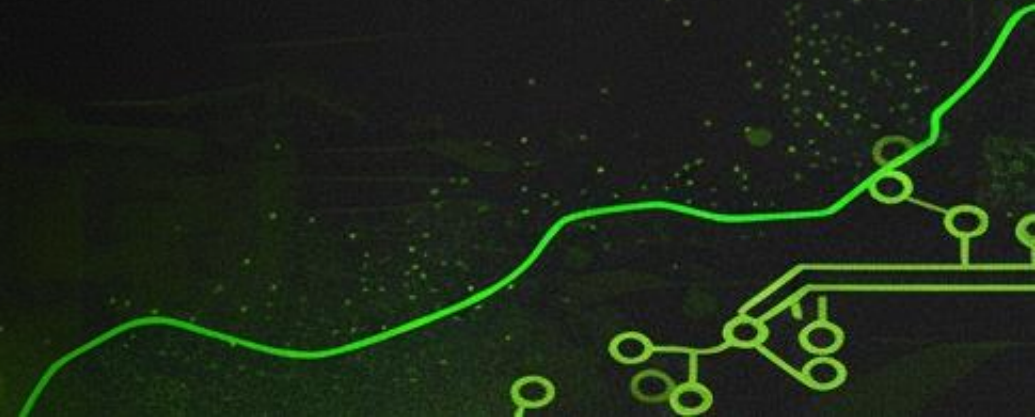




25+ Years of Pathfinding Problems with C++



About me

- Raymi Klingers
- Engineering Director at **FORGOTTEN EMPIRES**
- A long time Age of Empires (AoE) fan that got lost in codebases older than I am



What did he say?

I'll try not to but I live and breathe these games...

Legacy:

AoE1 = Age of Empires

AoE2 = Age of Empires 2

AoE3 = Age of Empires 3

AoM = Age of Mythology

Reboot:

(AoE2) HD Edition = Age of Empires 2: HD Edition

AoE2DE = Age of Empires 2: Definitive Edition

AoMEE = Age of Mythology: Extended Edition

AoM Retold = Age of Mythology: Retold

AoE3DE = Age of Empires 3: Definitive Edition



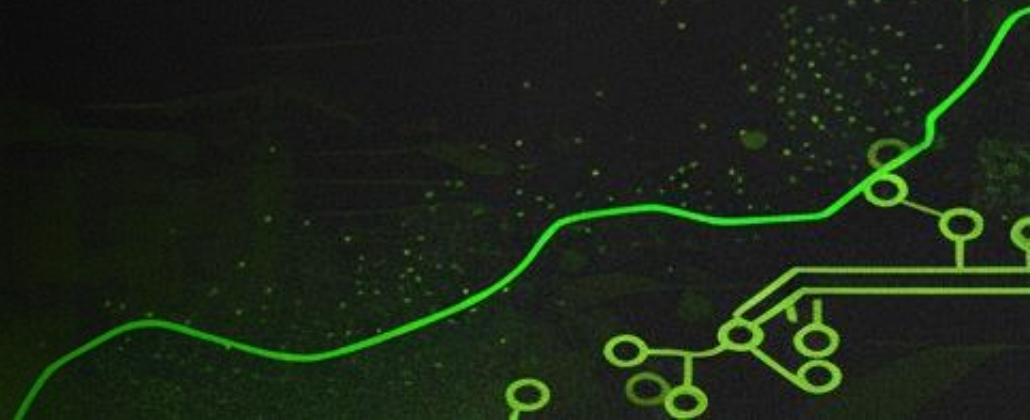
Problems

There are just so many ways it can go wrong



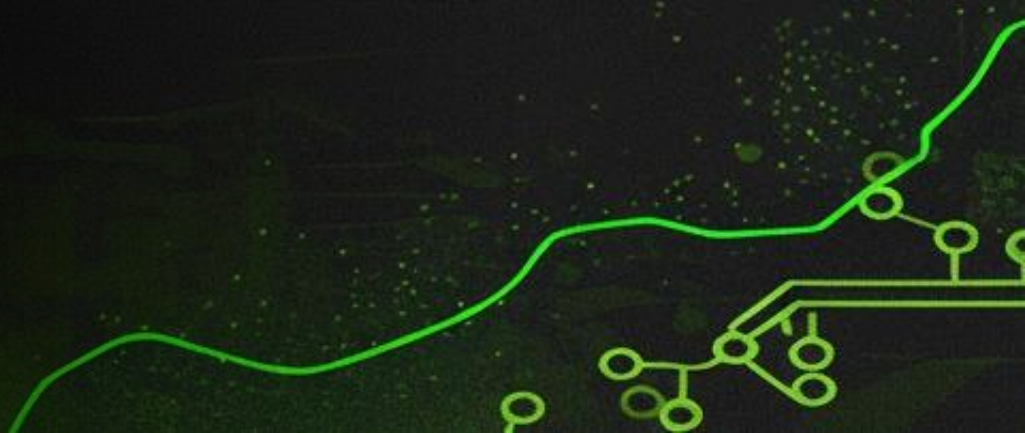


Problems: Community feedback



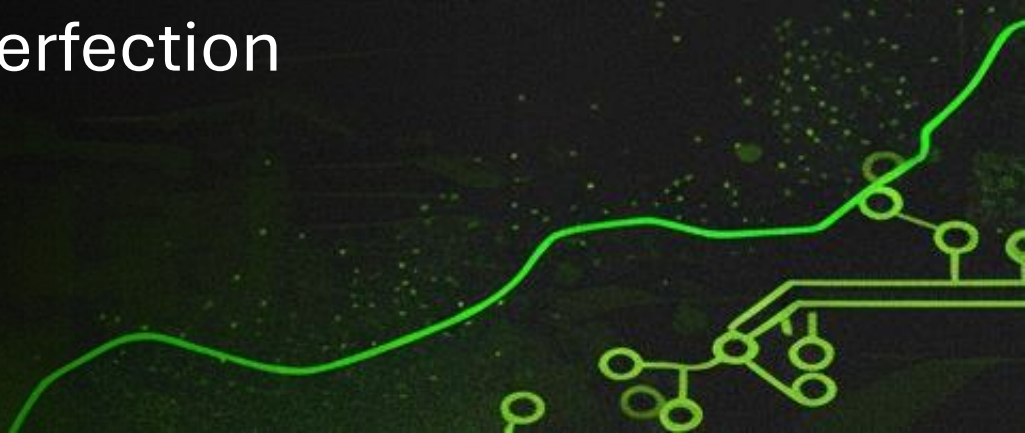


Problems: Community feedback

- Pathfinding is bad pls fix
 - Pathfinding is worse than Legacy
 - Pathfinding is worse this patch
 - Pathfinding is bad when you do X
 - Pathfinding is bad in this replay at time X
- 

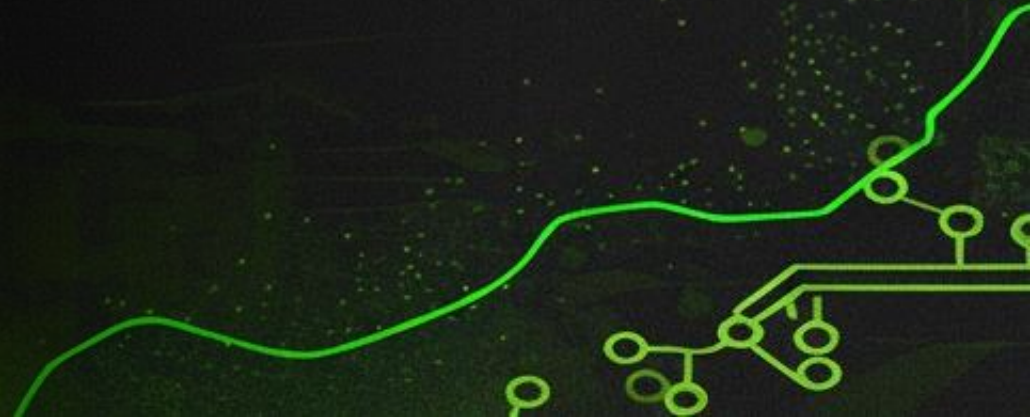


Problems: Community feedback

- Pathfinding is bad pls fix - The classic
 - Pathfinding is worse than Legacy - Regression indicator
 - Pathfinding is worse this patch - Regression indicator
 - Pathfinding is bad when you do X - Great
 - Pathfinding is bad in this replay at time X - Perfection
- 



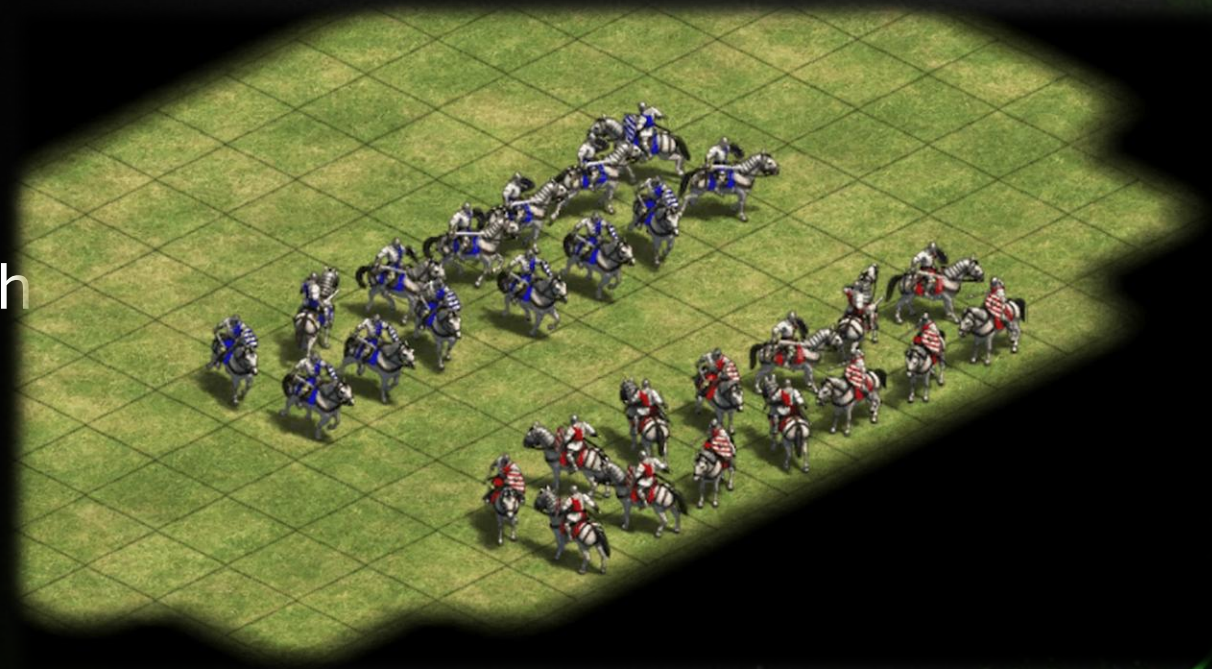
Problems: Not Taking The 'Easy' Path



Problems: Not Taking The 'Easy' Path

In Age of Empires 2:

- Units bump into others, stop and repath around. No pushing allowed
- Formations allow for some friendly overlap
- Maps are random and fully dynamic
- At the least we are somewhat grid based



Problems: Not Taking The 'Easy' Path

Starcraft 2 simplified the problem:

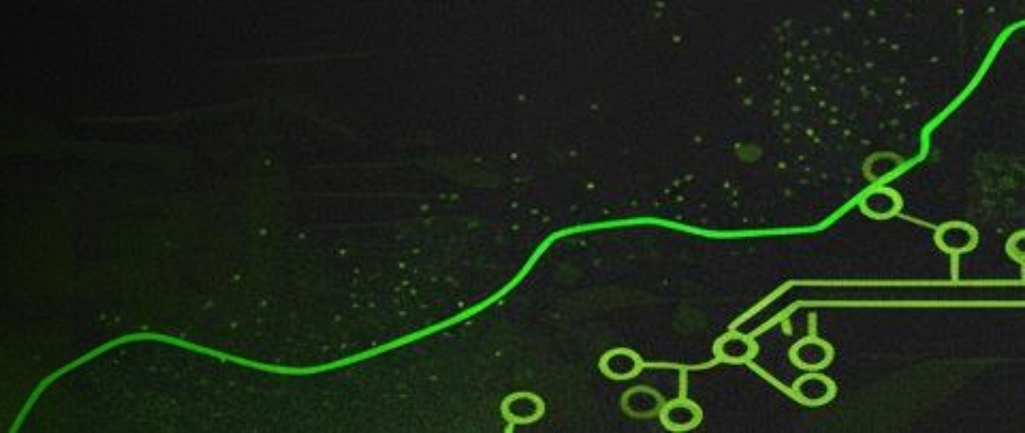
- Fixed maps
- Steering behaviours
- Allowing for small overlaps





Problems: Not Taking The 'Easy' Path

So why not?

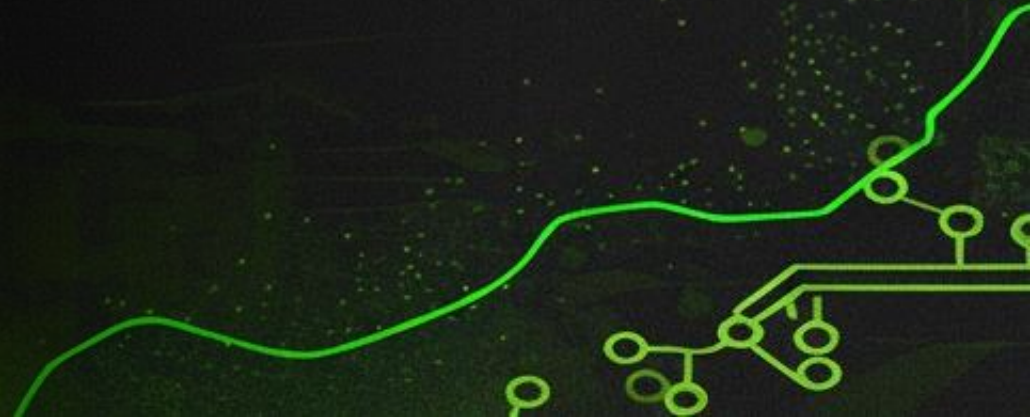
- It is not that easy to retrofit everything
 - We did try it in Age of Mythology: Retold
 - Community hates on it but loves it too
- 



We would have
missed lots of gems



Problems: Erosion of Knowledge

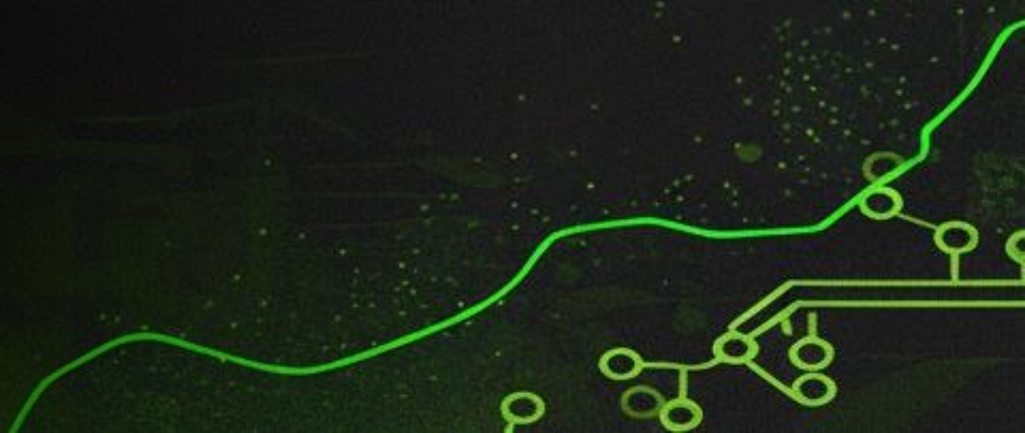




Problems: Erosion of Knowledge

- Some of the original developers still around for some time
- Another developer left some articles
- Some comments
- Some logging

For legacy code standards this is great

- [implementing-coordinated-movement](#)
 - [coordinated-unit-movement](#)
- 

Problems: Erosion of Knowledge

- Pathfinder was touched by multiple studios since launch



Problems: Erosion of Knowledge

- Source history was lost at least once



Problems: Erosion of Knowledge

- Community reverse engineered patch also lost



Launch



HD Edition



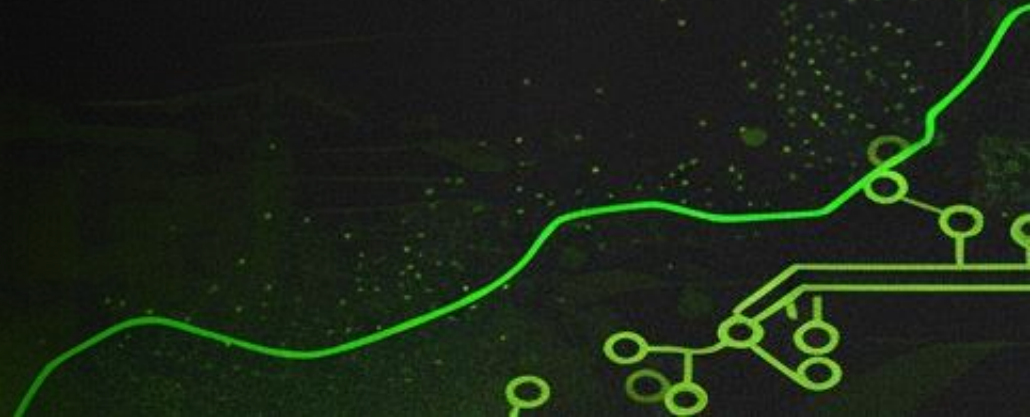
HD Edition



Definitive
Edition -
Current

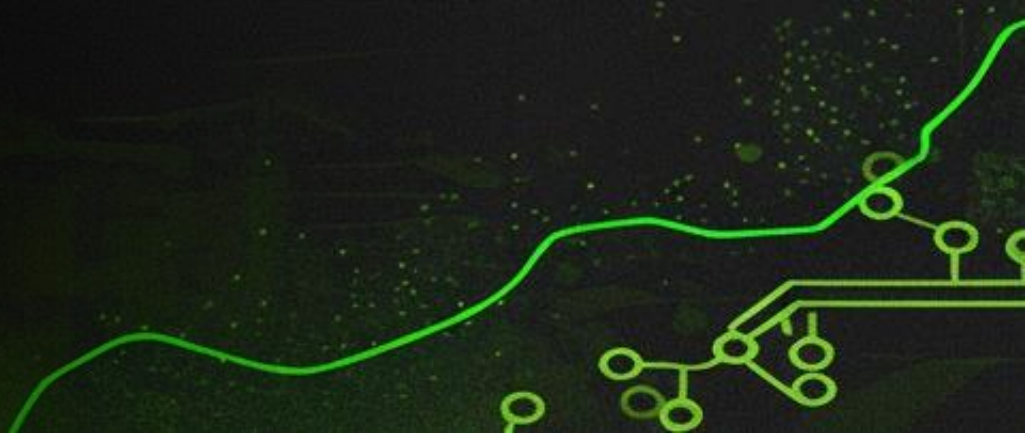


Problems: Regressions



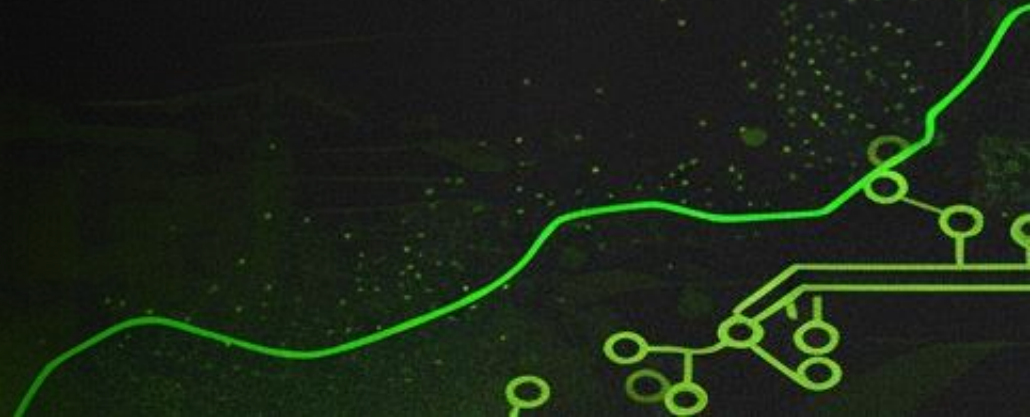


Problems: Regressions

- Fix a bug only to introduce a couple more
 - Implemented a new feature and forgot to do X,Y,Z
- 



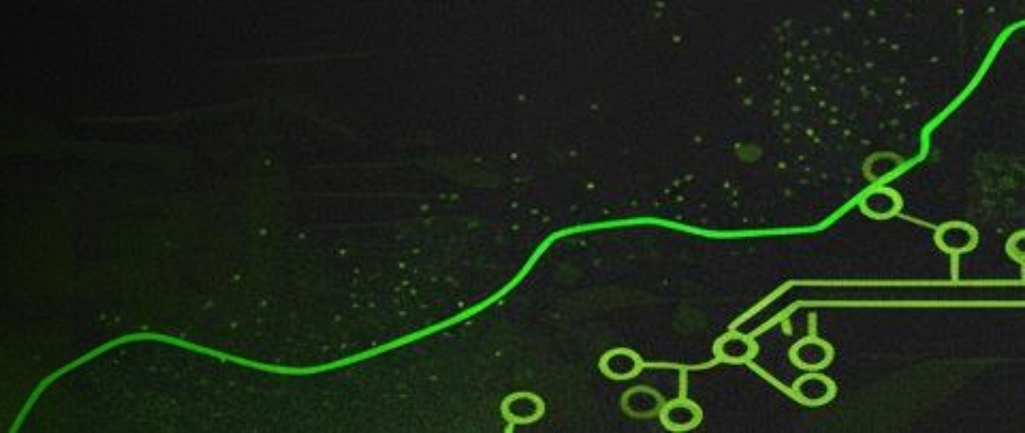
Problems: How does it even work?





Problems: How does it even work?

The Short Range Pathfinding Algorithm is:

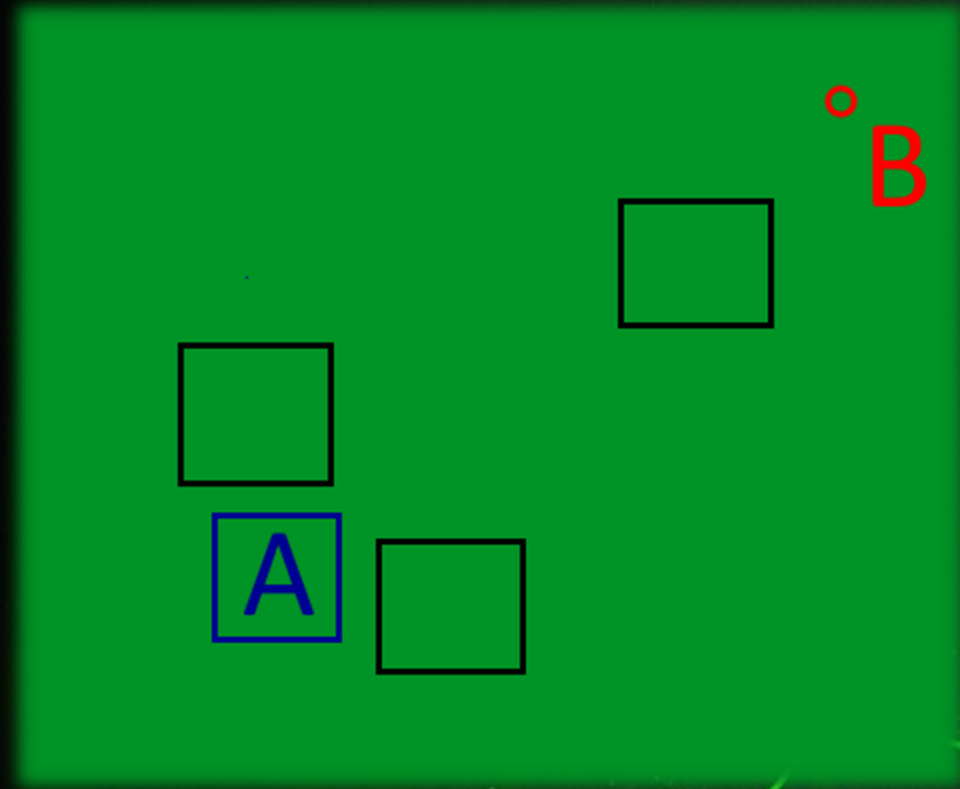
- Undocumented
 - Not performant enough
 - Functioned most of the time
- 

Problems: How does it even work?

