aimDir[i] = dot3(aim->positions[i], target) * aim->mod[i];
  }
}
```

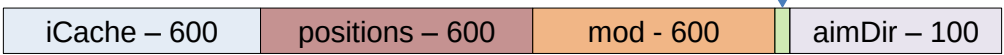
~20 cycles



Example - DOD

```
void updateAims(float* aimDir, const AimingData* aim,
                Vec3 target, uint count)
{
  for (uint i = 0; i < count; ++i)
  {
    aimDir[i] = dot3(aim->positions[i], target) * aim->mod[i];
  }
}
```

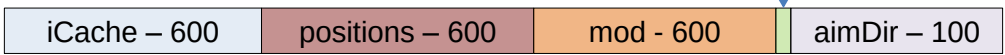
~20 cycles



Example - DOD

```
void updateAims(float* aimDir, const AimingData* aim,
                Vec3 target, uint count)
{
  for (uint i = 0; i < count; ++i)
  {
    aimDir[i] = dot3(aim->positions[i], target) * aim->mod[i];
  }
}
```

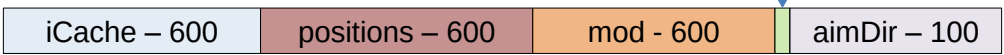
~20 cycles



Example - DOD

```
void updateAims(float* aimDir, const AimingData* aim,
                Vec3 target, uint count)
{
  for (uint i = 0; i < count; ++i)
  {
    aimDir[i] = dot3(aim->positions[i], target) * aim->mod[i];
  }
}
```

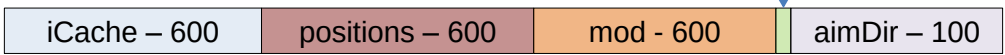
~20 cycles



Example - DOD

```
void updateAims(float* aimDir, const AimingData* aim,
                Vec3 target, uint count)
{
  for (uint i = 0; i < count; ++i)
  {
    aimDir[i] = dot3(aim->positions[i], target) * aim->mod[i];
  }
}
```

~20 cycles



1980

Data layout OOD vs DOD

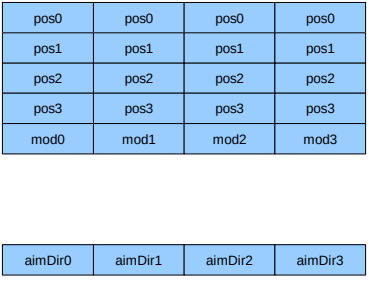
pos0	pos0	pos0	pos0
mod0			
aimDir0			

Pos1			
mod1			
aimDir1			

pos0	pos0	pos0	pos0
pos1	pos1	pos1	pos1
pos2	pos2	pos2	pos2
pos3	pos3	pos3	pos3
mod0	mod1	mod2	mod3

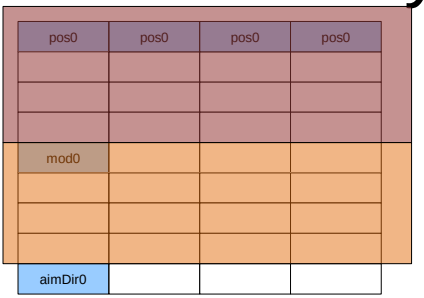
aimDir0	aimDir1	aimDir2	aimDir3

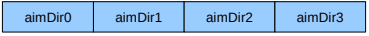
Data layout OOD vs DOD

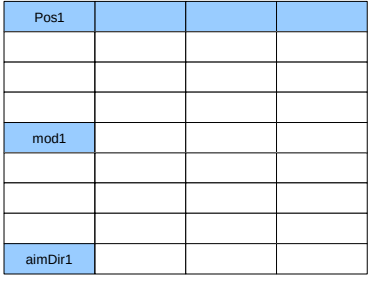


Each color block is one 128 byte cache line

Data layout OOD vs DOD

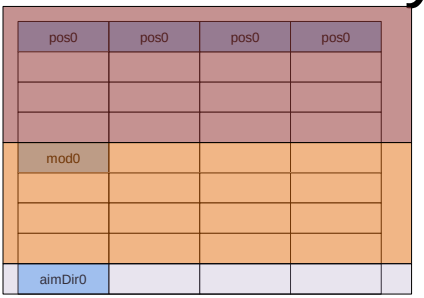






