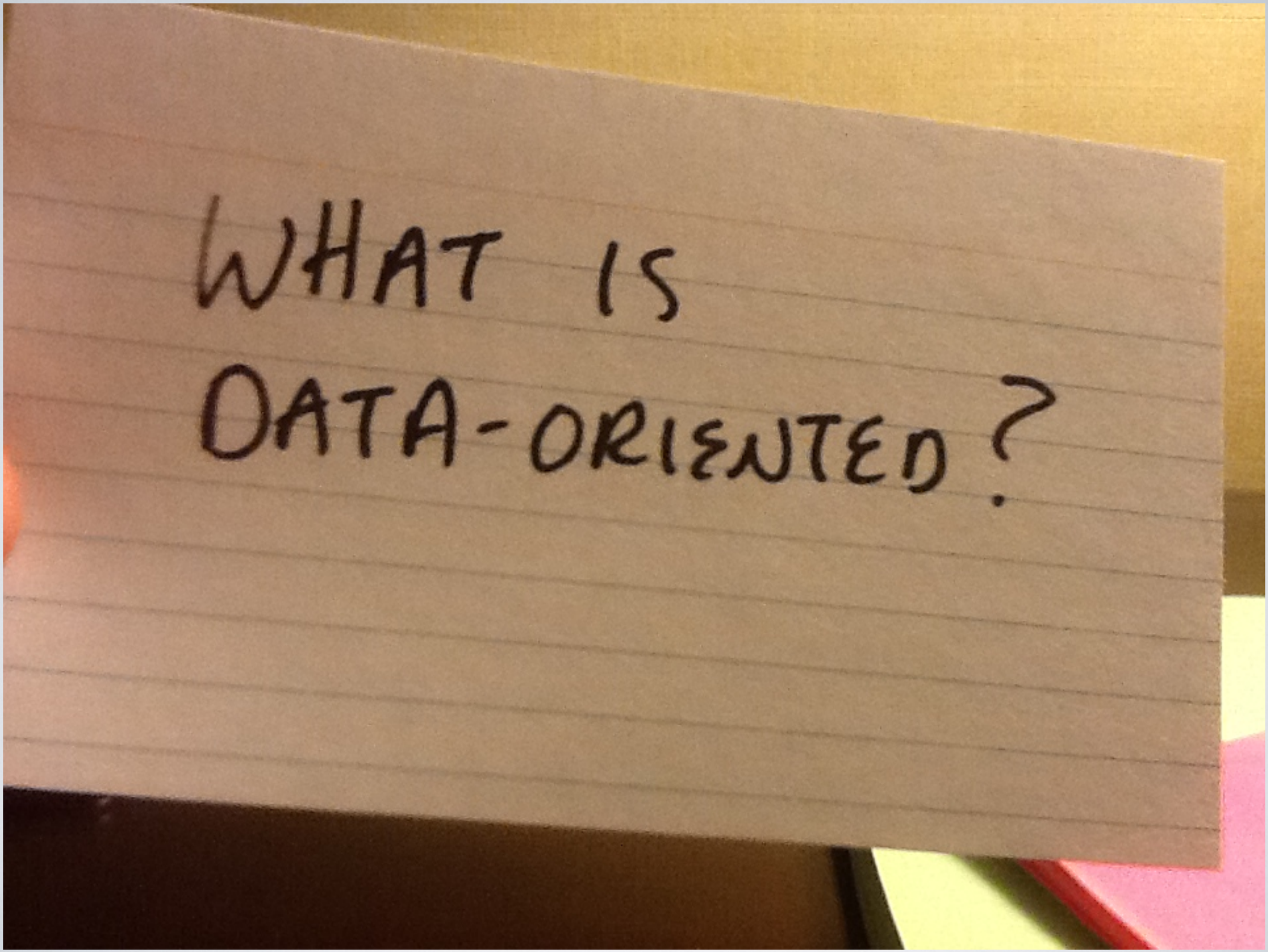
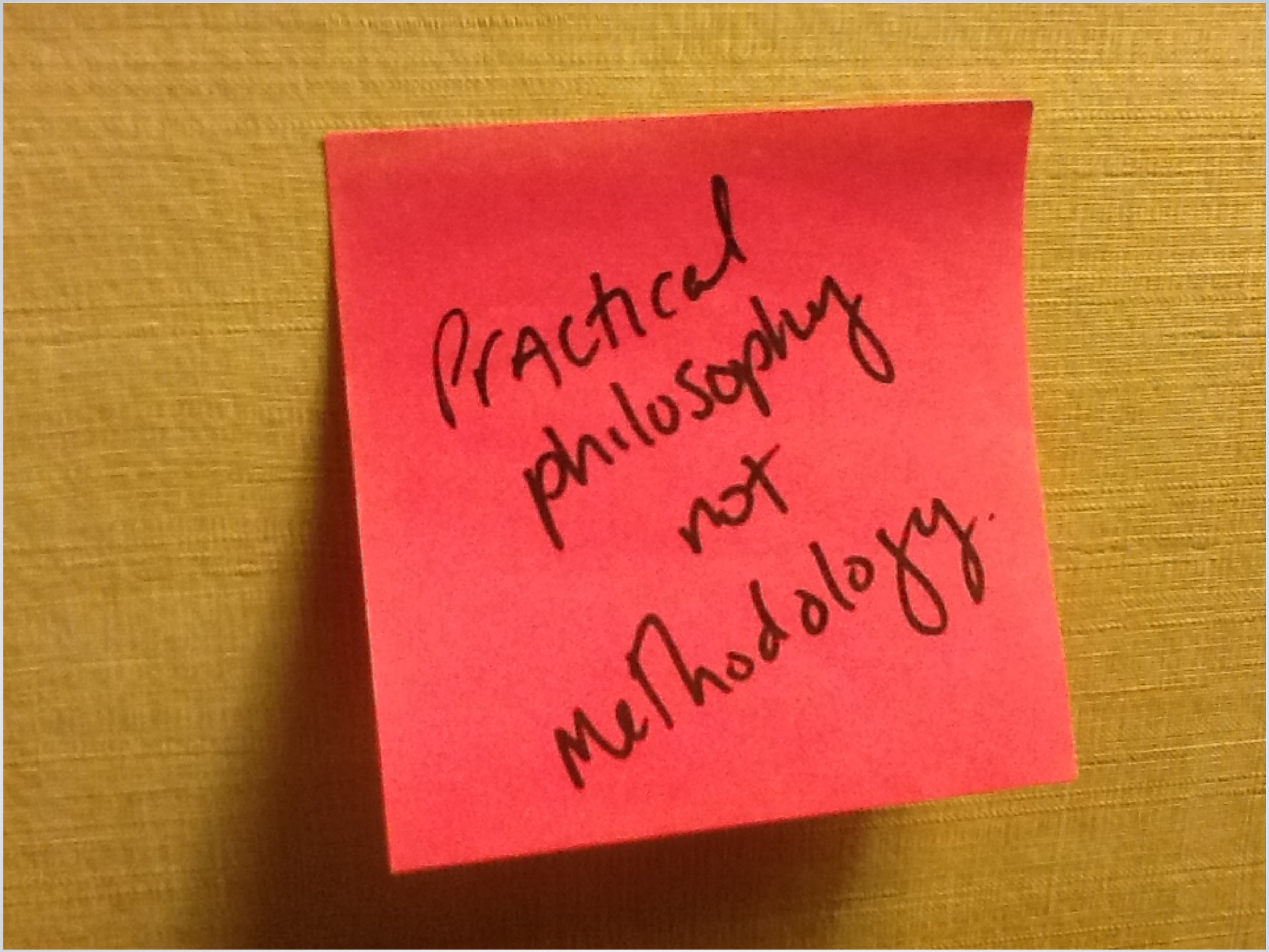
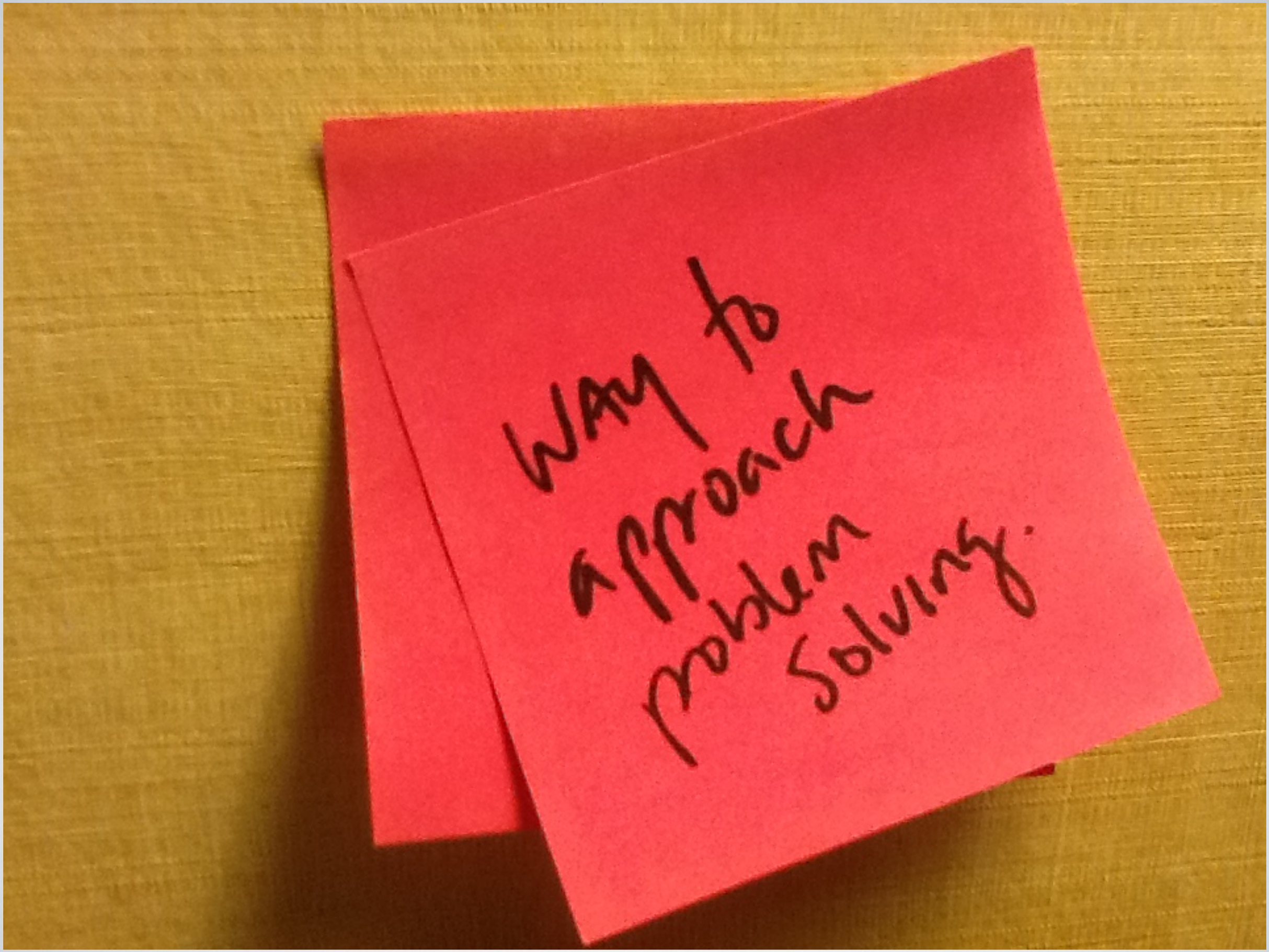
DATA - ORIENTED
DESIGN
(for math)

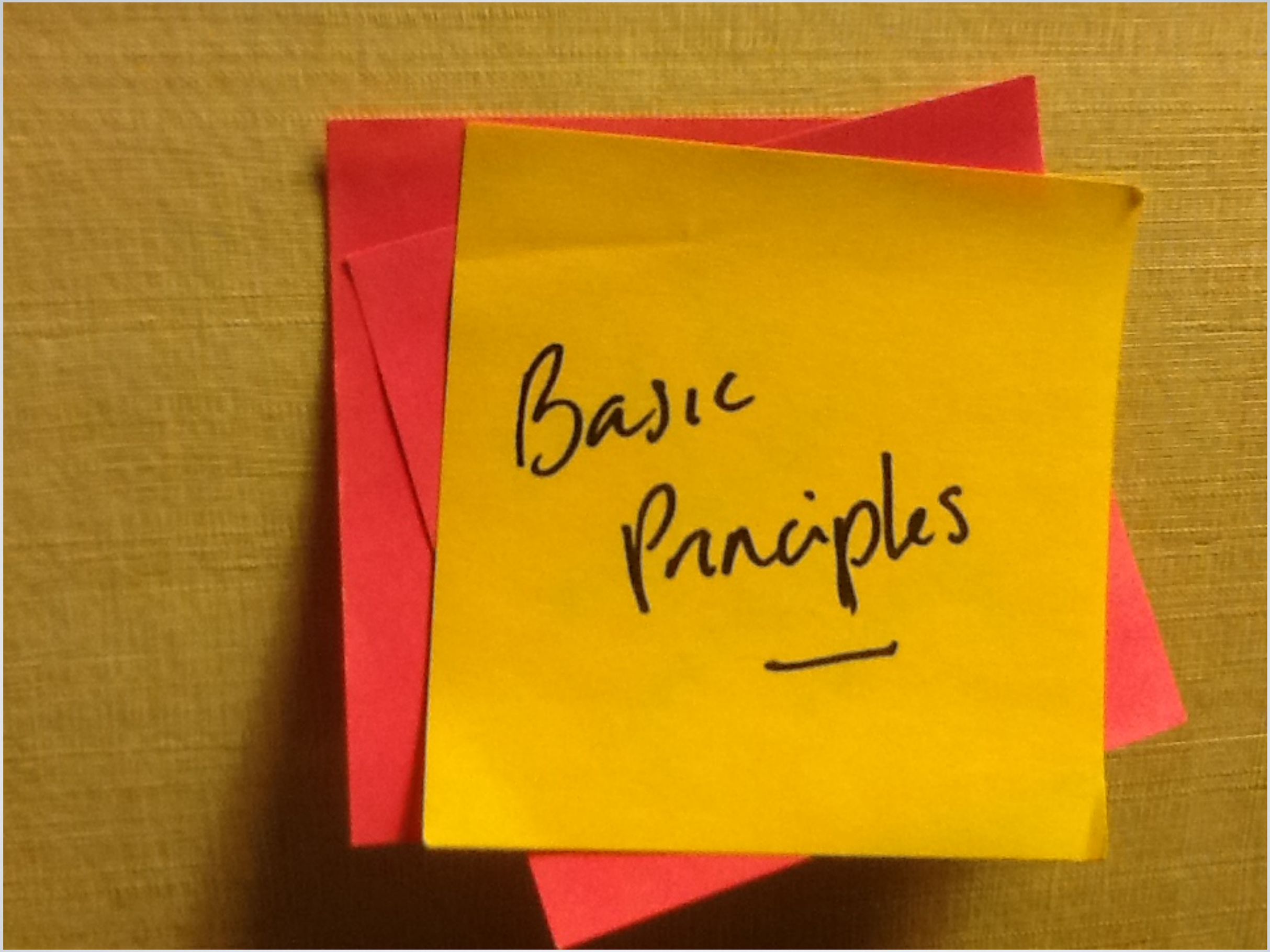

@mike-action

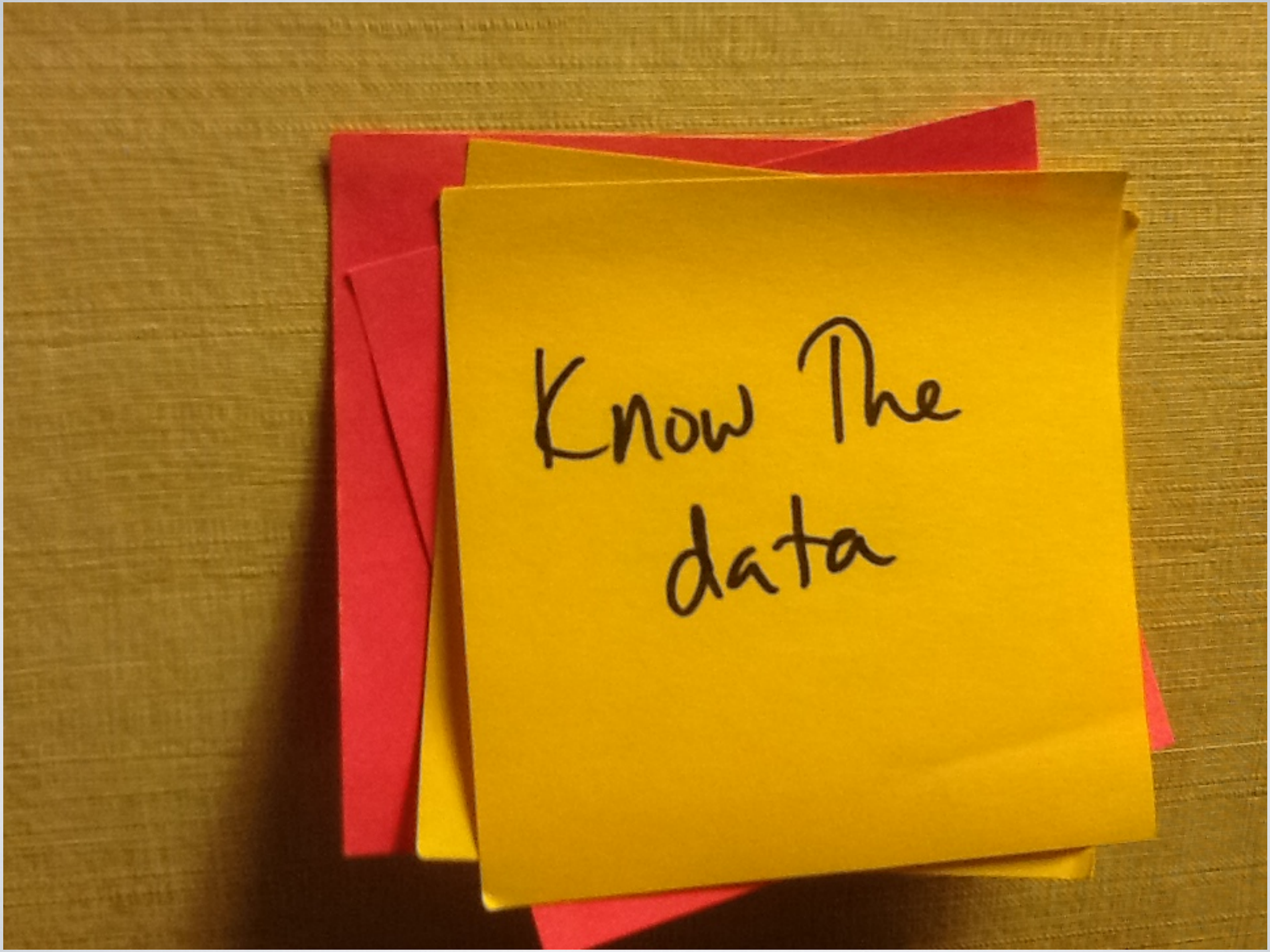
anti-math
fail!