To path with unit A to point B:

1. Find obstructions in area between A and B

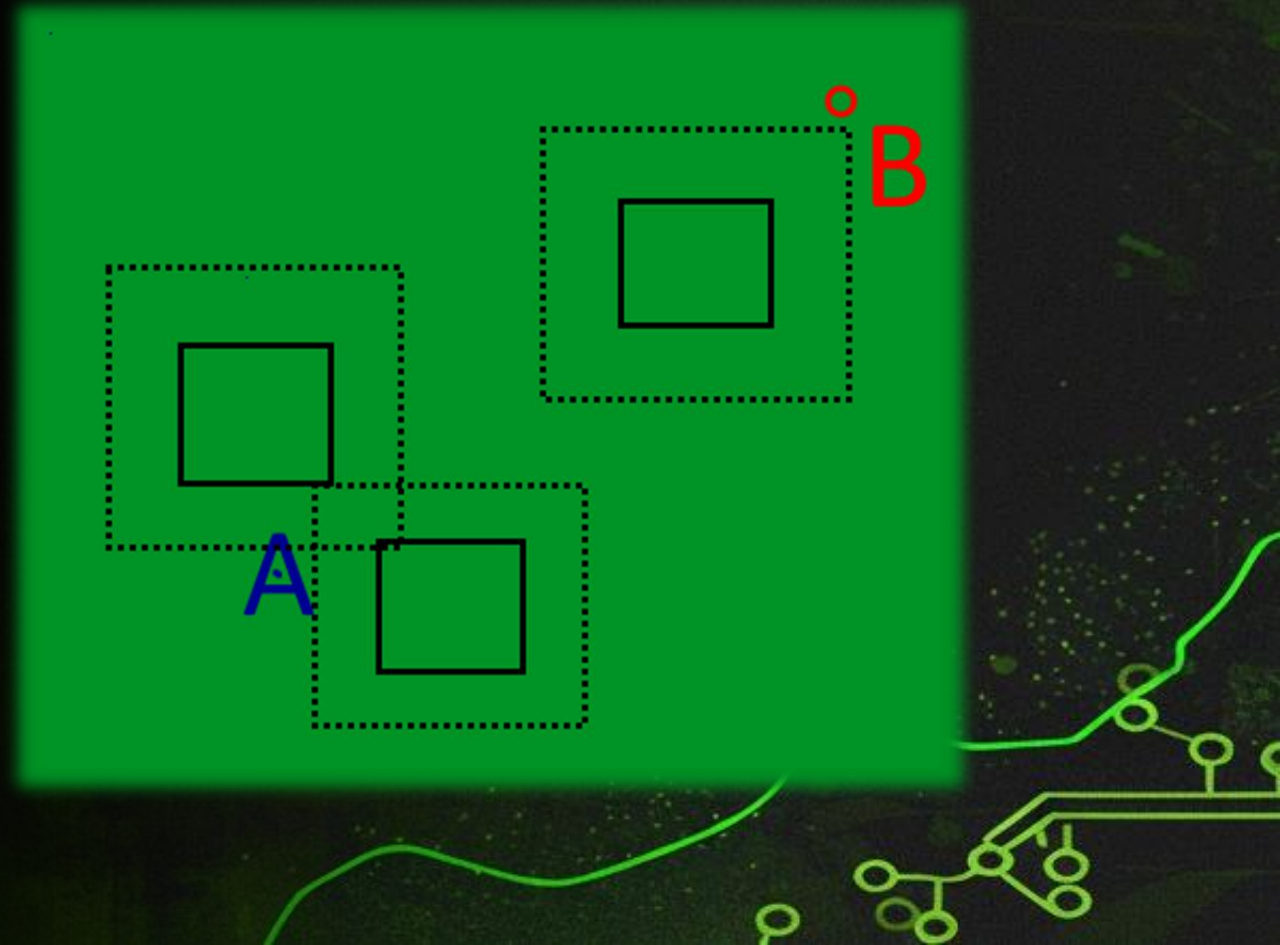
Units and buildings are axis aligned rectangles in AoE



Problems: How does it even work?

To path with unit A to point B:

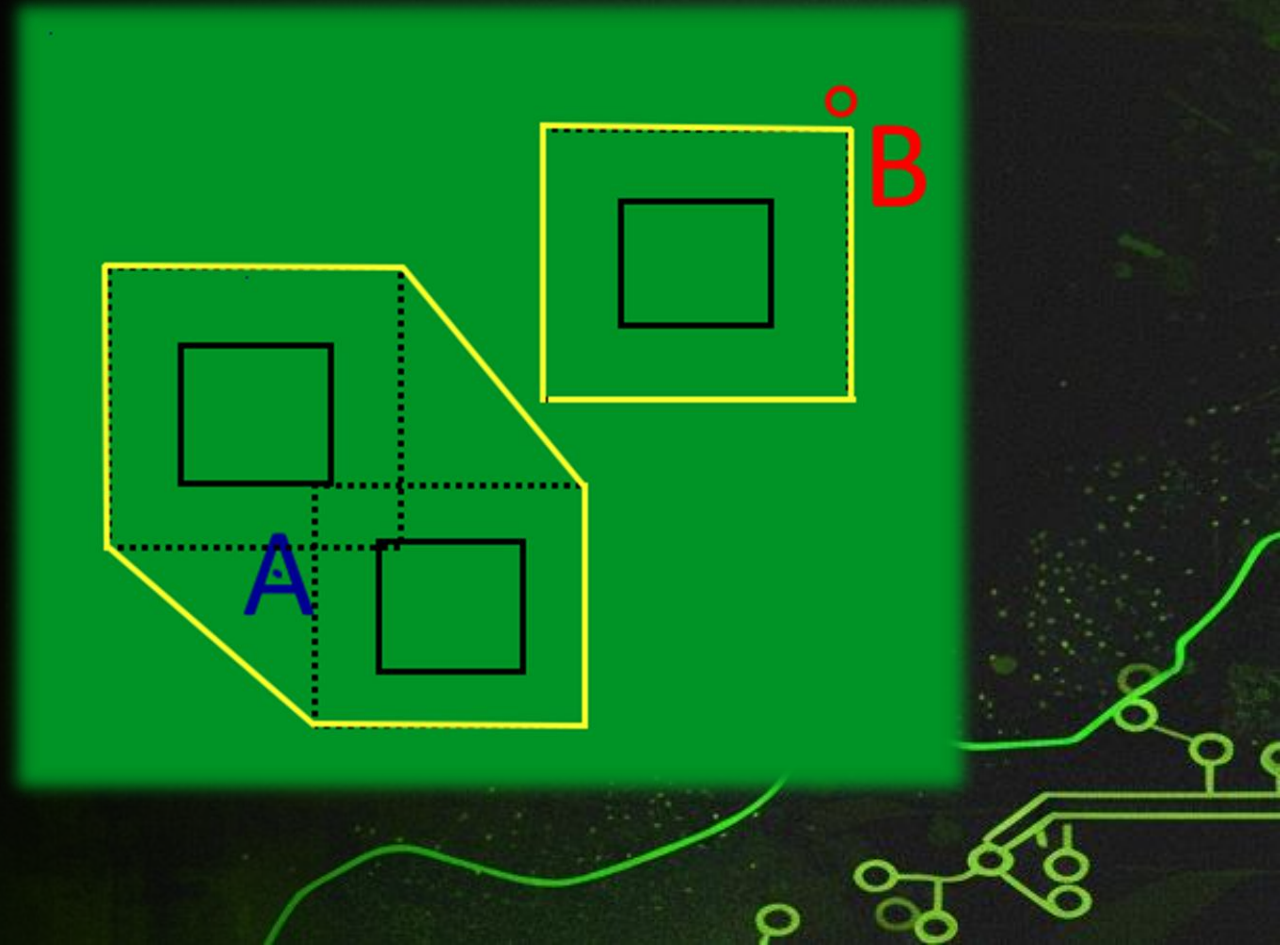
2. Shrink A to a point and expand all other obstructions



Problems: How does it even work?

To path with unit A to point B:

3. Create convex hulls using gift wrapping algorithm

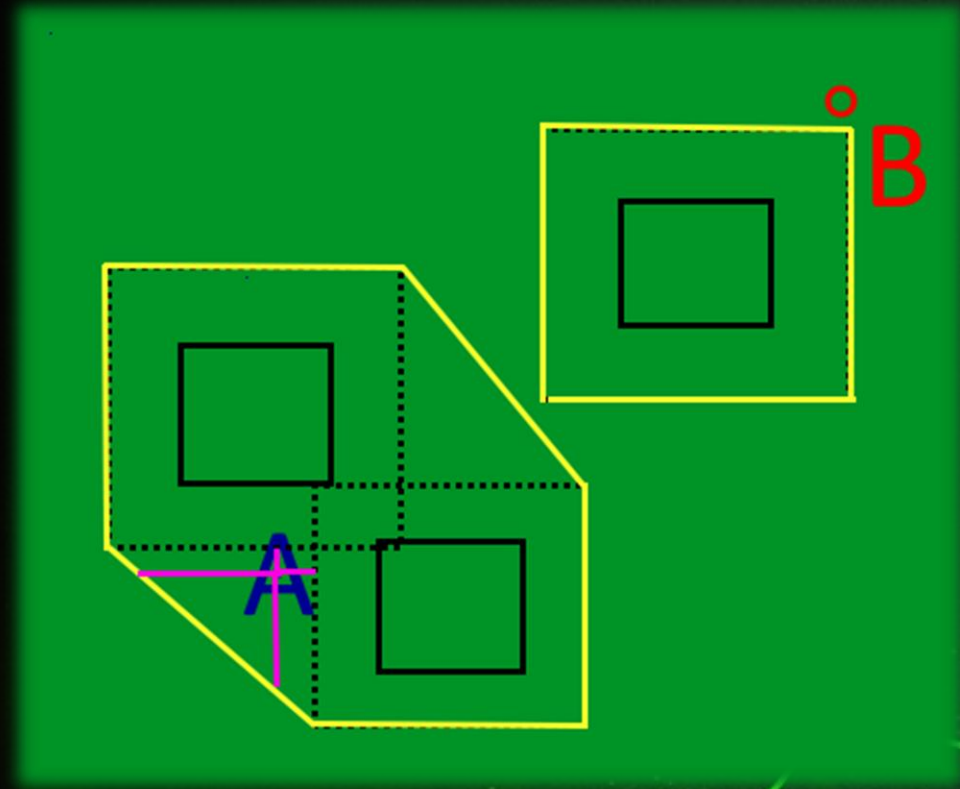
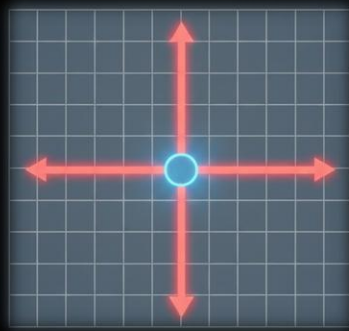


Problems: How does it even work?

To path with unit A to point B:

4. Are we in the same hull as point B?

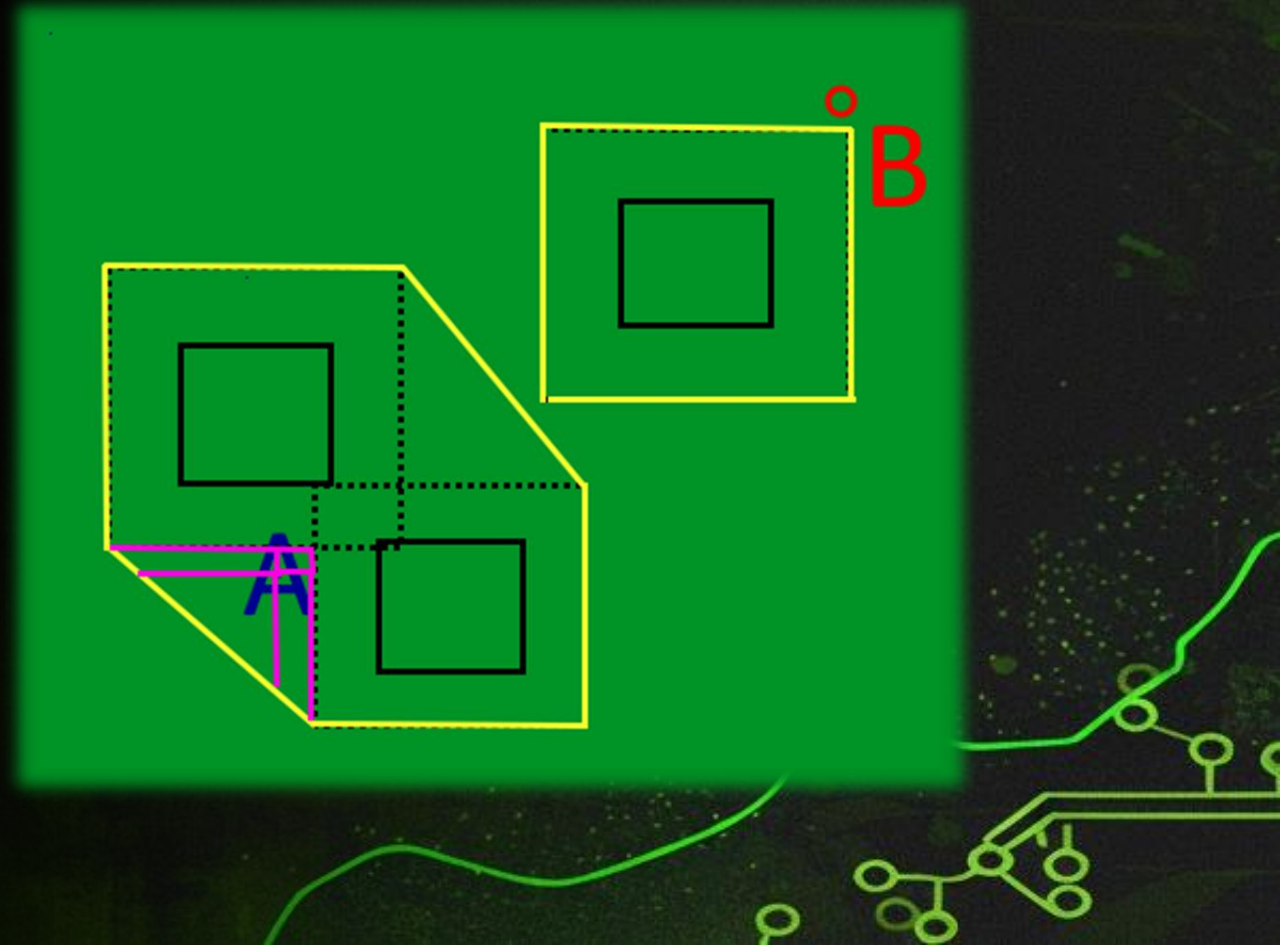
No? Start shooting cardinal rays to escape



Problems: How does it even work?

To path with unit A to
point B:

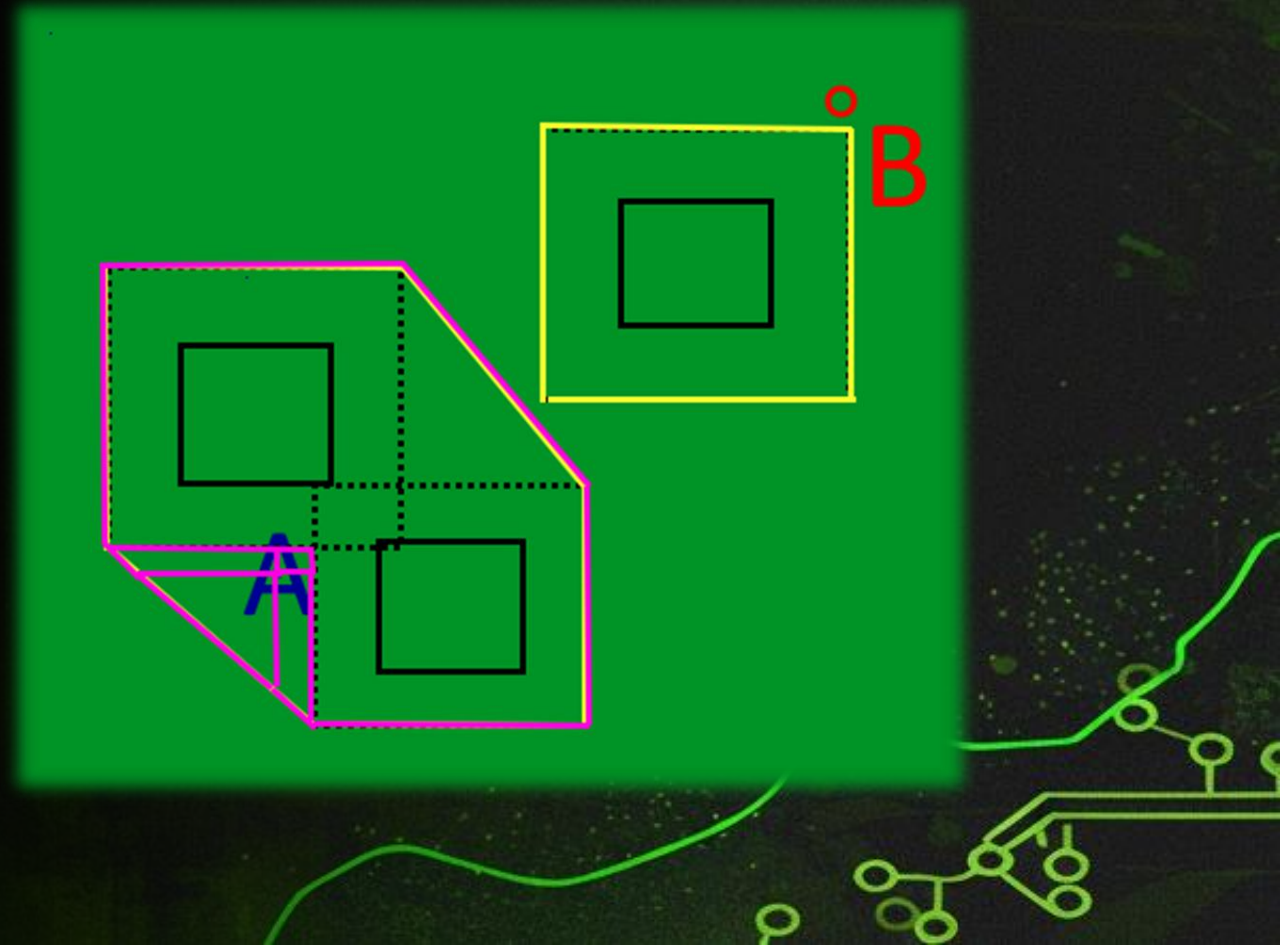
4.a if we hit an
obstruction shoot
sideways



Problems: How does it even work?

To path with unit A to point B:

4.b if we hit a hull we traverse it



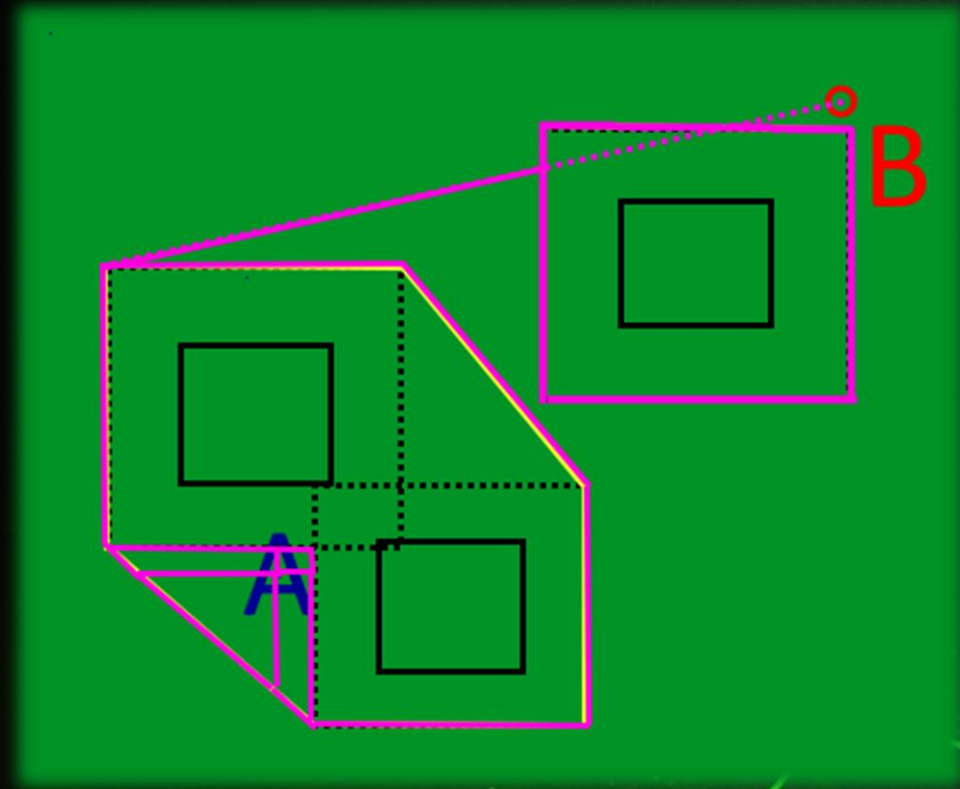
Problems: How does it even work?

5. At every vertex we try and shoot a ray towards our goal

Problems: How does it even work?

To path with unit A to
point B:

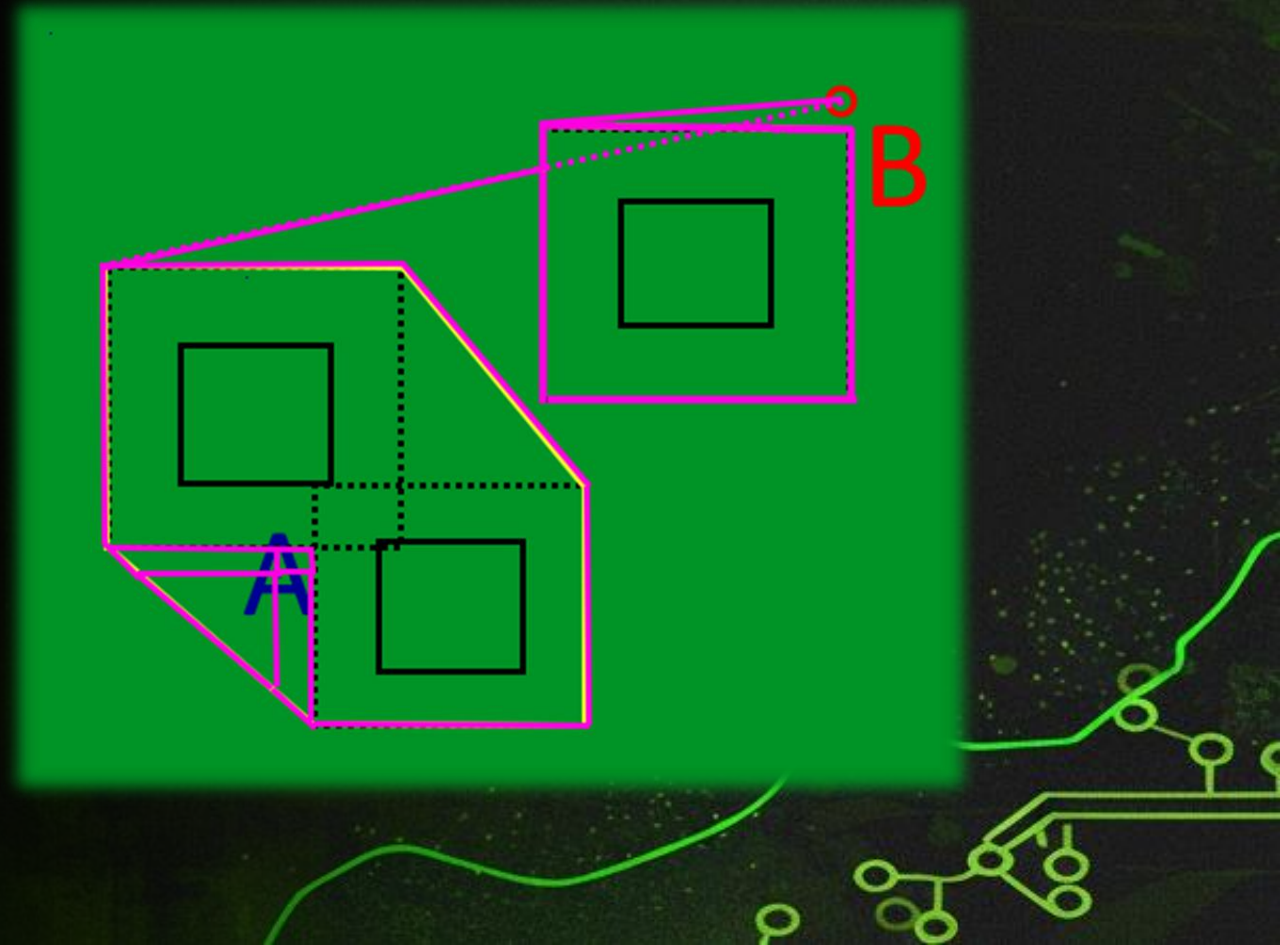
6. Repeat from step 4.b
till the goal is found



Problems: How does it even work?

To path with unit A to point B:

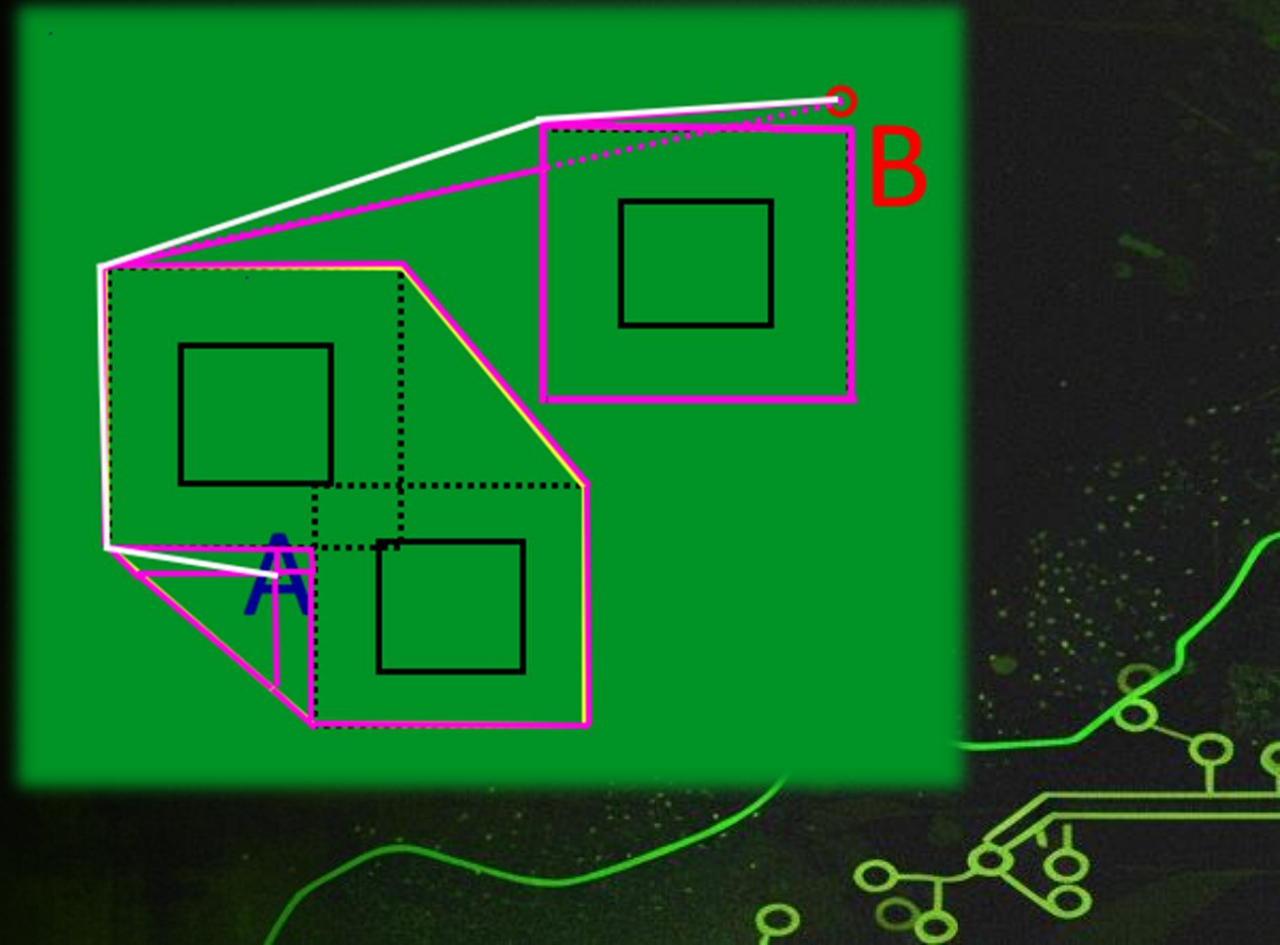
6. Repeat from step 4.b till the goal is found



Problems: How does it even work?

