

DATA - ORIENTED
DESIGN
(for math)

@Mike-acton

anti-math
talk?

WHAT IS
DATA-ORIENTED?

Practical
philosophy
not
methodology.

Way to
approach
problem
solving.

Basic
Principles
—

Know The
data

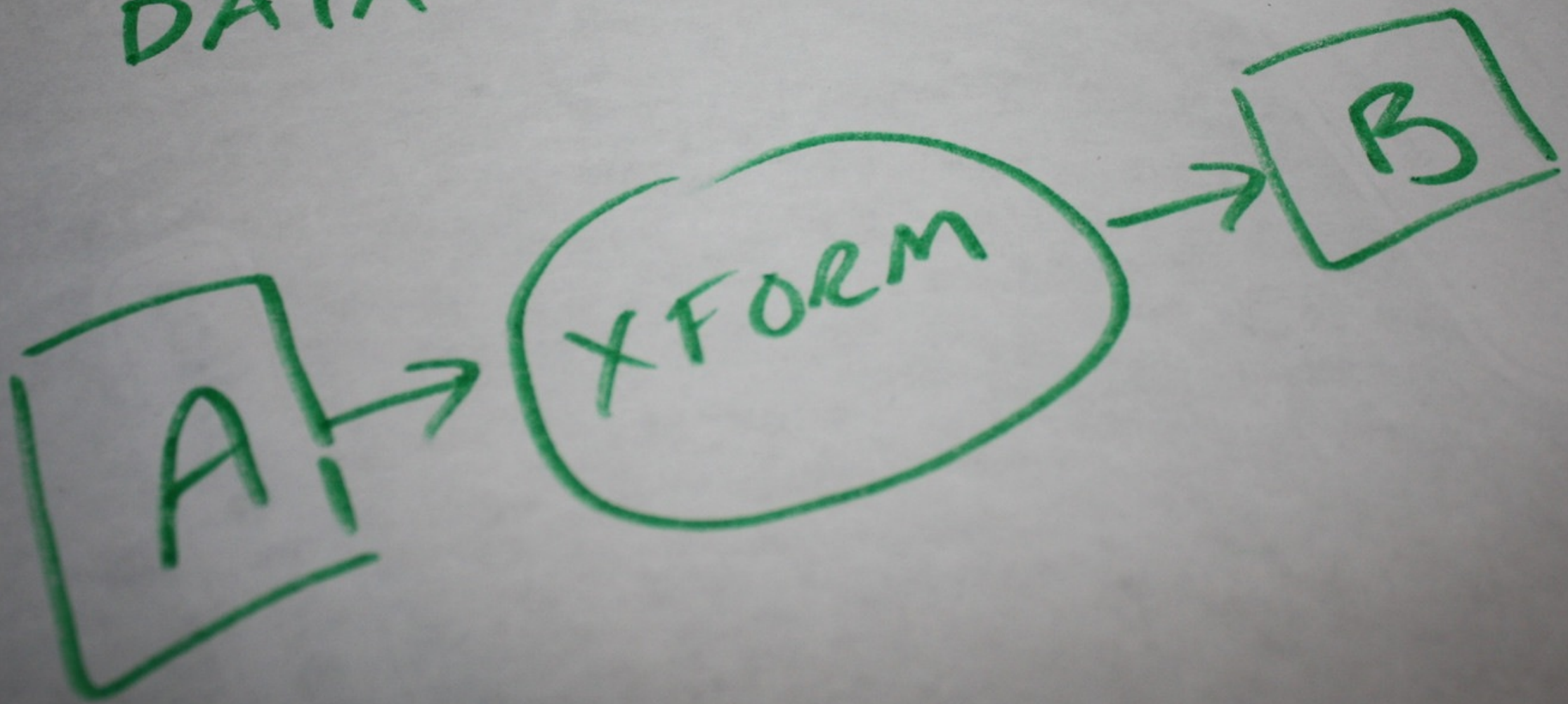
Know
The
context

Data >
Code

Nothing
new.

abandoning
Crap.

ONLY PURPOSE OF ANY
CODE IS TO TRANSFORM
DATA.



PROGRAMMER IS
FUNDAMENTALLY RESPONSIBLE
FOR THE DATA, NOT THE
CODE.

