Efficient Algorithms for Battleship

FUN with algorithms 2020, *June, 1st 2022, Favignana, Italy*

Yan Gerard

with Loïc Crombez and Guilherme Da Fonseca

Plan

I — The Game / Shooting Algorithms

II — Examples

III — Results

Plan

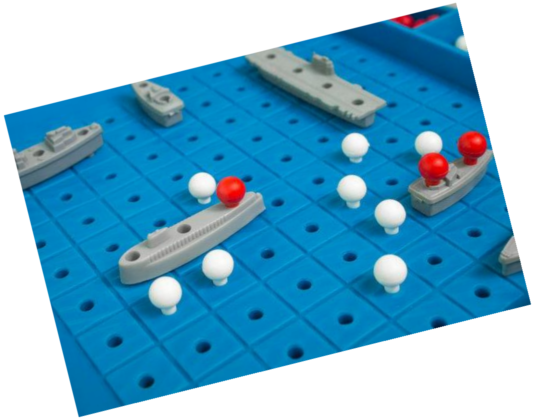
I The Game / Shooting Algorithms

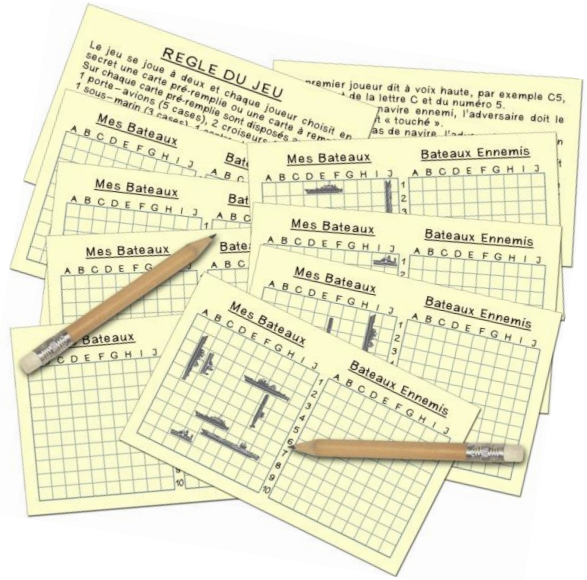
II Examples

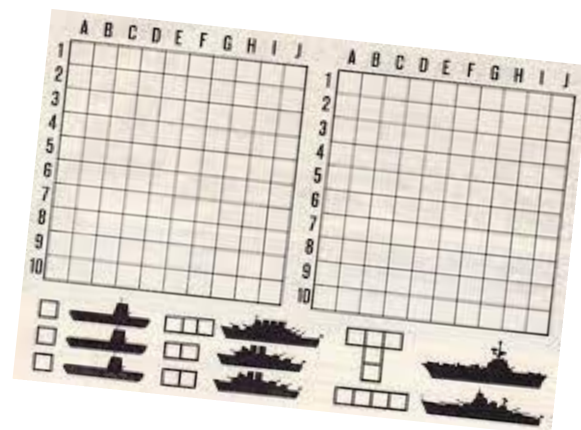
III Results

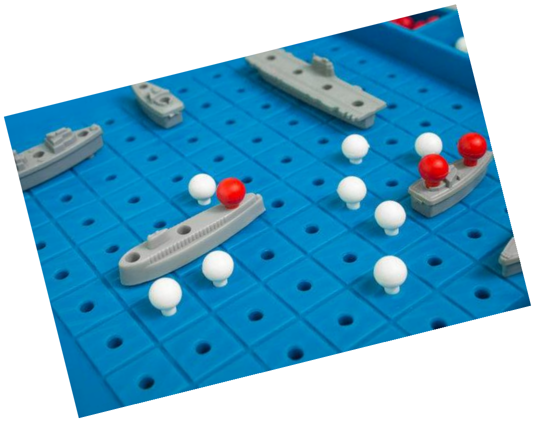
I Problem statement

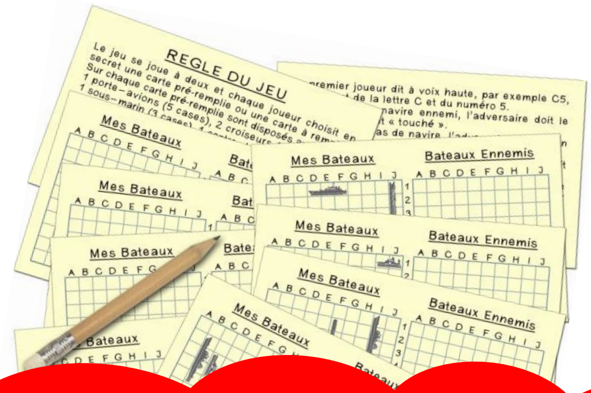
A classical game



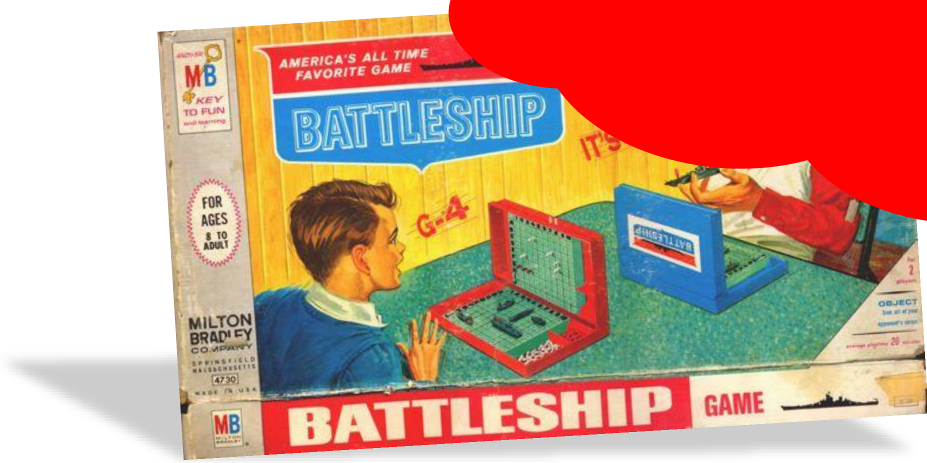


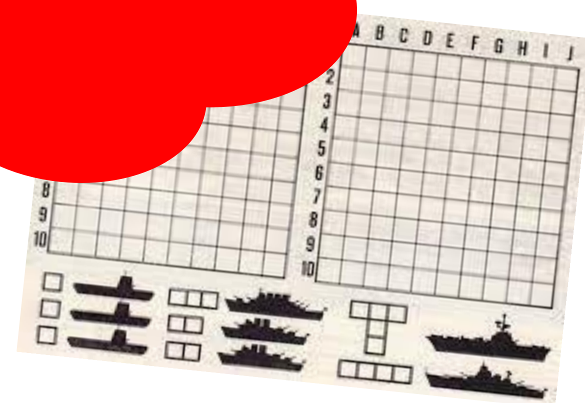




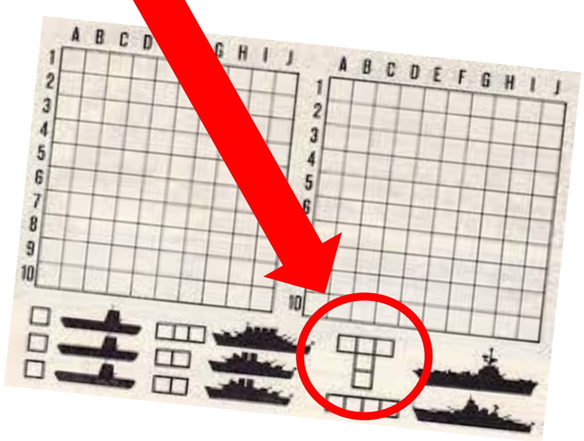


Let’s consider a simplified variant of the game





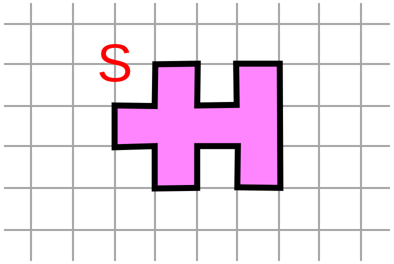
No restriction on the shapes of the ships to segments….
We play with polyominoes or more generally with any given digital shape!



First player: Alice

Second player: Together

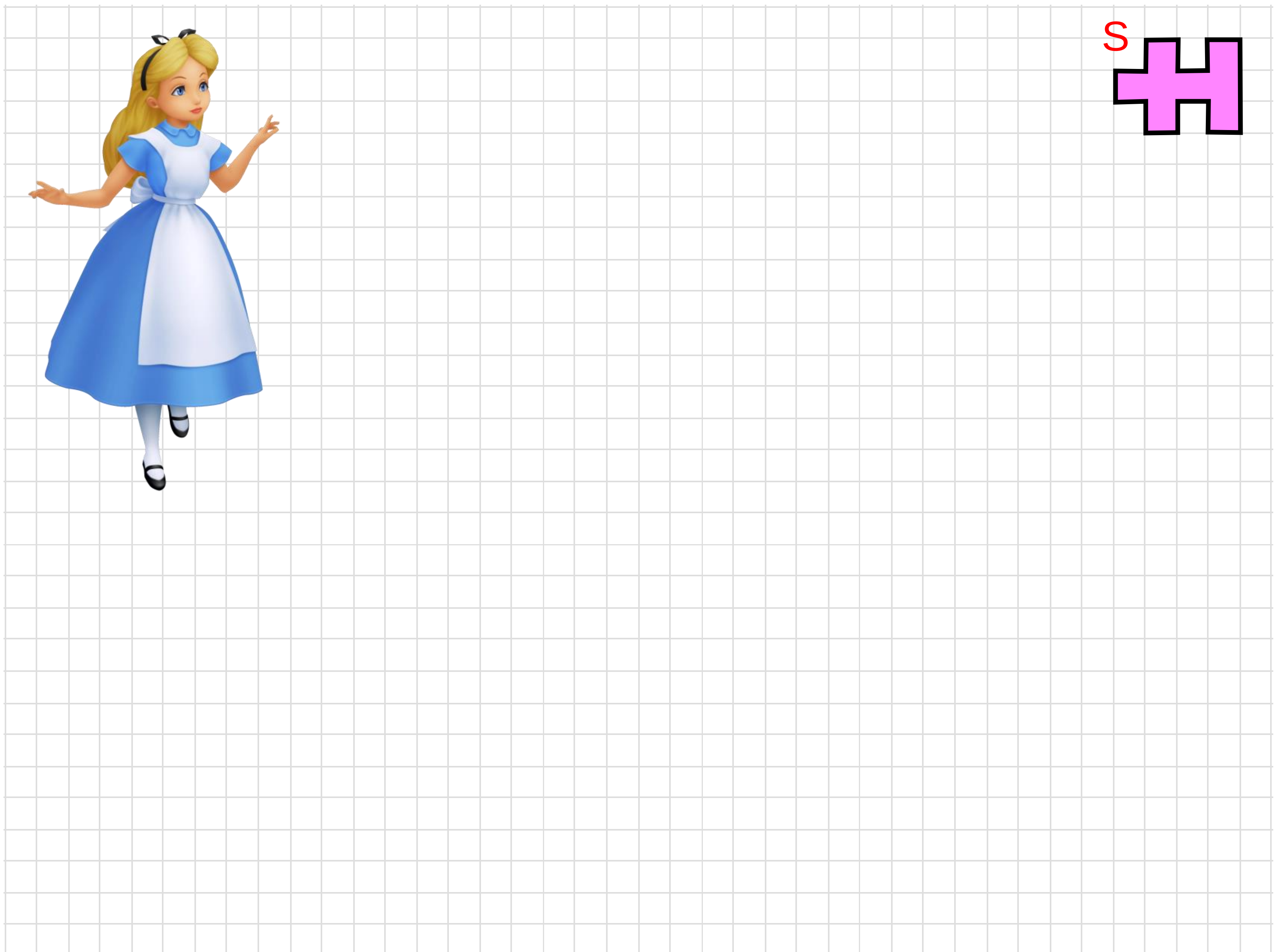
First player: Alice

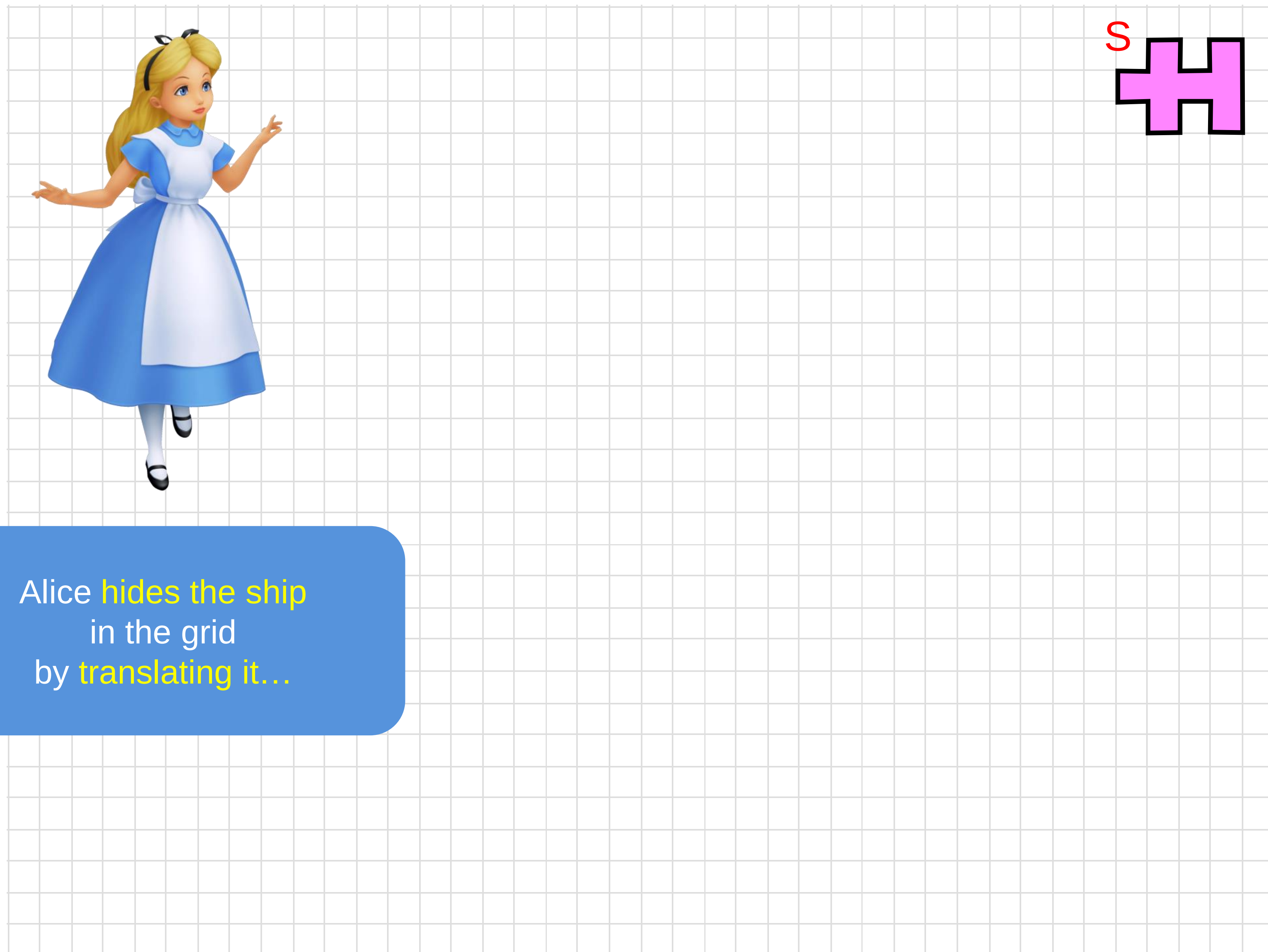


We choose together the shape of the ship…

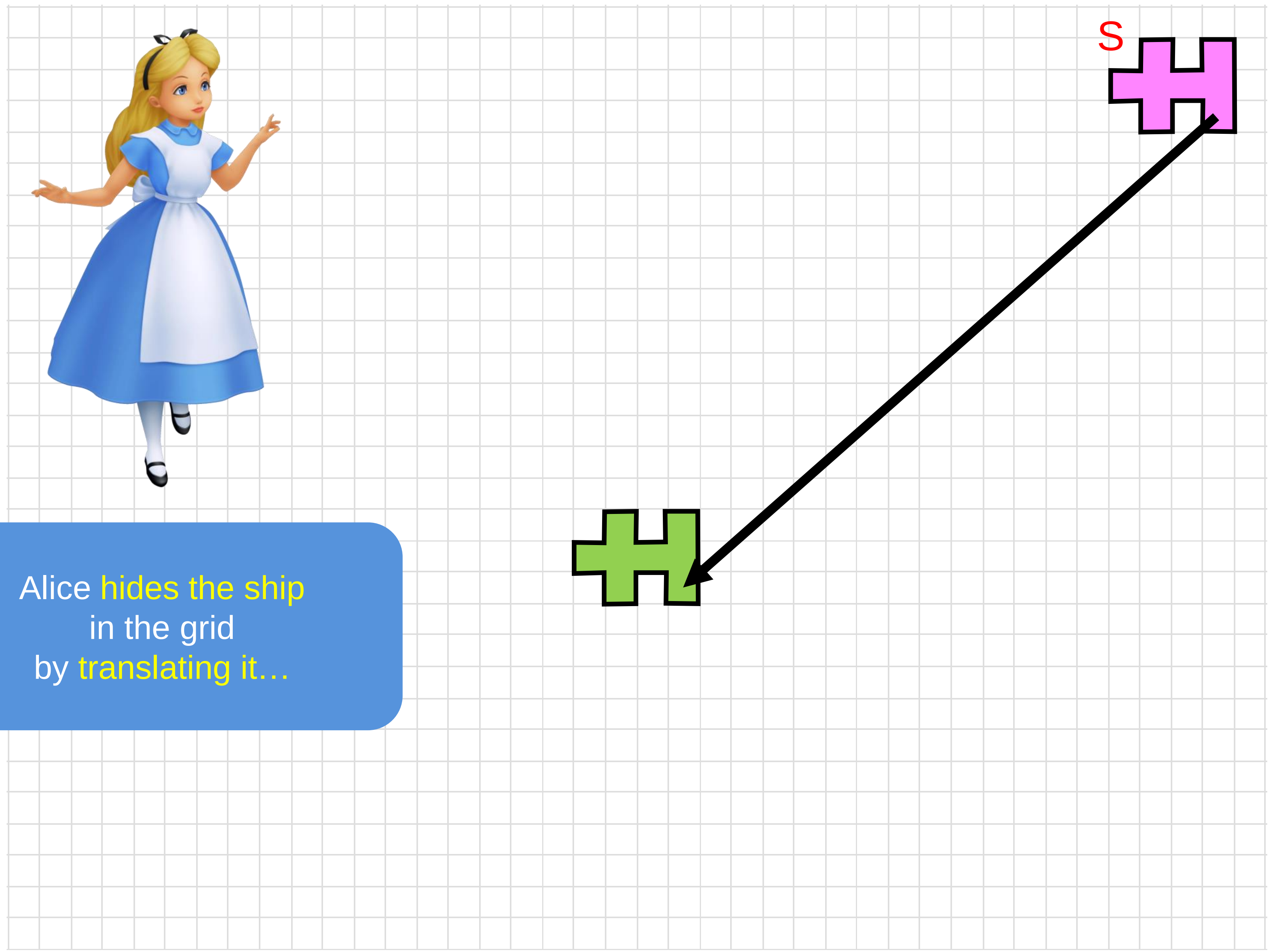
Second player: Together

The grid

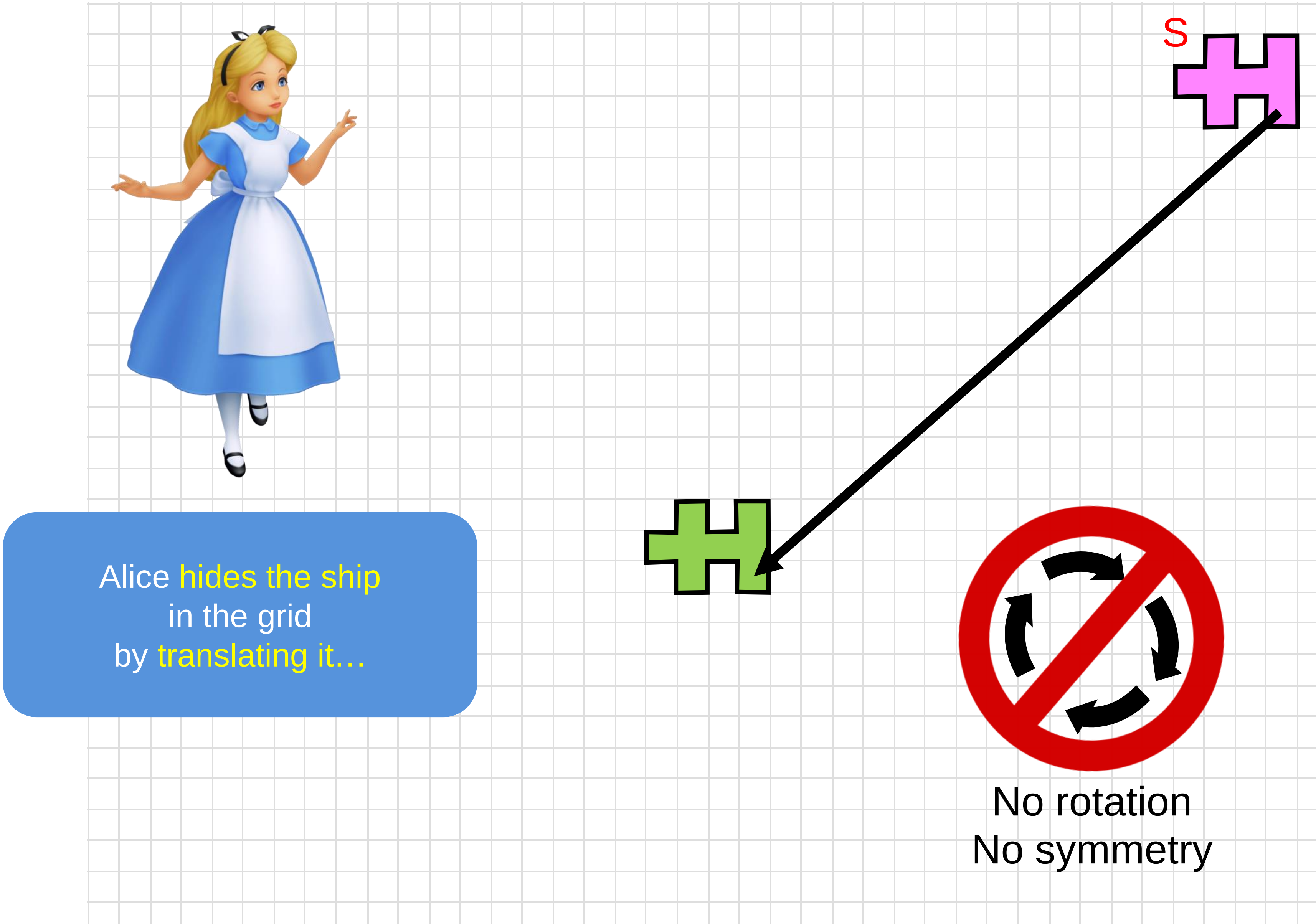


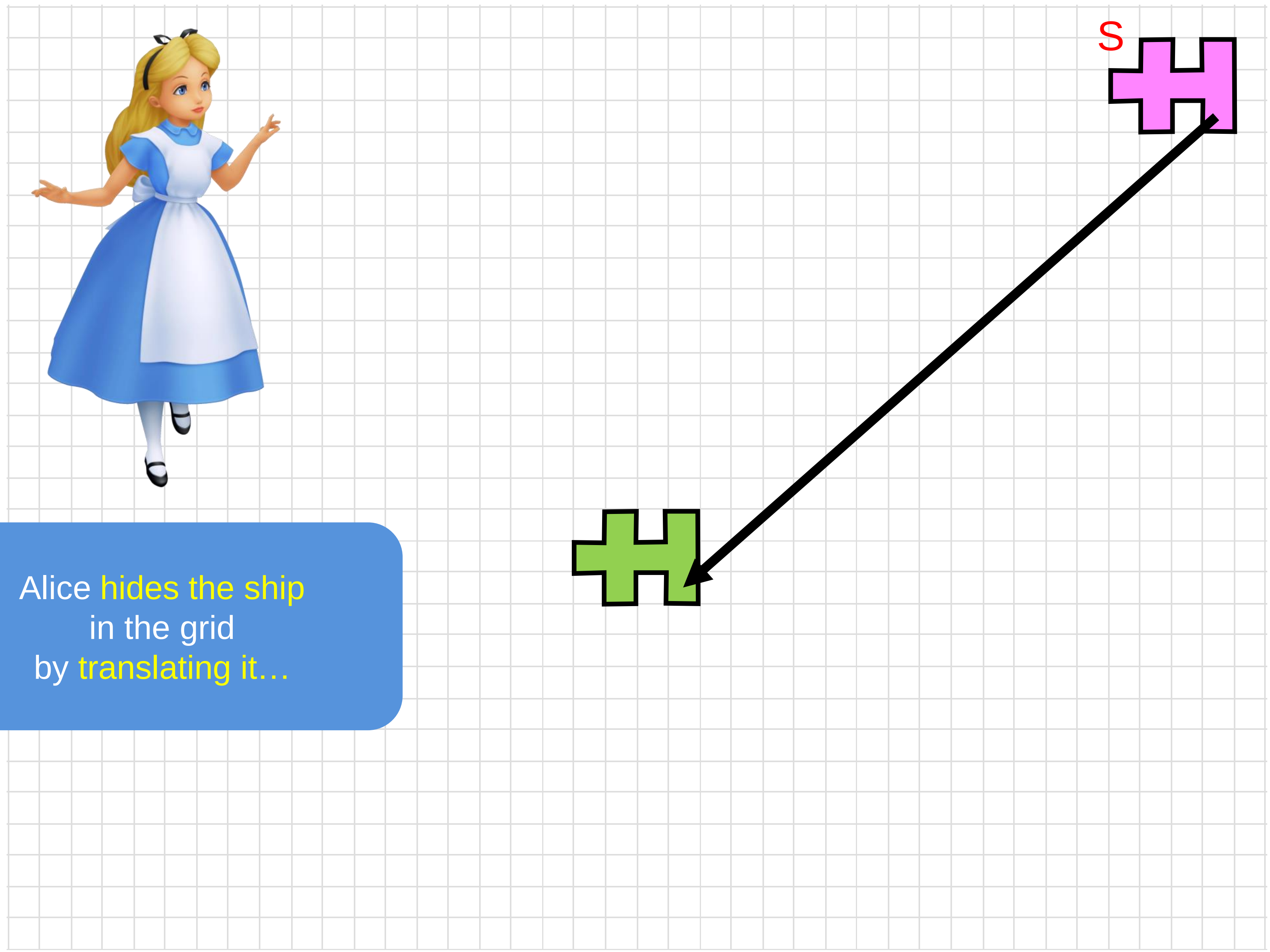


Alice hides the ship
in the grid
by translating it…

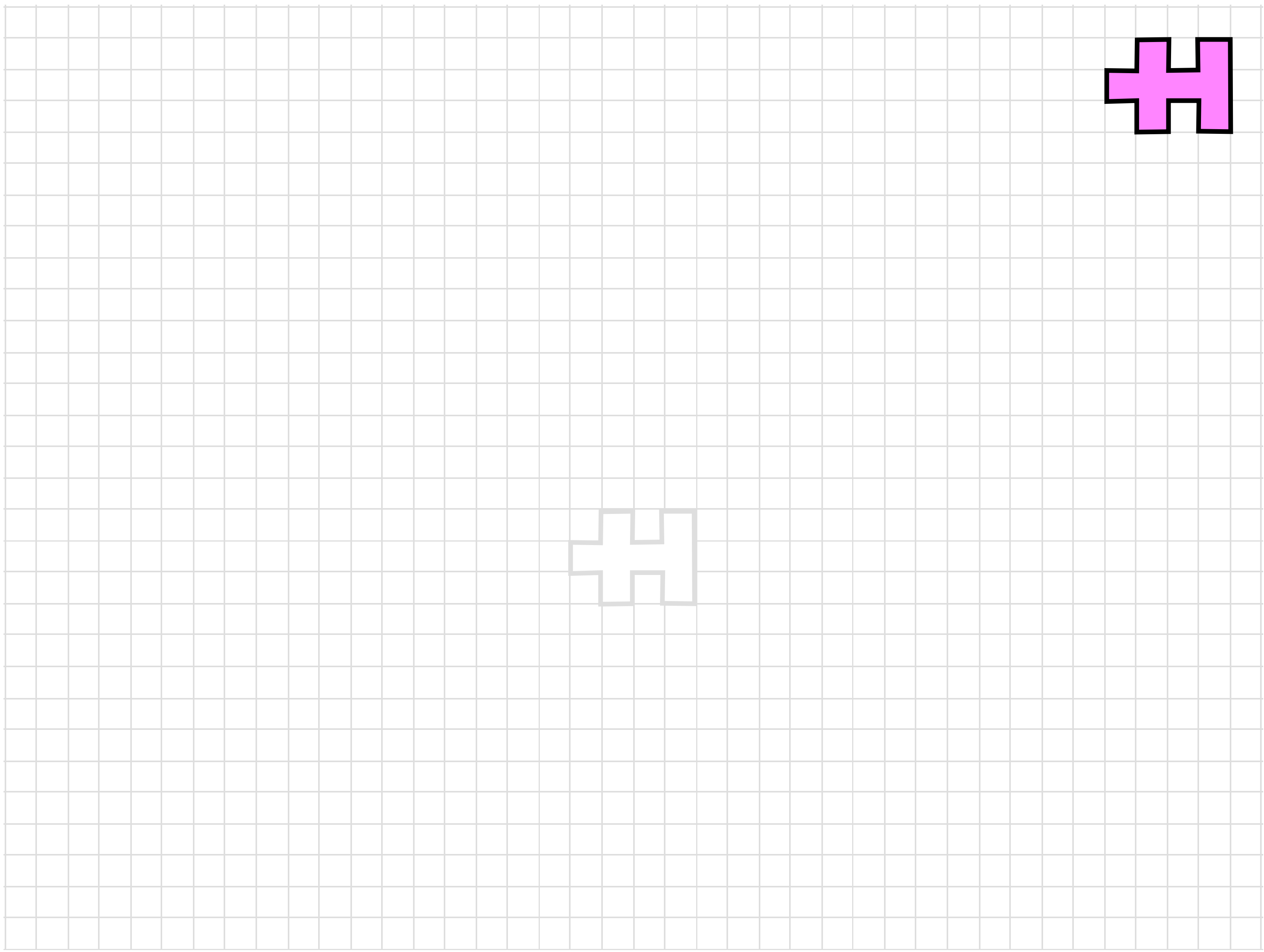


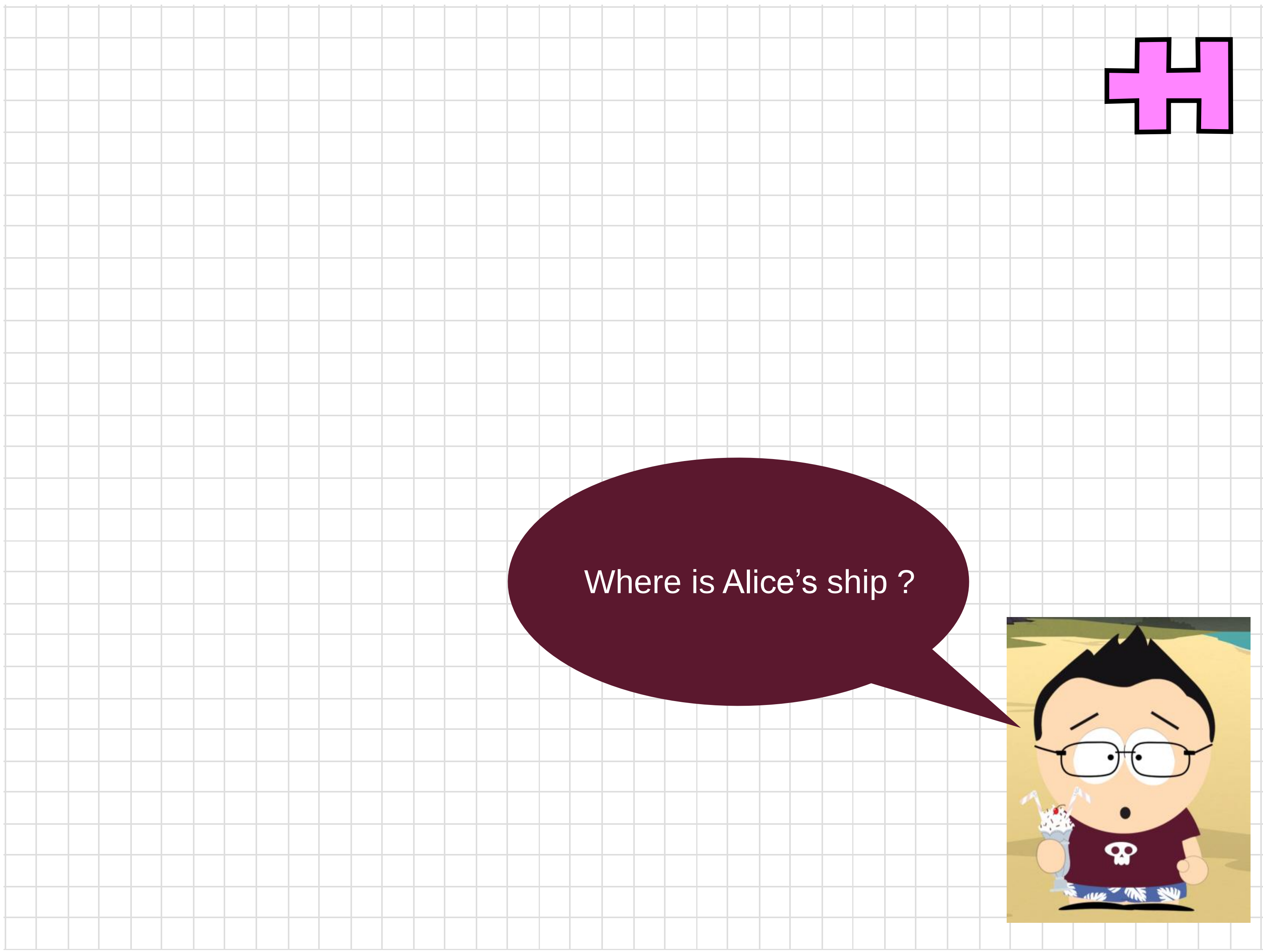
Alice hides the ship
in the grid
by translating it…

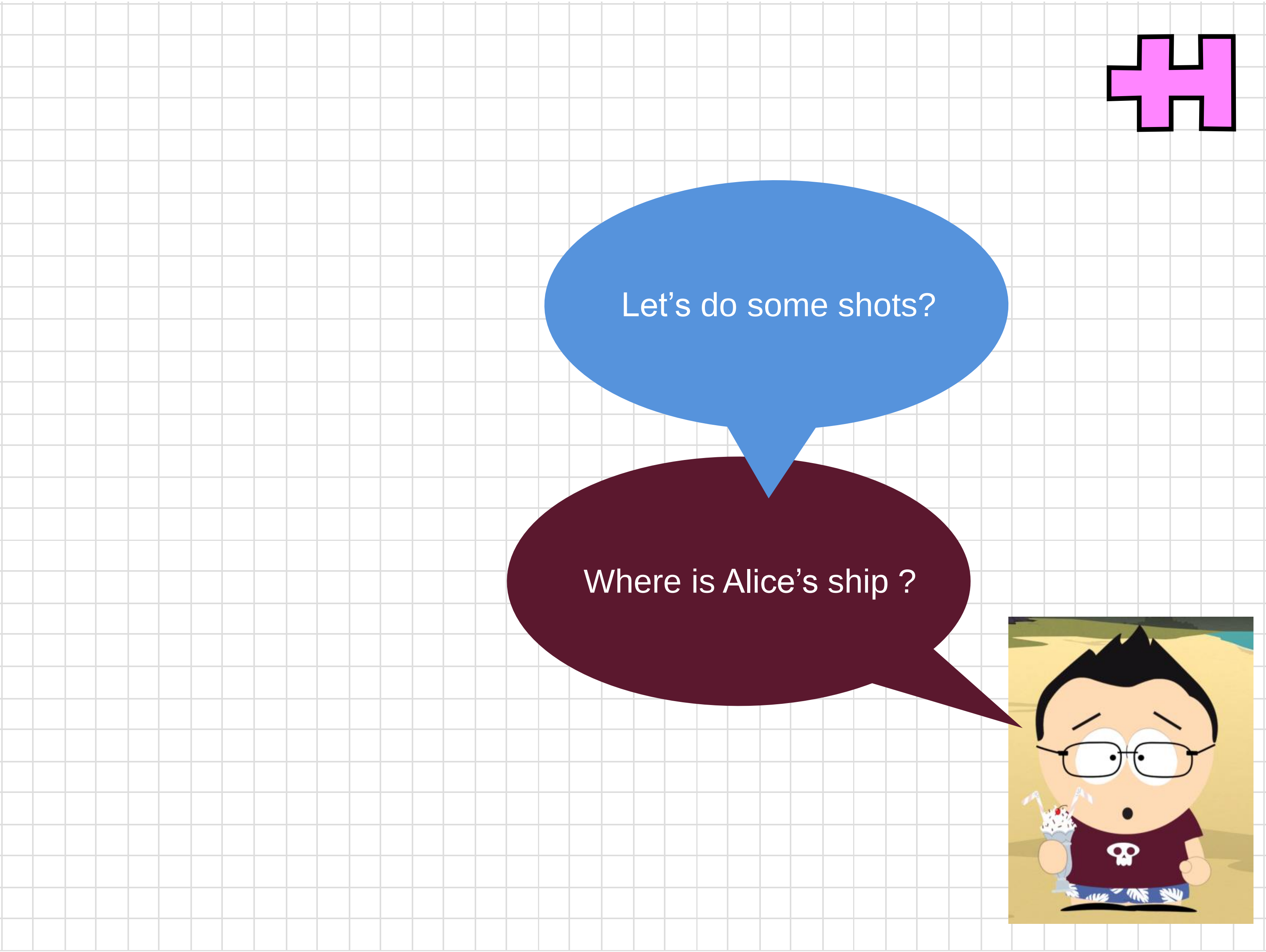

S
Alice hides the ship
in the grid
by translating it…
No rotation
No symmetry

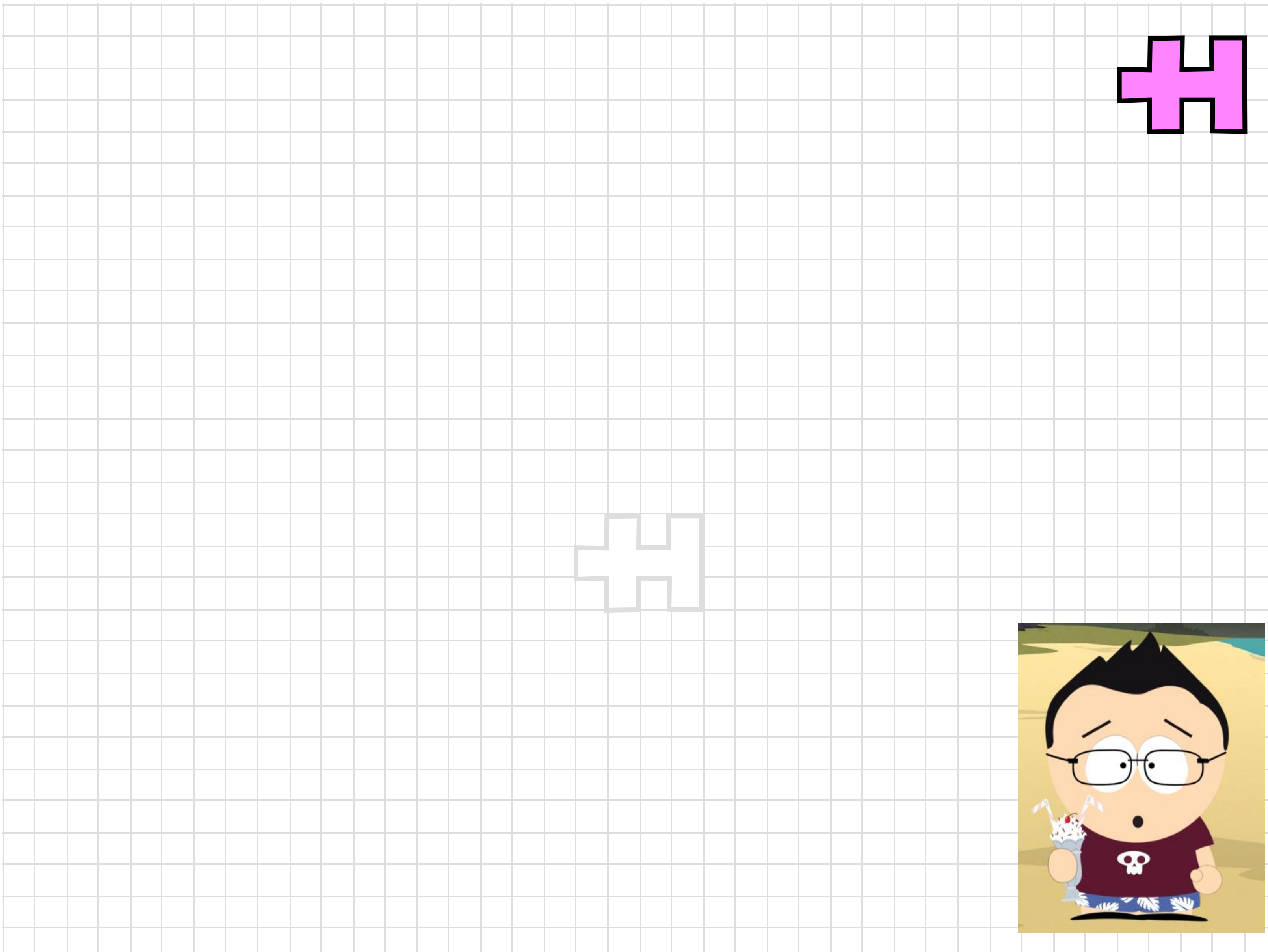


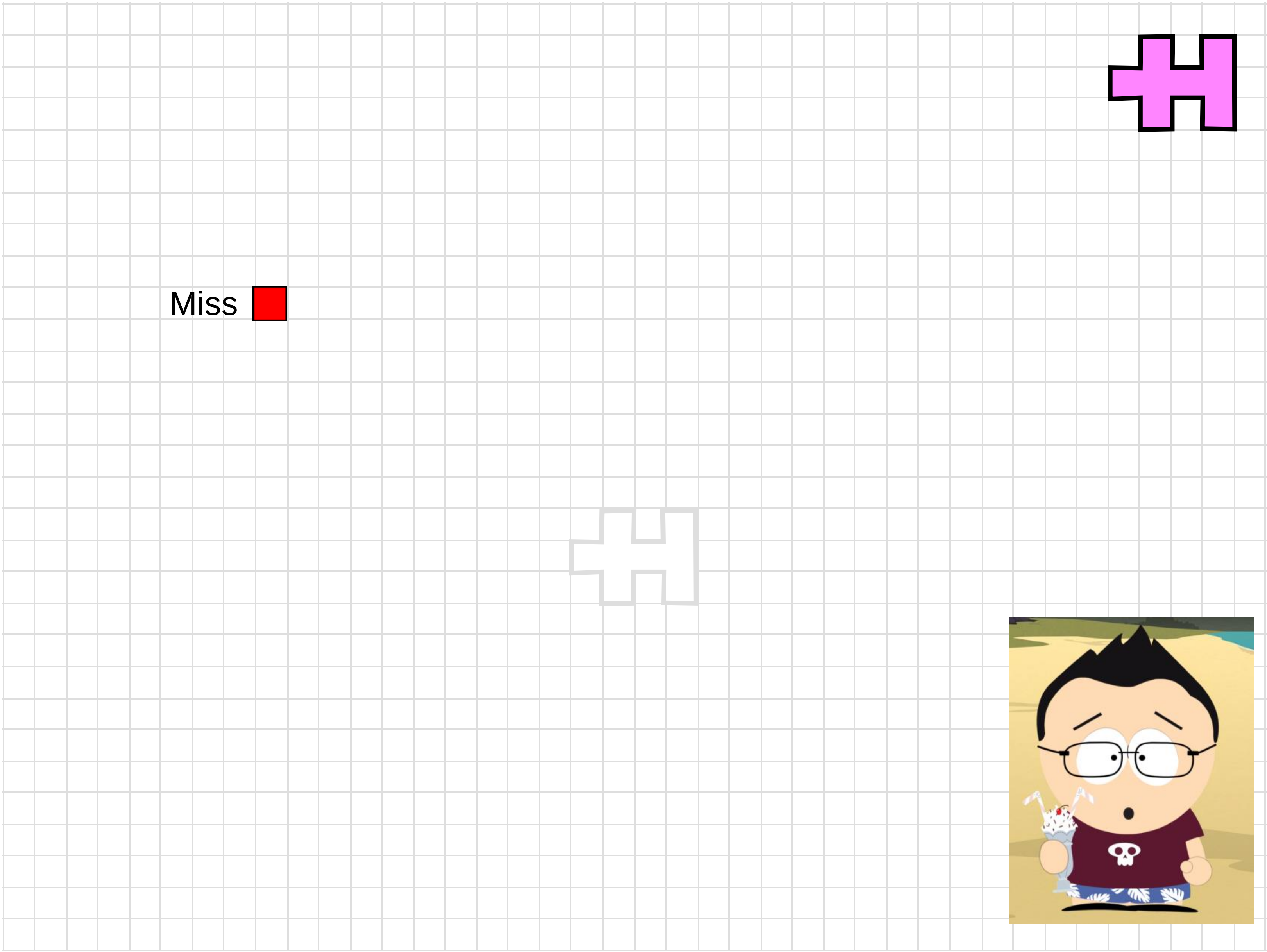
Alice hides the ship
in the grid
by translating it…

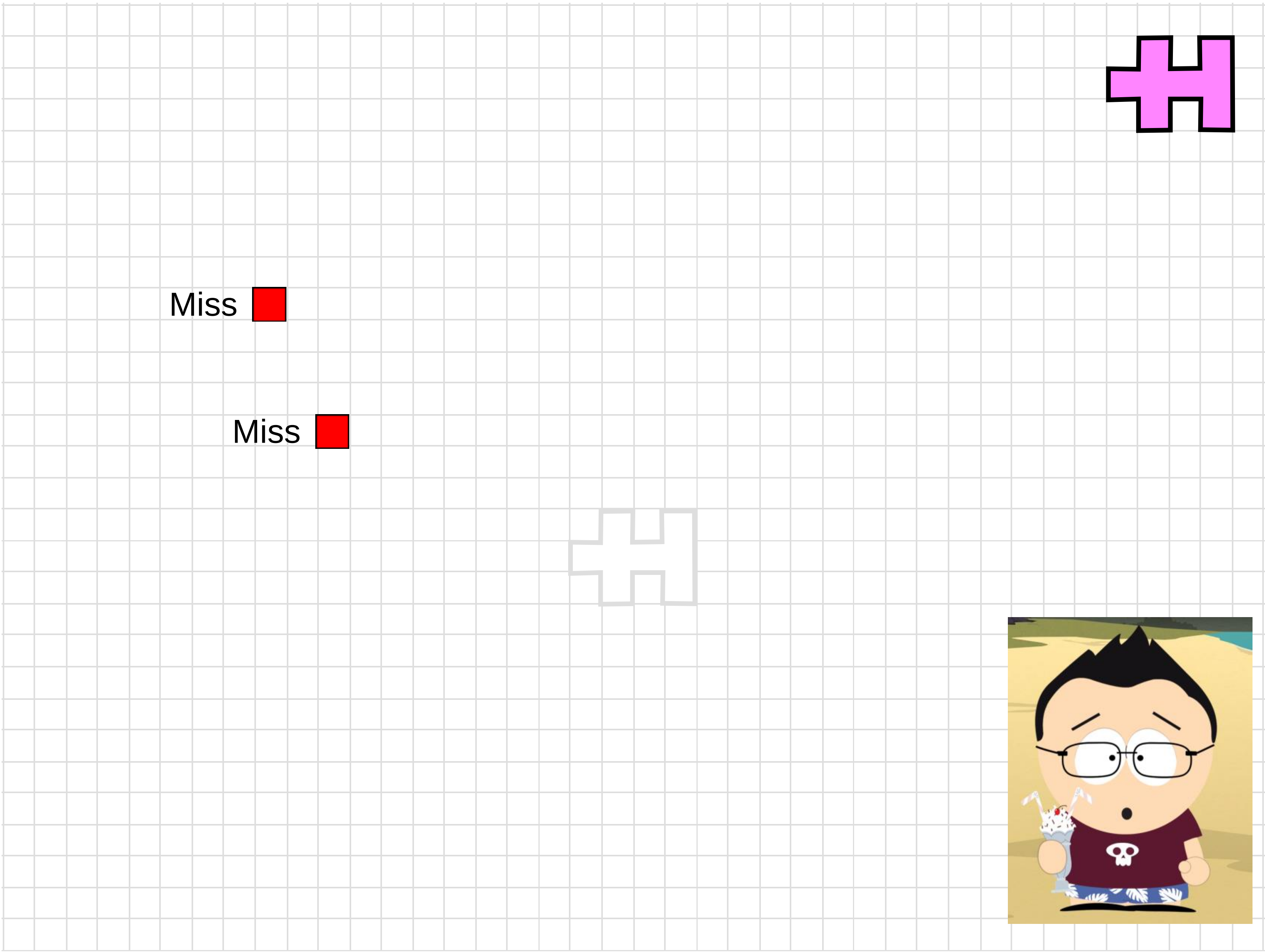


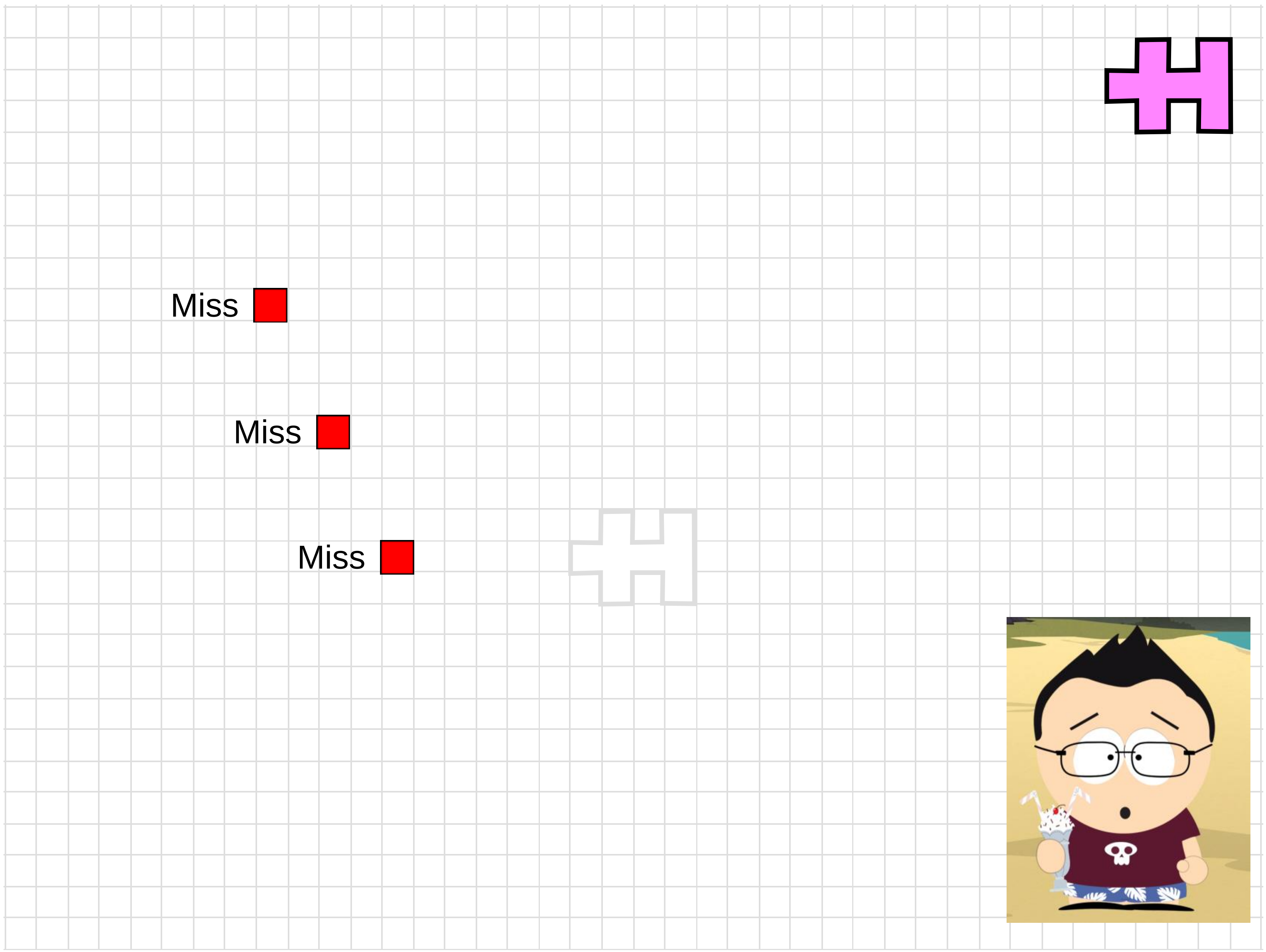

Where is Alice’s ship ?