To path with unit A to point B:

7. Brute-force path
smooth for final path

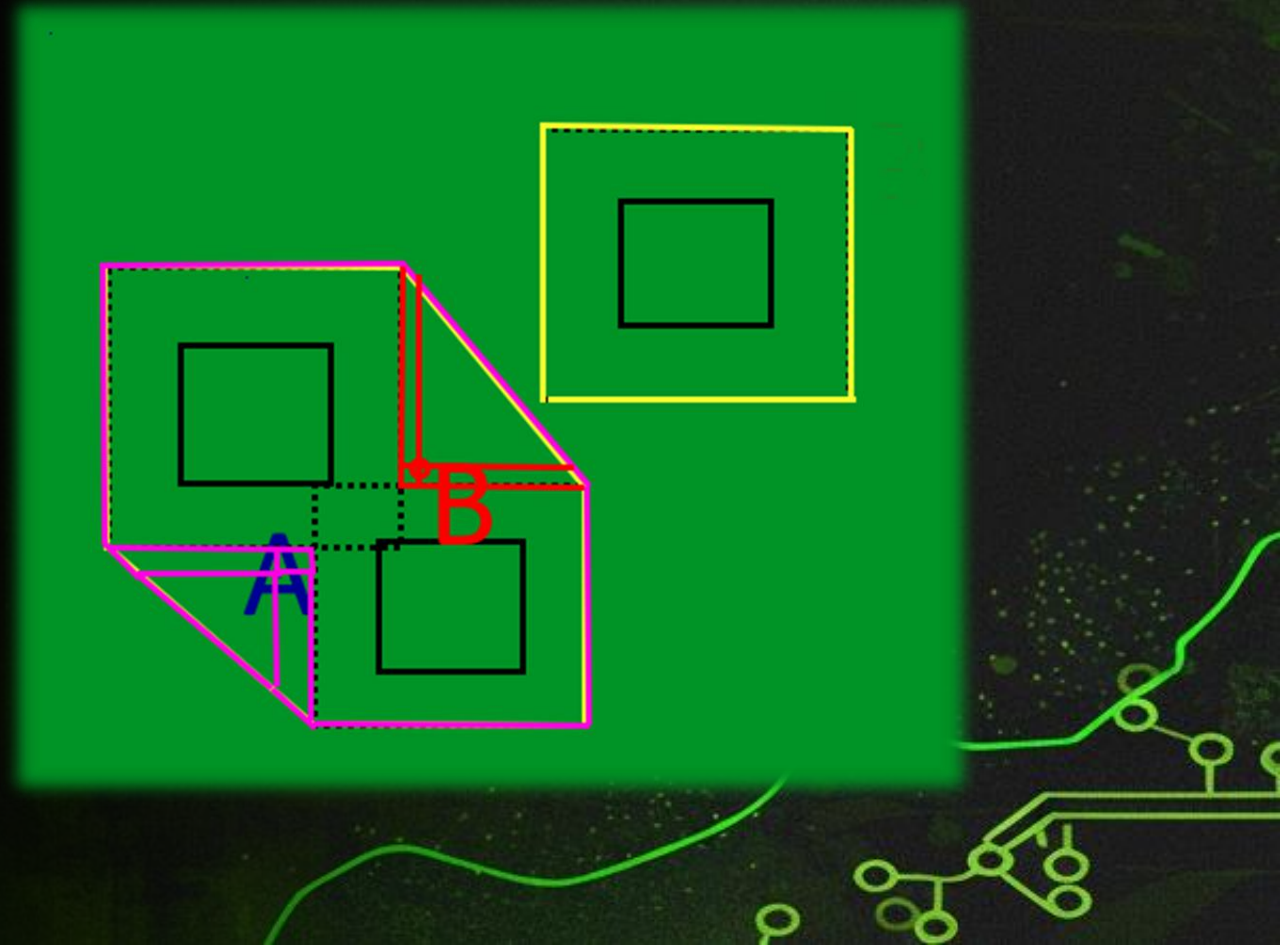


Problems: How does it even work?

To path with unit A to point B:

4. Are we in the same hull as point B? Yes.

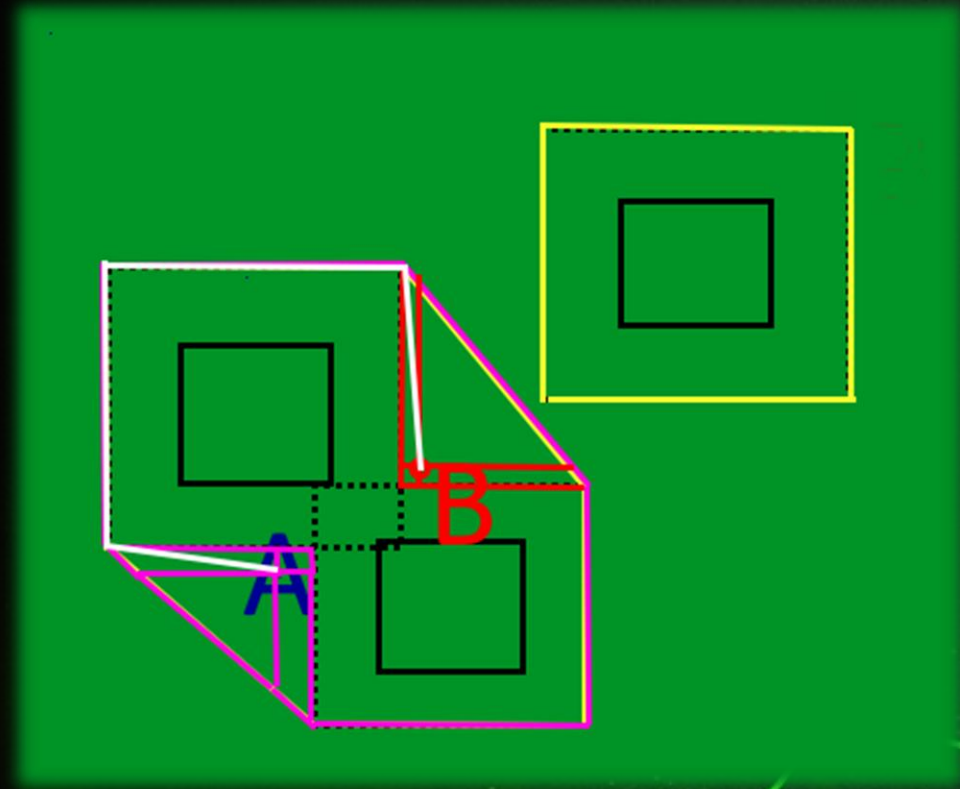
Also start shooting cardinal rays from B



Problems: How does it even work?

To path with unit A to
point B:

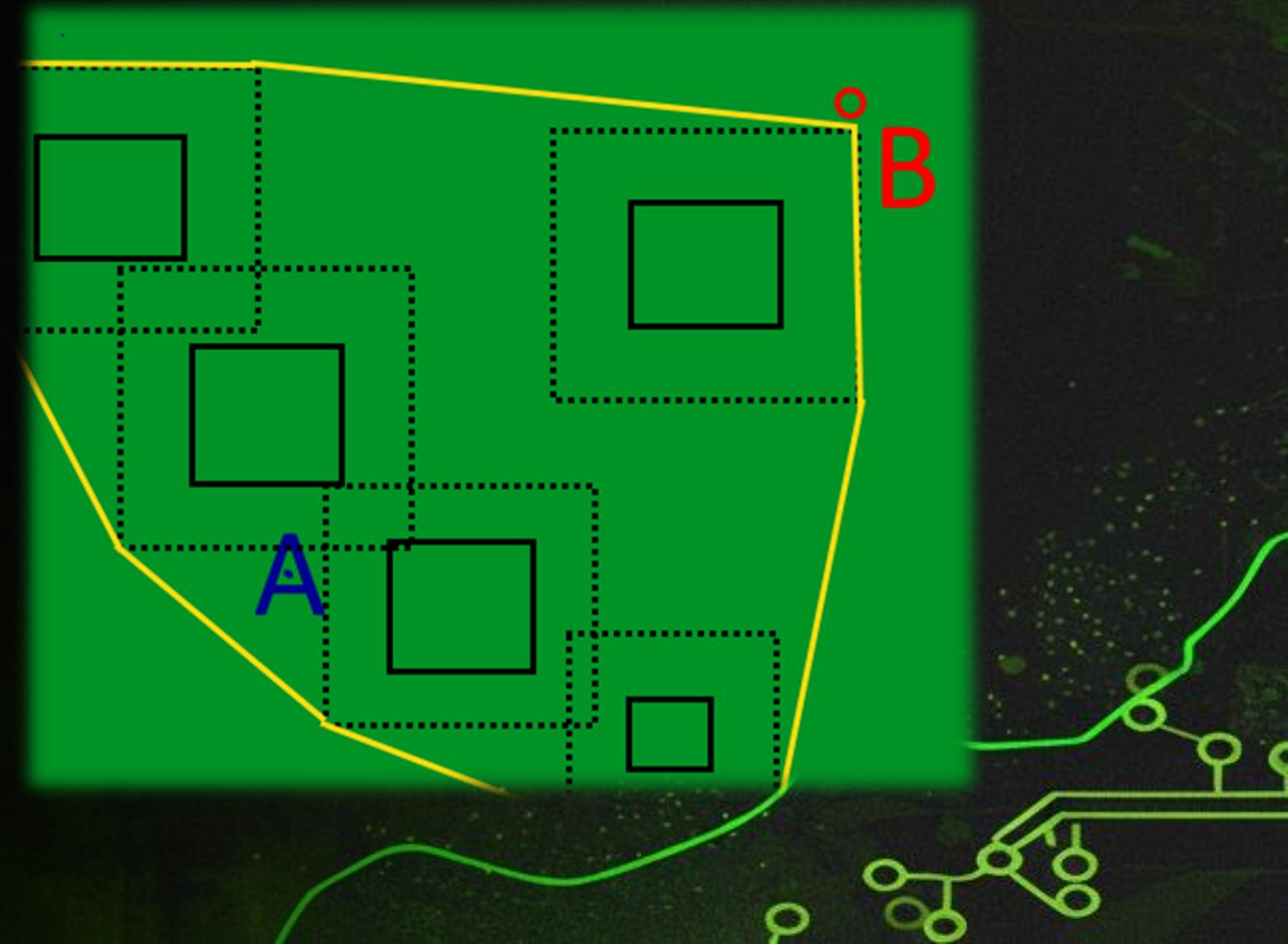
7. Again smooth to get
to our final path



Problems: How does it even work?

To path with unit A to point B:

3. What if our convex hulls escape our search area?

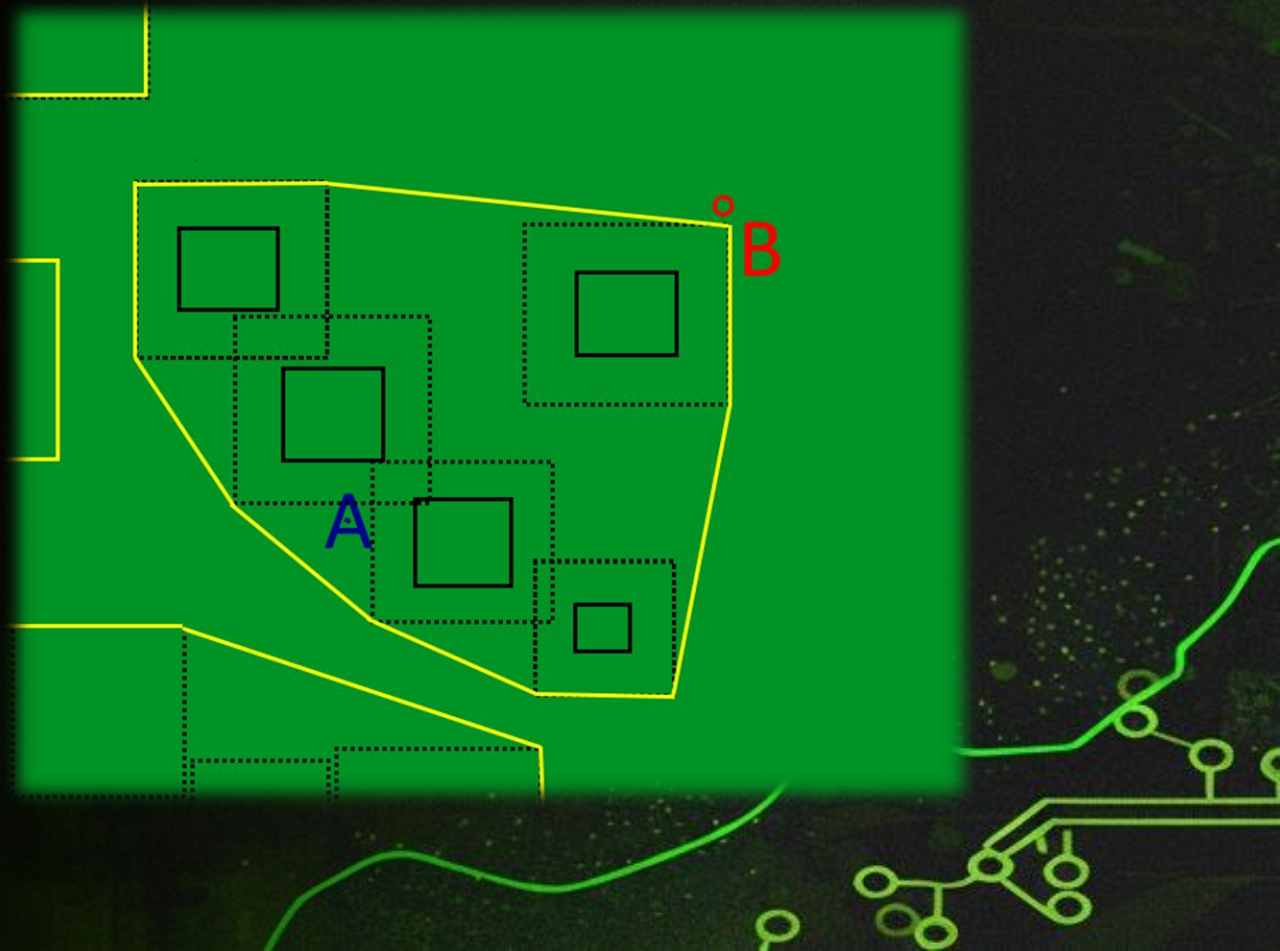


Problems: How does it even work?

To path with unit A to point B:

3. What if our convex hulls escape our search area?

Expand search area once. Otherwise give the closest point path

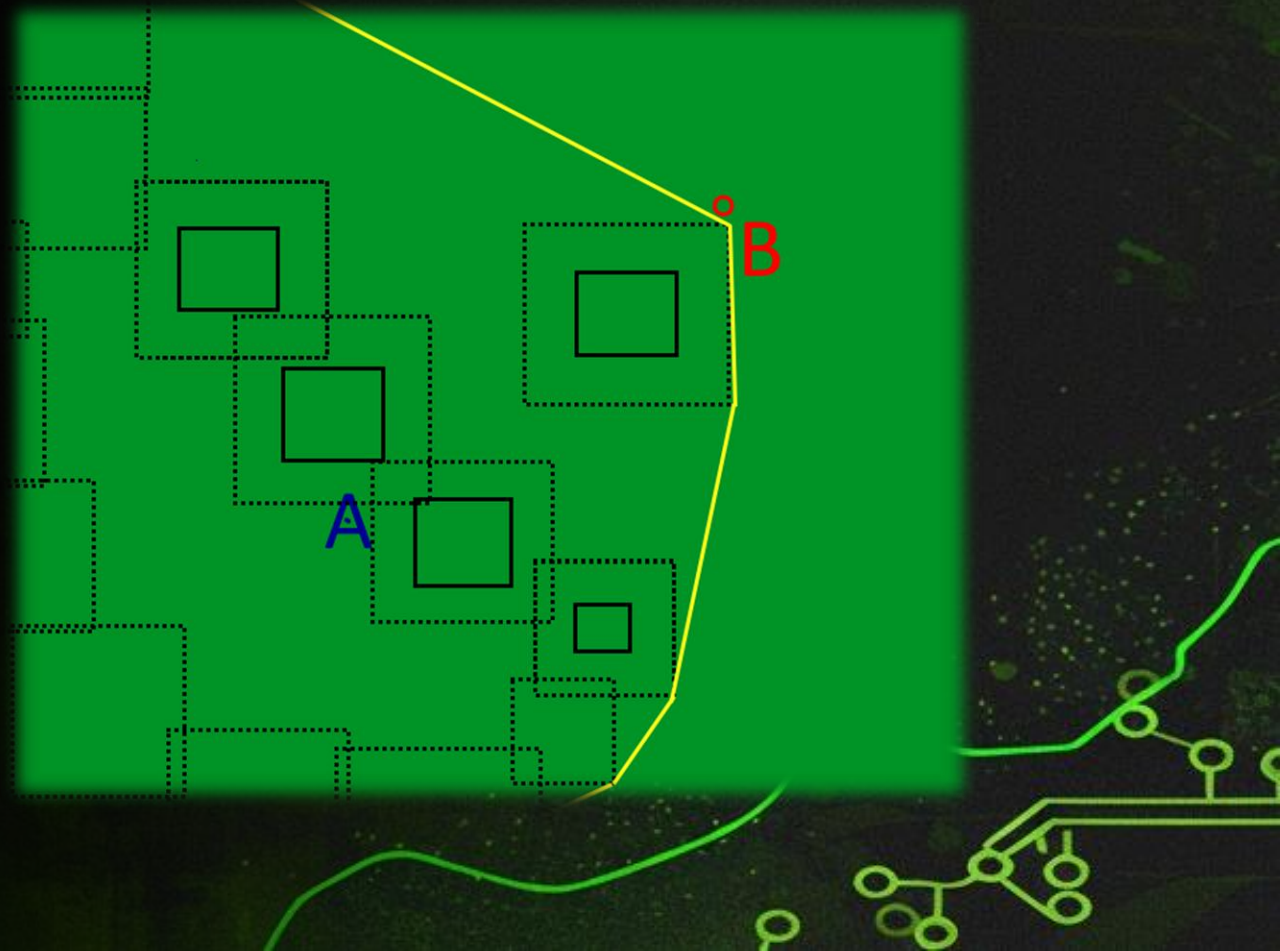


Problems: How does it even work?

To path with unit A to
point B:

3. What if we are
surrounded?

Do all the work, expand
once...

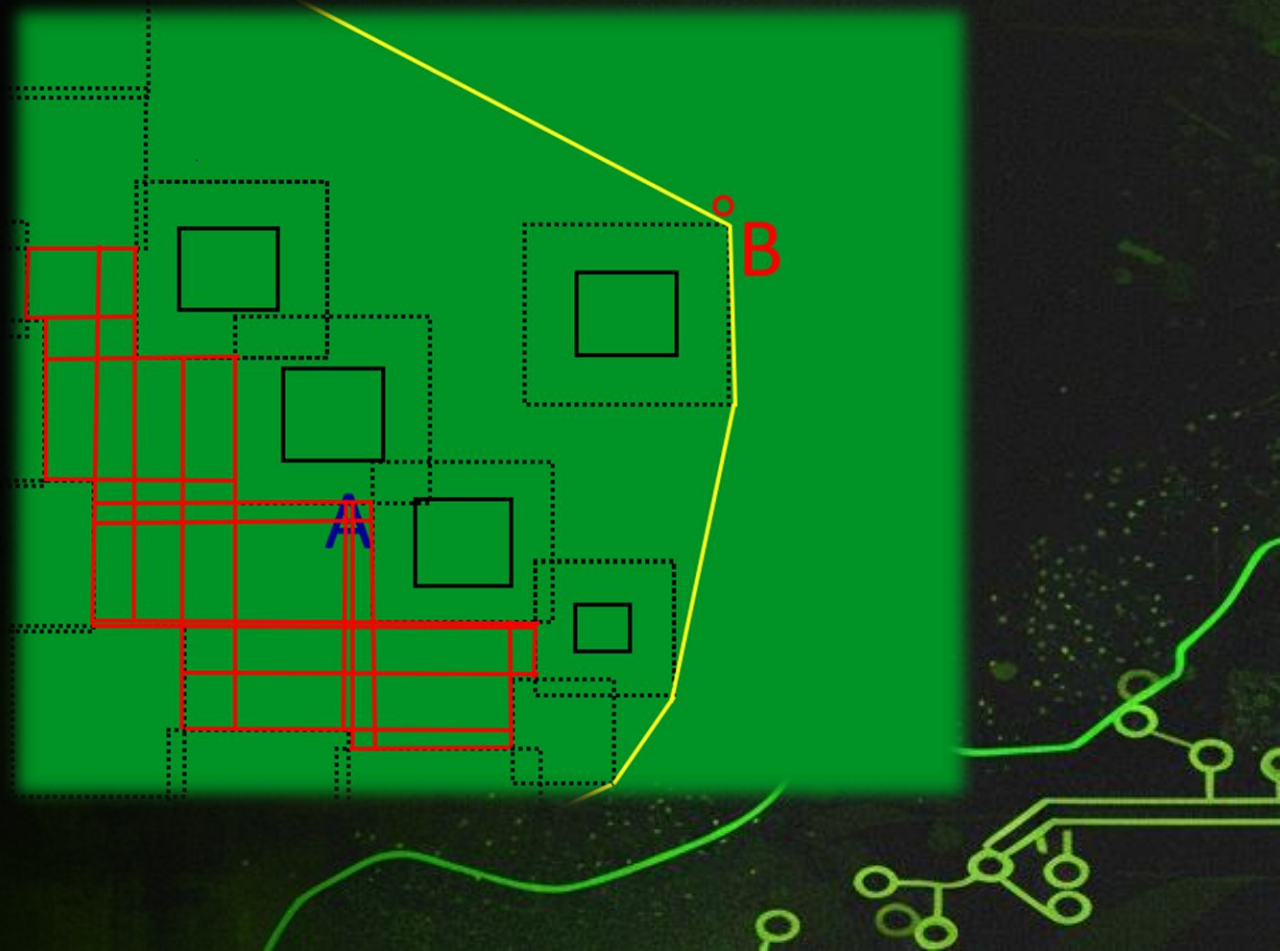


Problems: How does it even work?

To path with unit A to point B:

4. Are we in the same hull as point B? No. Start shooting cardinal rays to escape.

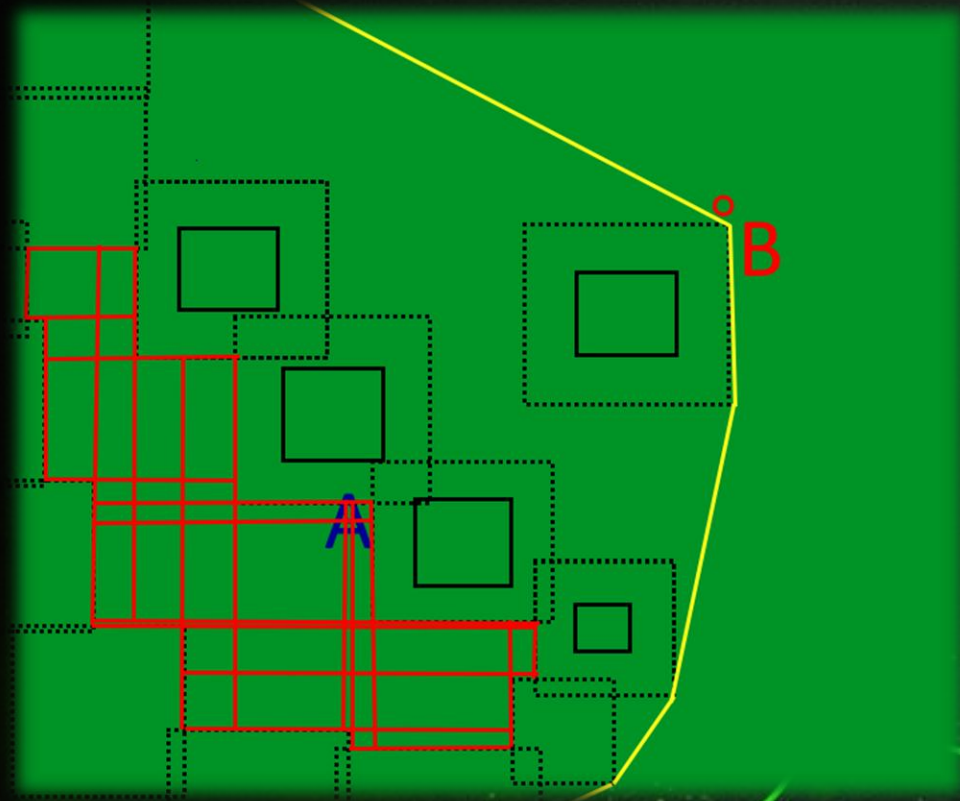
Up to 64 iterations or if we run out of unique vertices to try



Problems: How does it even work?