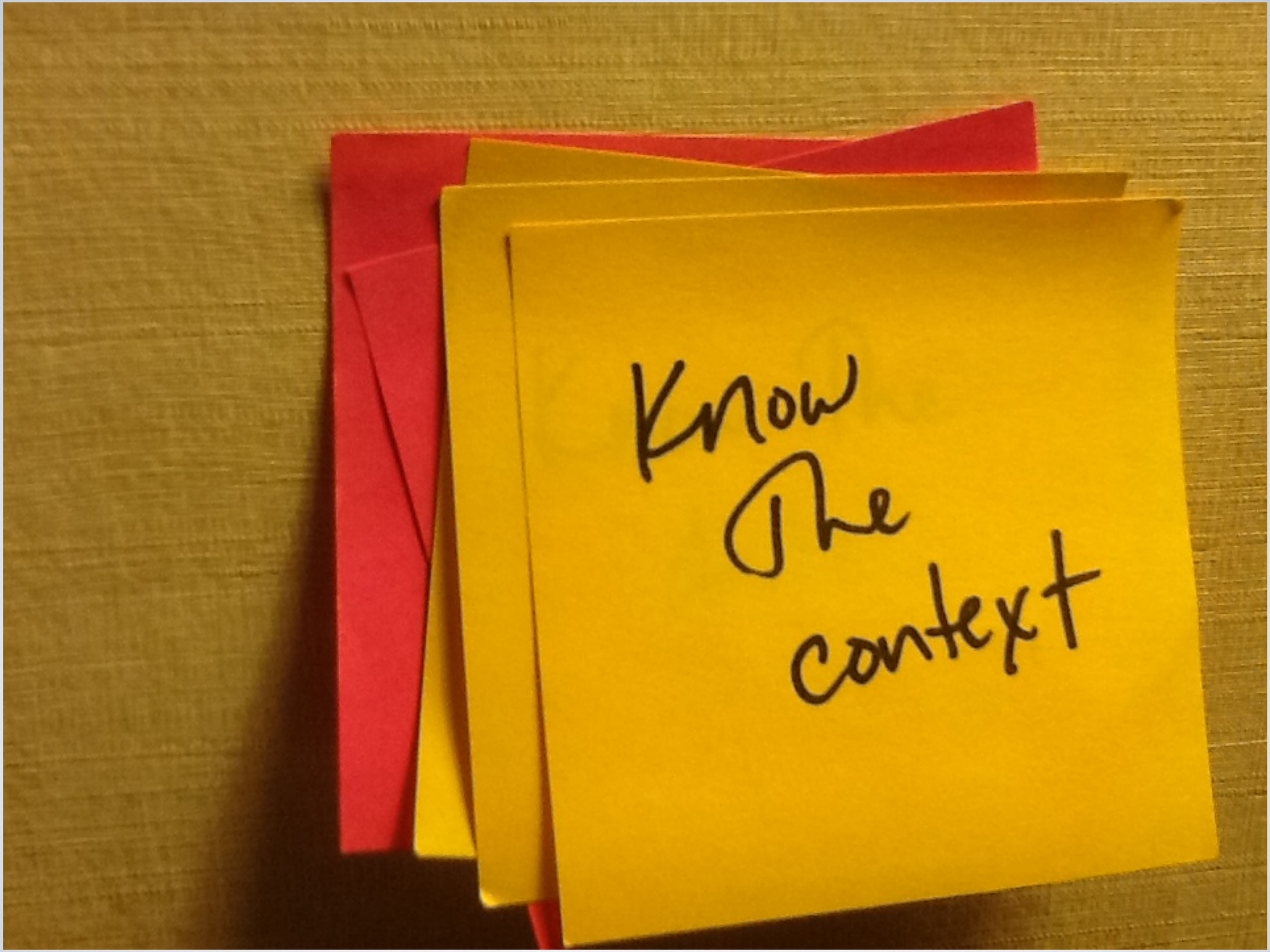

WHAT IS
DATA-ORIENTED?

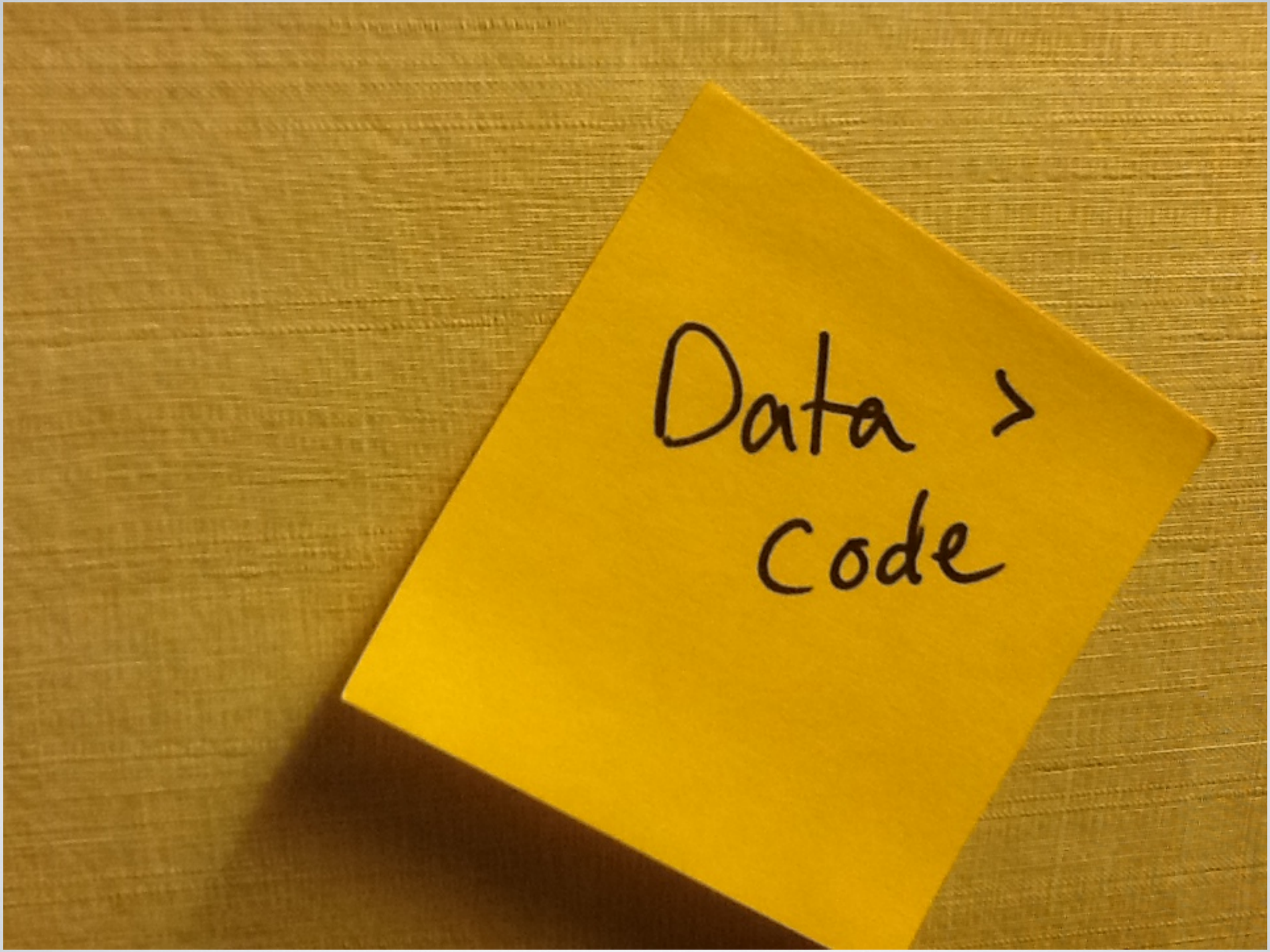

Practical
philosophy
not
methodology.

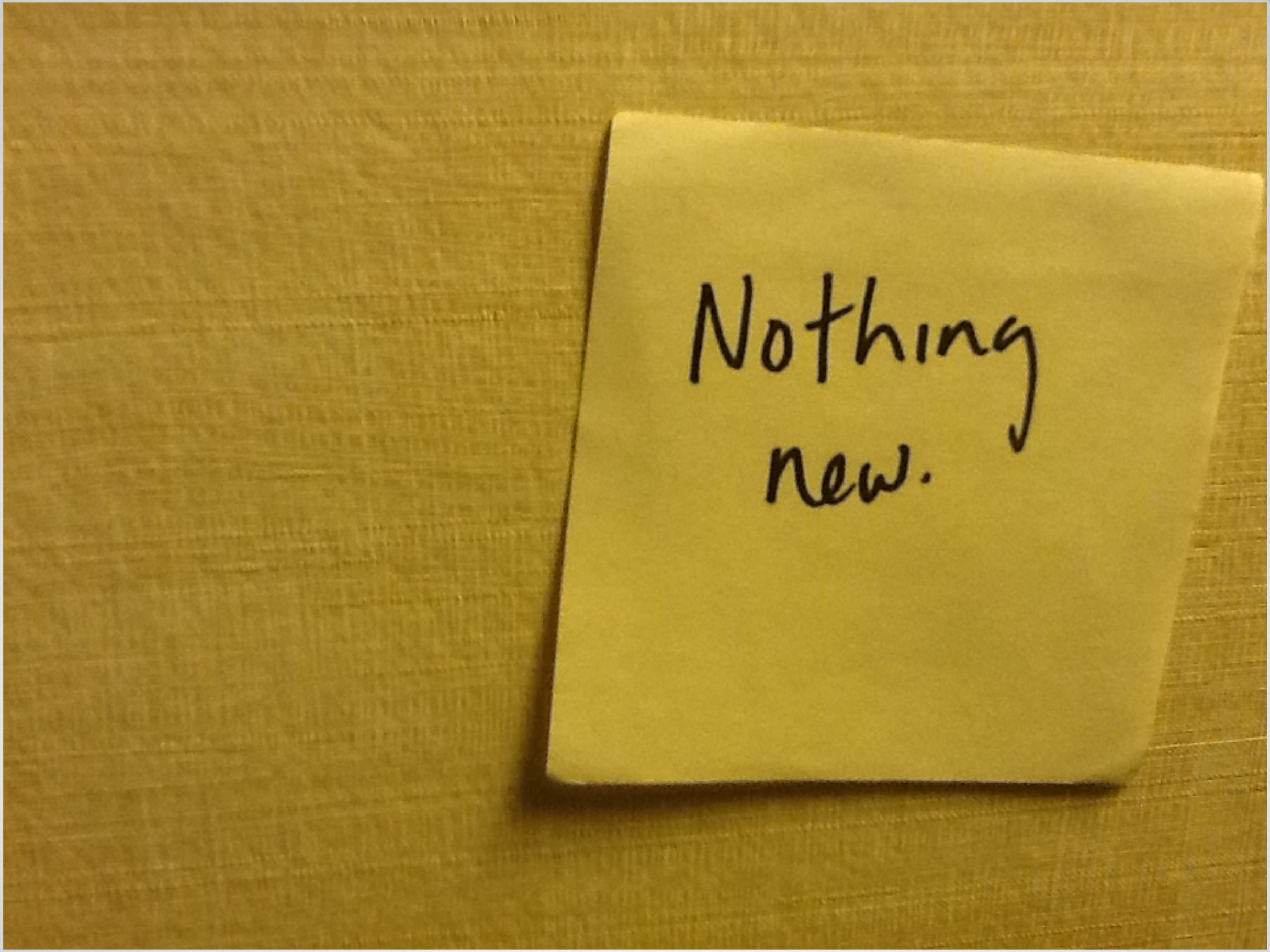

Way to
approach
Problem
Solving.


Basic
Principles


Know The
data


know
the
context

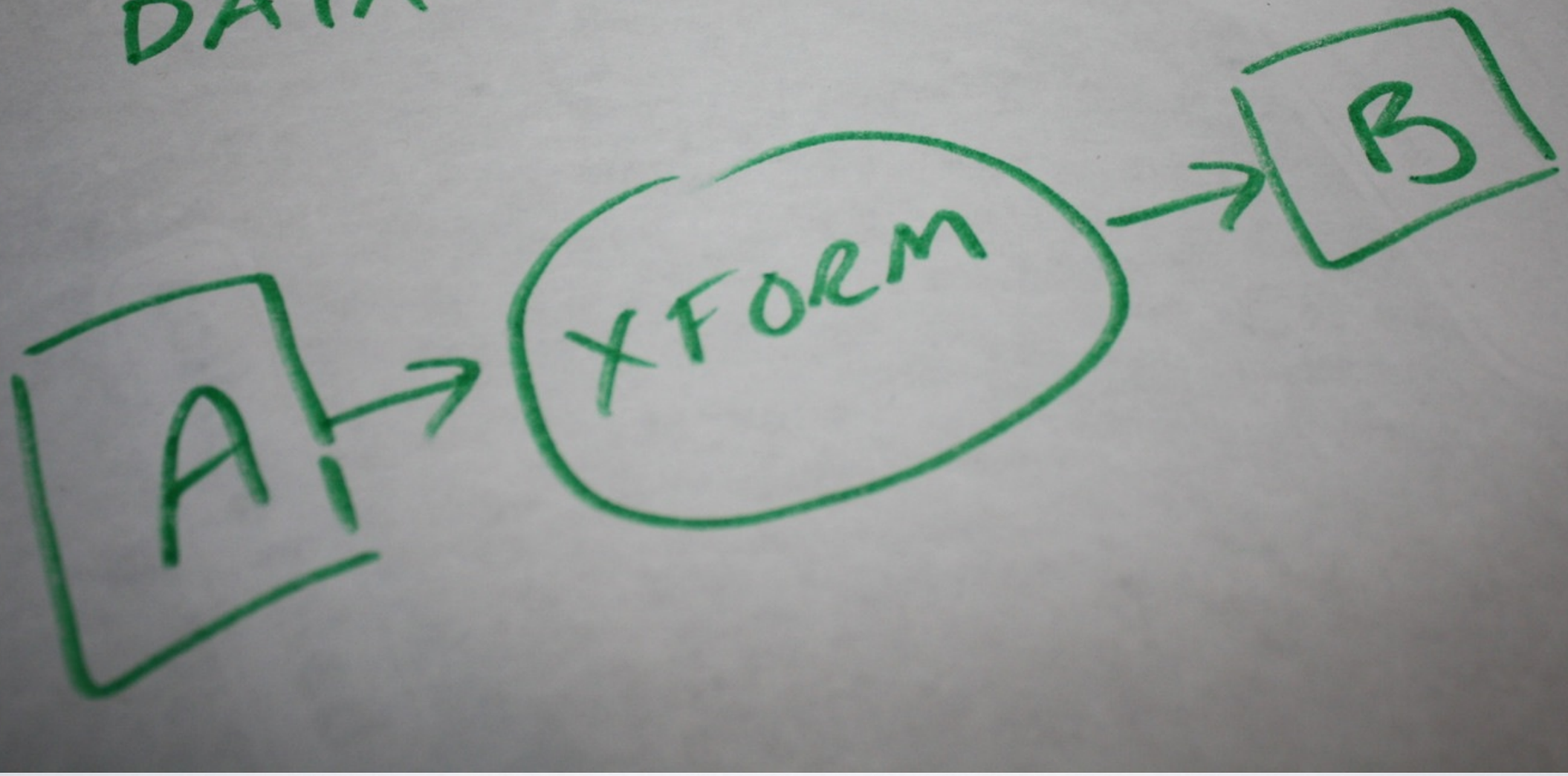

Data >
Code

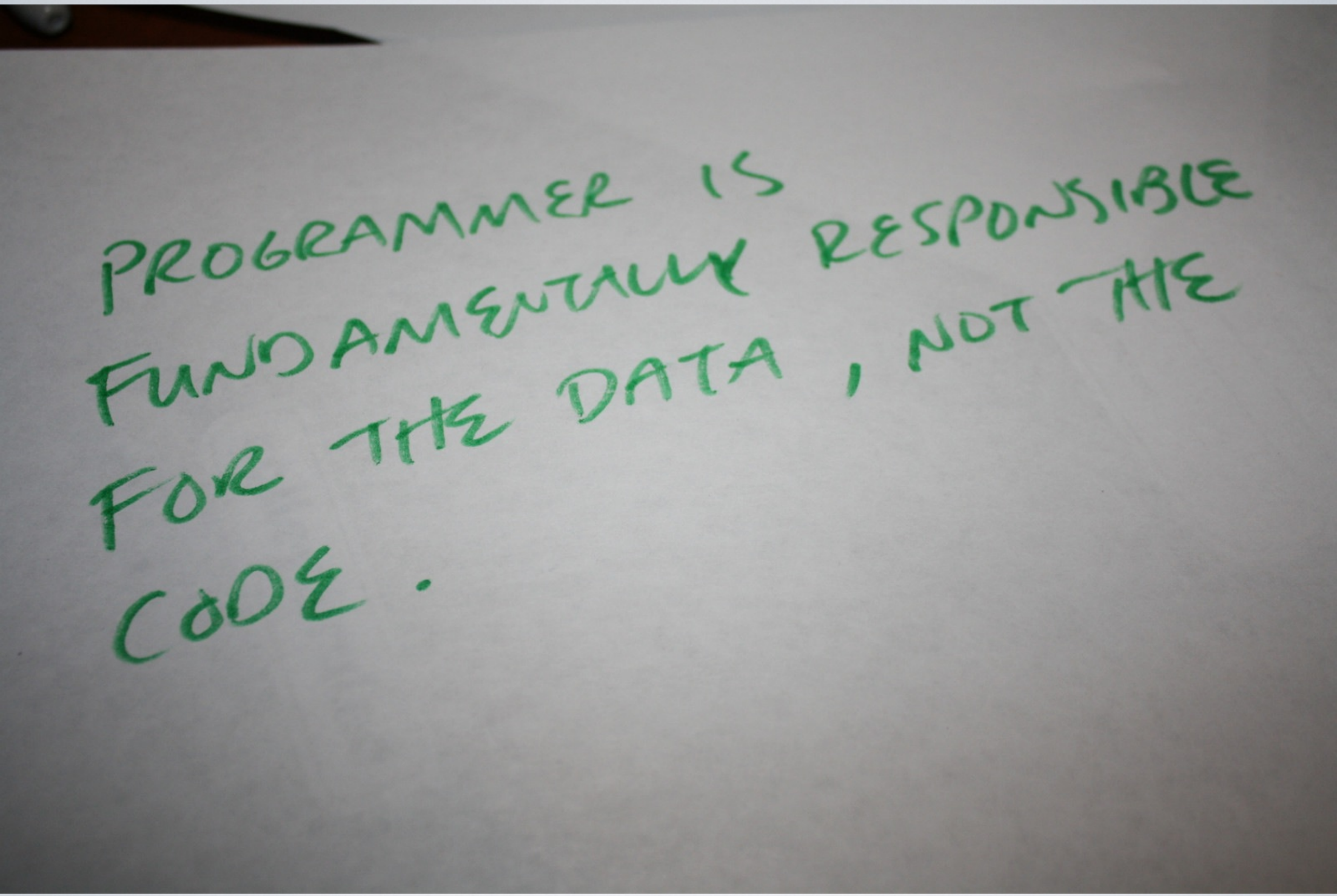

Nothing
new.

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.)

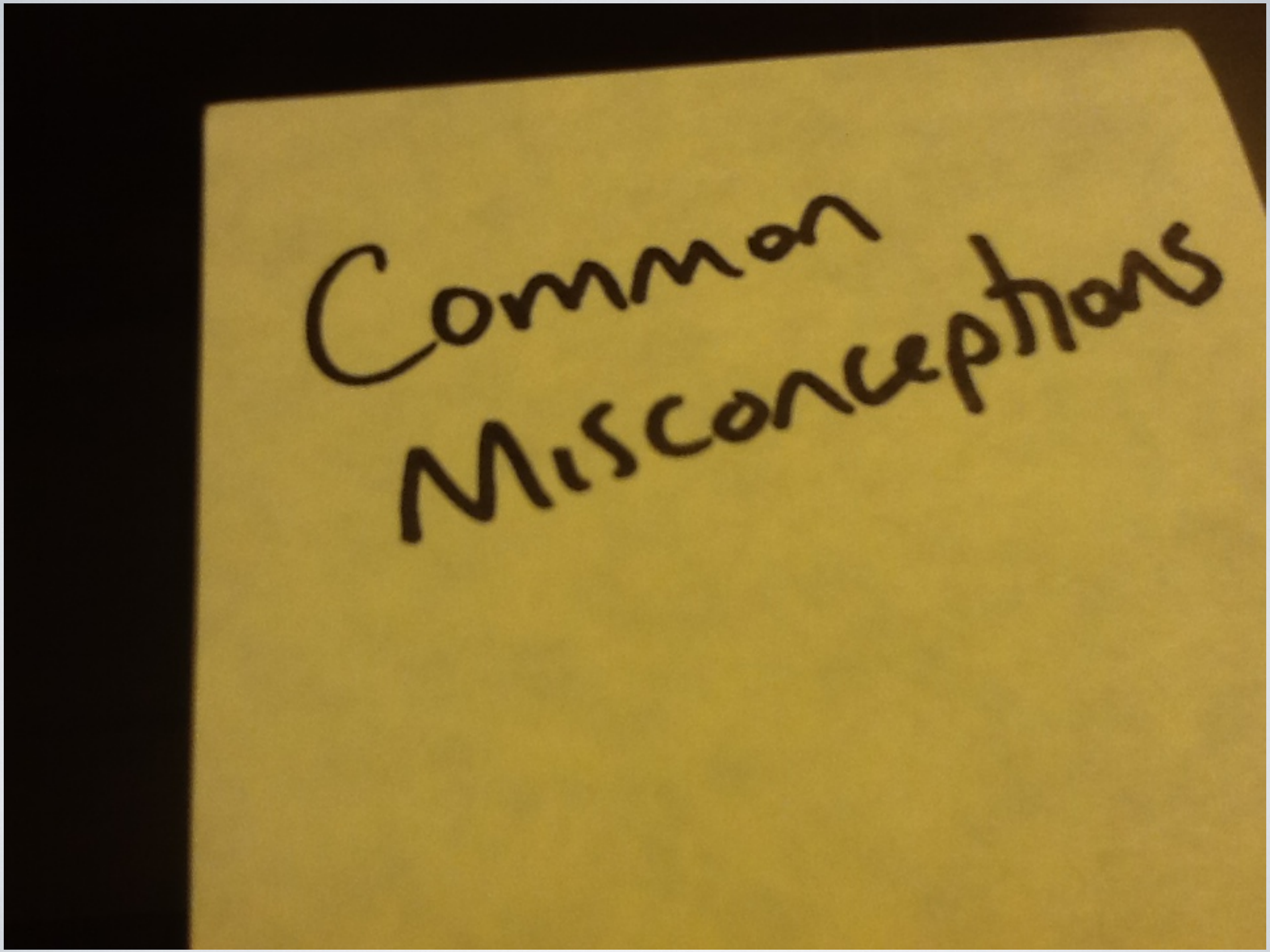
UNDERSTAND THE DATA
TO UNDERSTAND THE
PROBLEM.