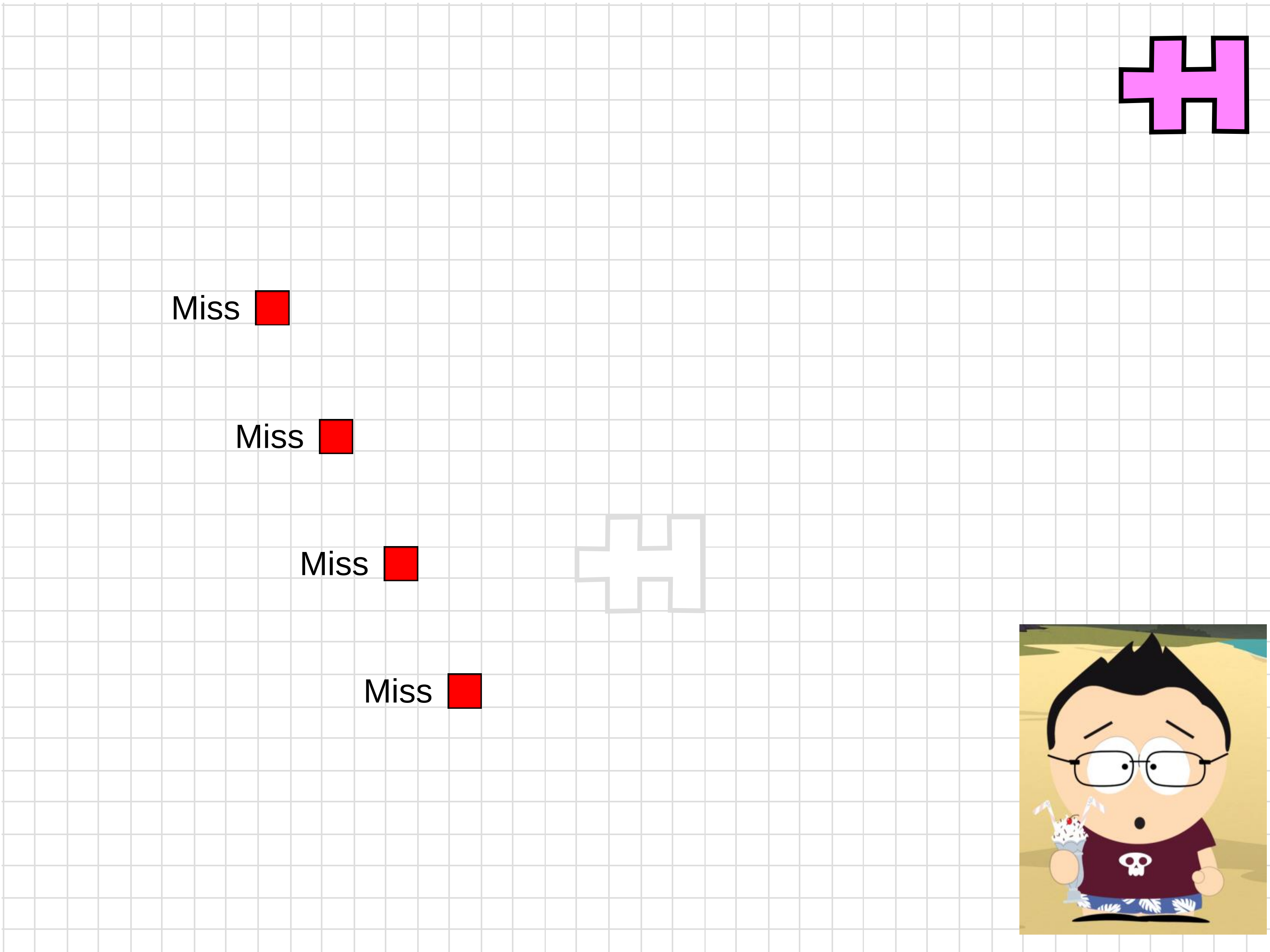

Let’s do some shots?
Where is Alice’s ship ?



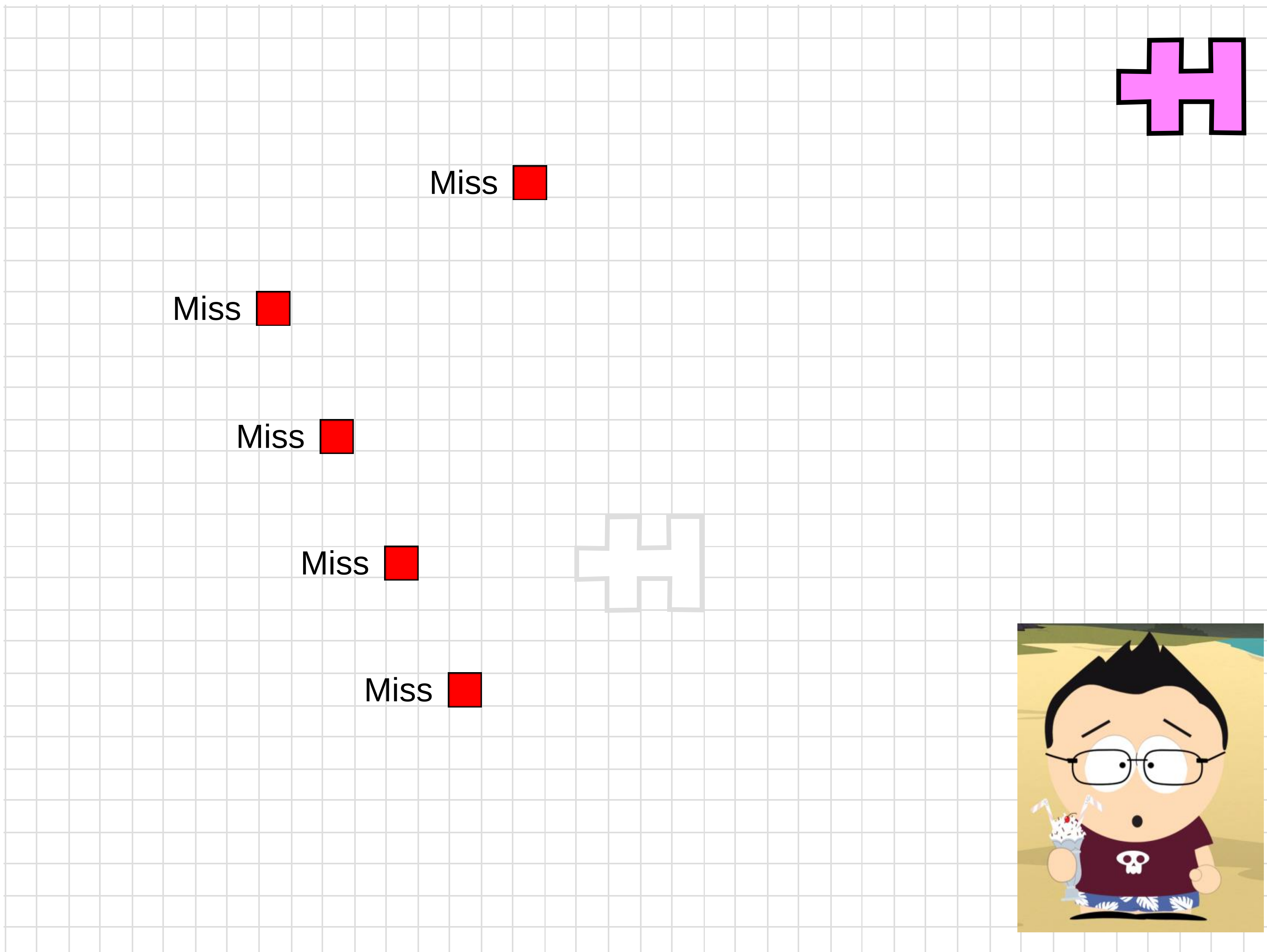

Miss

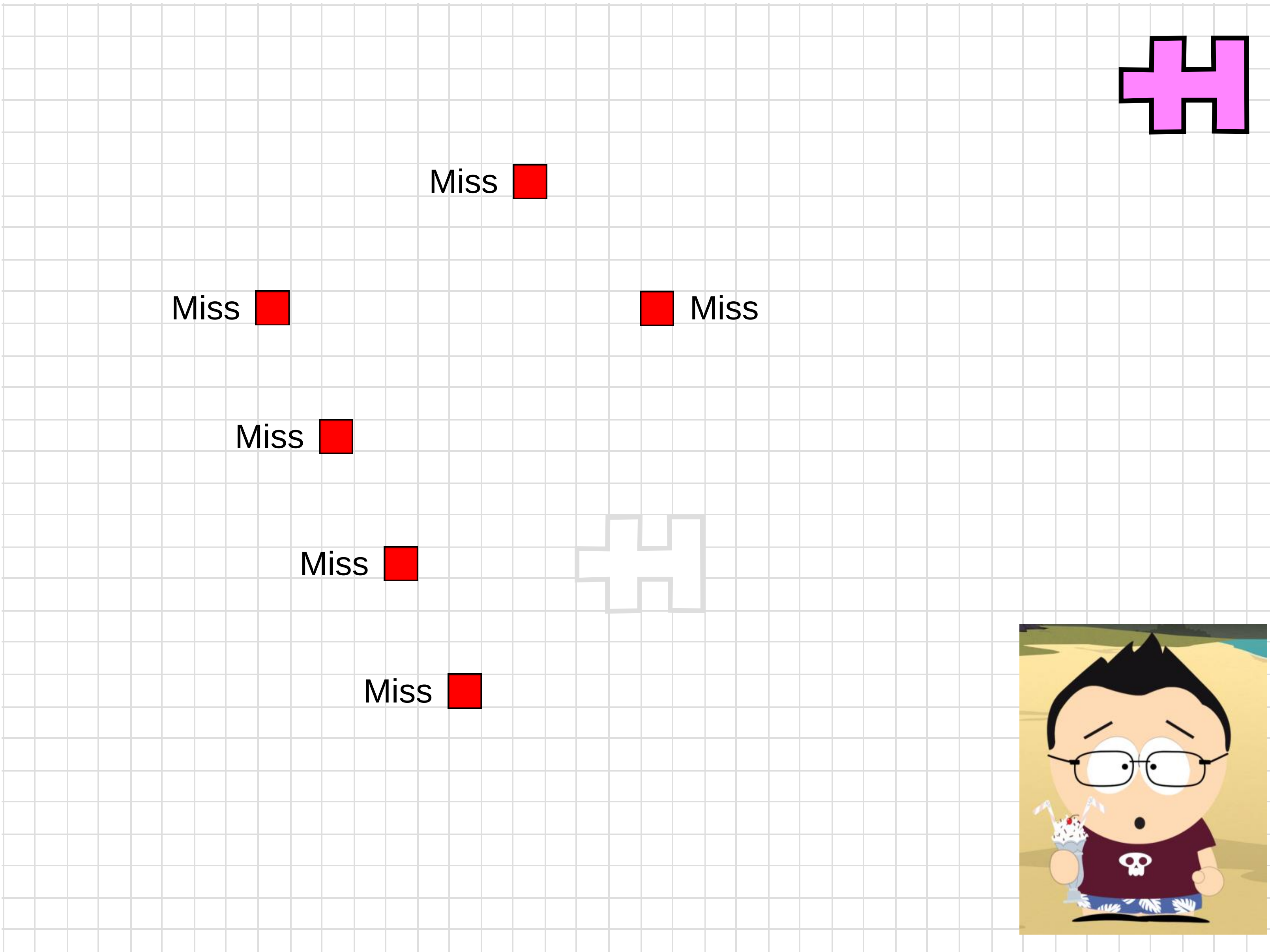

Miss
Miss

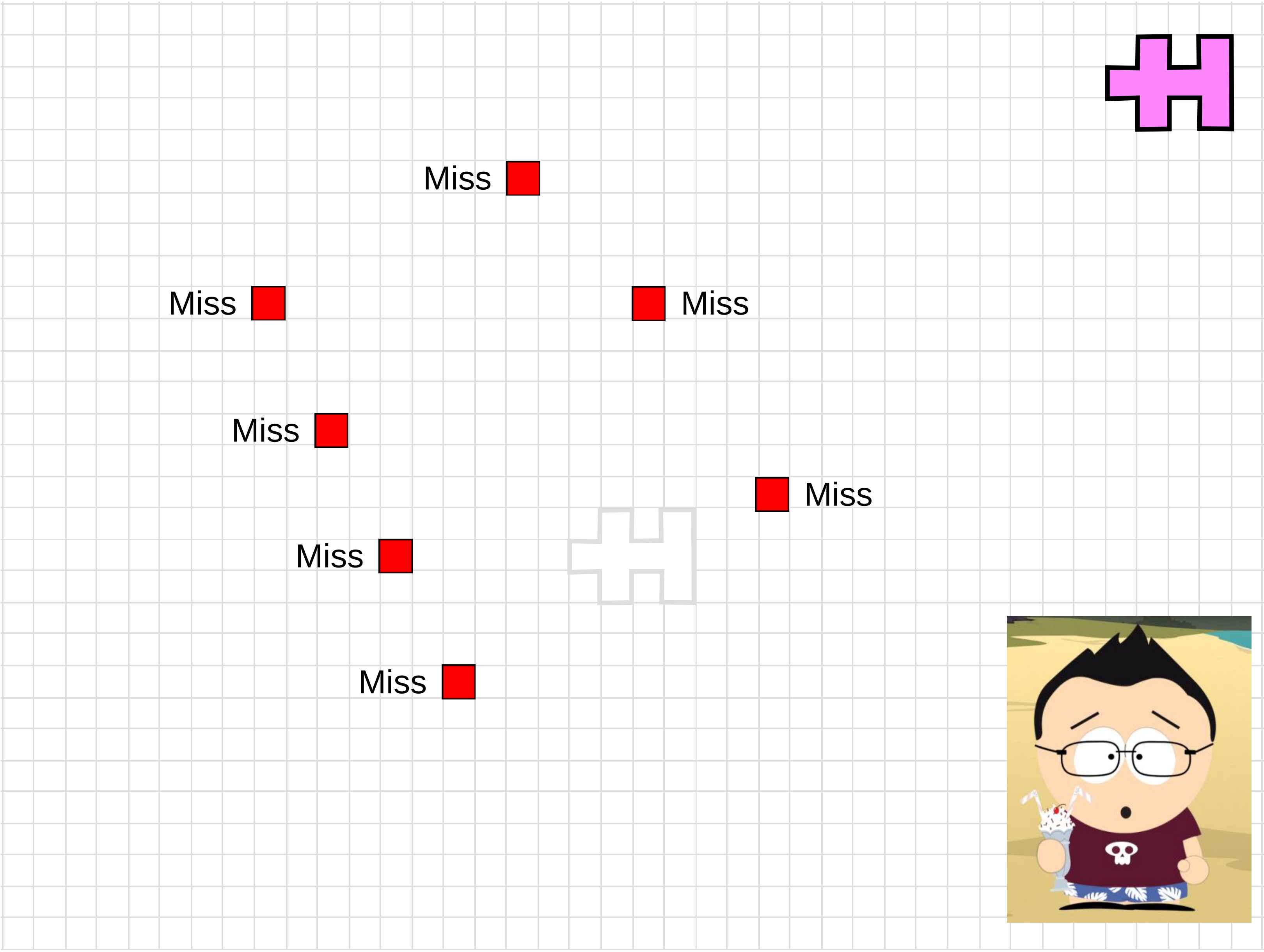

Miss
Miss
Miss

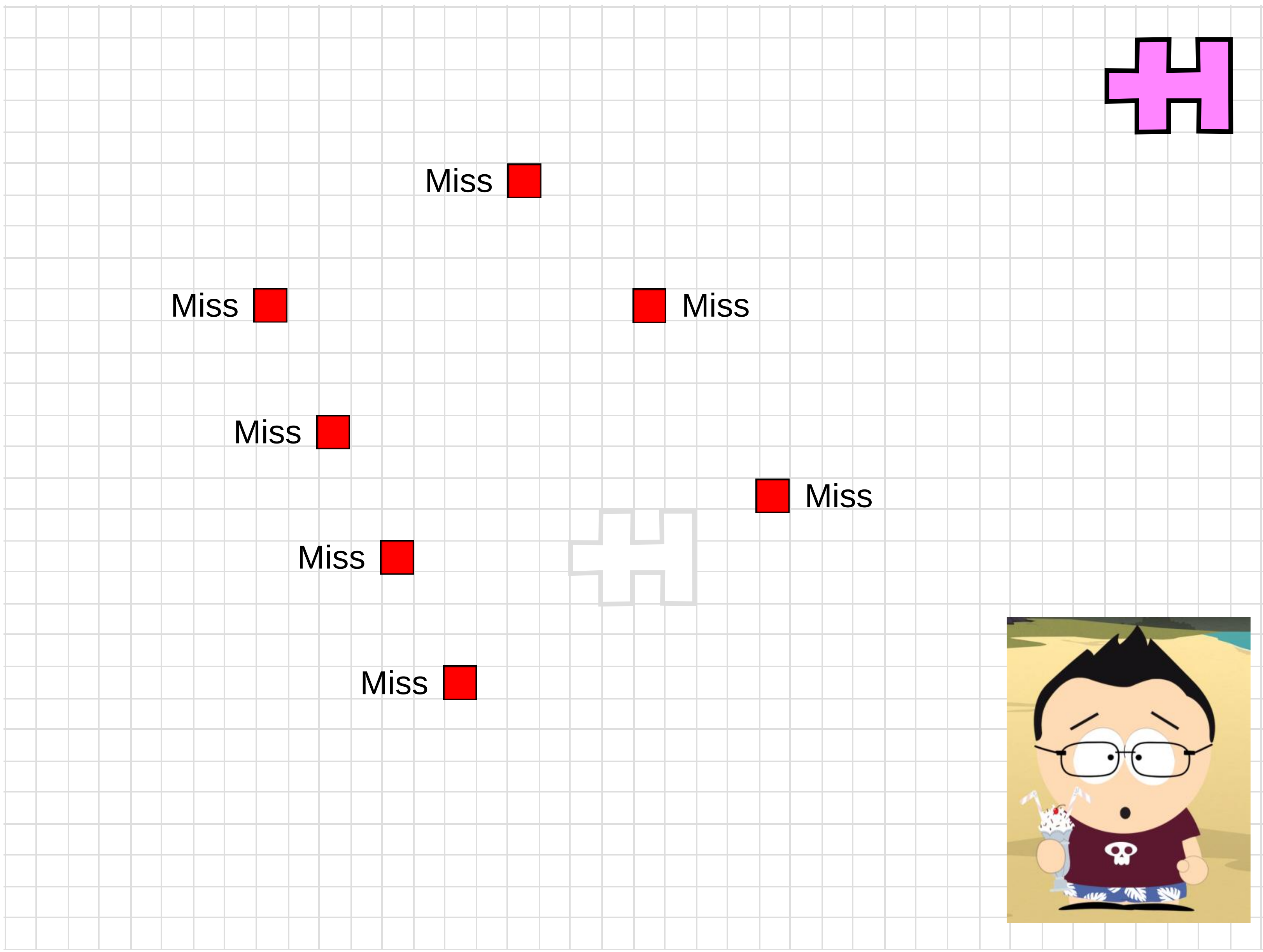

Miss
Miss
Miss
Miss

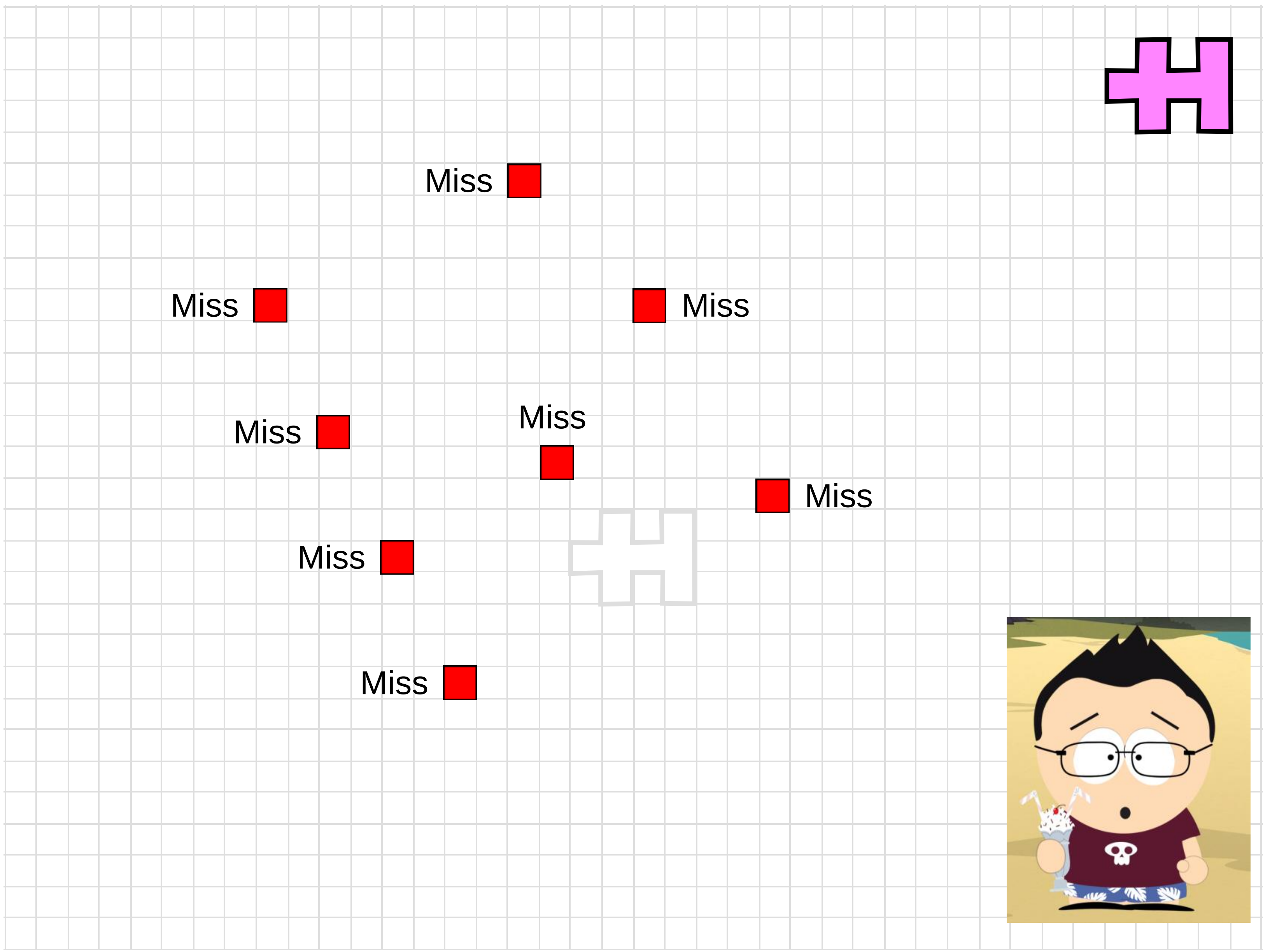
Problem statement

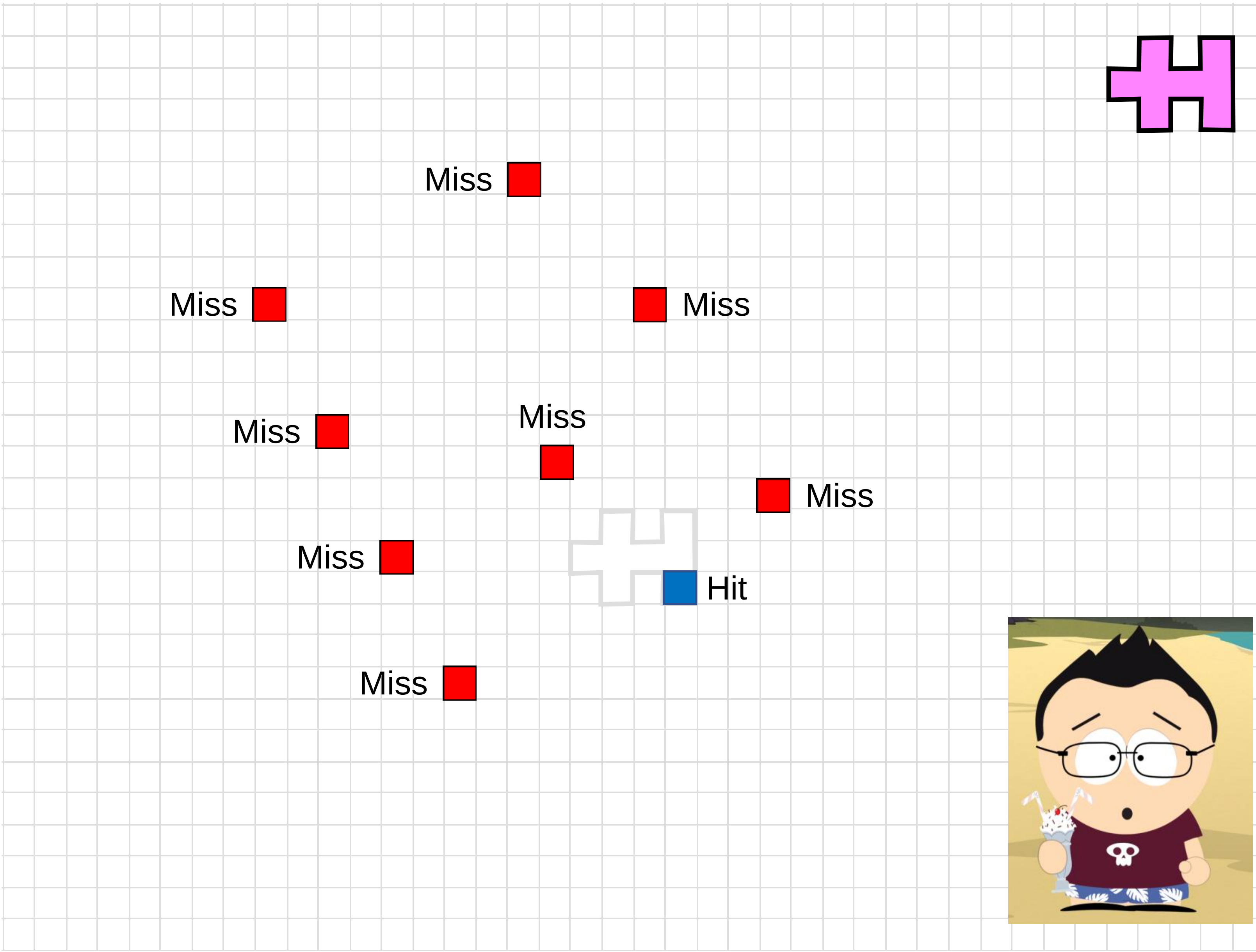



Miss
Miss
Miss
Miss
Miss
Miss

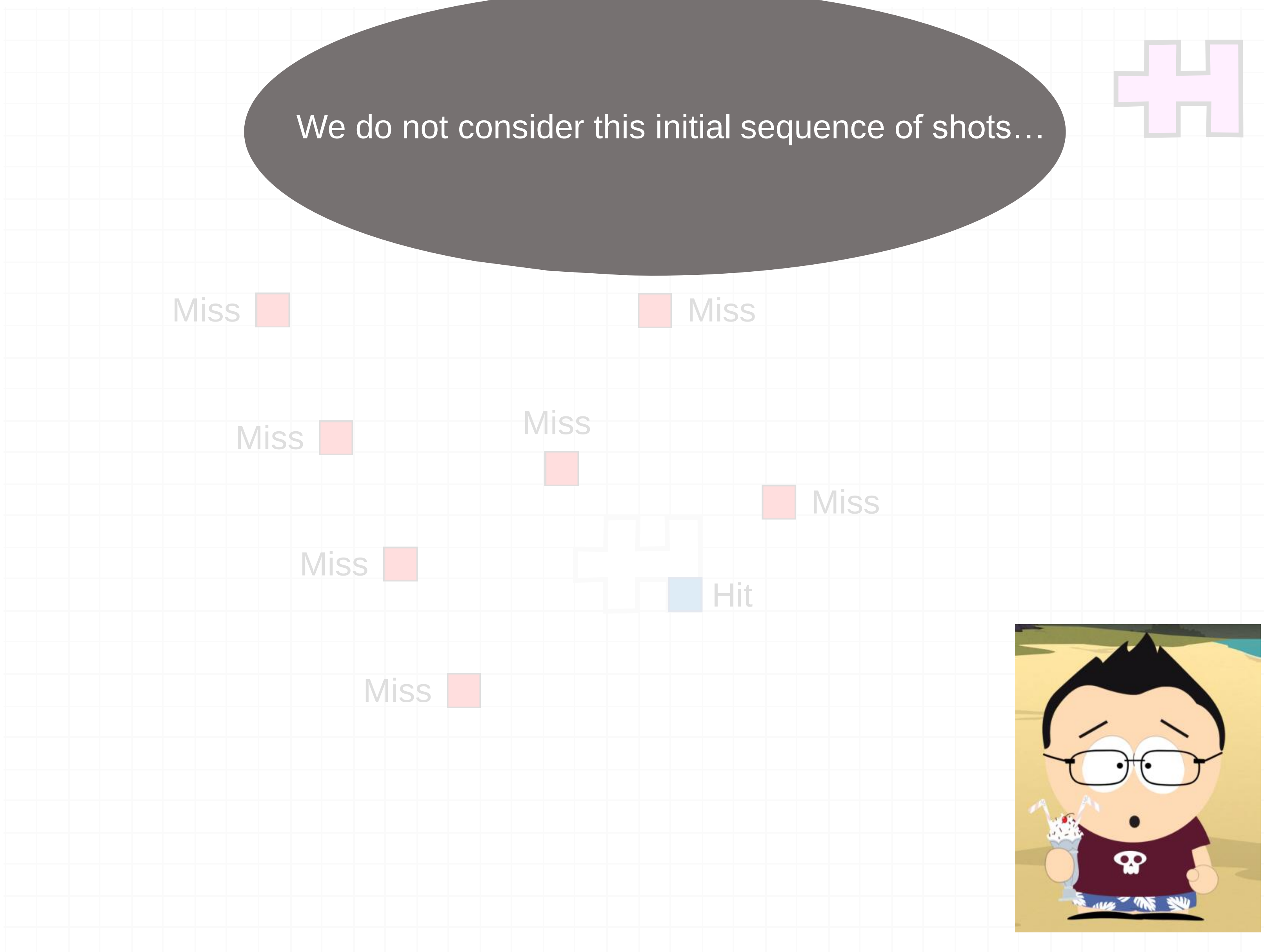

Miss
Miss
Miss
Miss
Miss
Miss
Miss

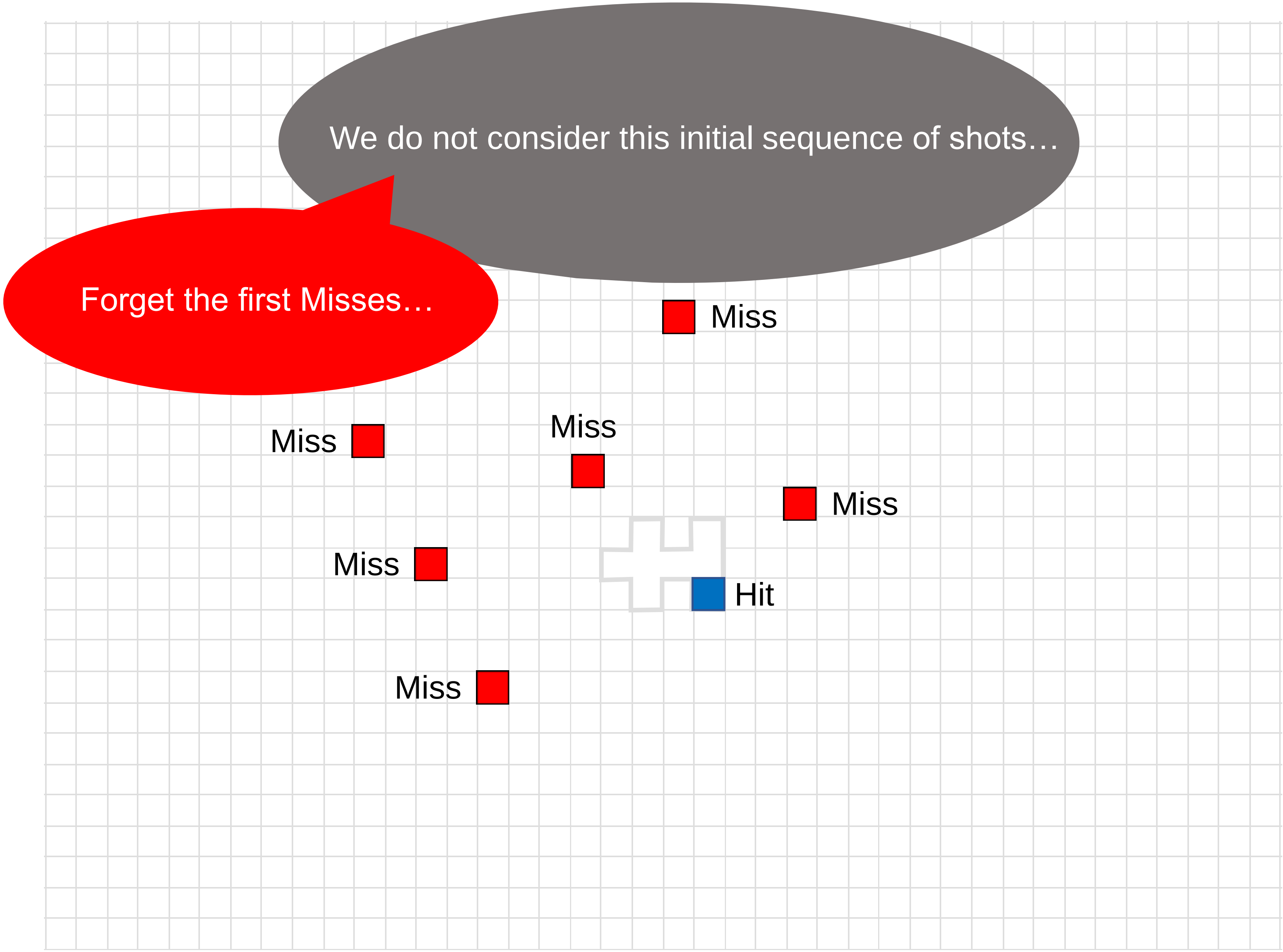

Miss
Miss
Miss
Miss
Miss
Miss
Miss

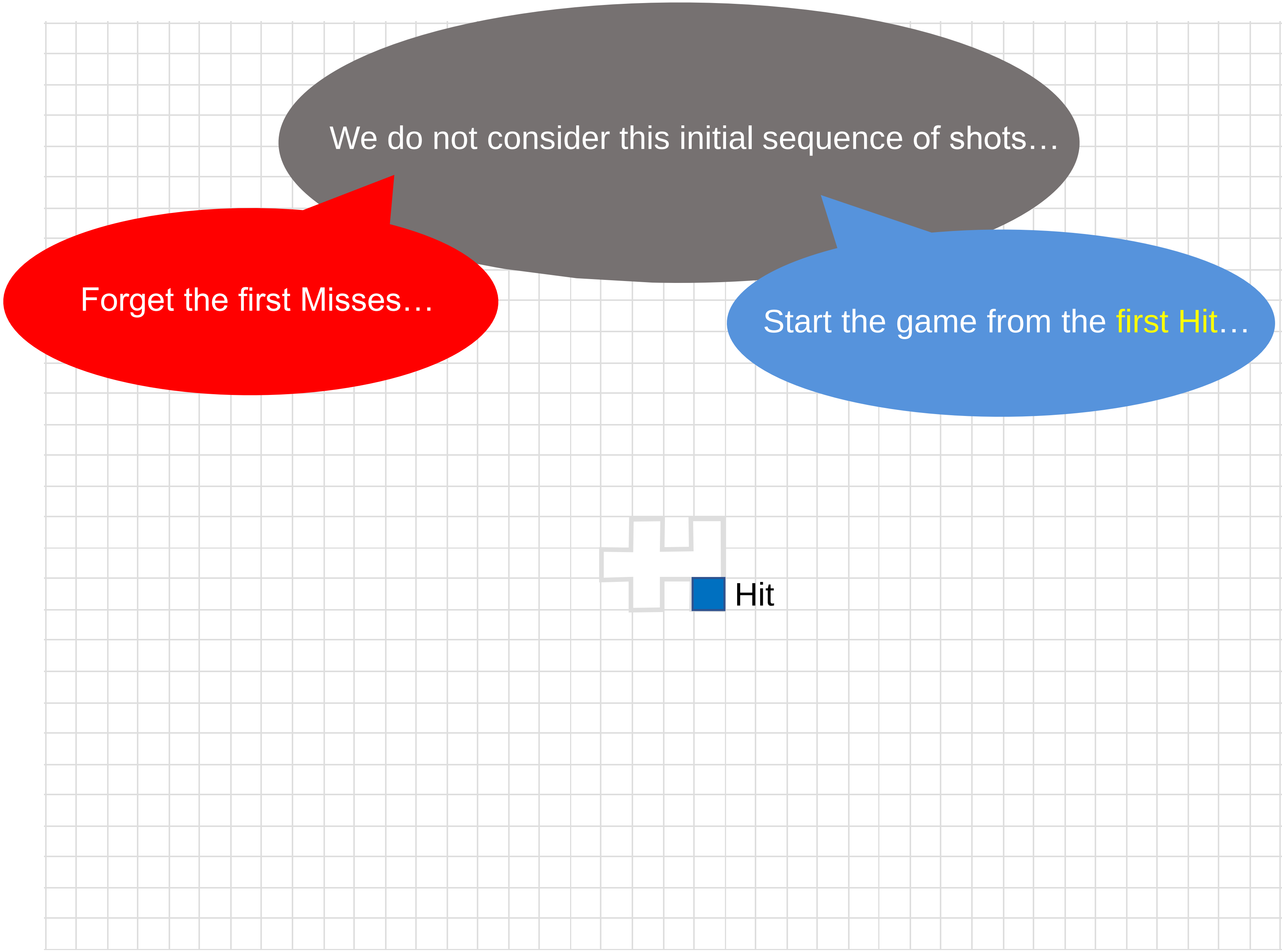

Miss
Miss
Miss
Miss
Miss
Miss
Miss
Miss

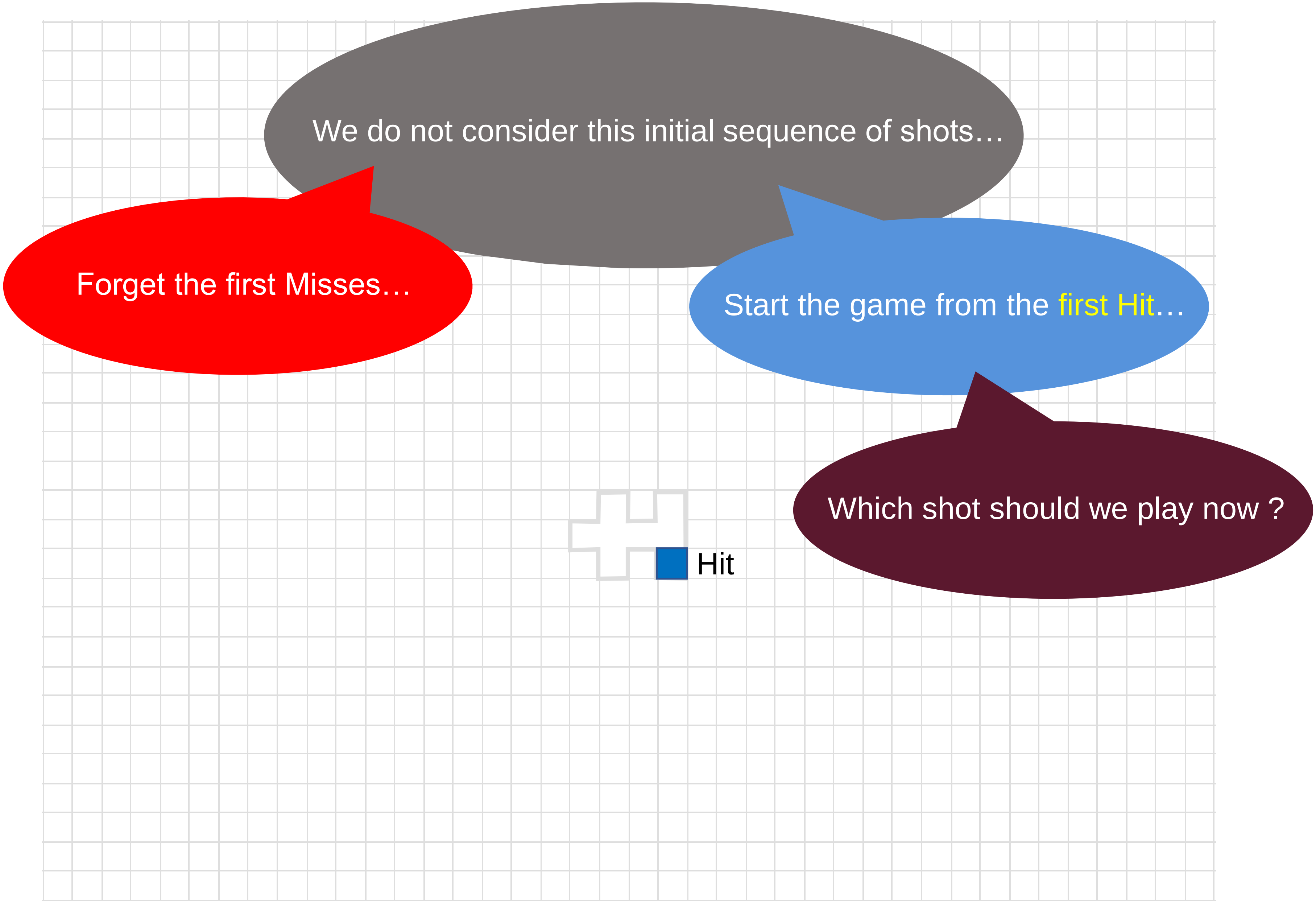

Miss
Miss
Miss
Miss
Miss
Miss
Miss
Miss
Hit

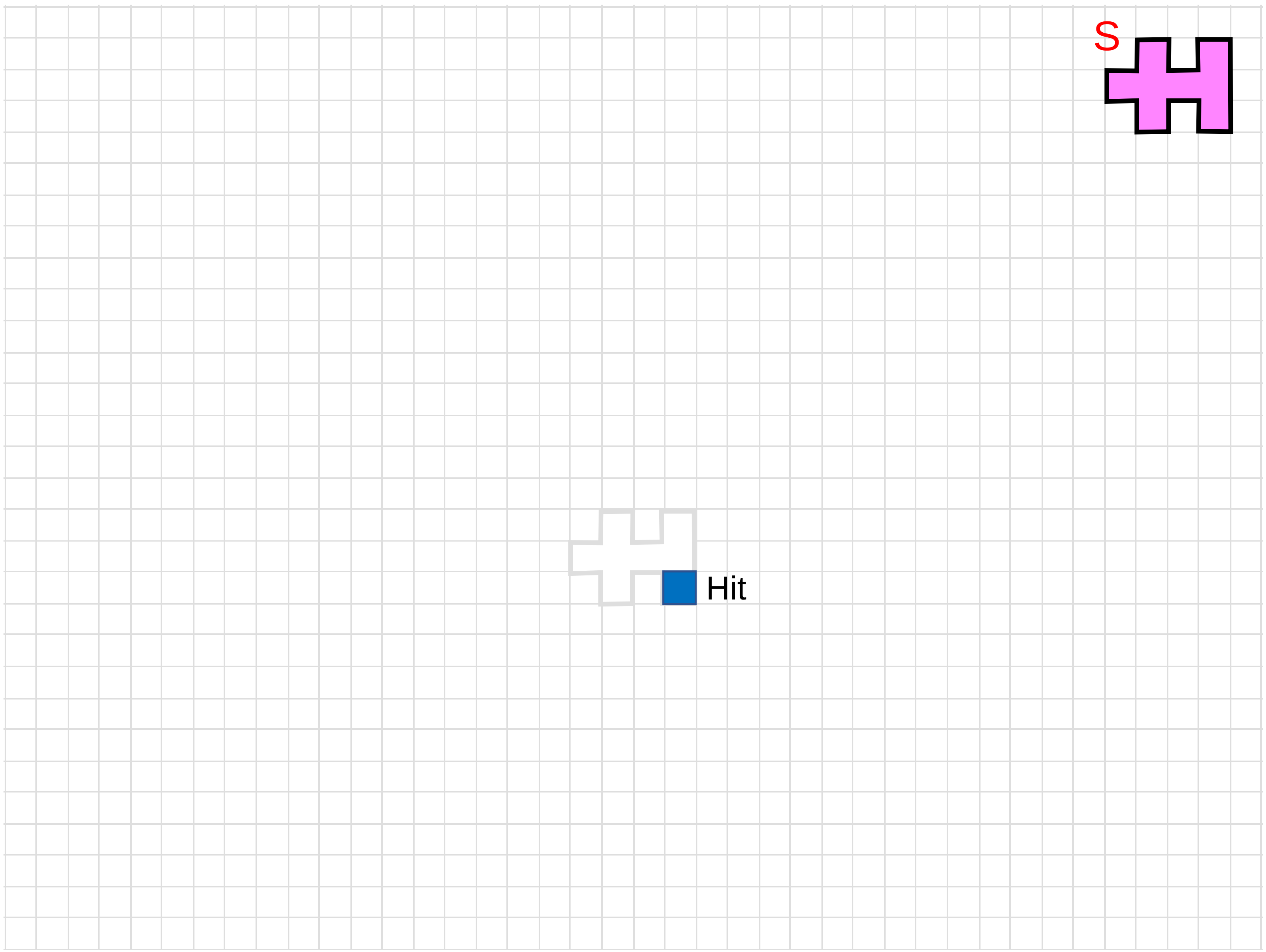
We do not consider this initial sequence of shots…



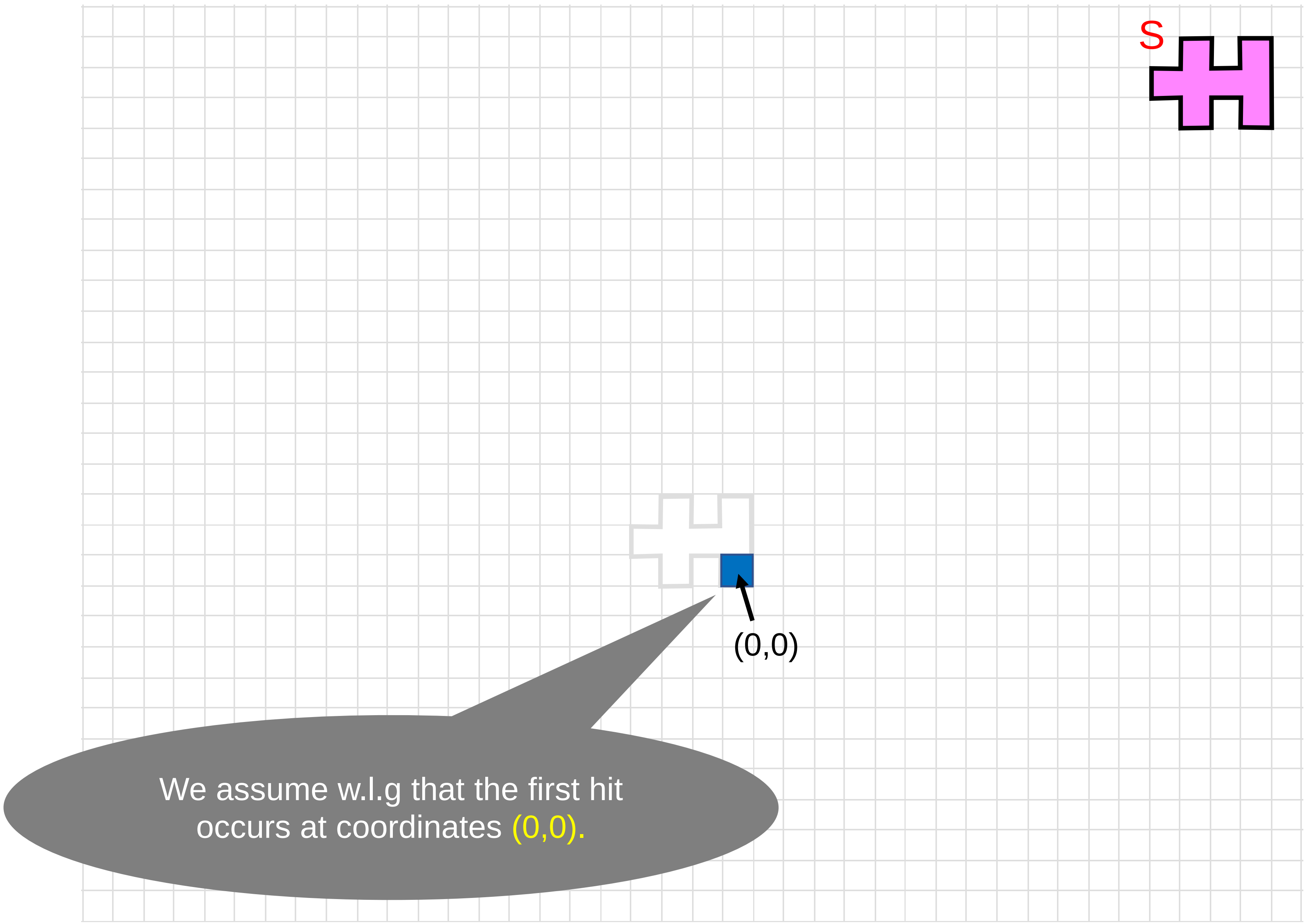

We do not consider this initial sequence of shots…
Forget the first Misses…
Miss
Miss
Miss
Miss
Miss
Hit
Miss

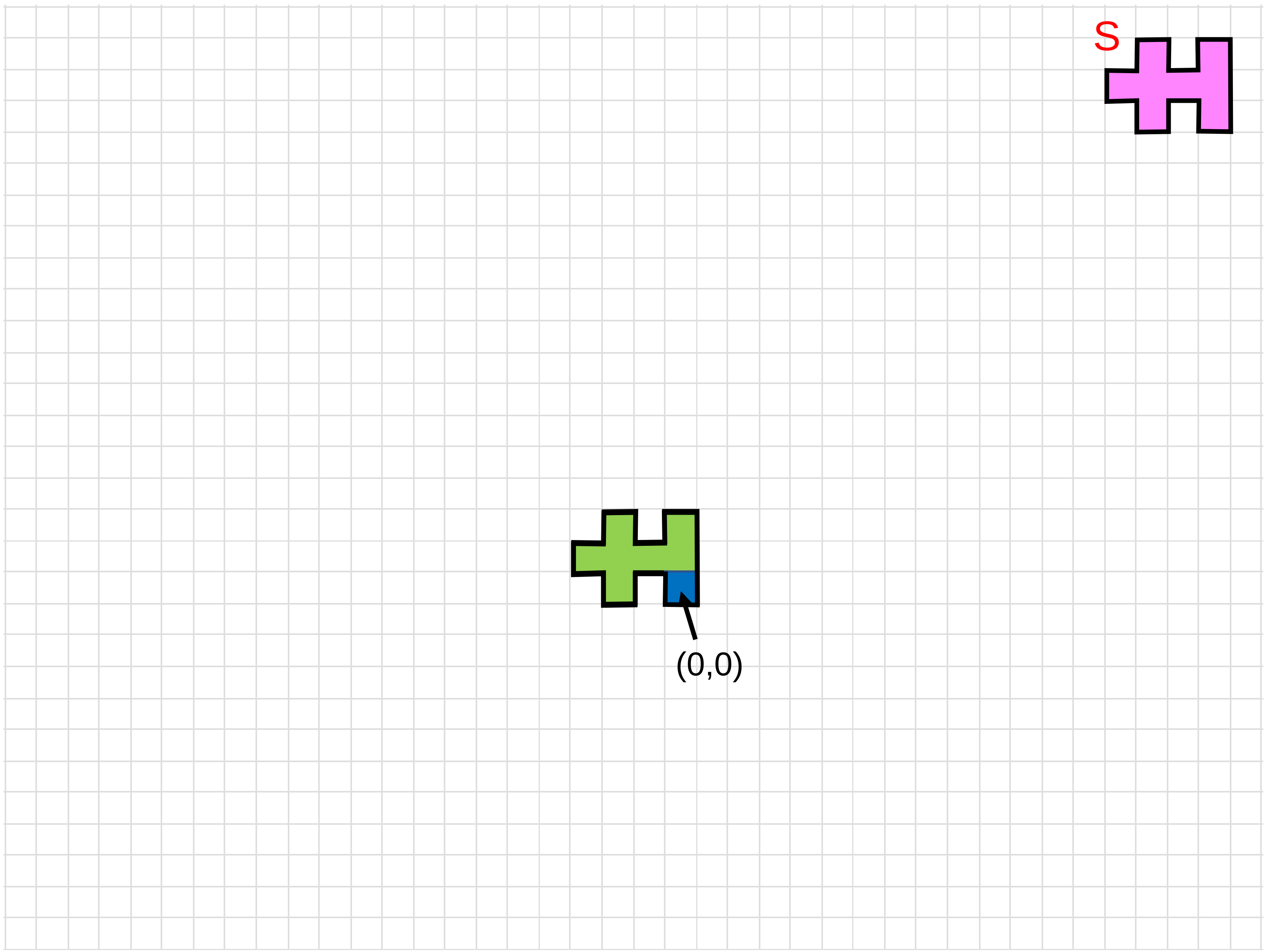

We do not consider this initial sequence of shots…
Forget the first Misses…
Start the game from the first Hit…
Hit