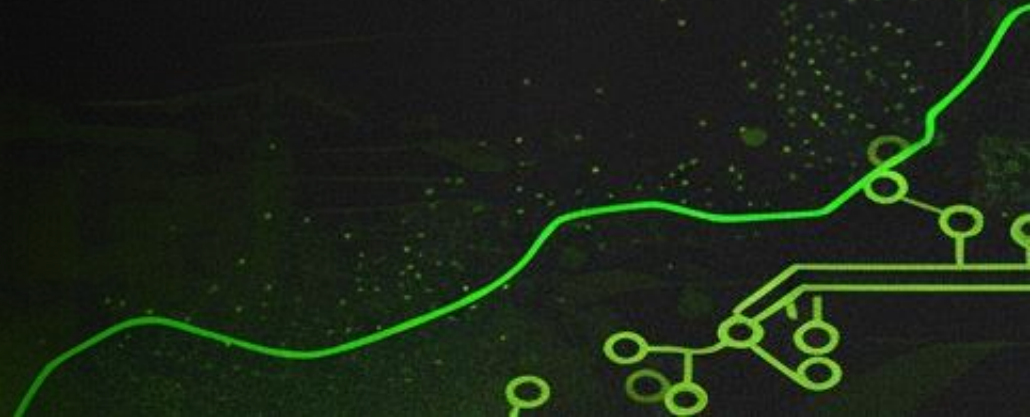
Computers are fast...

This was good enough
for short range paths
and hierarchical
pathfinding saved the
day back in 1999





Problems: Precision Errors



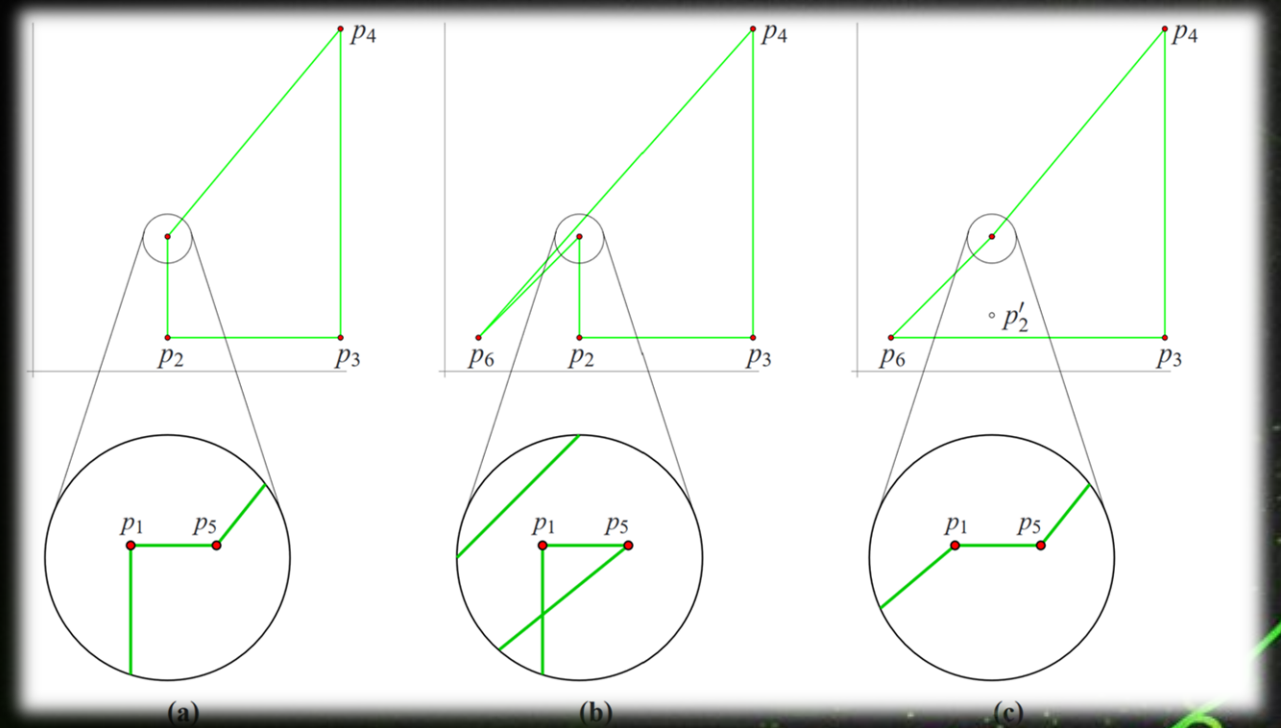
Problems: Precision Errors



Problems: Precision Errors

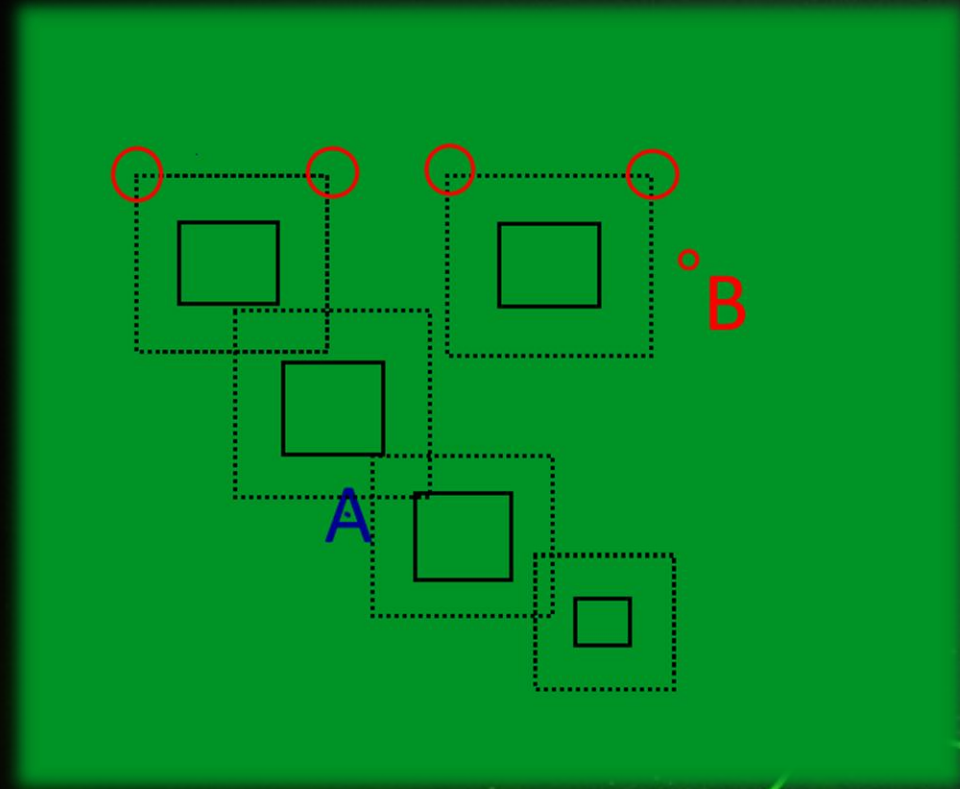
A classroom example of geometrical precision errors.

Age of empires 2, Age of mythology and Age of empires 3 and all remasters ran into this problem.



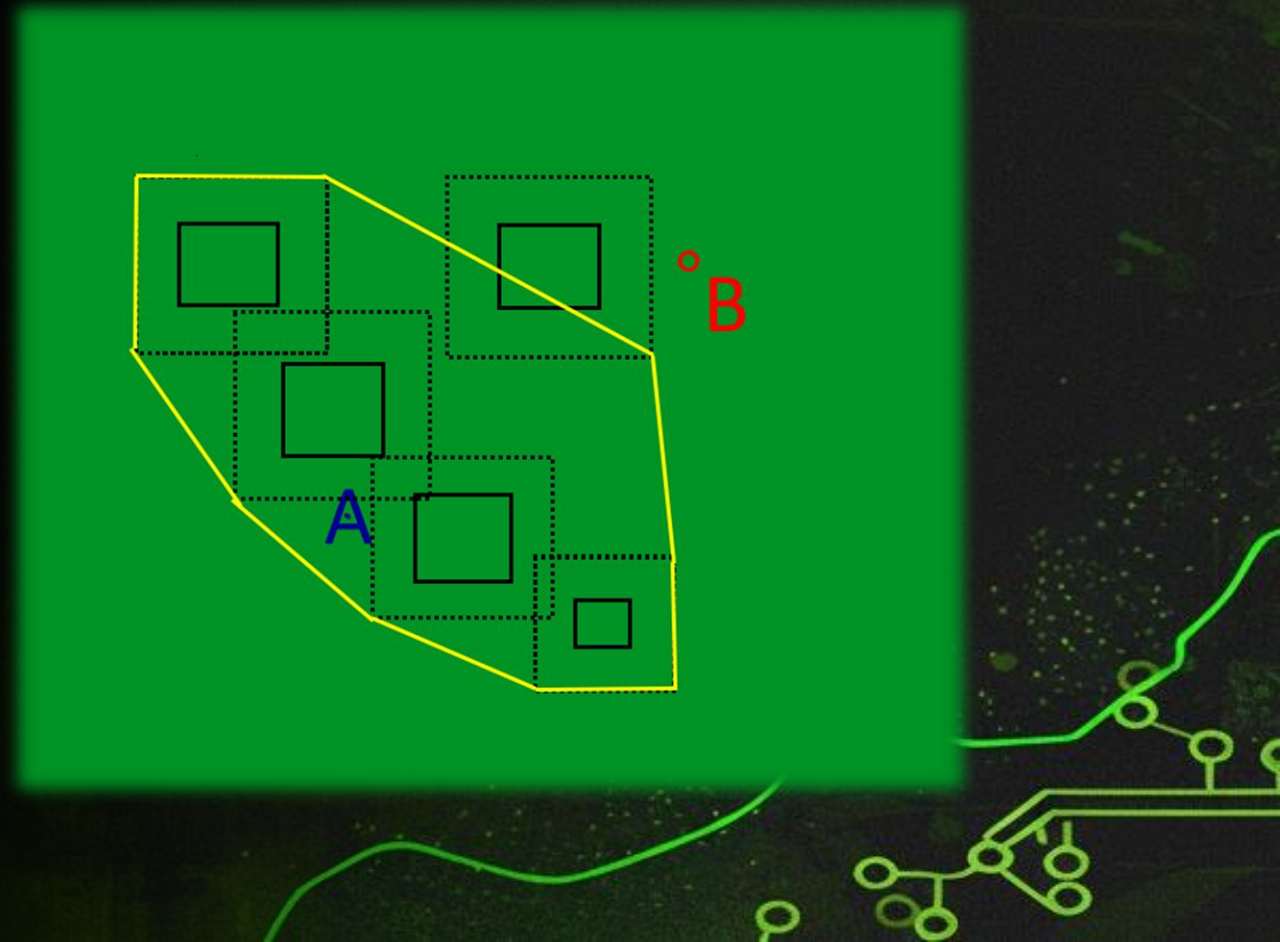
Problems: Precision Errors

These 4 points are nearly colinear. With floats there is not enough precision to say if it is a left or right turn



Problems: Precision Errors

The result is a hull like this and if we walk it we clip through obstructions



Problems: Precision Errors

Attempted fix on AoE 2
which made the bug
harder to reproduce
Filters out close points

```
for (i = NumPoints; ppoint <= pendpoint; ppoint++)
{
    for (x = 0; x < NumPoints; x++)
    {
        if (_fabs(ppoint->x - mWorkList[x].x) < FloatCompareEpsilon)
        {
            if (_fabs(ppoint->y - mWorkList[x].y) < FloatCompareEpsilon)
            {
                break;
            }
        }
    }
    if (x == NumPoints)
    {
        mWorkList[i] = *ppoint;
        i++;
    }
}
```

Problems: Precision Errors

Attempted fix on
AoE3:DE which also
made the bug harder to
reproduce

Increase precision
using doubles

```
static double calcDeterminant(const BNCHVector &v1, const BNCHVector &v2, const BNCHVector &v3)
{
    // Compute determinant of:
    // v1.x v1.z 1
    // v2.x v2.z 1
    // v3.x v3.z 1

    // Note: cast to double for the calculation to ensure accuracy
    //       using a float can result in the wrong sign for determinants that are close to 0
    double v1x = v1.x;
    double v1z = v1.z;
    double v2x = v2.x;
    double v2z = v2.z;
    double v3x = v3.x;
    double v3z = v3.z;

    return (v1x * (v2z - v3z) - v1z * (v2x - v3x) + (v2x * v3z - v2z * v3x));
}
```

Problems: Precision Errors

Might not be just Age of Empires



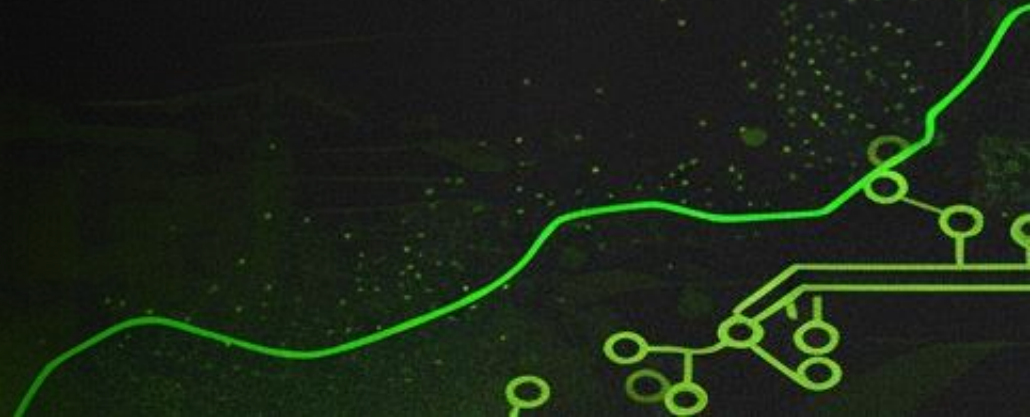
Problems: Precision Errors

Might not be just Age of Empires



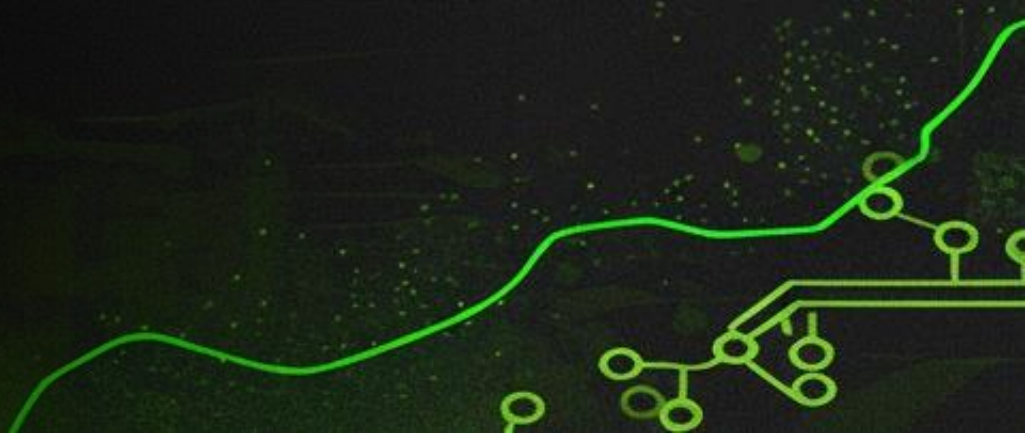


Problems: A Dead End?



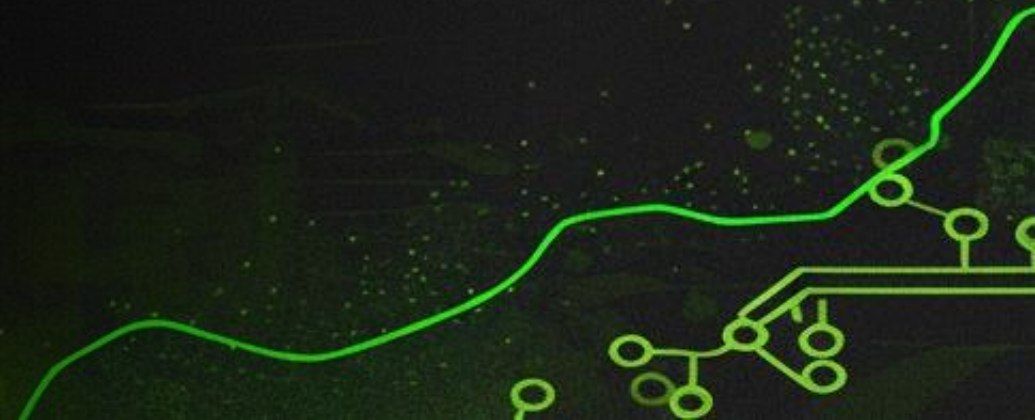


Problems: A Dead End?

- Original developers ran into the issue
 - Other developers tried to fix it
 - Other games might have the same problems
 - Very little information out there on real-time hull generation pathfinding
- 



Problems: Compiler Optimizations

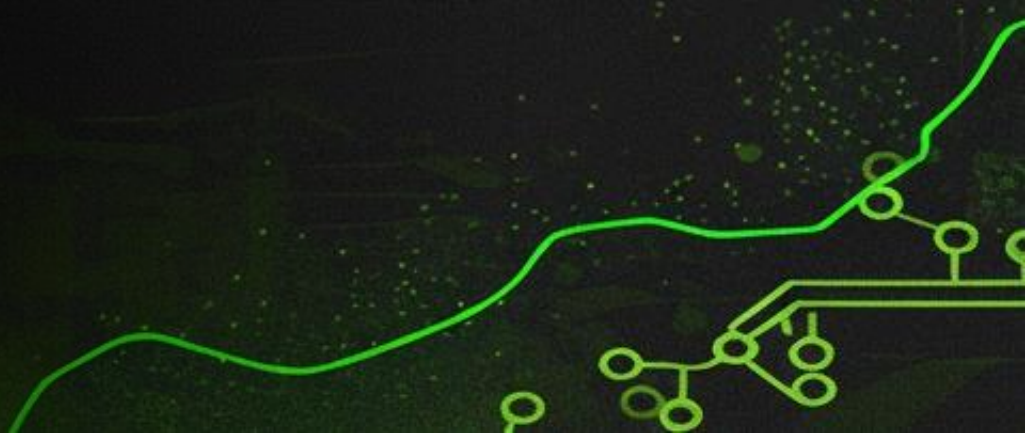




Problems: Compiler Optimizations

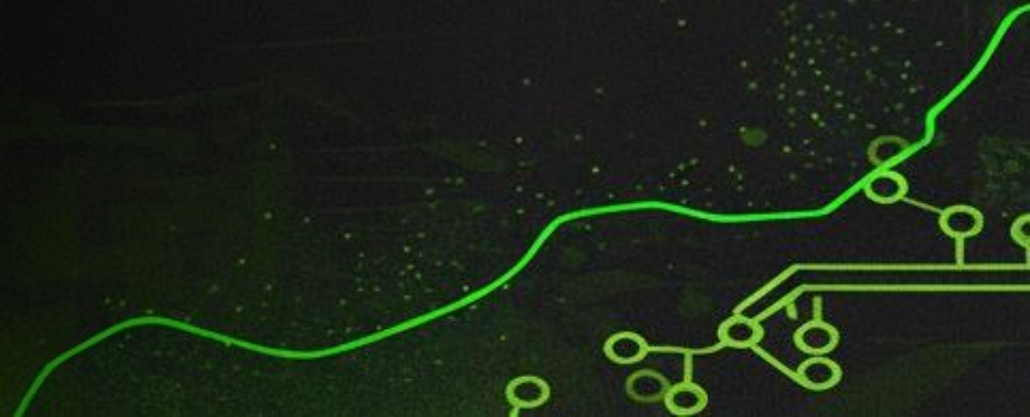
Turning off optimizations on *Age of Mythology: Extended Edition* to investigate the issue 'fixed' the issue
Specifically compiling without SIMD instructions (IA32)

But why?





Reason: 80bit Floating Point Precision

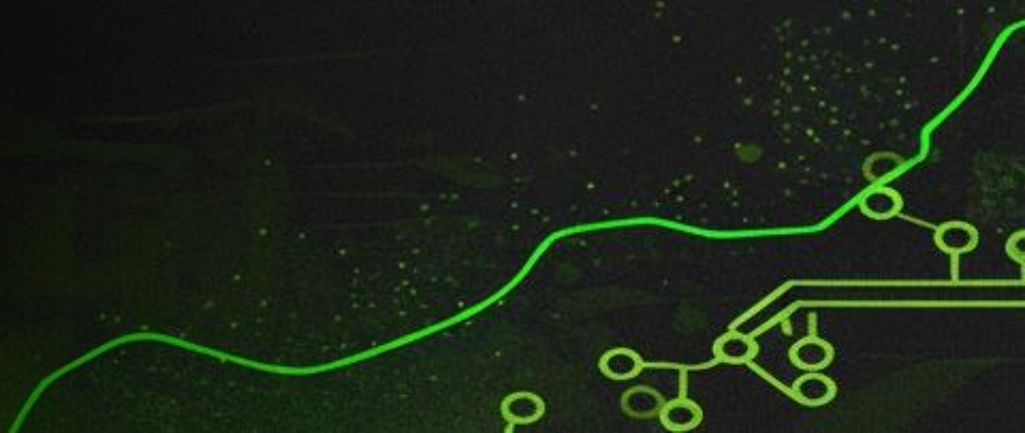




Reason: 80bit Floating Point Precision

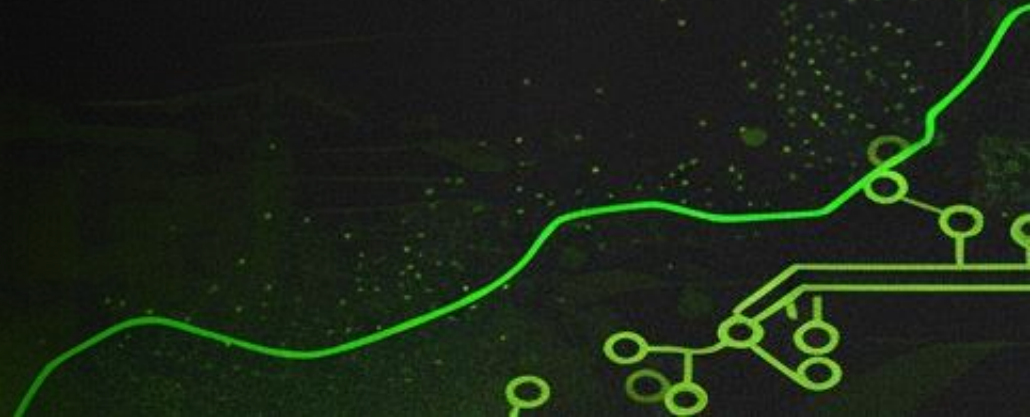
Thanks Wikipedia for being the resource I could find on this back then:

https://en.wikipedia.org/wiki/Extended_precision





Problems: The SIMD Wrecking Ball





Problems: The SIMD Wrecking Ball

Decision was made to drop the extended precision for good reasons:

- Extended floating point precision was hard to use
- Performance

But how many applications ran into issues because of this?