ONLY WRITE CODE THAT
HAS DIRECT, PROVABLE VALUE.

i.e. transforms DATA
IN MEANINGFUL WAY

CODE IS JUST A
DATA DESCRIPTION.

(AND IS JUST DATA.)

UNDERSTAND THE DATA
TO UNDERSTAND THE
PROBLEM.

THERE IS NO
IDEAL, ABSTRACT
SOLUTION TO THE
PROBLEM.

YOU CAN'T

"FUTURE

PROOF"

Common Misconceptions

All about
SMD

-hout

ONLY ABOUT
PERFORMANCE

OPTIMIZABILITY

CAN BE VS.

IMPOSSIBLE.

-want

Harder to
understand,
Maintain, etc.

Very likely,
you already

do it (to
some degree)

But it's
not the
exception

Lights

20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42

Shaders

particles

particles
animation

Calligraphy.

What do
we take
away from
These examples?

A generic solution
does not exist.

a generic

Solution

cannot

exist.

all solutions
have either
explicit or
implicit
context.

too often
implicit

& unknown

IMPLICIT LIMITS

ϕ ? ψ ?

θ ?

χ ?

Simple C++ HDS class definition

```
struct HalfEdge
{
    HalfEdgeVert *endPt;
    HalfEdge *next;
    HalfEdge *opposite;
    HalfEdgeFace *face;
    void *userData;
    unsigned char marker;
};
```

```
struct HalfEdgeFace
{
    HalfEdge *halfEdge;
    unsigned char marker;
};

struct HalfEdgeVert
{
    HalfEdge *halfEdge;
    int index;
    unsigned char marker;
};
```


ARCHITECTURE

CELL

MEM

GPUS

SERVER FARM

TRY TO SOLVE
ALL PROBLEMS,
SAVE VERY
FEW.

WHAT CASE TO
SOLVE FOR?

WORST?

BEST?

MOST COMMON?

EPIGE?

TOTALLY
UNKNOWN
CASES?

MATH LIB INFLUENCE
ON CODE.

BAD

"

CONTEXT

FREE

"

Squirrel -

• best Things about
parametric...
without memory
or context"

SOMEONE HAS
TO SAY

"BUT THAT'S
ONLY USEFUL
FOR GAMES!"

= "THEY" OR

"USER"

MENTAL

MODEL =

PARADIGM

ass Cullen
P,
amiphish - 92 lines - codepad

codepad [create a new paste]

Link: <http://codepad.org/u3yqprP> [raw code | output | fork]

kamiphish - C++, pasted 1 hour ago:

```
1 #define MAX_PLANES 6
2
3 enum { NONE, IN, OUT };
4
5 struct Sphere
6 {
7     float x, y, z;
8     float r;
```

TYPICAL.


```

19
20 class Object
21 {
22 public:
23     virtual void onVisible(Culler&) = 0;
24     const Sphere& WorldBound() const;
25 };
26
27 int WhichSide(const Plane& p, const Sphere& s)
28 {
29     float d = p.x * s.x + p.y * s.y + p.z * s.z + p.c;
30
31     if (d <= - s.r)
32         return OUT;
33     else if (d >= s.r)
34         return IN;
35     else
36         return NONE;
37 }
38
39 class Culler
40 {
41     Plane m_planes[MAX_PLANES];
42     unsigned int m_planeState;

```



```
class
{
public:
    virtual void onVisible(Culler&) = 0;
    const Sphere& WorldBound() const;
};
```

```
int WhichSide(const Plane& p, const Sphere& s)
{
    float d = p.x * s.x + p.y * s.y + p.z * s.z + p.c;
```

```
    if (d <= - s.r)
        return OUT;
    else if (d >= s.r)
        return IN;
    else
        return NONE;
```

```
}

class Culler
```

```
{
    Plane m_planes[MAX_1]
    unsigned int m_plane
```

```
public:
```

```
    void CullInvisibleOb
```

← WHY ARE
WE CONVERTING
FLOAT TO INT
HERE?


```
22   virtual  
23   const Sphere&  
24   };  
25  
26   int WhichSide(const Plane& p, const Sphere& s)  
27   {  
28       + p.z * s.z + p.  
29
```