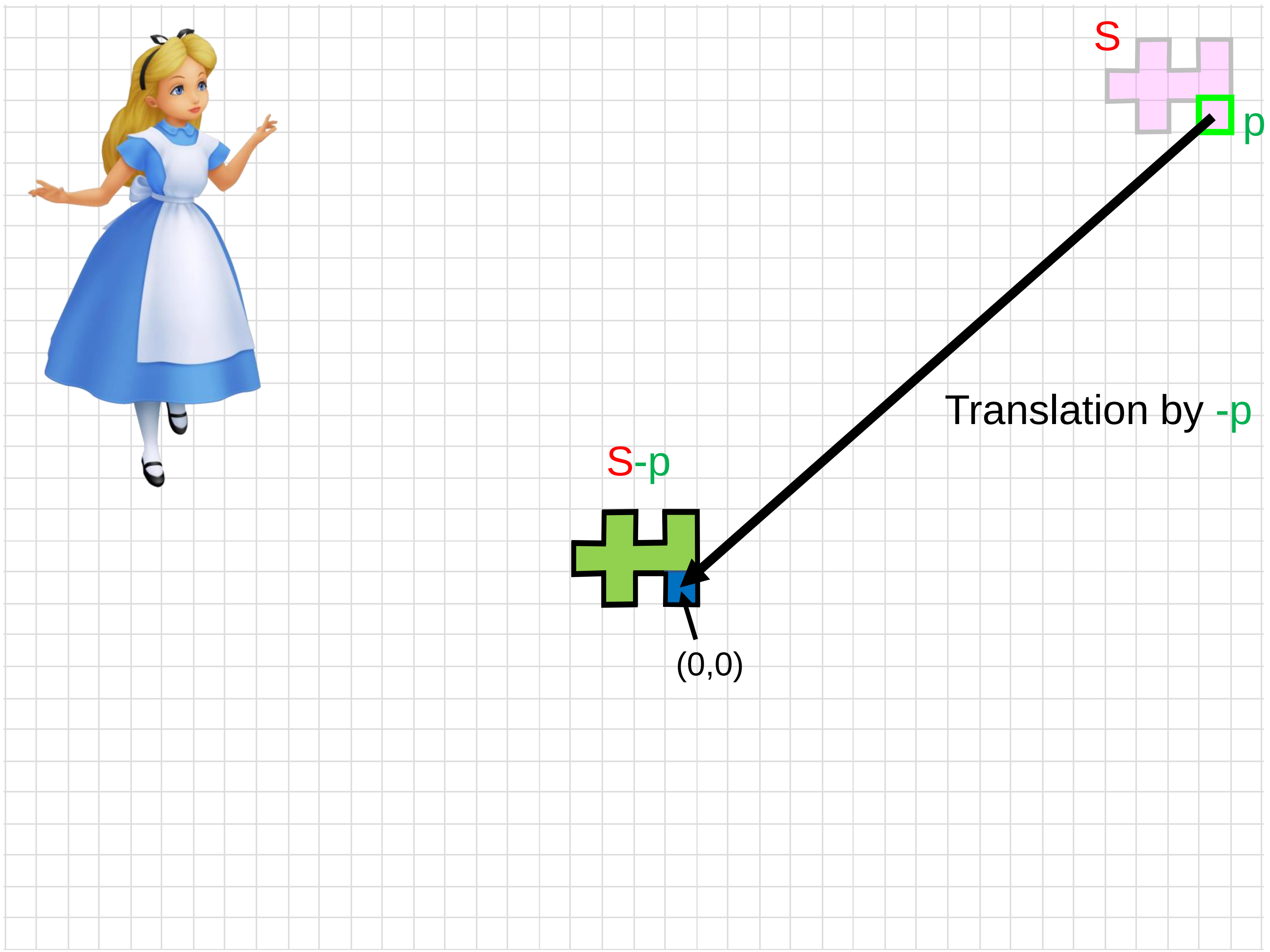

We do not consider this initial sequence of shots…
Forget the first Misses…
Start the game from the first Hit…
Which shot should we play now ?
Hit

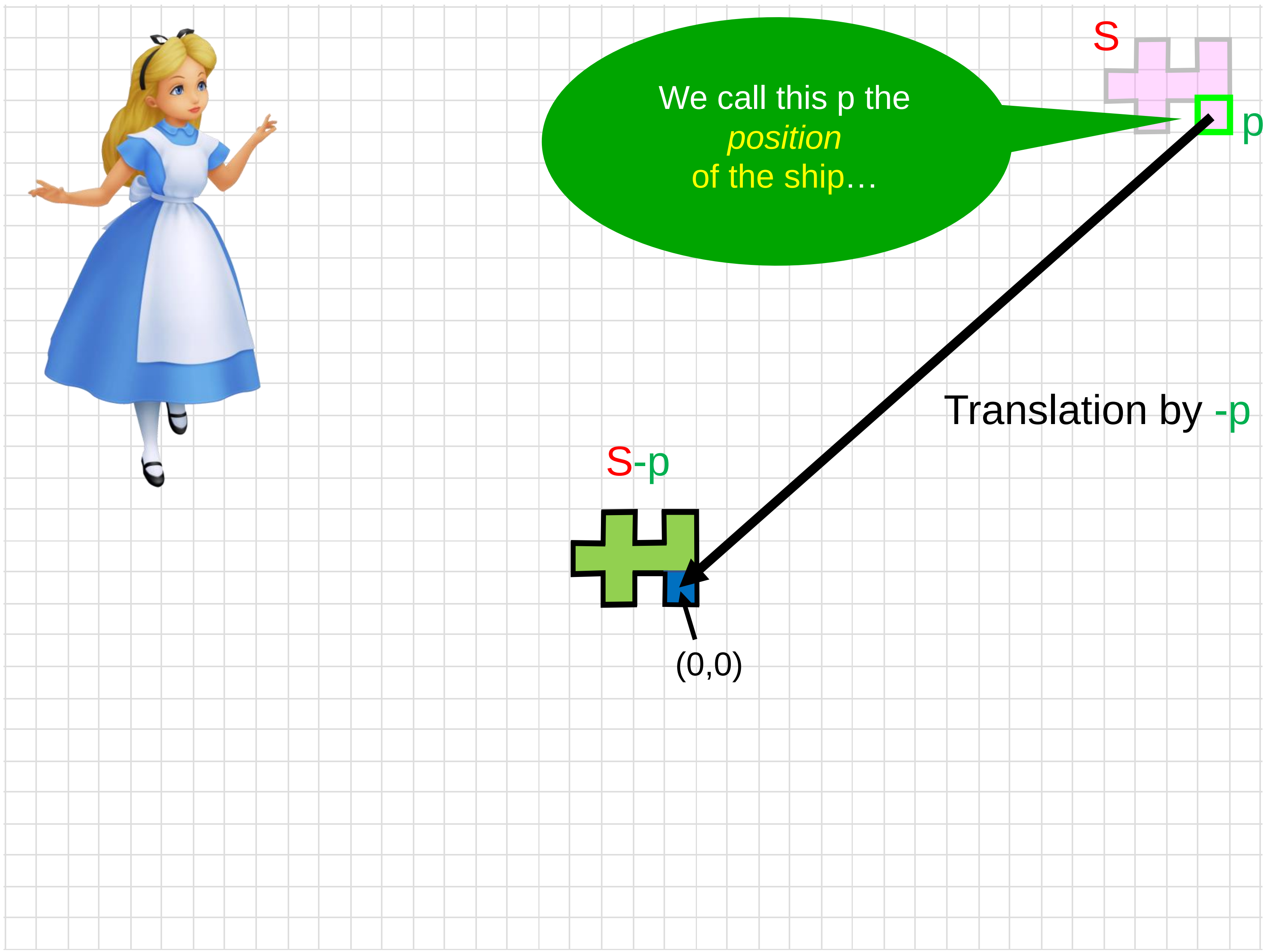


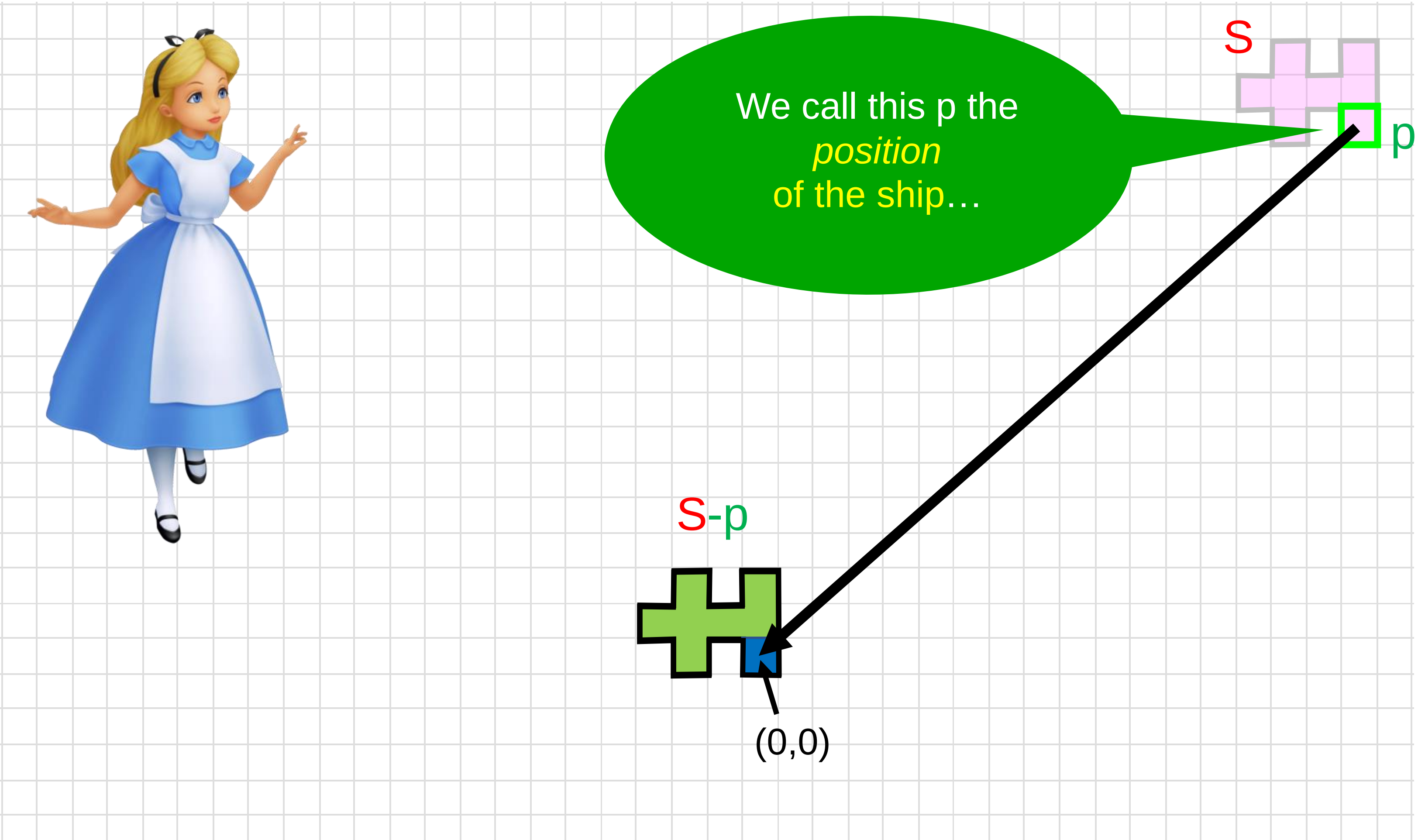
S
Hit


S
(0,0)
We assume w.l.g that the first hit occurs at coordinates (0,0).


S
(0,0)

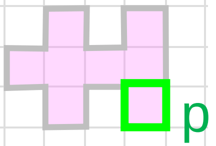

S
p
Translation by -p
S-p
(0,0)


We call this p the
position
of the ship…
S
p
Translation by -p
S-p
(0,0)

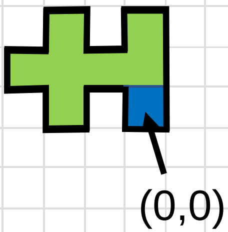


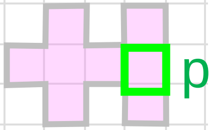
The position of the ship is the point of the shape that Alice sent to the origin…
and which is hit at the initial shot.

Position

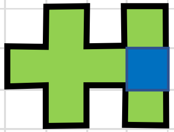


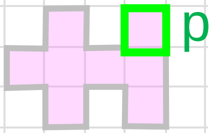
For a shape of size n,
Alice has the choice
between n positions p…
and we have to guess p



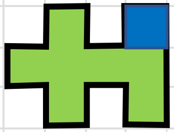


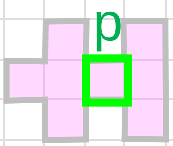
For a shape of size n,
Alice has the choice
between n positions p…
and we have to guess p



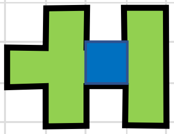


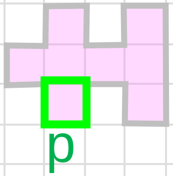
For a shape of size n,
Alice has the choice
between n positions p…
and we have to guess p



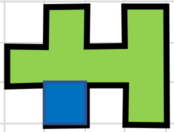


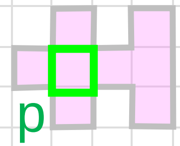
For a shape of size n,
Alice has the choice
between n positions p…
and we have to guess p



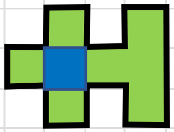


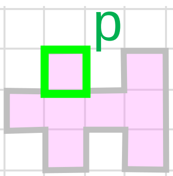
For a shape of size n,
Alice has the choice
between n positions p…
and we have to guess p



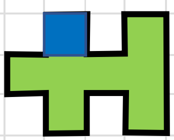


For a shape of size n,
Alice has the choice
between n positions p…
and we have to guess p

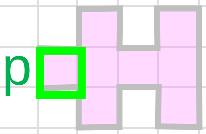




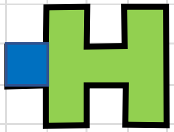
For a shape of size n,
Alice has the choice
between n positions p…
and we have to guess p

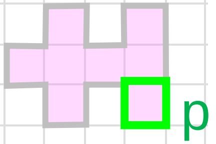


Problem statement

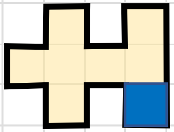


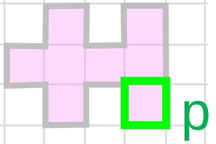
For a shape of size n,
Alice has the choice
between n positions p…
and we have to guess p



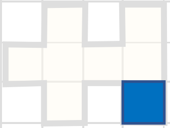


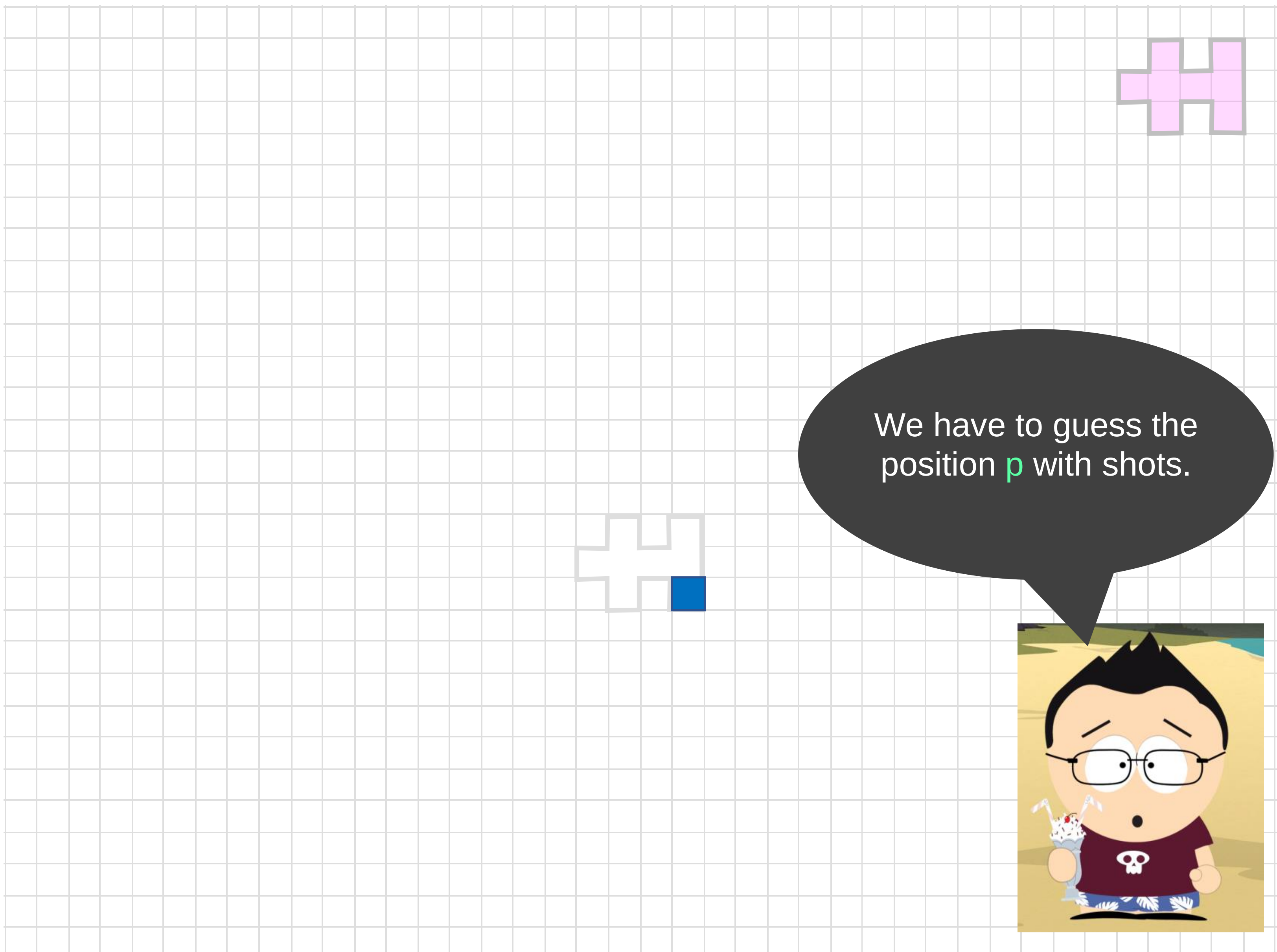
Alice chooses this position p

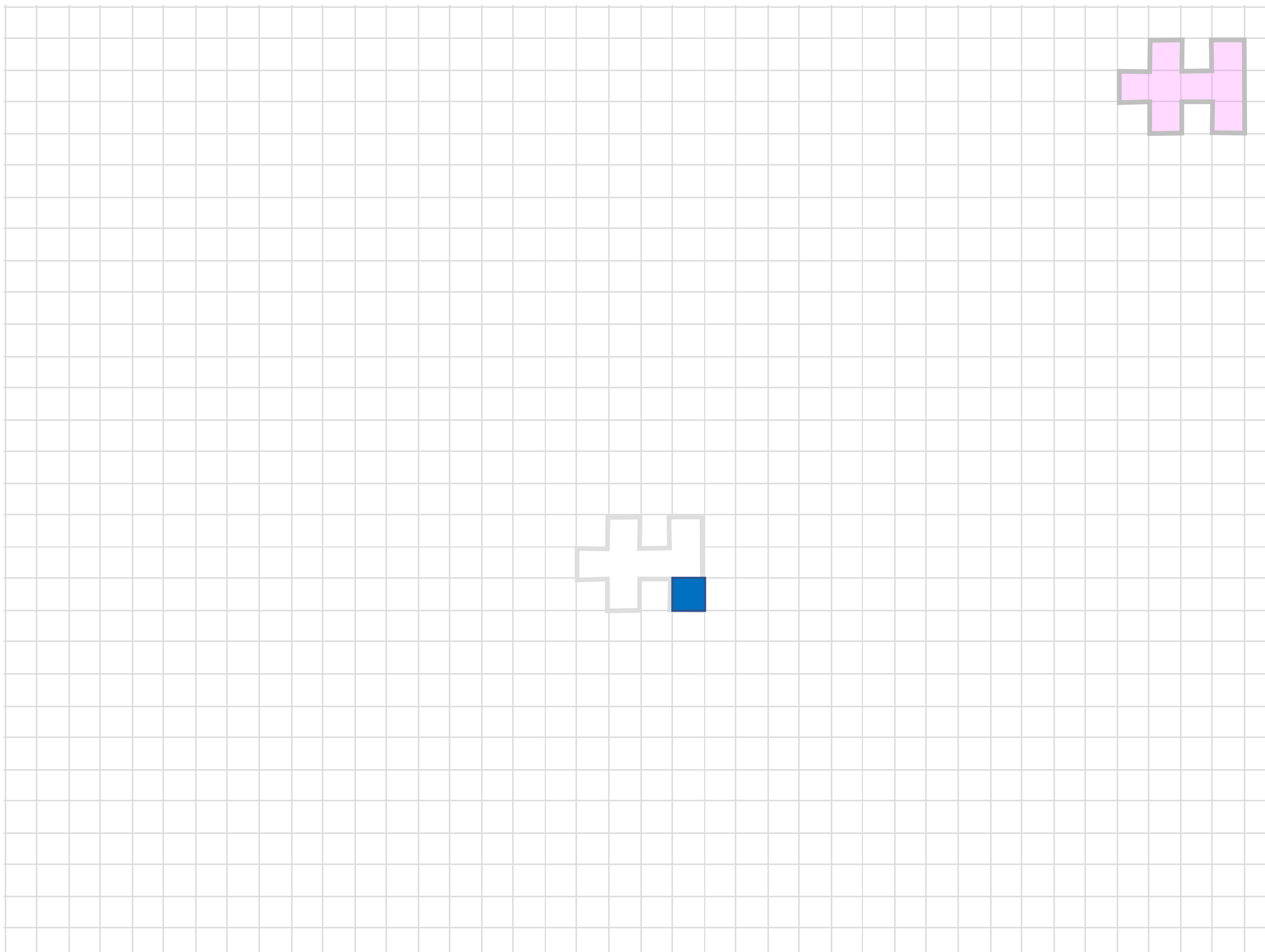


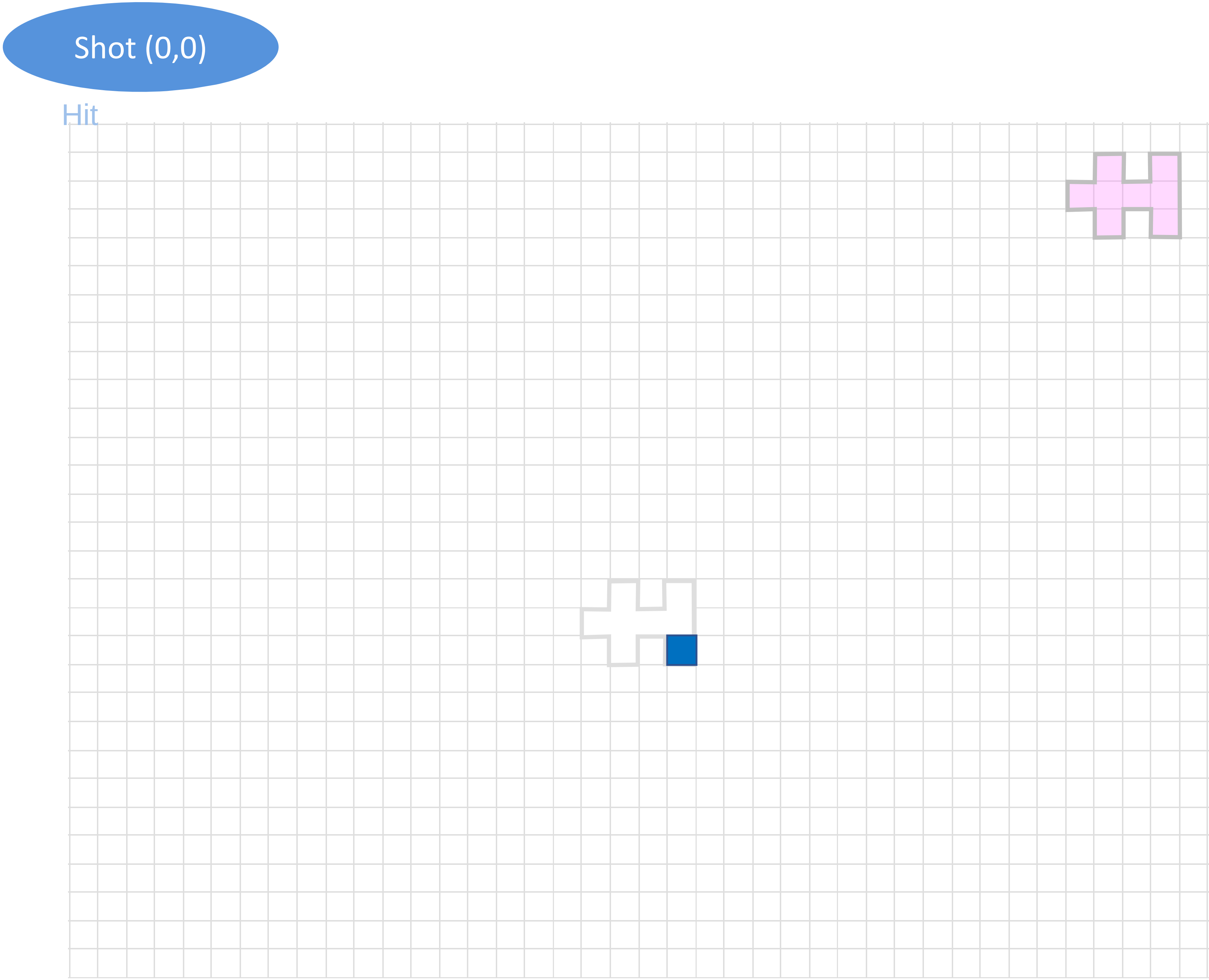


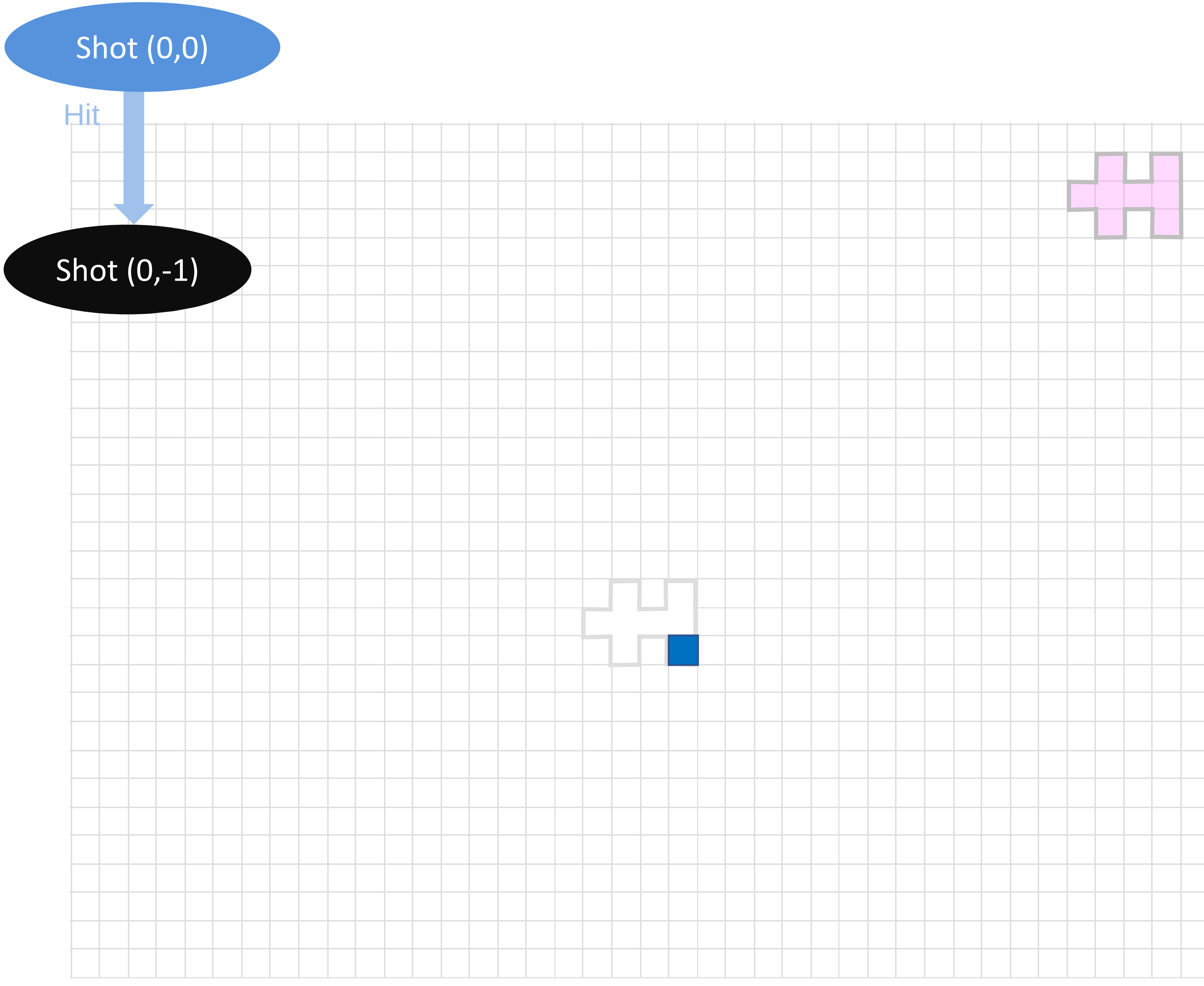
Alice chooses this position p

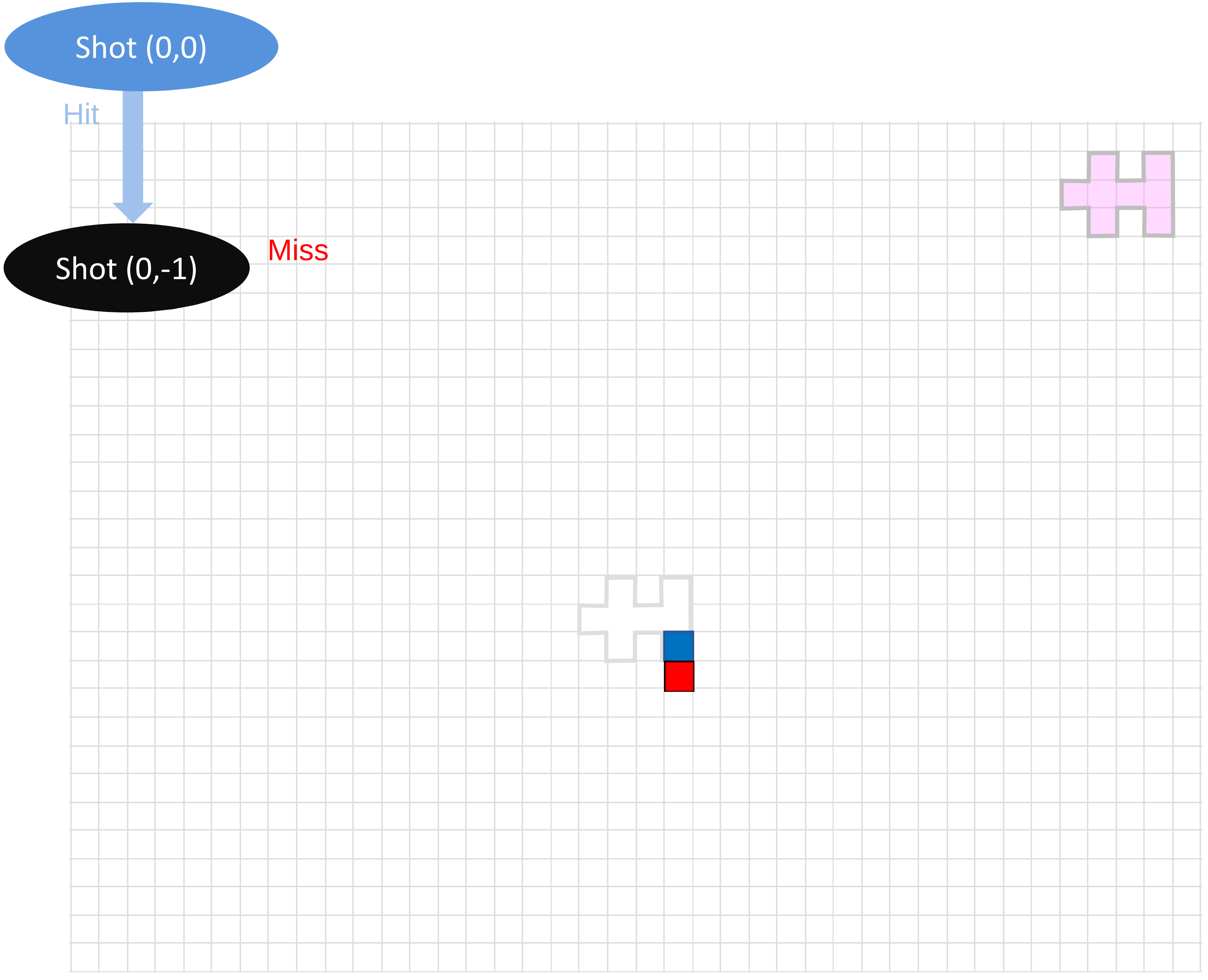


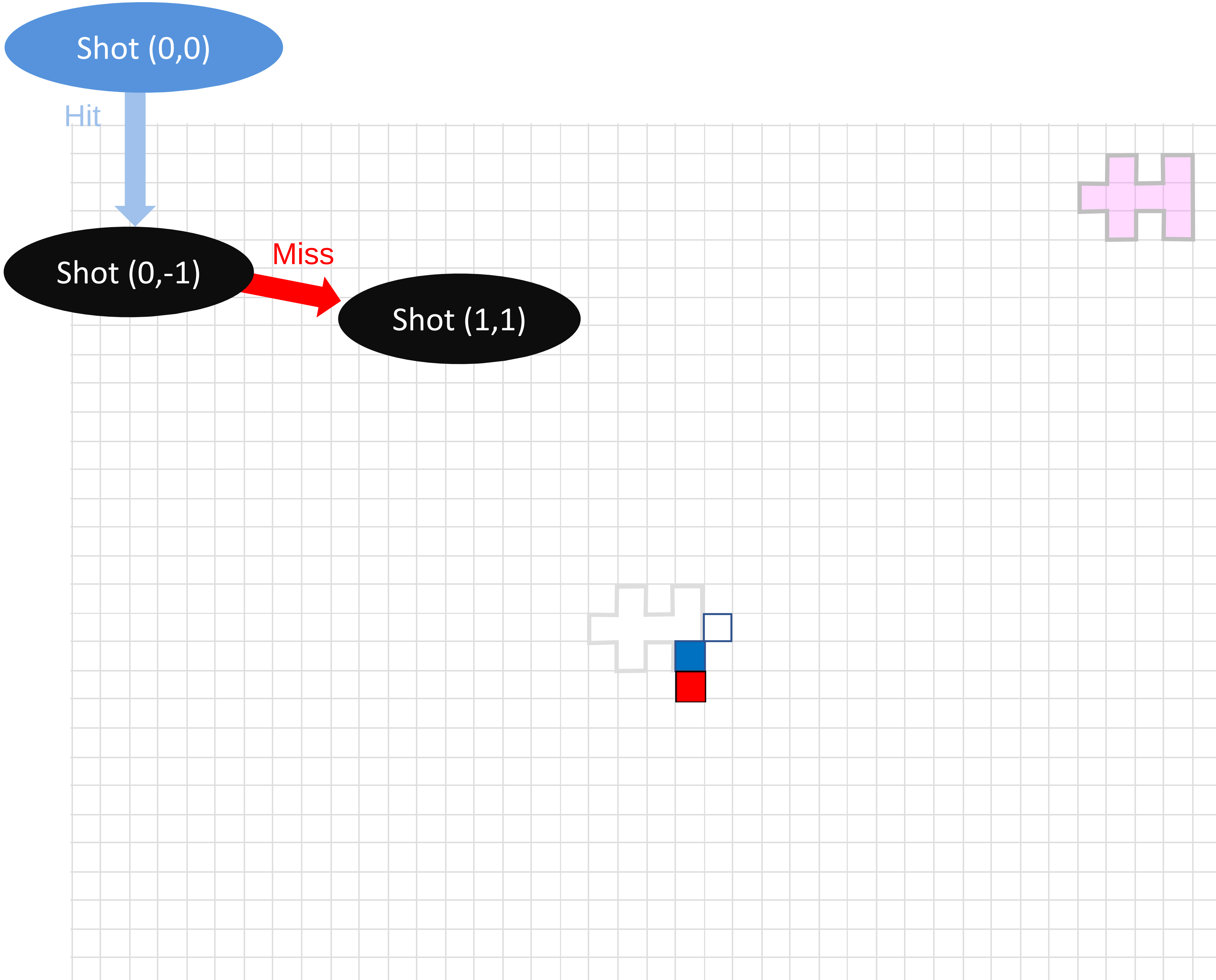

We have to guess the position p with shots.

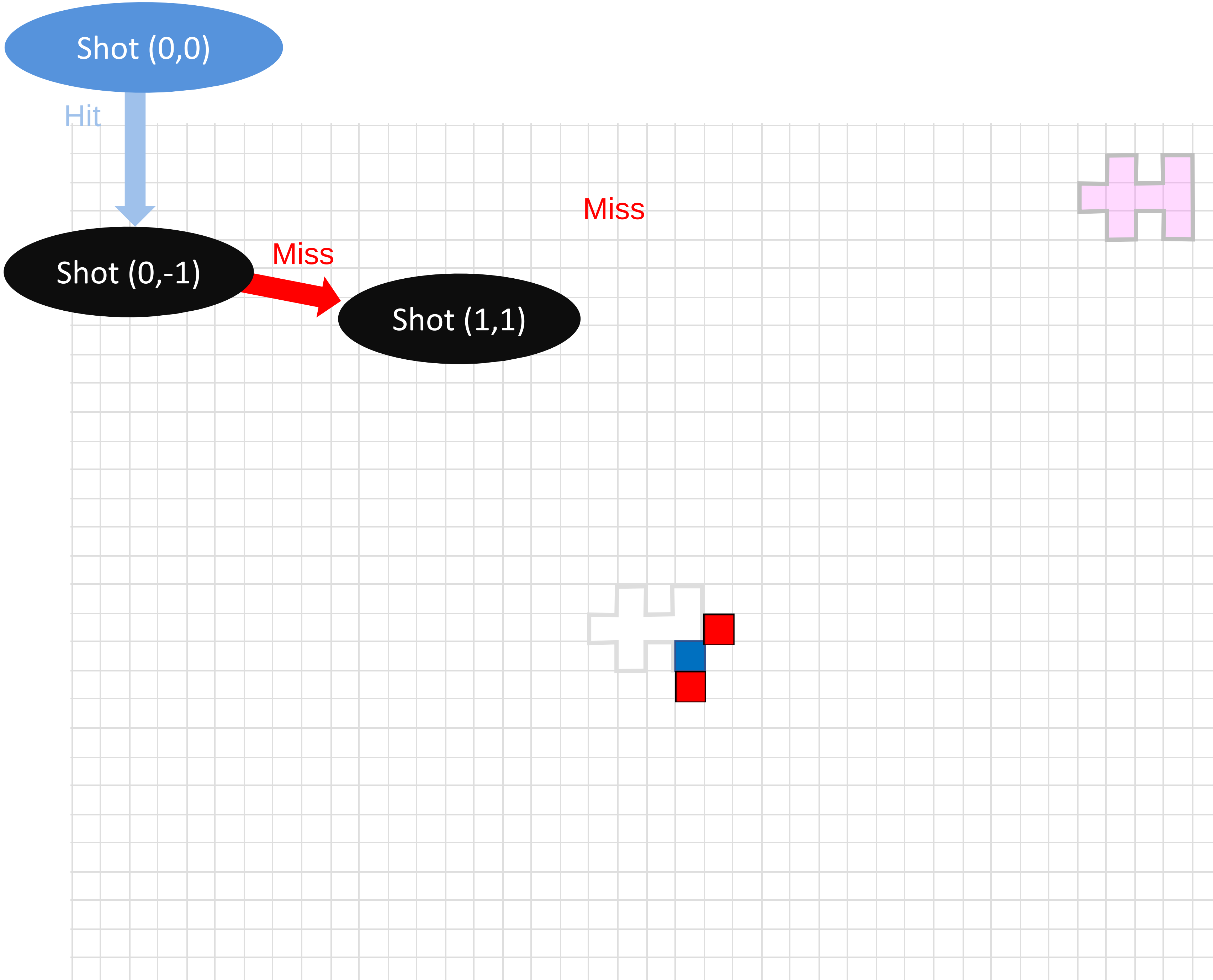


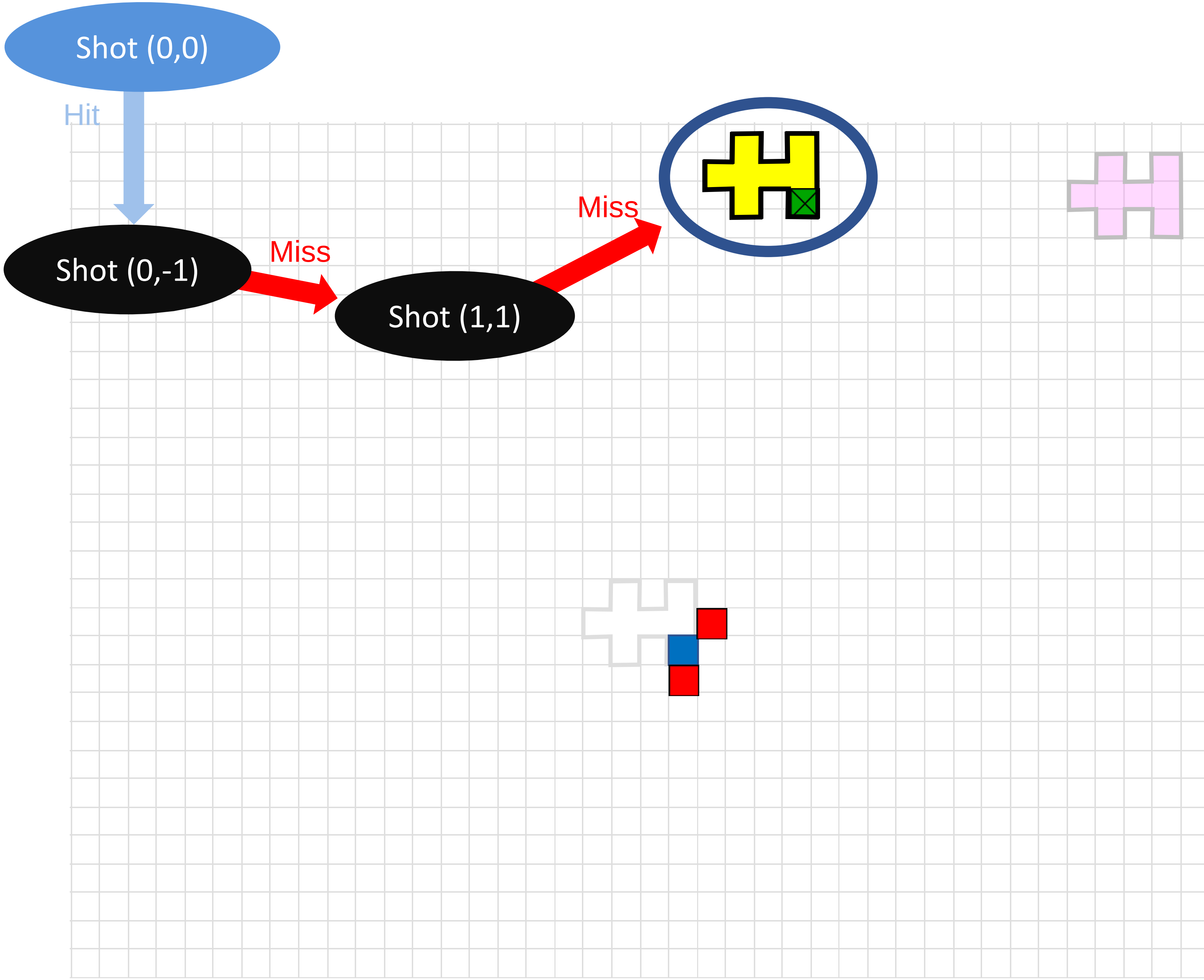

Shot (0,0)
Hit

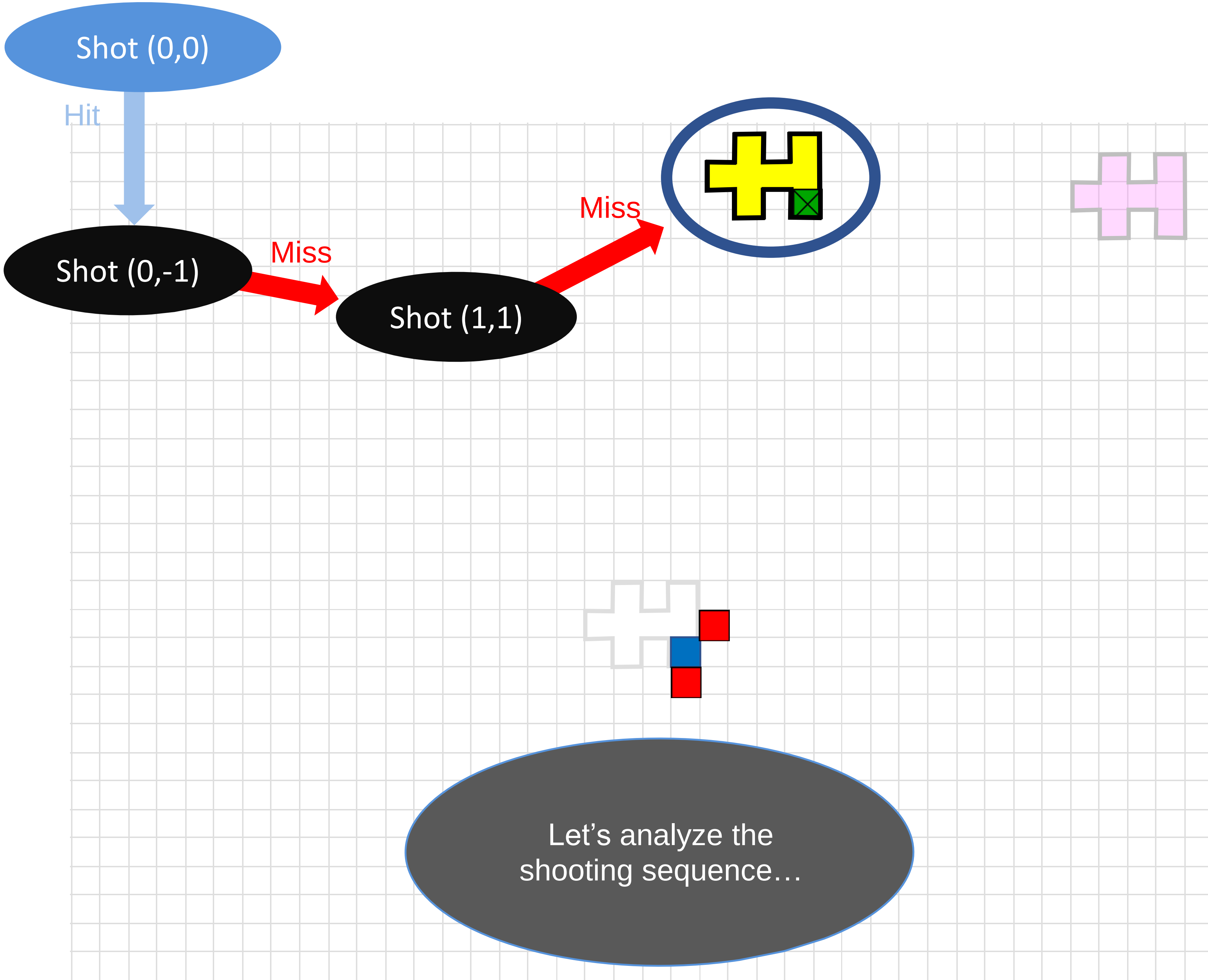

Shot (0,0)
Hit
Shot (0,-1)

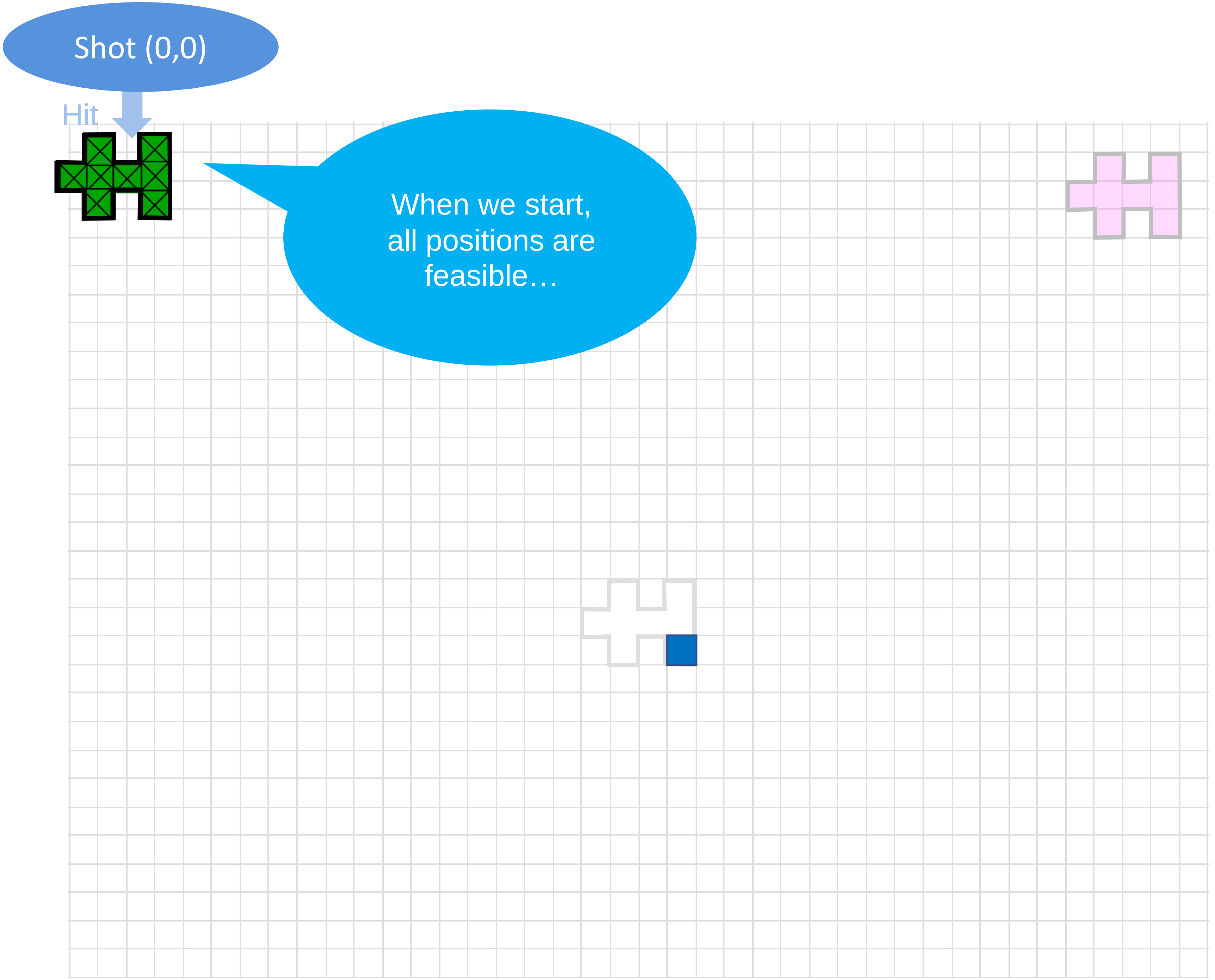

Shot (0,0)
Hit
Shot (0,-1)
Miss

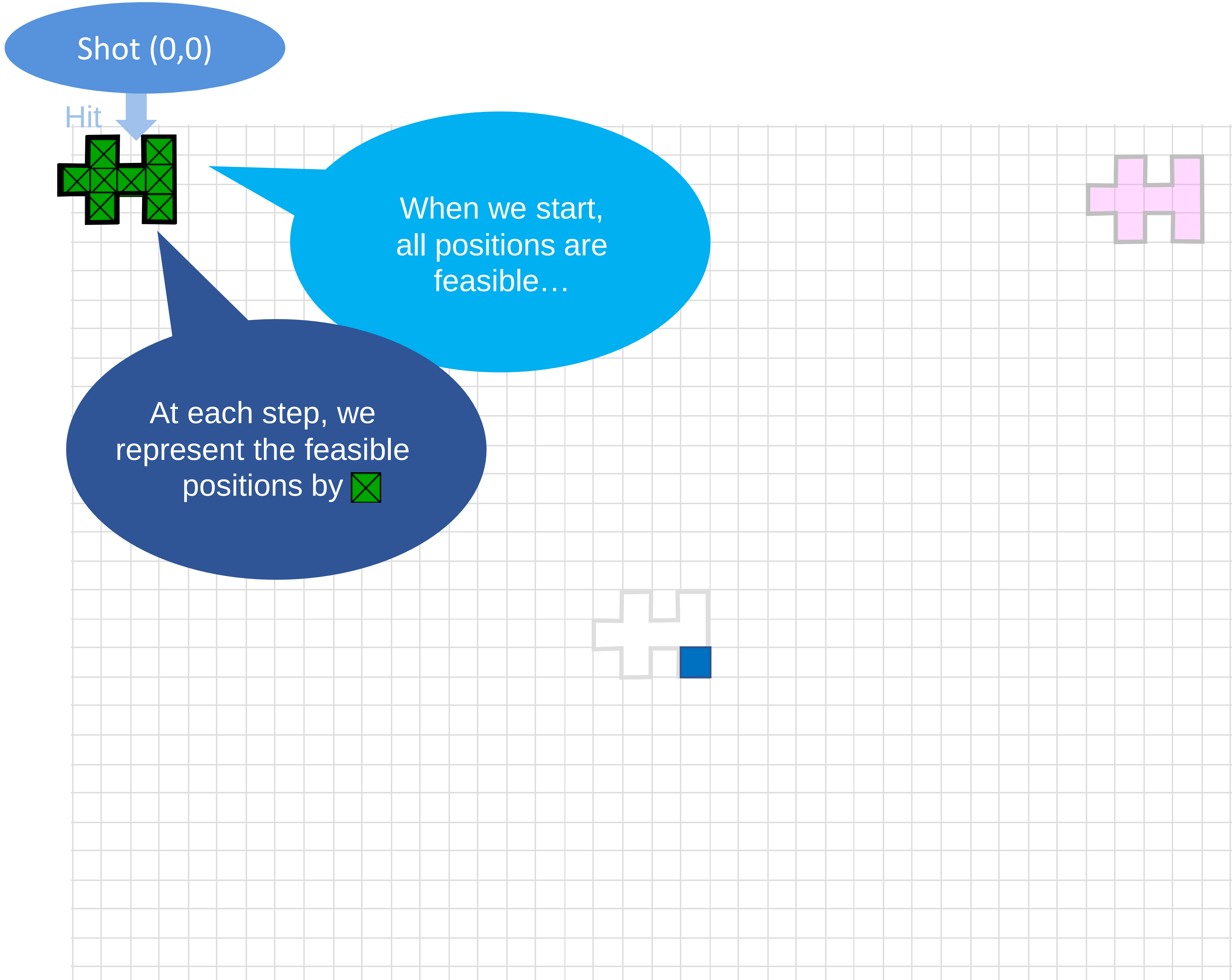

Shot (0,0)
Hit
Shot (0,-1)
Miss
Shot (1,1)