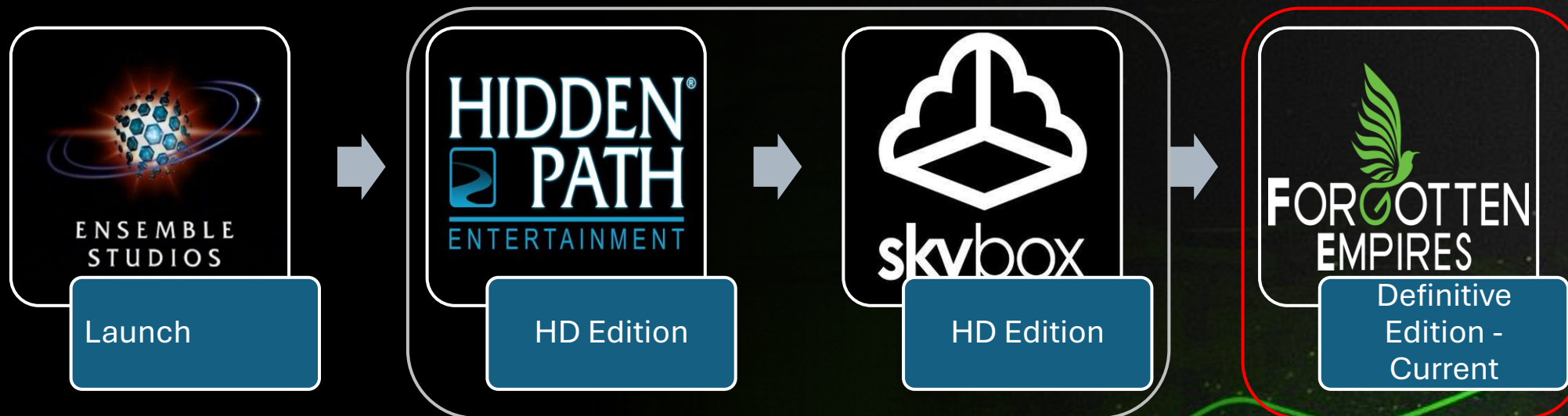
Problems: The Perfect Crime

- Sometime here SIMD got enabled



Problems: The Perfect Crime

- SIMD got disabled due to determinism concerns
- Game ported to 64bit

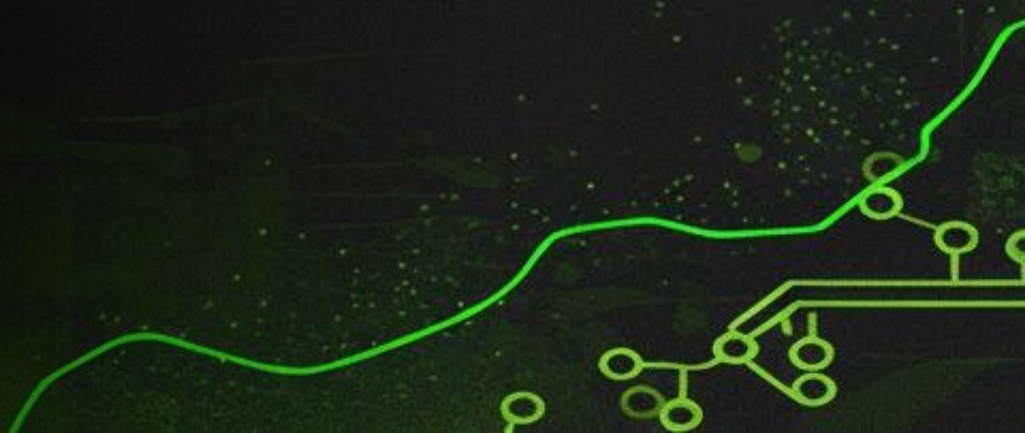




Problems: Regressions Part II

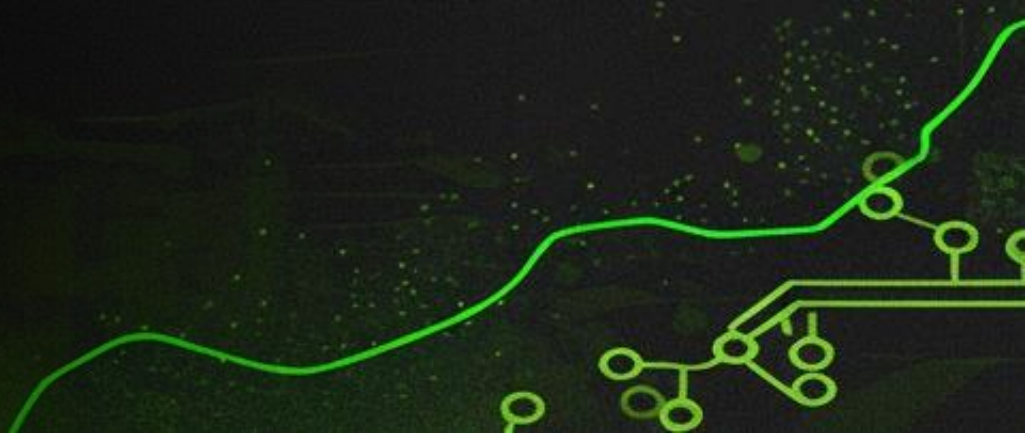
- Fix a bug only to introduce a couple more
- Implemented a new feature and forgot to do X,Y,Z

Adding to the list:

- Enabling SIMD instructions
 - Porting to 64bit
- 



Problems: Community feedback

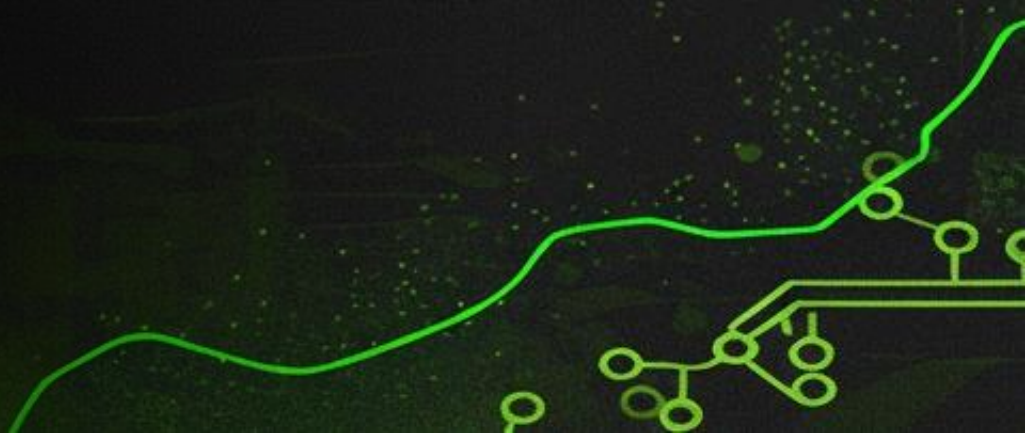
- Pathfinding is bad pls fix
 - Pathfinding is worse than Legacy
 - Pathfinding is worse this patch
 - Pathfinding is bad when you do X
 - Pathfinding is bad in this replay at time X
- 



Problems: Community feedback

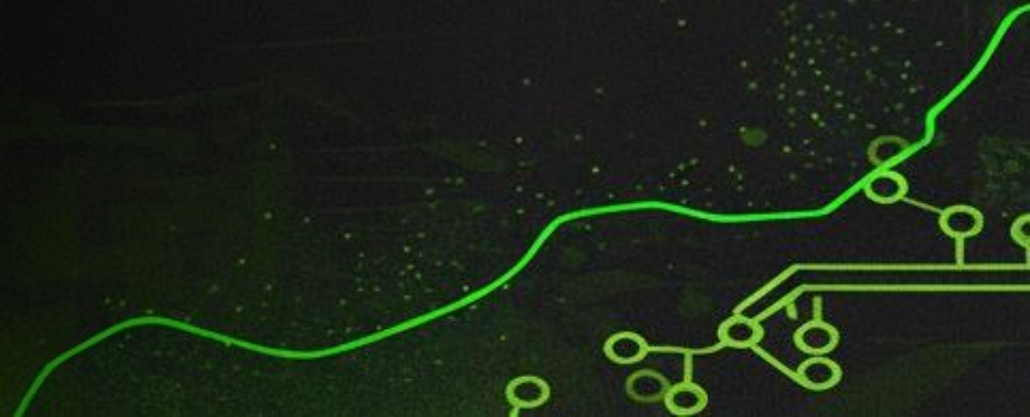
- Pathfinding is bad pls fix
- Pathfinding is worse than Legacy
- Pathfinding is worse this patch
- Pathfinding is bad when you do X
- Pathfinding is bad in this replay at time X

It finally makes sense





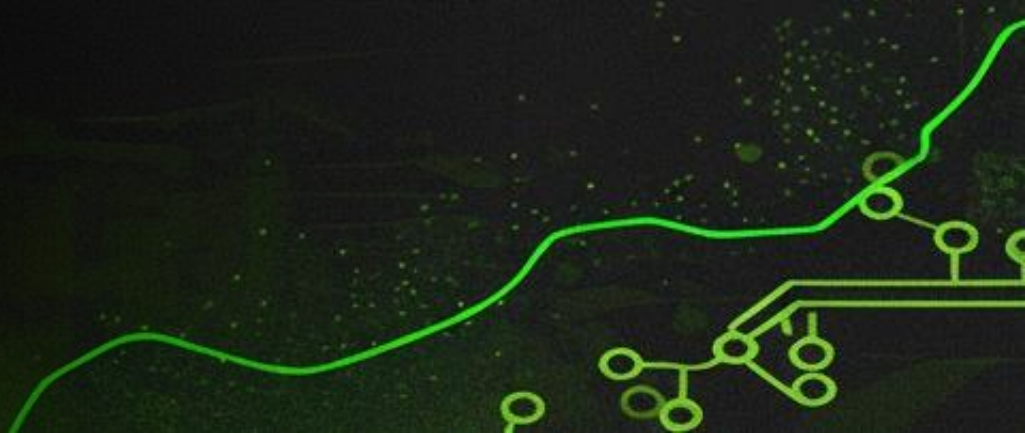
Solutions: So how do we fix it?





Solutions: So how do we fix it?

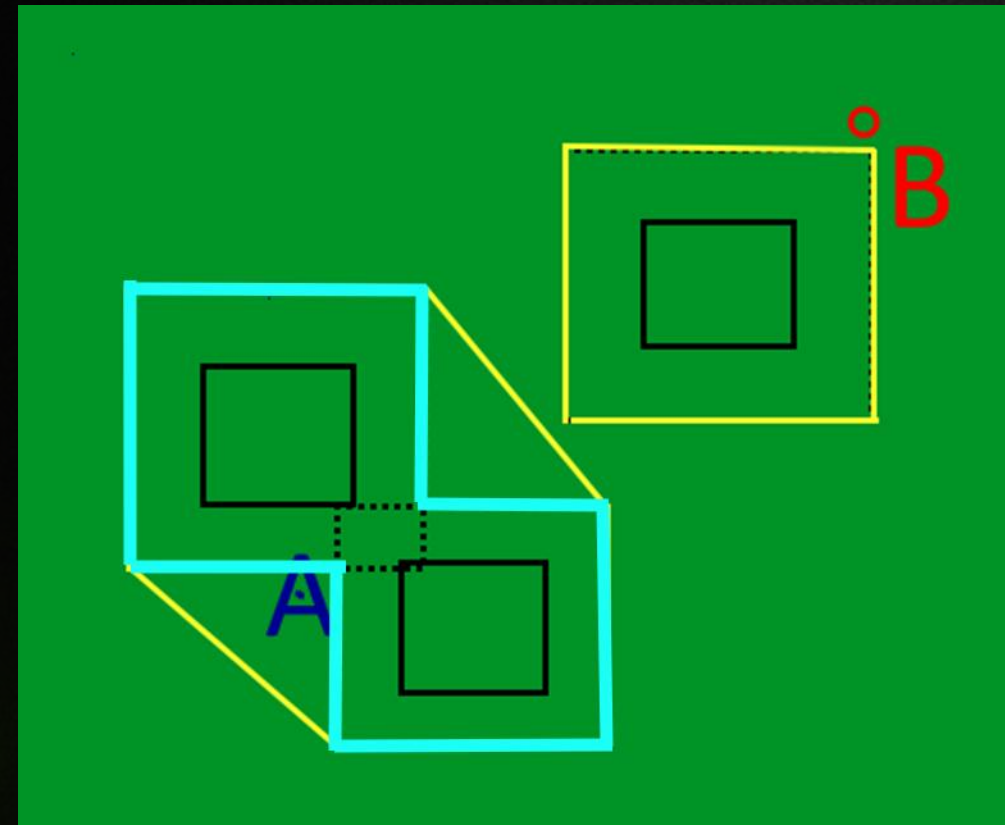
Started as a side-project to see if performance could be improved.
Mutated into something better



Solutions: Better Performance

Age of Mythology iterated on the algorithm by replacing the random bouncing with a concave hull step:

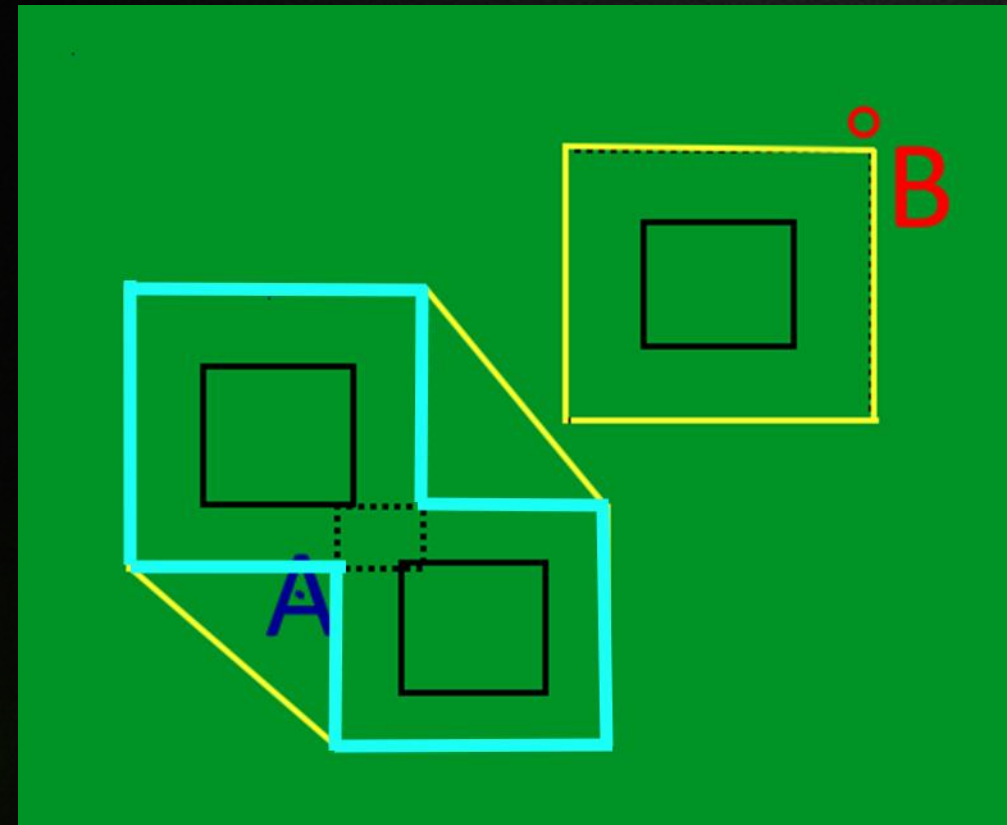
Suffered from more precision issues



Solutions: Better Performance

Can we do better?

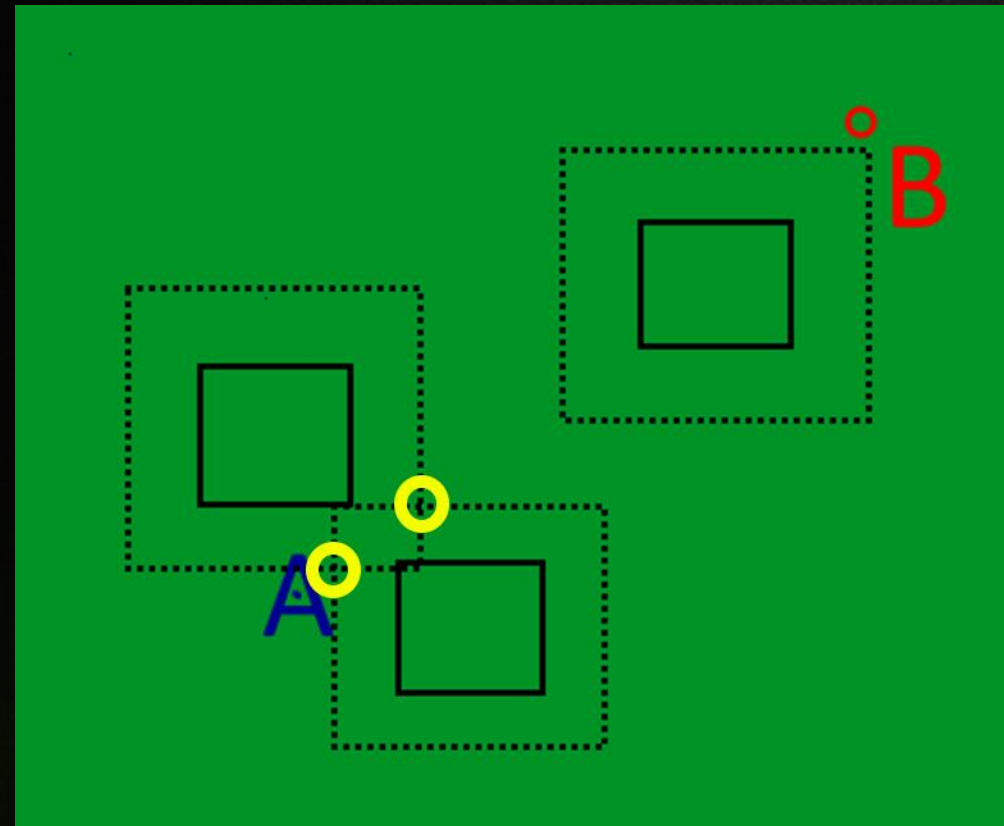
Merging rectangles does not need a full concave hull algorithm



Solutions: Better Performance

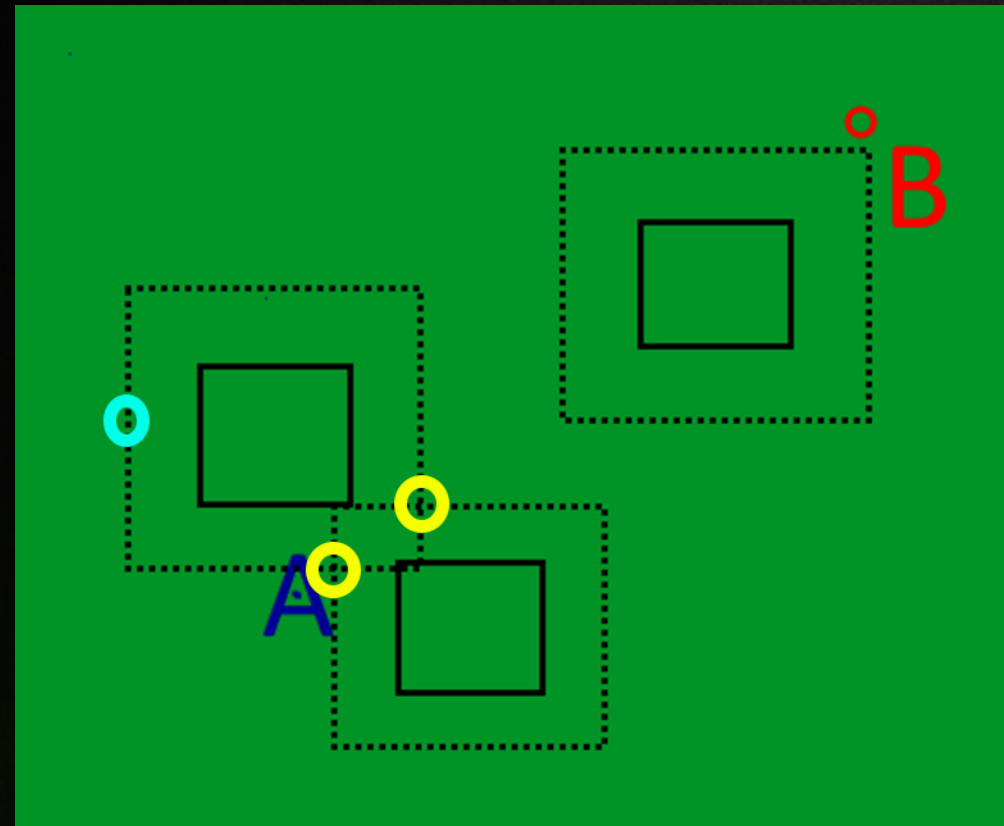
1. Calculate intersection points of rectangles

This is compatible with the radius expansion approach



Solutions: Better Performance

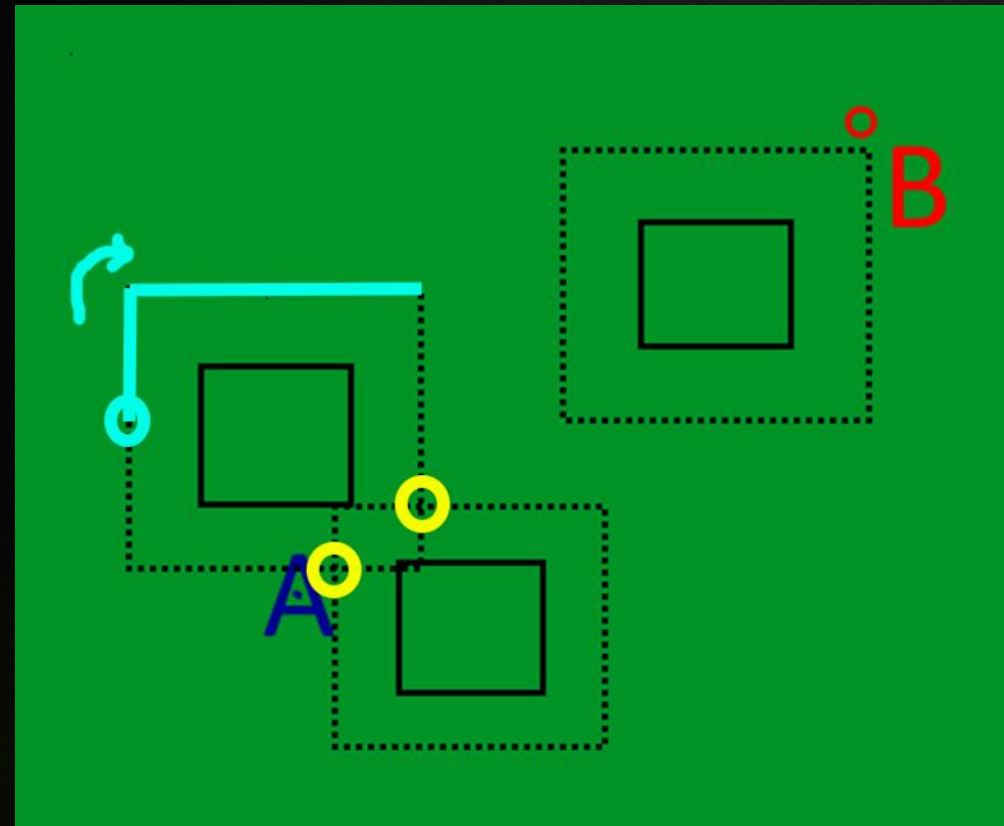
2. Pick a starting point on an edge and you start walking it



Solutions: Better Performance

3. Walk the edges:

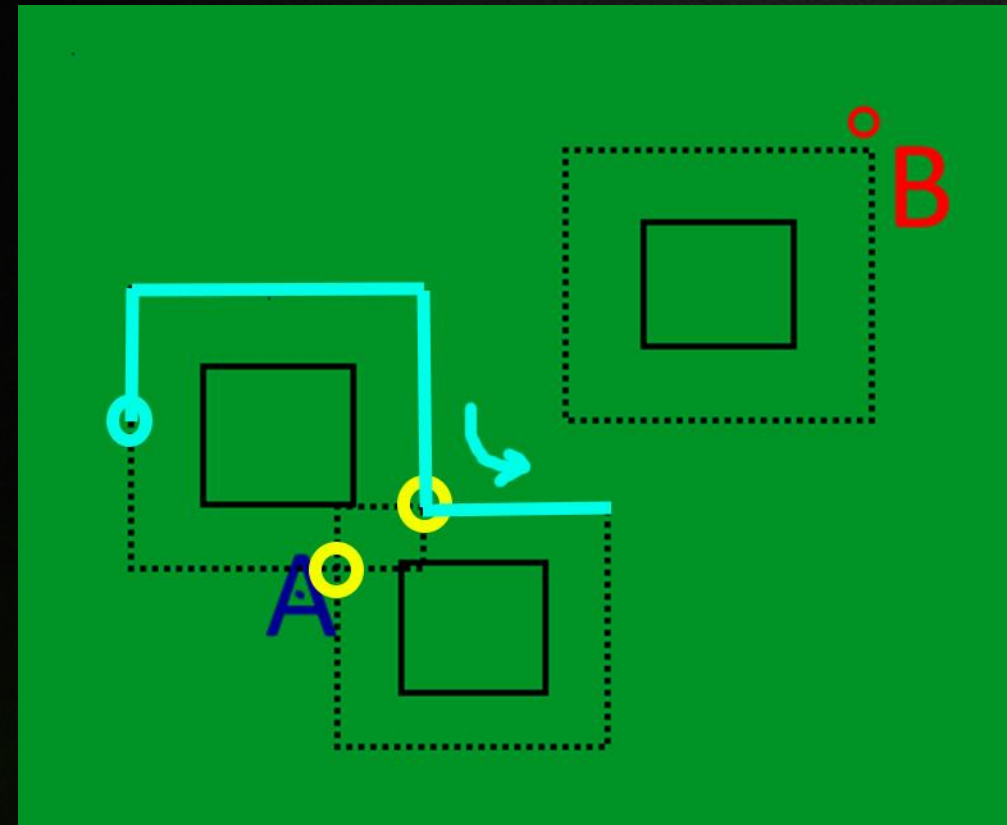
a) Turn right at the end of the edge



Solutions: Better Performance

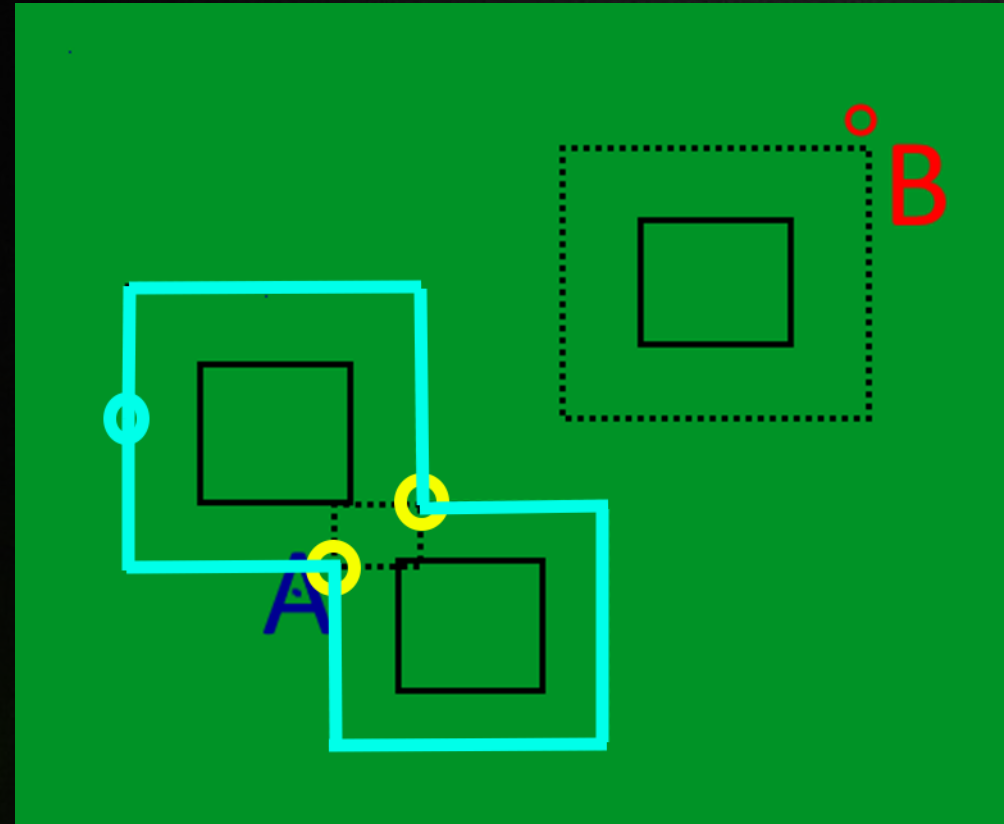
3. Walk the edges:

- a) Turn right at the end of the edge
- b) Turn left on intersection



Solutions: Better Performance

4. Continue till looped



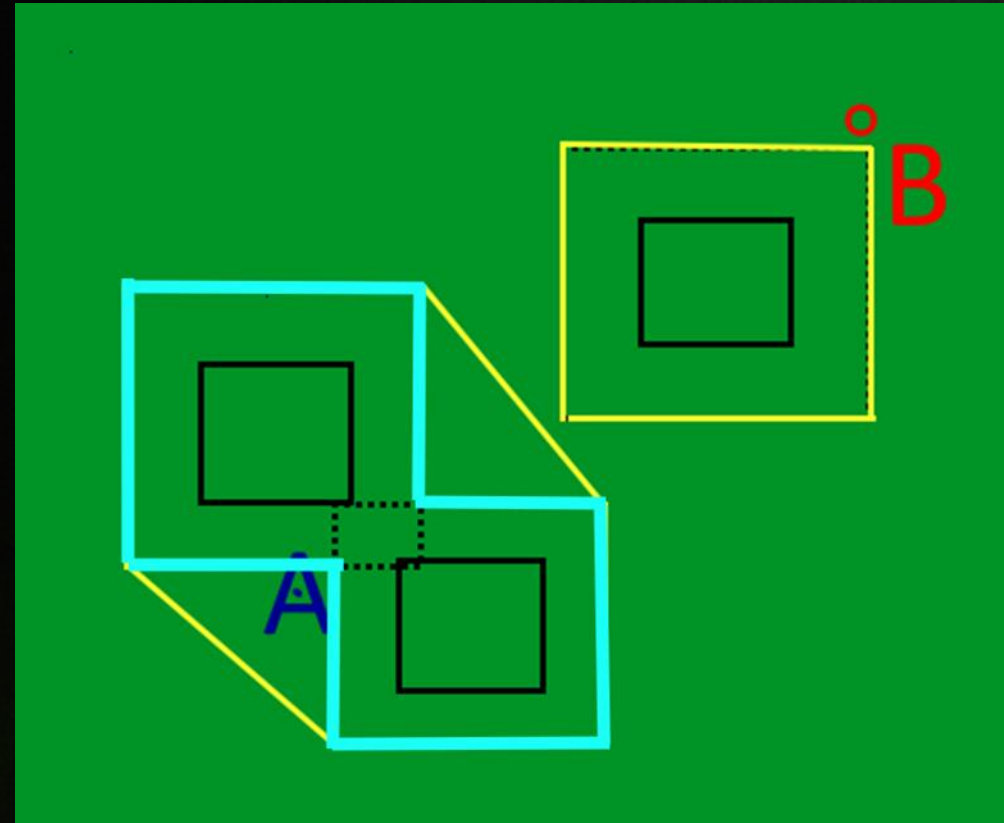
Solutions: Better Performance

5. Convex hull algorithm

https://en.wikipedia.org/wiki/Convex_hull_of_a_simple_polygon

Linear time complexity

Ended up not being required

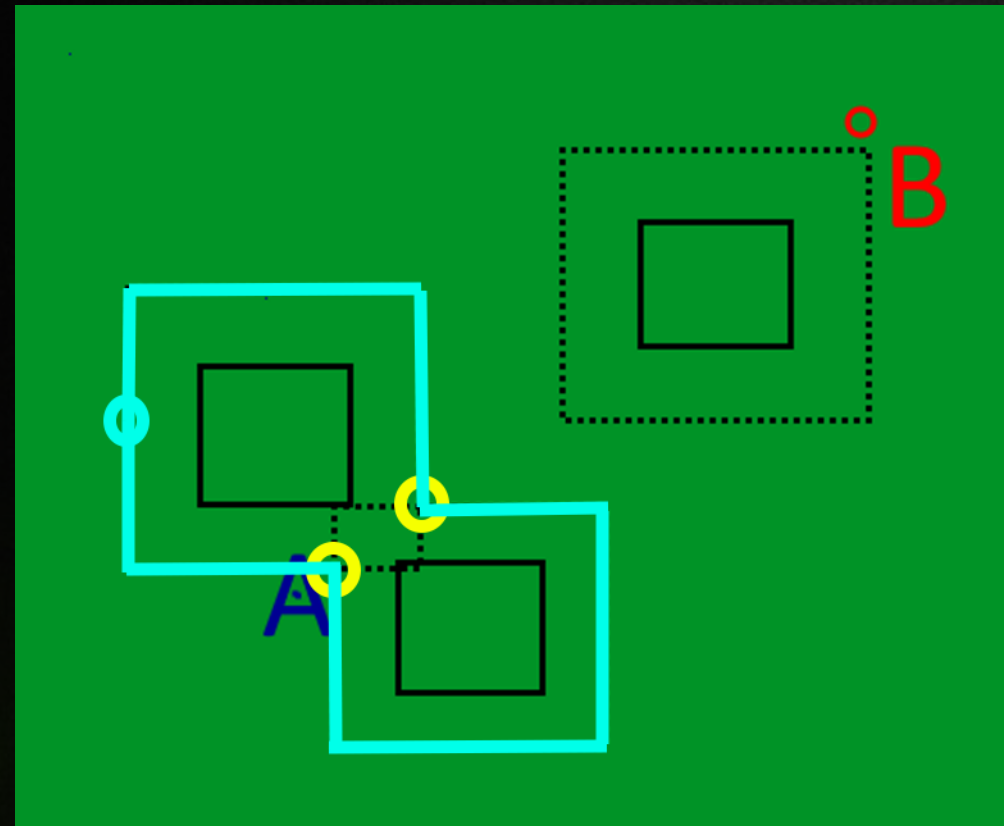


Solutions: Better Performance

This is too easy right?

Only requires $> < == + -$ operations

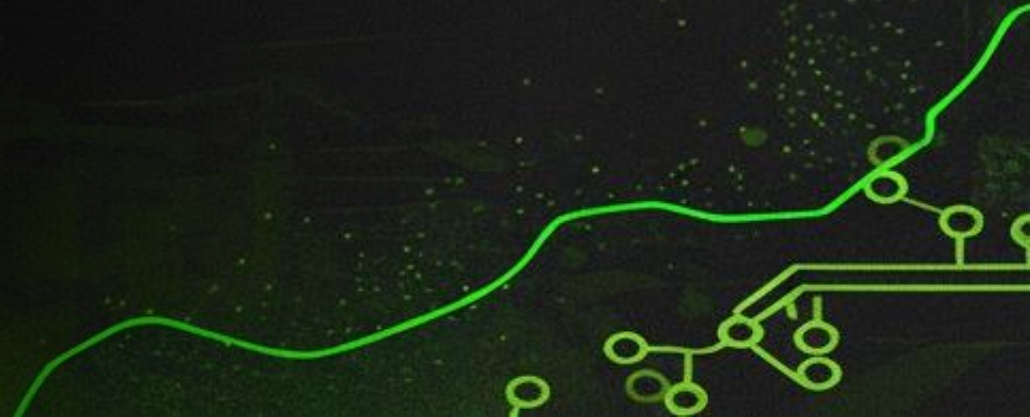
Can we verify this?





Solutions: Verifiable Algorithm

If we have a point on an edge that is not overlapped by any other rectangle **the edge walk will always give us a shape that does not overlap with any other rectangle**





Solutions: Self Verifiable Algorithm

If we fail we can output the data and debug

```
if (result == EdgeWalkResult::kInvalidHull || result == EdgeWalkResult::kOverlappingHull)
{
    OUTPUT_PATH();
    return PathResult::cPathResultError;
}
```

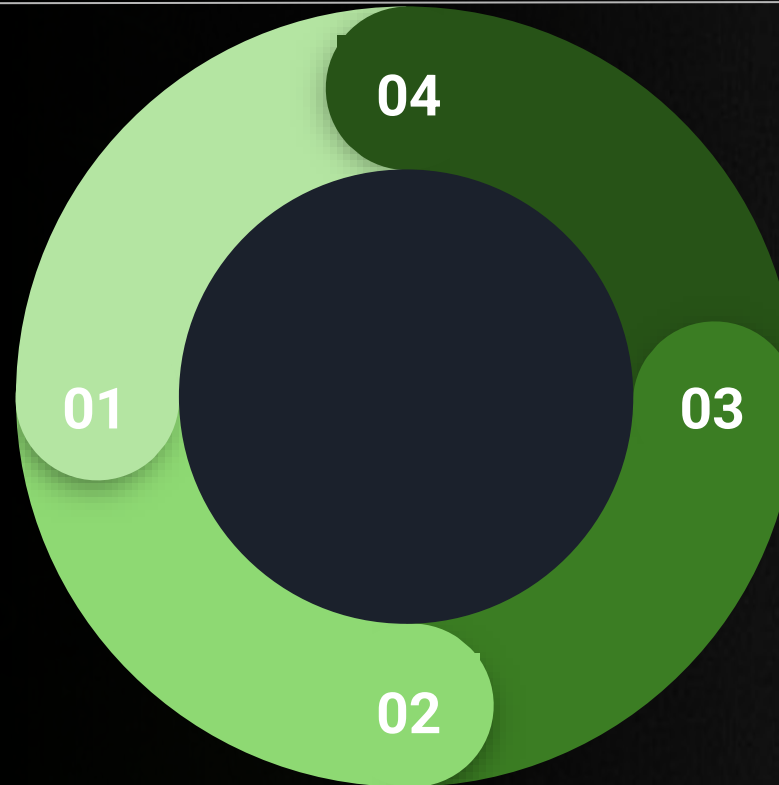

Solutions: The Path to Robustness

1. Run Game

Lots of AI's spamming
literally millions of paths

2. Find Bug

Algorithm failed
Weird path
Unit stuck



4. Run Suite

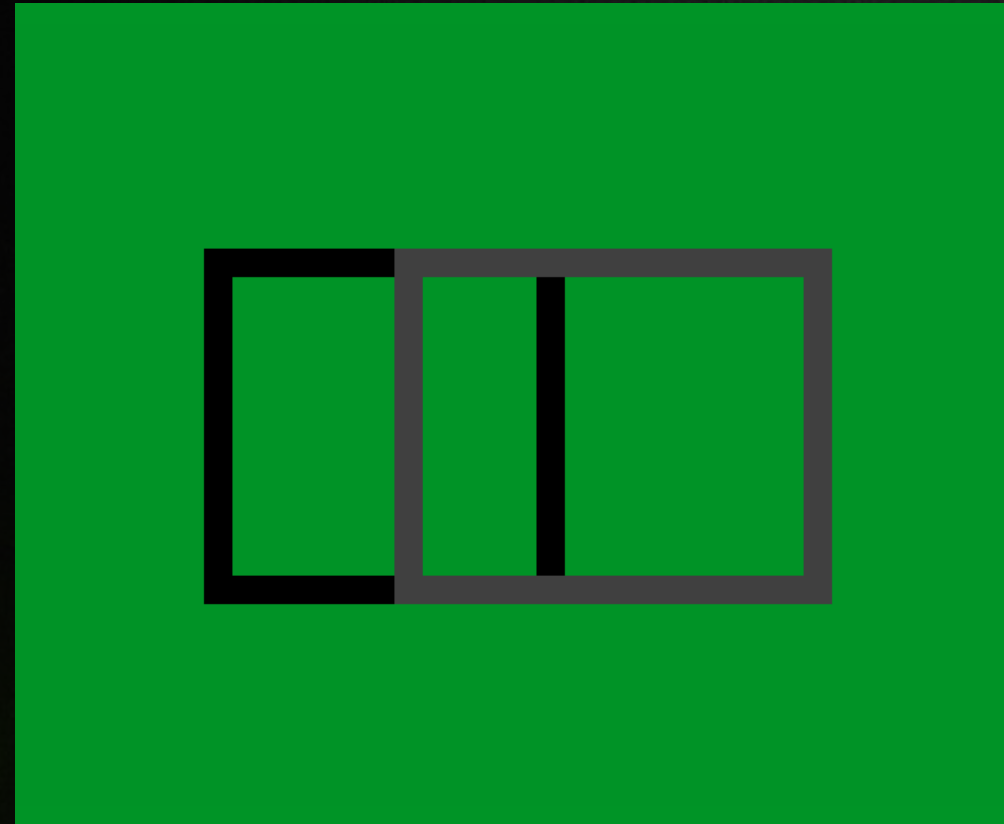
Check all previous cases
whether they still work

3. Fix Bug

Changes to the algorithm
to fix the broken path

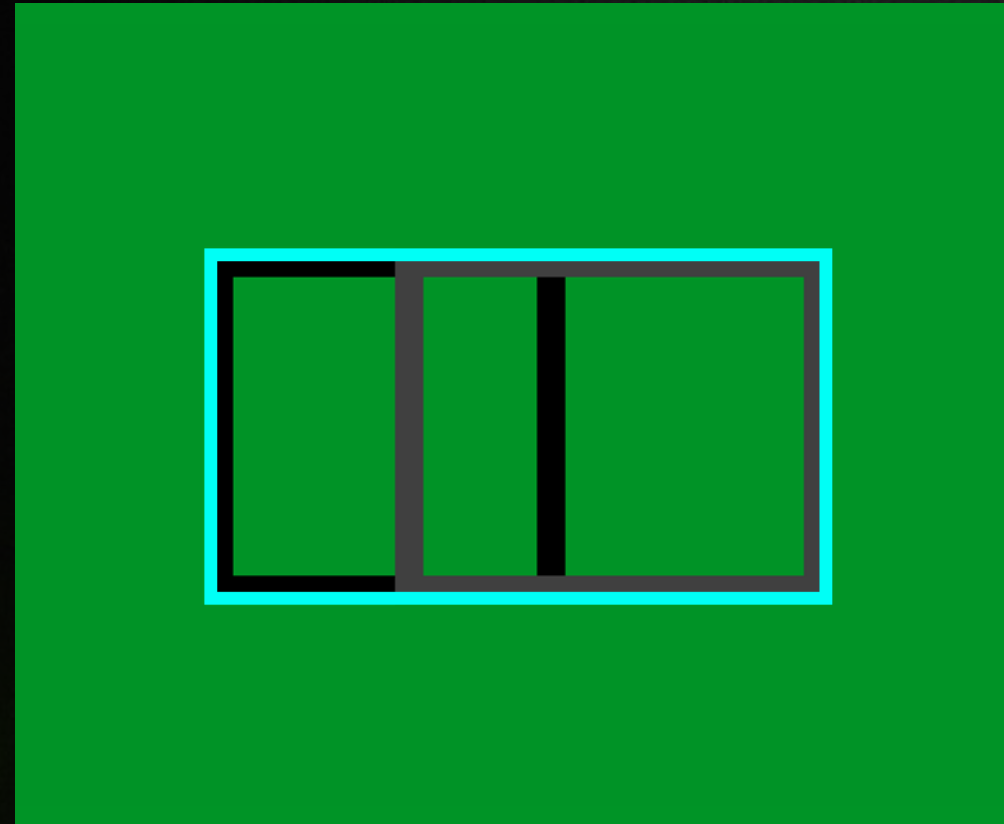
Problems: More Edge Cases

Perfectly aligning overlapping rectangles need to be handled as going left or right is wrong here.



Solutions: More Edge Cases

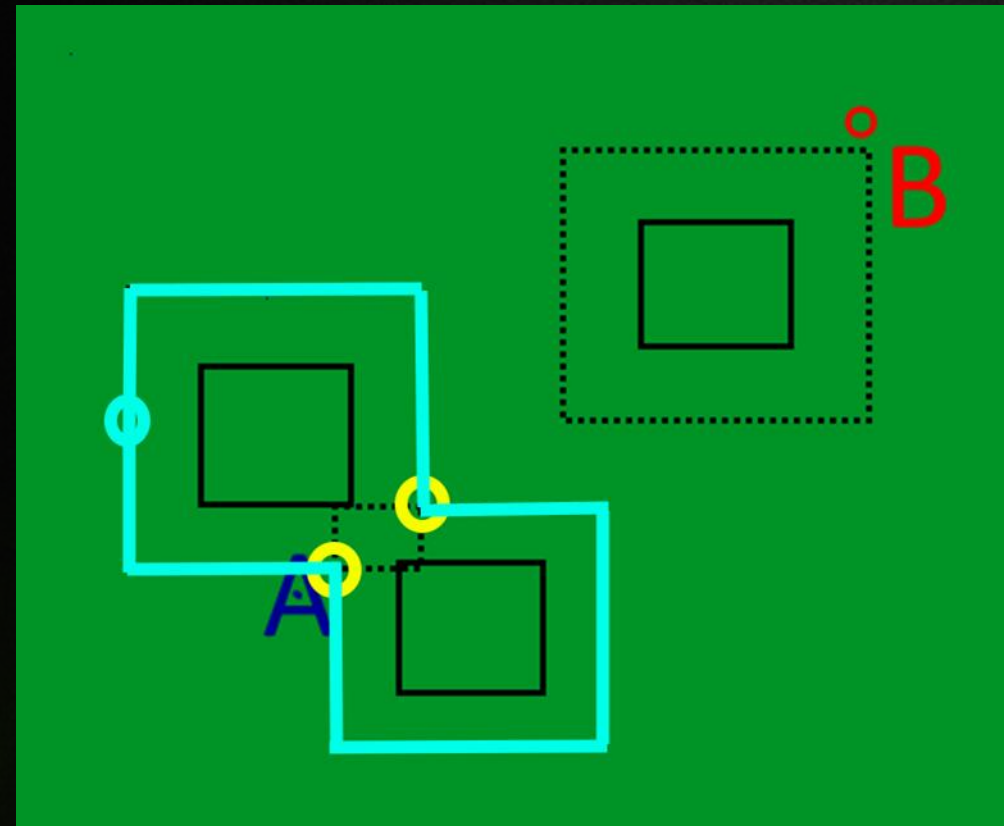
Just bridge to the intersected rectangle



Solutions: More Edge Cases

This is still too easy right?

Only requires $>$ $<$ $=$ $+$ $-$ operations





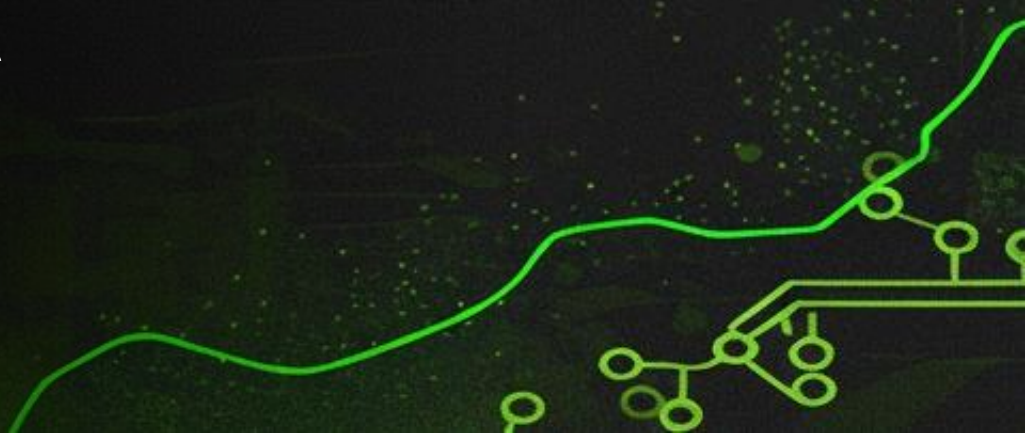
Problems: Floating Point Math

Precision can be lost on every mutable operation:

- Multiplication and division have to round
- Addition and subtraction specifically when crossing precision ranges (this was figured out the hard way)

$$180.0f + 1.0f \neq 182.0f - 1.0f^*$$

*Disclaimer I did not actually check this specific example



Problems: Floating Point Math

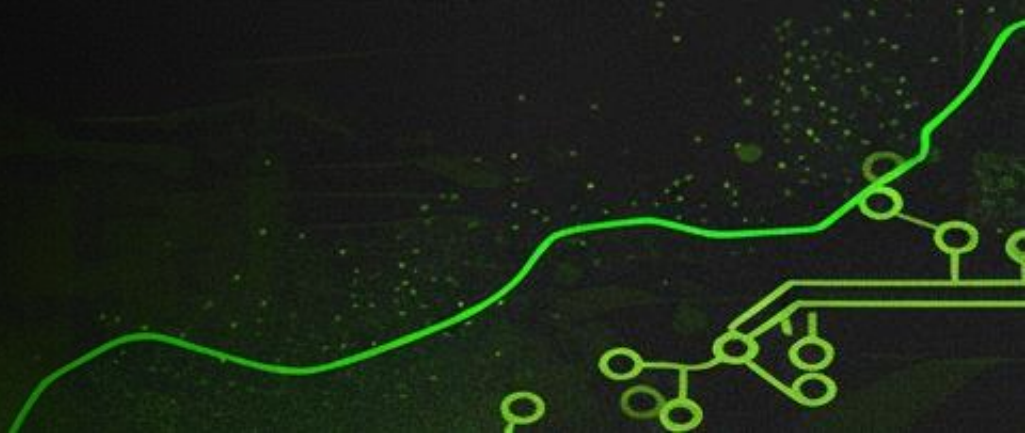
<i>exponent</i>	<i>range</i>	<i>half</i>	<i>float</i>	<i>double</i>
0	[1, 2)	0.0009765625	0.00000011920929	0.0000000000000002220446
1	[2, 4)	0.001953125	0.000000238418579	0.00000000000000044408921
2	[4, 8)	0.00390625	0.000000476837158	0.00000000000000088817842
9	[512, 1024)	0.5	0.00006103515	0.000000000000011368684
10	[1024, 2048)	1	0.00012207031	0.000000000000022737368
11	[2048, 4096)	2	0.00024414062	0.000000000000045474735
12	[4096, 8192)	4	0.00048828125	0.00000000000009094947
15	[32768, 65536)	32	0.00390625	0.0000000000072759576
16	[65536, 131072)	N/A	0.0078125	0.0000000000014551915
17	[131072, 262144)	N/A	0.015625	0.000000000002910383
18	[262144, 524288)	N/A	0.03125	0.0000000000058207661
19	[524288, 1048576)	N/A	0.0625	0.000000000011641532
23	[8388608, 16777216)	N/A	1	0.00000000186264515
52	[4503599627370496, 9007199254740992)	N/A	536870912	1



Solutions: Fixed Point Math

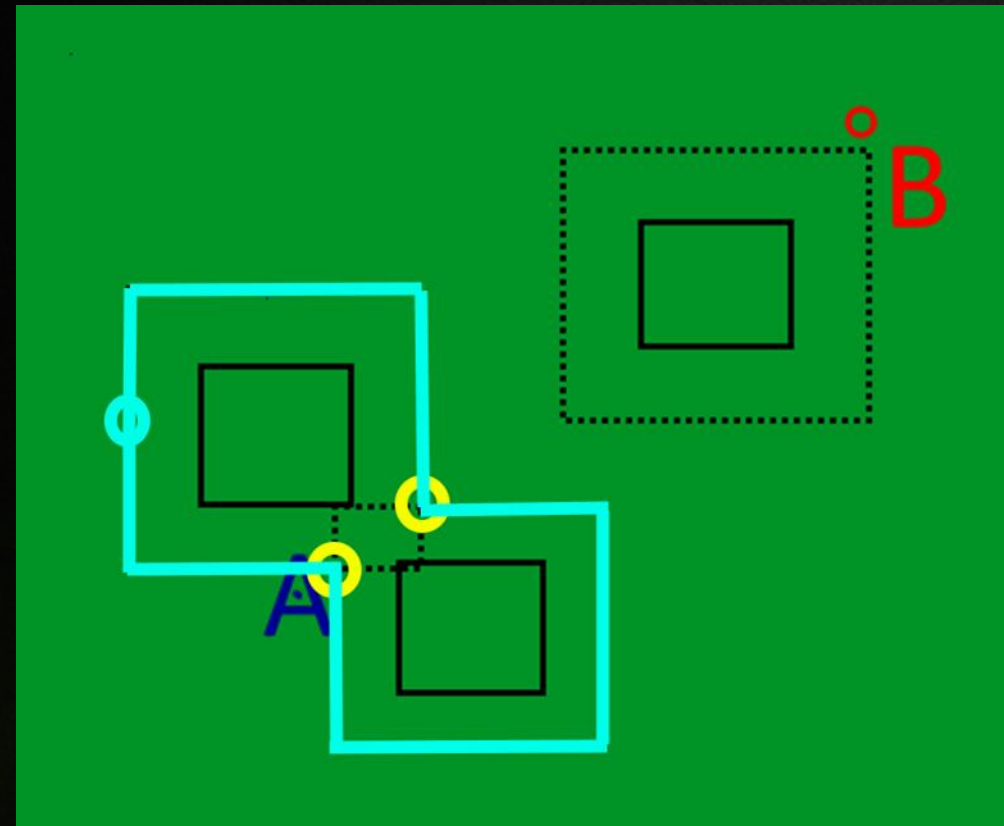
Precision is only lost on multiplication and division!