↑
OH YEAAH
BIC WE THINK
LOGICAL RESULT
"CAN'T" BE
FLOAT.


```
class Object
```

```
{  
public:  
    virtual void onVisible(Culler&) = 0;  
    const Sphere& WorldBound() const;
```

```
};  
  
int WhichSide(const Plane& p, const Sphere& s)  
{  
    float d = p.x * s.x + p.y * s.y + p.z * s.z + p.c;
```

```
    if (d <= - s.r)  
        return OUT;  
    else if (d >= s.r)  
        return IN;  
    else  
        return NONE;
```

```
}
```

```
class Culler
```

```
{  
    Plane m_planes[MAX_P]  
    unsigned int m_plane;
```

← NOT TO
MENTION
CRAZY
BRANCHES.


```

19 class Object
20 {
21 public:
22     virtual void onVisible(Culler&) = 0;
23     const Sphere& WorldBound() const;
24 };
25
26
27 int WhichSide(const Plane& p, const Sphere& s)
28 {
29     float d = p.x * s.x + p.y * s.y + p.z * s.z + p.c;
30
31     if (d < 0)
32         return -1;
33     else if (d > 0)
34         return 1;
35     else
36         return 0;
37 }
38
39 class Culler
40 {
41     Plane m;
42     unsigned
43
44 public:
45

```


 WHEN WAS THE
 LAST TIME YOU
 HAD ONE PLANE
 AND ONE SPHERE?


```
18 class
19
20 class Object
21 {
22 public:
23     virtual void onVisible(Culler&) = 0;
24     const Sphere& WorldBound() const;
25 };
26
27 int WhichSide(const Plane& p, const Sphere& s)
28 {
29     float c = p.x * s.x + p.y * s.y + p.z * s.z - p.c;
30
31     if (d < 0)
32         re
33     else if (d > 0)
34         re
35     else
36         re
37
38 }
39
40 class Cull
41 {
42     plane
43     unsign
```

↑

HWT:
NEVER.


```
class Culler
{
public:
    virtual void onVisible(Culler& c) const;
    const Sphere& WorldBound() const;
};
```

```
int WhichSide(const Plane& p, const Sphere& s)
```

```
{
    float d = p.x * s.x + p.y * s.y + p.z * s.z - p.c;

    if (d < 0)
        return -1;
    else if (d > 0)
        return 1;
    else
        return 0;
}
```

```
class Cull
{
    plane
    unsign
public:
```

↑
WHERE THERE'S
ONE, THERE'S
MORE THAN
ONE.

23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51

```
const Sphere& s)  
) ;  
  
int WhichSide(const Plane& p, const Sphere& s)  
{  
    float c = p.x * s.x + p.y * s.y + p.z * s.z - p.d;  
    if (d < -0.01) return -1;  
    else if (d > 0.01) return 1;  
    else return 0;  
}
```

```
class Cull  
{  
    Plane  
    unsigned
```

```
public:
```

```
void C
```

```
{  
    if
```

```
{
```

```
}
```

↑
DON'T BUILD
FUNC FOR
 $< -0.01\%$ CASE
HERE AS
"GENERAL"
SOLUTION.

object is not visible

const Plane& p, const Sp
float c = p.x * x + p.y * y + p.z * z - p.d;



THE "GENERAL"
CASE IS
MANA

if (d > 0) re
else if (d < 0) re
else re

class Cull
{
plane
unsign

public:

void C


```
}
```

```
class Culler
```

```
{
```

```
    Plane m_planes[MAX_PLANES];
```

```
    unsigned int m_planeState;
```

```
public:
```

```
void CullInvisibleObjects(Object* object)
```

```
{
```

```
    if (!m_planeState)
```

```
    {
```

```
        object->onVisible(*this);
```

```
    }
```

```
else
```

```
{
```

```
    // Determine if the object is not visible
```

```
    // bound to each culling plane.
```



```
else  
    return NONE;
```

```
class Culler
```

```
    Plane m_planes[MAX_PLANES];  
    unsigned int m_planeState;
```

```
public:
```

```
    void CullInvisibleObjects(Object* object)
```

```
{
```

```
    if (!m_planeState)
```

```
{
```

```
        object->setVisible(*this);
```

```
    }
```

```
    else
```

```
{
```

```
        //
```

```
        //
```