Each color block is one 128 byte cache line

Data layout OOD vs DOD



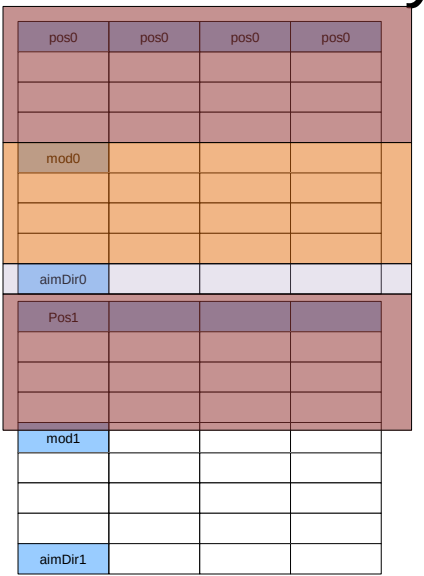
pos0	pos0	pos0	pos0
pos1	pos1	pos1	pos1
pos2	pos2	pos2	pos2
pos3	pos3	pos3	pos3
mod0	mod1	mod2	mod3

aimDir0	aimDir1	aimDir2	aimDir3

Pos1			
mod1			
aimDir1			

Each color block is one 128 byte cache line

Data layout OOD vs DOD

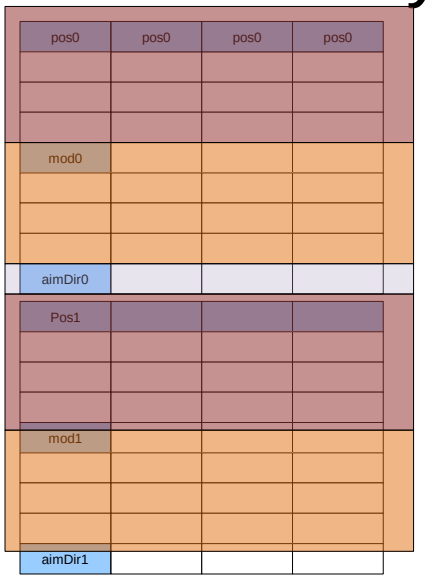


pos0	pos0	pos0	pos0
pos1	pos1	pos1	pos1
pos2	pos2	pos2	pos2
pos3	pos3	pos3	pos3
mod0	mod1	mod2	mod3

aimDir0	aimDir1	aimDir2	aimDir3

Each color block is one 128 byte cache line

Data layout OOD vs DOD



pos0	pos0	pos0	pos0
pos1	pos1	pos1	pos1
pos2	pos2	pos2	pos2
pos3	pos3	pos3	pos3
mod0	mod1	mod2	mod3

aimDir0	aimDir1	aimDir2	aimDir3

Each color block is one 128 byte cache line

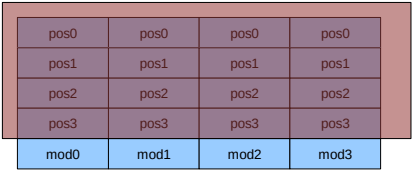
Data layout OOD vs DOD

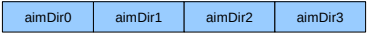
pos0	pos0	pos0	pos0
pos1	pos1	pos1	pos1
pos2	pos2	pos2	pos2
pos3	pos3	pos3	pos3
mod0	mod1	mod2	mod3

aimDir0	aimDir1	aimDir2	aimDir3

Each color block is one 128 byte cache line

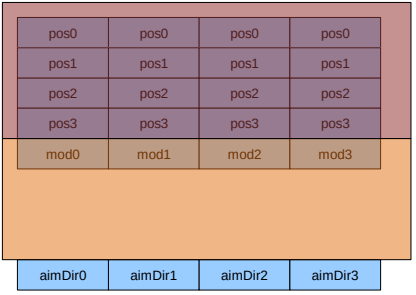
Data layout OOD vs DOD





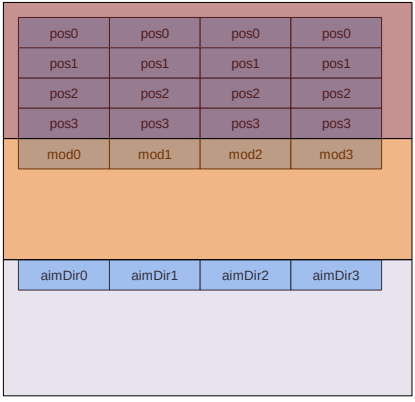
Each color block is one 128 byte cache line

Data layout OOD vs DOD



Each color block is one 128 byte cache line

Data layout OOD vs DOD



Each color block is one 128 byte cache line

Its all about memory

Its all about memory

- Optimize for data first then code

Its all about memory

- Optimize for data first then code
- Most code is likely bound by memory access

Its all about memory

- Optimize for data first then code
- Most code is likely bound by memory access
- Not everything needs to be an object

Remember

Remember

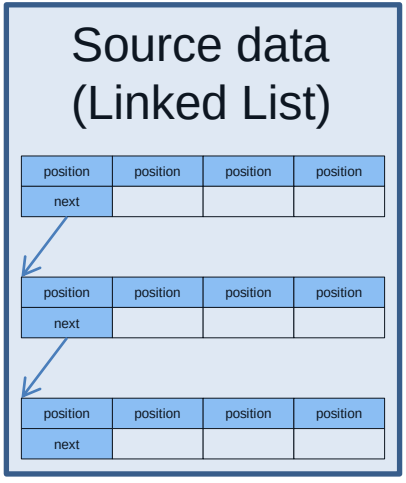
- We are doing games, we know our data.