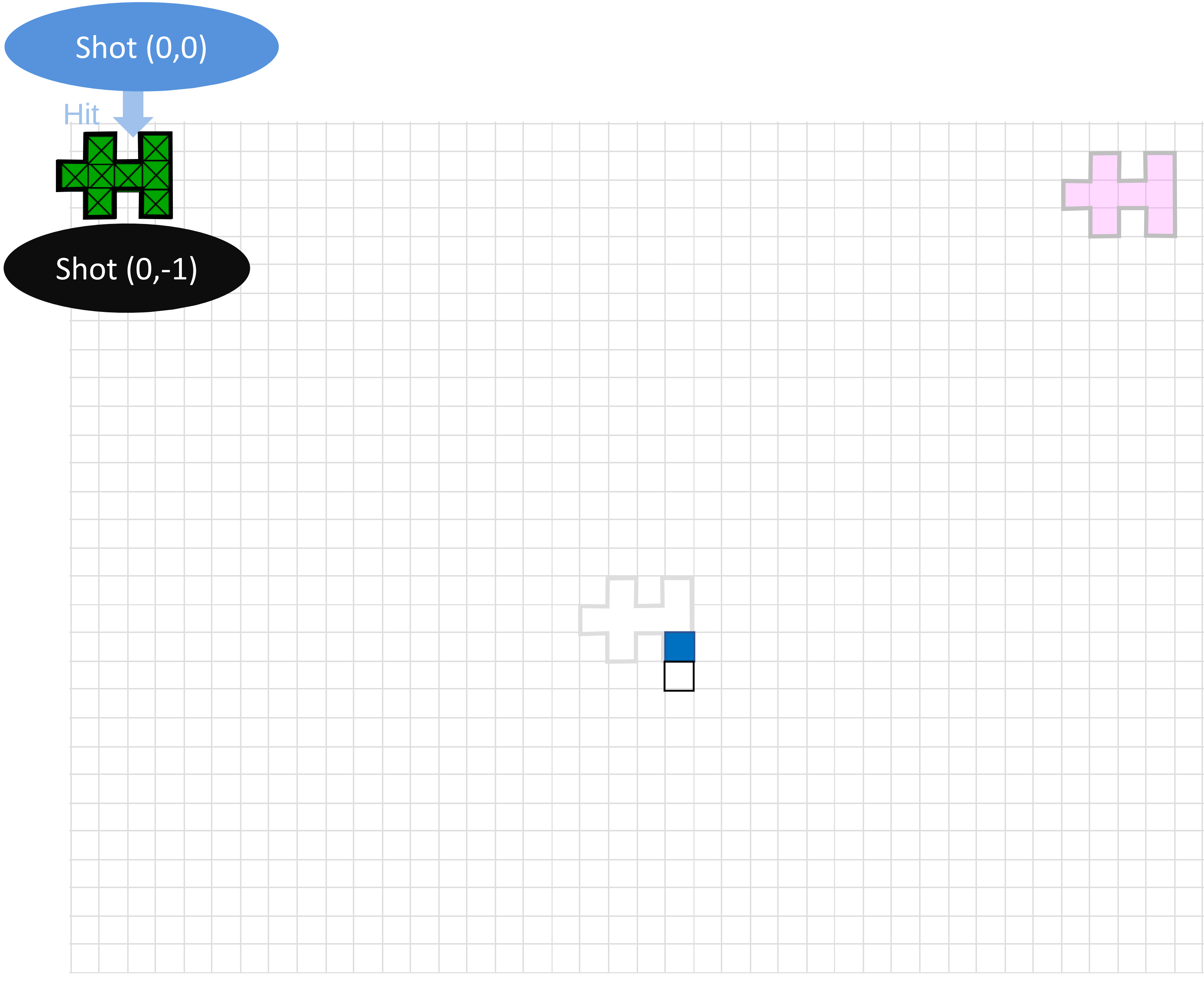

Shot (0,0)
Hit
Shot (0,-1)
Miss
Shot (1,1)
Miss

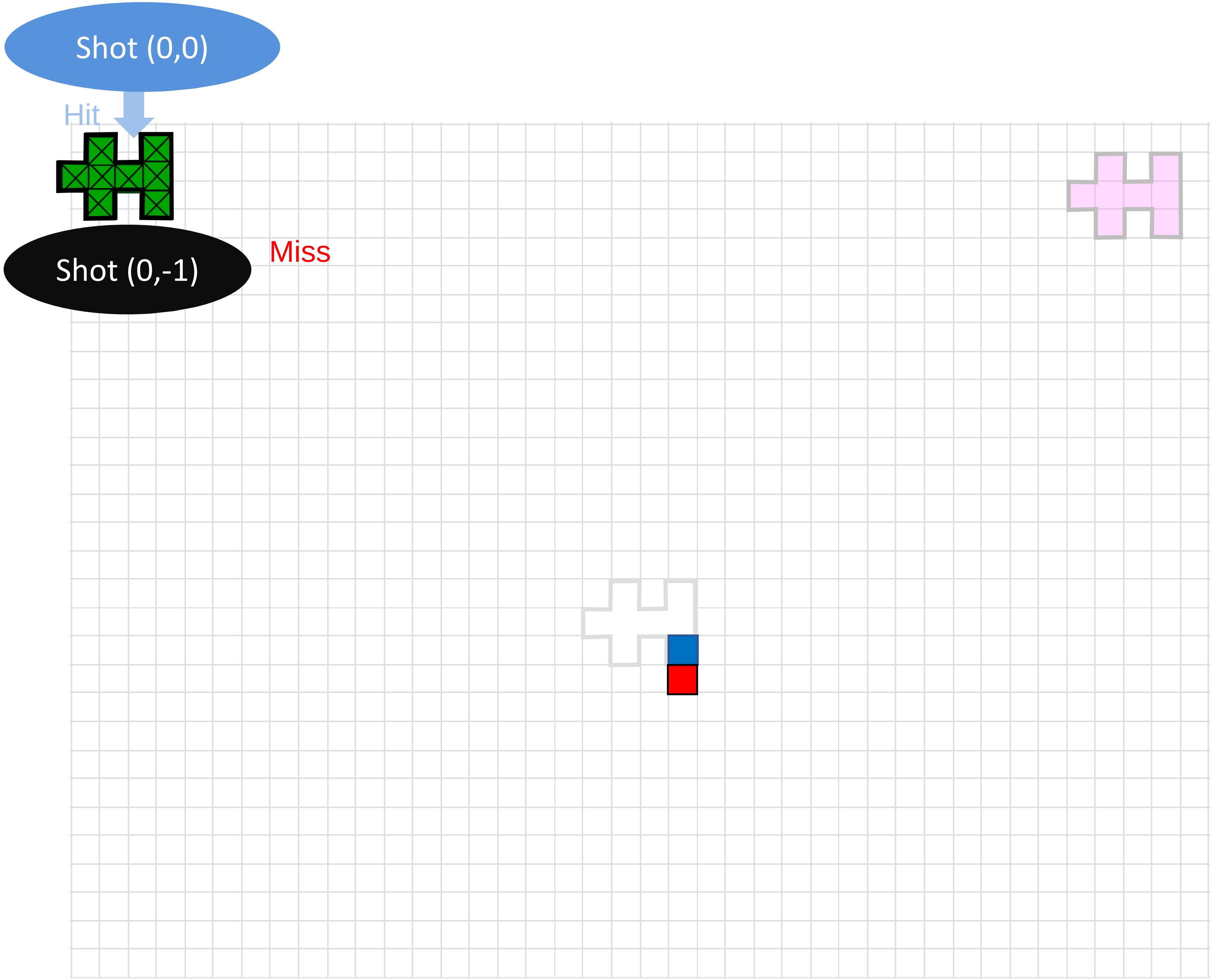

Shot (0,0)
Hit
Shot (0,-1)
Miss
Shot (1,1)
Miss

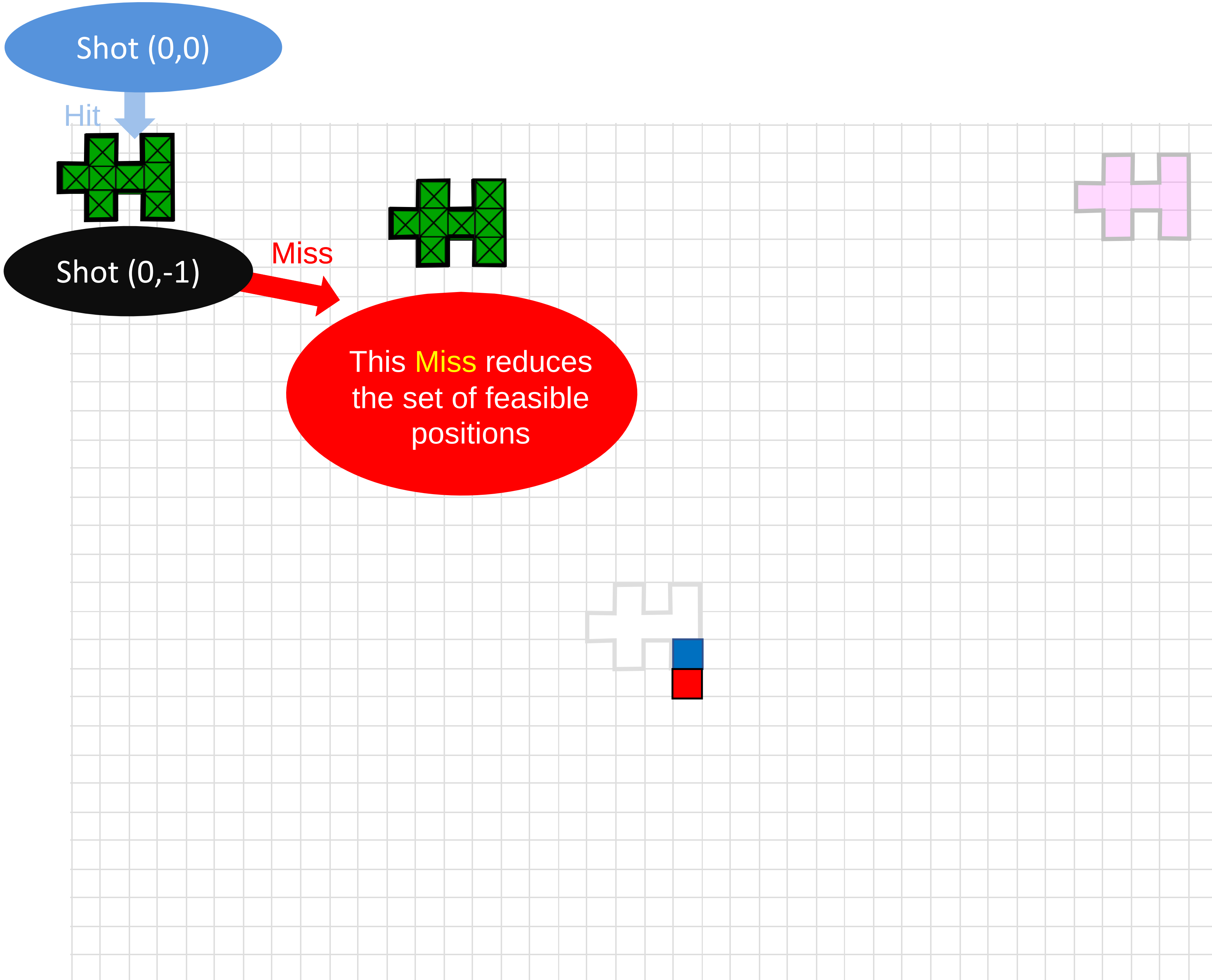

Shot (0,0)
Hit
Shot (0,-1)
Miss
Shot (1,1)
Miss
Let’s analyze the
shooting sequence…


Shot (0,0)
Hit
When we start, all positions are feasible…


Shot (0,0)
Hit
When we start, all positions are feasible…
At each step, we represent the feasible positions by

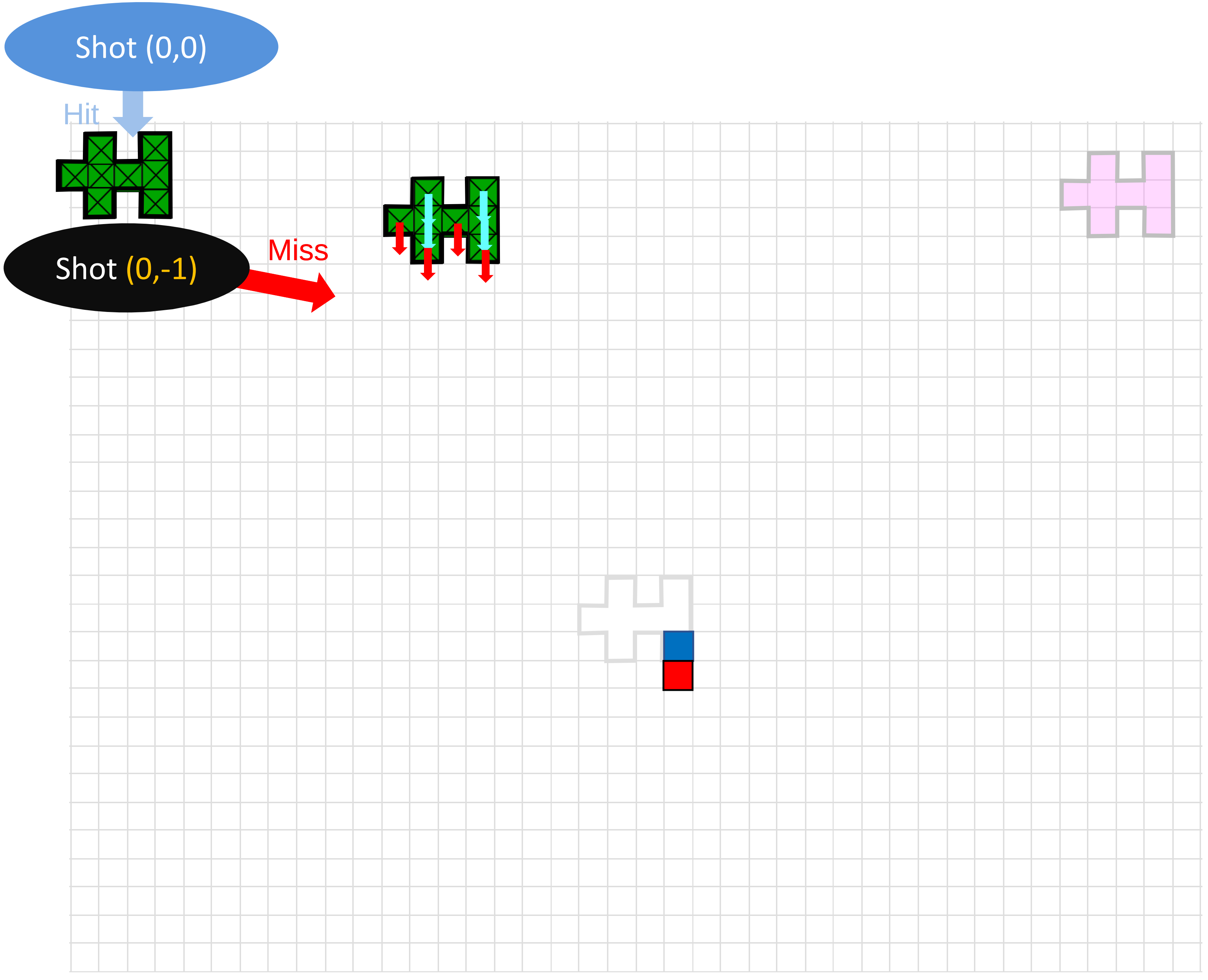

Shot (0,0)
Hit
Shot (0,-1)

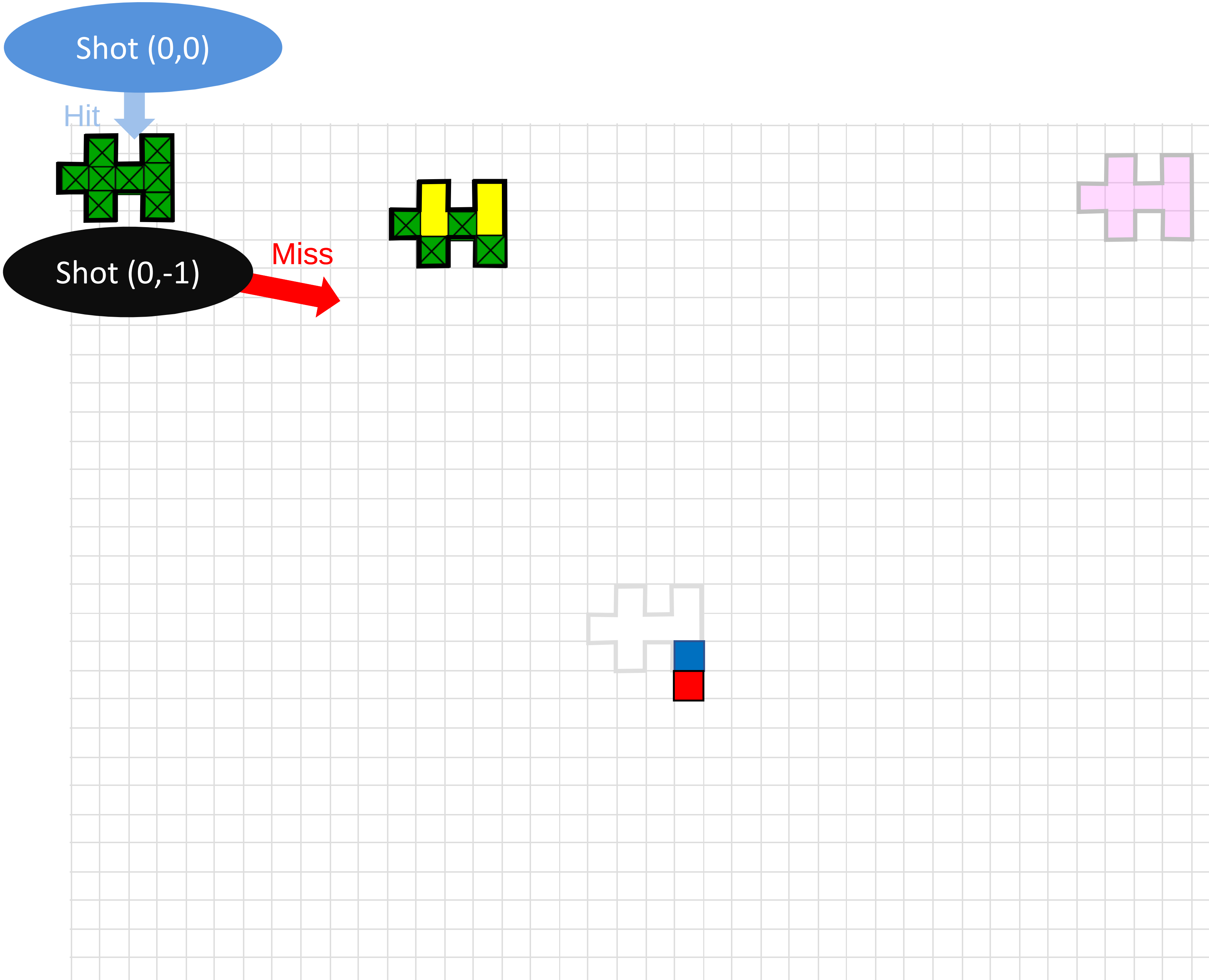

Shot (0,0)
Hit
Shot (0,-1)
Miss

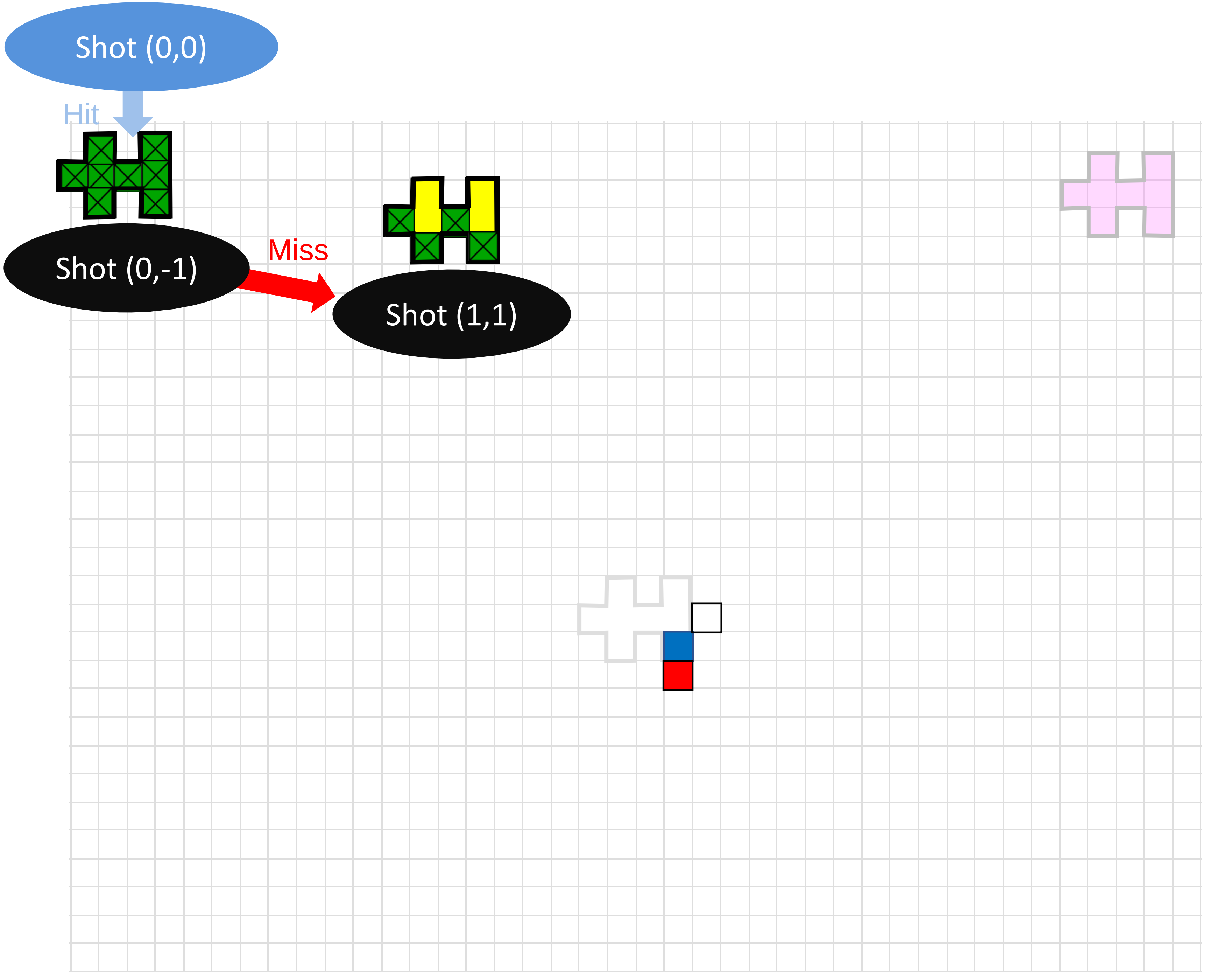


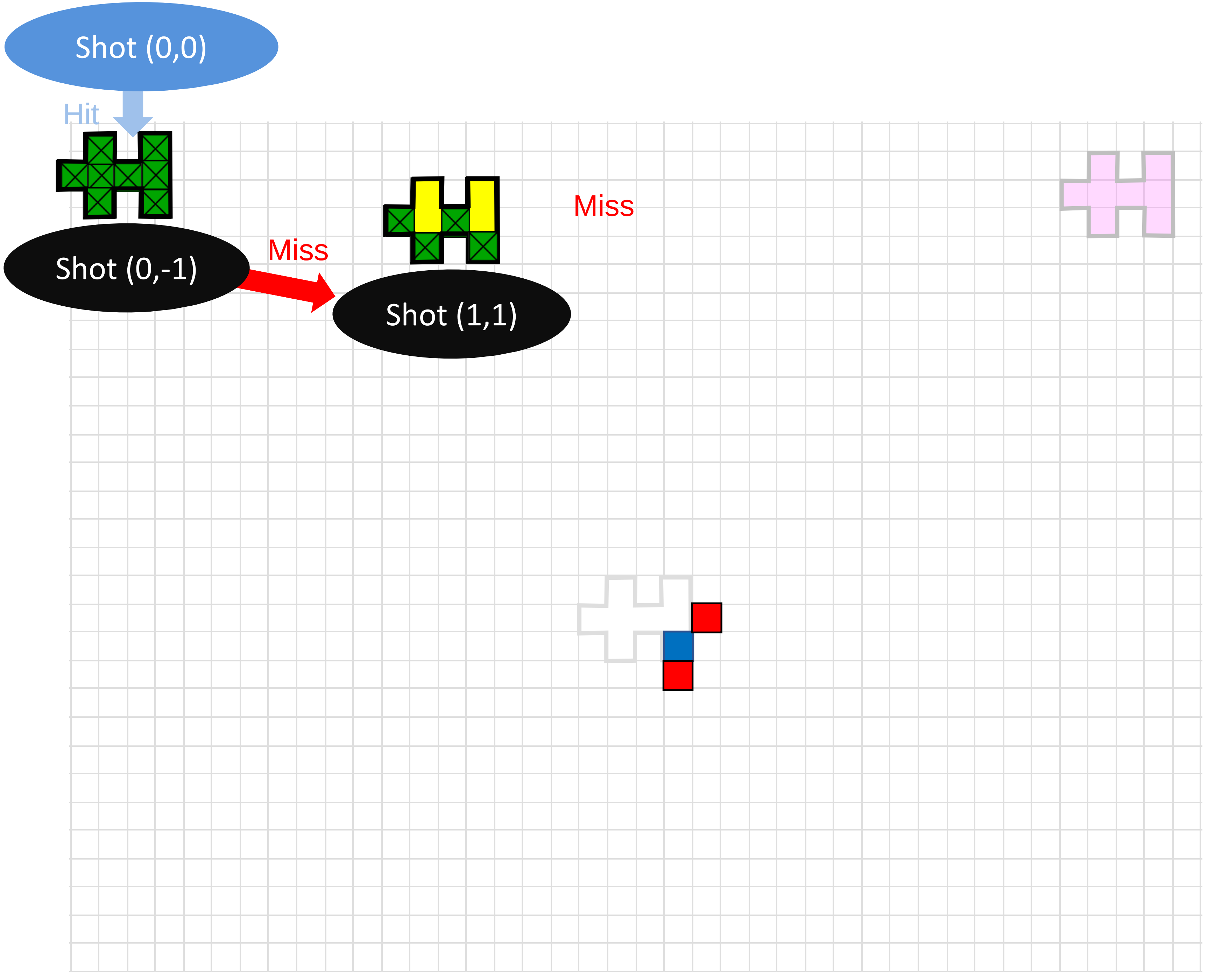
Shot (0,0)
Hit
Shot (0,-1)
Miss
This Miss reduces the set of feasible positions


Shot (0,0)
Hit
Shot (0,-1)
Miss

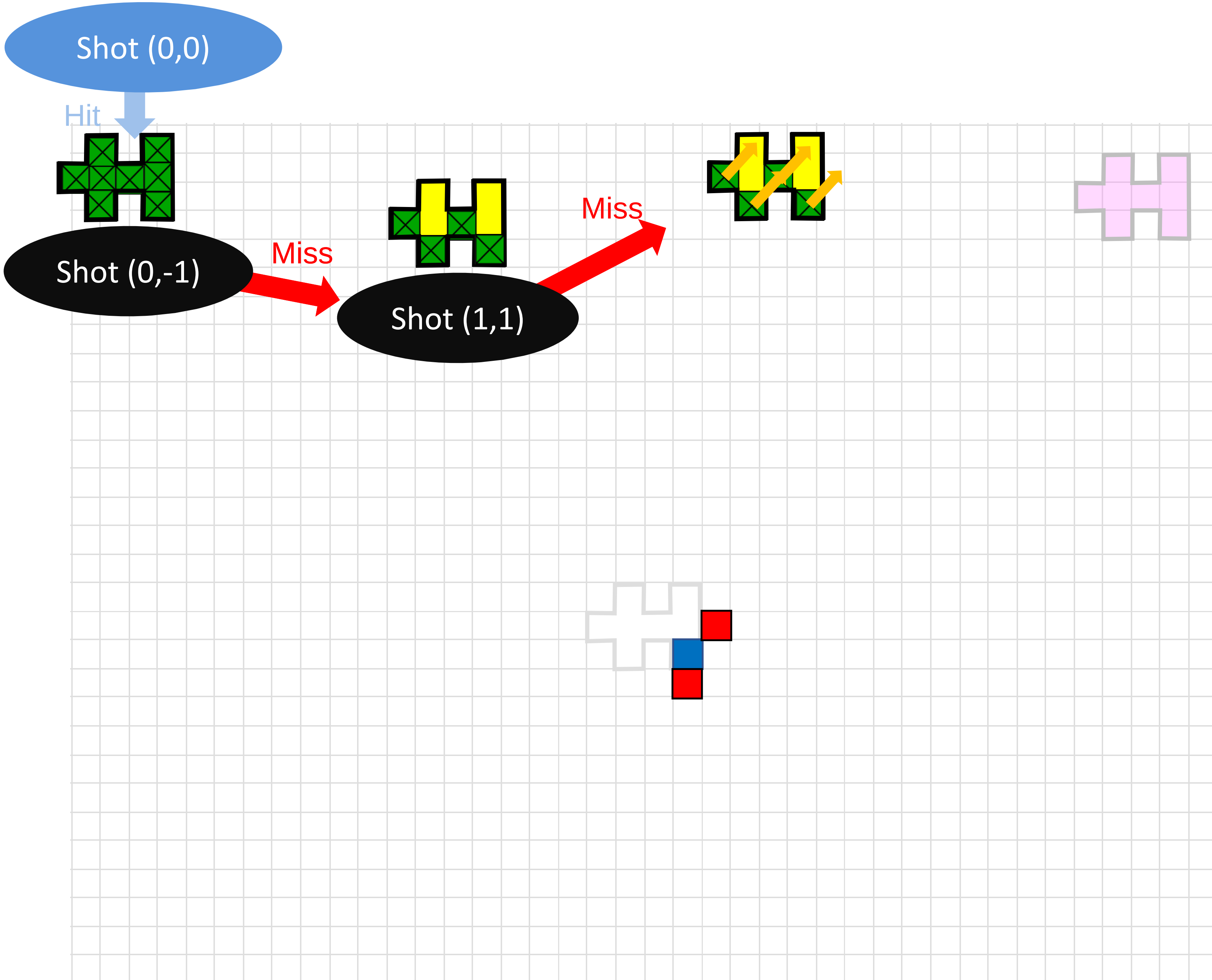

Shot (0,0)
Hit
Shot (0,-1)
Miss

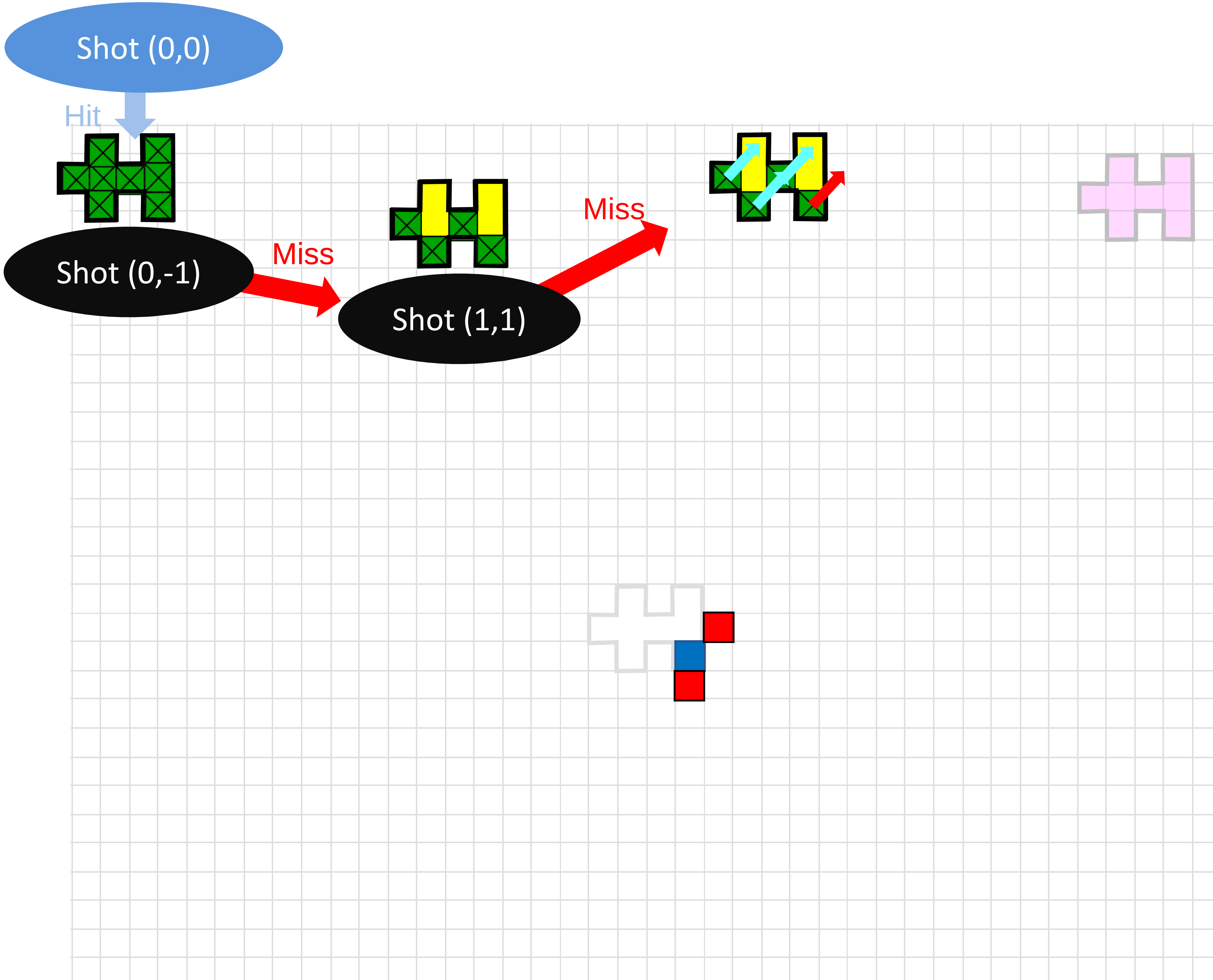

Shot (0,0)
Hit
Shot (0,-1)
Miss


Shot (0,0)
Hit
Shot (0,-1)
Miss
Shot (1,1)

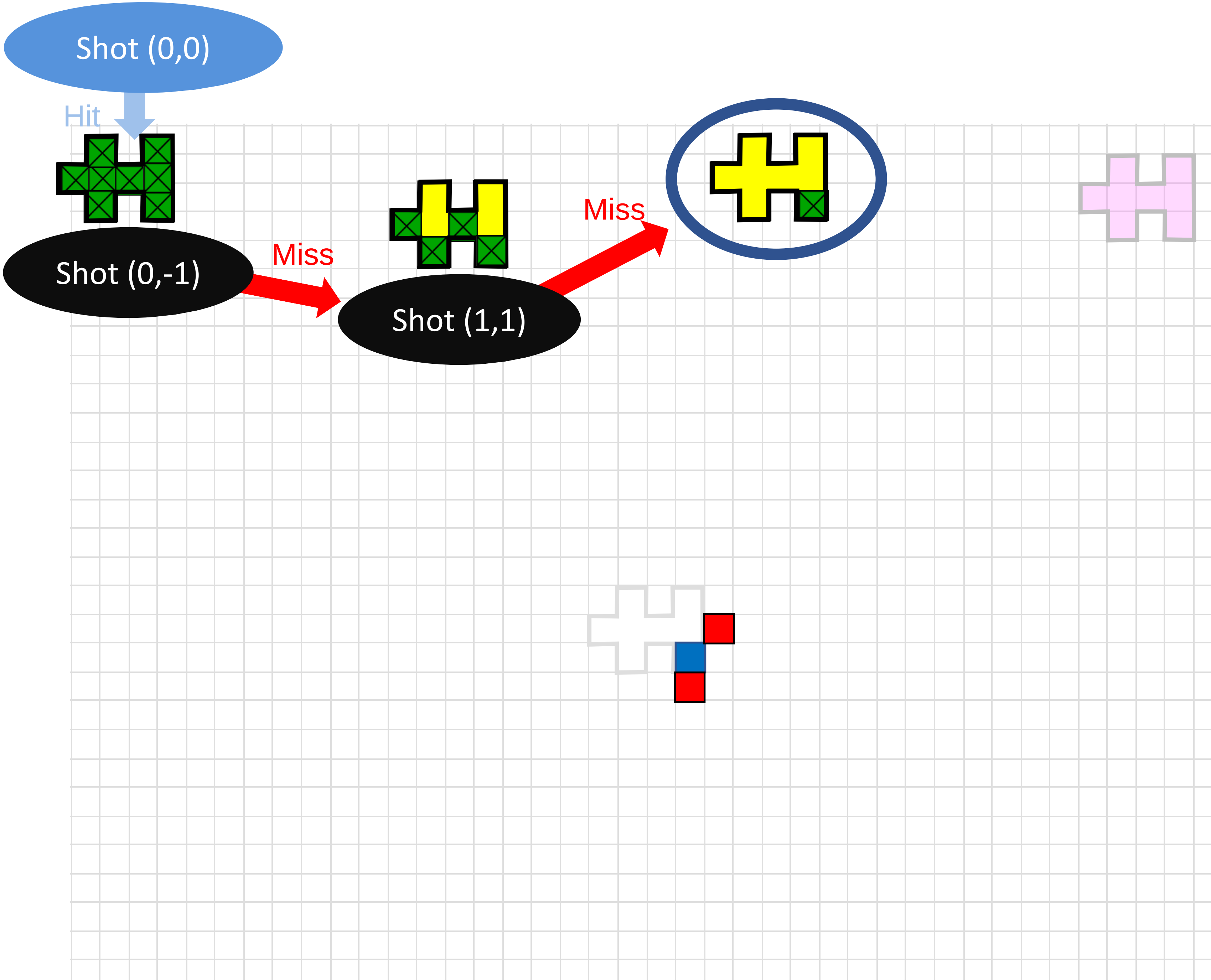


Shot (0,0)
Hit
Shot (0,-1)
Miss
Shot (1,1)
Miss

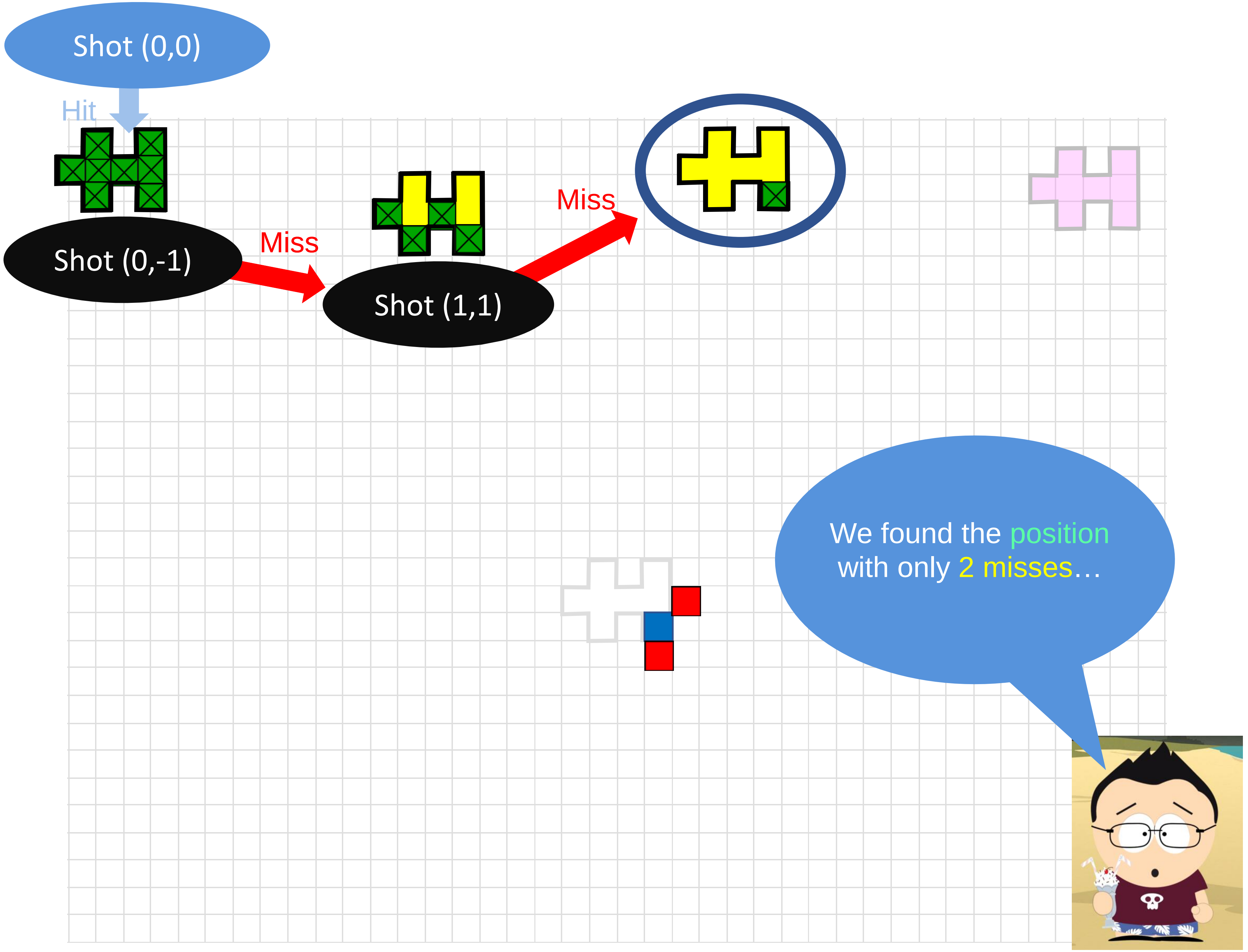

Shot (0,0)
Hit
Shot (0,-1)
Miss
Shot (1,1)
Miss



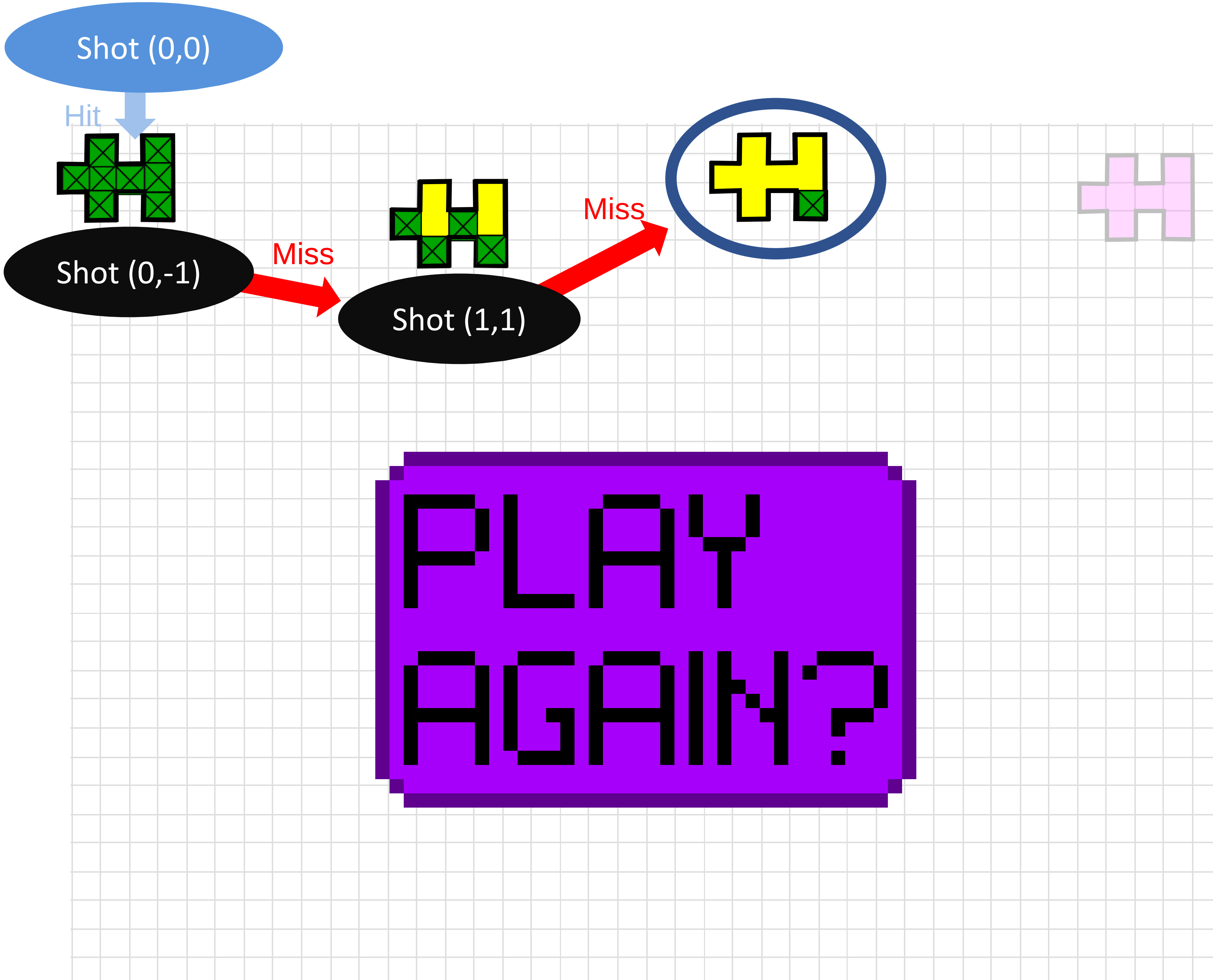
Shot (0,0)
Hit
Shot (0,-1)
Miss
Shot (1,1)
Miss

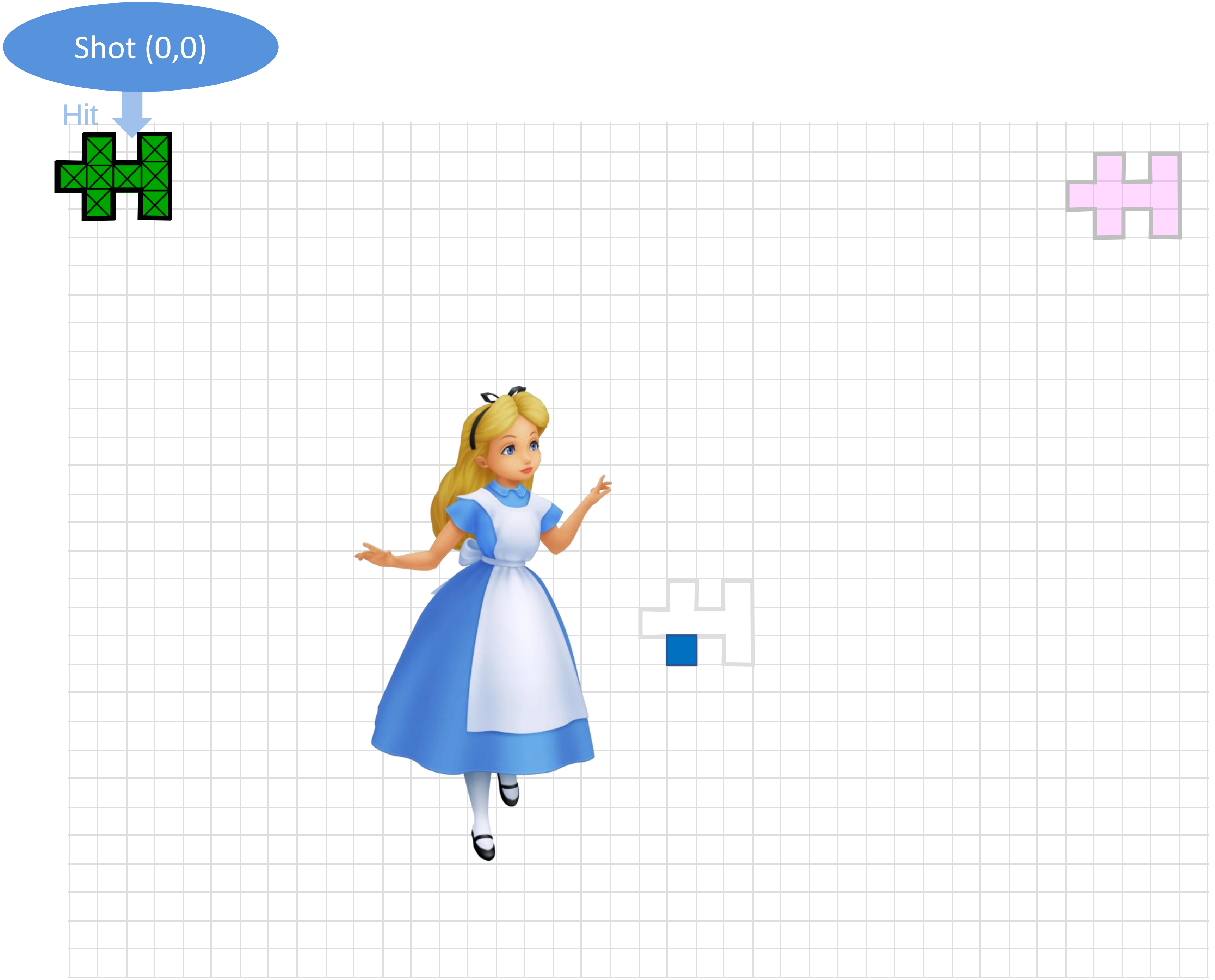


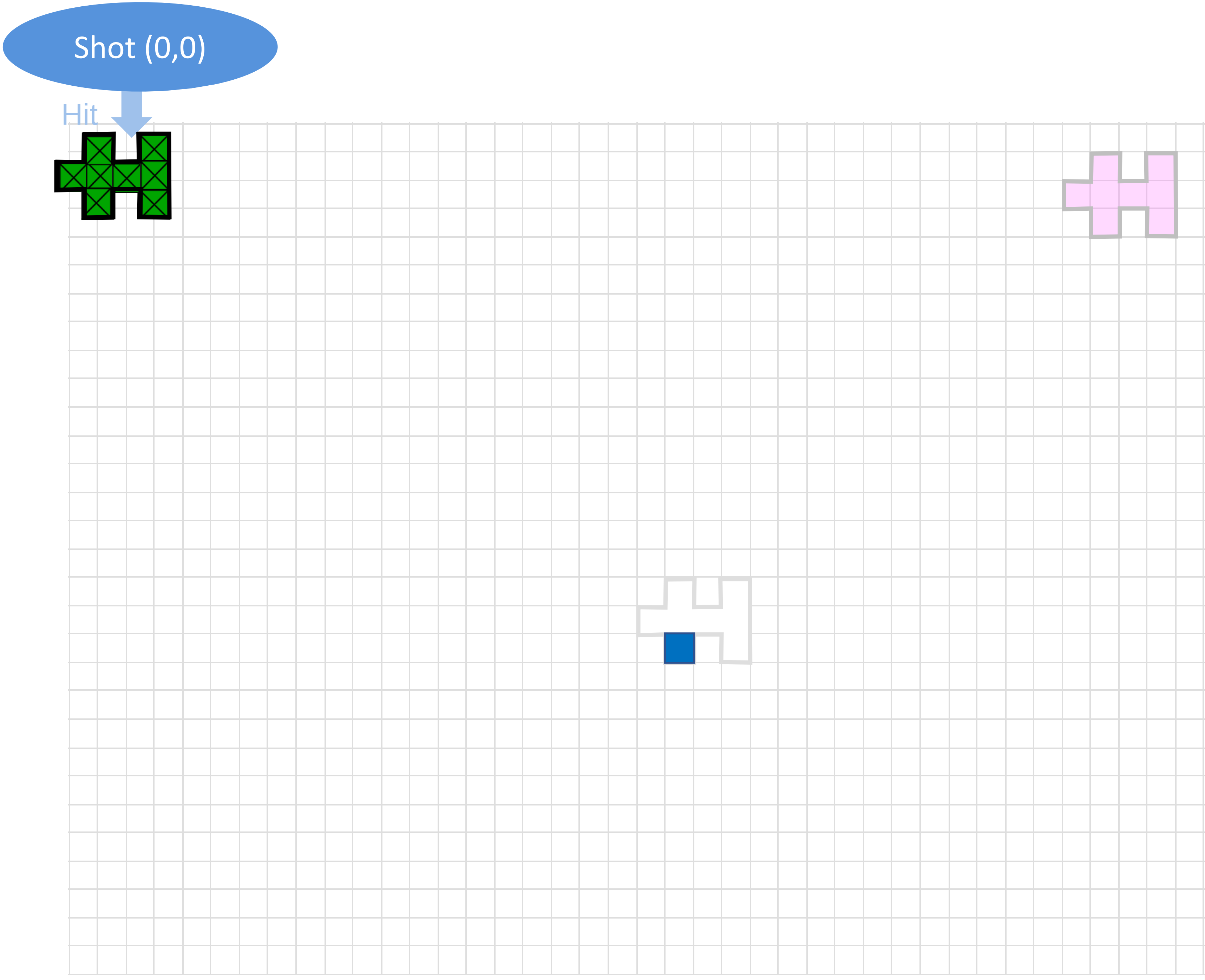
Shot (0,0)
Hit
Shot (0,-1)
Miss
Shot (1,1)
Miss