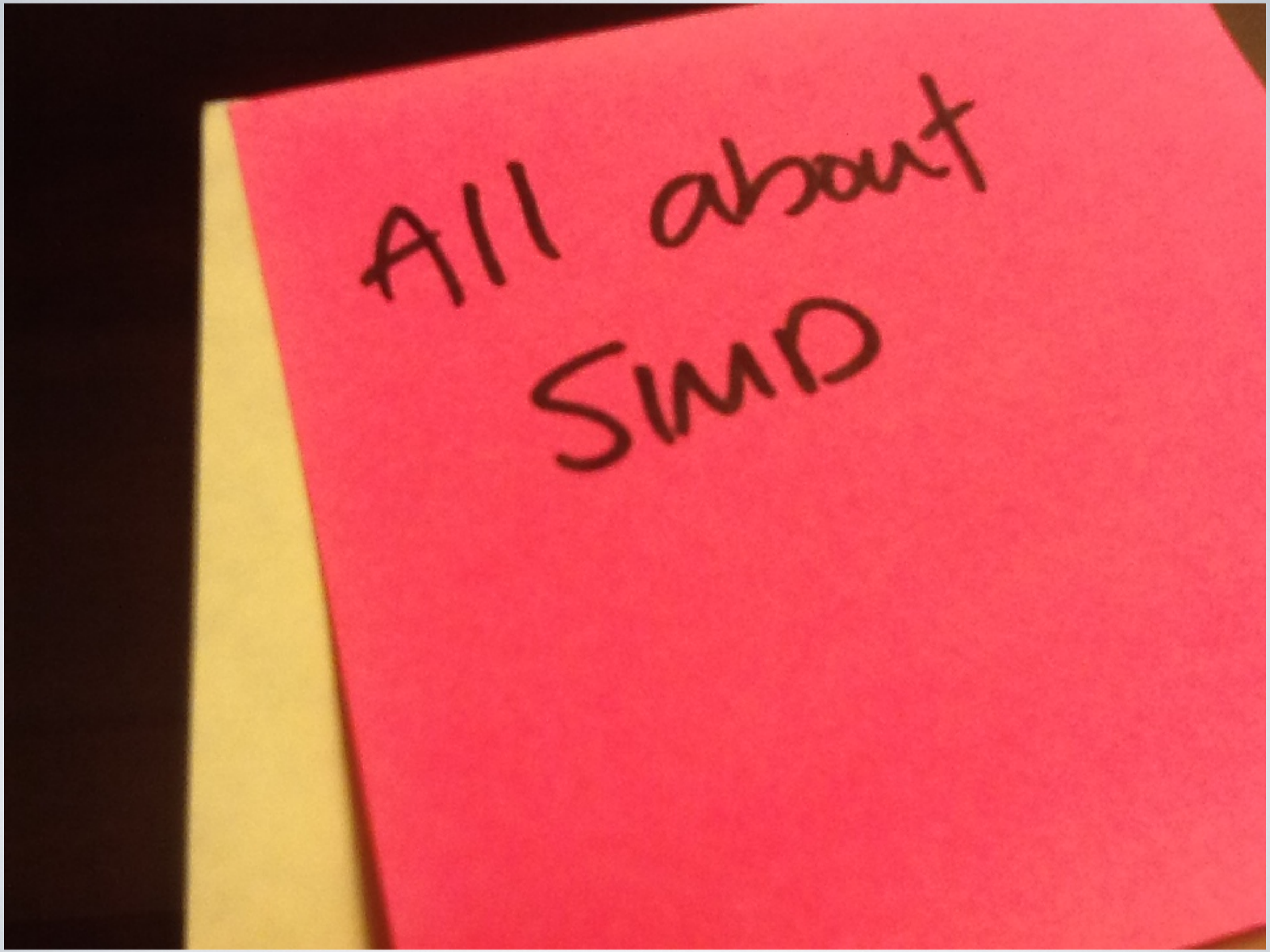
THERE IS NO
IDEAL, ABSTRACT
SOLUTION TO THE
PROBLEM.

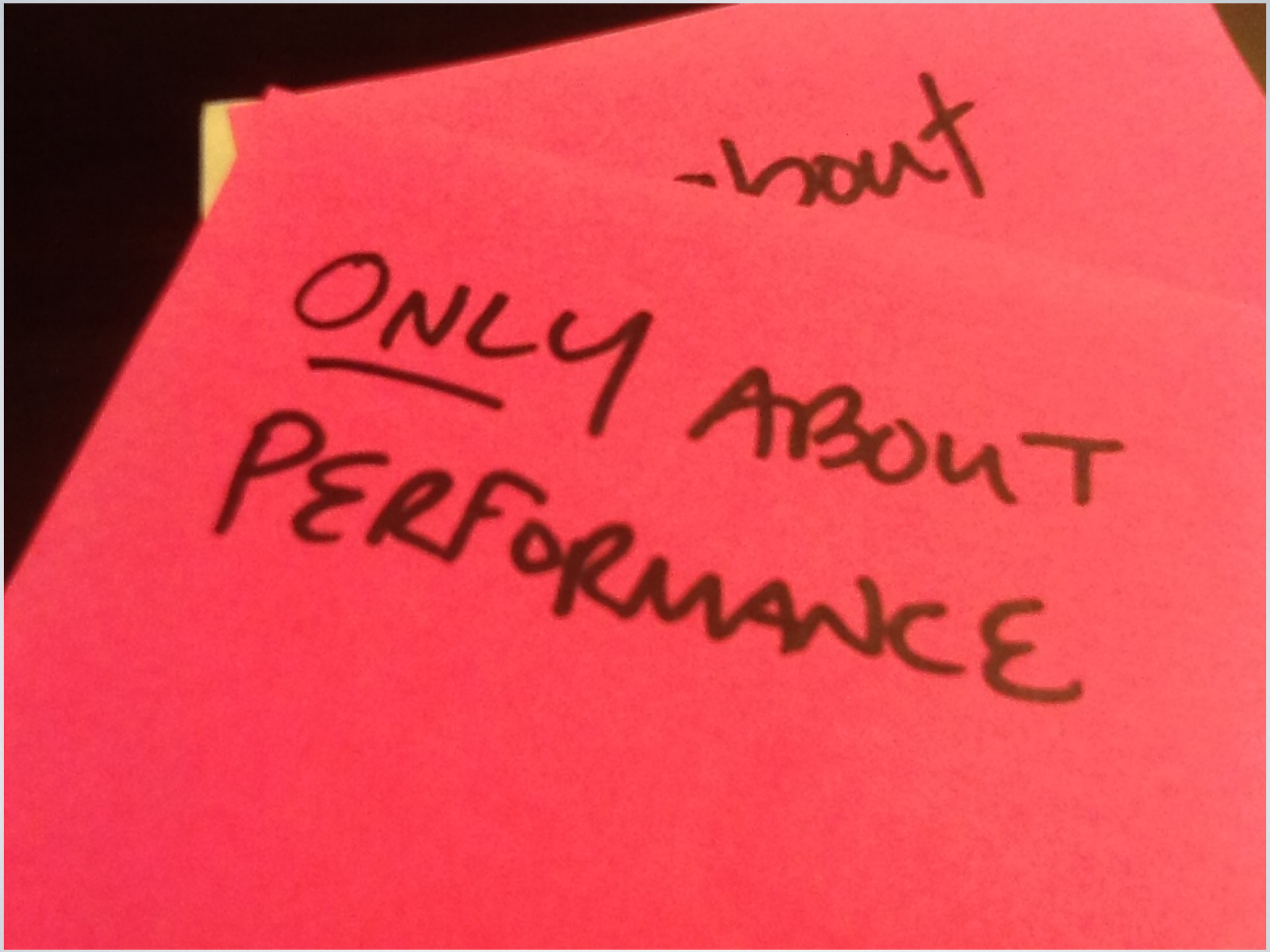
YOU CAN'T

PROOF"

"FUTURE


Common
Misconceptions

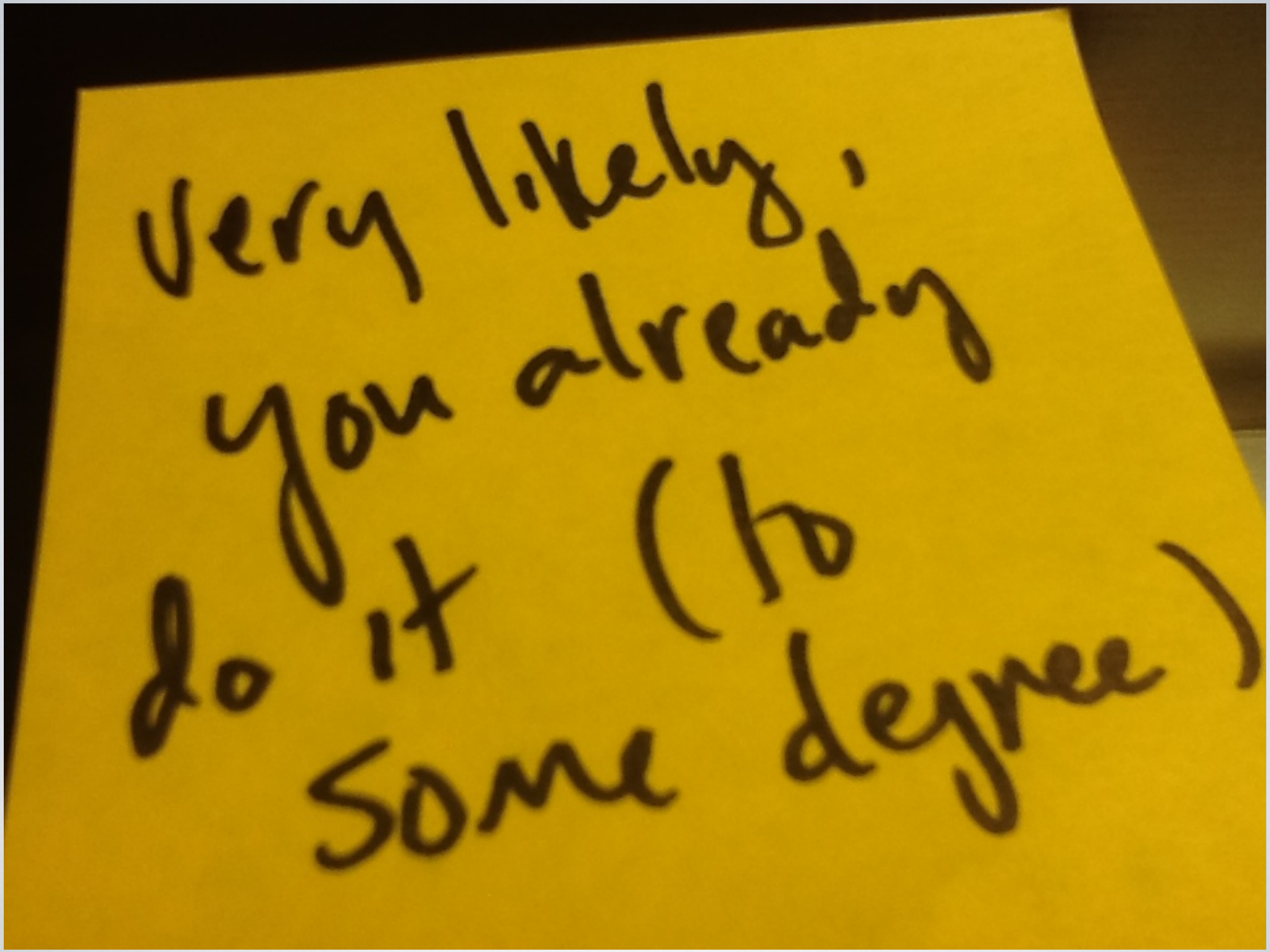

All about
SMD

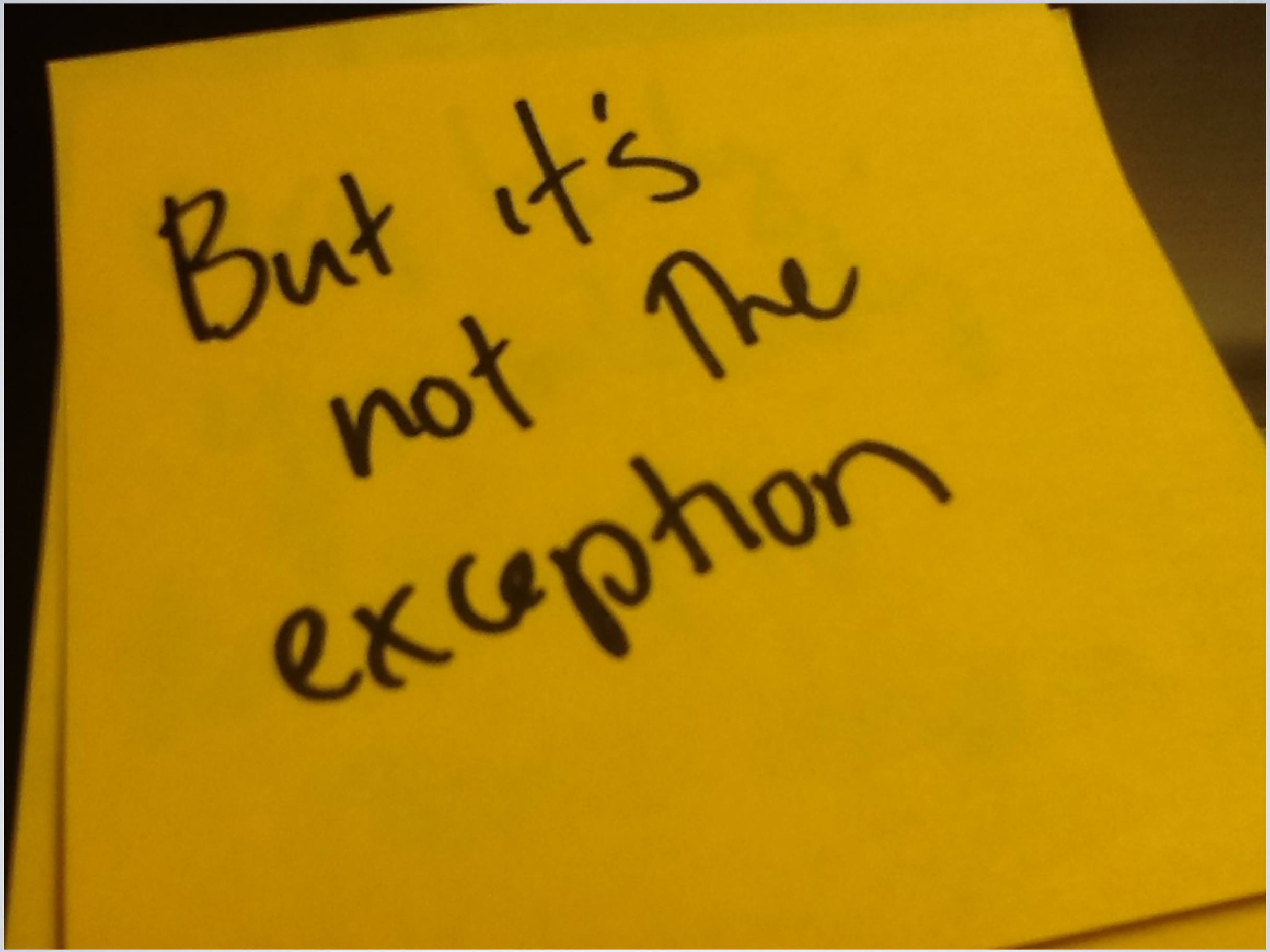

-bout
ONLY
ABOUT
PERFORMANCE

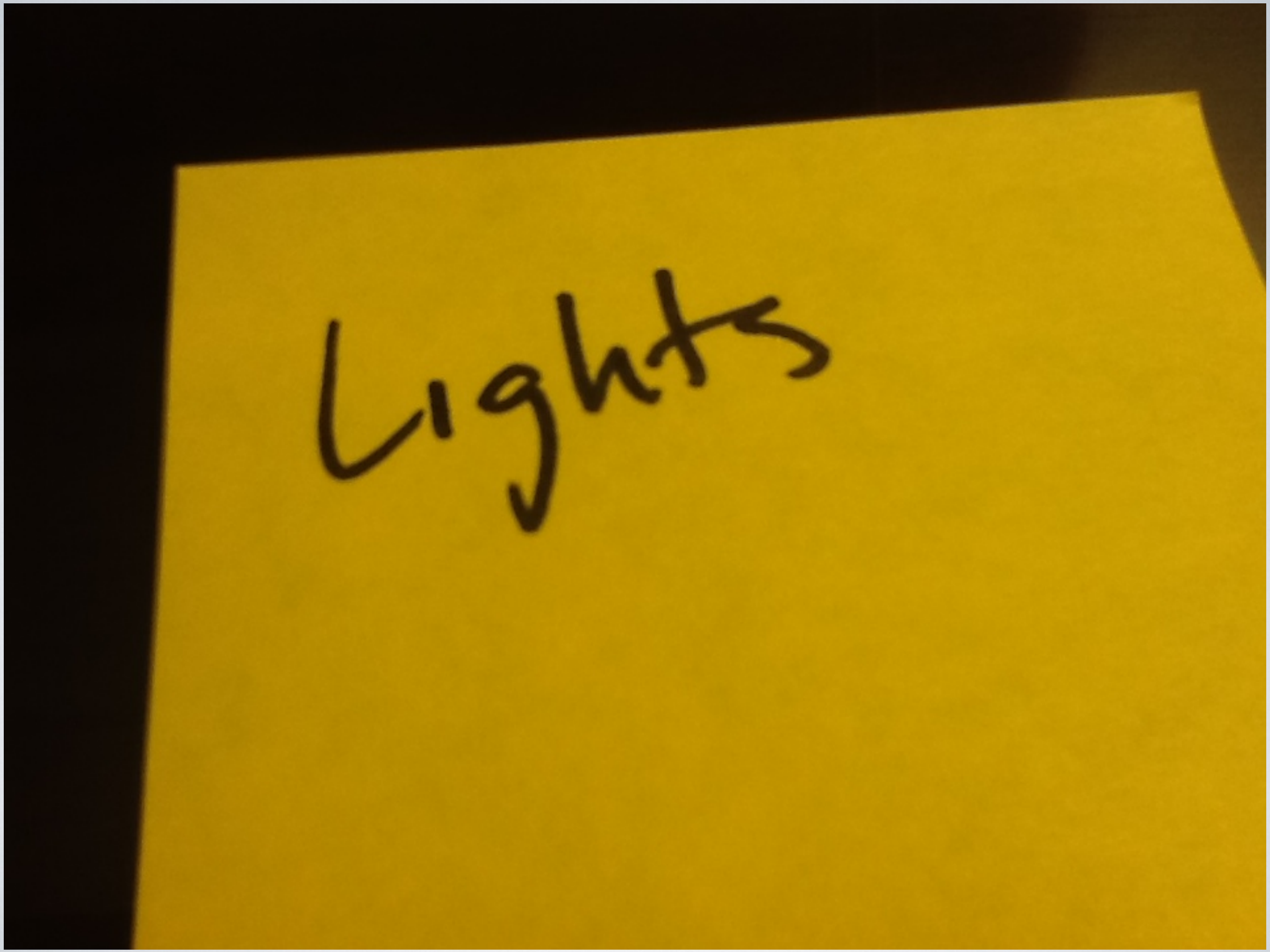
OPTIMIZABILITY
CAN BE VS.
IMPOSSIBLE.