Remember

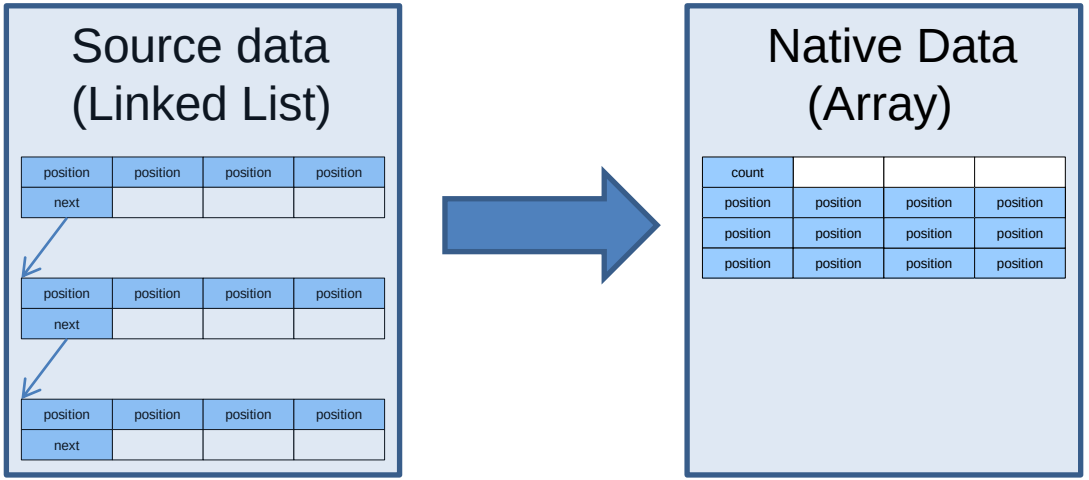
- We are doing games, we know our data.
- Pre-format. Source data and native data doesn’t need to be the same

Example: Area Triggers

Example: Area Triggers

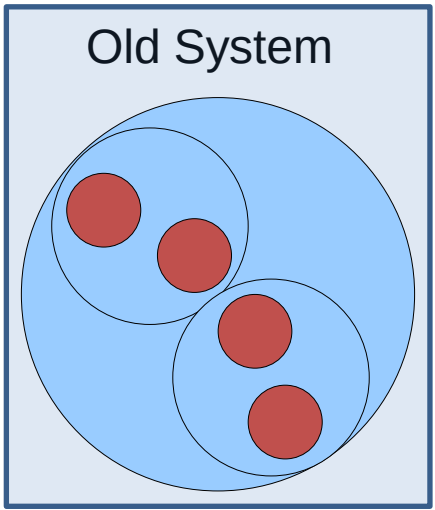


Example: Area Triggers

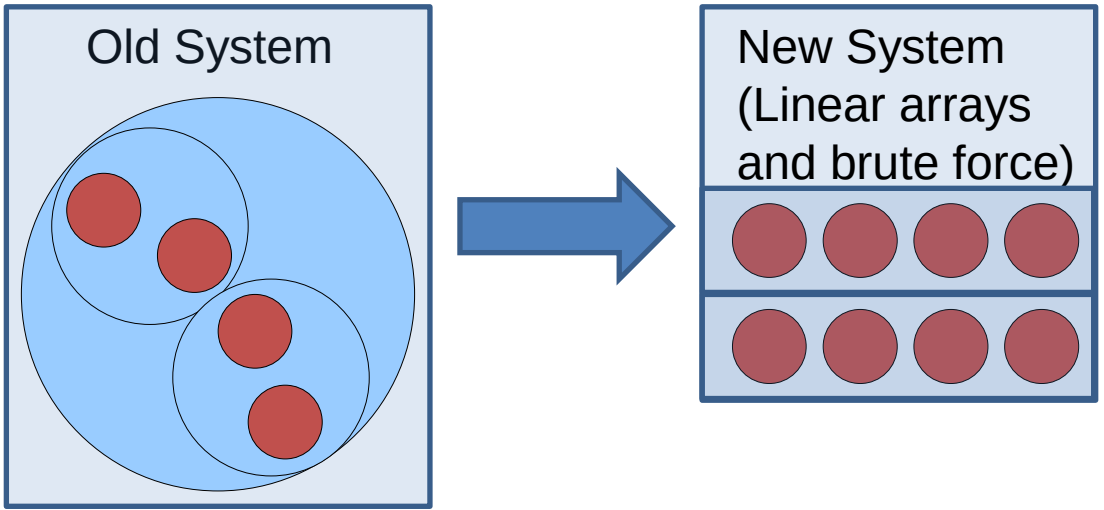


Example: Culling System

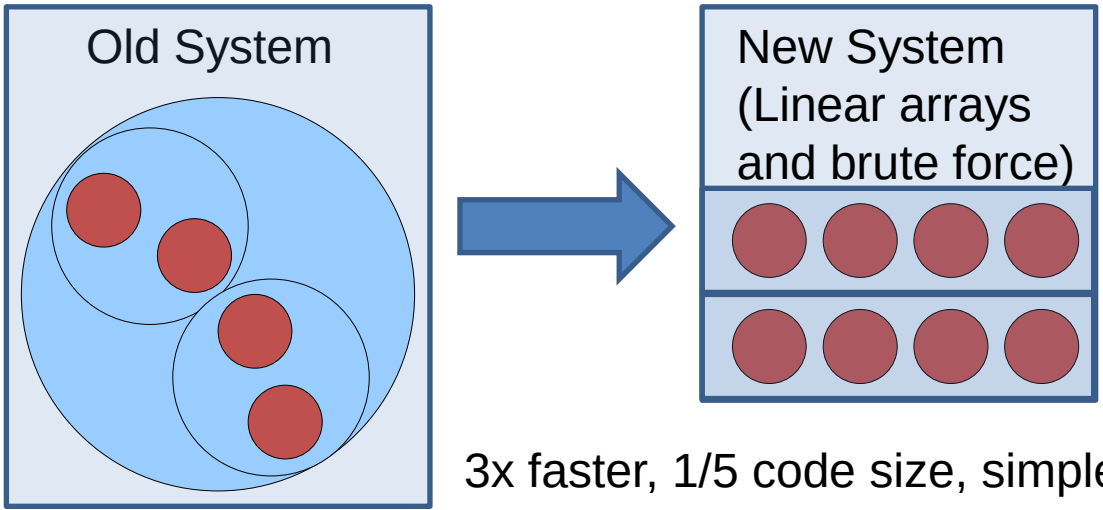
Example: Culling System



Example: Culling System



Example: Culling System

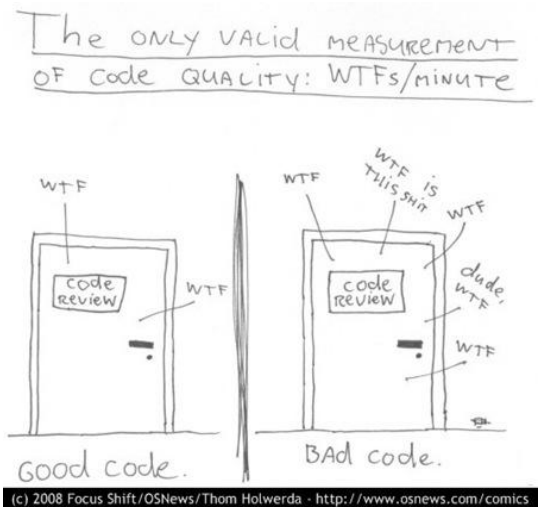


3x faster, 1/5 code size, simpler

Data Oriented Design Delivers:

Better Performance

Often simpler code



More parallelizable code

Questions?

Links

- **Data-Oriented Design (Or Why You Might Be Shooting Yourself in The Foot With OOP)**
  http://gamesfromwithin.com/data-oriented-design
- **Practical Examples in Data Oriented Design**
  http://bitsquid.blogspot.com/2010/05/practical-examples-in-data-oriented.html
- **The Latency Elephant**
  http://seven-degrees-of-freedom.blogspot.com/2009/10/latency-elephant.html
- **Pitfalls of Object Oriented Programming**
  http://seven-degrees-of-freedom.blogspot.com/2009/12/pitfalls-of-object-oriented-programming.html
- **Insomniac R&D**
  http://www.insomniacgames.com/research_dev
- **CellPerformance**

Image credits

- Cat image: http://icanhascheezburger.com/2007/06/24/uninterested-cat
  photo by: Arinn capped and submitted by: Andy
- Playstation 3 and Playstation 2 Copyright to Sony Computer Entertainment
- Xbox 360 Image Copyright to Microsoft
- “WTF” Code quality image: Copyright by **Thom Holwerda**
  http://www.osnews.com/comics