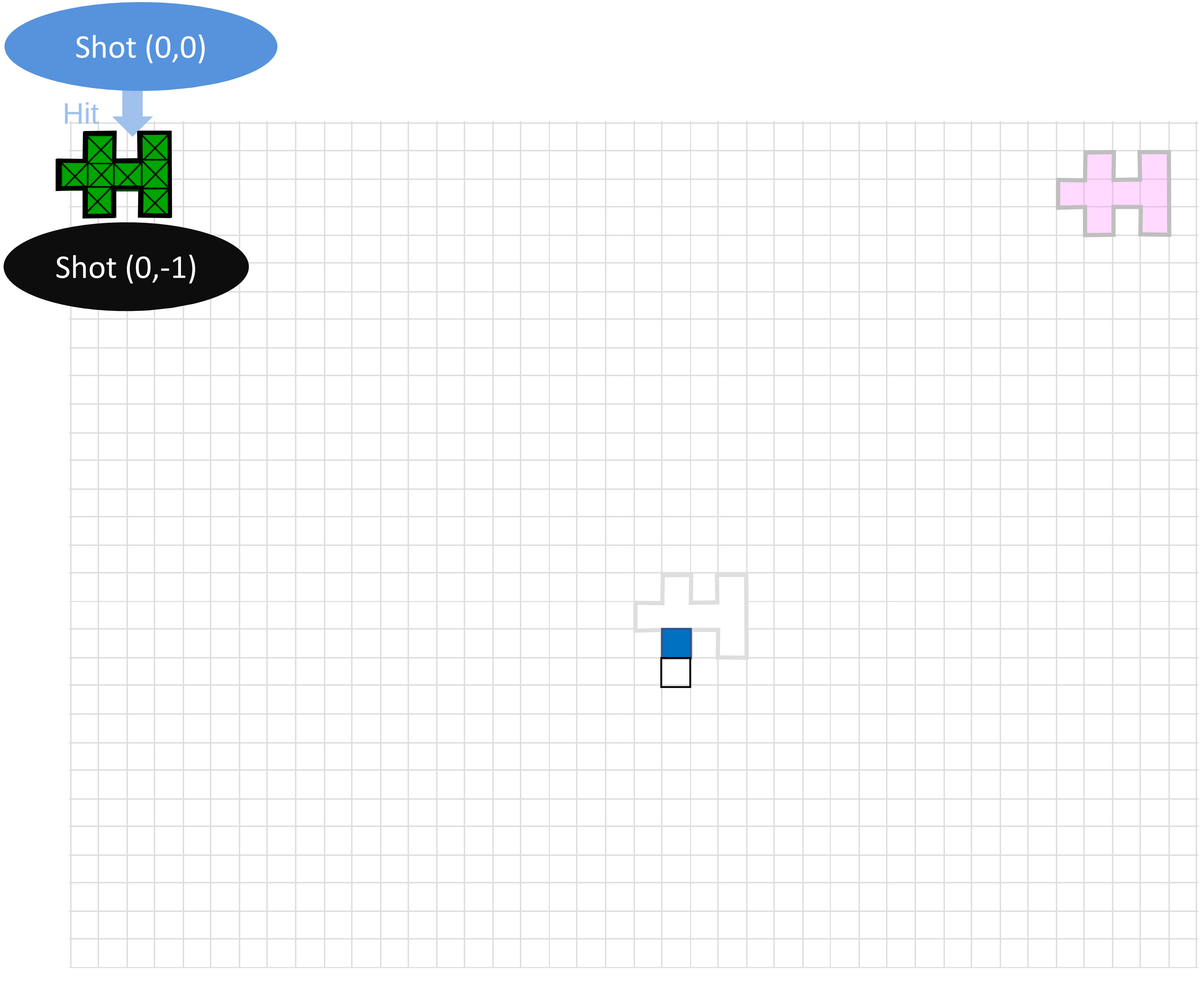


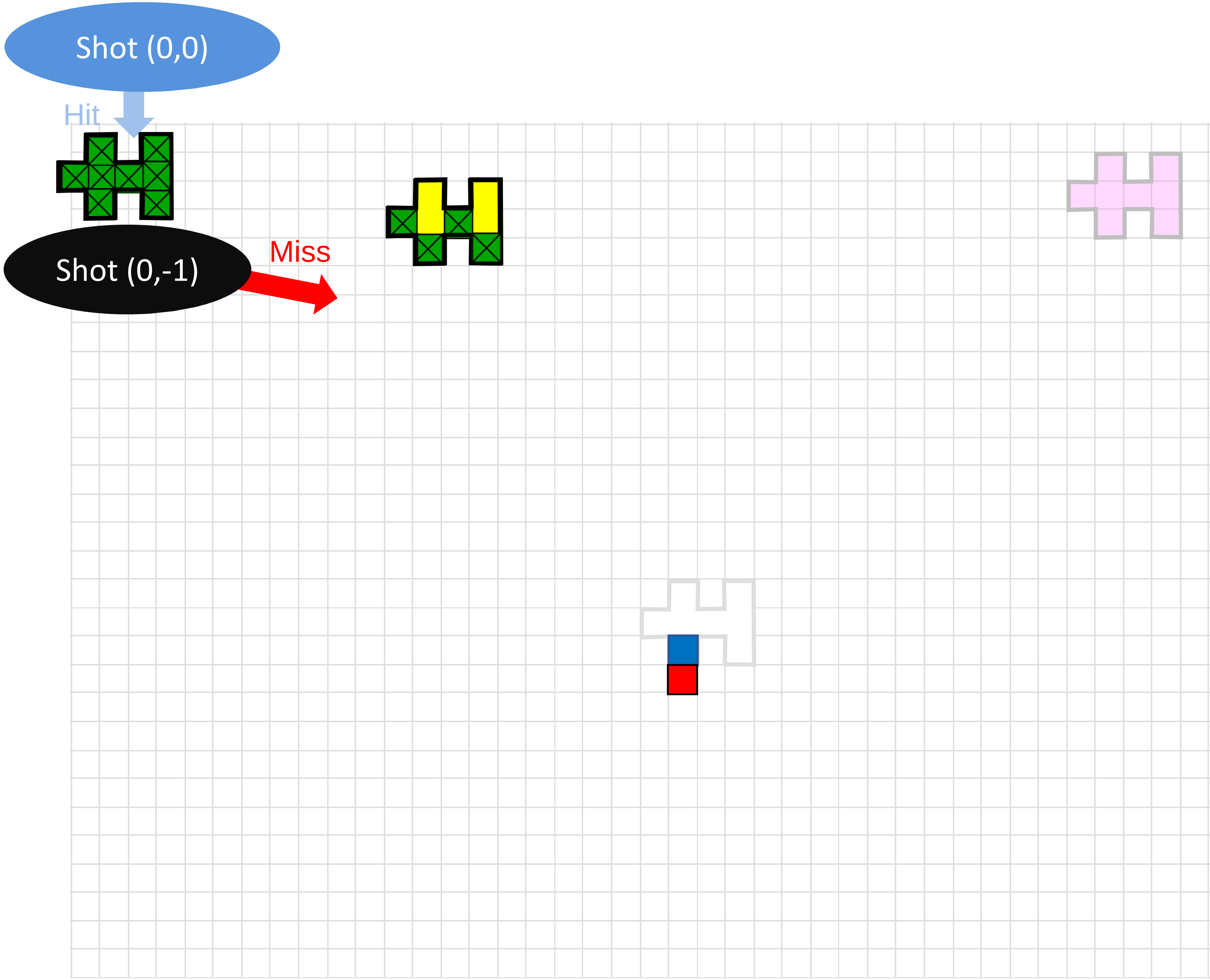
Shot (0,0)
Hit
Shot (0,-1)
Miss
Shot (1,1)
Miss
We found the position with only 2 misses…

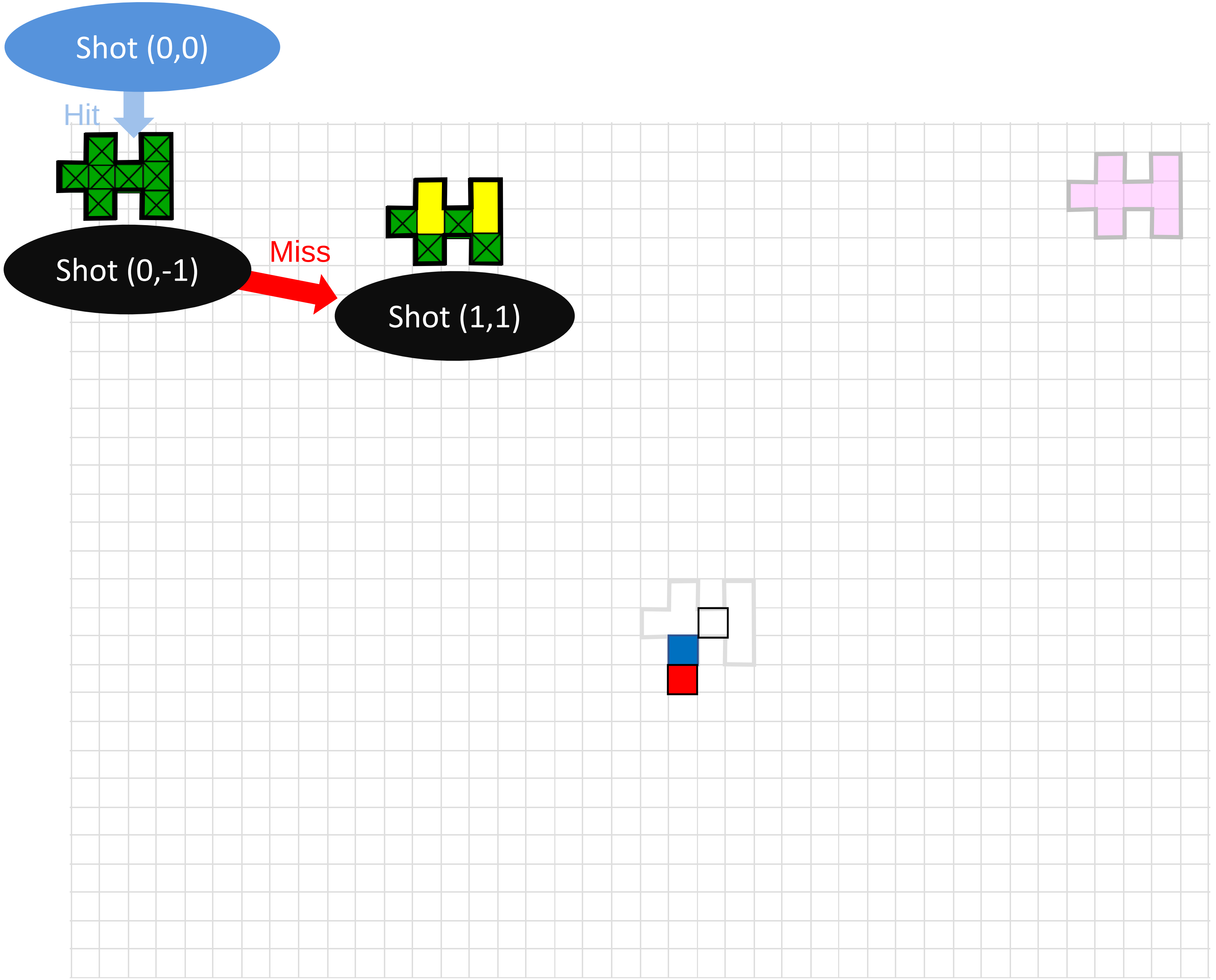

Shot (0,0)
Hit
Shot (0,-1)
Miss
Shot (1,1)
Miss
PLAY
AGAIN?

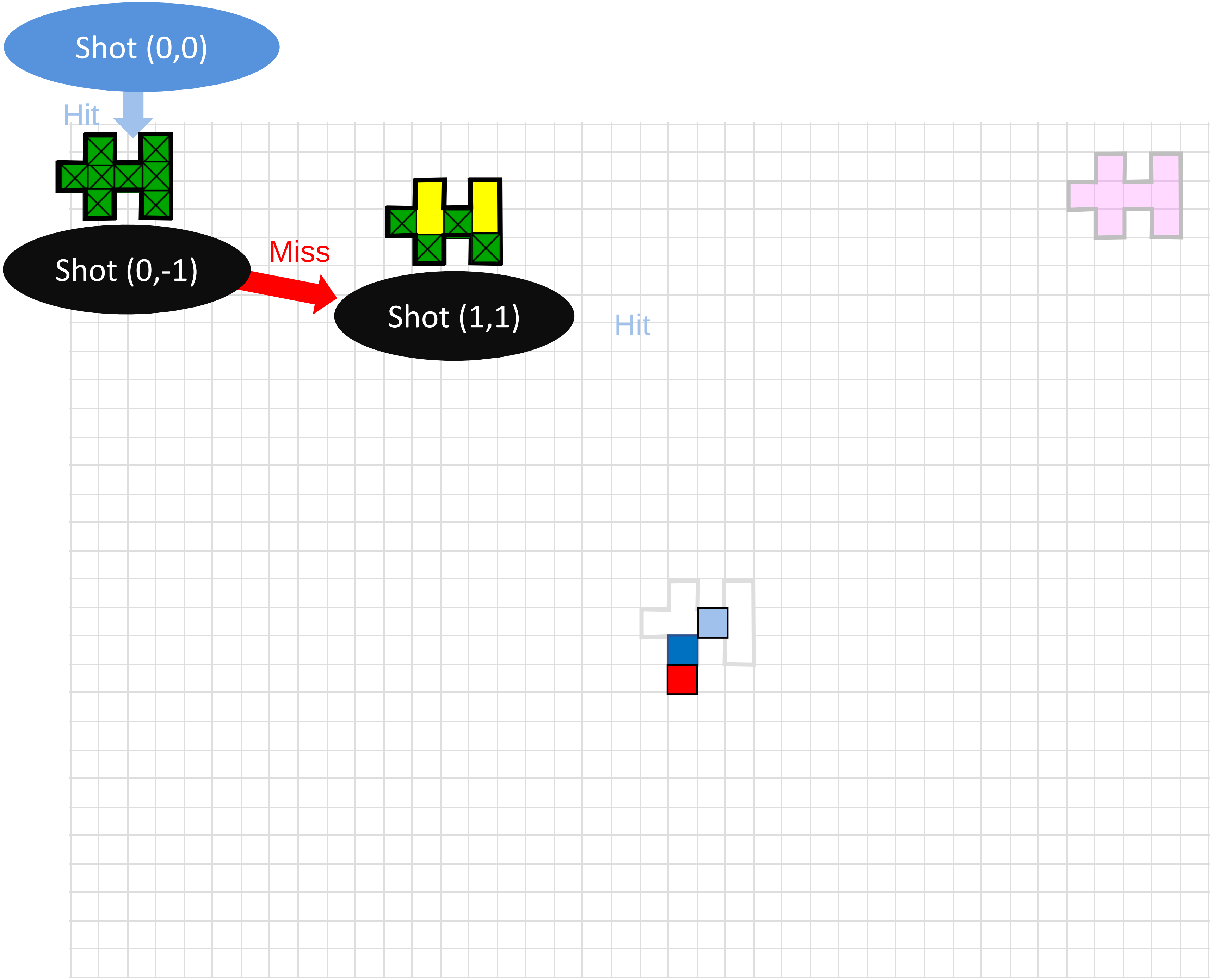

Shot (0,0)
Hit


Shot (0,0)
Hit

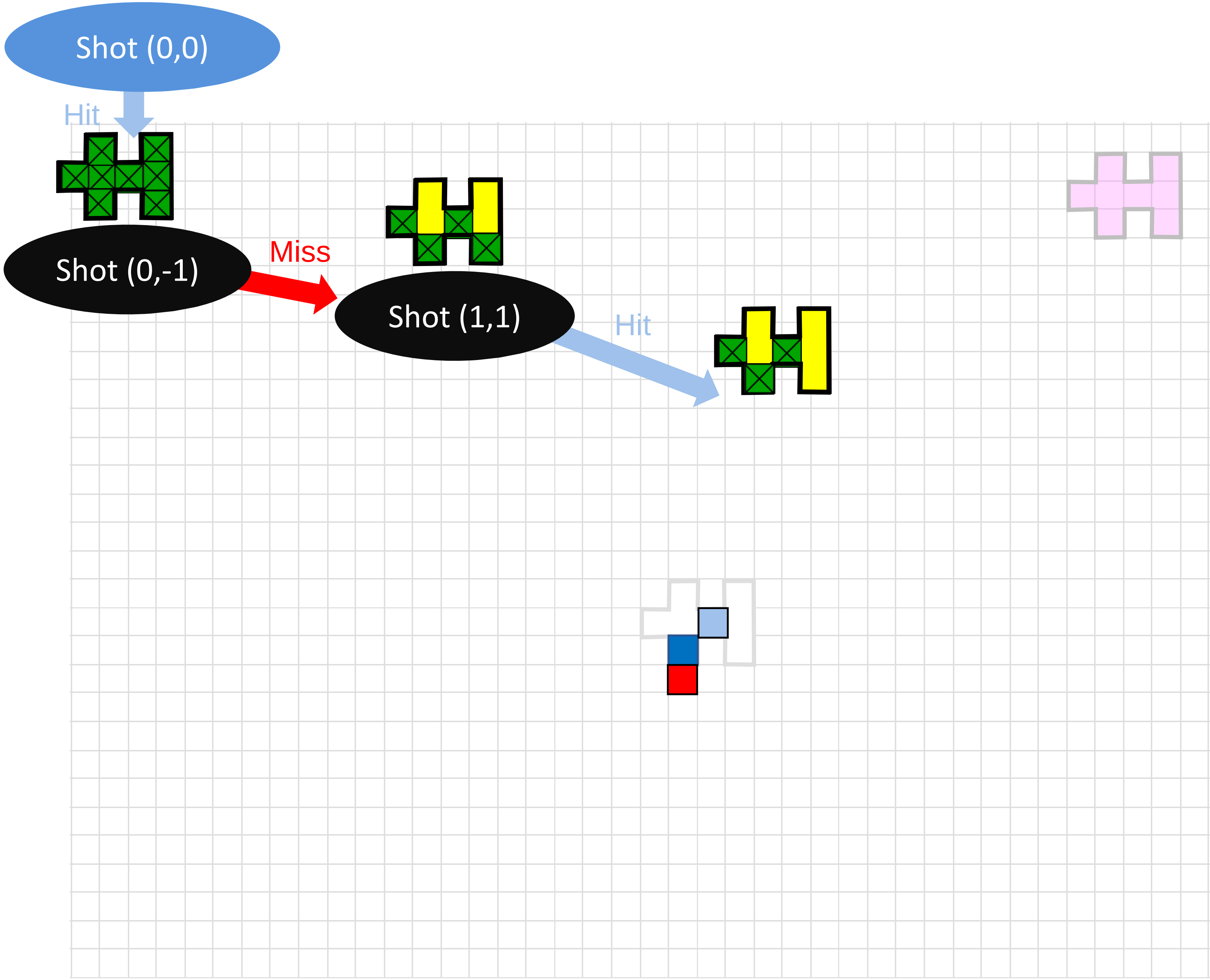

Shot (0,0)
Hit
Shot (0,-1)


Shot (0,0)
Hit
Shot (0,-1)
Miss

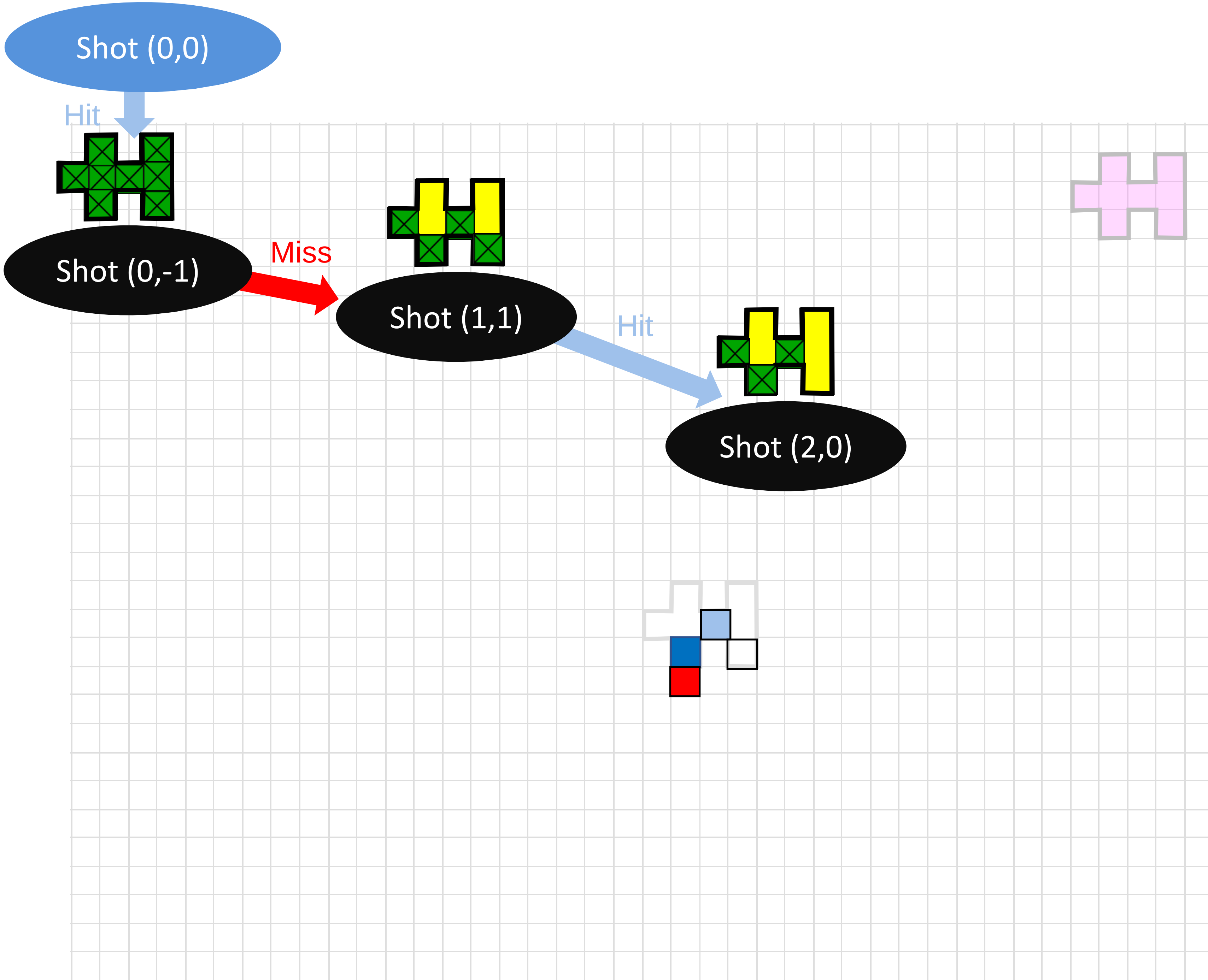

Shot (0,0)
Hit
Shot (0,-1)
Miss
Shot (1,1)

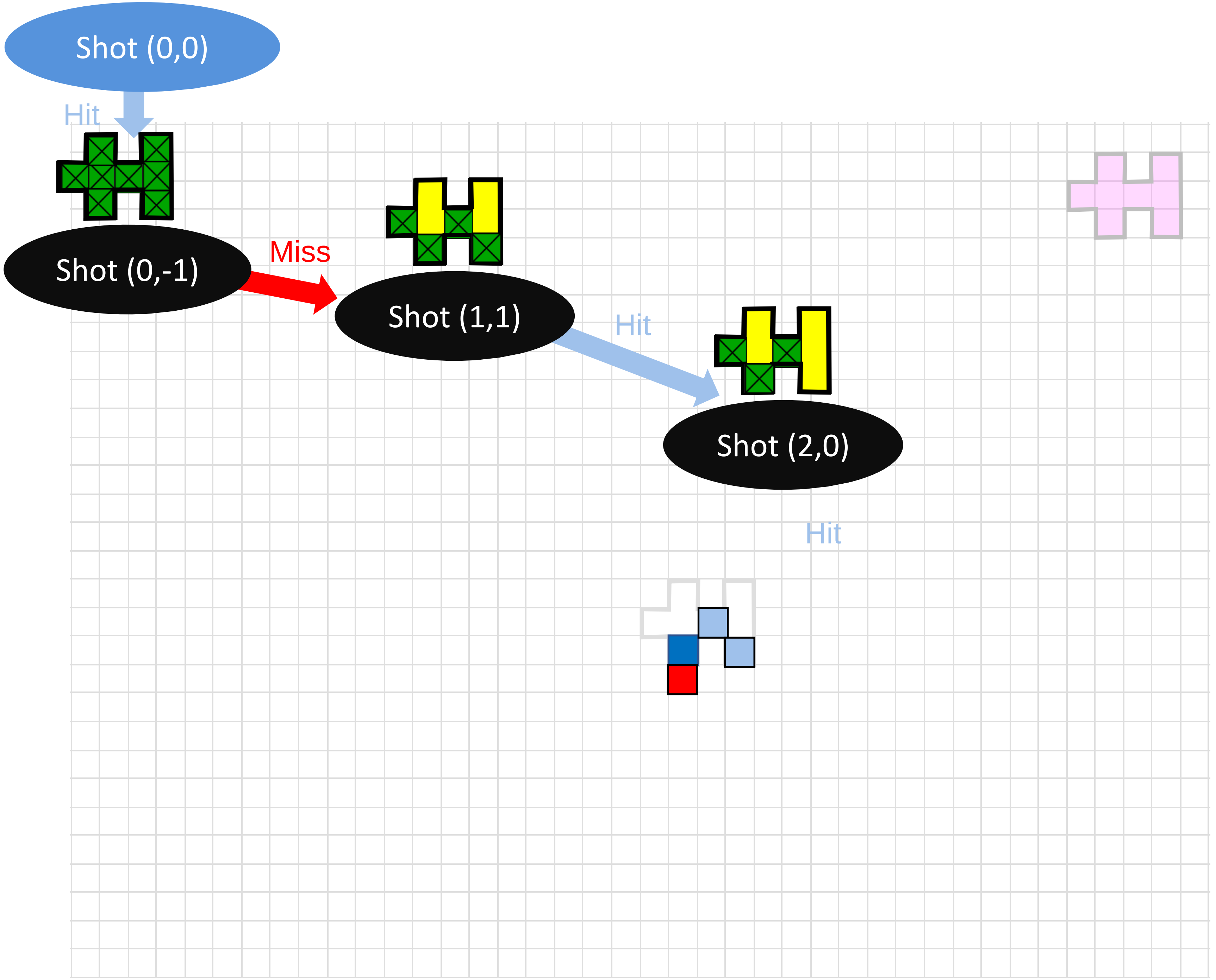


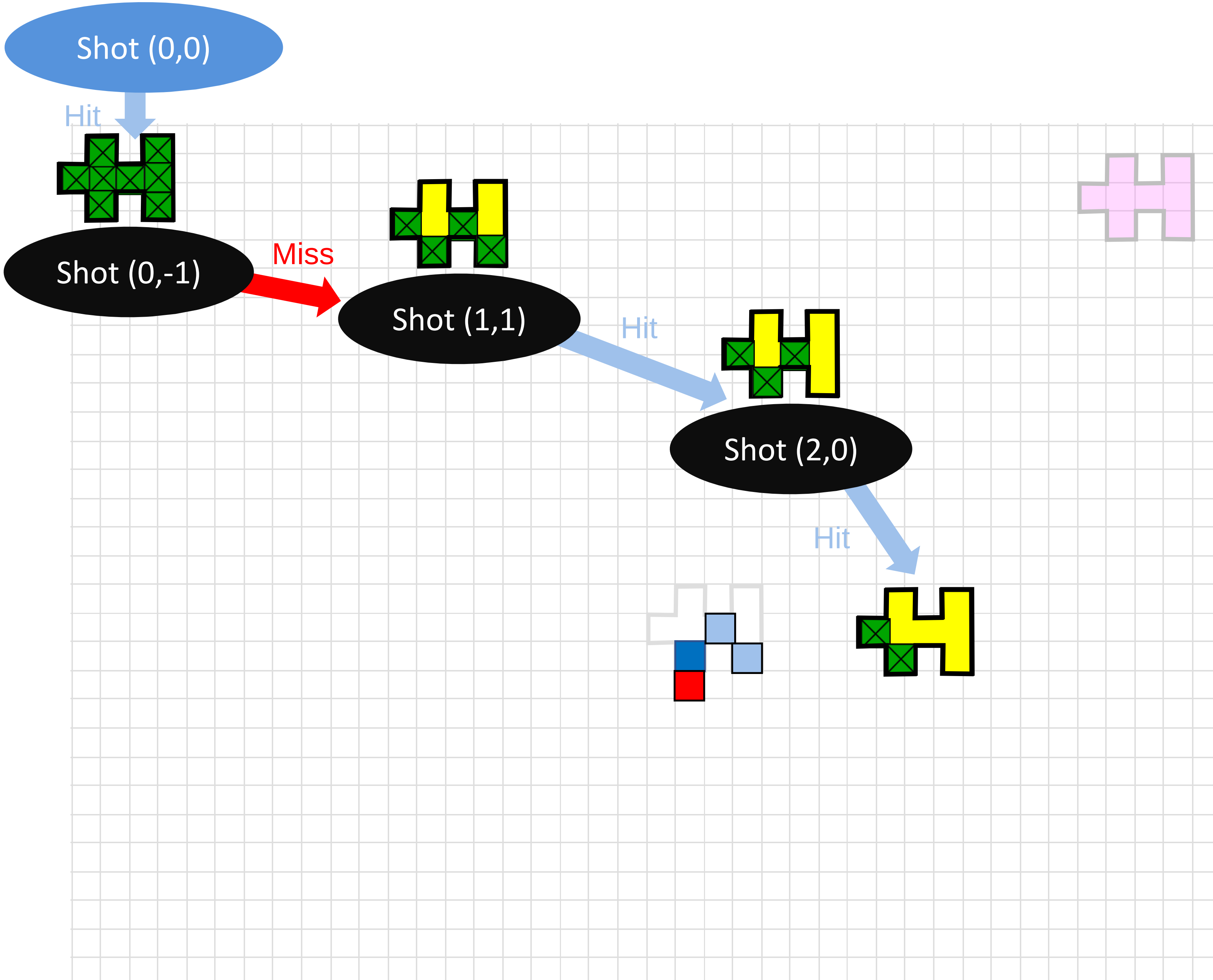
Shot (0,0)
Hit
Shot (0,-1)
Miss
Shot (1,1)
Hit

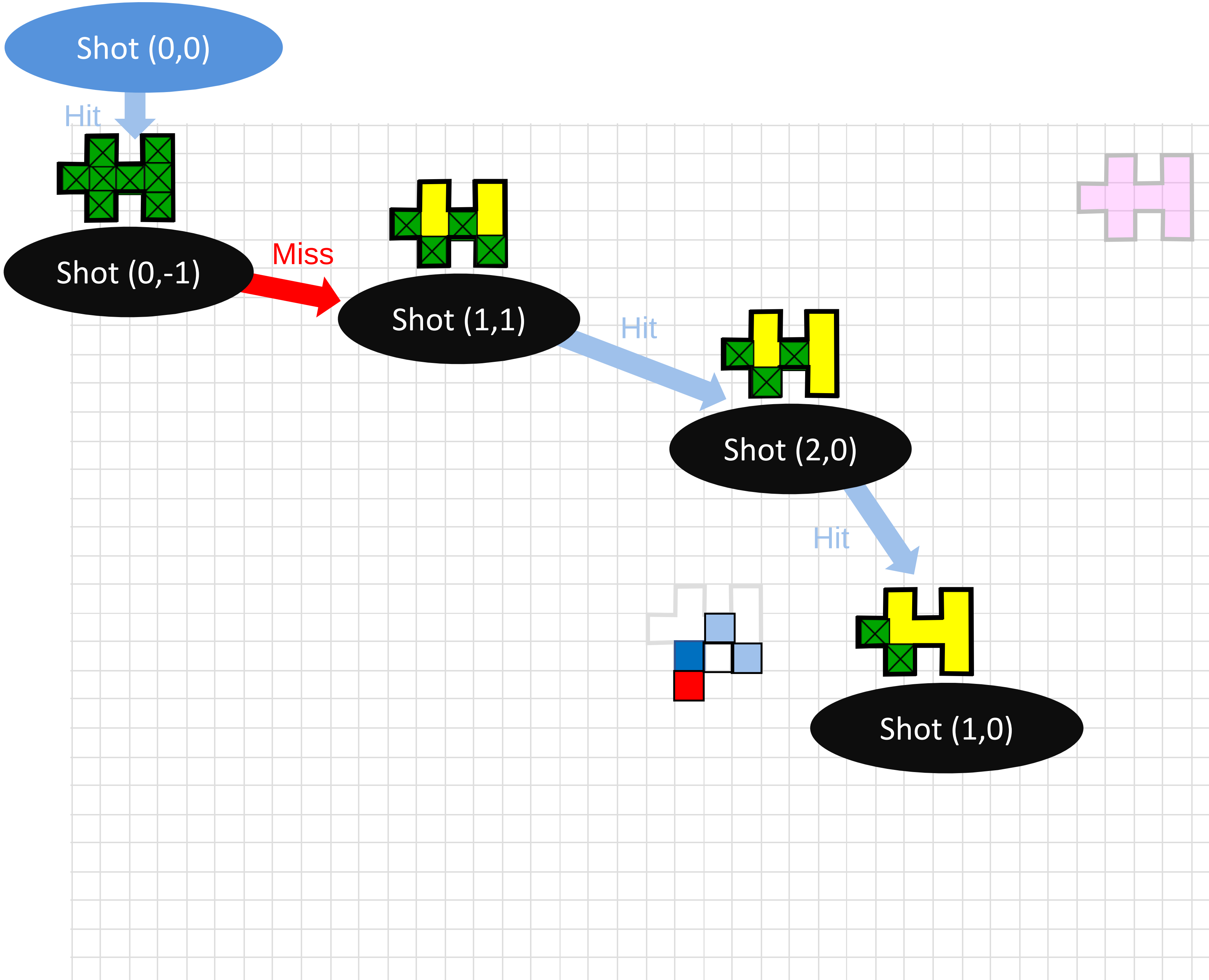


Shot (0,0)
Hit
Shot (0,-1)
Miss
Shot (1,1)
Hit

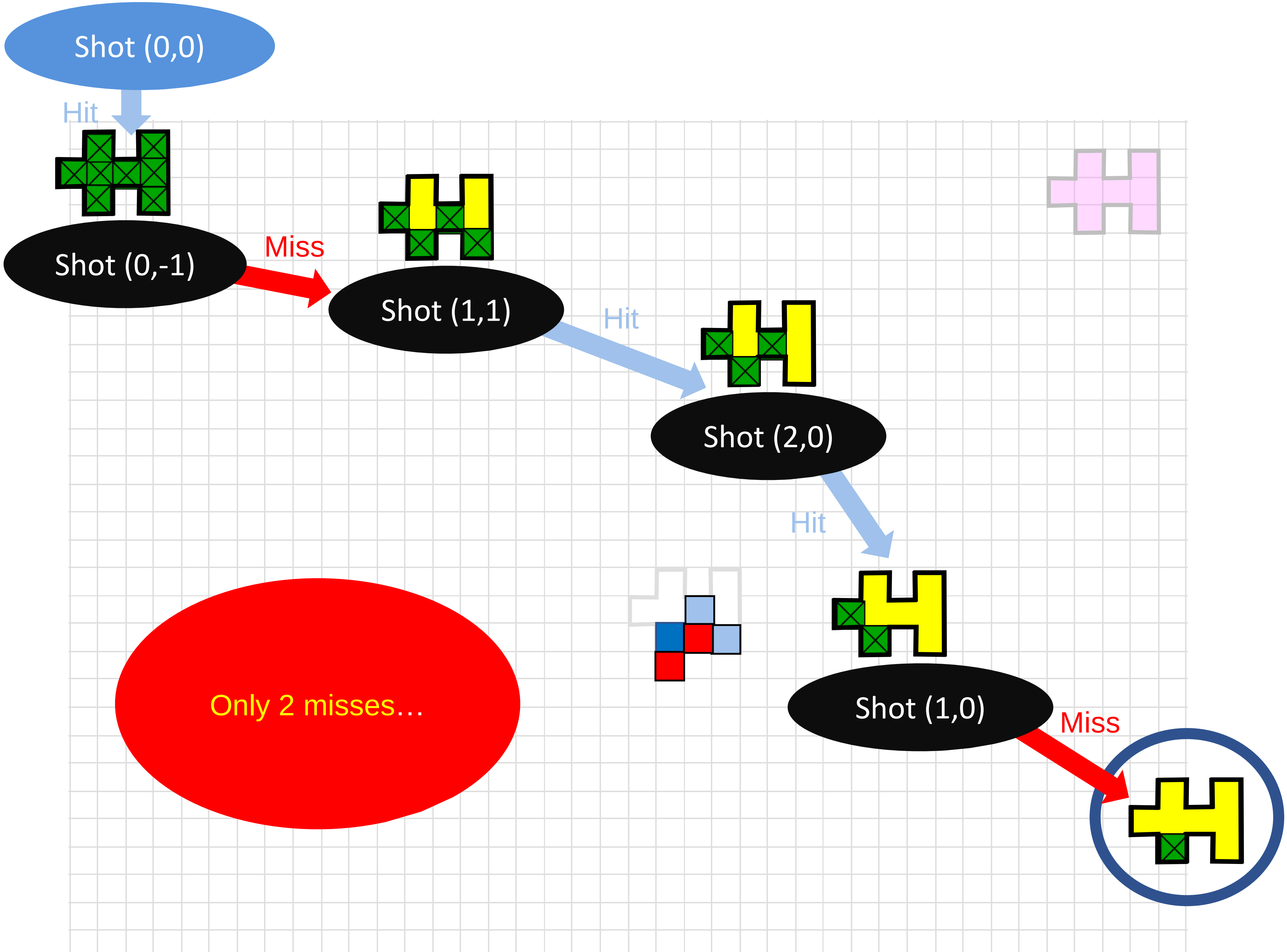

Shot (0,0)
Hit
Shot (0,-1)
Miss
Shot (1,1)
Hit
Shot (2,0)

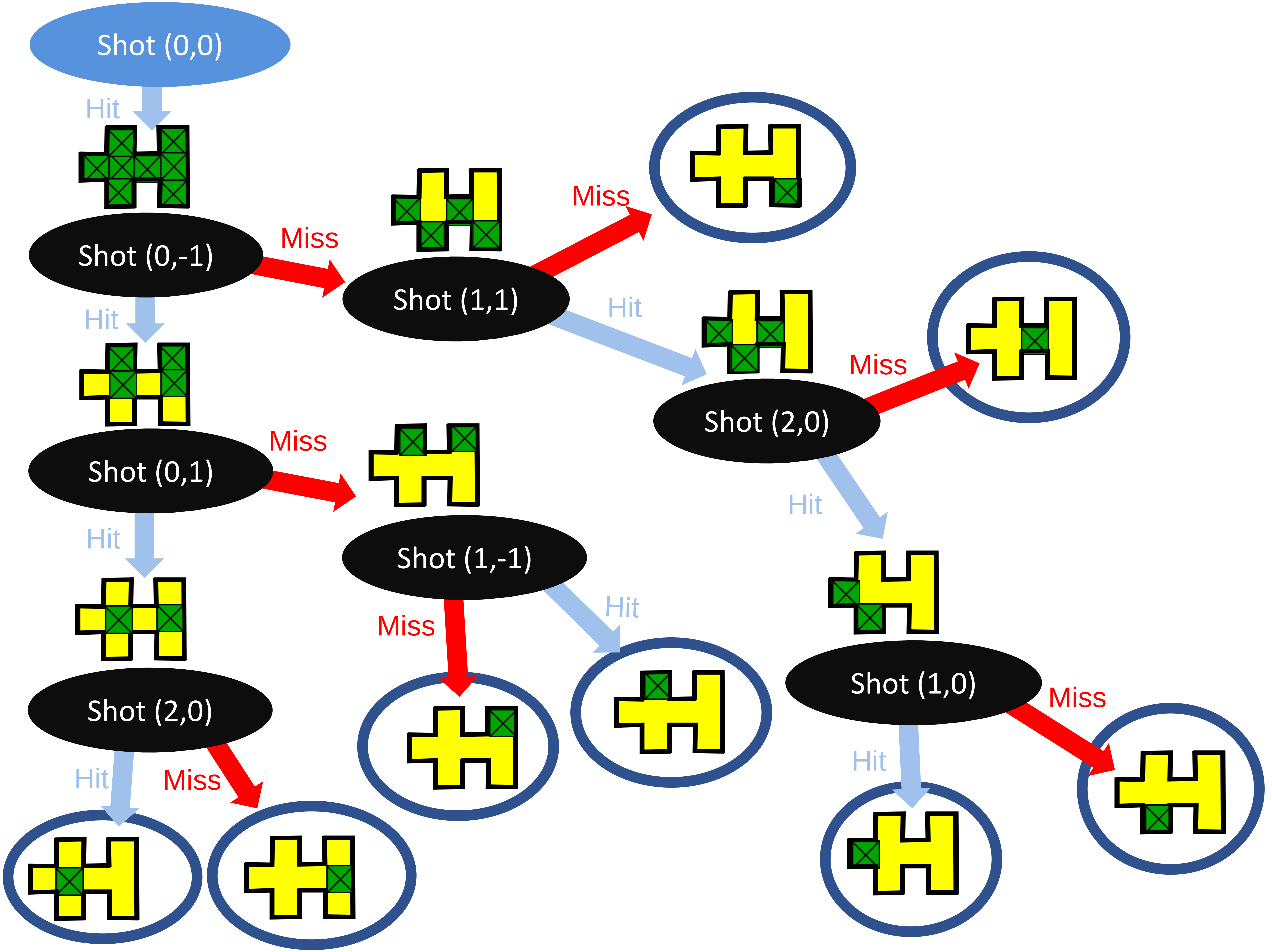

Shot (0,0)
Hit
Shot (0,-1)
Miss
Shot (1,1)
Hit
Shot (2,0)
Hit


Shot (0,0)
Hit
Shot (0,-1)
Miss
Shot (1,1)
Hit
Shot (2,0)
Hit

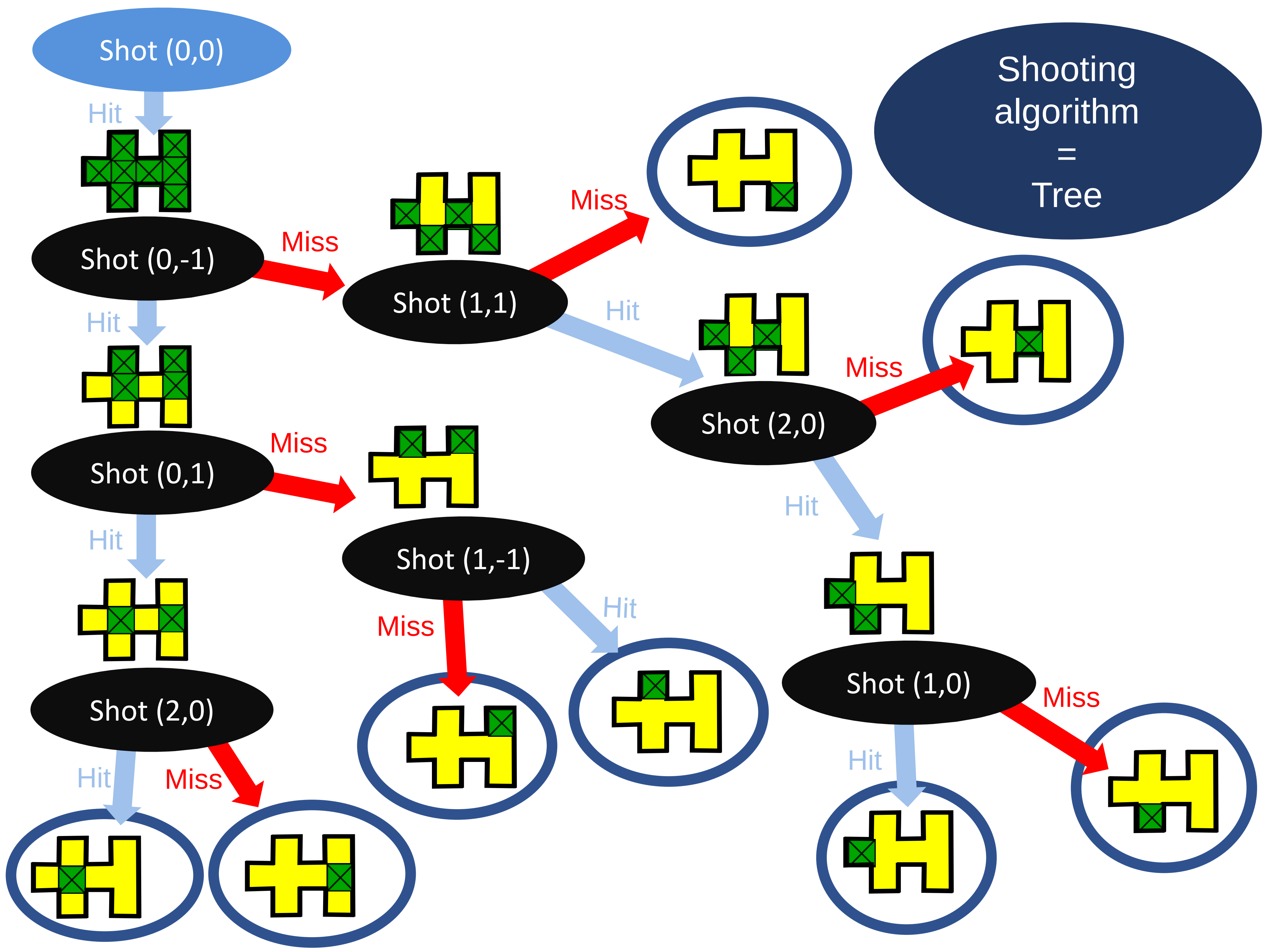


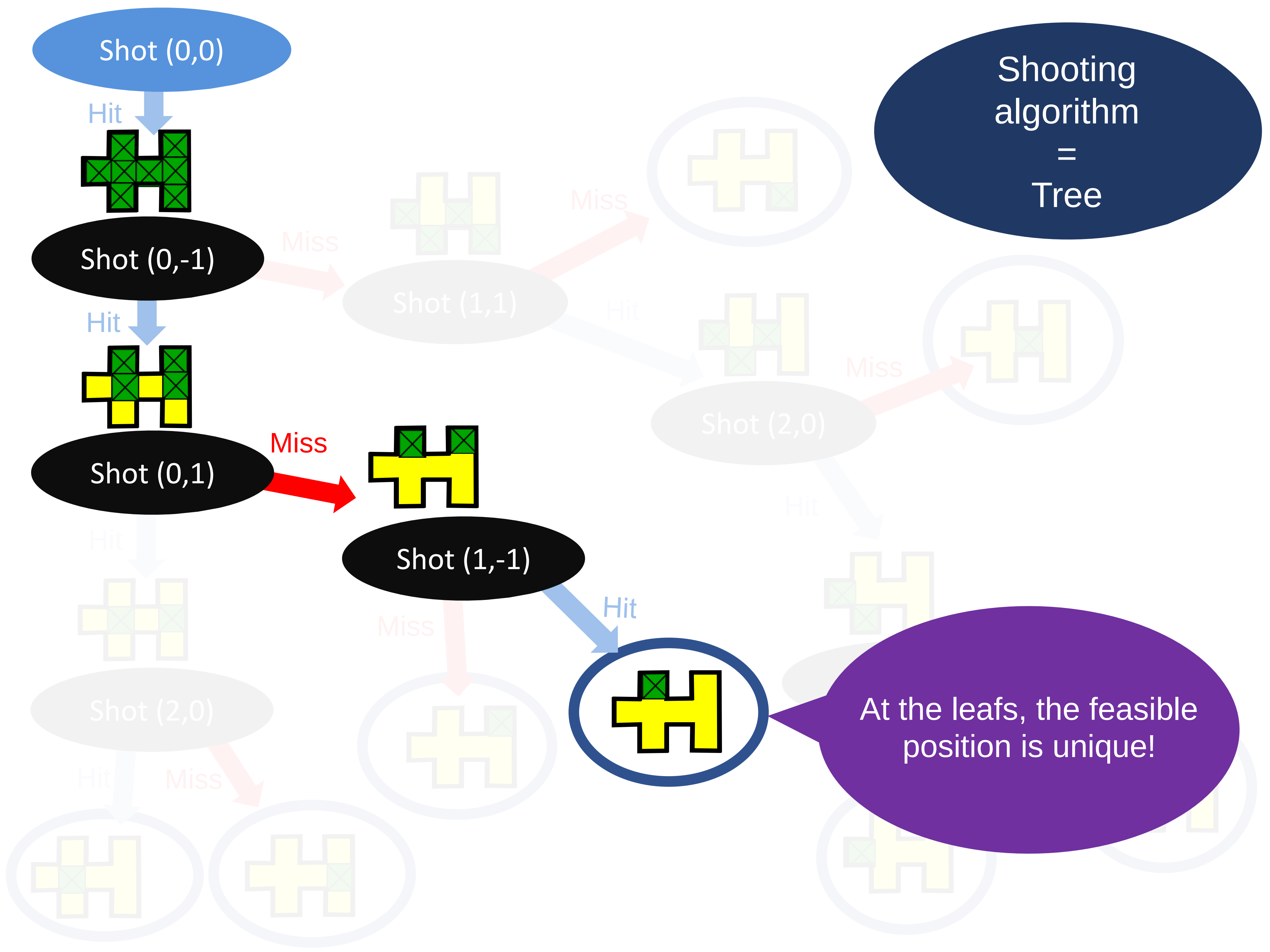
Shot (0,0)
Hit
Shot (0,-1)
Miss
Shot (1,1)
Hit
Shot (2,0)
Hit
Shot (1,0)


Shot (0,0)
Hit
Shot (0,-1)
Miss
Shot (1,1)
Hit
Shot (2,0)
Hit
Shot (1,0)
Miss
Only 2 misses…

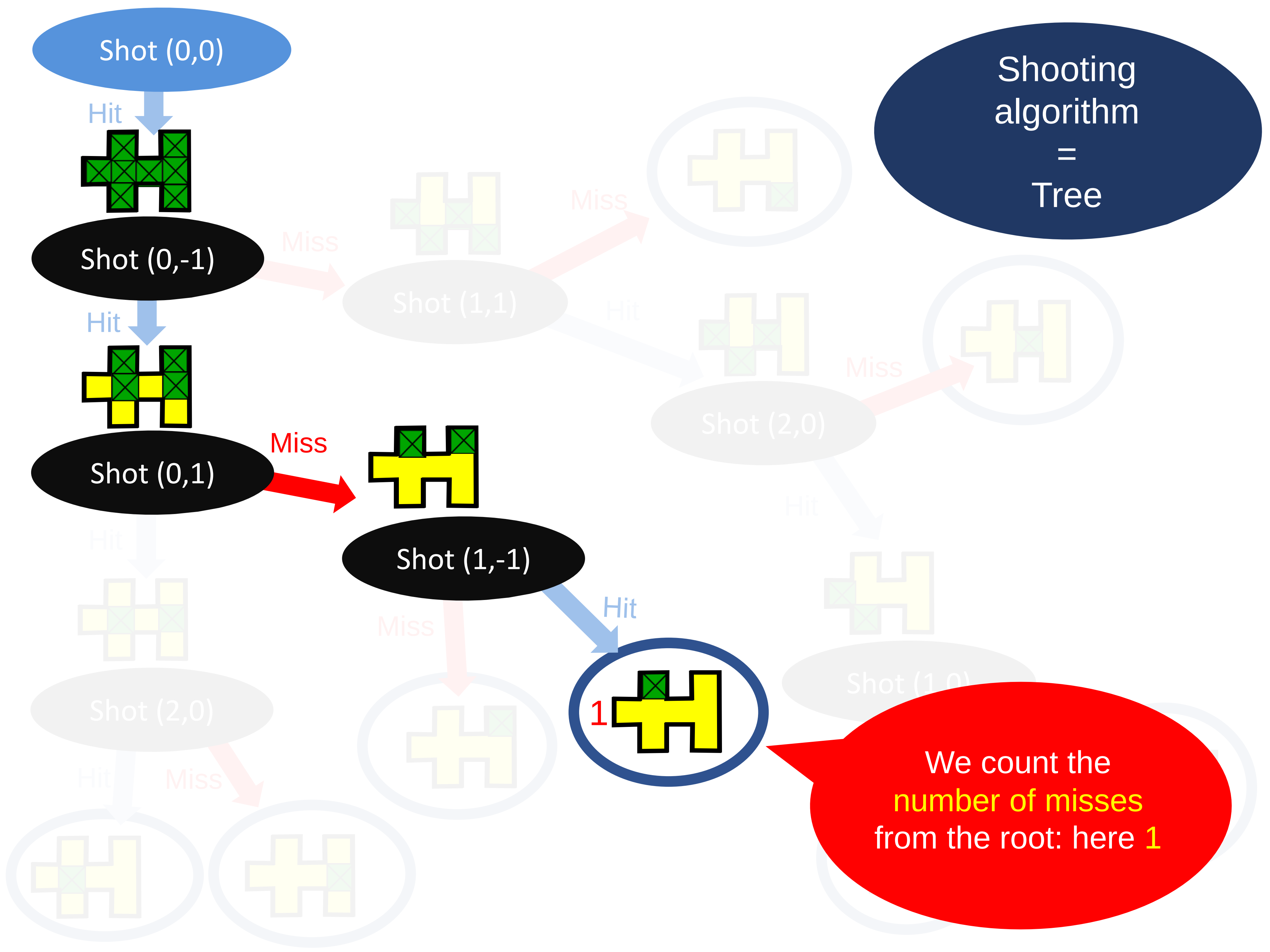


Shot (0,0)
Hit
Shot (0,-1)
Miss
Shot (1,1)
Miss
Hit
Shot (2,0)
Miss
Hit
Shot (1,0)
Miss
Hit
Hit
Shot (0,1)
Miss
Shot (1,-1)
Miss
Hit
Hit
Shot (2,0)
Hit
Miss