↑
SAME BS.


```
ane m_planes[...]  
signed int m_planeState;
```

```
c:  
oid CullInvisibleObjects(Object* object)
```

```
if (!m_planeState)
```

```
{  
    object
```

```
}
```

```
else
```

```
{
```

```
// Det
```

```
// bou
```

comparing

IS THERE
REALLY JUST
ONE UNIQUE
OBJECT IN
THE APP?


```
es[MAX_PLANE]  
t m_planeState;
```

```
InvisibleObjects(object* object)
```

```
m_planeState  
object->onVisible(*this);
```

```
e
```

```
// De  
// bot
```

OK.
NO.

object is not visible
plane.


```
if (d <= - s.r)
    return OUT;
else if (d >= s.r)
    return IN;
else
    return NONE;
```

class Culler

```
Plane m_planes[MAX_PLANES];
unsigned int m_planeState;
```

public:

```
void CullInvisibleObjects(Object* object)
```

```
{
    if (!m_planeState)
    {
        object->onVisible(*this);
    }
    else
    {
        // Determine if the object
        // bound to each culling pl.
```

← READY?


```
t Plane  
x * s.x + p.y  
s.r)  
OUT;  
(d >= s.r)  
turn IN;  
return NONE;
```

```
s Culler  
Plane m_planes[MAX_PLANES];  
unsigned int m_planeState;
```

```
public:  
void CullInvisibleObjects(Object* object)  
{  
    if (!m_planeState)  
        object->onVisi  
    else  
        // Determine if  
        // bound to ea
```

← Take you
won't need
this test on
the exam, 1A


```
MAX_  
NONE, IN, OUT )
```

```
Sphere  
float x, y, z;  
float r;
```

```
1.x + c = 0  
uct Plane  
float x, y, z;  
float c;
```

```
class Culler;  
class Object
```

```
{  
public:  
    virtual void onVisible(Culler&) = 0;  
    const Sphere& WorldBound() const;  
};
```

```
int WhichSide(const Plane& p, const Spher  
( float d = p.x * s.x + p.y * s.y + p.z  
( = - s.r)  
OUT;  
(r)
```

← WAIT, LET
ME BACK UP
A MINUTE.


```
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94  
95  
96  
97  
98  
99  
100
```

```
};  
class Culler;  
class Object  
{  
public:  
    virtual void onVisible(Culler&) = 0;  
    virtual void onHidden(Culler&) = 0;  
};
```



```
    inst Spher  
    s.y + p.y
```



```
...
class Object
{
public:
    virtual void onVisible(Culler&) = 0;
    virtual void onHidden() const;
};
```



FIRST, YOU CAN
SORT BY TYPE

SO YOU CAN

MAKE

CORRECT

CALL

Spher
+ p.z

public.

15
16
17
18
19
20
21
22
23
24
25
26

```
};  
class Culler;  
class Object  
{  
public:  
    virtual void onVisible(Culler&) = 0;  
    virtual void onWorldBound() const;
```



CORRECT CALL
WOULD
TAKE 71
CULLER.

Spher
+ p.z

← WAIT
ME BACK
A MIN

ct*


```
9
10 // n.x * 0
11 struct Plane
12 {
13     float x, y, z;
14     float c;
15 };
```

```
16 class Culler;
17
18 class Object
```

```
20 {
21     public:
22     virtual void onVisible(Culler*) = 0;
23     virtual void onHidden(Culler*) = 0;
```

```
    Sphere* s;
    + p.z * s.z +
```

↑
REQUIRE
USE OF
VIRTUAL.


```
13 float c;  
14  
15 };  
16  
17 class Culler;  
18  
19 class Object  
20 {  
21 public:  
22     virtual void onVisible(Culler c) const;  
23
```

... Although
A "6000" use
is hard to
come by.

```
Sphere& s)  
+ p.z * s.z +
```


depad

```
unsigned int saveActivePlanes = m_planeState;
```

```
unsigned int i;  
for (i = 0; i < MAX_PLANES; i++)
```

```
{  
    int m = 1 << i;  
    if (m_planeState & m)
```

```
{  
    int side = WhichSi
```

```
if (side == OUT)
```

```
{  
    // The object is  
    // side of the  
    break;
```

```
}  
if (side == IN)
```

```
{  
    // The object is  
    // so there is  
    // plane.  
    m_planeState &=
```

← ARGH!