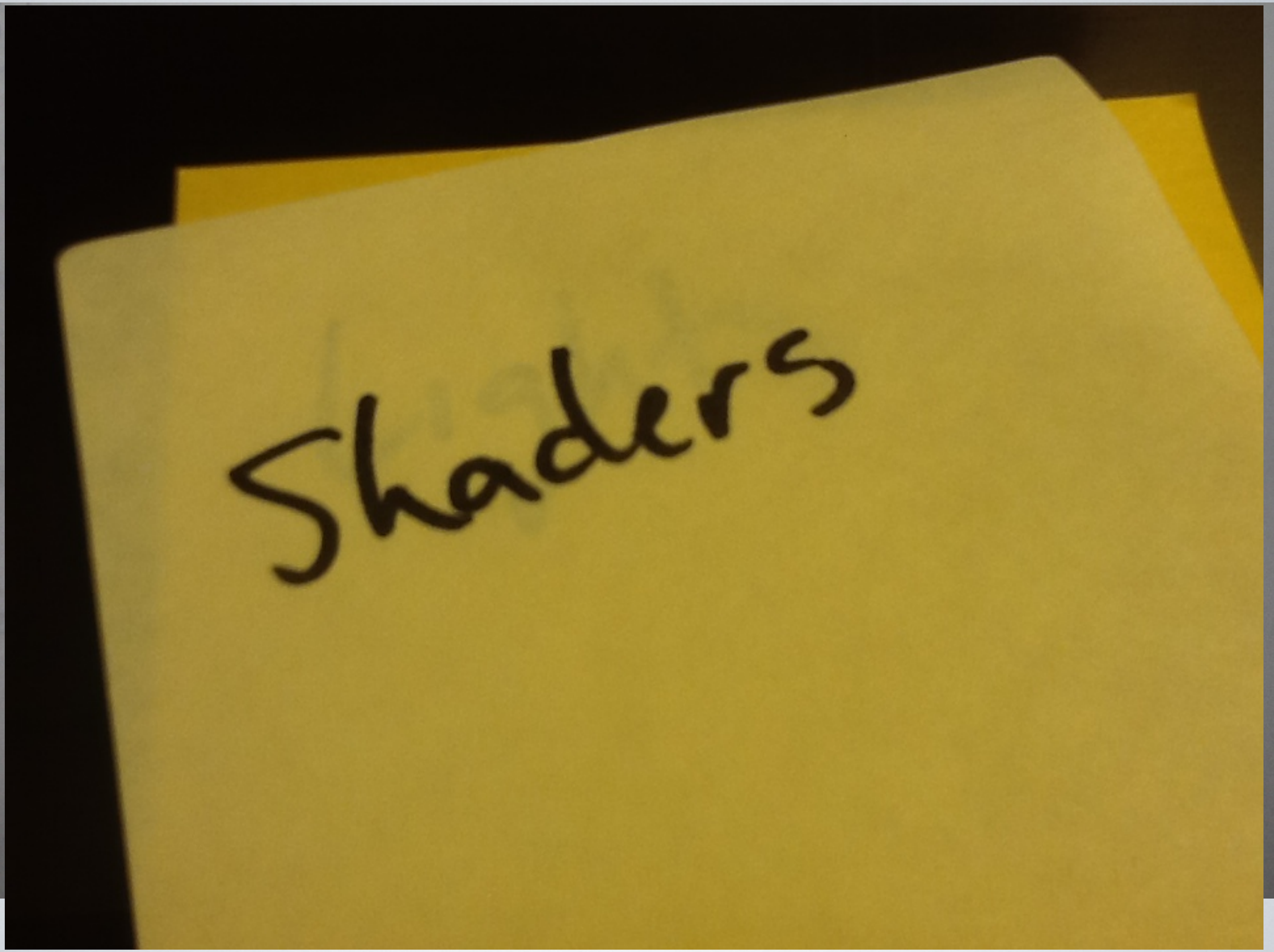
-Scout

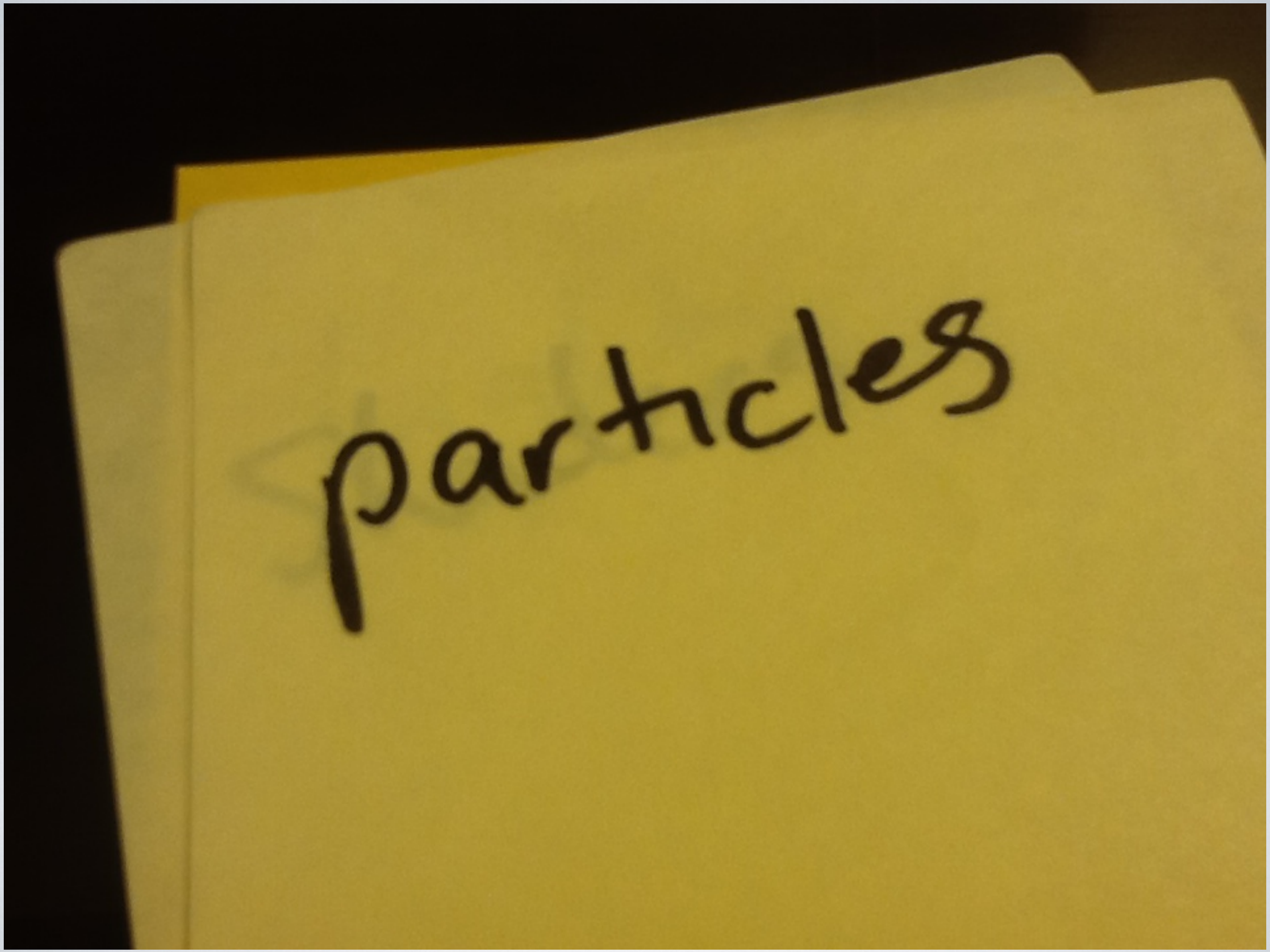
Harder to
understand,
maintain, etc.

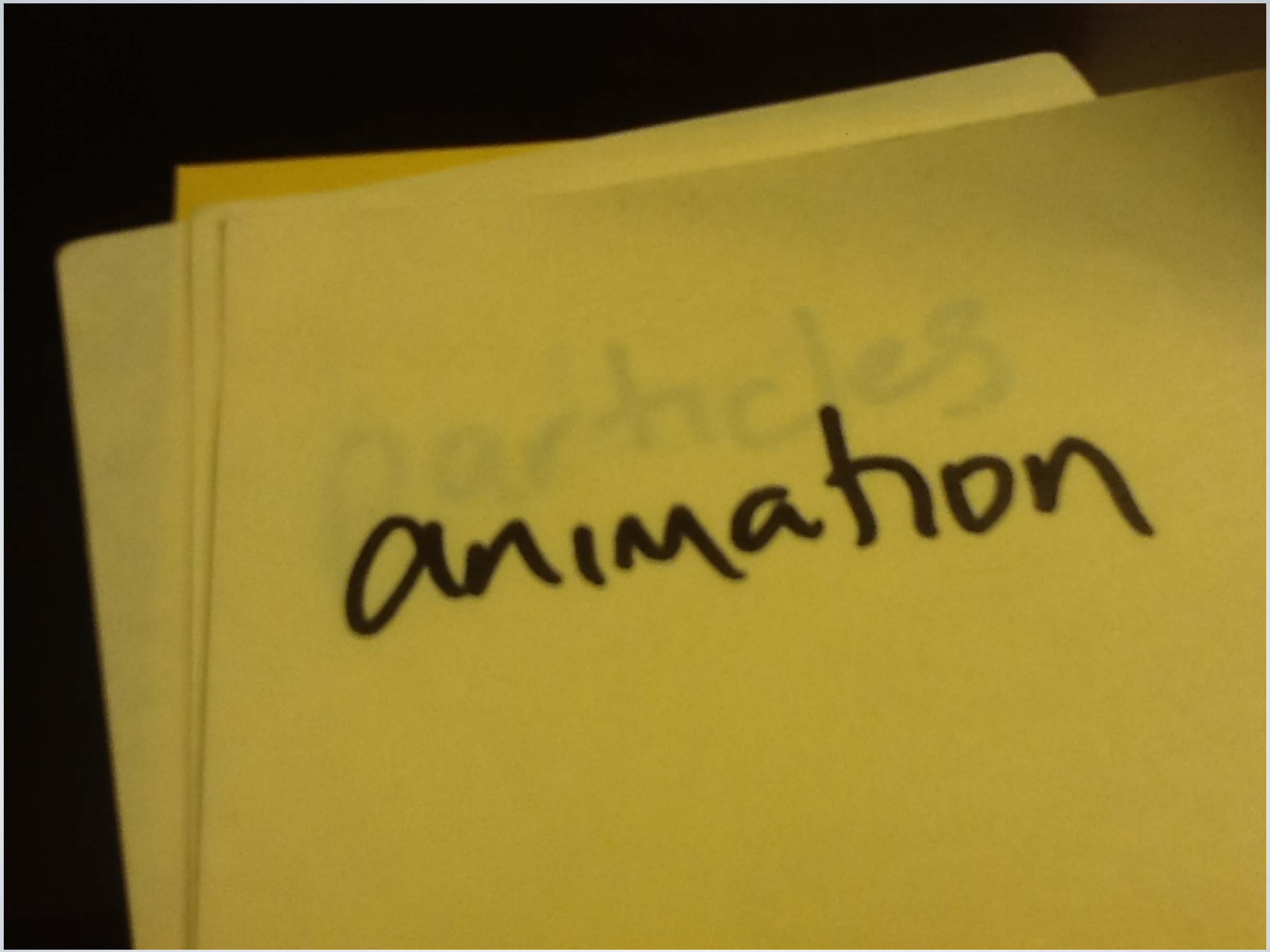

Very likely:
You already
do it (to
some degree)


But it's
not the
exception


Lights


Shadders


particles


animation

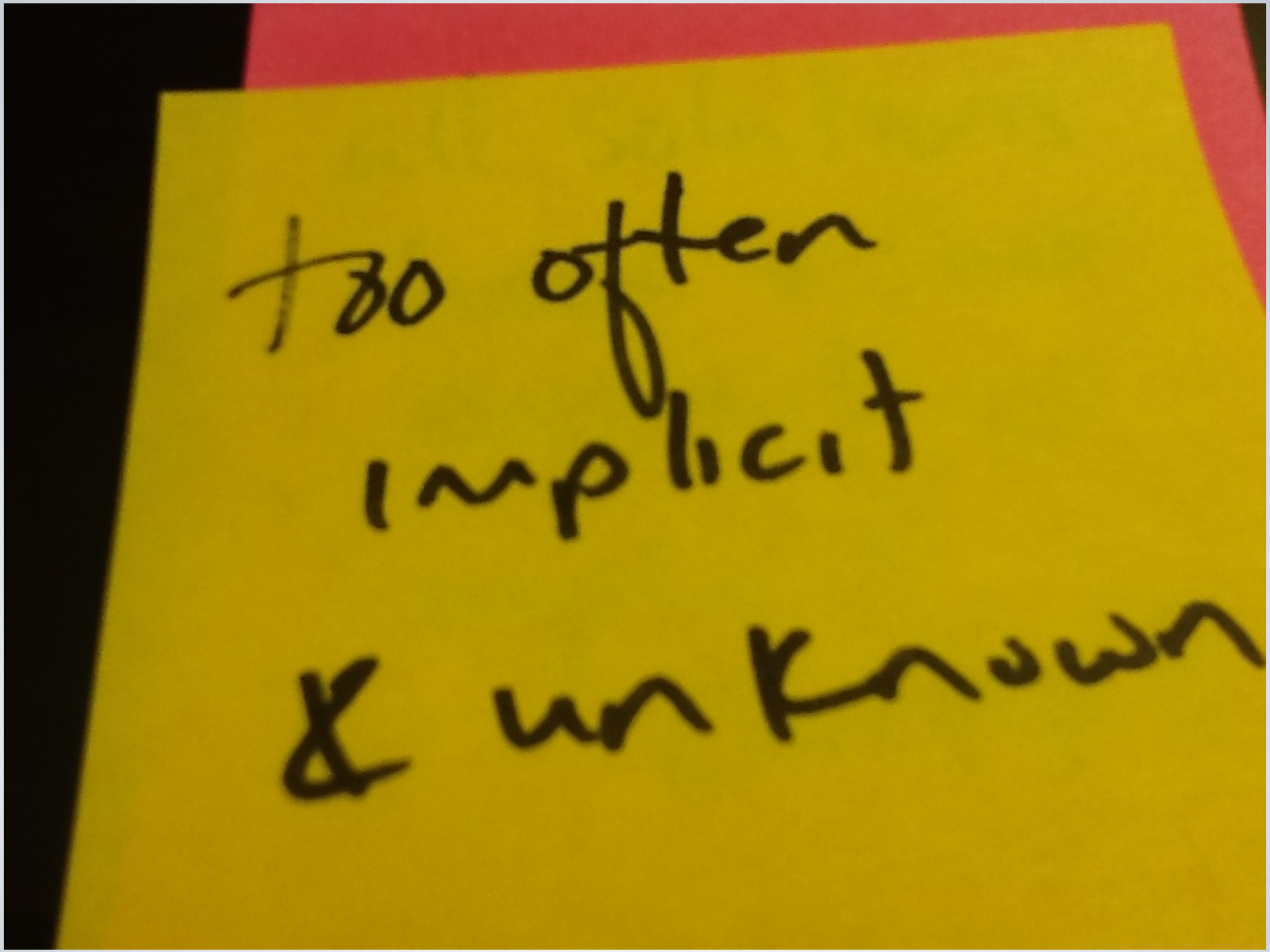
calling.

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

10?

100?

1000?

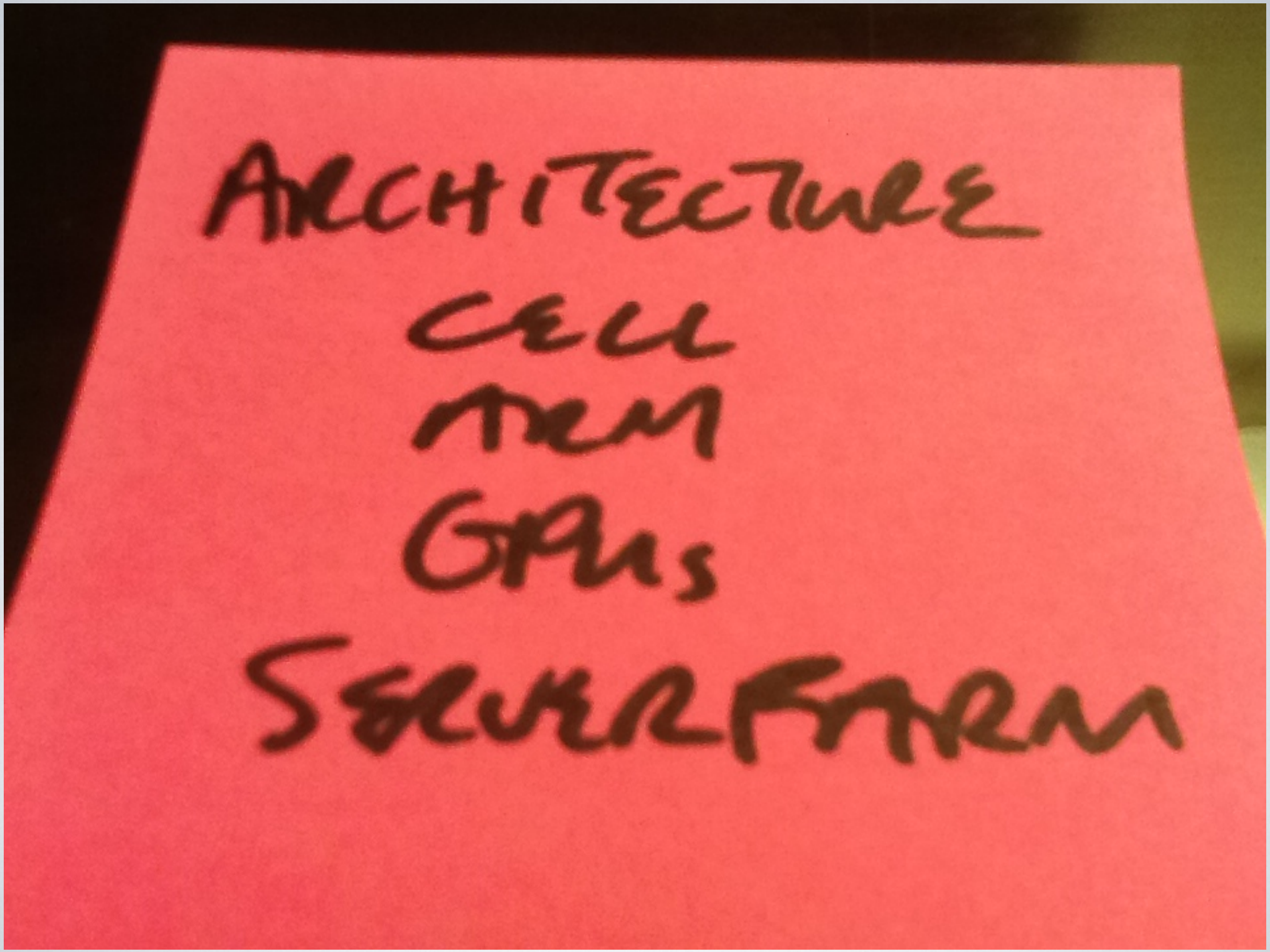
10,000?

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
TEM
GPUs
SOFTWARE

TRY TO SOLVE
RL PROBLEMS,
SOLVE VERY
FEW.

WHAT CASE TO SOLVE FOR?

WORST?

BEST?

MOST COMMON?

EDGE?