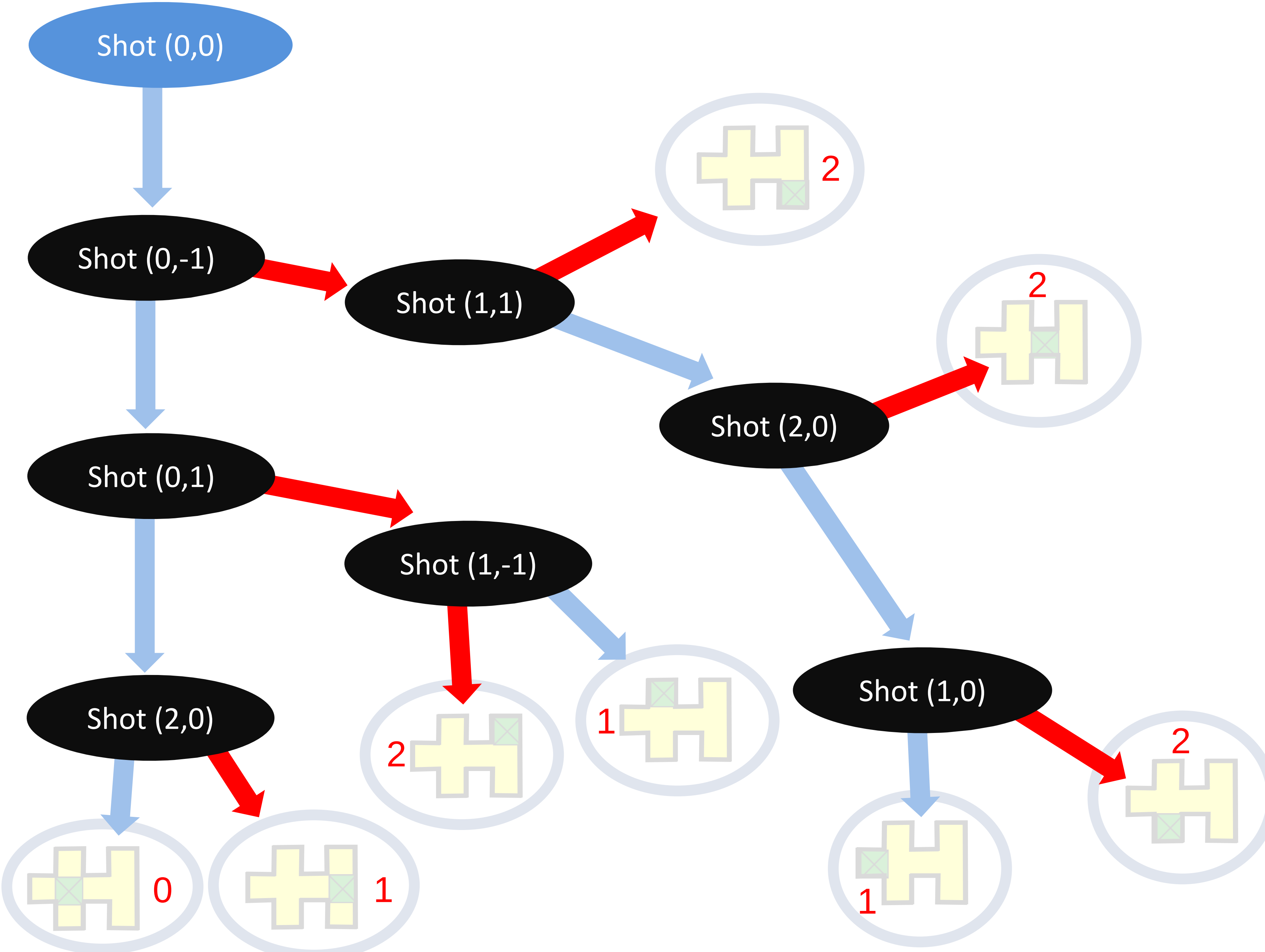

Shot (0,0)
Hit
Shot (0,-1)
Miss
Shot (1,1)
Miss
Hit
Hit
Shot (0,1)
Miss
Shot (2,0)
Miss
Hit
Hit
Shot (1,-1)
Miss
Hit
Shot (1,0)
Miss
Hit
Shot (2,0)
Hit
Miss
Shooting
algorithm
=
Tree

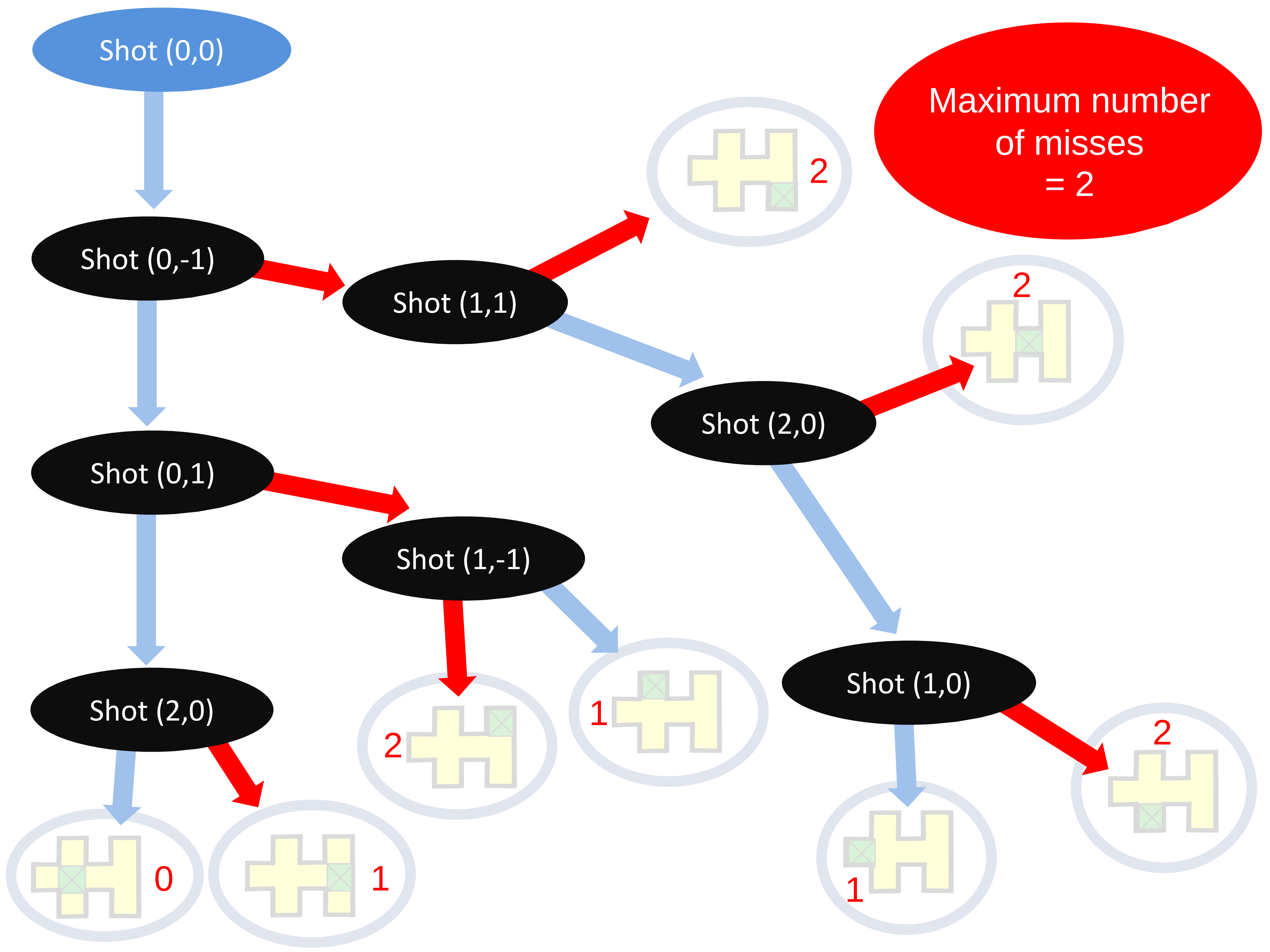


Shot (0,0)
Hit
Shot (0,-1)
Miss
Hit
Shot (0,1)
Miss
Shot (1,-1)
Hit
Shooting algorithm = Tree
At the leafs, the feasible position is unique!
Shot (1,1)
Shot (2,0)

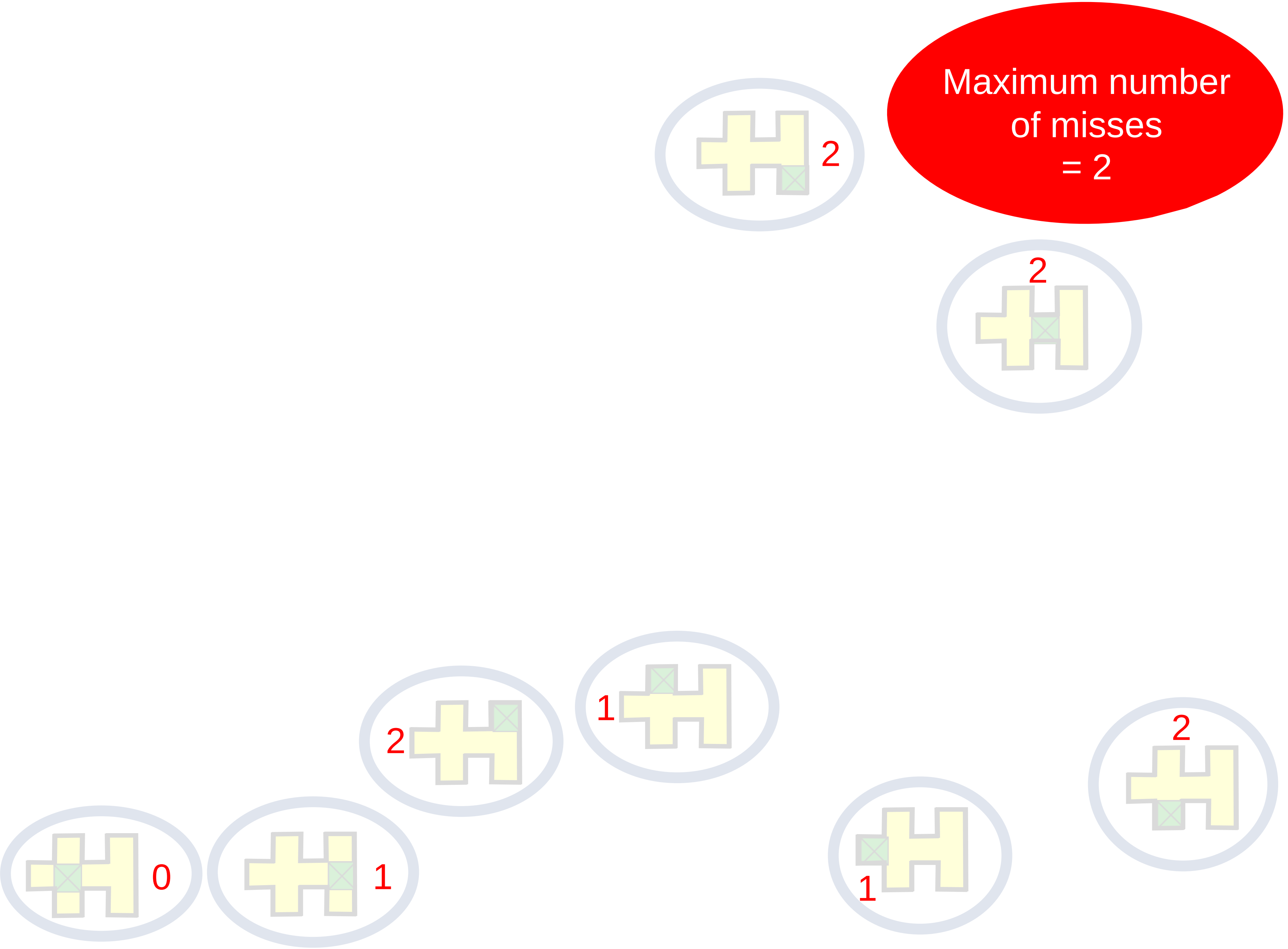


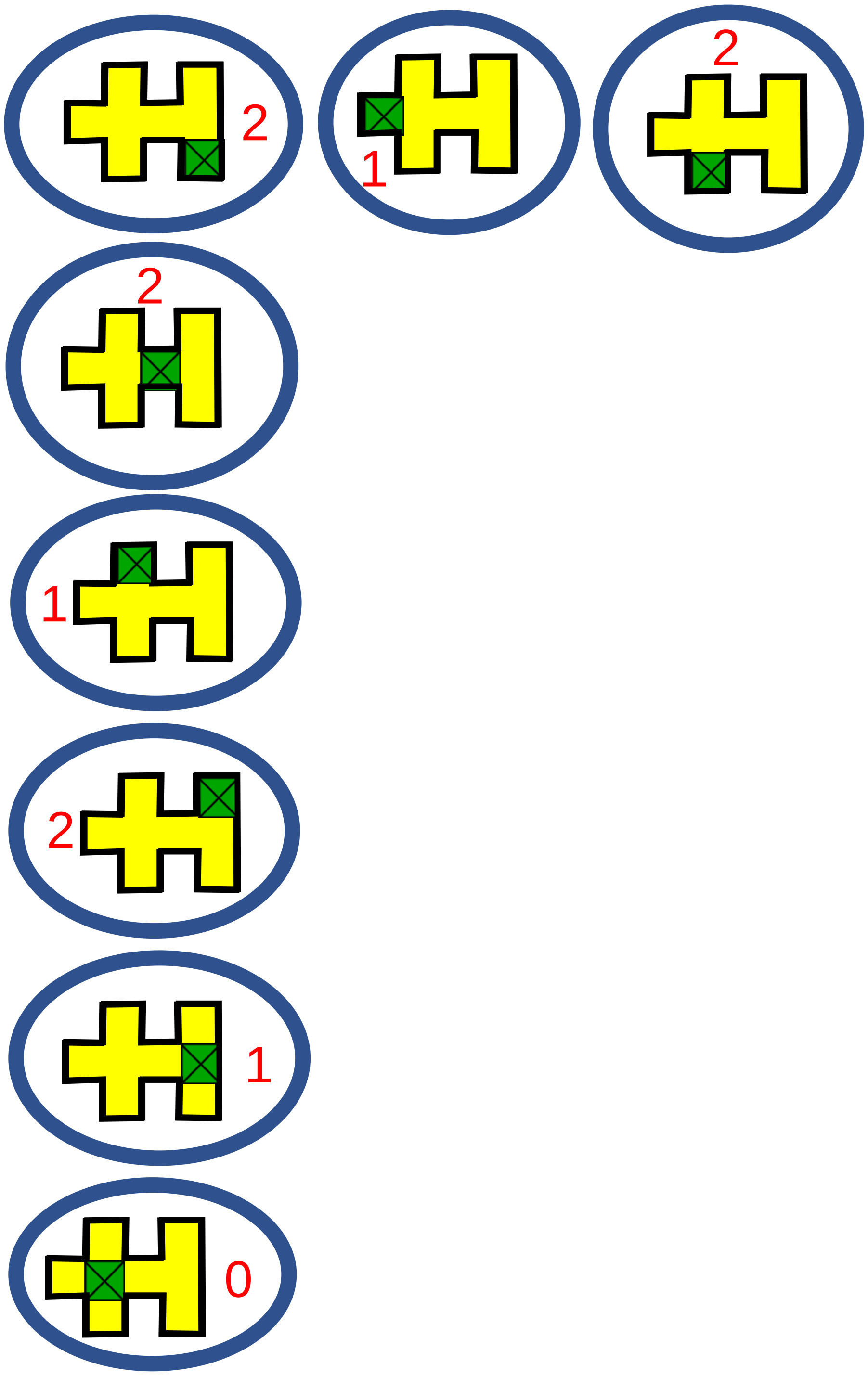
Shot (0,0)
Hit
Shot (0,-1)
Miss
Hit
Shot (0,1)
Miss
Shot (1,-1)
Hit
1
Shot (1,1)
Miss
Hit
Shot (2,0)
Miss
Shot (2,0)
Miss
Shooting algorithm = Tree
We count the
number of misses
from the root: here 1


Shot (0,0)
Shot (0,-1)
Shot (1,1)
2
2
Shot (2,0)
Shot (0,1)
Shot (1,-1)
Shot (1,0)
Shot (2,0)
2
1
2
0
1
1

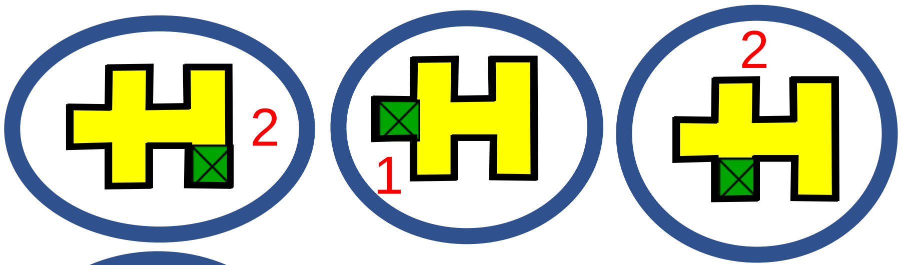


Shot (0,0)
Shot (0,-1)
Shot (1,1)
2
Maximum number of misses = 2
2
Shot (2,0)
Shot (0,1)
Shot (1,-1)
Shot (2,0)
2
1
Shot (1,0)
2
0
1
1

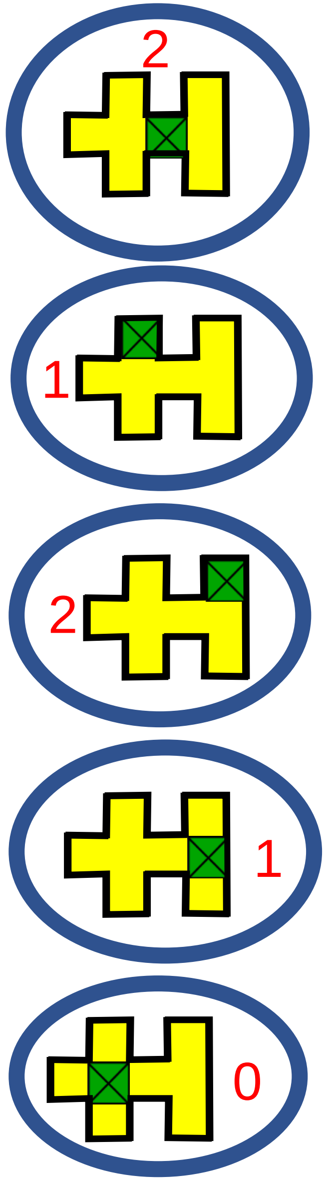

Maximum number of misses = 2
2
2
2
1
2
0
1
1

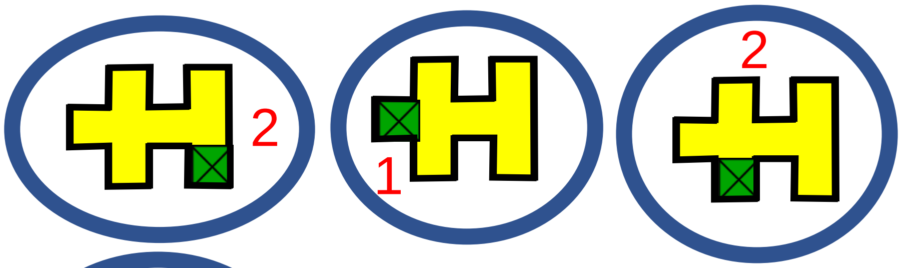


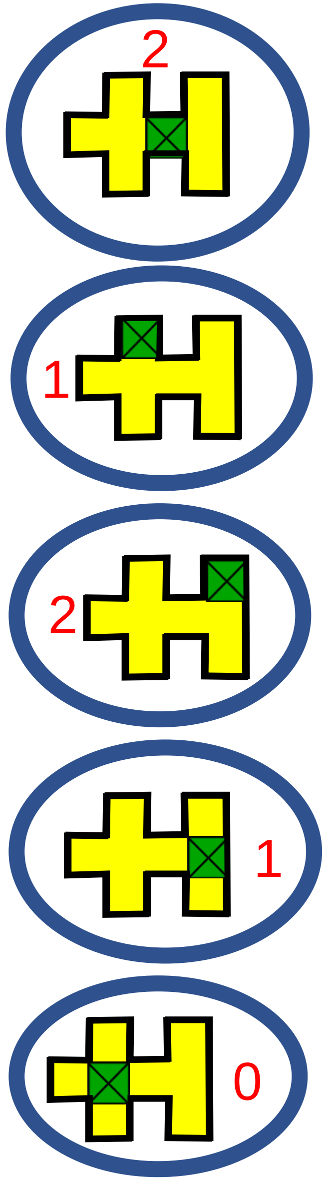
Maximum number of misses = 2



What's an efficient shooting algorithm ?
Which efficiency measure should we use ?



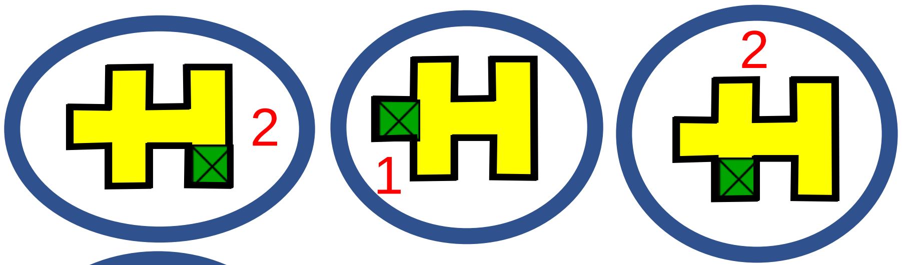


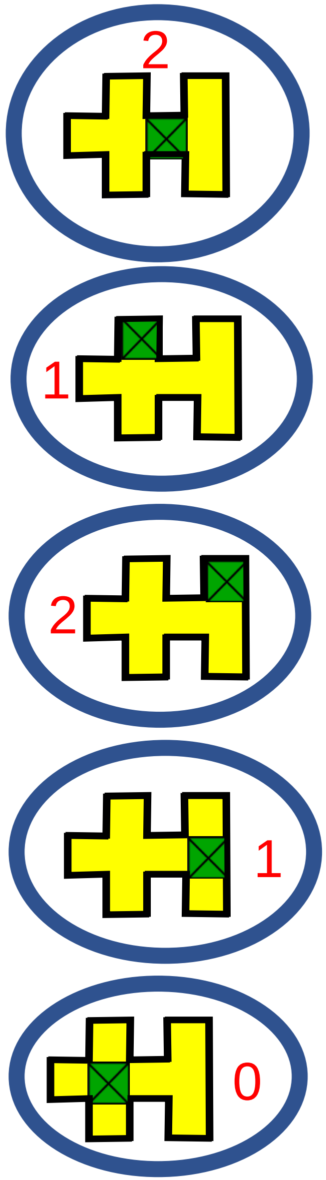


What's an efficient shooting algorithm ?
Which efficiency measure should we use ?

the average ?

$$\frac{0+1+2+1+2+2+1+2}{8} = 1{,}375$$



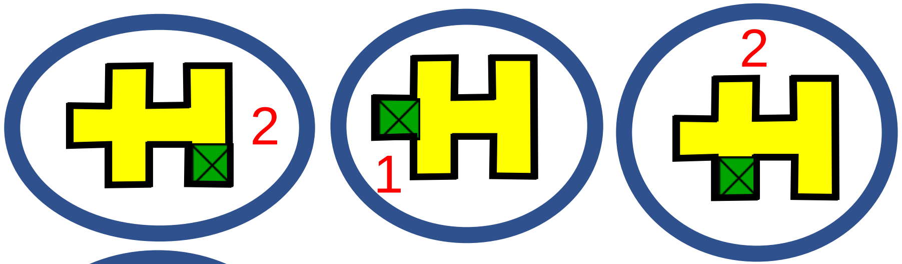


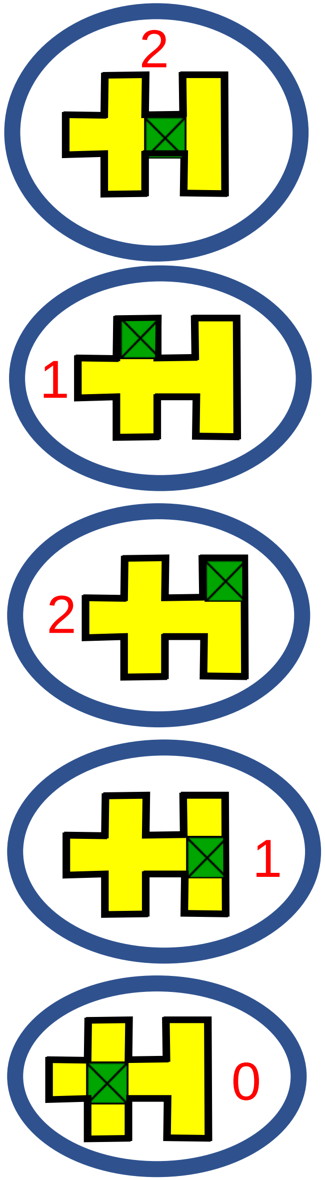
What's an efficient shooting algorithm ?
Which efficiency measure should we use ?

the average ?

$$\frac{0+1+2+1+2+2+1+2}{8}=1{,}375$$

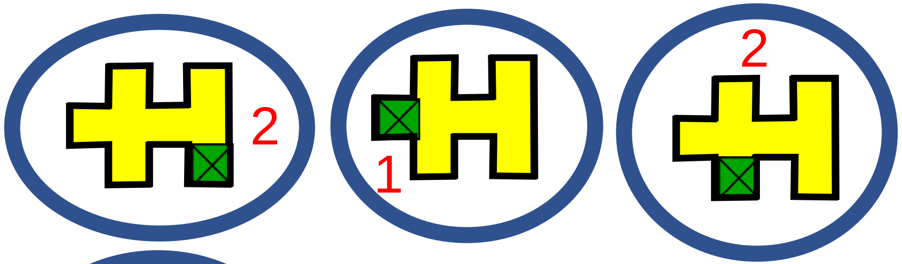
Alice is not a
« uniform random player »

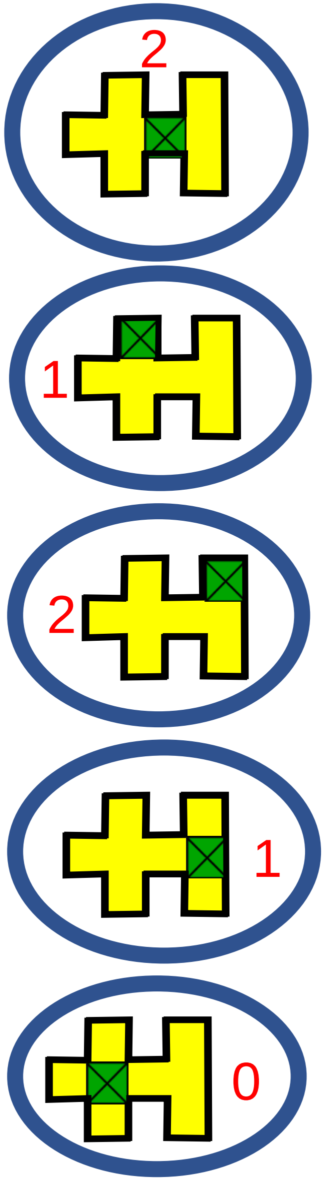




What’s an efficient shooting algorithm ?
Which efficiency measure should we use ?

Alice is not a
« uniform random player »

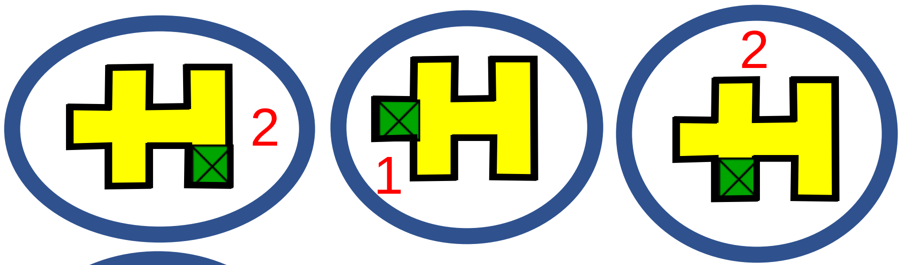


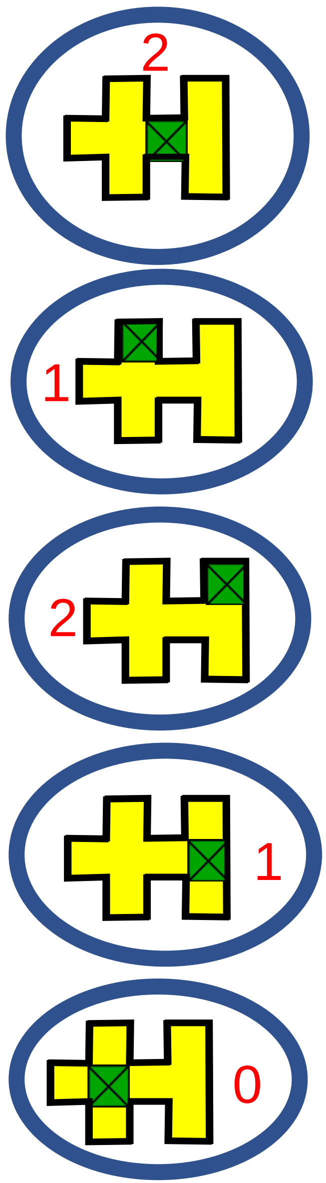


What’s an efficient shooting algorithm ?
Which efficiency measure should we use ?

If Alice knows our algorithm or
if she is cheating,
she will always choose
our worst case…

Alice is not a
« uniform random player »



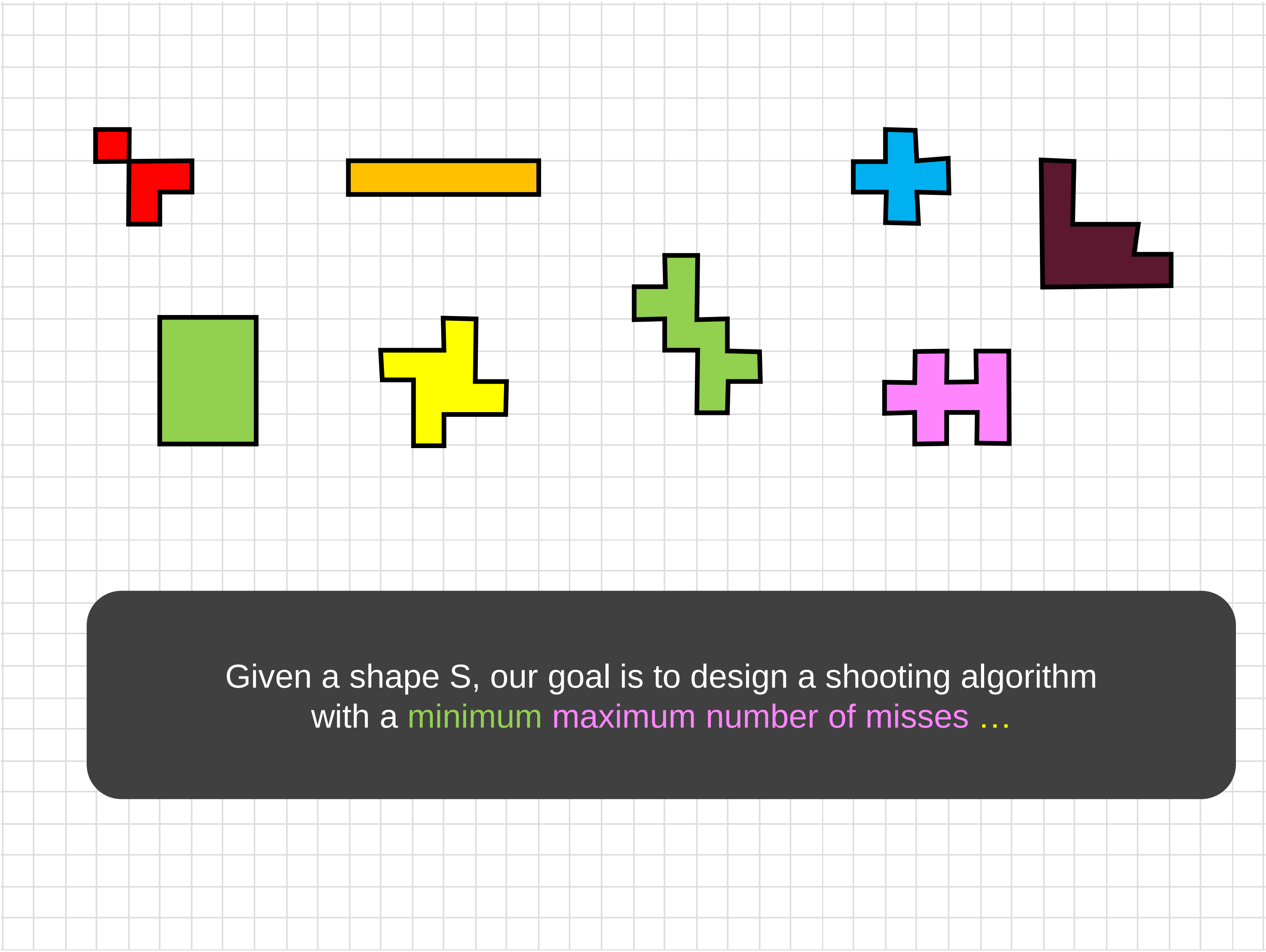


What's an efficient shooting algorithm ?
Which efficiency measure should we use ?

~~the average ?~~

$$\cancel{\frac{0+1+2+1+2+2+1+2}{8}} = 1{,}375$$

Worst case
=
Maximum number of misses



Given a shape S, our goal is to design a shooting algorithm
with a minimum maximum number of misses …

Min Max Problem

Given a shape S, our goal is to design a shooting algorithm
with a minimum maximum number of misses …

For a given algorithm

$$\text{Min} \quad \underbrace{\text{Max} \quad \text{Number of misses}}$$

Over all positions of the shape

Given a shape S, our goal is to design a shooting algorithm with a minimum maximum number of misses …

For a given algorithm

$\text{Min} \quad \underbrace{\text{Max} \quad \text{Number of misses}}$

Over all shooting algorithms

Over all positions of the shape

Given a shape S, our goal is to design a shooting algorithm with a minimum maximum number of misses …

Battleship Compexity(S) = Min Max Number of misses

Over all shooting algorithms

Over all positions of the shape

Given a shape S, our goal is to design a shooting algorithm with a minimum maximum number of misses …

Battleship Compexity(S) = Min Max Number of misses

Over all shooting algorithms

Over all positions of the shape

How to compute the B.complexity of a given shape ?

Polynomial Time ?

NP-hard ?

Approximations ?

Open questions….

How to compute the B.complexity of a given shape ?

Polynomial Time ?

NP-hard ?

Approximations ?

Give some algorithms for some classes of shapes

Open questions….

How to compute the B.complexity of a given shape ?

Polynomial Time ?

NP-hard ?

Approximations ?

Plan

I The Game / Shooting Algorithms

II Examples

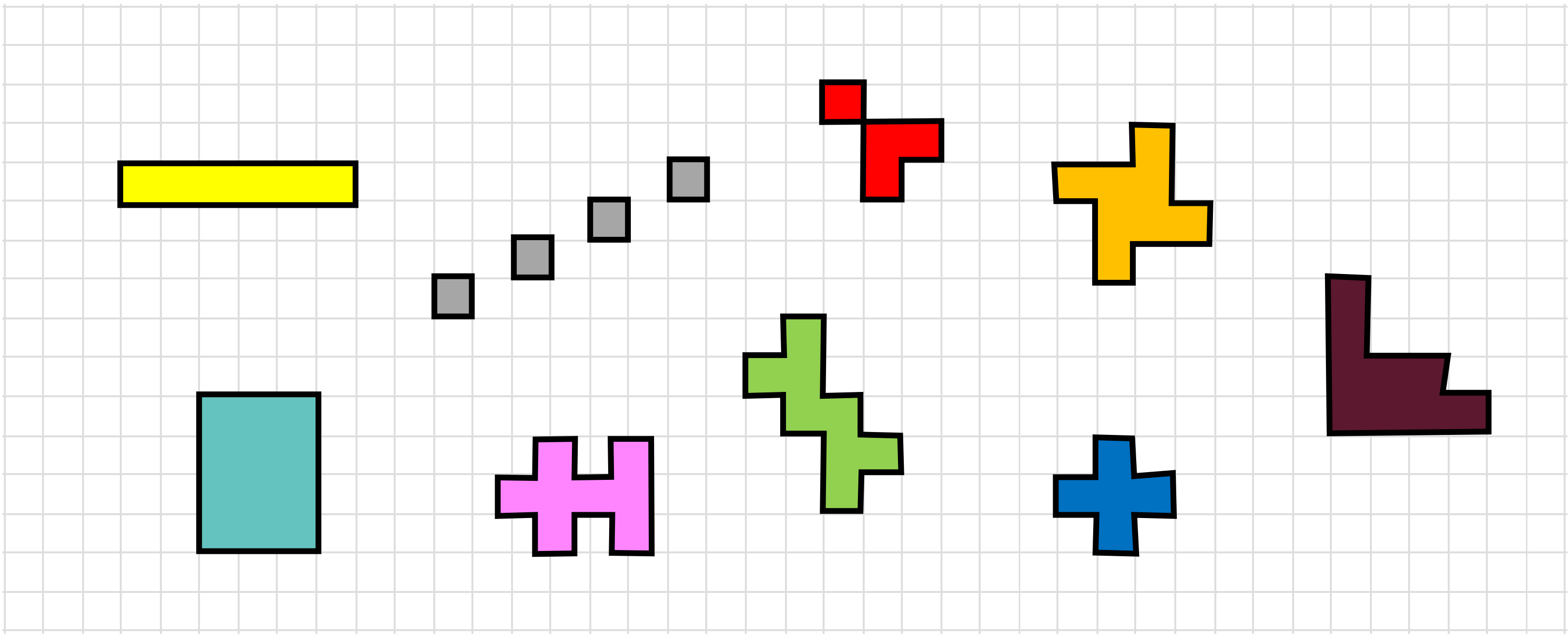
III Results

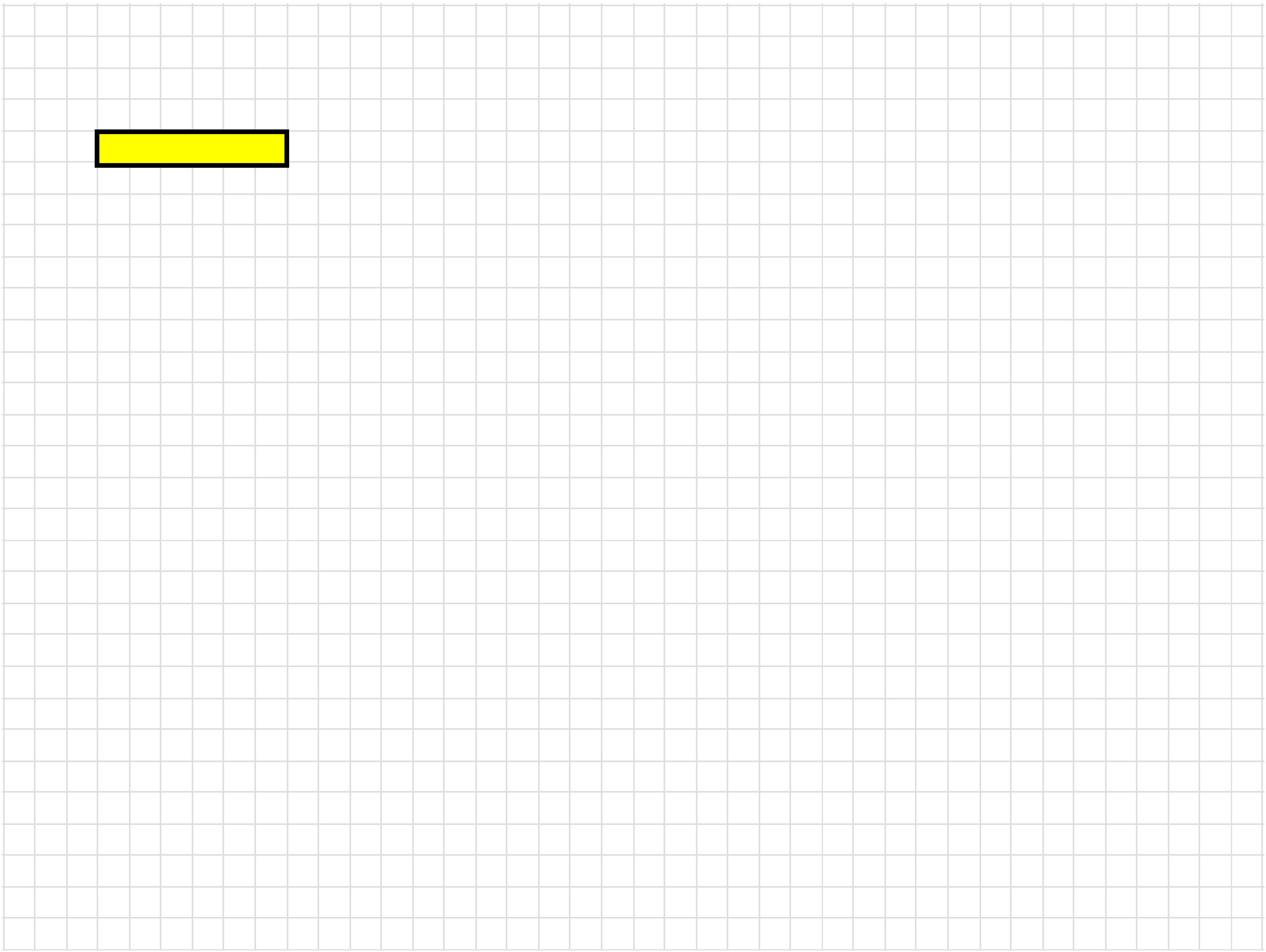
Plan

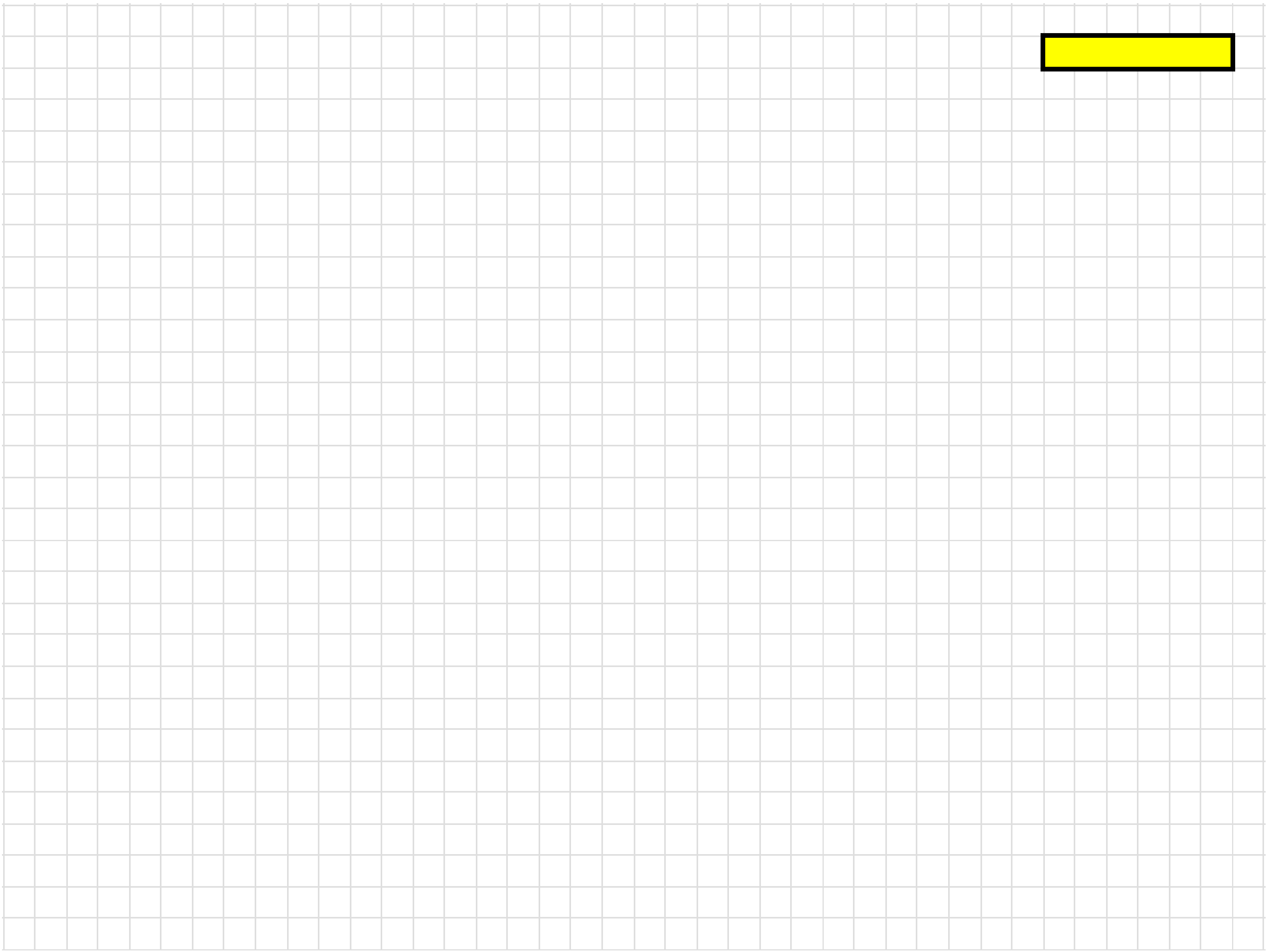
I The Game / Shooting Algorithms

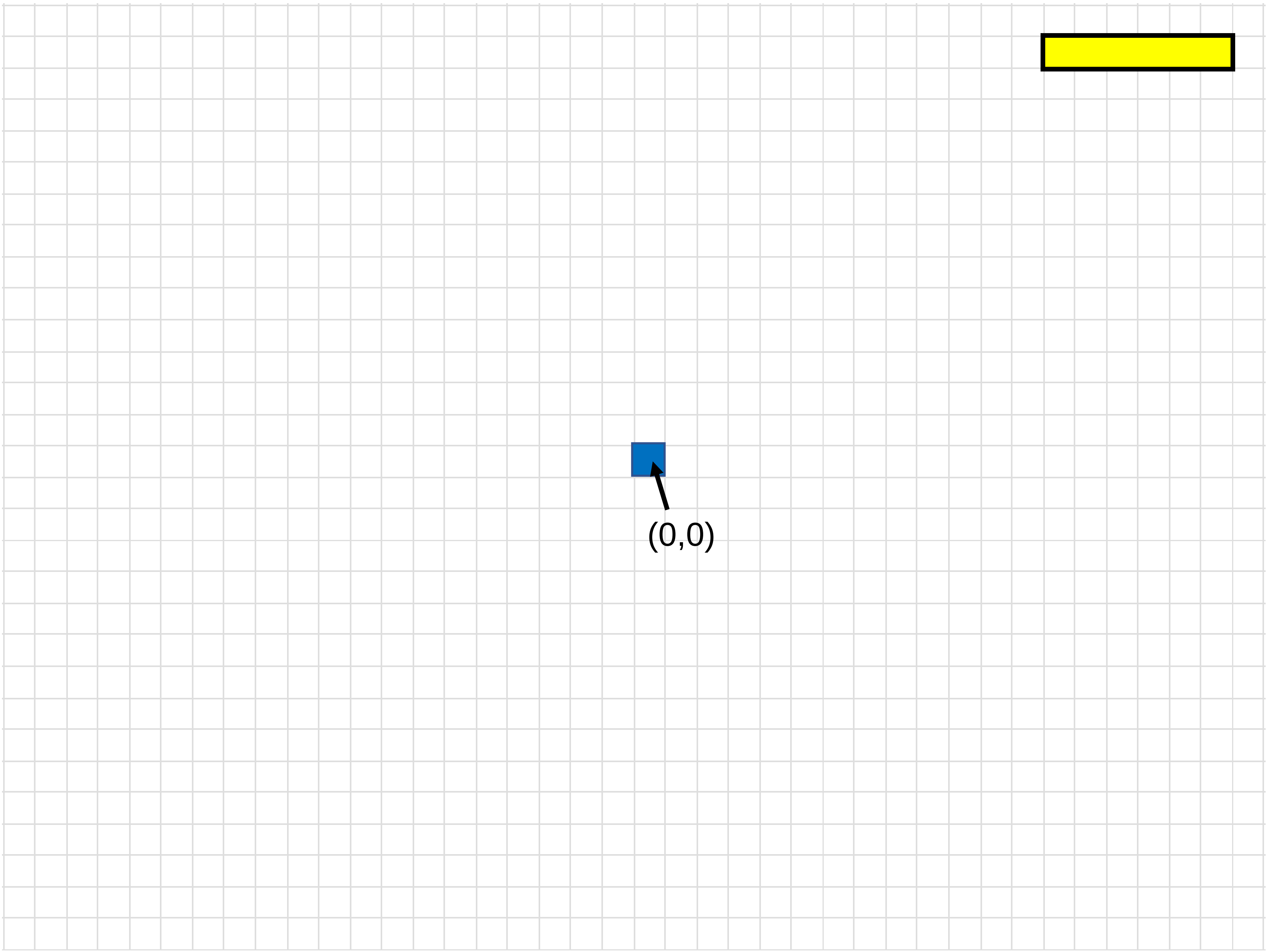
II Examples

III Results

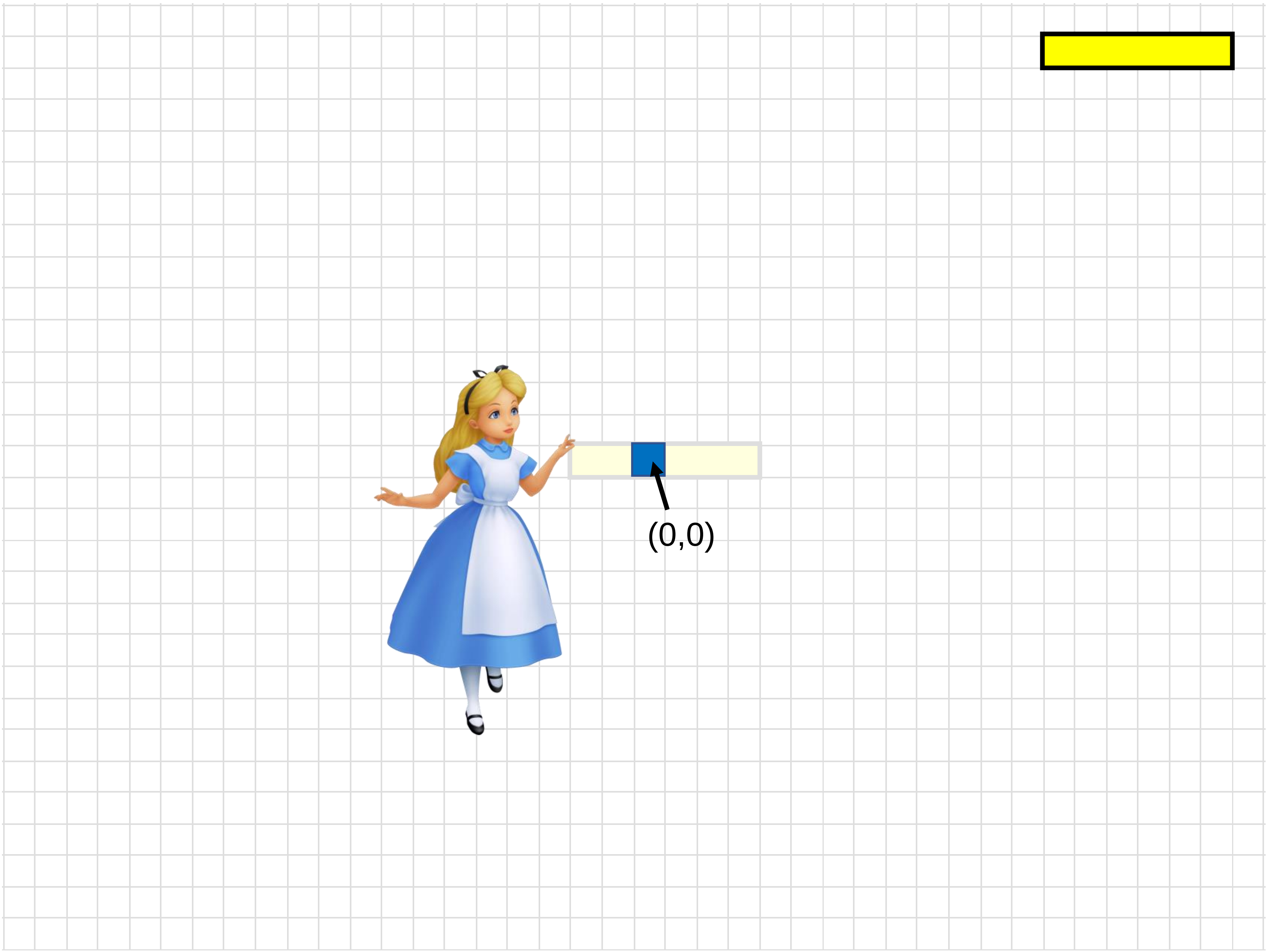


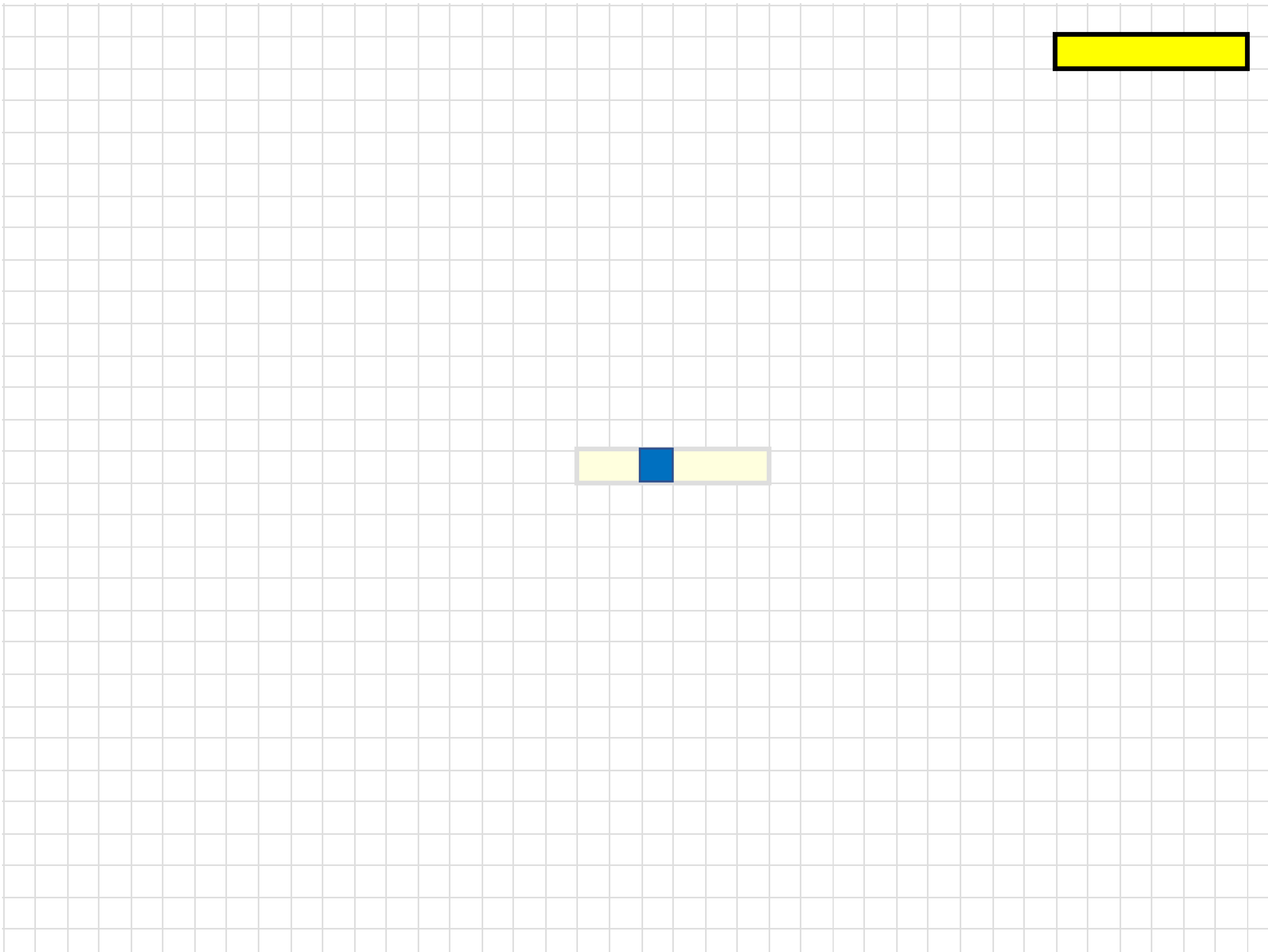




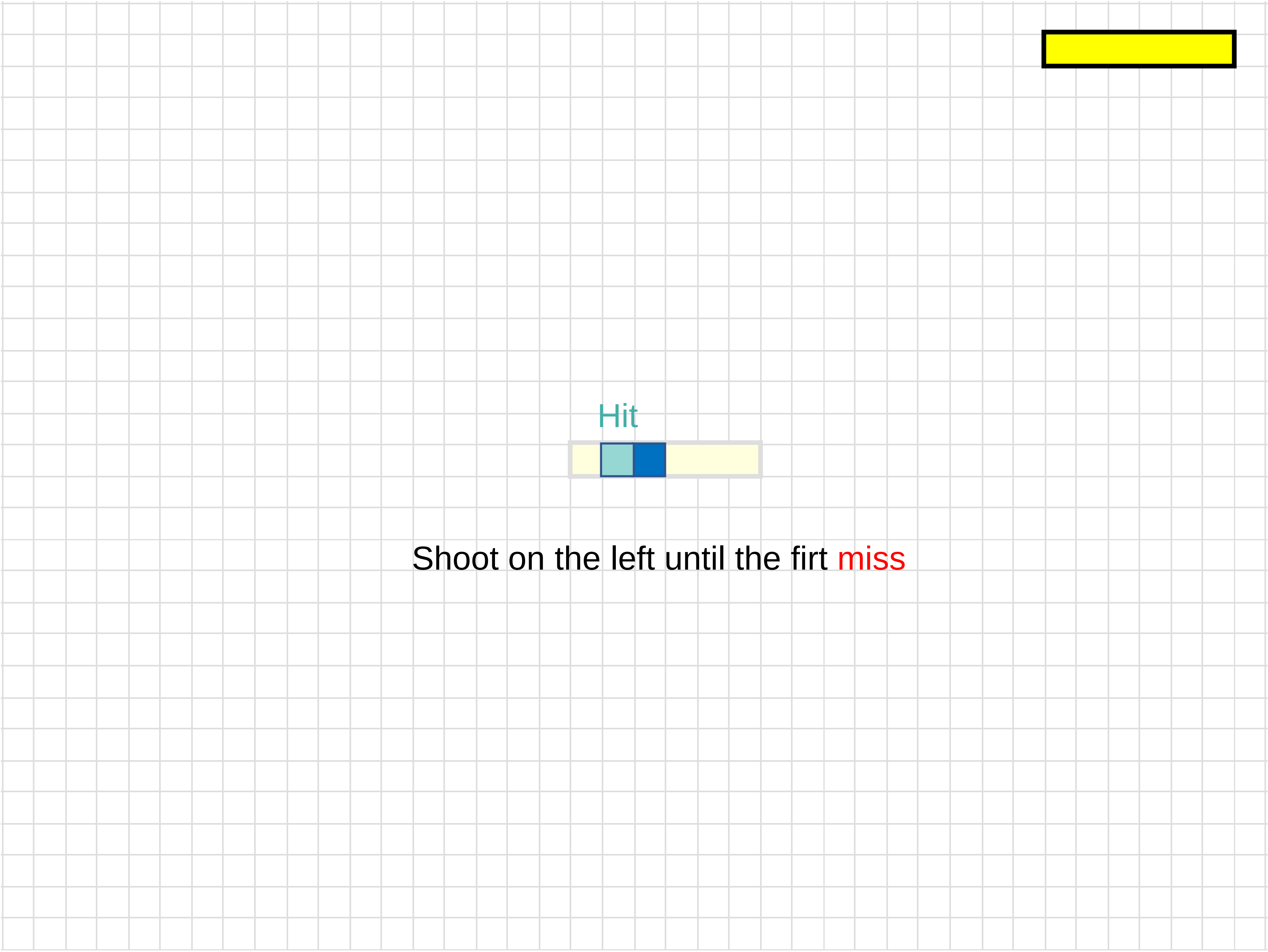

(0,0)