AGAIN?

SORT THE
DATA!

SORT IT!


```
igned int saveActivePlanes = m_planeState;
```

```
igned int i;  
(i = 0; i < MAX_PLANES;
```

```
int m = 1 << i;  
if (m_planeState & m)  
{  
    int side = WhichSi
```

```
if (side == OUT)
```

```
{  
    // The object i  
    // side of the  
    break;  
}
```

```
if (side == IN)
```

```
{  
    // The object i  
    // so there is  
    // plane.  
    state &=
```

← KNOW
THE DATA!

How MANY
PLANE ARRANGE-
MENTS Do you
REALLY use?


```
int saveActivePlanes = m_planeState
```

```
int i;  
0; i < MAX_PLANES; i++)
```

```
= 1 << i;  
planeState & m)
```

```
int side = WhichSi
```

```
if (side == OUT)
```

```
{  
    // The object is  
    // side of the  
    break;  
}
```

```
if (side == IN)
```

```
{  
    // The object is  
    // so there is r  
    // plane.  
    planeState &=
```

← THEN CALL
CORRECT FUNC
TO XFORM
THAT DATA
(IN GROUP)
NO TEST. NO BRANCH.


```
unsigned int  
for (i = 0; i < MAX_PLANE  
{  
    int m = 1 << i;  
    if (m_planeState & m)  
    {  
        int side = WhichSide(m_planes[i], object->WorldBound
```

↑
NOTE YOU
CLEARLY HAVE
> 1 PLANE AT
THIS PT.

it is on the

positive side
child o


```
unsigned int i;  
for (i = 0; i < MAX_PLANES; i++)
```

```
{  
    int m = 1 << i;  
    if (m_planeState & m)  
    {
```

```
        int side = WhichSide(m_planes[i], object->WorldBou
```

it is on

↑
BUT STILL
CALLING
INDV. FUNK.
SO LAME!


```
unsigned int i;  
for (i = 0; i < MAX_PLANES; i++)
```

```
{  
    int m = 1 << i;  
    if (m_planeState & m)
```

```
{  
    int side = WhichSide(m_planes[i], object-  
since
```

↑
NOW WE'RE
STUCK w/
THIS INT


```
unsigned int  
for (i = 0; i < m; i++)  
{  
    int m = 1 << i;  
    if (m_planeState & m)  
    {  
        int side = WhichSide(m_planes[i], object->World)
```

since it is

↑
BASICALLY
FORCING YOU
TO BRANCH

e position
compare c


```
int saveActivePlane
```

```
int i;  
0; i < MAX_PLANES; i++)
```

```
m = 1 << i;  
(m_planeState & m)
```

```
int side = WhichSide(m_planes[i], object->WorldBound())
```

```
if (side == OUT)
```

```
{  
    // The object  
    // side of the  
    break;  
}
```

```
if (side == IN)
```

```
{  
    // The object  
    // there is
```

← LIKE THIS.


```
}  
}  
if (i == MAX_PLANES)  
{  
    object->onVisible(*this);  
}
```

↑
DON'T JUMP
OUT HERE!


```
}  
}  
if (i == MAX_PLANES)  
{  
    object->onVisible(*this);  
}
```

↑
Killing
1 character for
no reason.


```
}  
if (i == MAX_PLANES,  
{ object->onVisible(*this,  
s_planeState, saveActivePlanes;  
}
```

↑
CALL
ONVISIBLE
CATER, ON
WHOLE LIST.

so function
defined re


```
object->onv...
```

```
m_planeState = saveActivePlanes;
```

↑
GHOST WRITES.
BAD FOR
CREATE. BAD
FOR THE SOUL.

22 public:

23 virtual void onVisible(Culler&) = 0;

24 const Sphere& WorldBound() const;



25
26
27
28 WHILE I'M

29
30 HERE.

31
32
33
34
35
36
37
38 REFERENCES?

39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
POINTLESS

CRUFT.



Mike Acton

#AltDevBlogADay #gamedev

@mike_acton

macton@gmail.com

<http://www.linkedin.com/in/mikeacton>