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 =

PROBLEM

ass Culle

NONE

P1

amiphish - 92 lines - codepad

codepad [create a new paste]

Link: http://codepad.org/u3yqprPJ [raw code | output | fork]

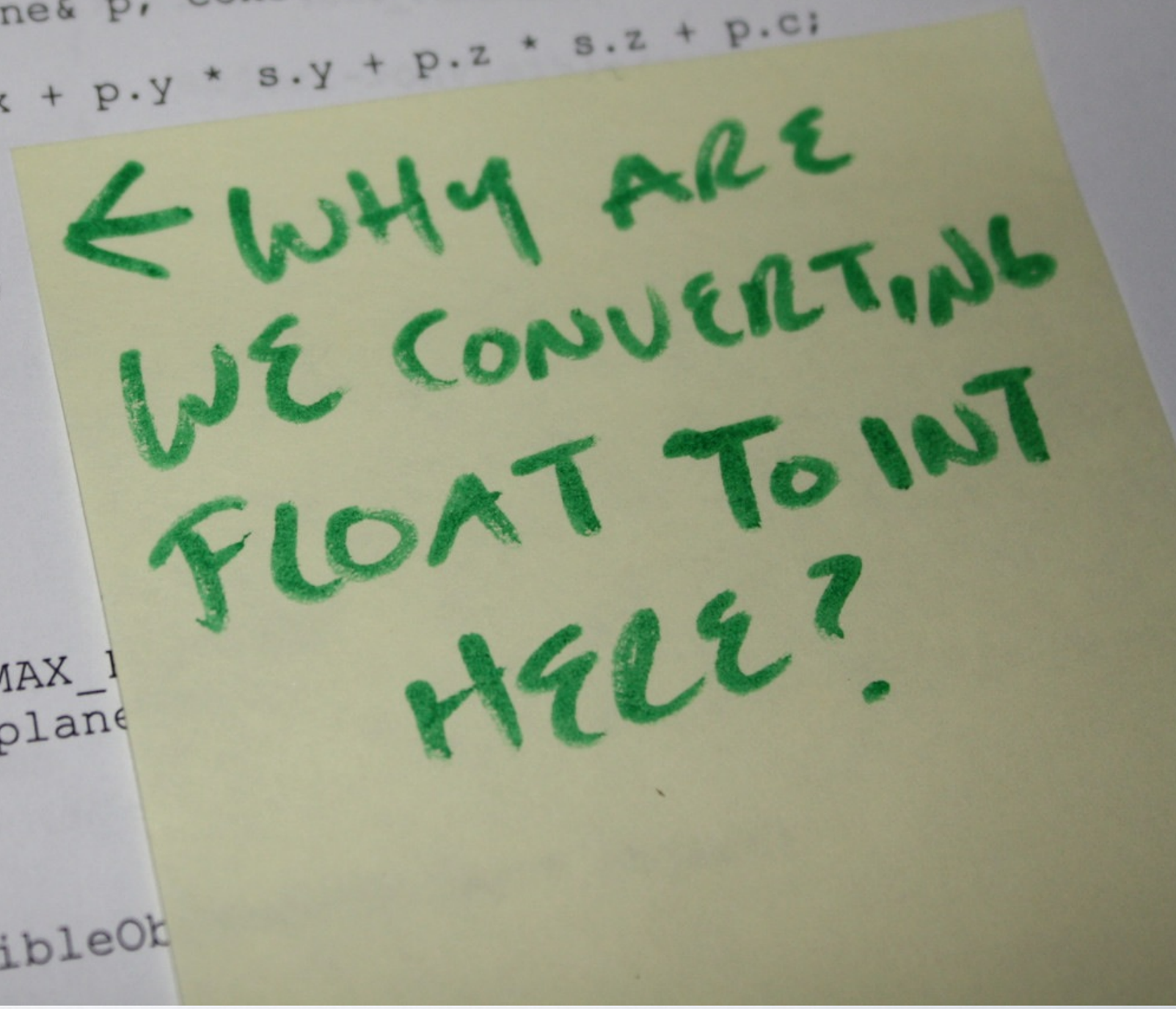
kamiphish - C++, pasted 1 hour ago:

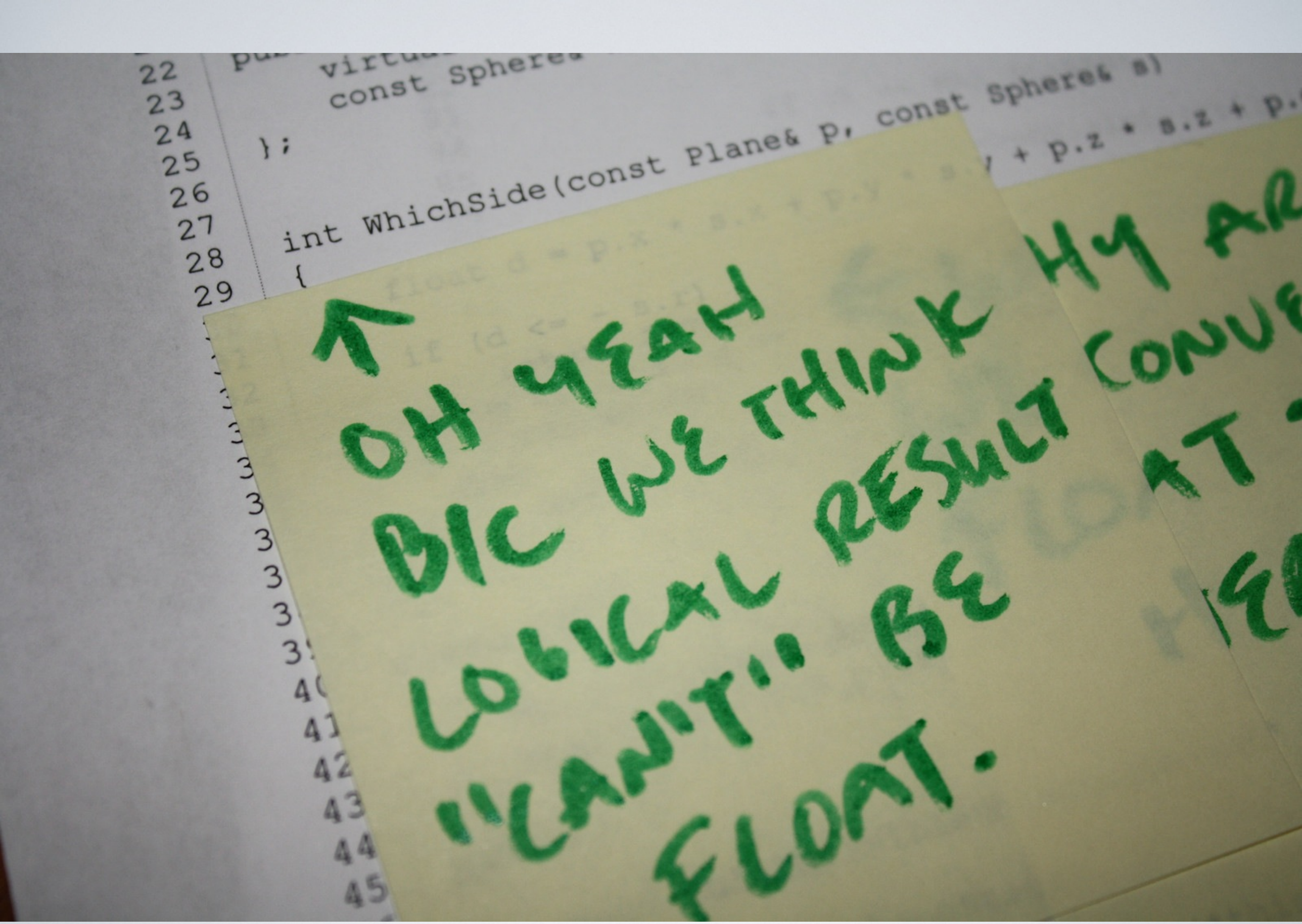
```
1   #define MAX_PLANES 6
2
3   enum { NONE, IN, OUT };
4
5   struct Sphere
6   {
        float x, y, z;
        float r;
```

TYPICAL.

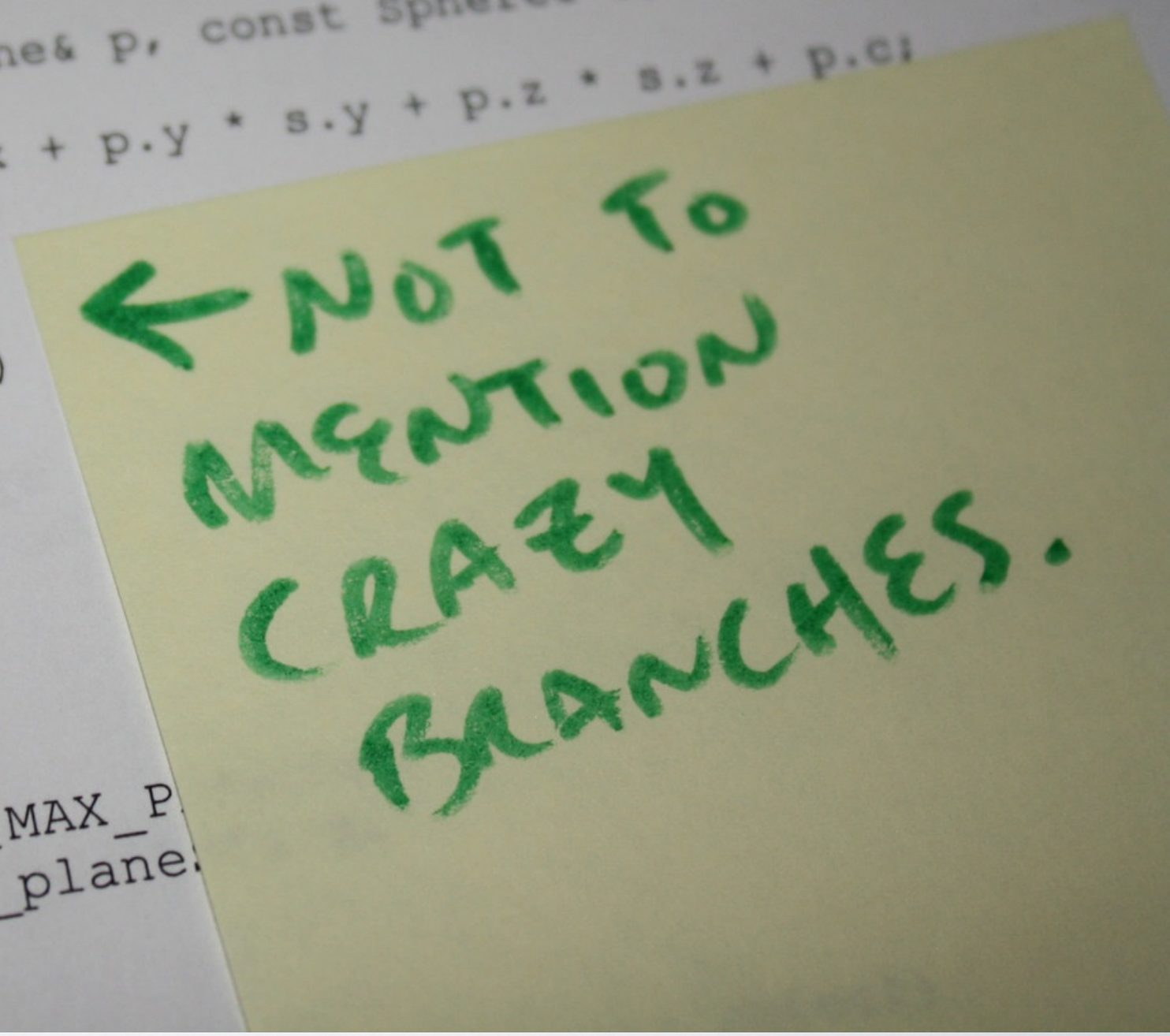
```
20  class Object
21  {
22  public:
23      virtual void onVisible(Culler&) = 0;
24      const Sphere& WorldBound() const;
25  };
26
27
28  int WhichSide(const Plane& p, const Sphere& s)
29  {
30      float d = p.x * s.x + p.y * s.y + p.z * s.z + p.c;
31
32      if (d <= - s.r)
33          return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

```
{
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_
    unsigned int m_plane

public:
    void CullInvisibleOb
```

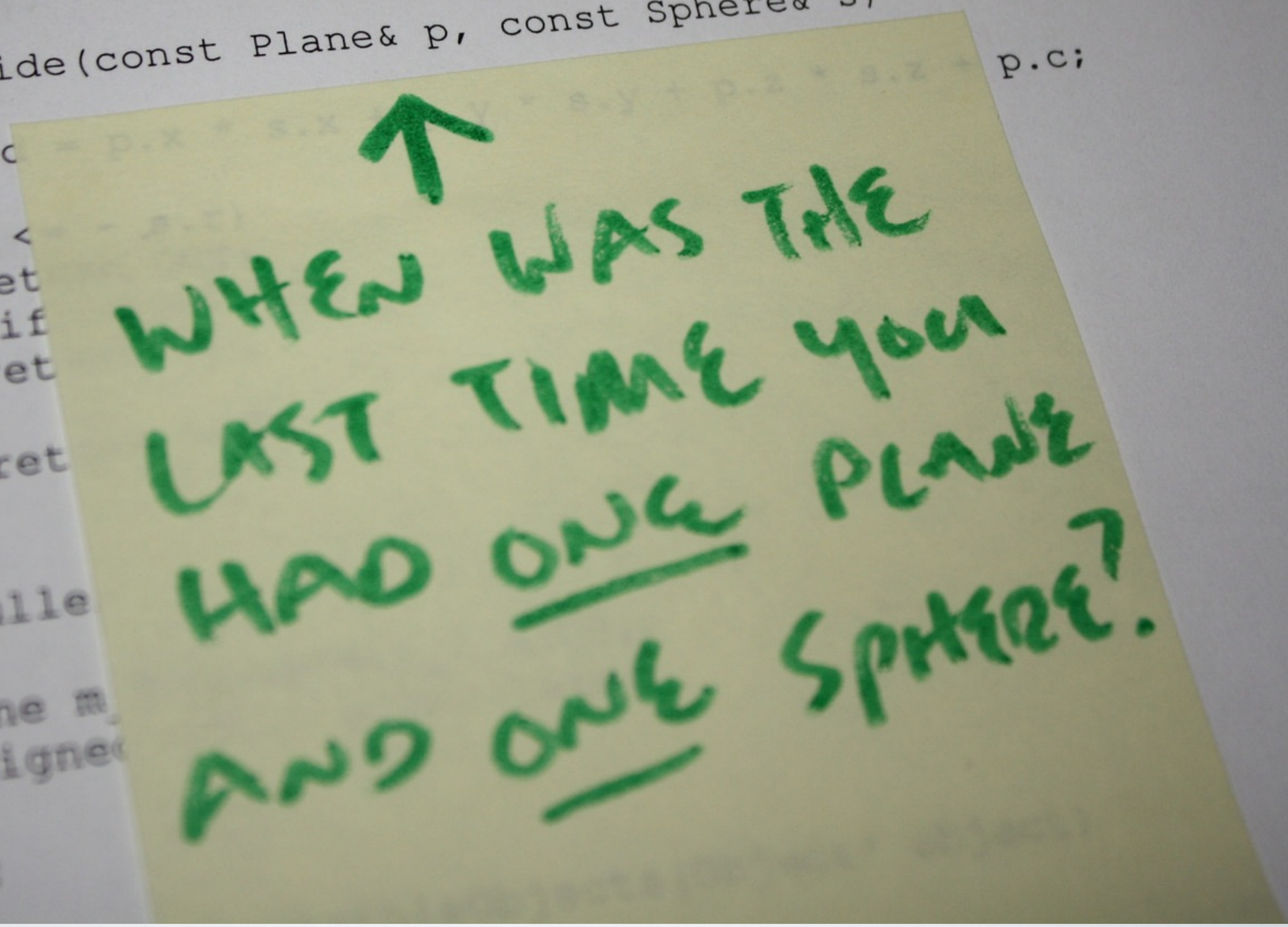



const Sphere
};
int WhichSide(const Plane& p, const Sphere& s)
{
+ p.z * s.z + p.
OH YEAH
WE THINK
RESULT
BIC
LOGICAL
"CAN'T" BE
FLOAT.

```
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
```

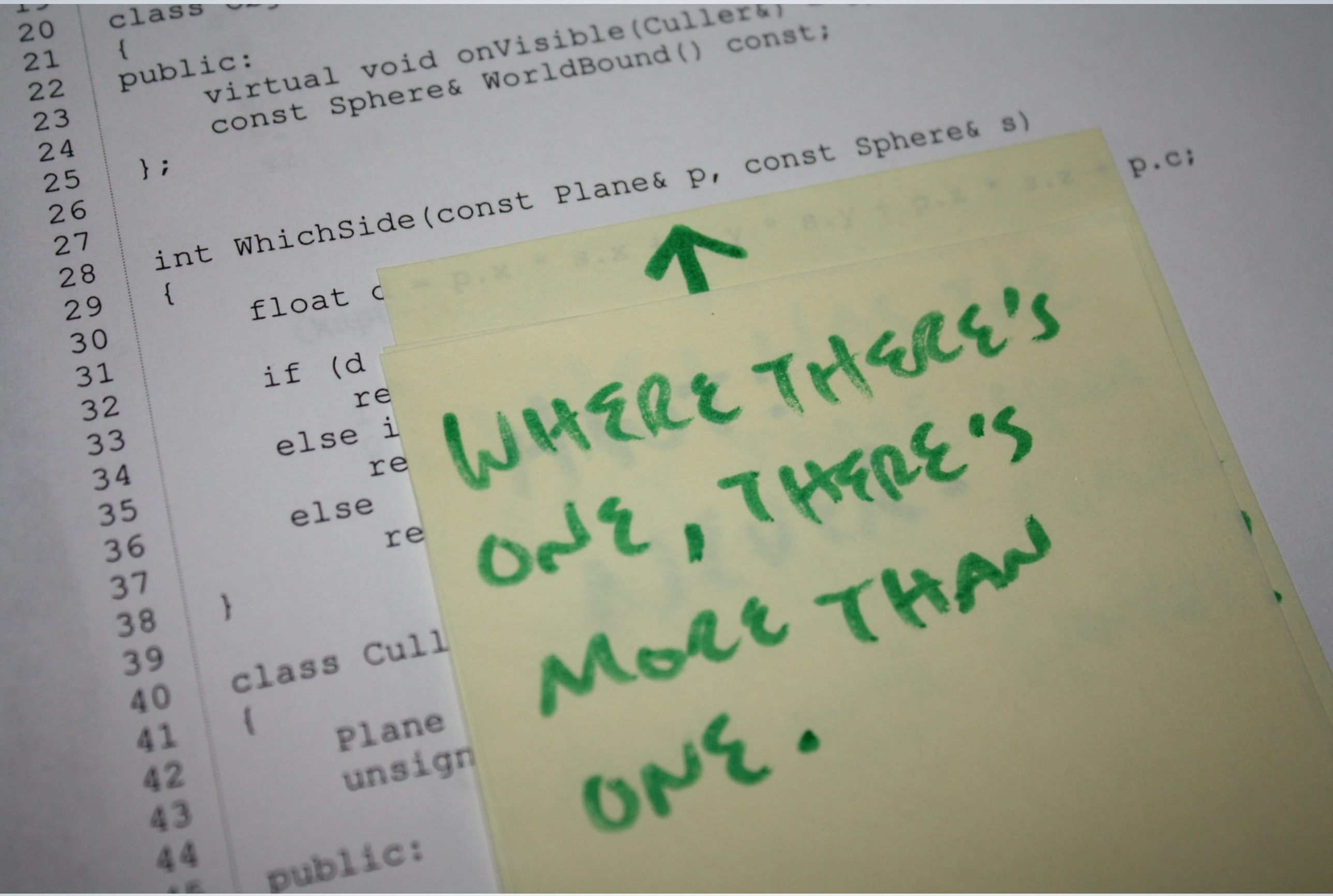


```
class Object
{
public:
    virtual void onVisible(Culler&) = 0;
    const Sphere& WorldBound() const;
};


int WhichSide(const Plane& p, const Sphere& s)
{
    float d                                          p.c;

    if (d <
        ret
    else if
        ret
    else
        ret
}

class Culle
{
    Plane m_
    unsigne

public:
```



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

HWT:
NEVER.


class
{
public:
virtual void onVisible(Culler&)
const Sphere& WorldBound() const;
};
int WhichSide(const Plane& p, const Sphere& s)
{
float
if (d
re
else i
re
else
re
}
class Cull
{
Plane
unsign
public:
p.c;
WHERE THERE'S
ONE, THERE'S
MORE THAN
ONE.