Unknown position chosen by Alice

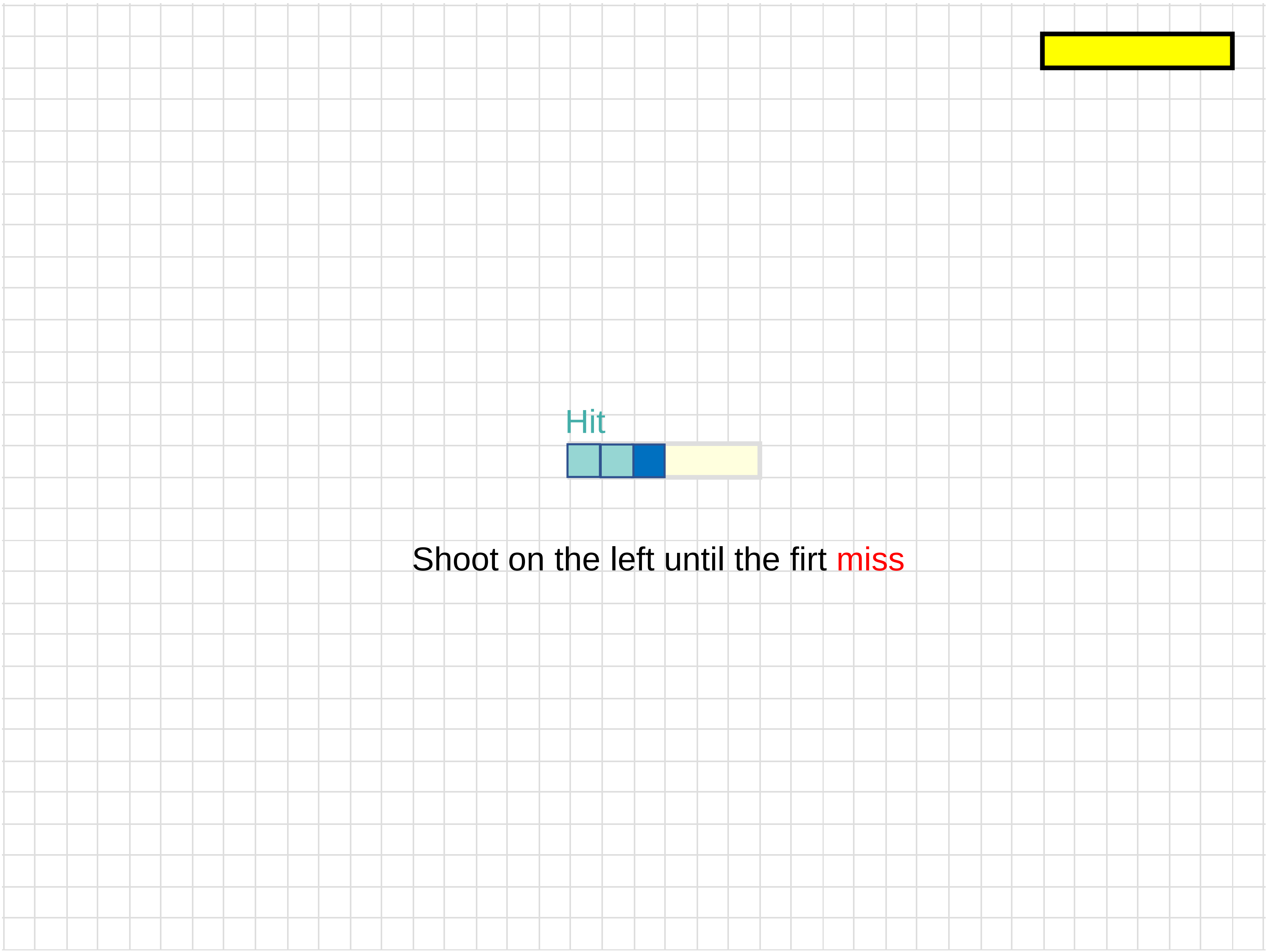




Shot (-1,0)

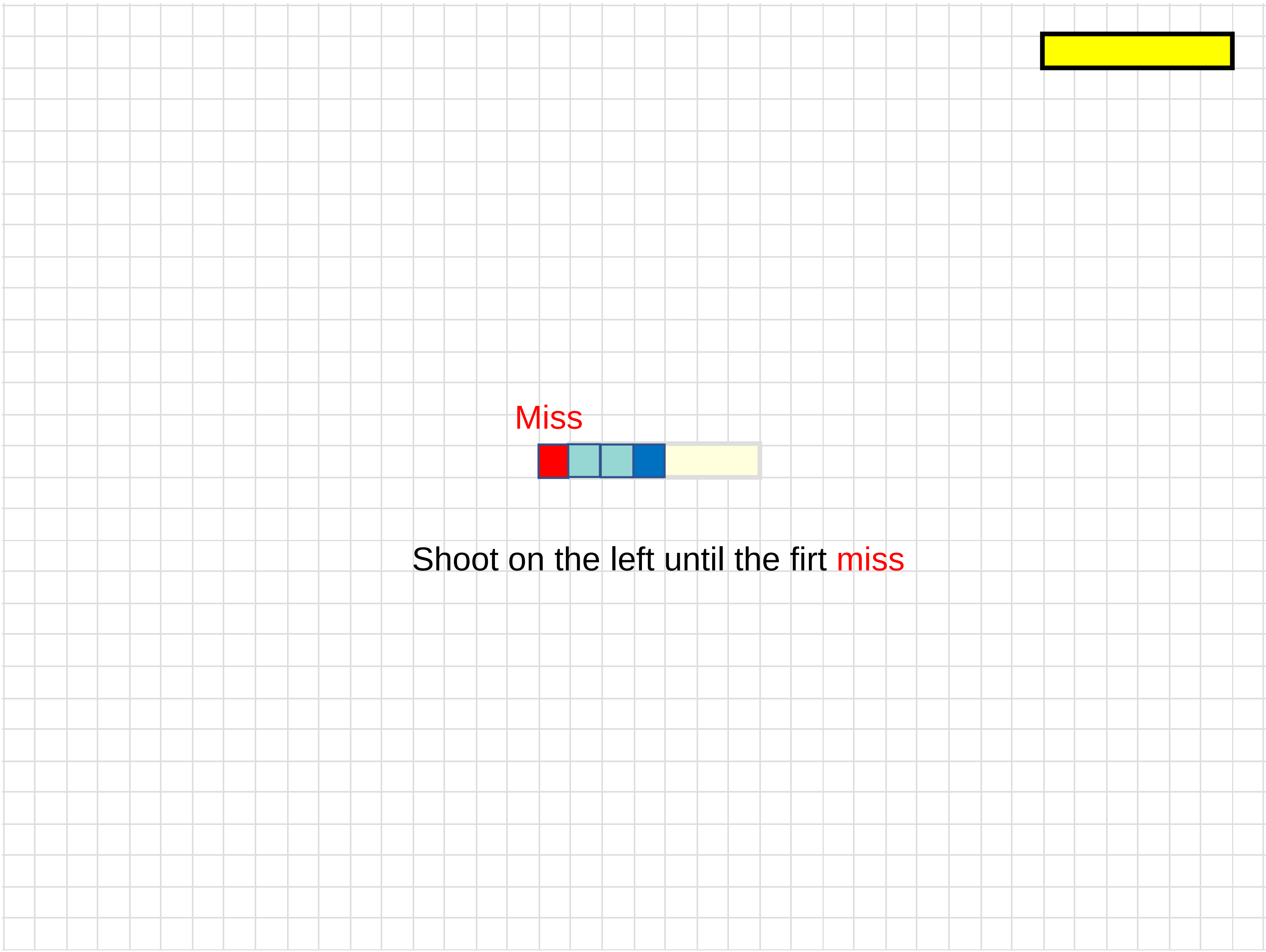


Shoot on the left until the firt miss



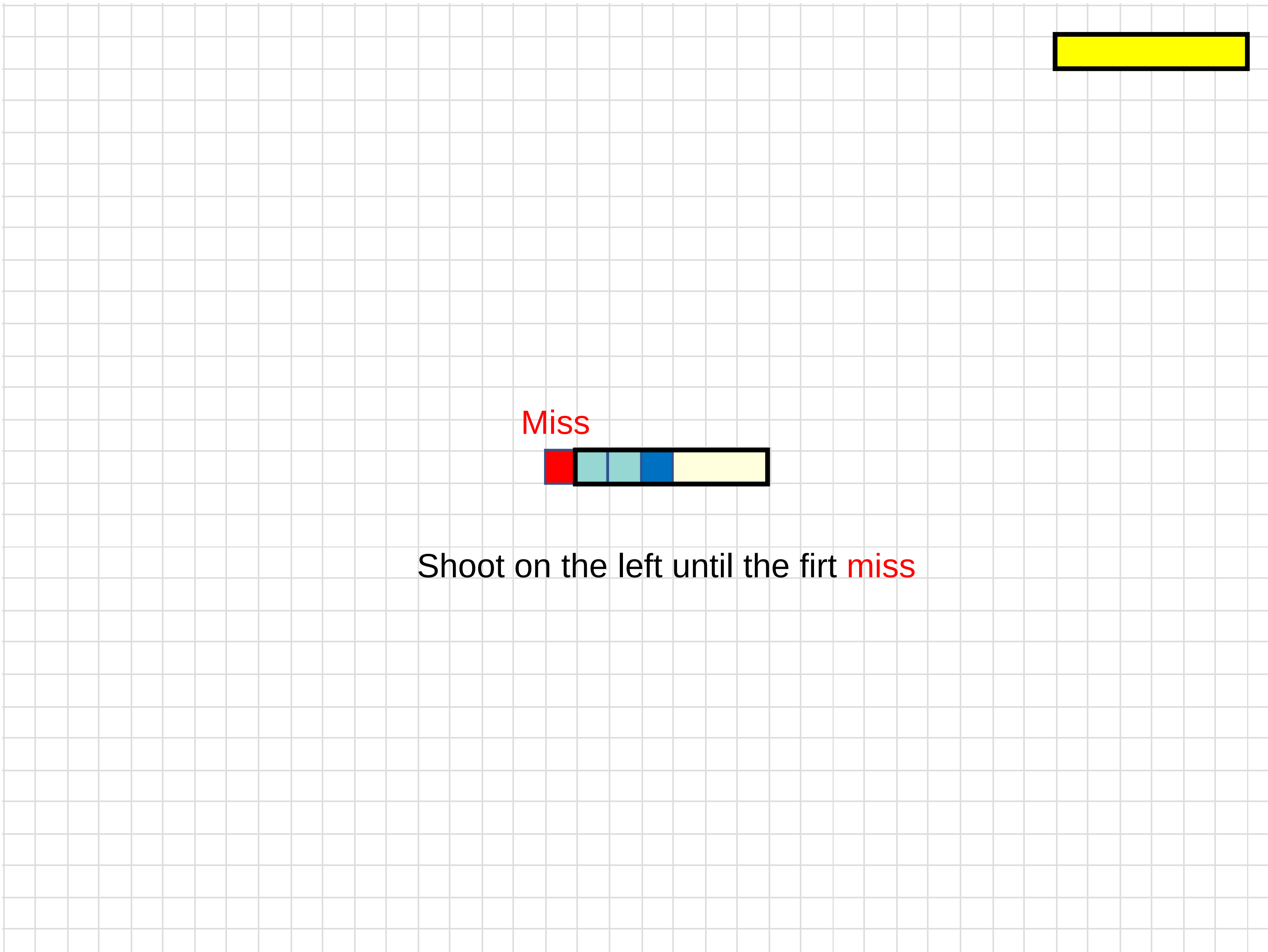
Shoot on the left until the firt miss

Shot (-3,0)

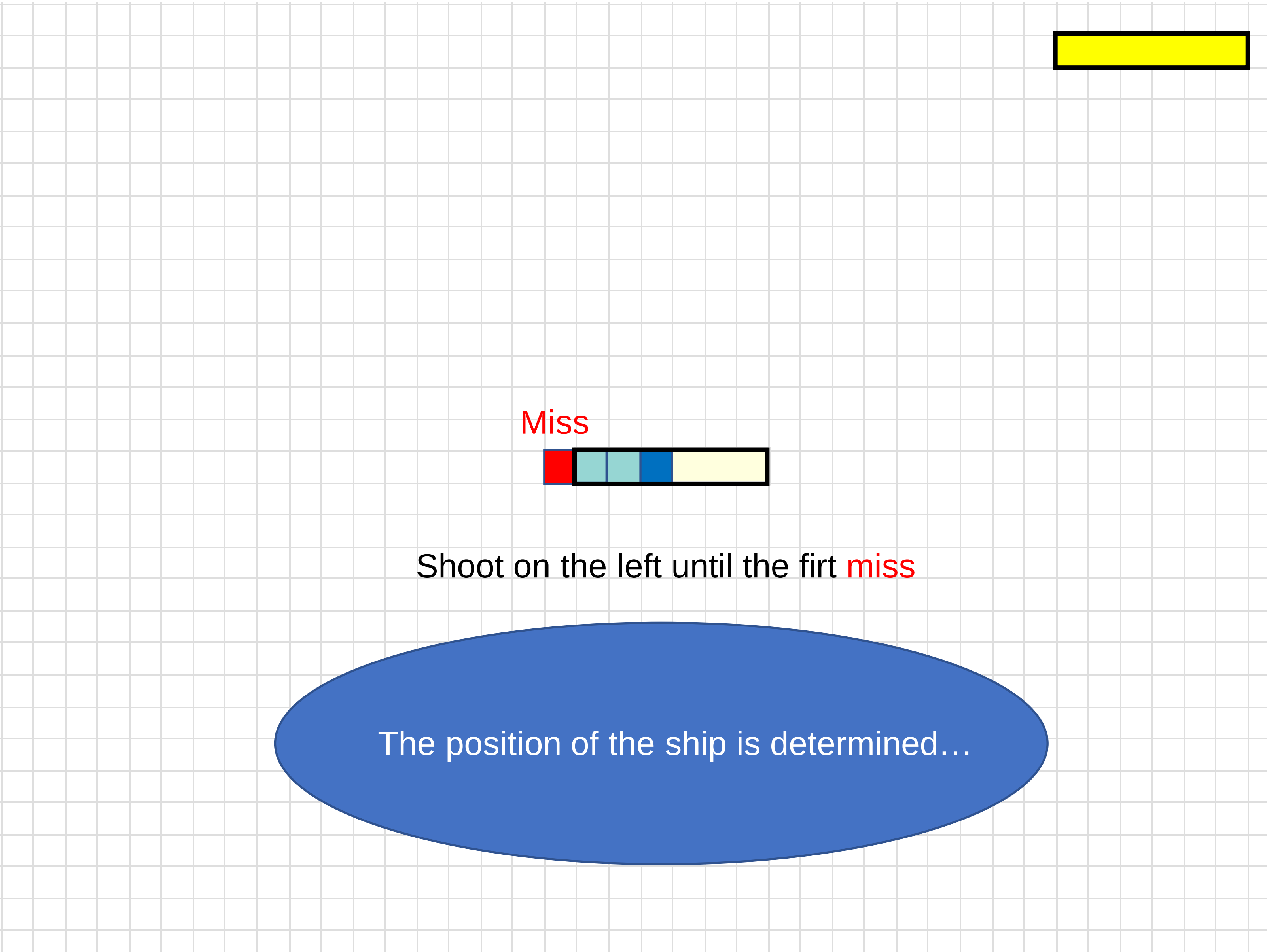


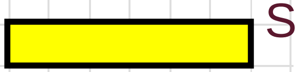
Shoot on the left until the firt miss

Shot (-3,0)



Shoot on the left until the firt miss


Miss
Shoot on the left until the firt miss
The position of the ship is determined…



We need at most 1 miss!