This will allow us to use addition and subtraction in our edge walk:

$$180.0 + 1.0 == 182.0 - 1.0$$


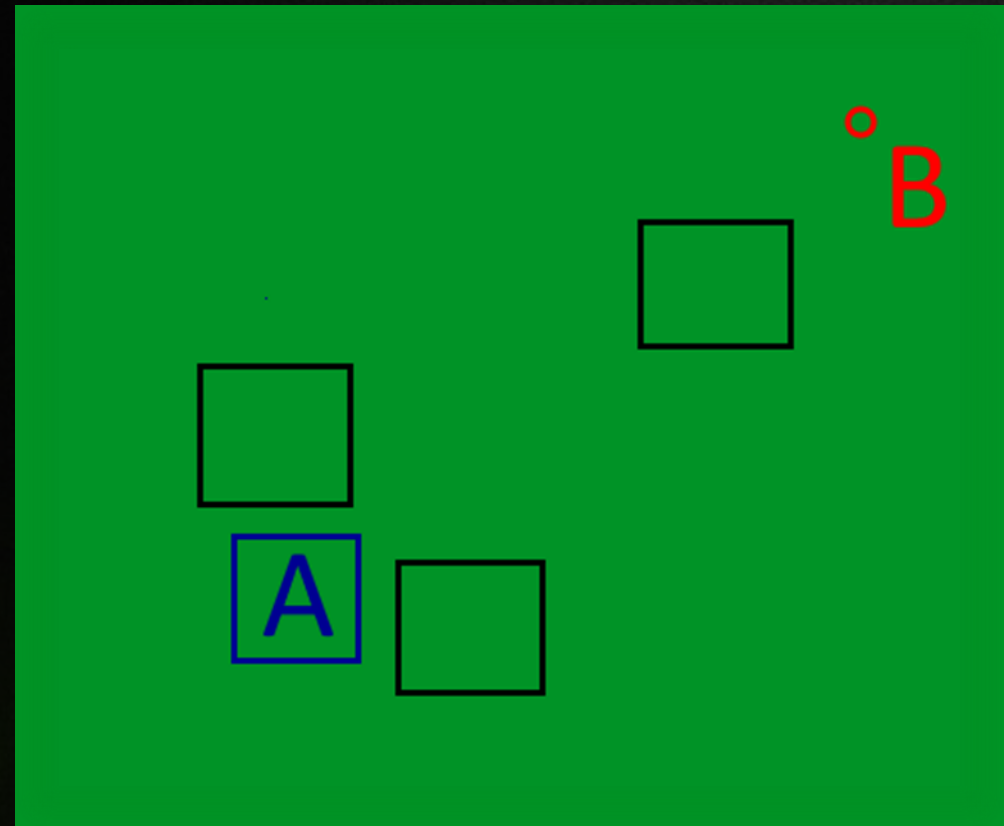
Solutions: More Edge Cases

Only requires $> < == + -$ operations



Solutions: The New Way

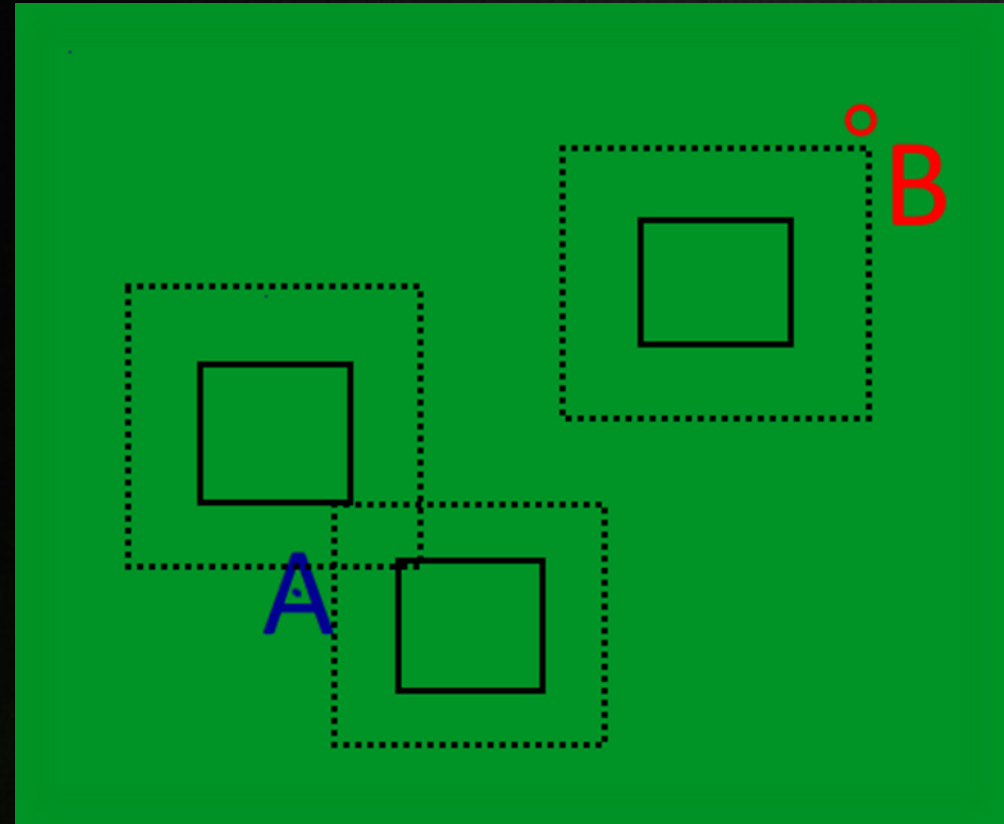
To path with unit A to point B:



Solutions: The New Way

To path with unit A to point B:

1. Shrink ourselves to a point and expand the obstructions

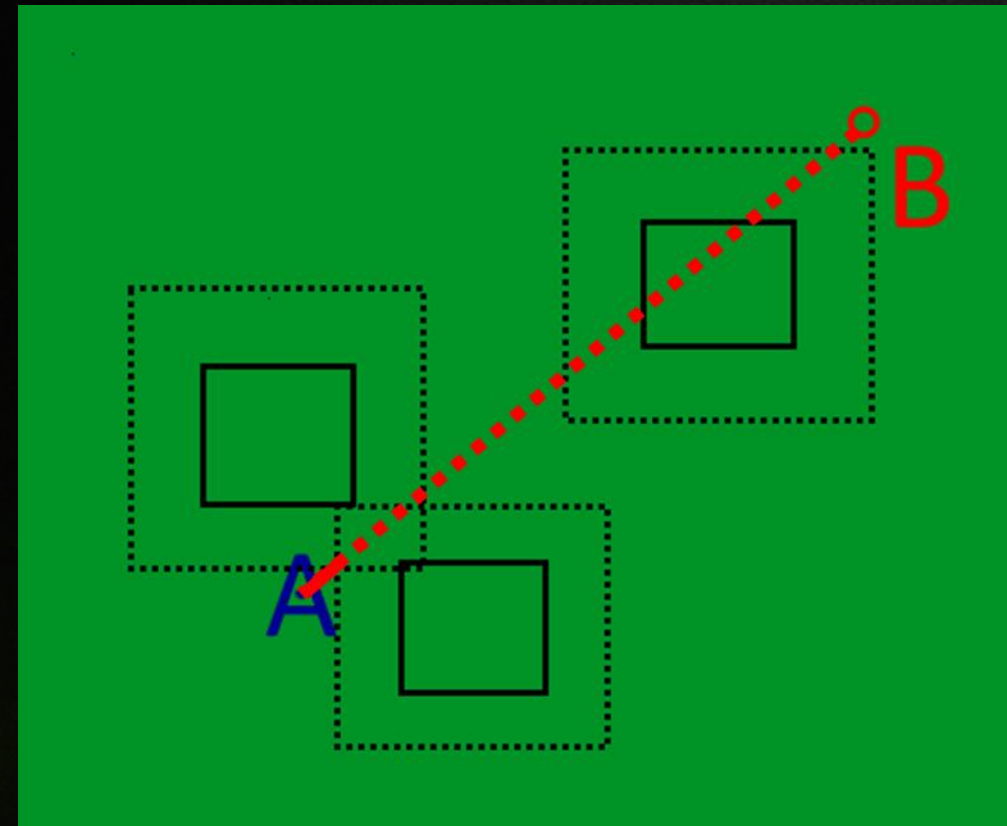


Solutions: The New Way

To path with unit A to point B:

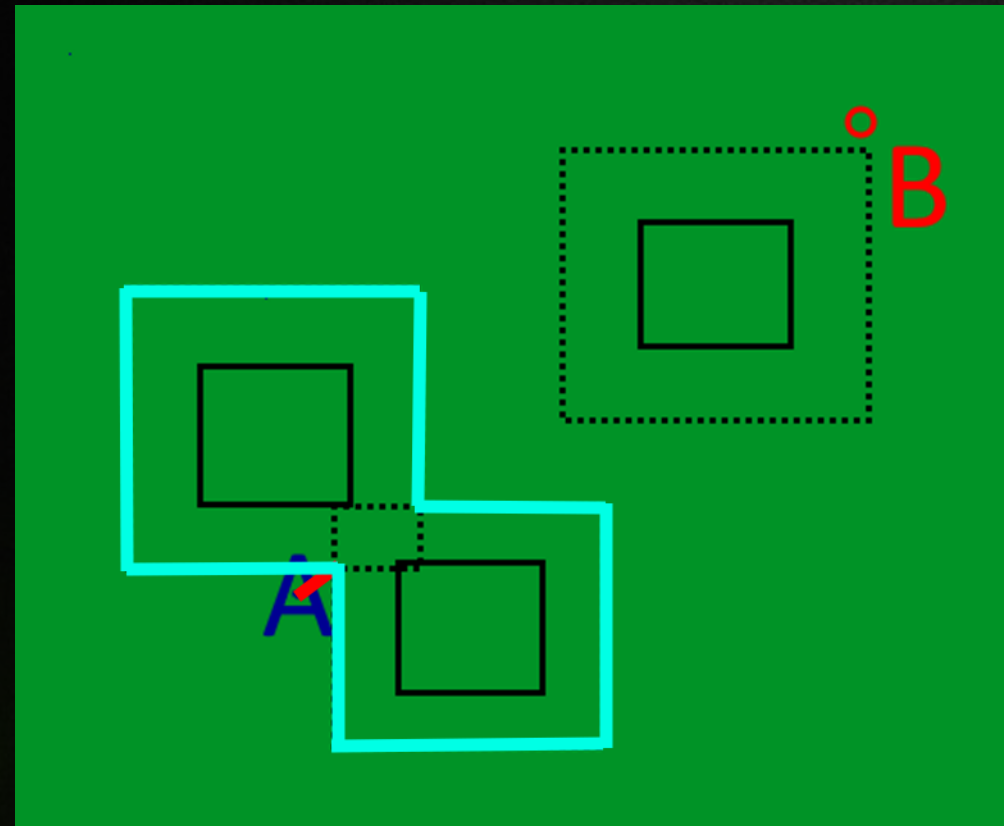
2. Shoot a ray towards our goal

- a) If we hit we have our starting edge walk point
- b) Otherwise we have our path



Solutions: The New Way

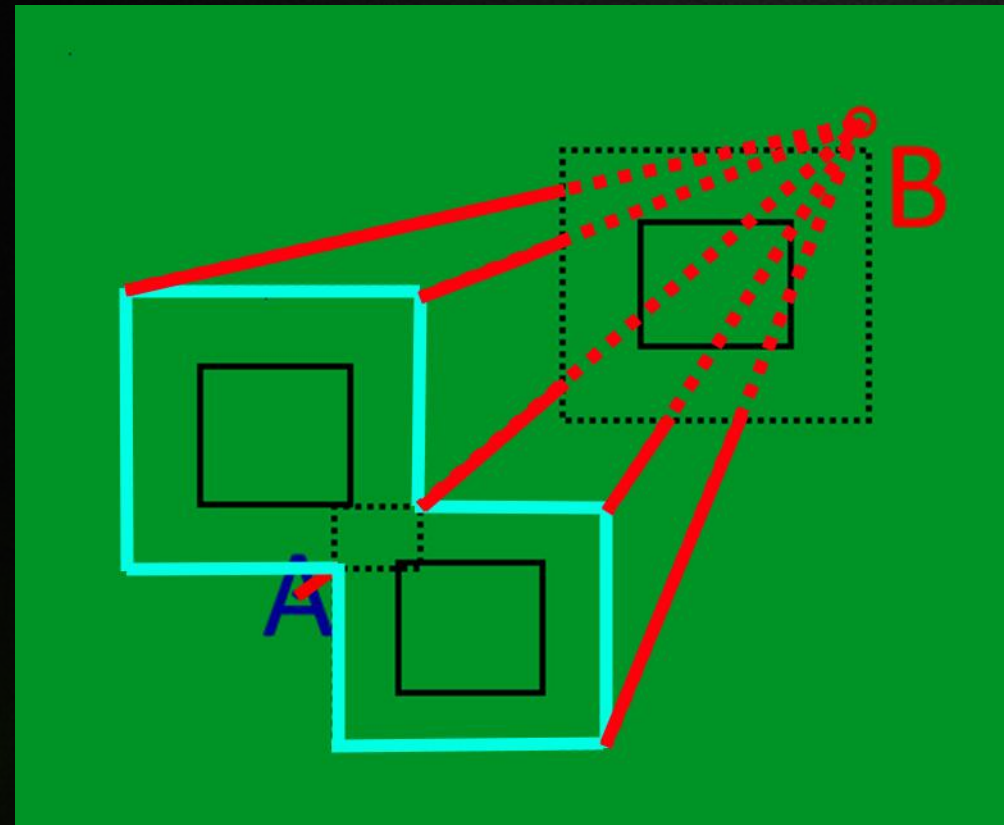
To path with unit A to point B:
3. Walk our edges



Solutions: The New Way

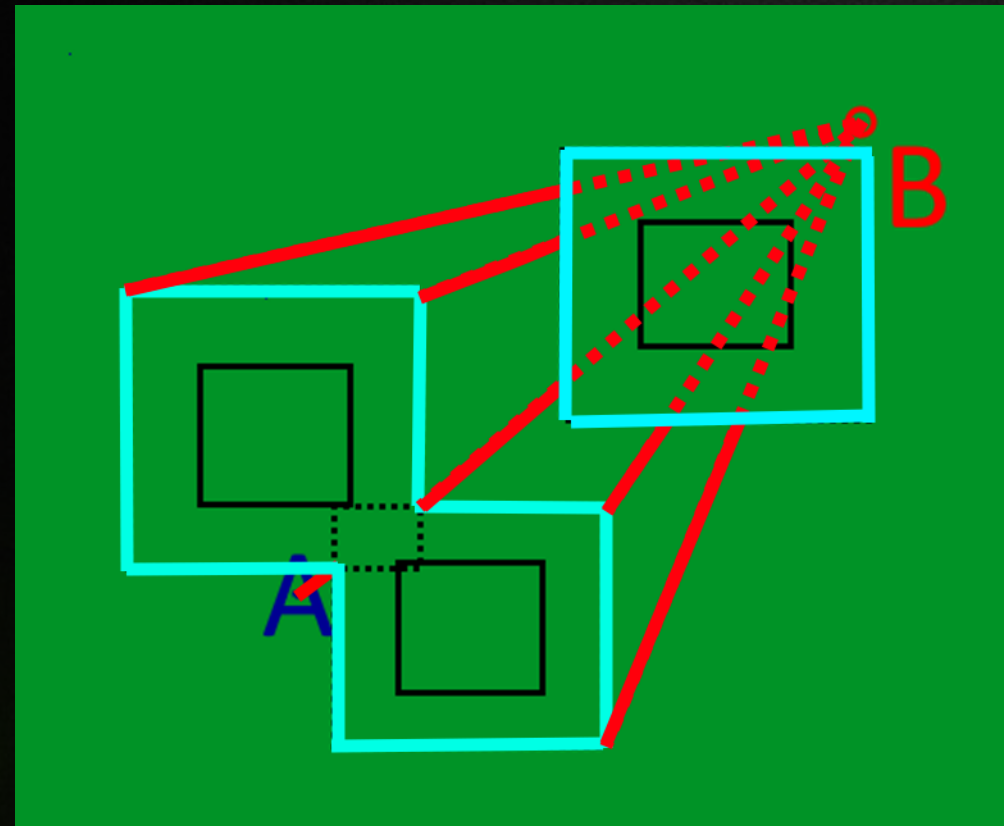
To path with unit A to point B:

4. At every vertex go back to step 2

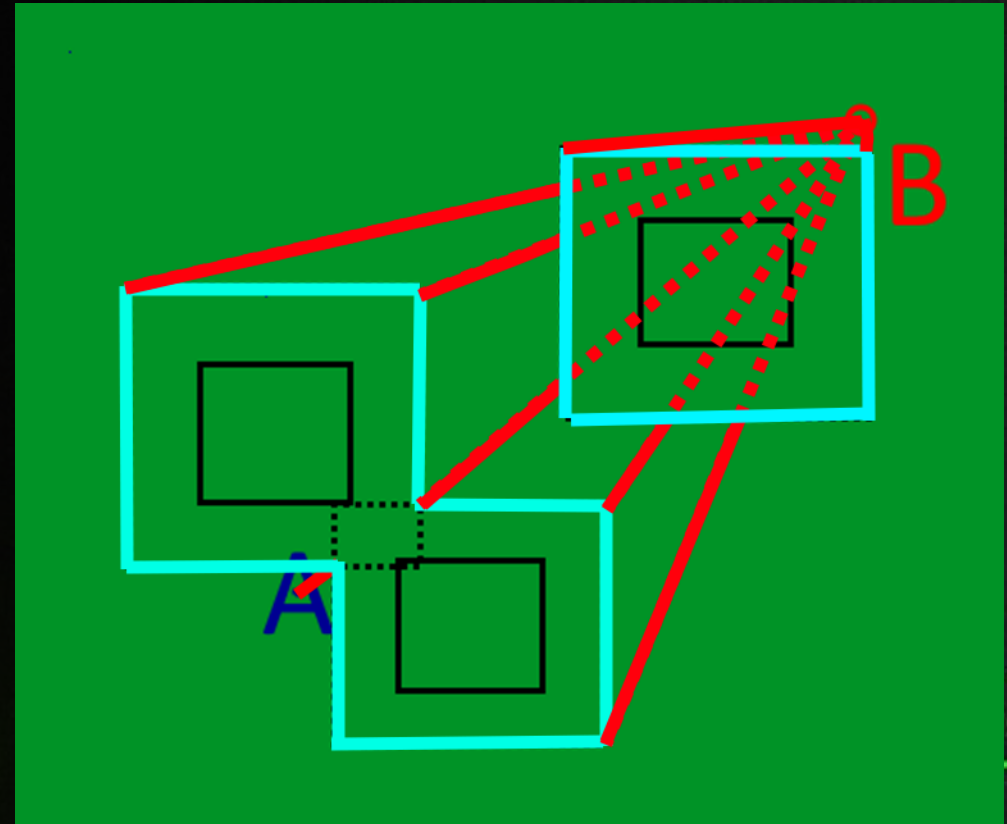


Solutions: The New Way

To path with unit A to point B:
3. Walk our edges again

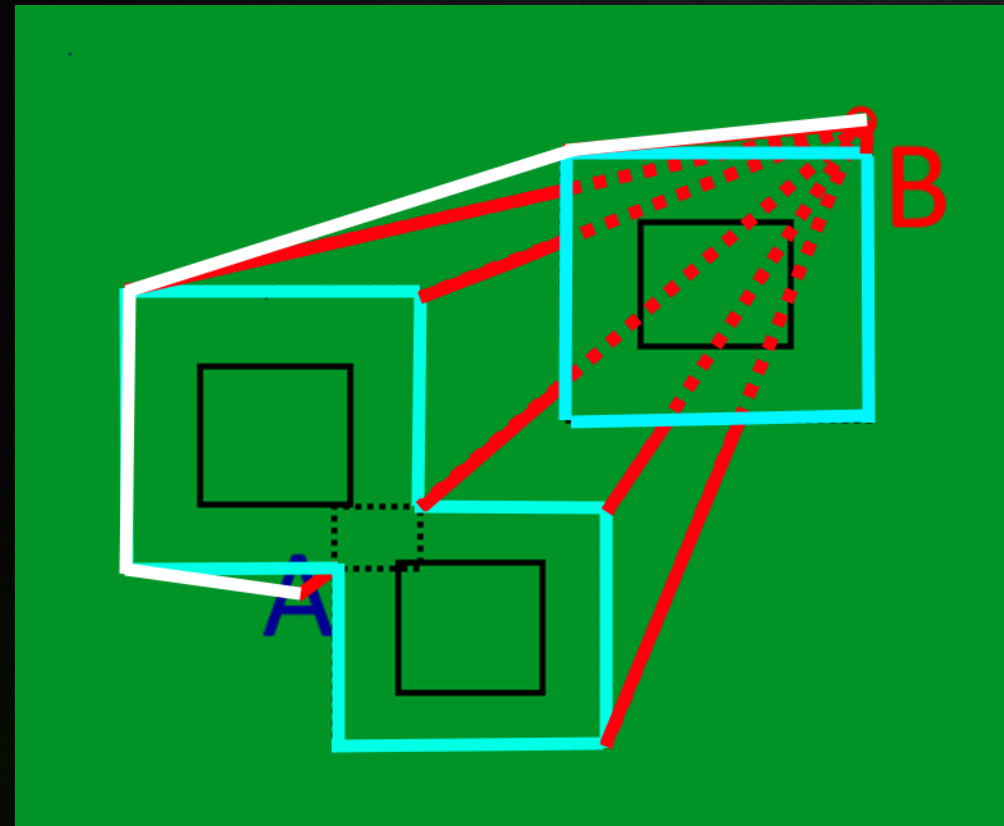


Solutions: The New Way



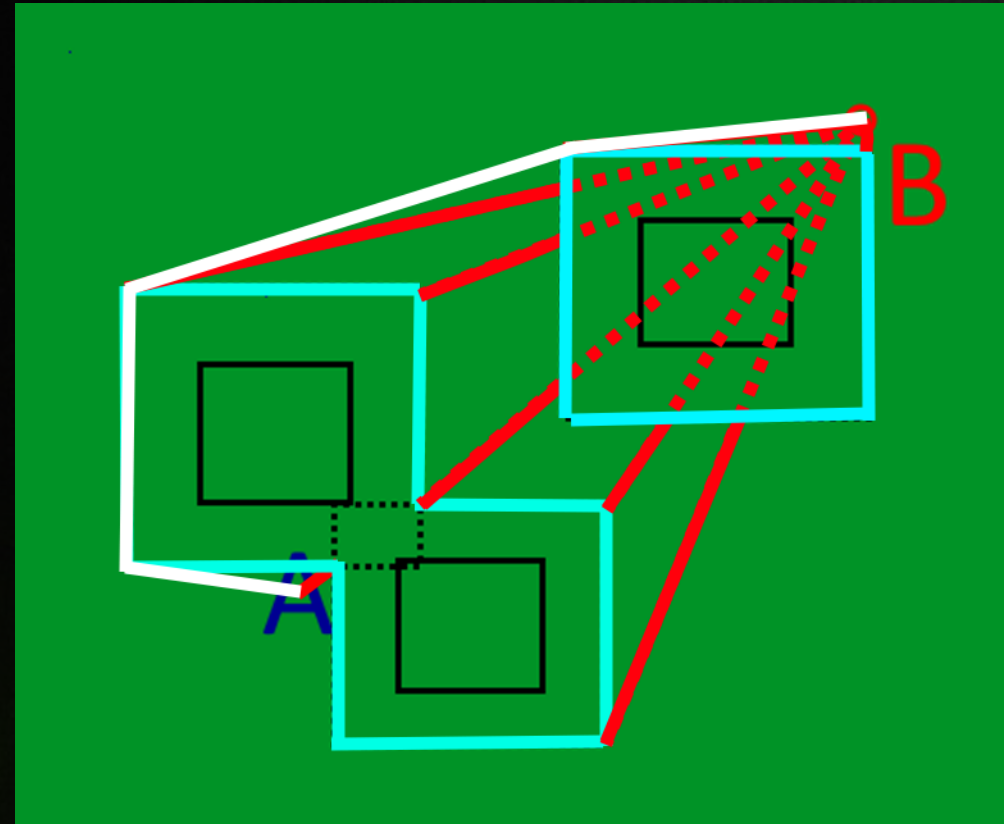
Solutions: The New Way

To path with unit A to point B:
5. Brute force smooth



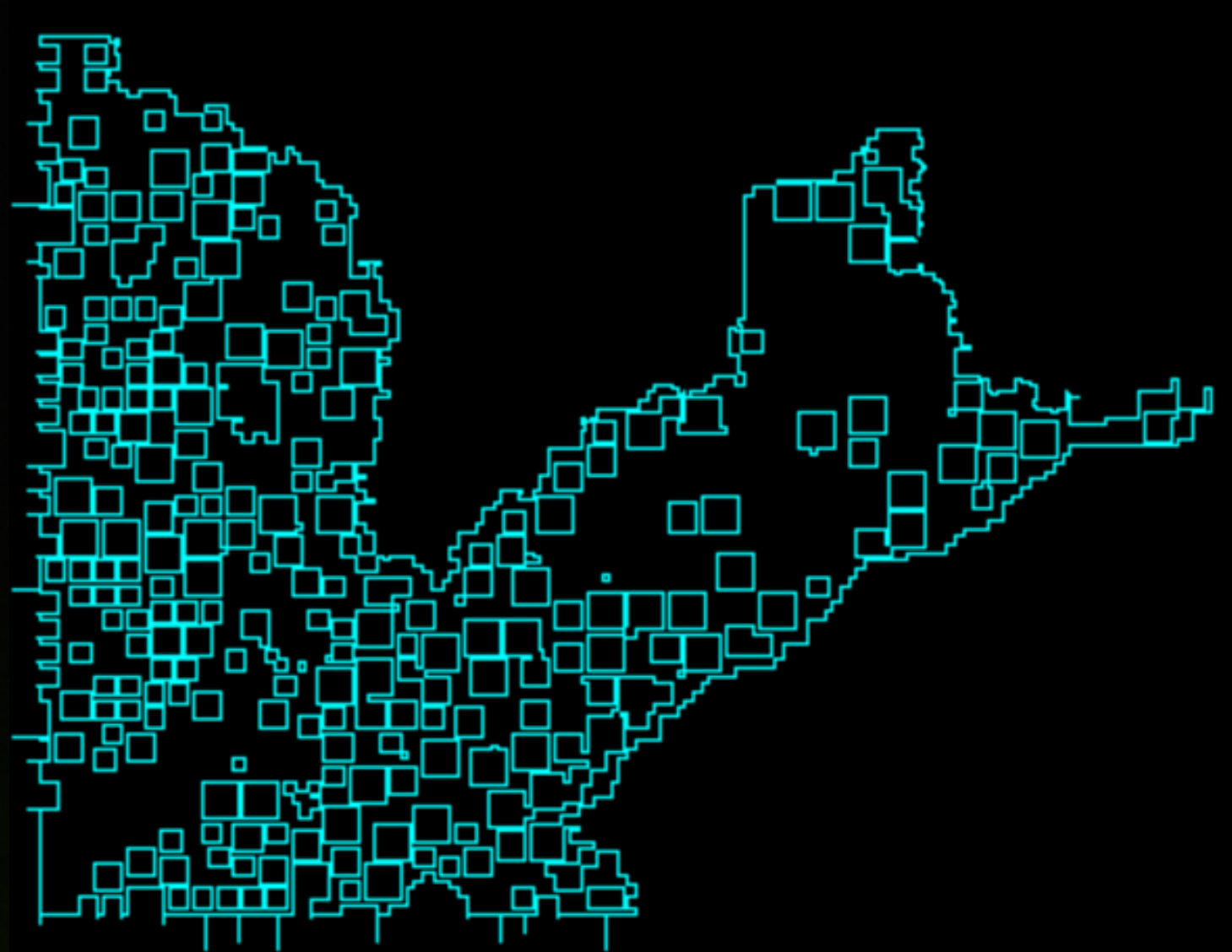
Solutions: The New Way

Too easy again right?