```
23      const Sphere&
24
25  };
26
27
28  int WhichSide(const Plane& p, const Sphere& s)
29  {
30      float d
31
32      if (d
33          re
34      else i
35          re
36      else
37          re
38  }
39
40  class Cull
41  {
42      Plane
43      unsign
44
45  public:
46
47      void C
48      {
49          if
50          {
51
```

DON'T BUILD FUNC PO... FOR L .01% CASE HERE AS "GENERAL" SOLUTION.

...ject is not visib...

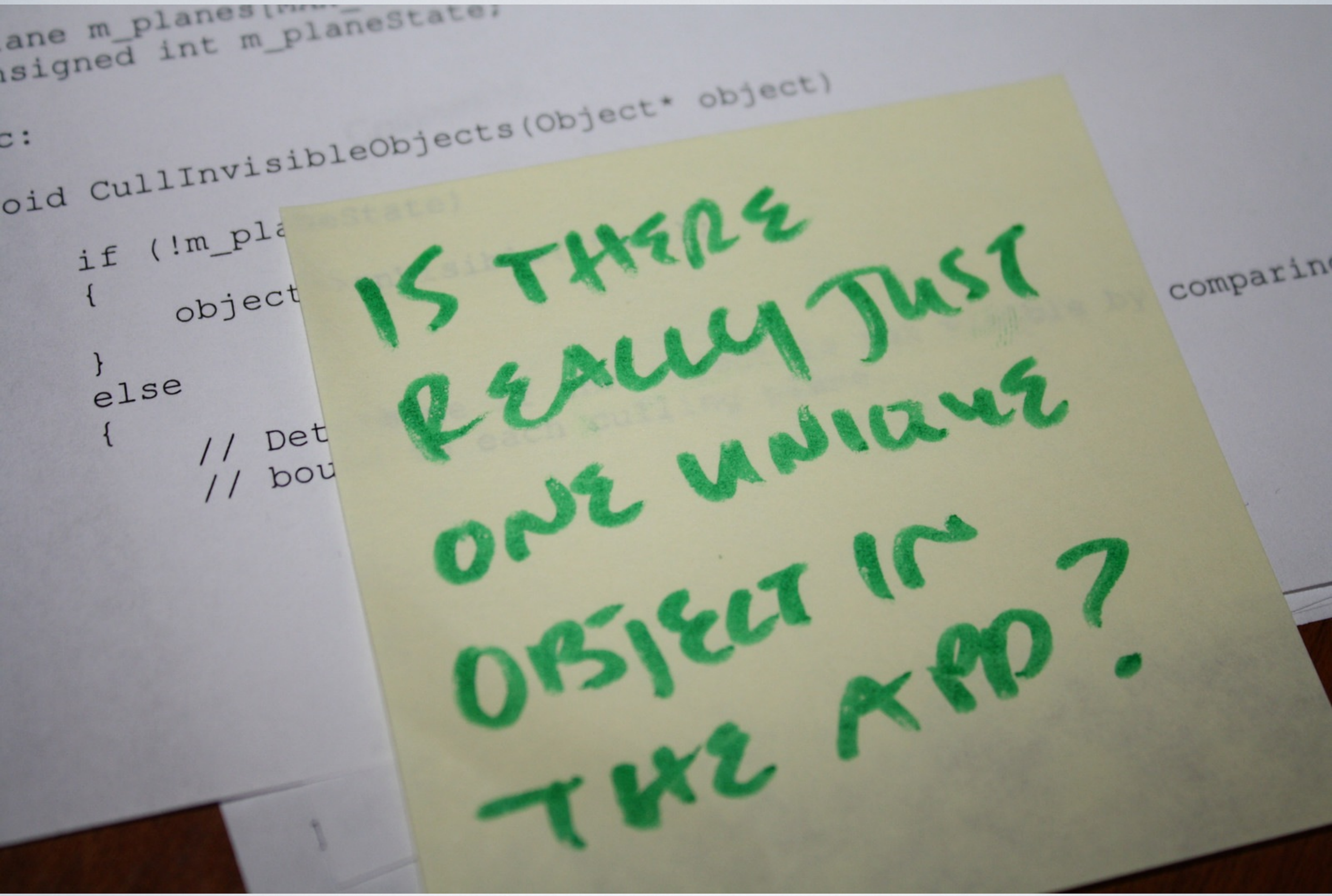
```
t WhichSide(const Plane& p, const Spl
   float c
   if (d
         re
   else i
         re
   else
         re
}
class Cull
{     Plane
      unsign
public:
   void C
```

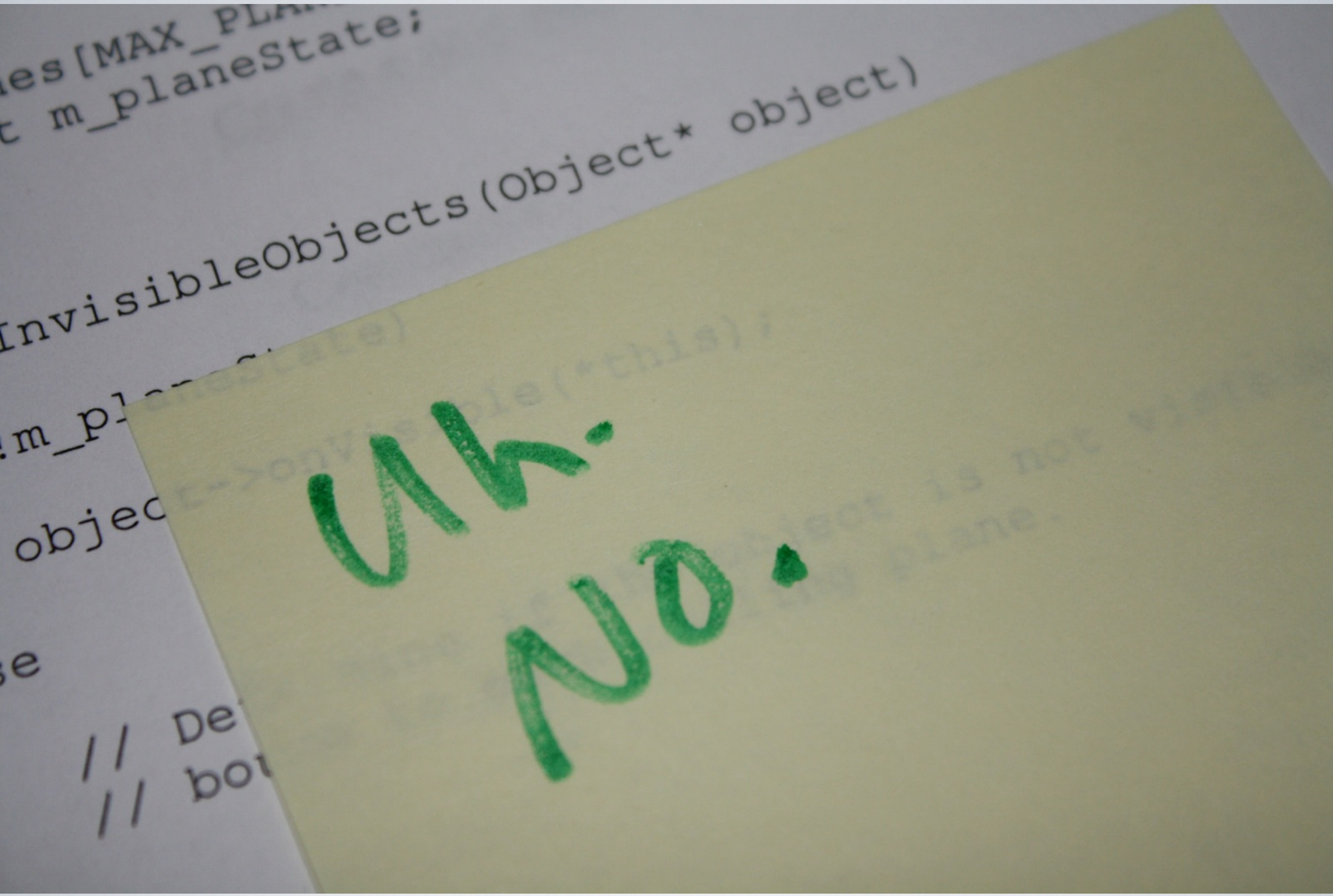
THE "GENERAL" CASE MANY

```
}

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.
```

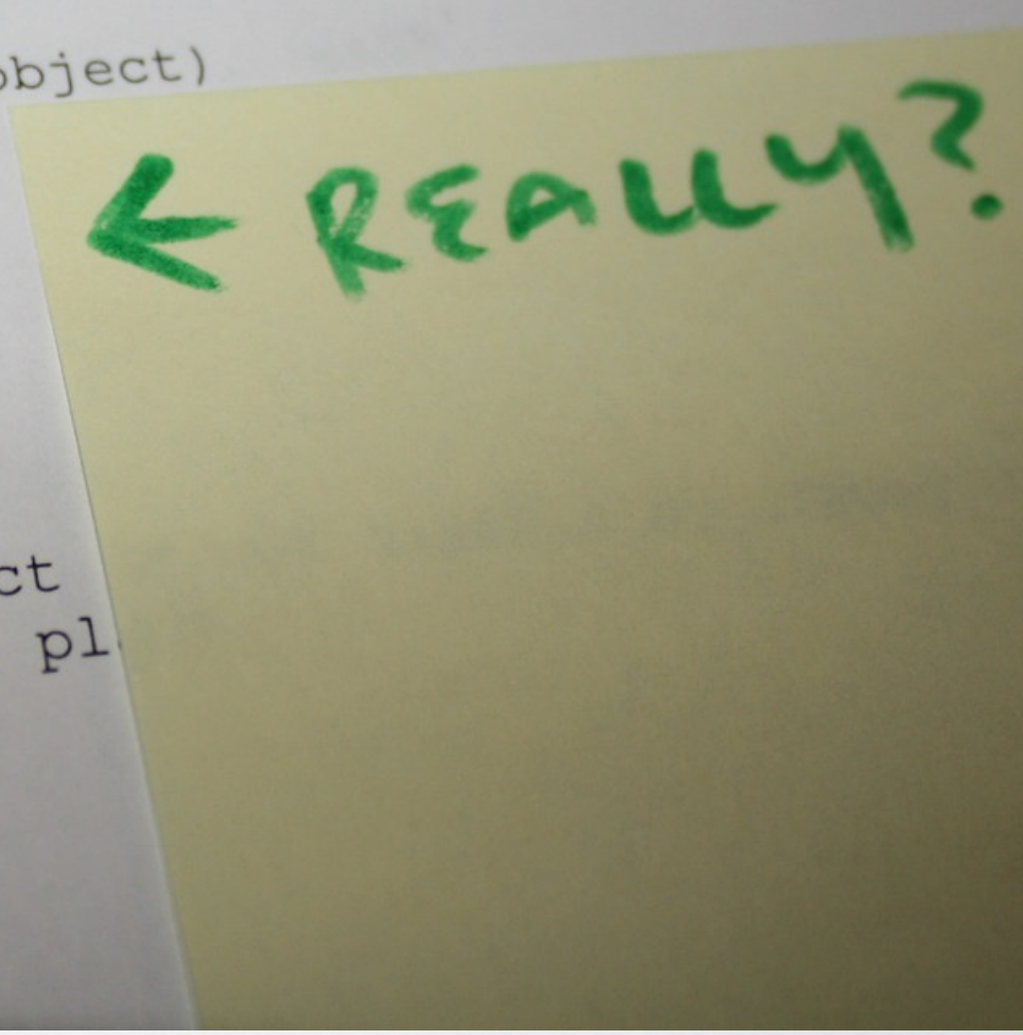
```
    else
        return NONE;


class Culler
{
    Plane m_planes[MAX_PLANES];
    unsigned int m_planeState;

public:

    void CullInvisibleObjects(Object* object)
    {
        if (!m_P
        {
            obje

        }
        else
        {
            // I
            // }
```

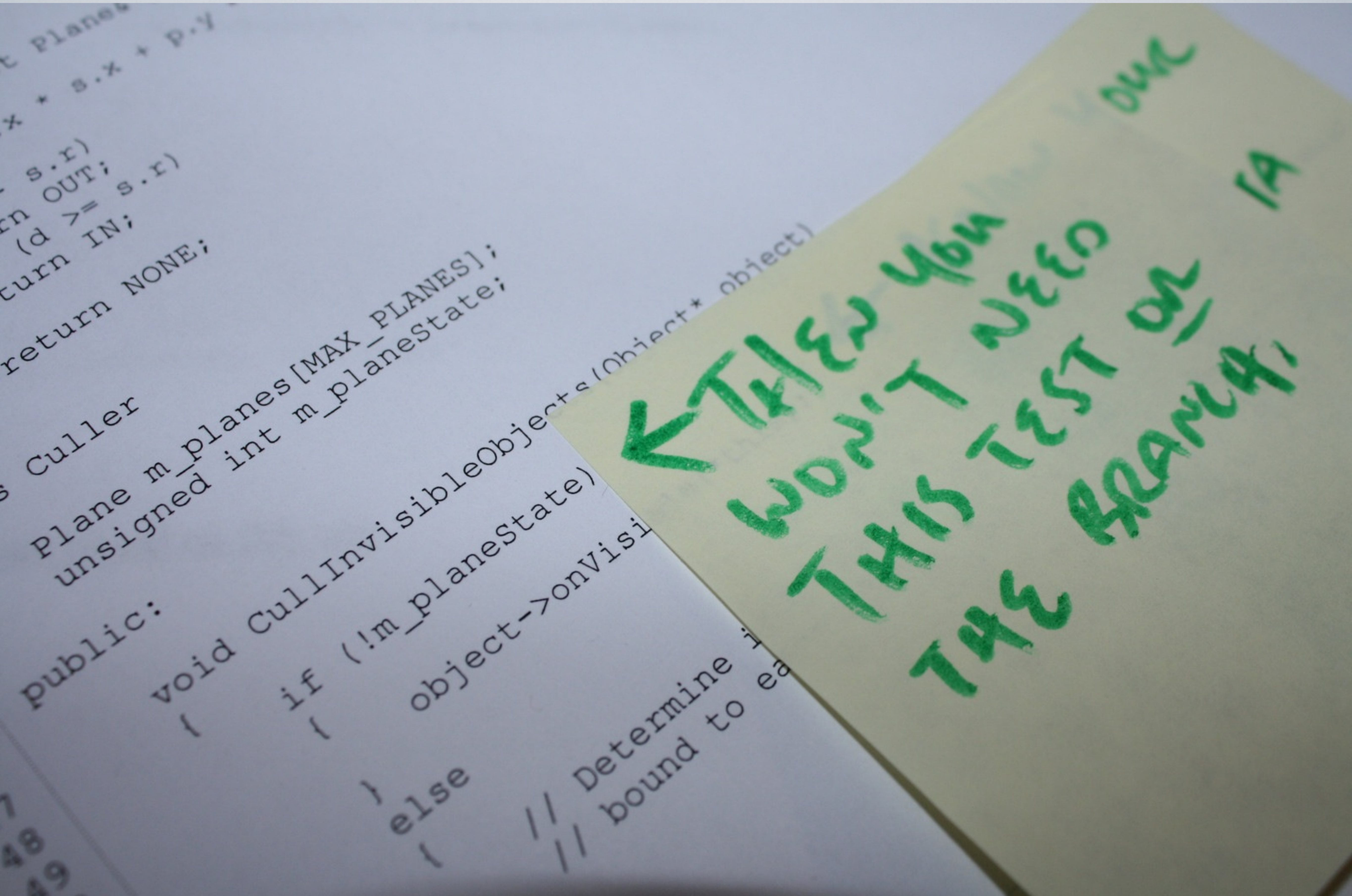
SAME B.S.


ane m_planes
nsigned int m_planeState;
c:
oid CullInvisibleObjects(Object* object)
if (!m_pla
{
object
}
else
{
// Det
// bou
comparin
IS THERE
REALLY JUST
ONE VARIABLE
OBJECT IS
THE AXE?


es[MAX_PLA
t m_planeState;
Invisible0bjects(Object* object)
m_pl
objec
se
// De
// bo
No.

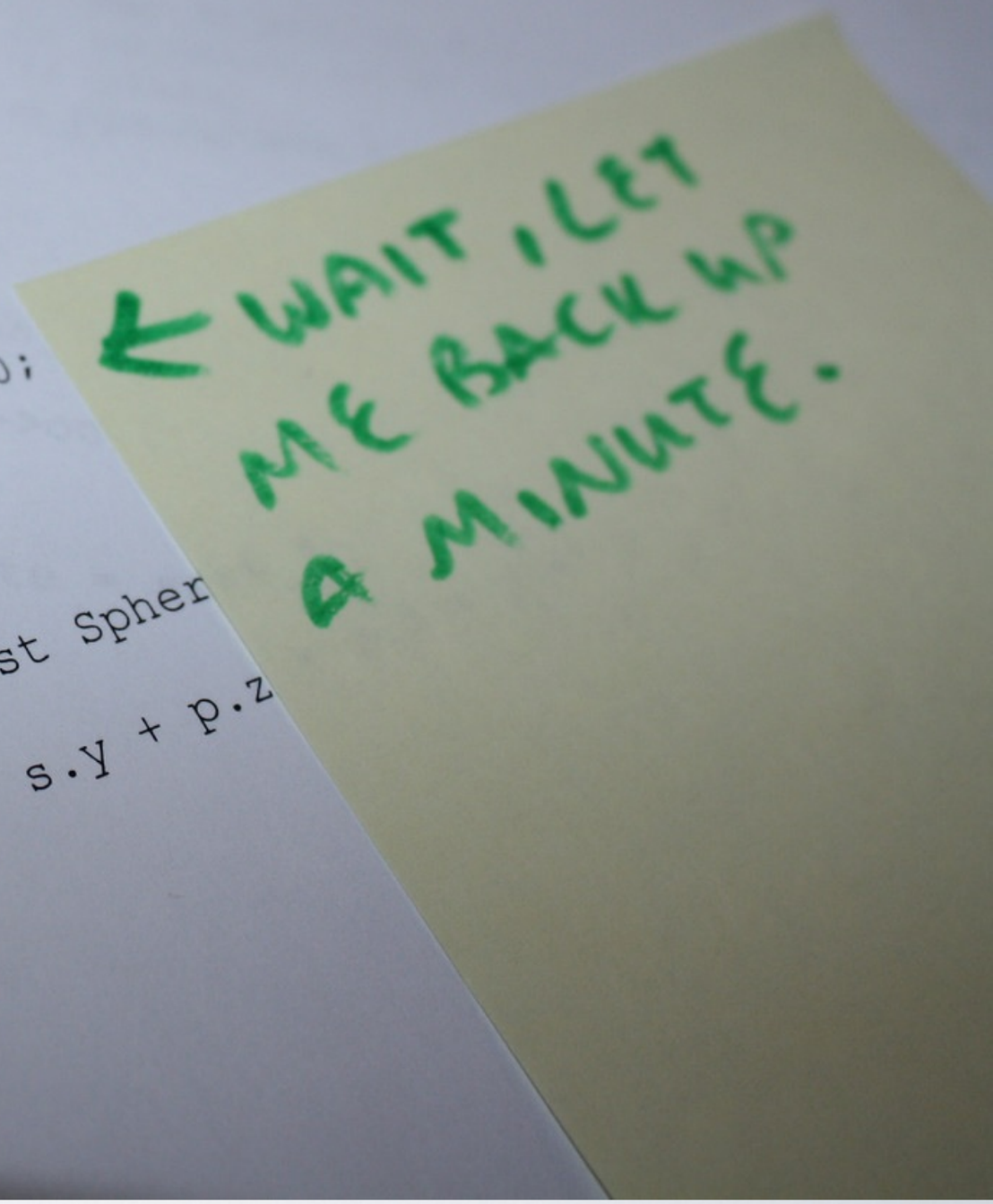
```
    if (d <= - s.r)
        return OUT;
    else if (d >= s.r)
        return IN;
    else
        return NONE;


lass Culler

    Plane m_planes[MAX_PLANES];
    unsigned int m_planeState;

blic:

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

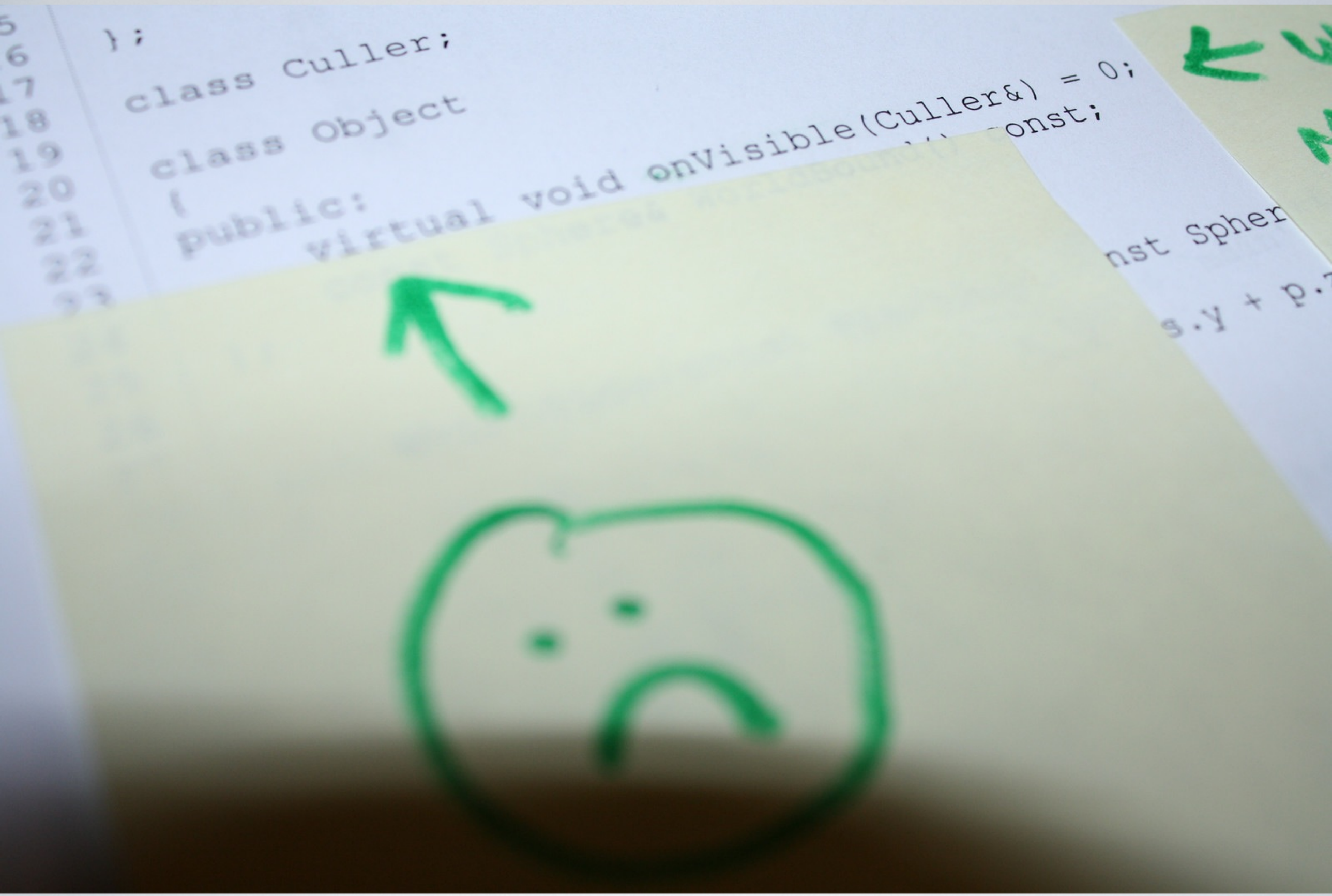


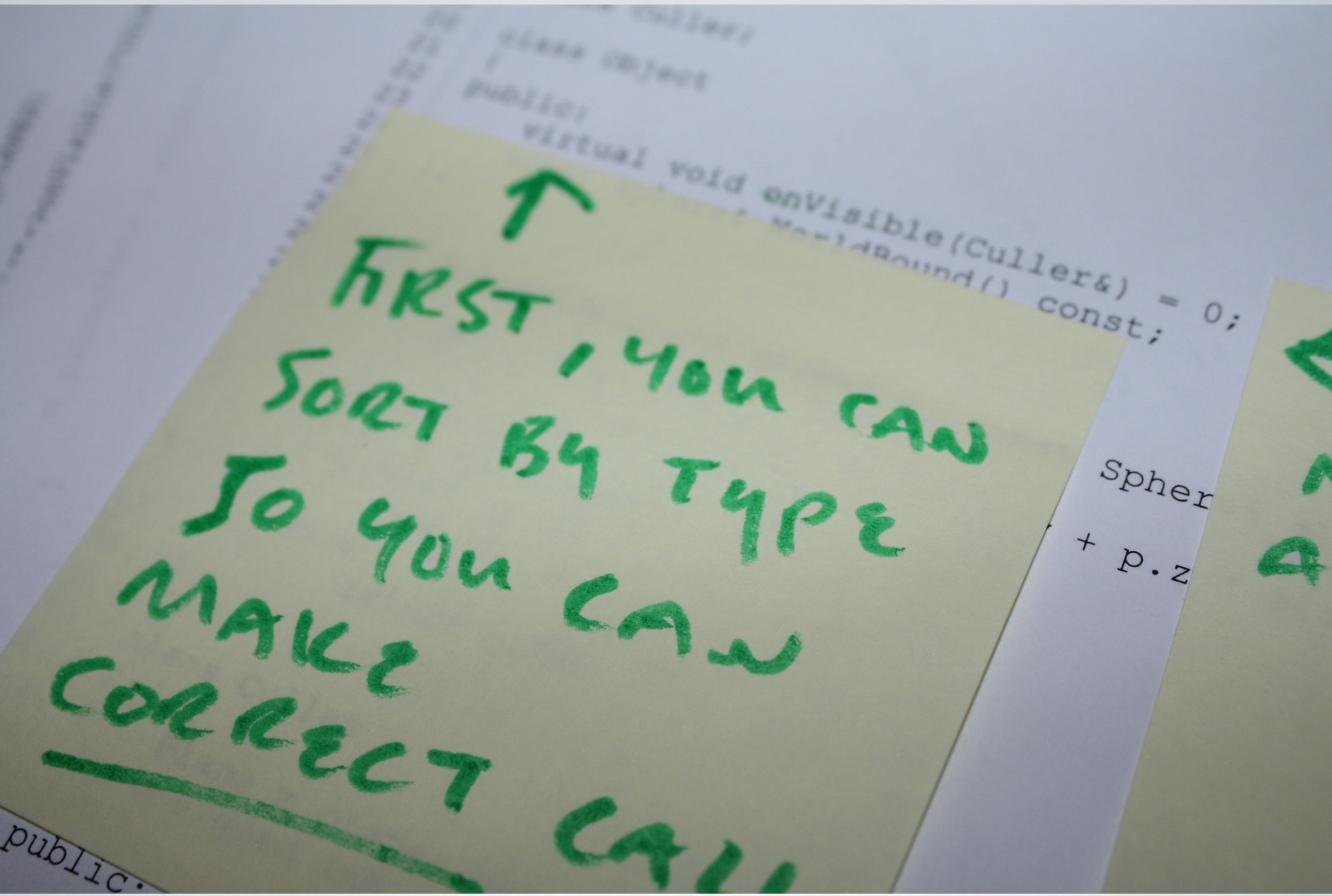


return NONE;
Culler
Plane m_planes[MAX_PLANES];
unsigned int m_planeState;
public:
void CullInvisibleObjects(Object* object)
if (!m_planeState)
object->onVisi
else
// Determine
// bound to ea
← THEN YOU WON'T NEED THIS TEST OR THE BRANCH.

```
NONE, IN, OUT );

Sphere

loat x, y, z;
loat r;


.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;
                    s.r)
```

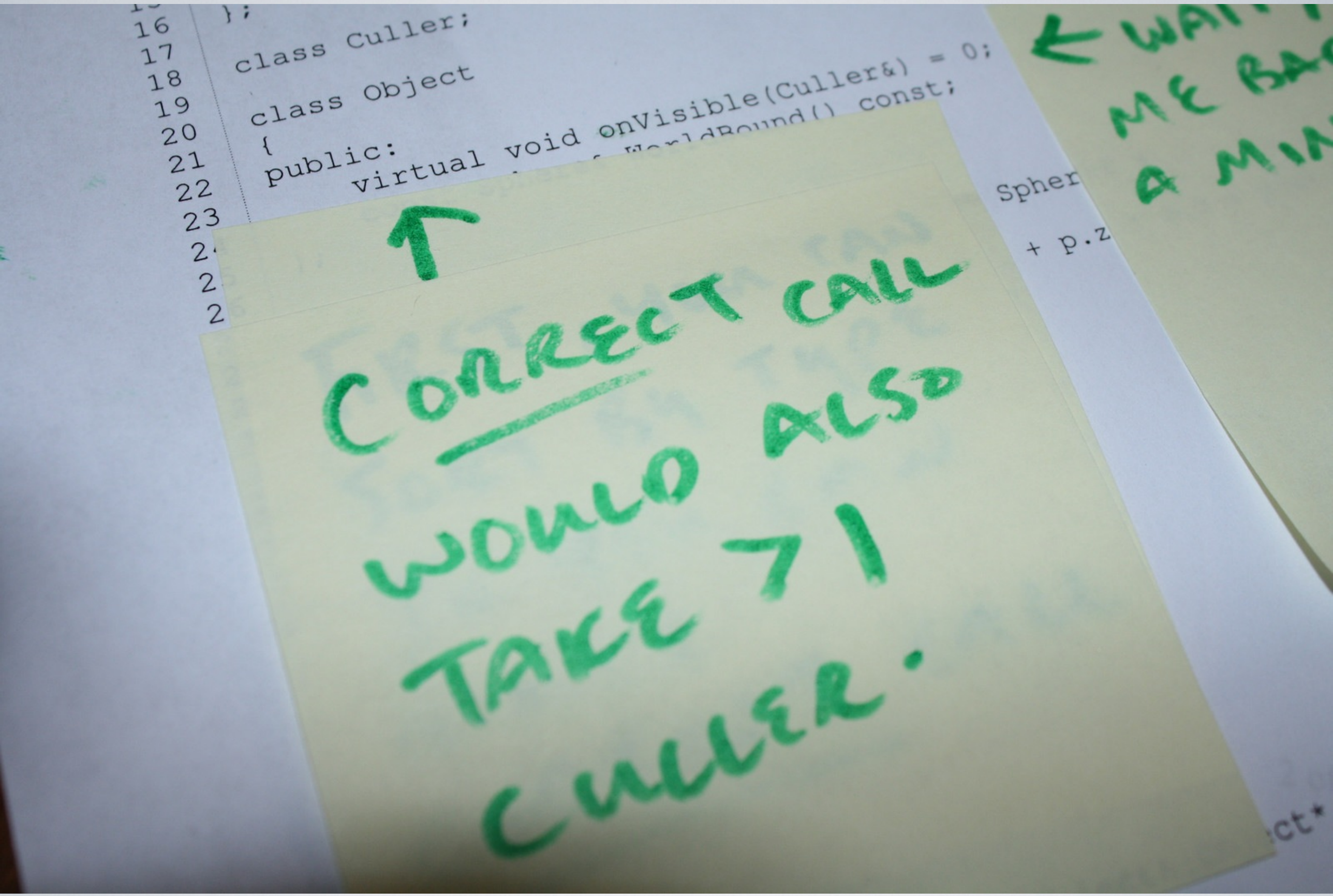


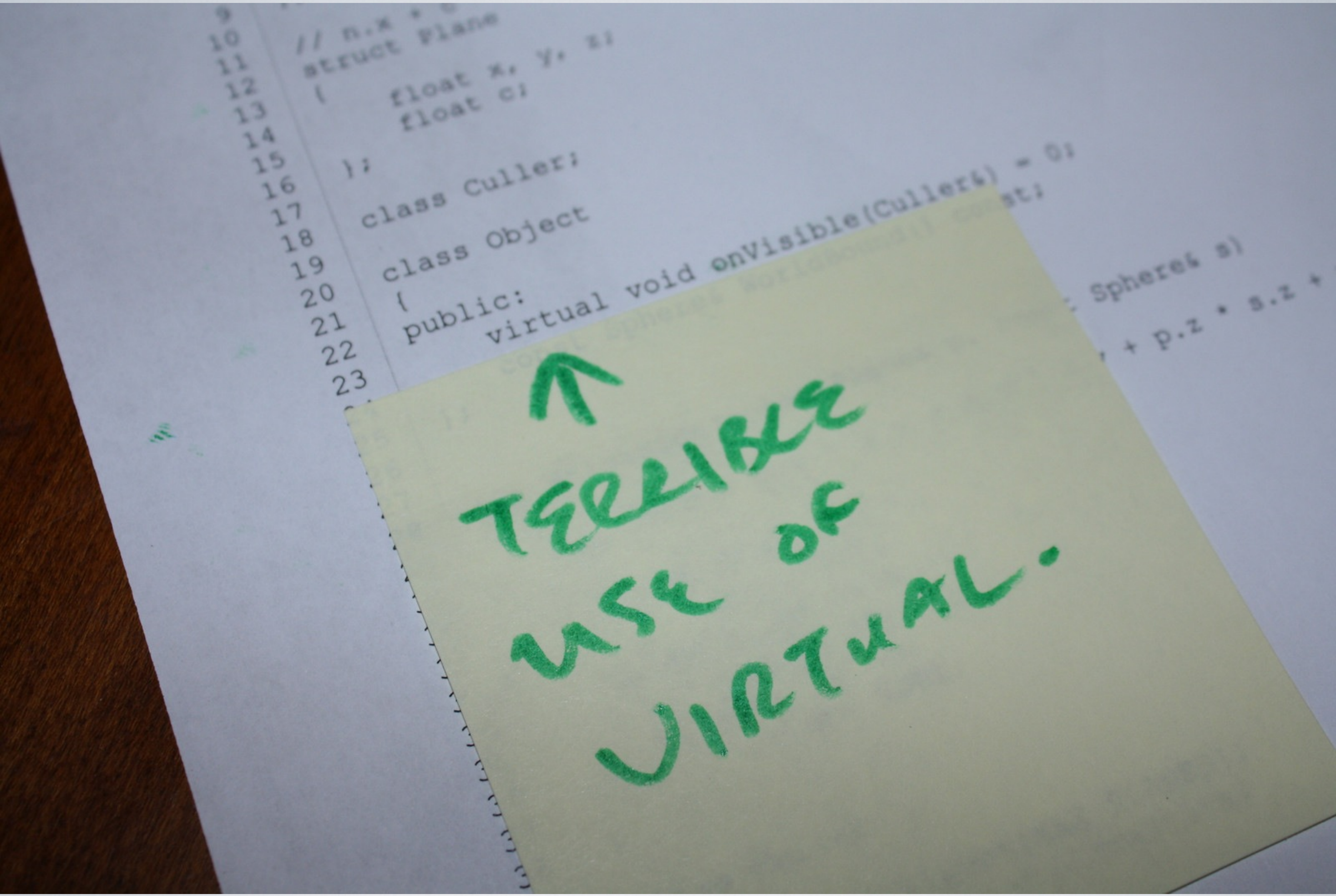

};
class Culler;
class Object
{
public:
virtual void onVisible(Culler&) = 0;
onst;
nst Spher
s.y + p.

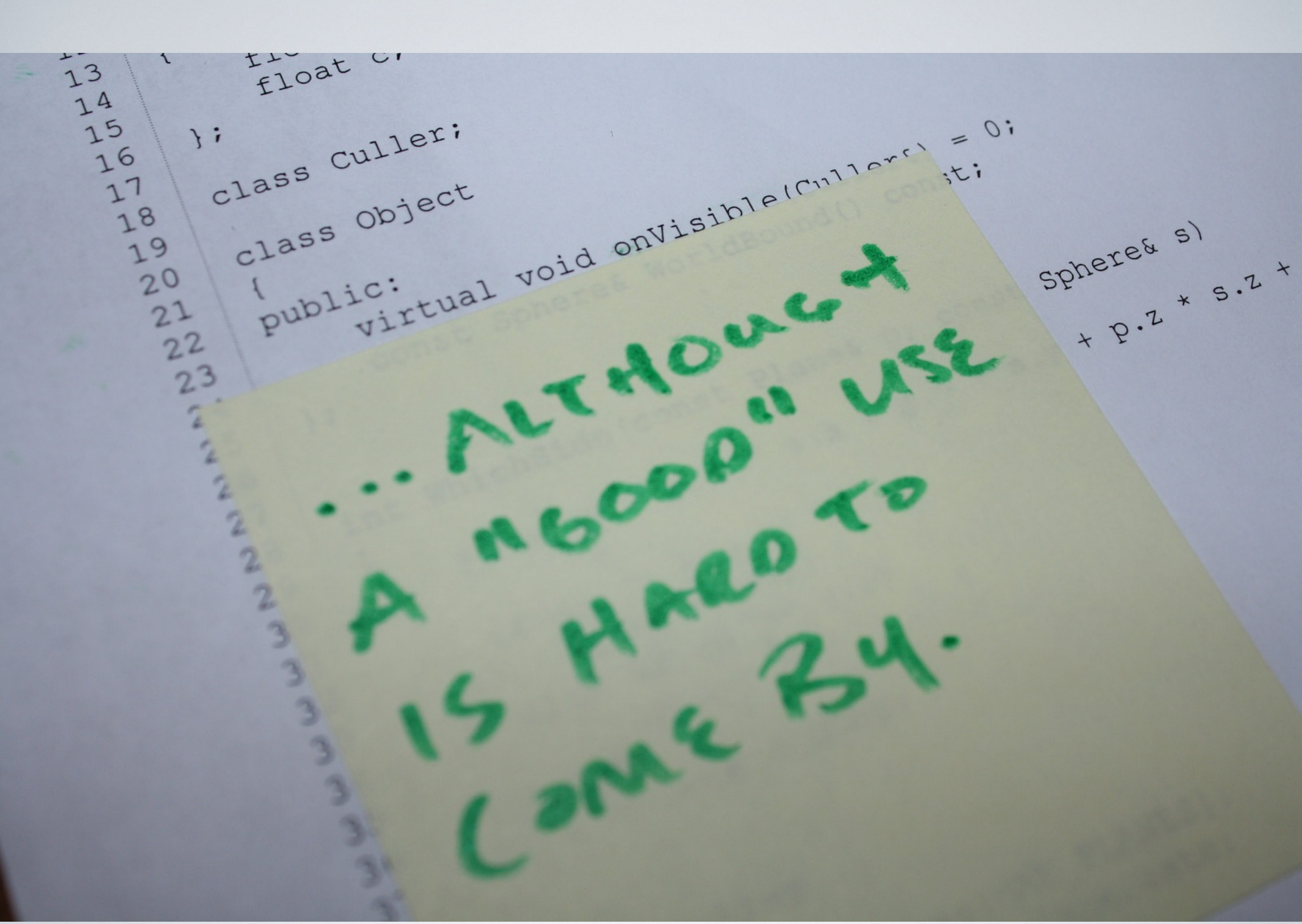

virtual void onVisible(Culler&) = 0;
FIRST, YOU CAN
SORT BY TYPE
SO YOU CAN
MAKE
CORRECT

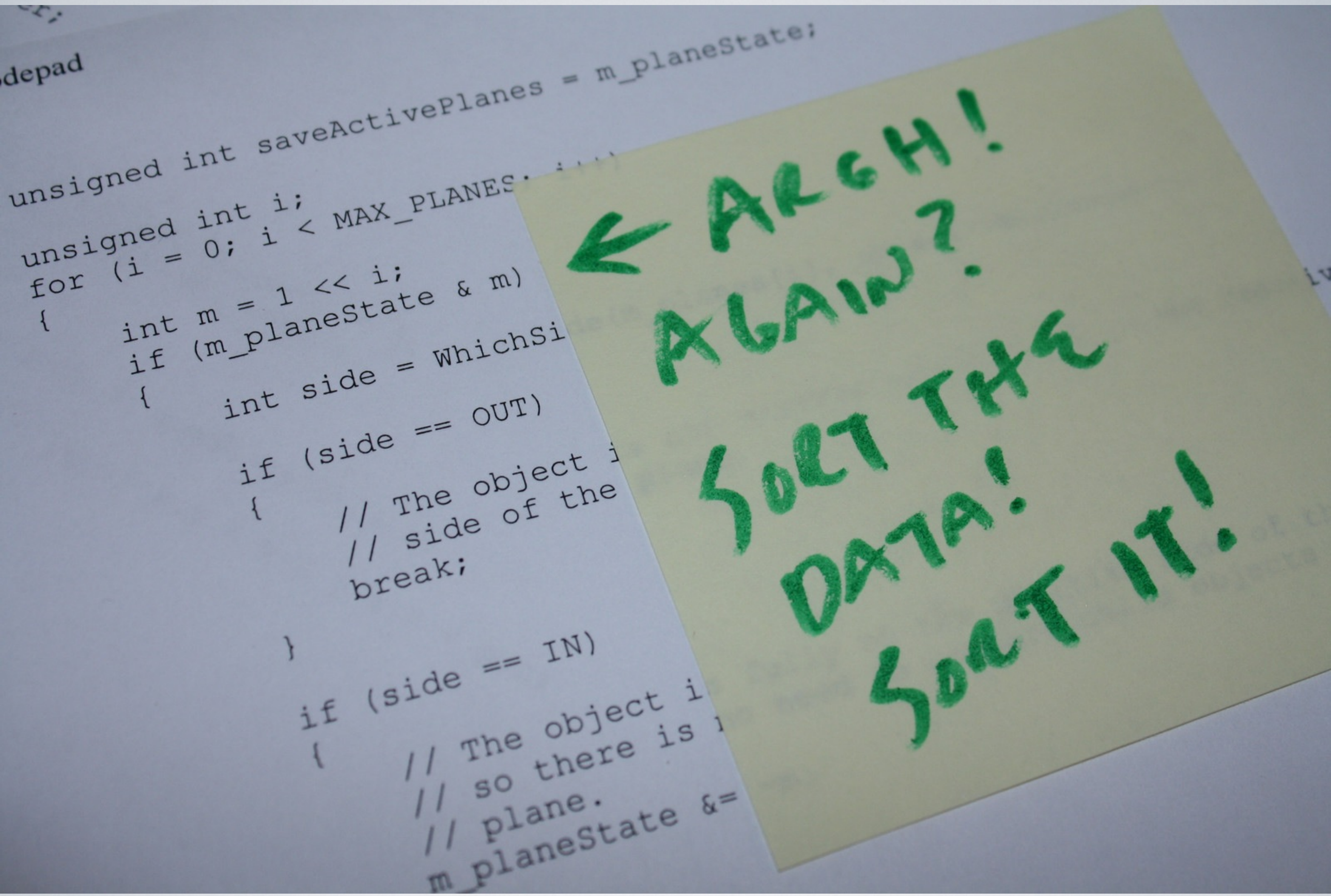
```
16  };
17
18  class Culler;
19
20  class Object
21  {
22  public:
23      virtual void onVisible(Culler&) = 0;
24      virtual WorldBound() const;
25
26
```

```
Spher
+ p.z
```




struct Plane
{
float x, y, z;
float c;
};
class Culler;
class Object
{
public:
virtual void onVisible(Culler&) = 0;
Sphere& s)
TERRIBLE
USE OF
VIRTUAL.


class Culler;
class Object
{
public:
virtual void onVisible(
Sphere& s)
+ p.z * s.z +
...Although
a good use
is hard to
come by.


depad
unsigned int saveActivePlanes = m_planeState;
unsigned int i;
for (i = 0; i < MAX_PLANES;
{
int m = 1 << i;
if (m_planeState & m)
{
int side = WhichSi
if (side == OUT)
{
// The object
// side of the
break;
}
if (side == IN)
{
// The object
// so there is
// plane.
m_planeState &=
ARGH!
AGAIN?
SORT THE
DATA!
SORT IT!

```
gned int saveActivePlanes = m_planeState;

gned int i;
 (i = 0; i < MAX_PLANES;

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

 if (side == OUT)

 if (side == IN)
```

← KNOW THE DATA!