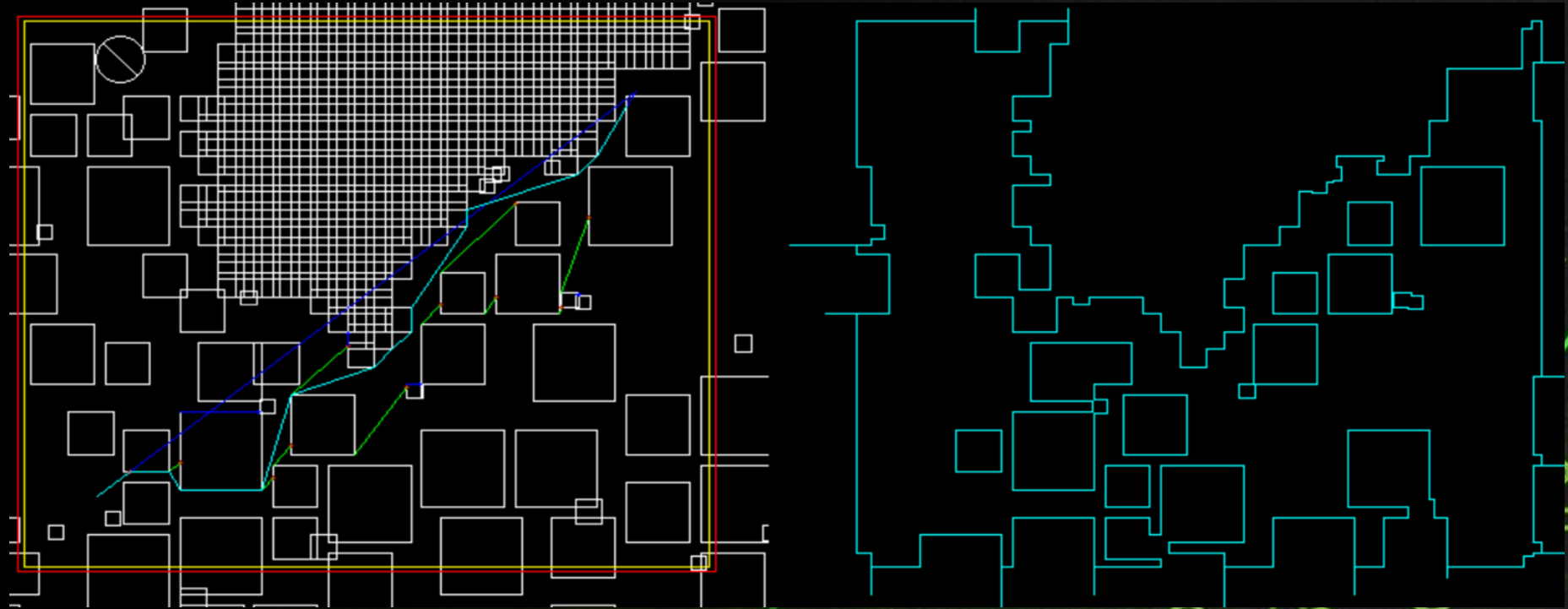
Problems: The New Way

Whoops, we edge-walked the whole map



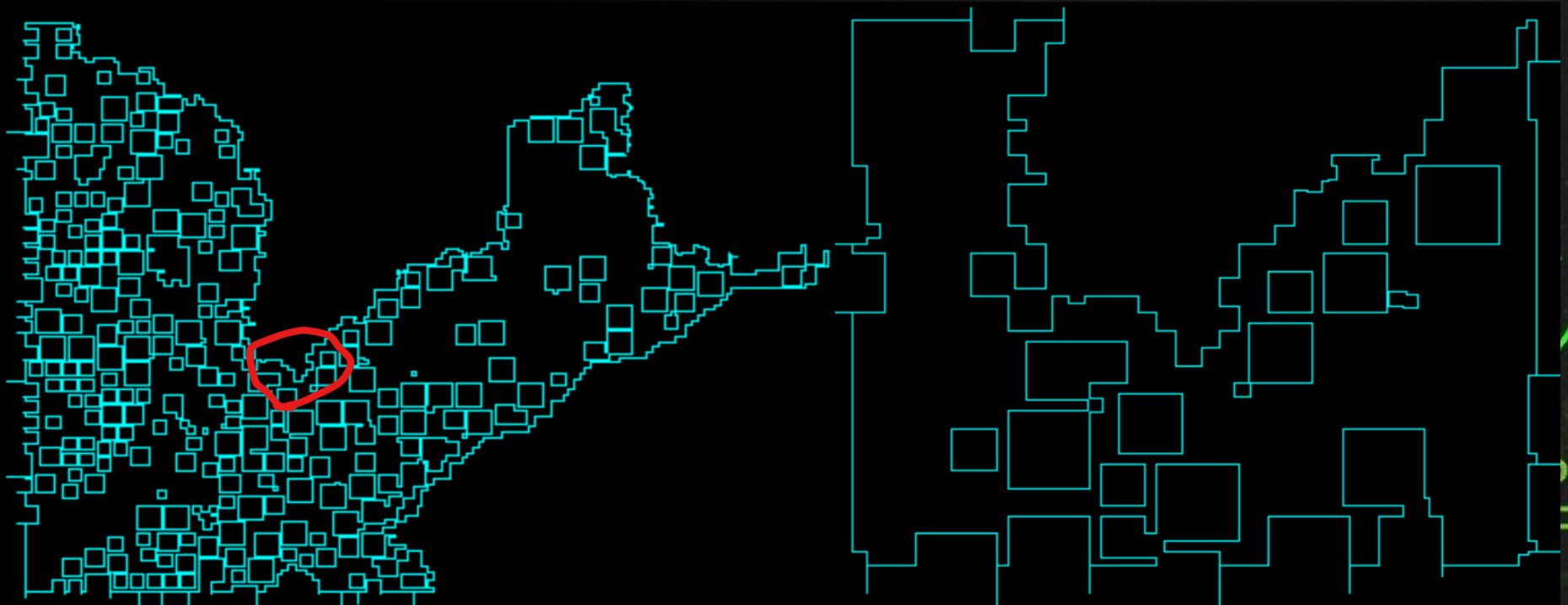
Solutions: The New Way

Add rectangles around
our search area:



Solutions: The New Way

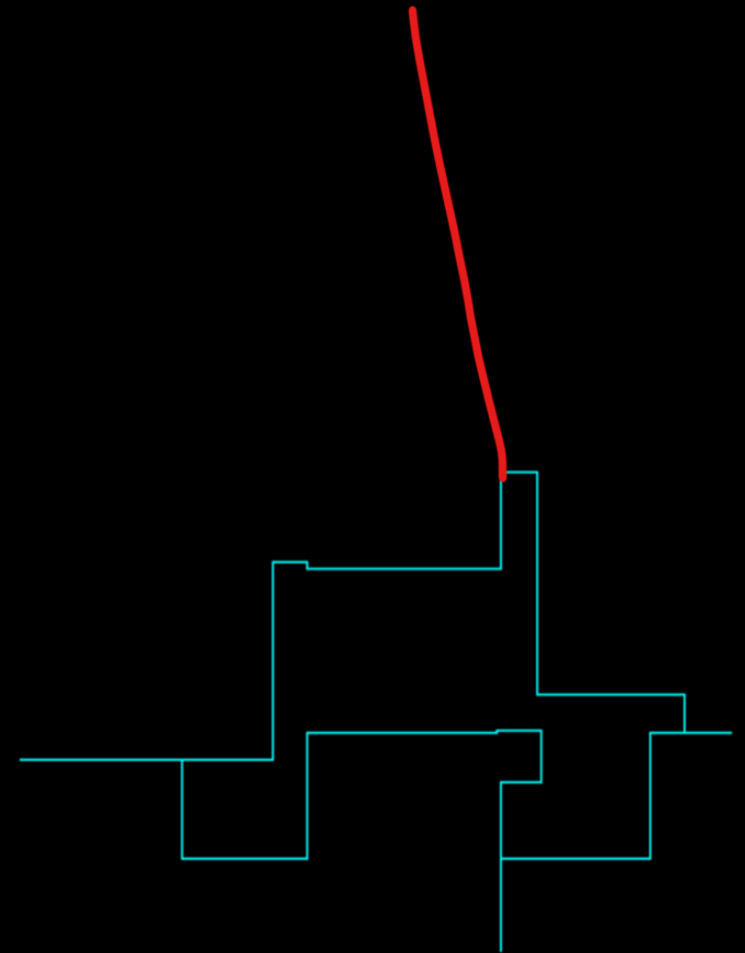
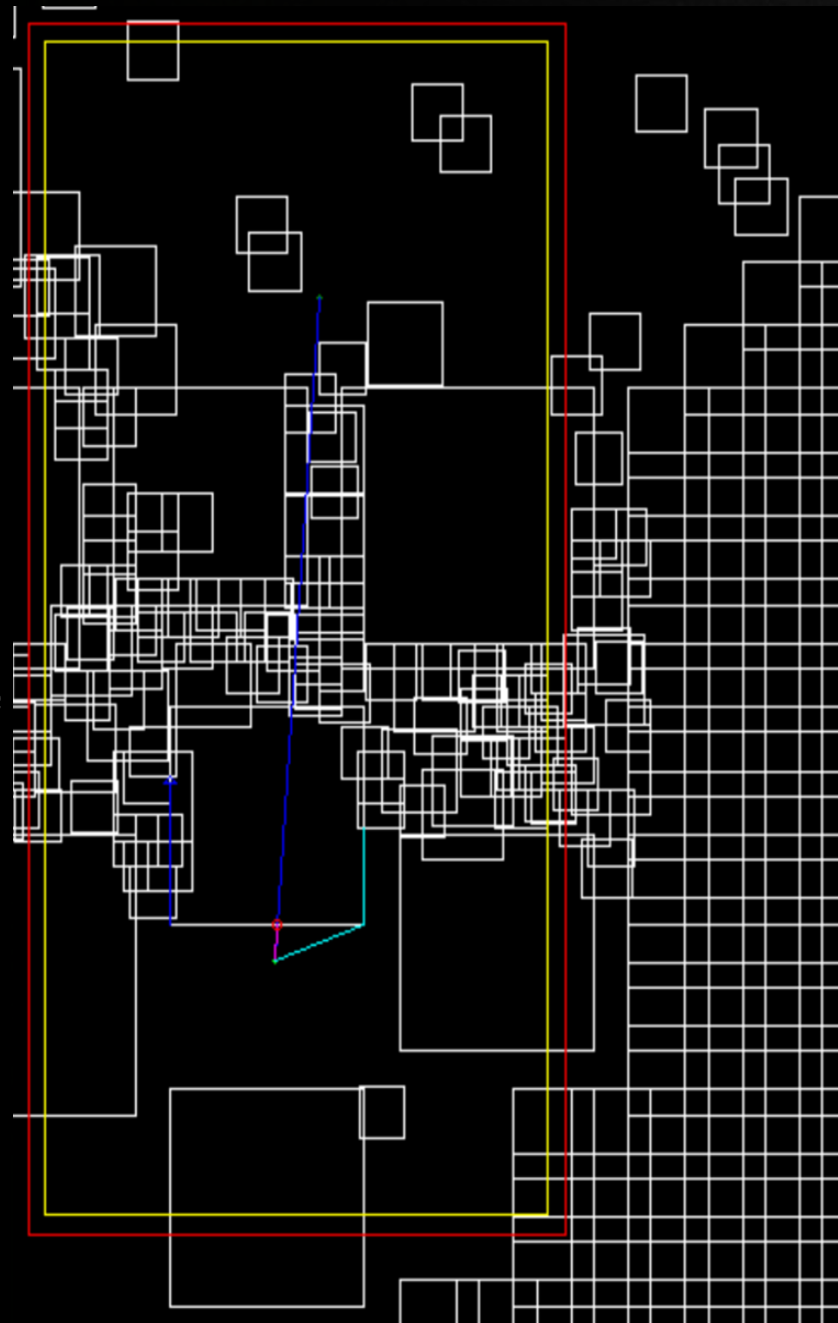
Add rectangles around
our search area:



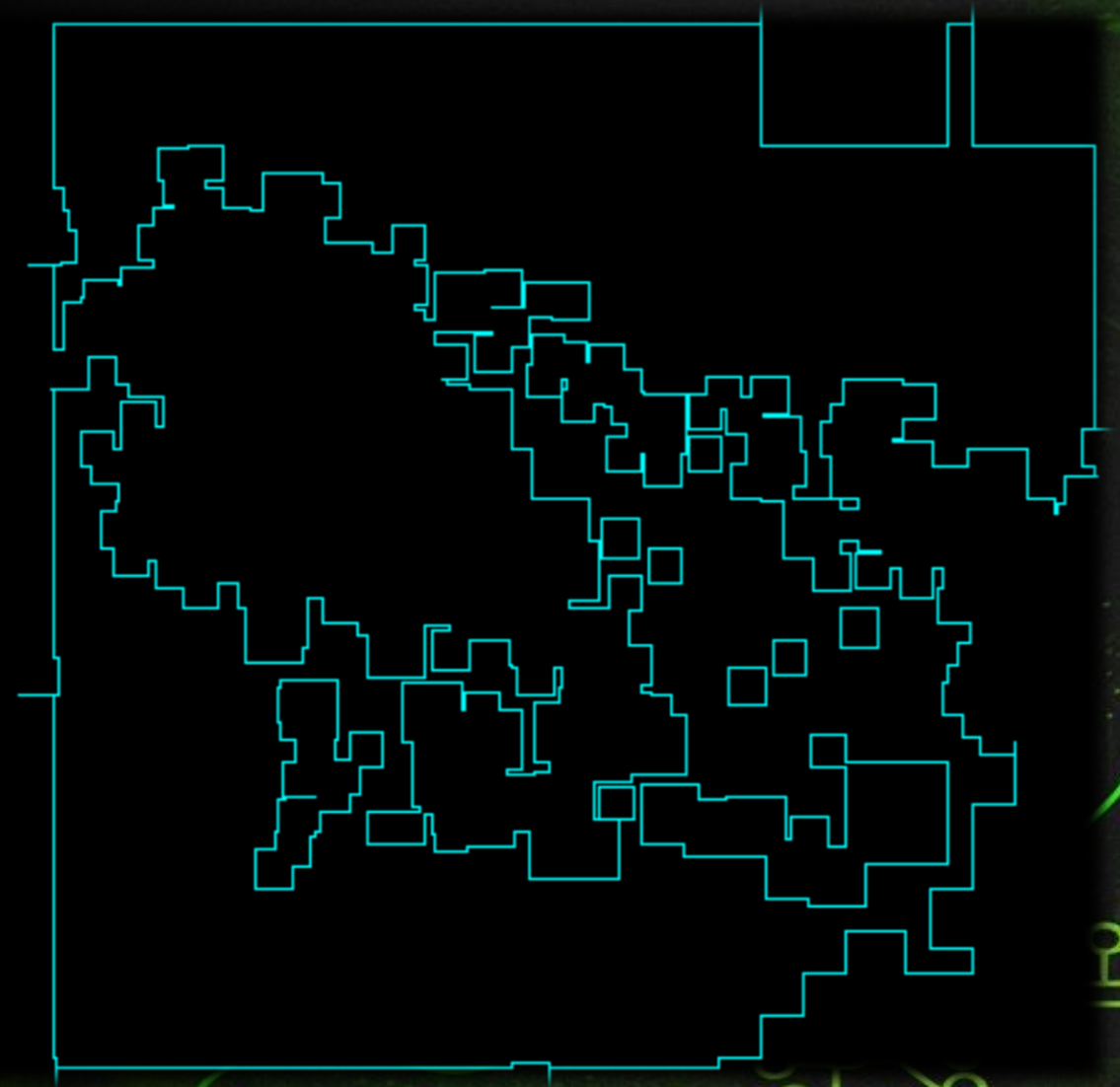
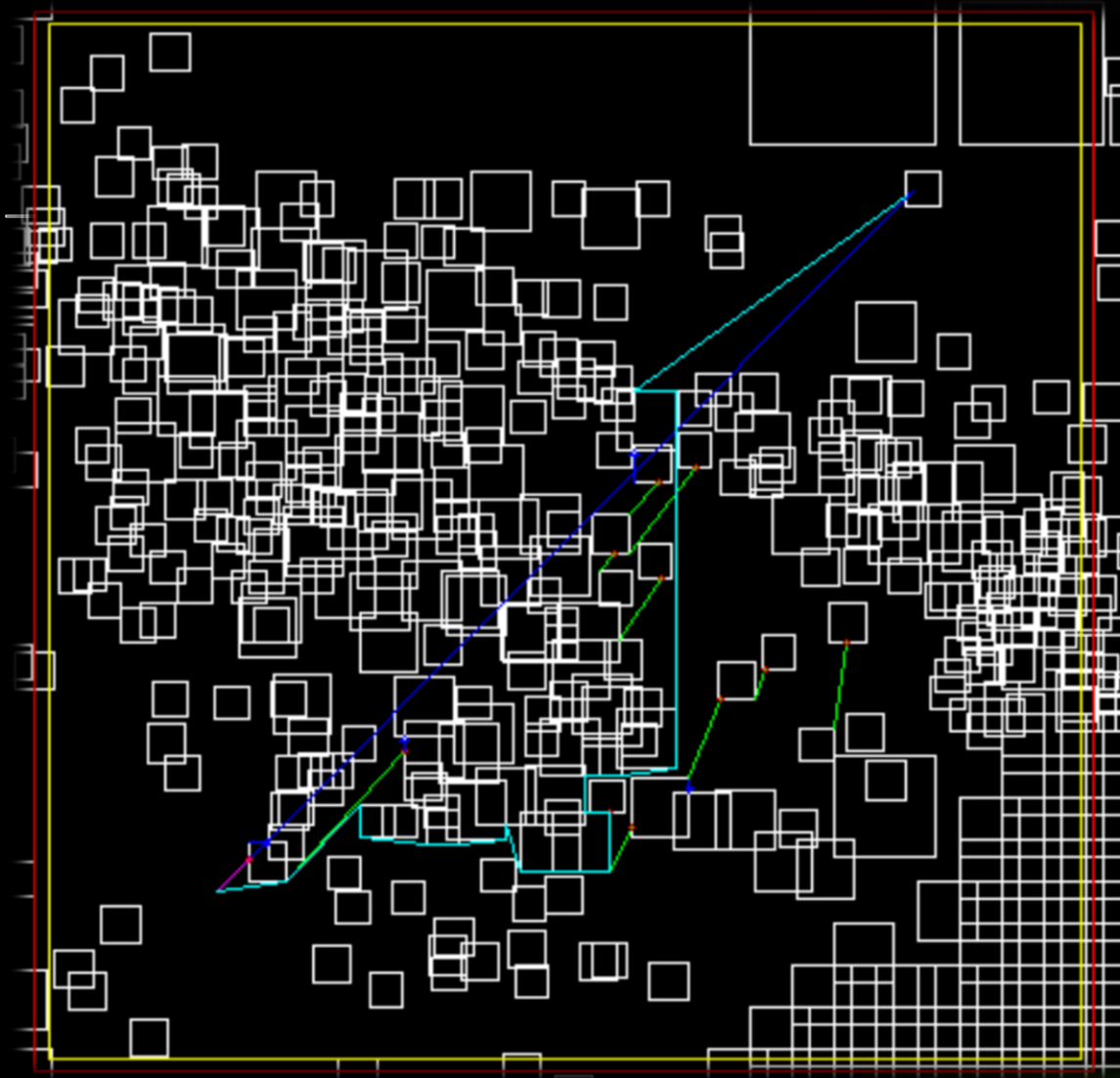
Solutions: Worst case?

One neat trick is to count turns:

- 4 more rights = we are in the open still
- 4 more lefts = we are in a hole

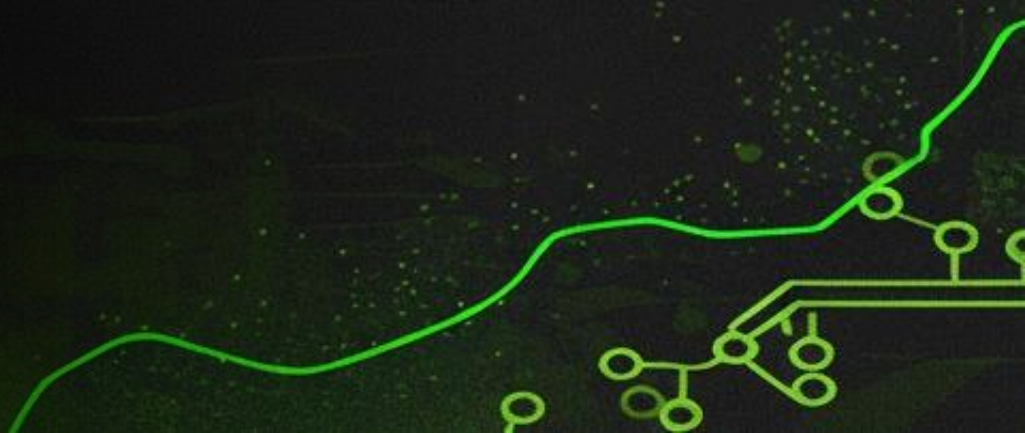


Solutions: New Worst Case?



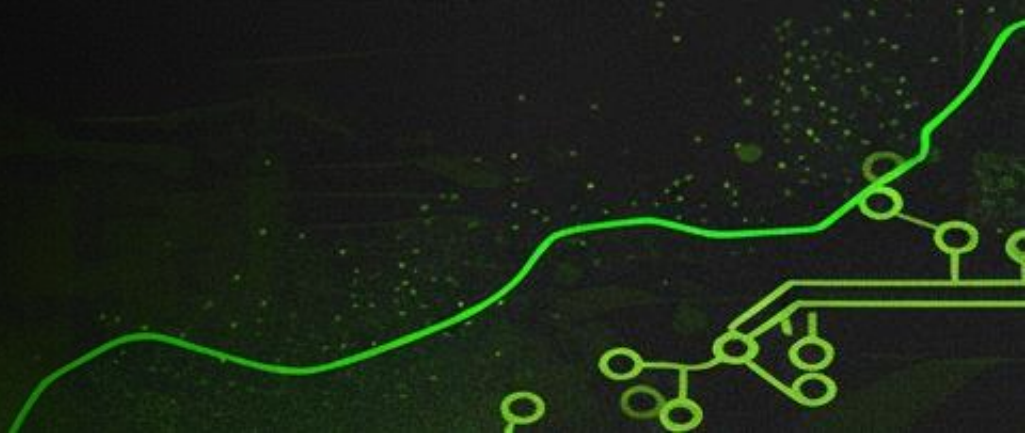


Results: Performance

- Average run 2x more performant
 - Old worst case 40x more performant
 - New 'worst' case 10% faster than original performance
- 



Results: Robustness

- Self verifiable algorithm
 - Guarantees a valid answer
 - Backed by an evolving test suite of over 100 test cases
- 

Results: Community Feedback



r/aoe2 • 1 day ago
kampalolo

New pathing after patch

Poll

Open • 141 total votes

18 perfect

97 better but not perfect ✓

17 same

9 worse

Results: Community Feedback



r/aoe2 • 1 day ago
kampalolo

New pathing after patch

Poll

Open • 141 total votes

18 perfect

97 better but not perfect ✓

17 same

9 worse



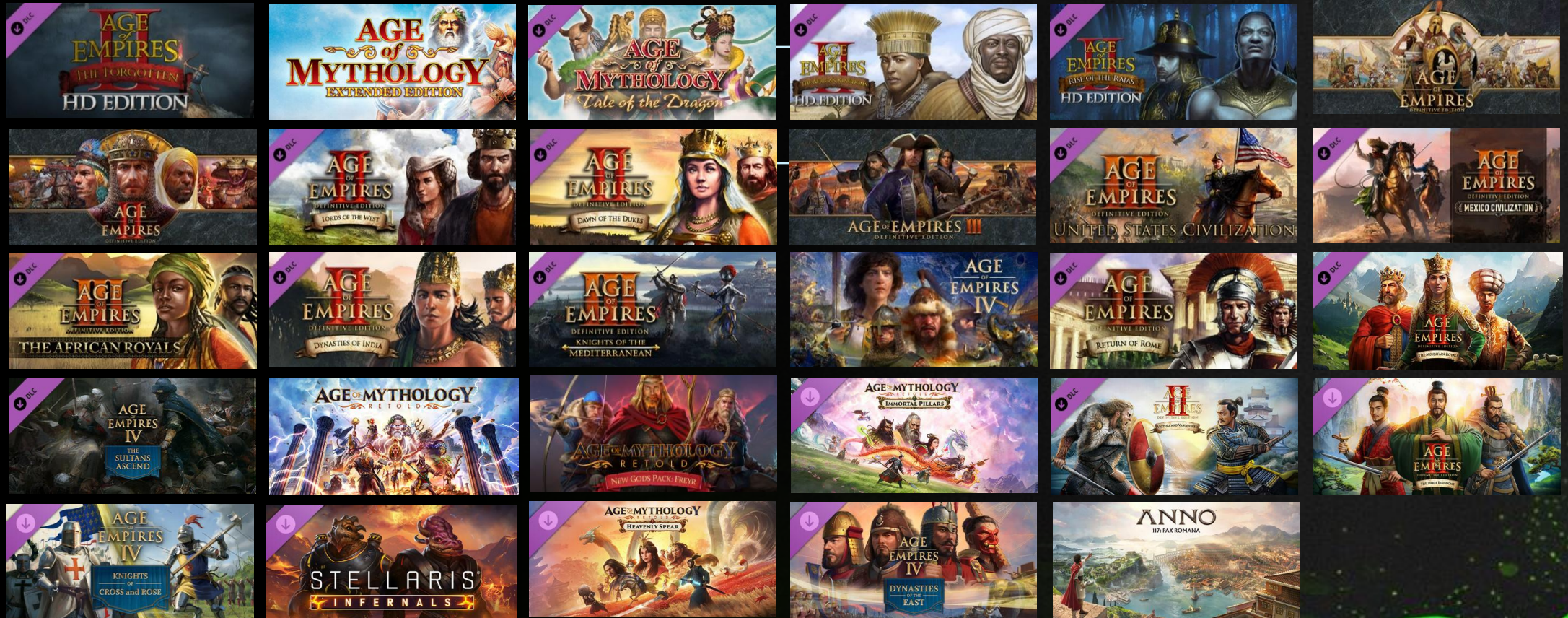


FORGOTTEN EMPIRES

It has been a long road

FORGOTTEN EMPIRES

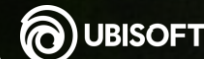
Specialized in Strategy Games since 2013 - Shipped 7 main game and 20+ DLC releases



Our clients:



Microsoft





FORGOTTEN EMPIRES

It has been a long road

Any questions?