HOW MANY PLANE ARRANGEMENTS DO YOU REALLY USE?

```
int saveActivePlanes = m_planeStat

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

 = 1 << i;
_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.
     planeState &=
```

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

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

↑

NOTE YOU CLEARLY HAVE 71 PLANE AT THIS PT.

```
                                                    it is on th
                                                    itive si
                                                    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

tive
chil

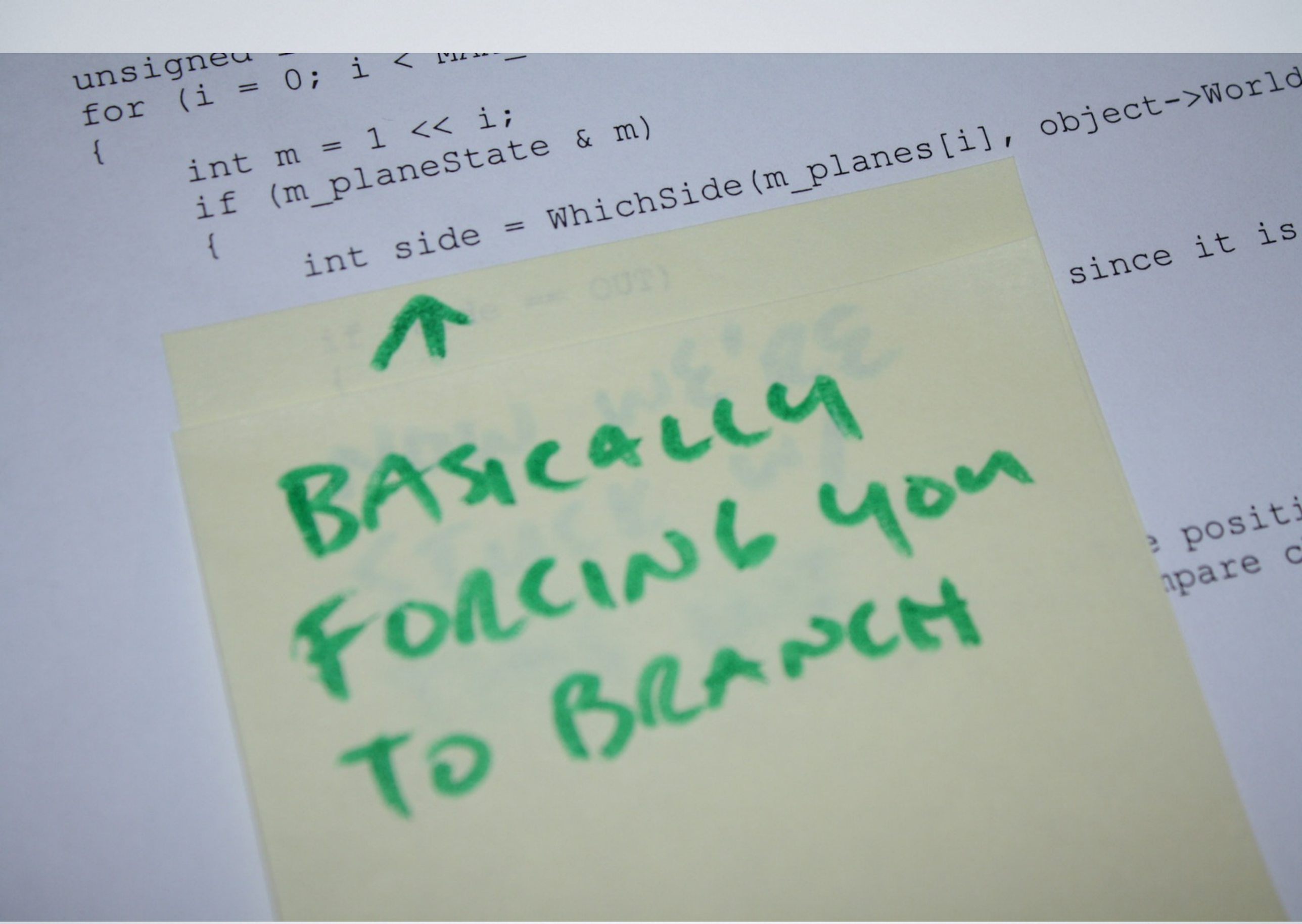
But still
calling
indv. func.
So lame!

```
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

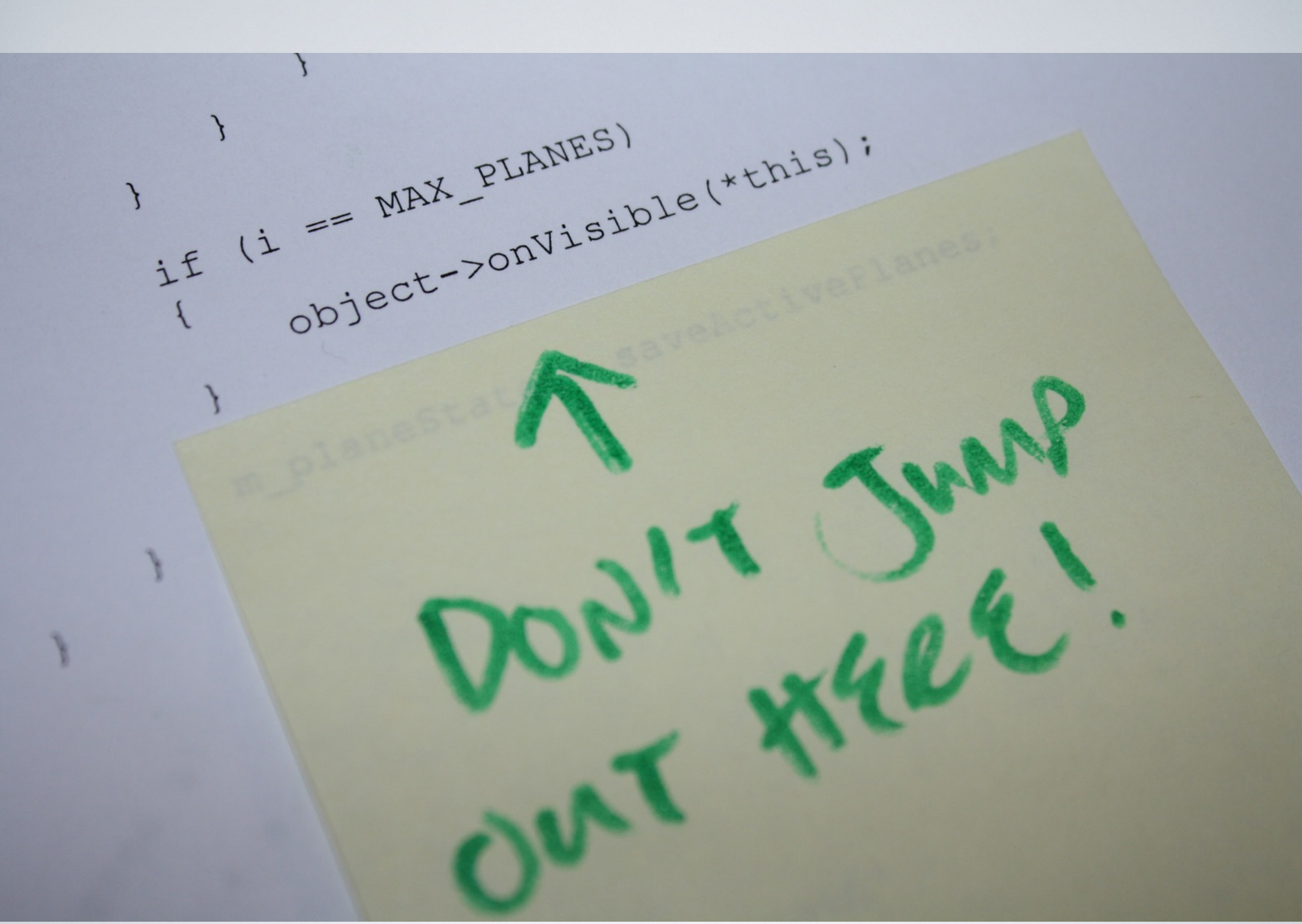
↑

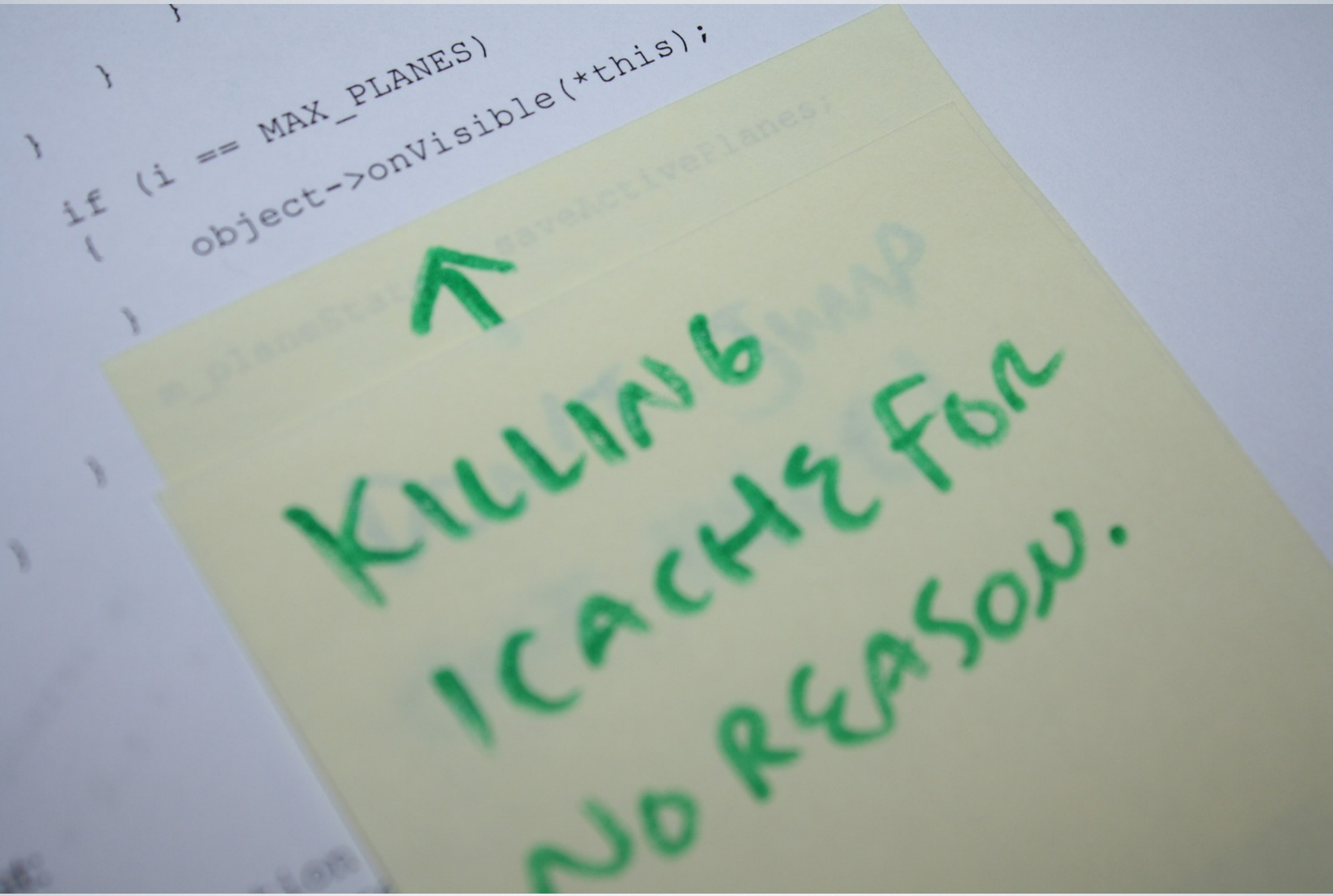
NOW WE'RE STUCK IN THIS INT

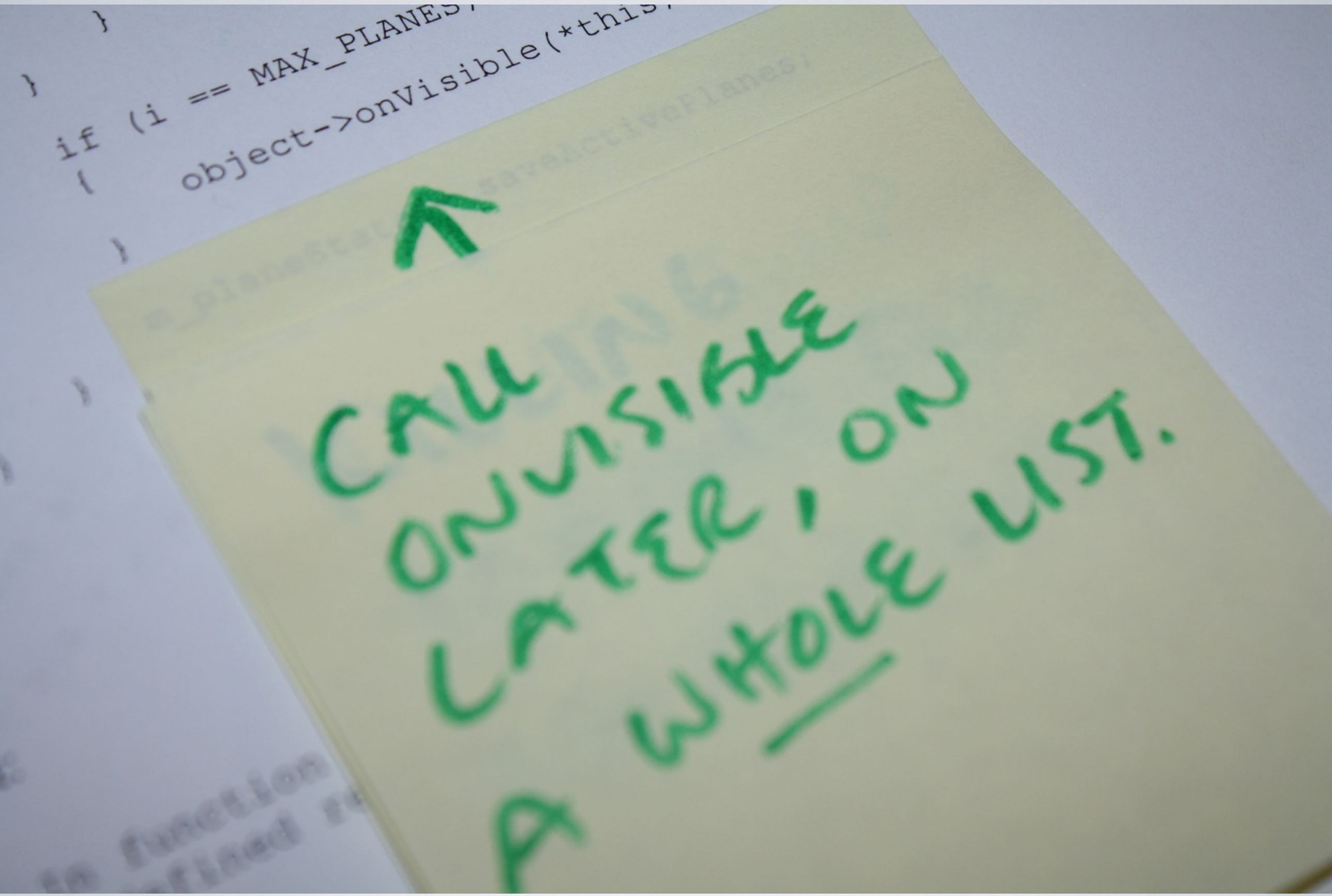

unsigned
for (i = 0; i <
{
int m = 1 << i;
if (m_planeState & m)
{
int side = WhichSide(m_planes[i], object->World
since it is
positi
mpare c
BASICALLY
FORCING YOU
TO BRANCH

```
int saveActivePla

 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 th
    break;

}

 if (side == IN)
 {
    // The object
    there i
```

LIKE THIS.


}
}
}
if (i == MAX_PLANES)
{ object->onVisible(*this);
}
Don't Jump
out there!


}
}
}
if (i == MAX_PLANES)
object->onVisible(*this);
Killing
icache for
no reason.


if (i == MAX_PLANES
object->onVisible(*this
CALL
ONVISIBLE
LATER,
A WHOLE LIST.

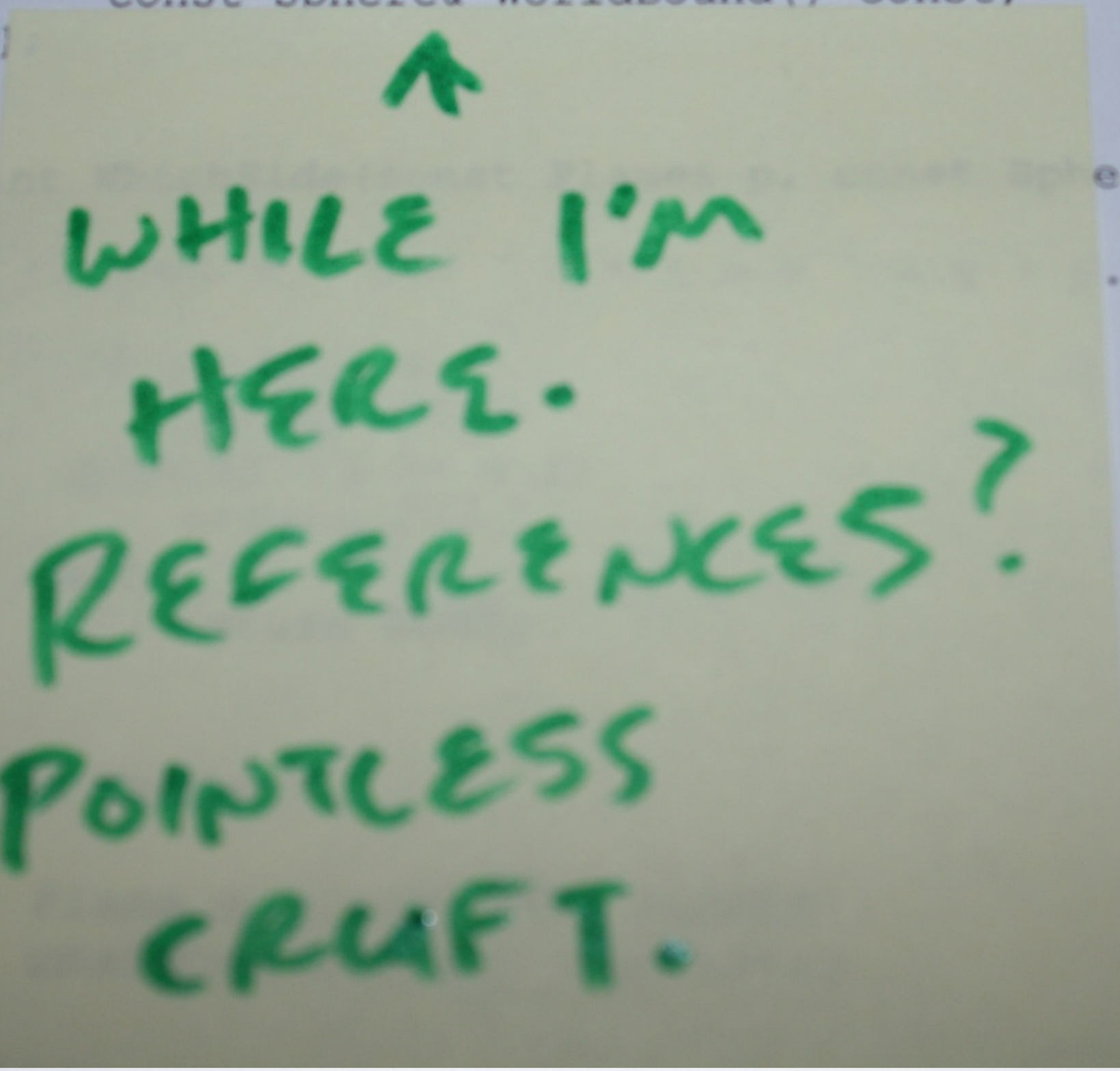
```
{       object->onVi
}

m_planeState = saveActivePlanes;
    }
}
```

↑

GHOST WRITES.

BAD FOR CACHE. BAD FOR THE SOUL.

```
22  public:
23      virtual void onVisible(Culler&) = 0;
24      const Sphere& WorldBound() const;
25  }
26
27
28  ere& s)
29
30                                  .z * s.z + p.
31
32
33
34
35
36
37
38
39
40
41
42
43
44
```




Mike Acton
#AltDevBlogADay #gamedev
@mike_acton
macton@gmail.com
http://www.linkedin.com/in/mikeacton