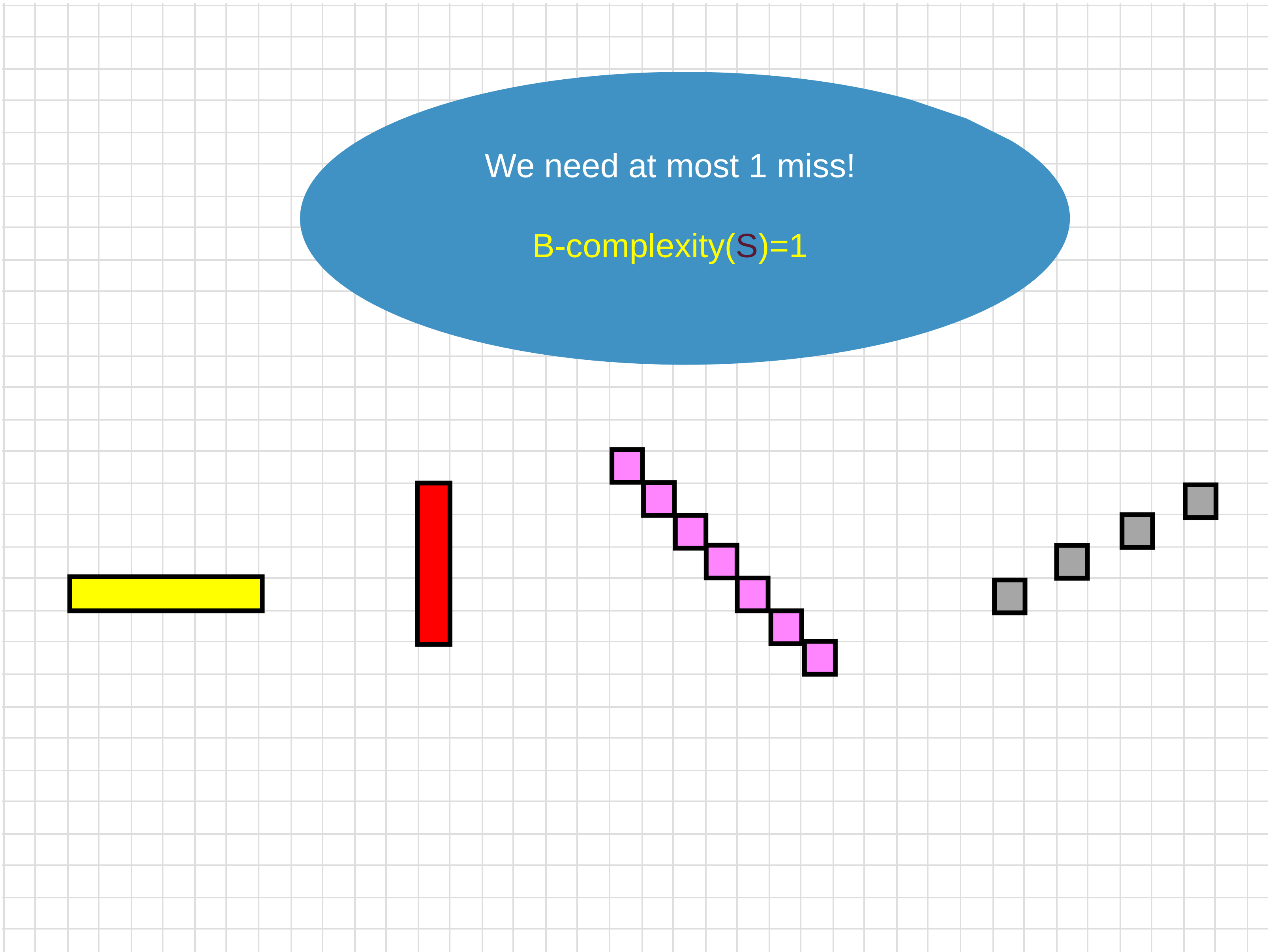
B-complexity(S)=1

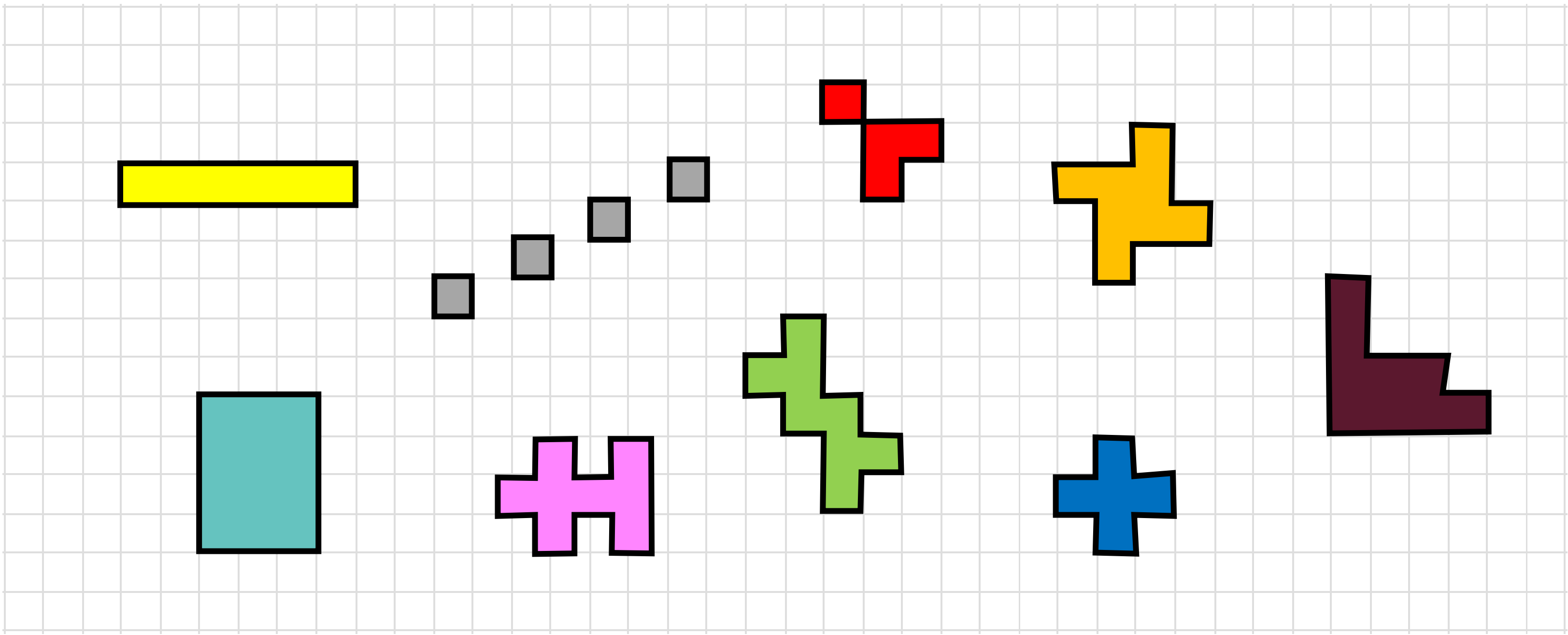
Miss

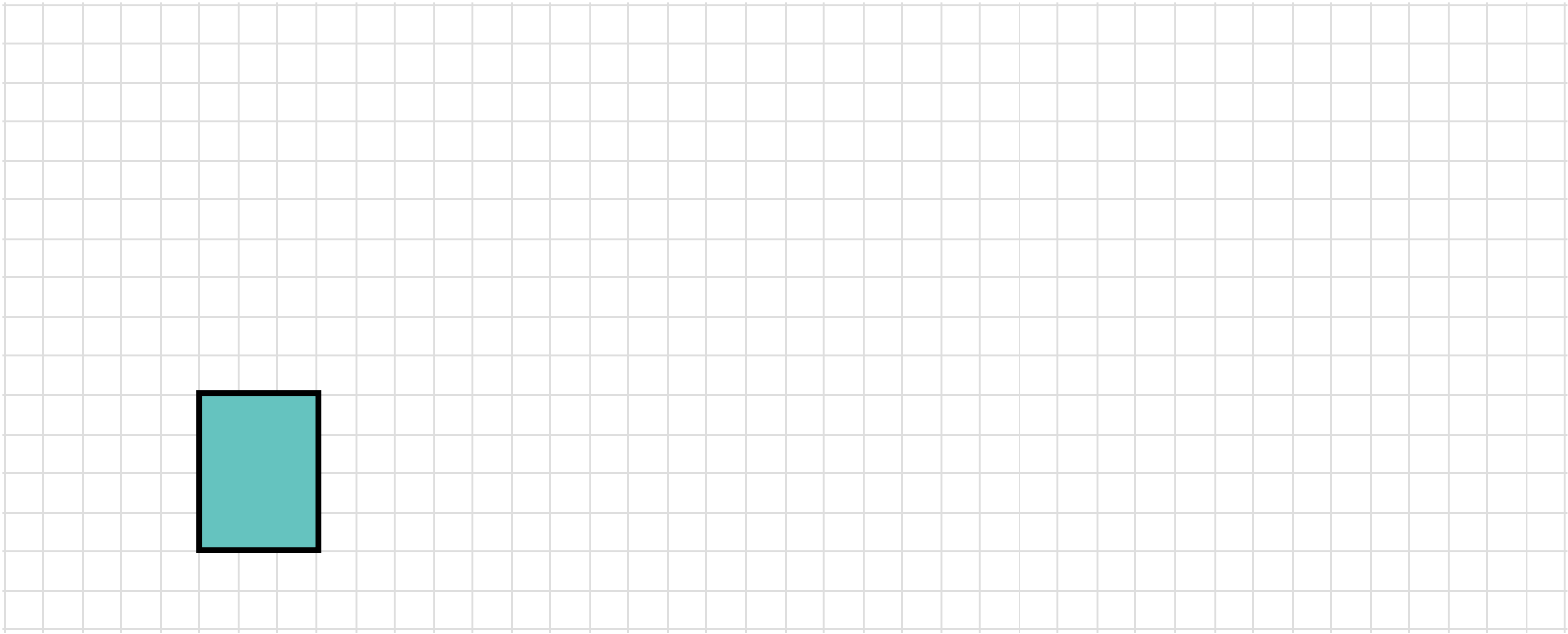
Shoot on the left until the firt miss

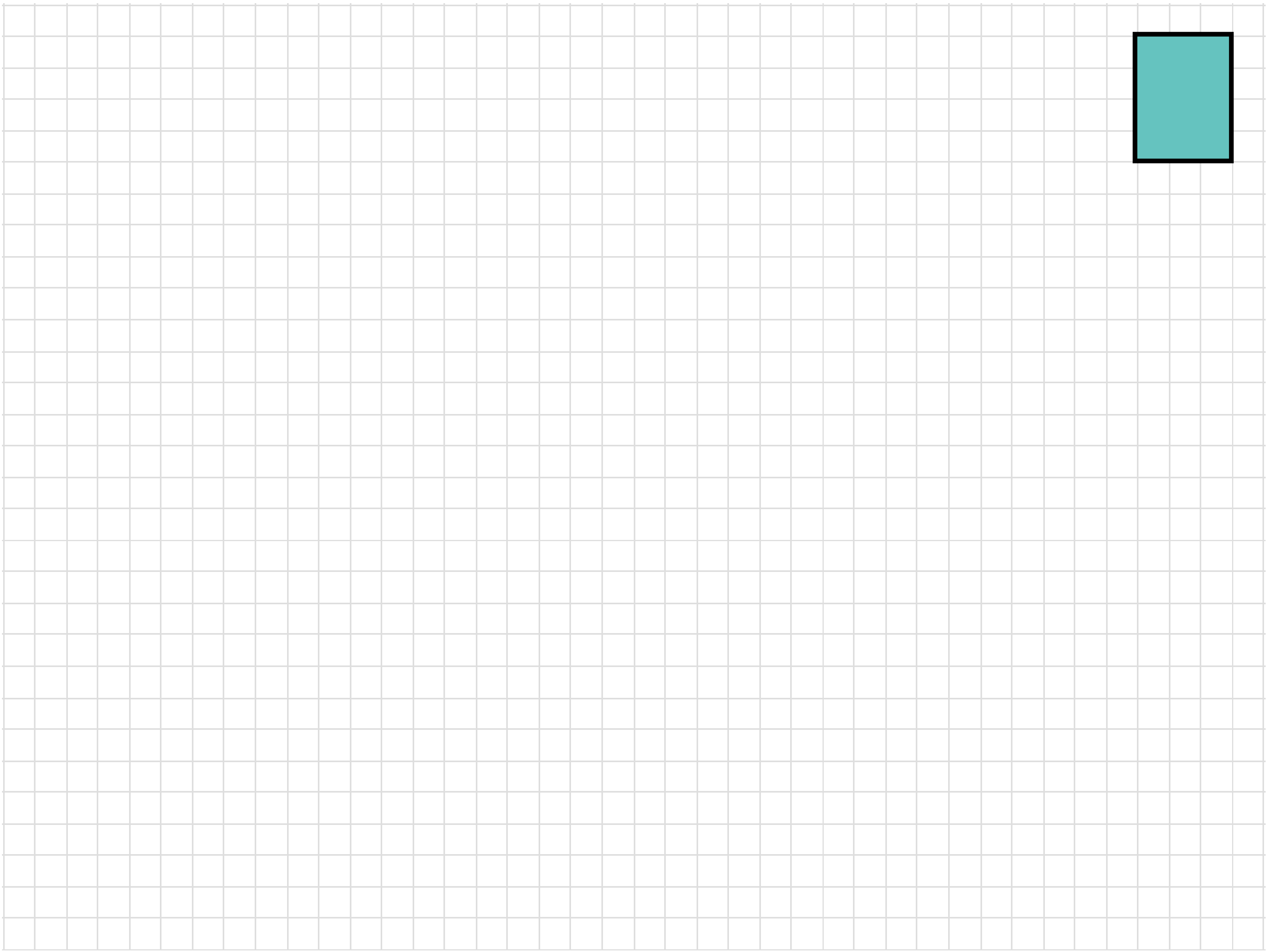
II Examples

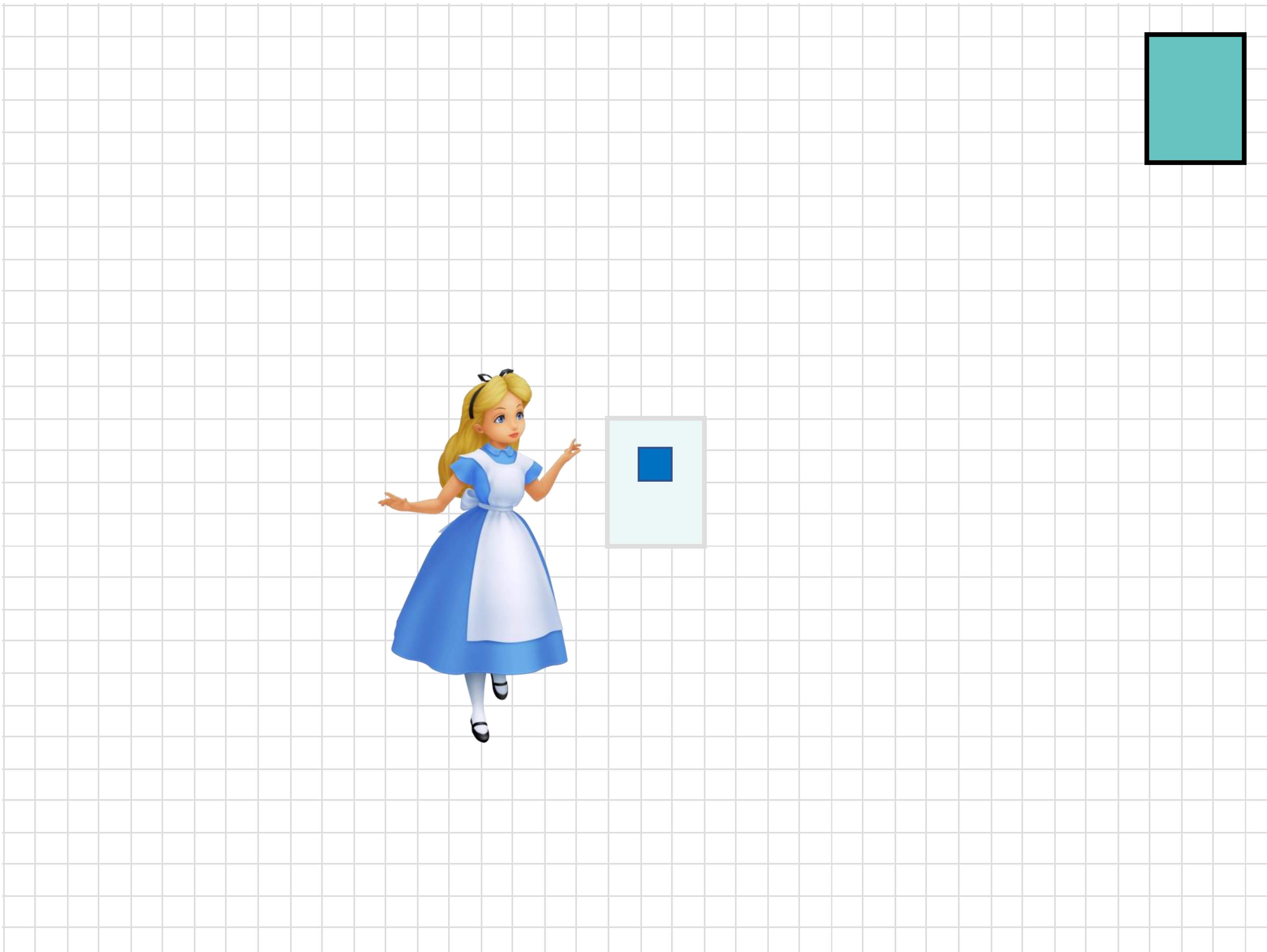
Only 1 miss

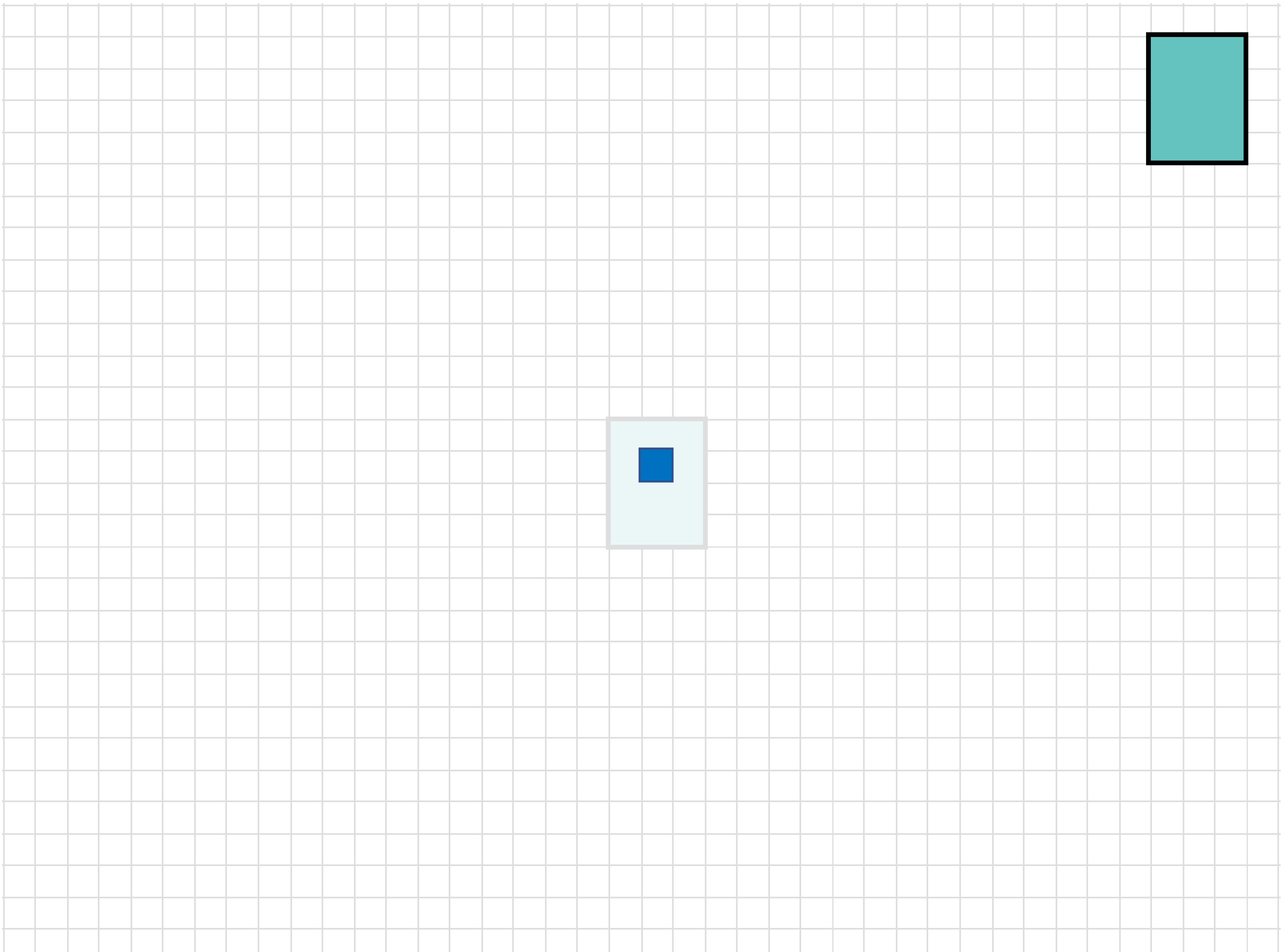


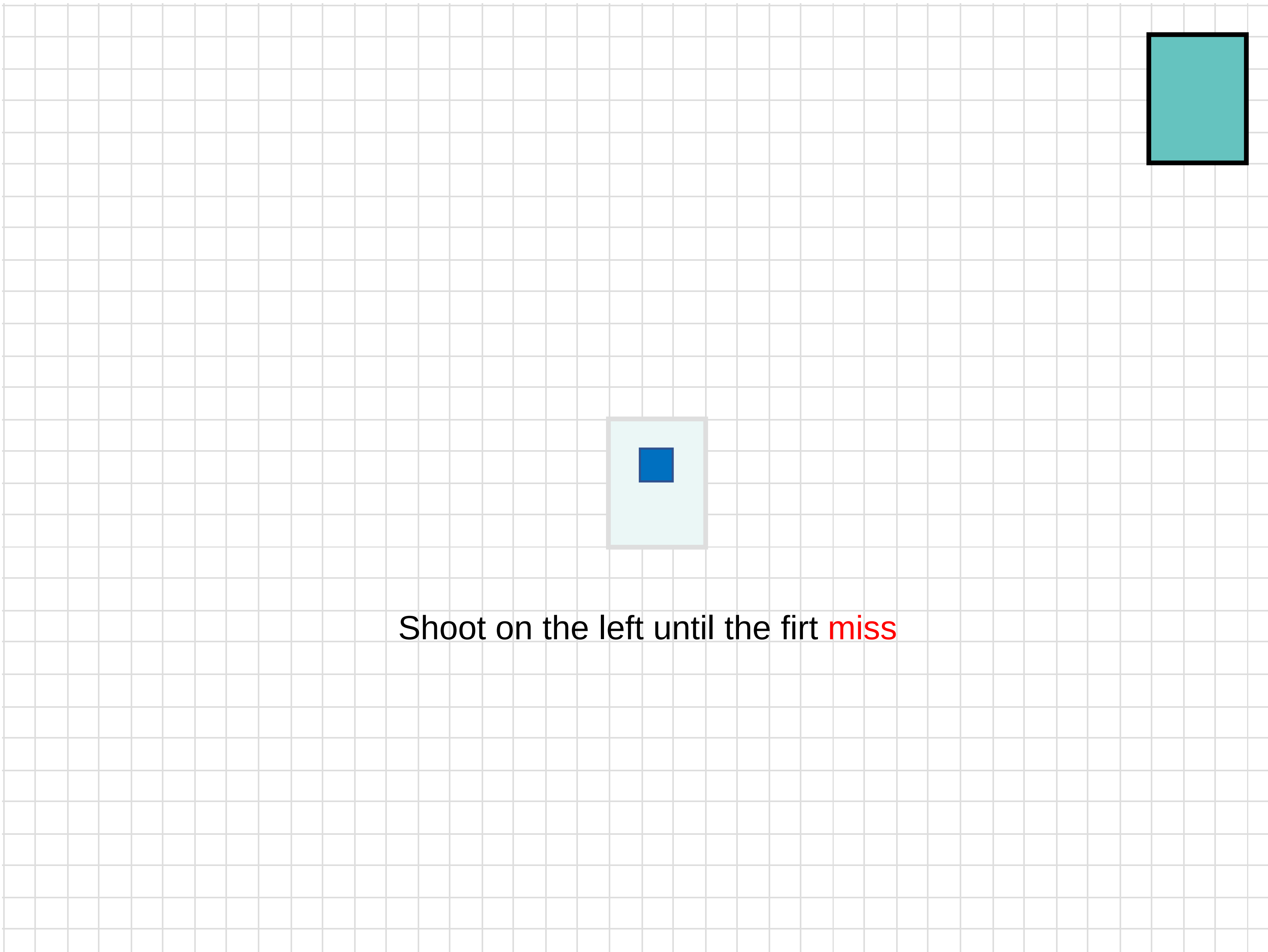


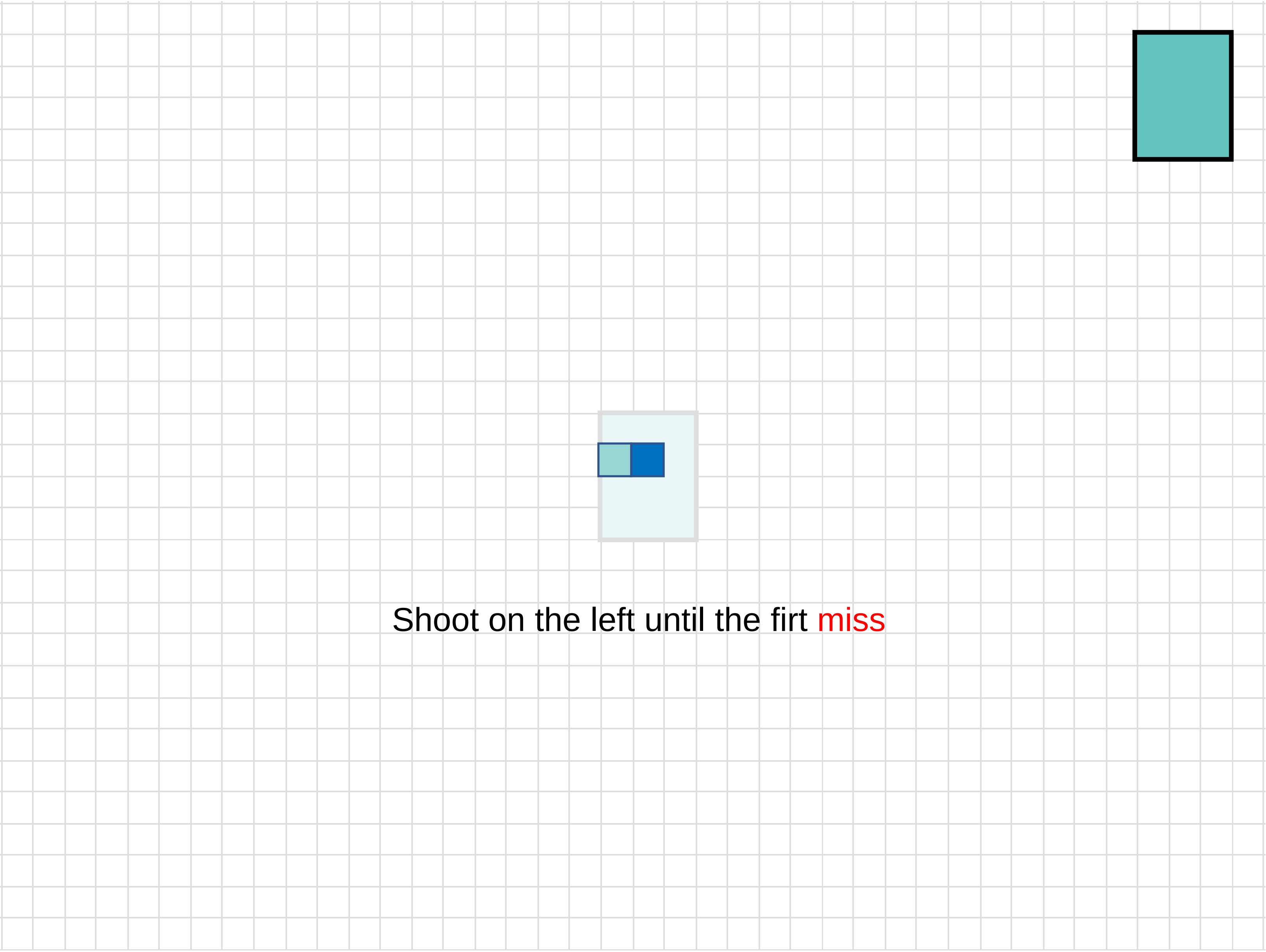




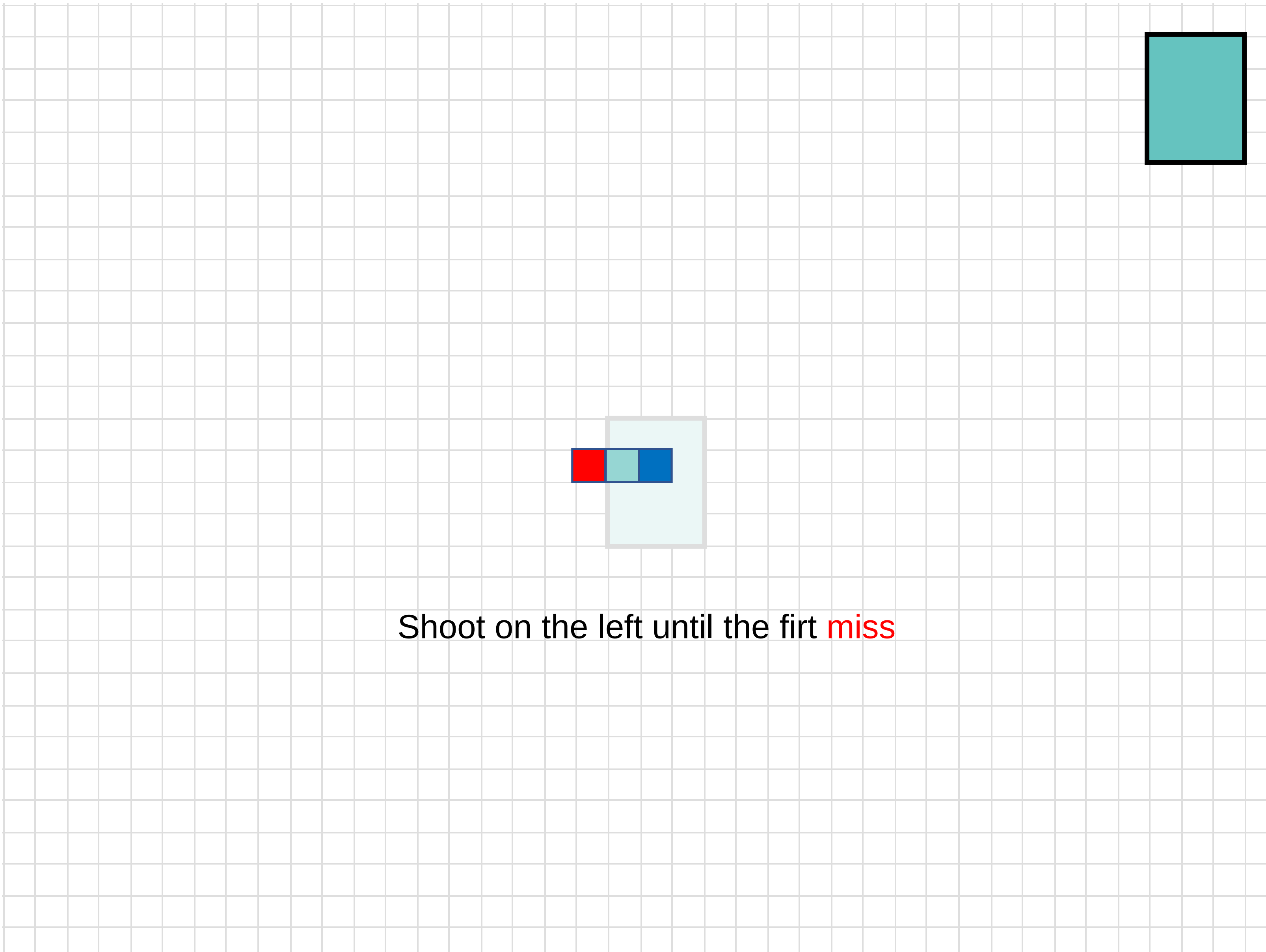




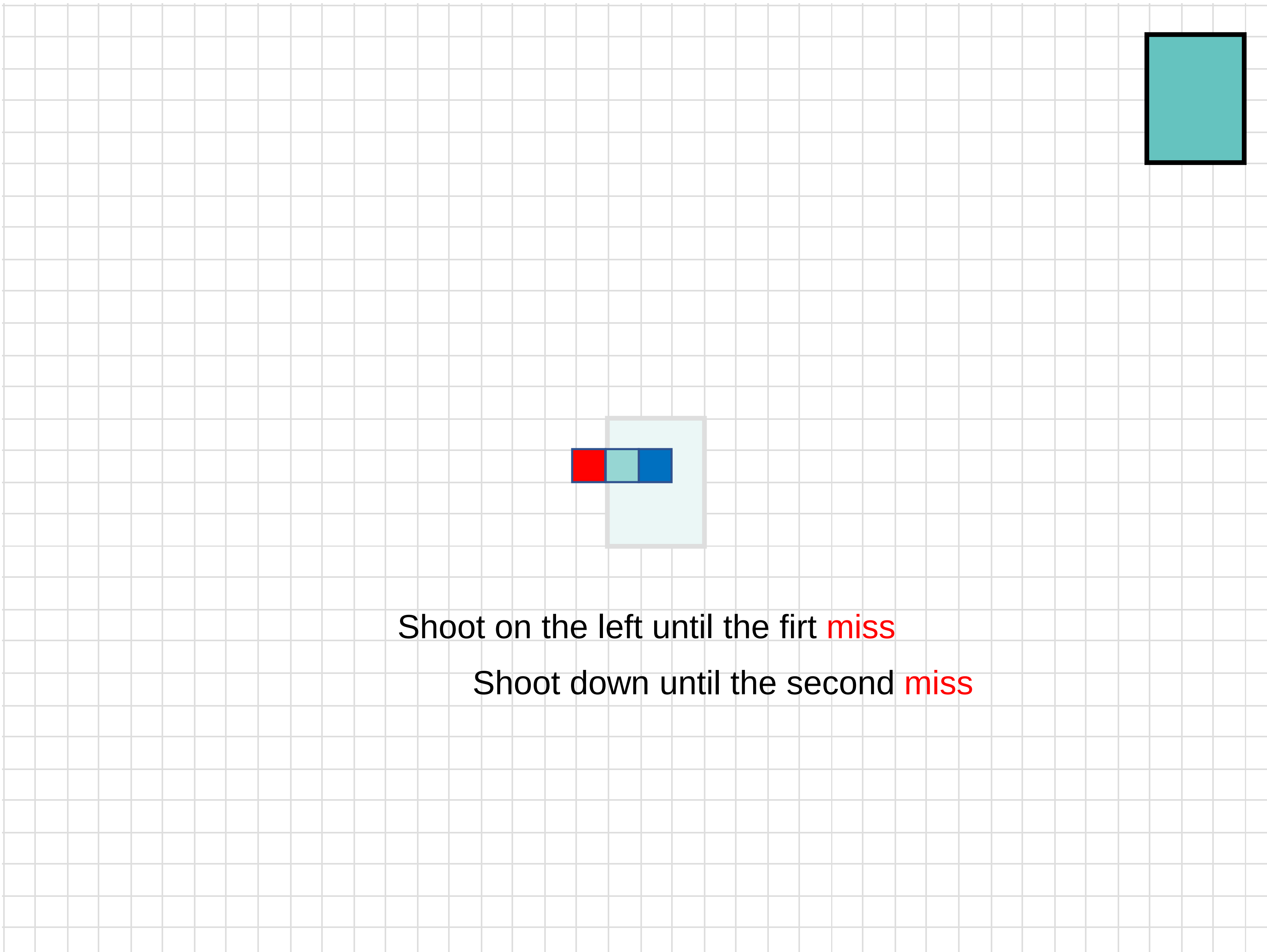

Shoot on the left until the firt miss

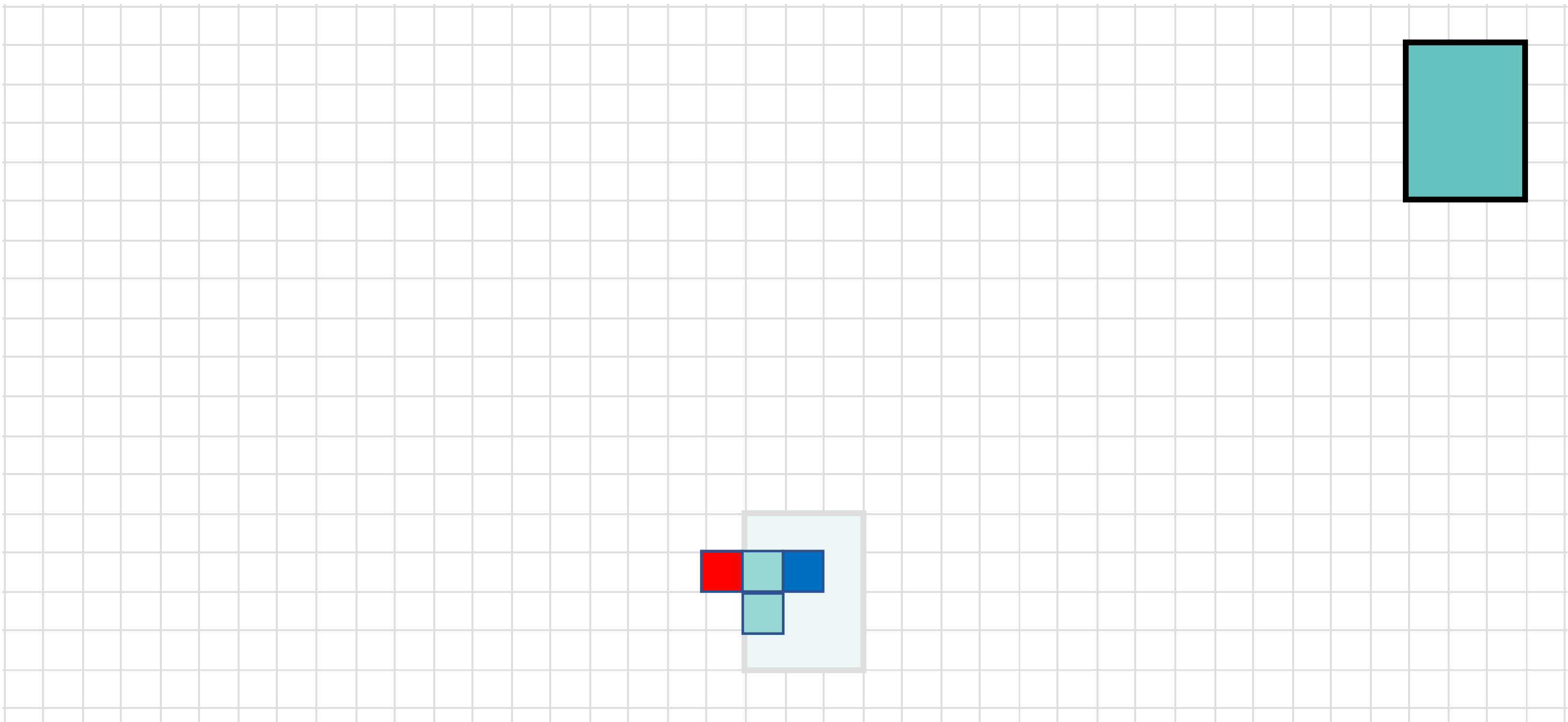


Shoot on the left until the firt miss


Shoot on the left until the firt miss

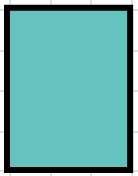
Rectangle

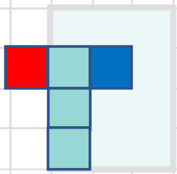

Shoot on the left until the firt miss
Shoot down until the second miss



Shoot on the left until the firt miss

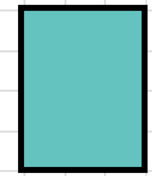
Shoot down until the second miss

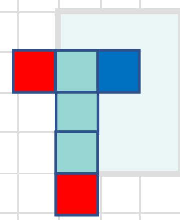




Shoot on the left until the firt miss

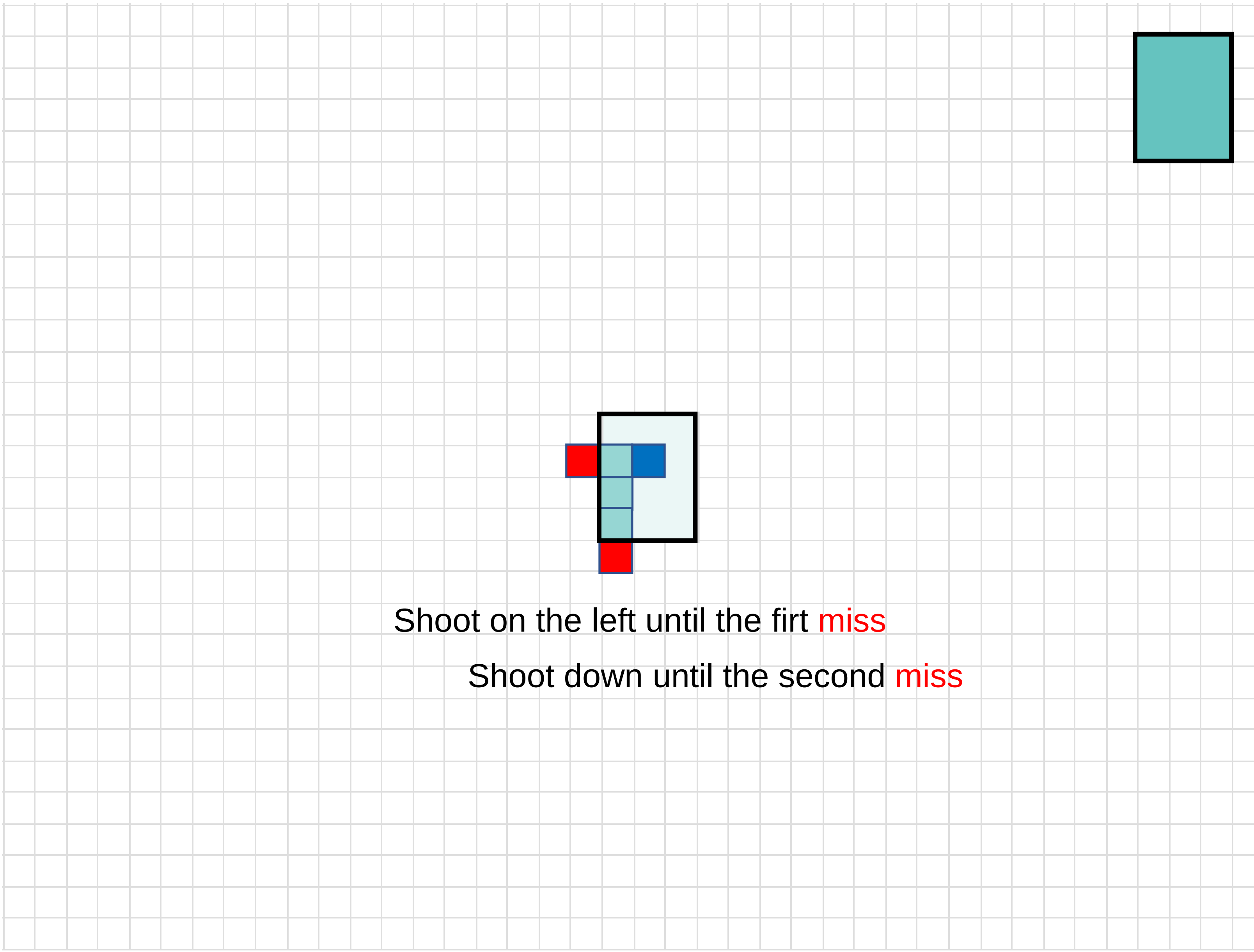
Shoot down until the second miss

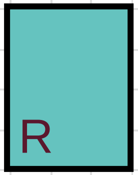




Shoot on the left until the firt miss

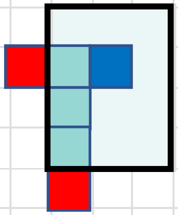
Shoot down until the second miss


Shoot on the left until the firt miss
Shoot down until the second miss



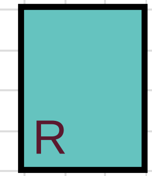
We have an algorithm with at most 2 misses.
There is no algorithm with at most 1 miss.

B-complexity(R)=2



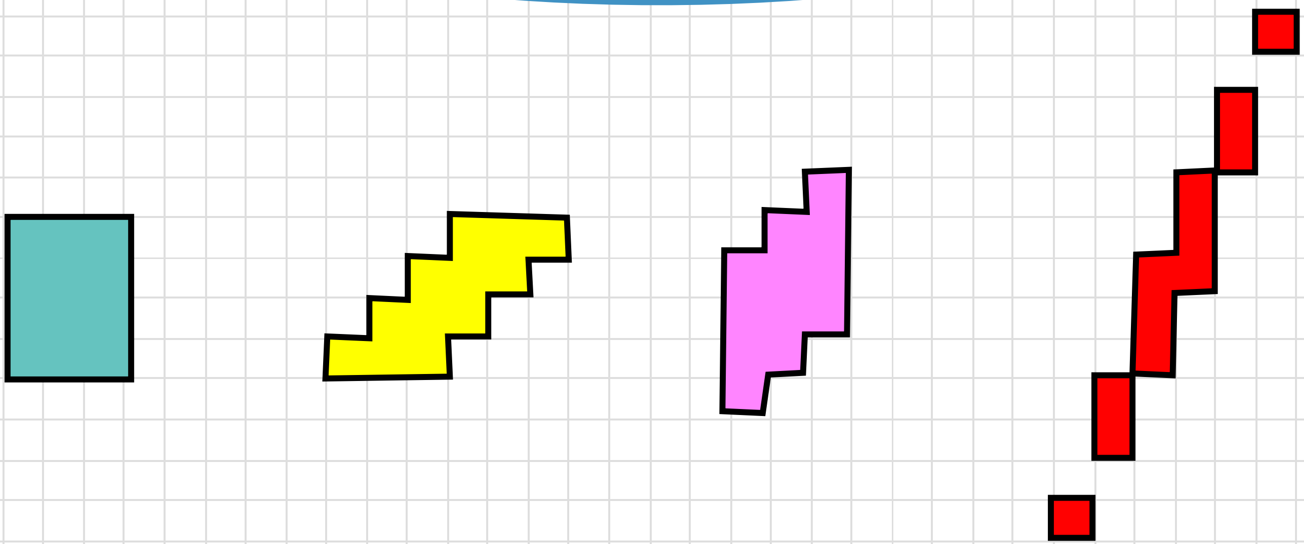
Shoot on the left until the firt miss

Shoot down until the second miss

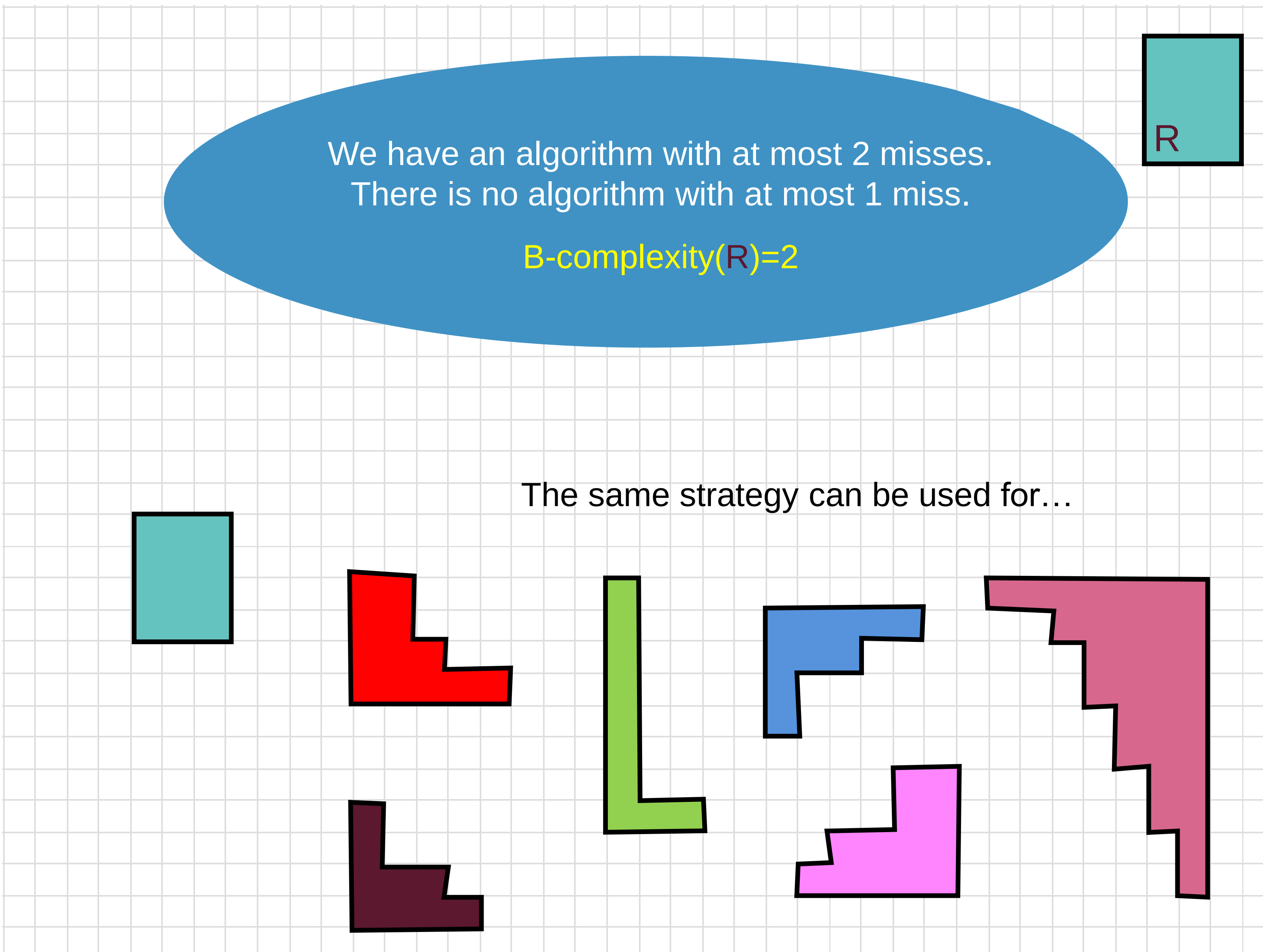


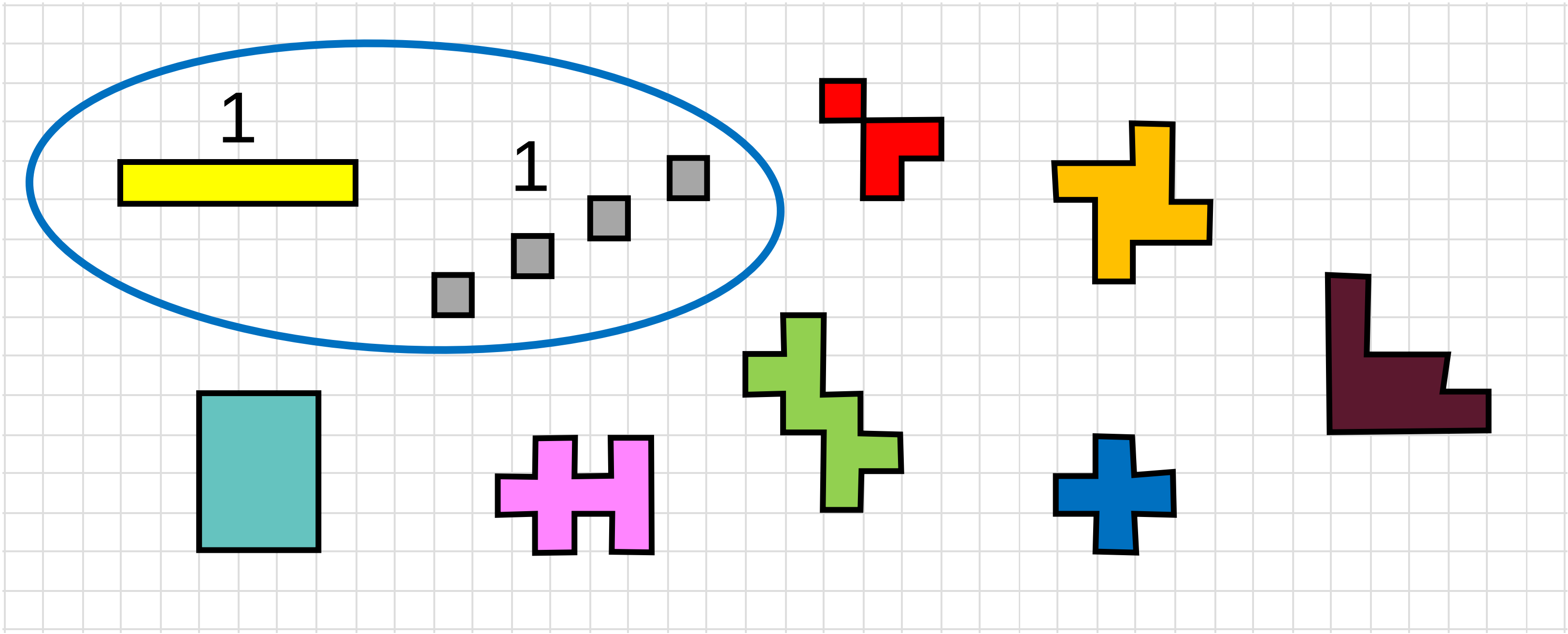
We have an algorithm with at most 2 misses.
There is no algorithm with at most 1 miss.

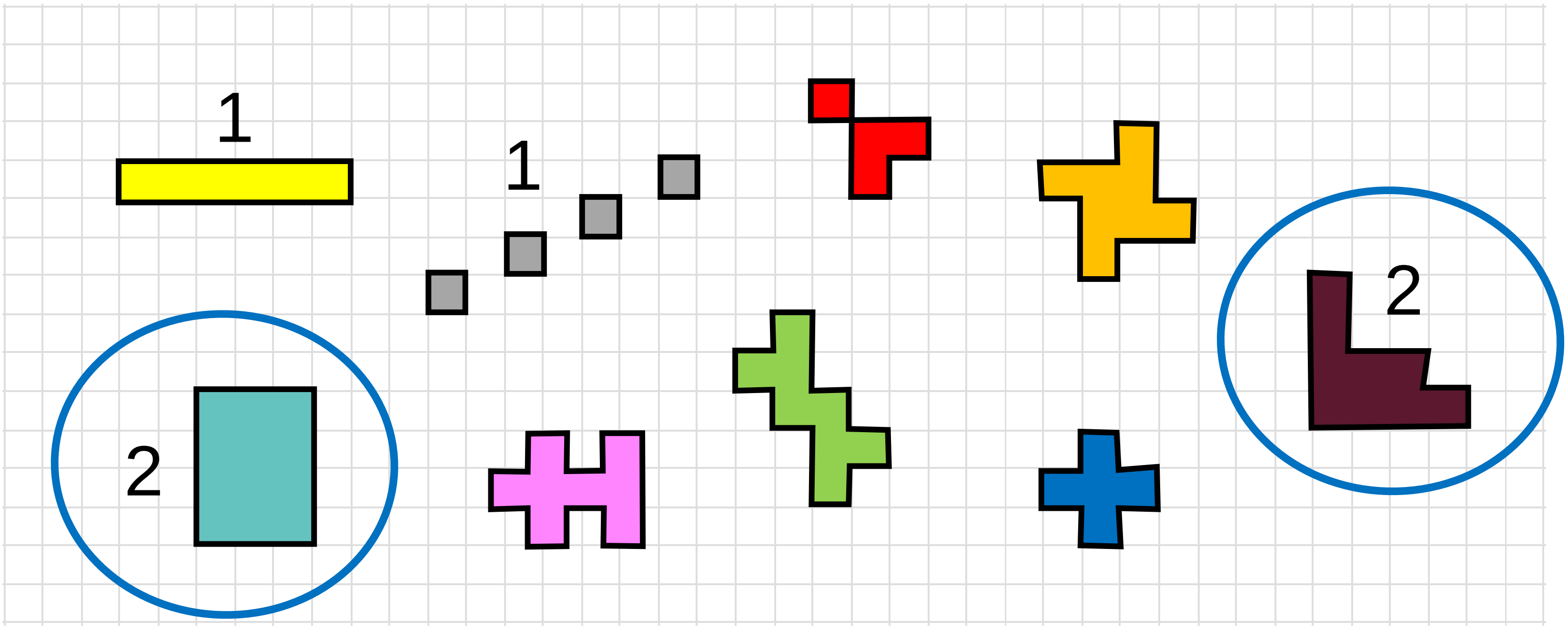
B-complexity(R)=2

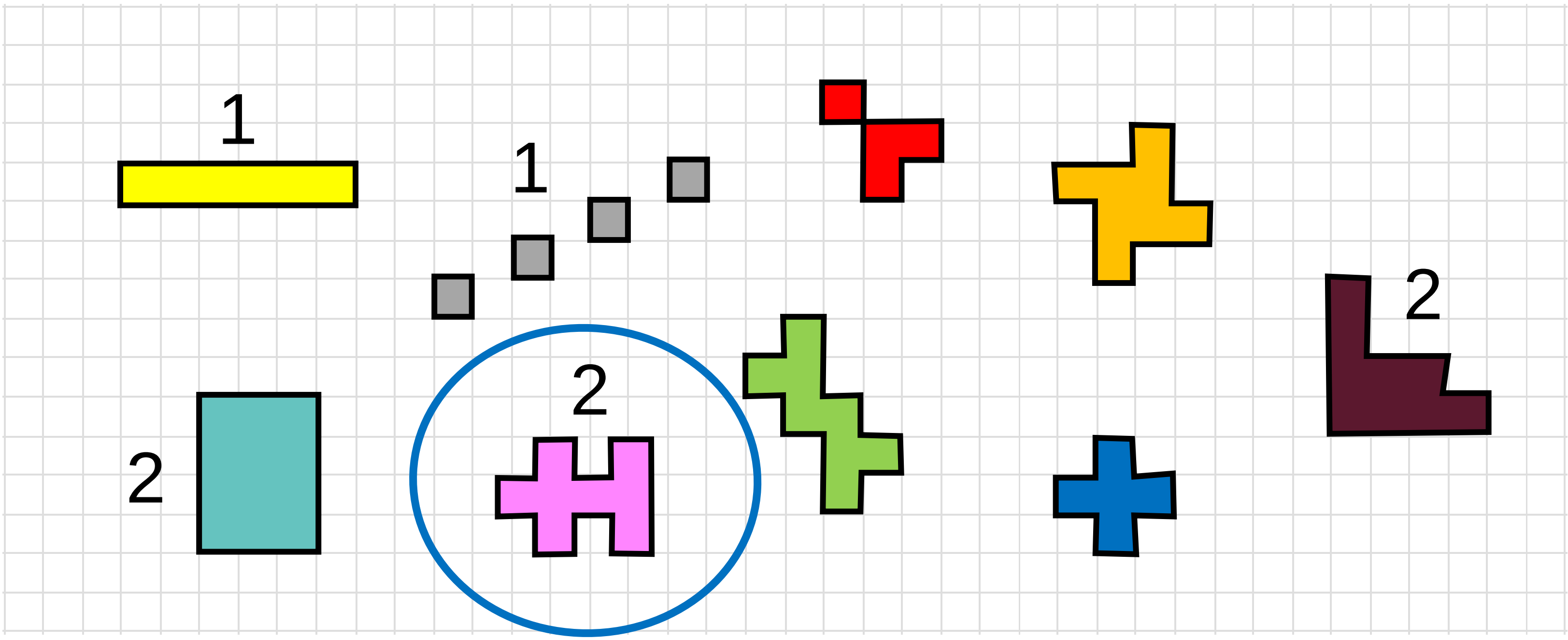


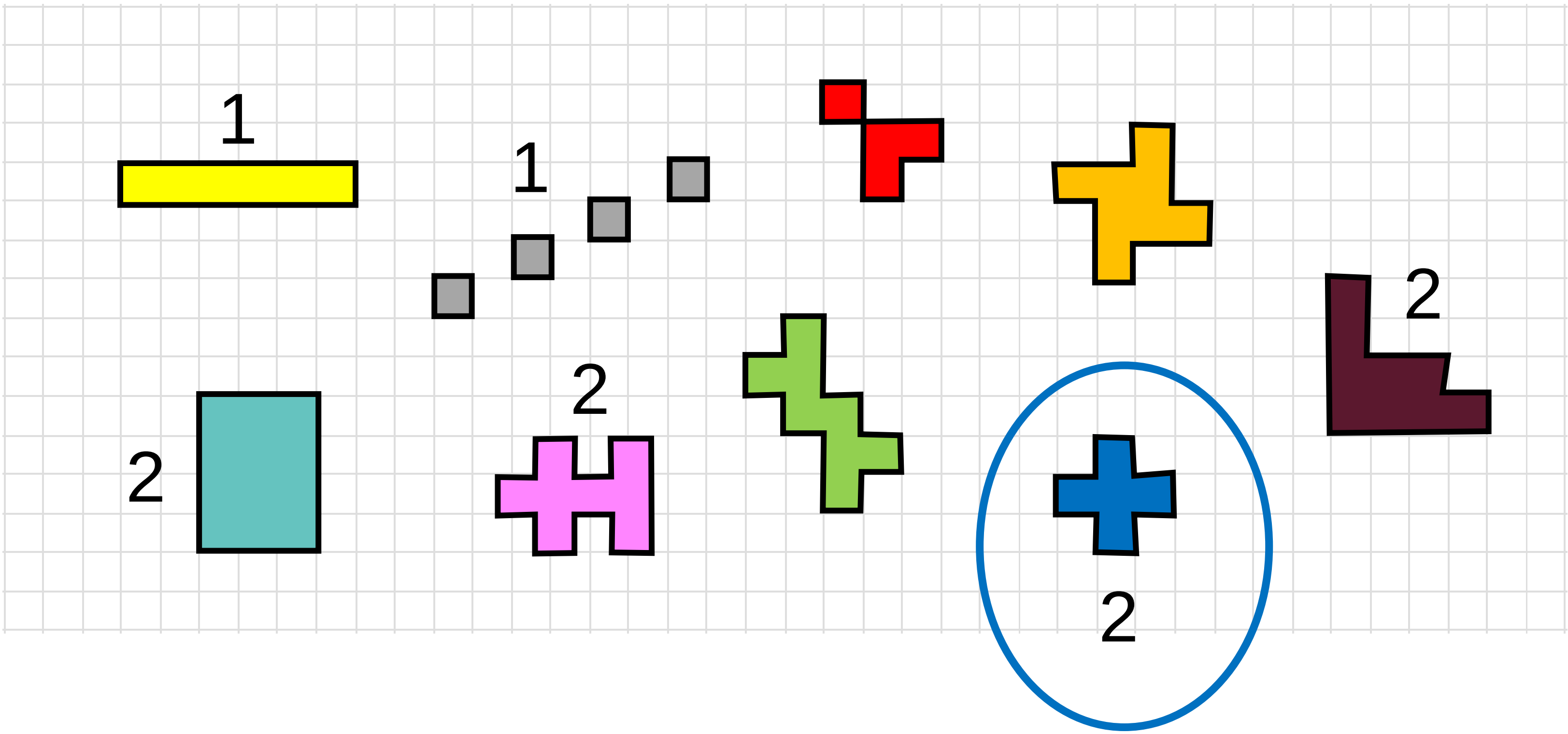
The B-complexity is invariant by shifting
(more generally by morphisms (injective on S-S)).

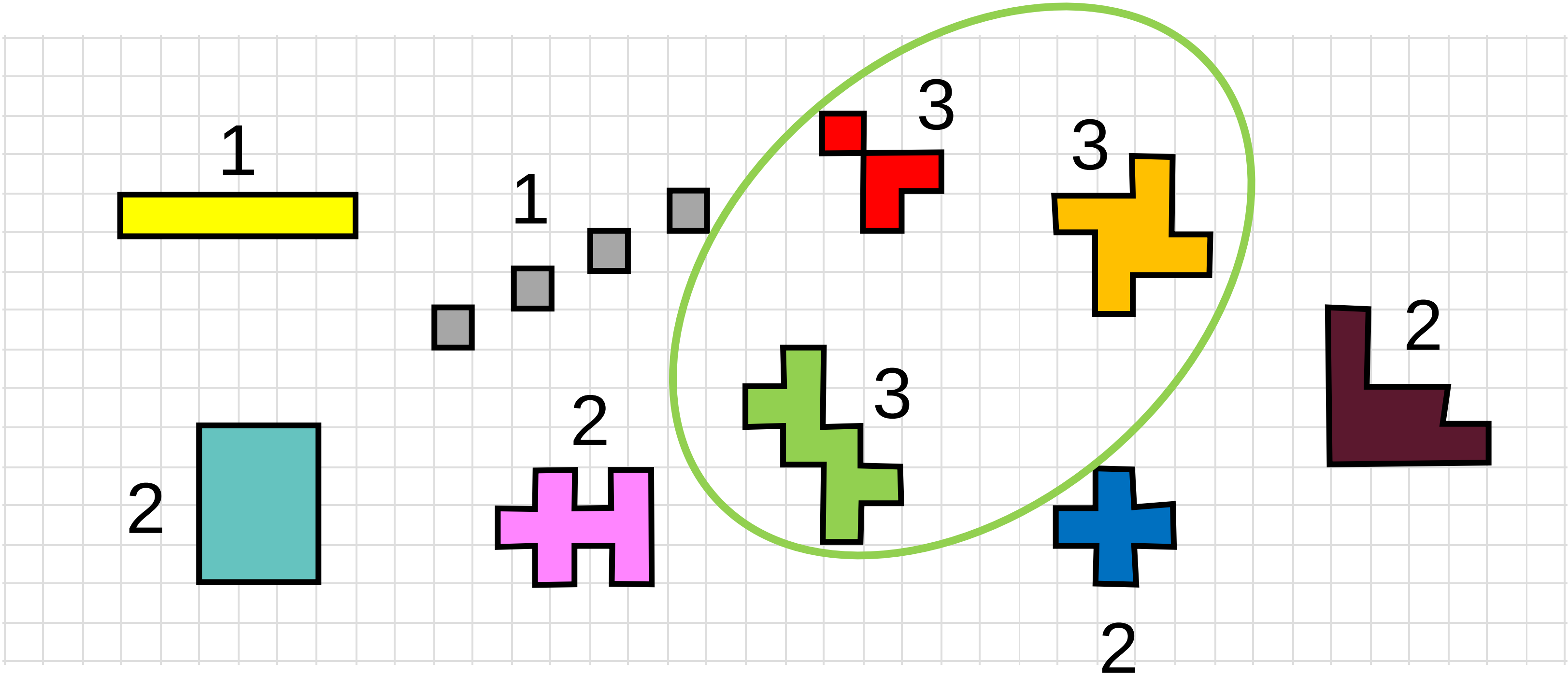

R
We have an algorithm with at most 2 misses.
There is no algorithm with at most 1 miss.
B-complexity(R)=2
The same strategy can be used for…

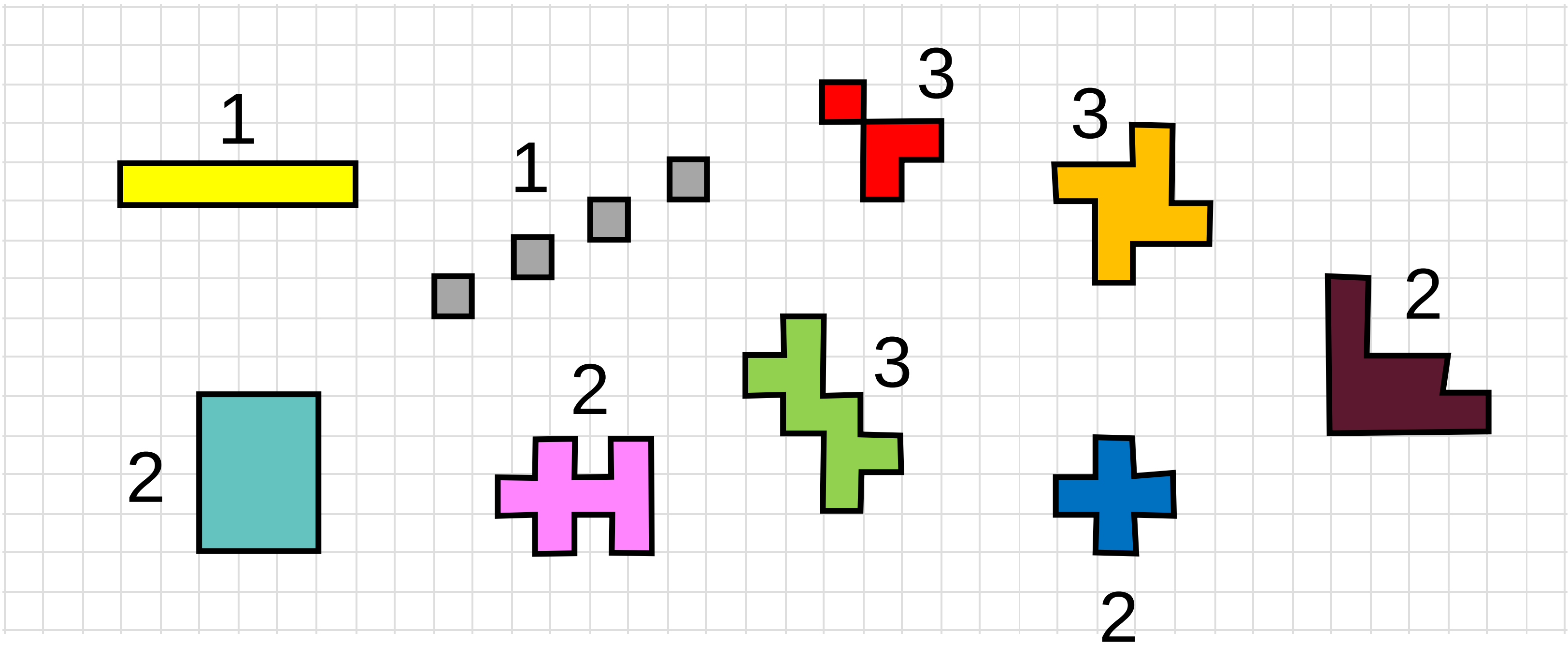

1
1


1
1
2
2


1
1
2
2
2


1
1
2
2
2
2


1
1
3
3
2
3
2
2
2
2



Can we say something more general than a few observations ?

Plan

I — The Game / Shooting Algorithms

II — Examples

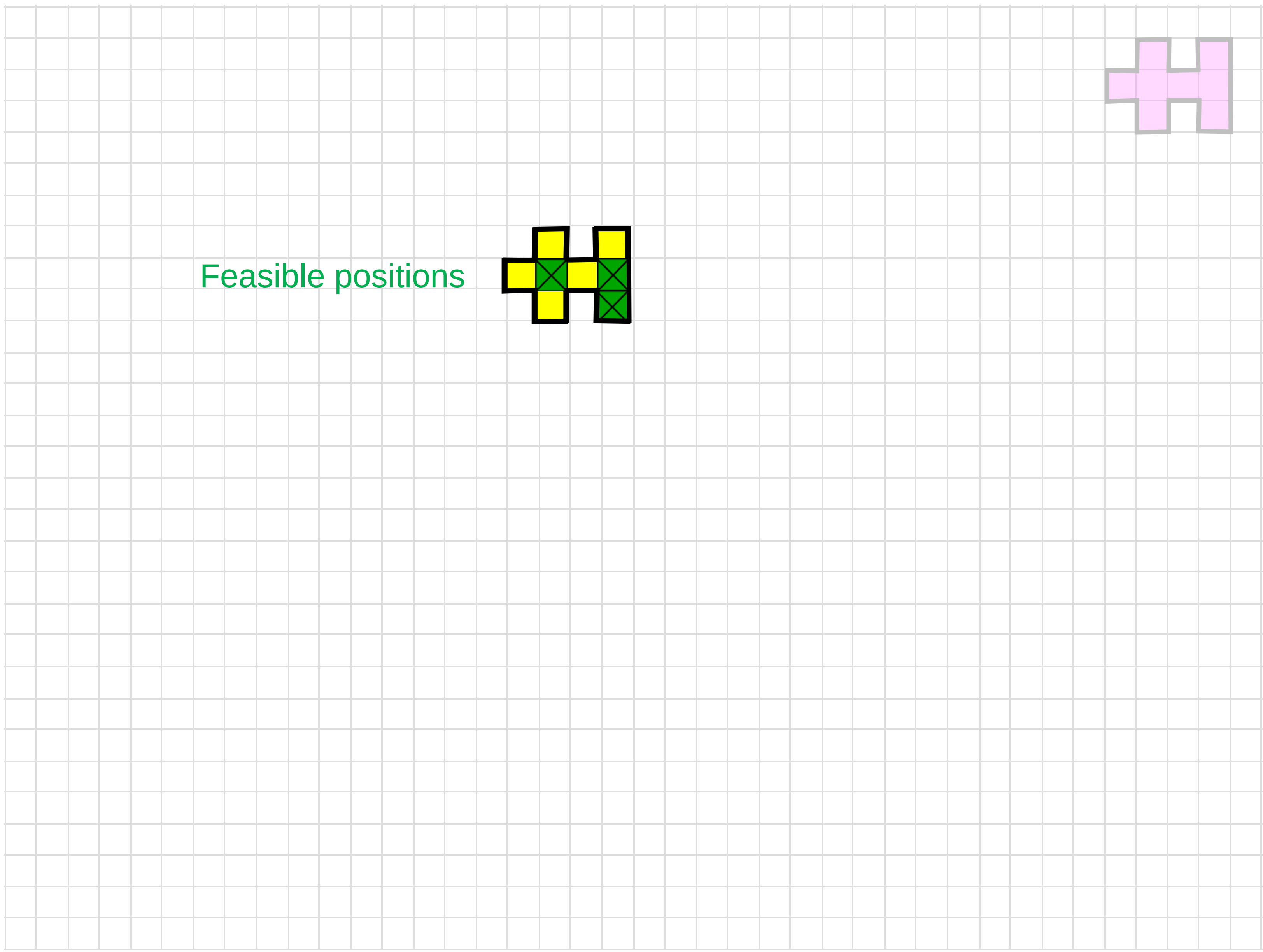
III — Results

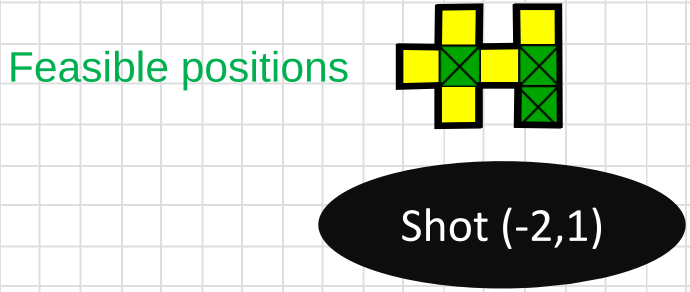
Plan

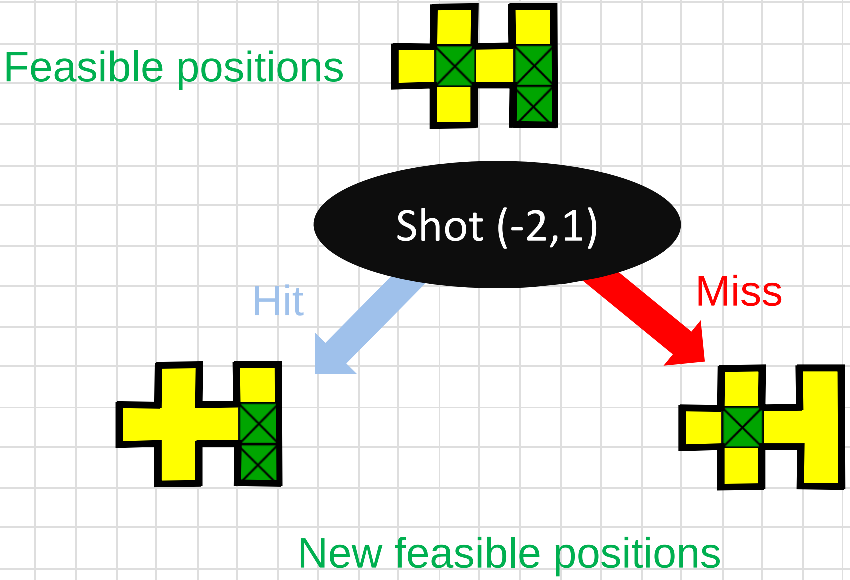
I The Game / Shooting Algorithms

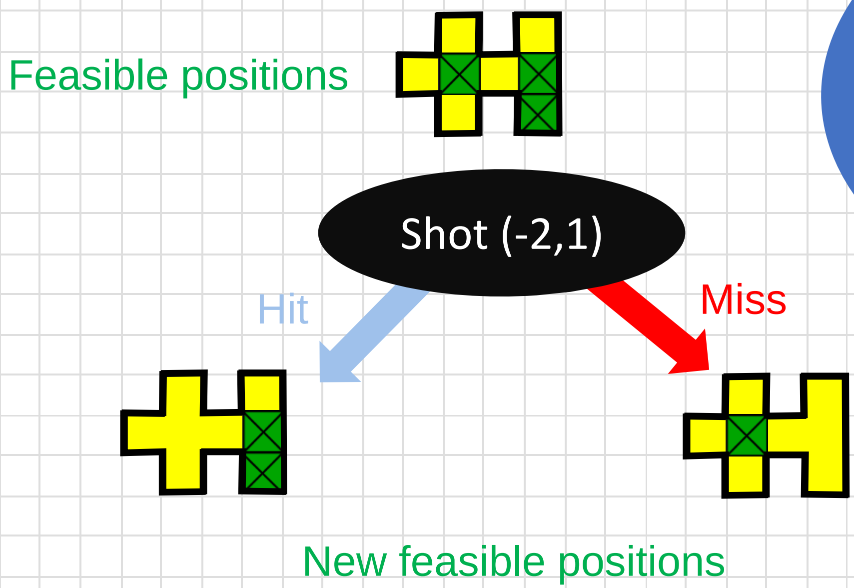
II Examples

III Results

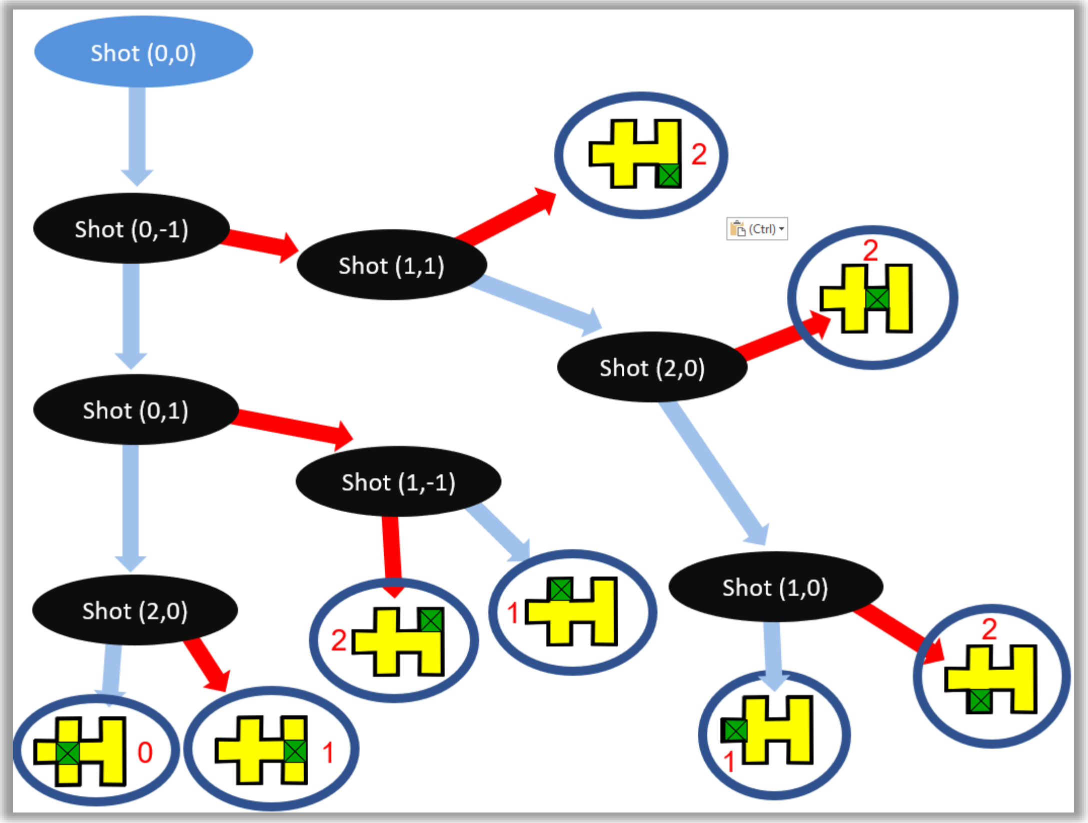

Feasible positions


Feasible positions
Shot (-2,1)

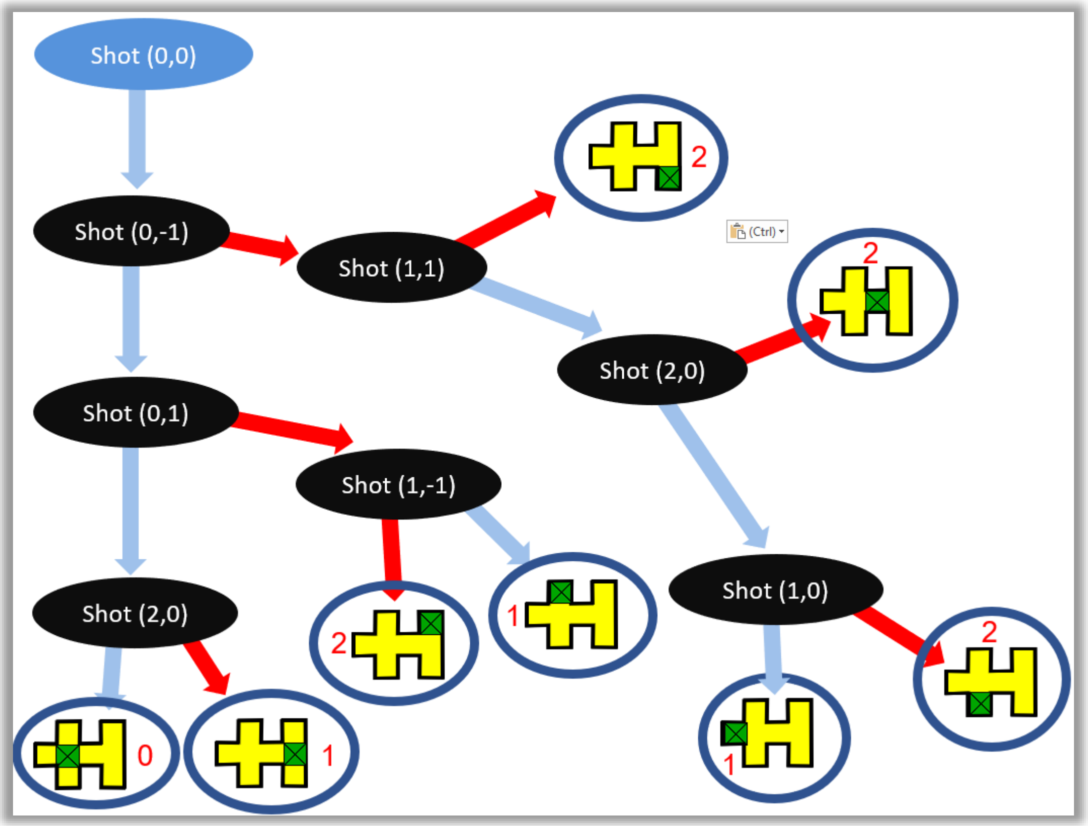

Feasible positions
Shot (-2,1)
Hit
Miss
New feasible positions



The set of the
feasible positions
decreases
at each shot!

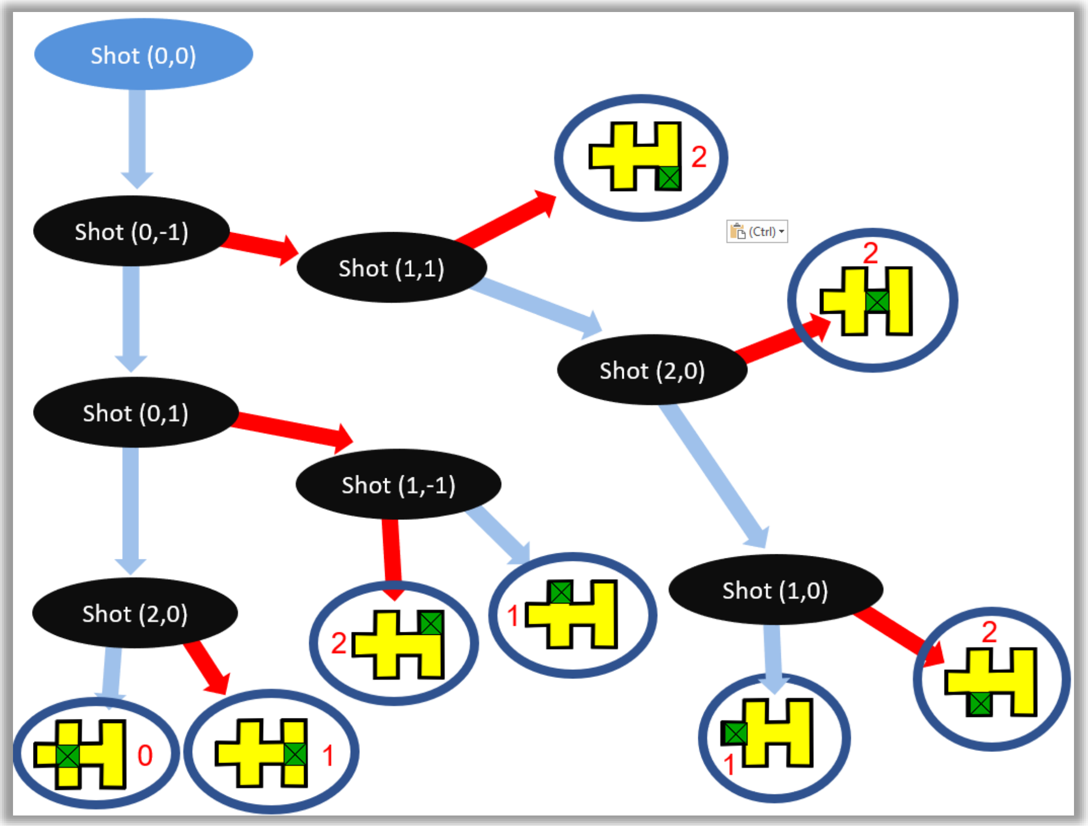


The set of the feasible positions decreases at each shot!



The set of the
feasible positions
decreases
at each shot!

height of the tree ≤ n
(n=size of the shape)



The set of the feasible positions decreases at each shot!

height of the tree ≤ n (n=size of the shape)

Number of misses ≤ n-1

Number of misses ≤ n-1

Bound 1

For any shape of size n

B-complexity(S) ≤ n-1

Number of misses ≤ n-1

Bound 1
For any shape of size n
B-complexity(S) ≤ n-1

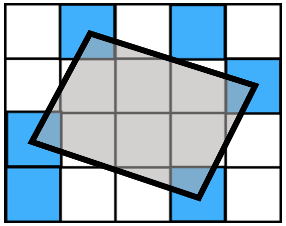
Tight ?

Bound 1
For any shape of size n
B-complexity(S) ≤ n-1

Tight ?

YES: for any parallelogram-free shape, we have
B-complexity(S)=n-1

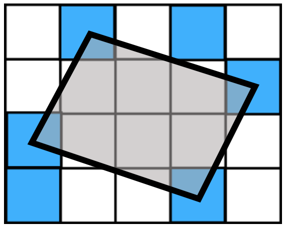
YES: for any parallelogram-free shape, we have
B-complexity(S)=n-1



Non Parallelogram-free shape

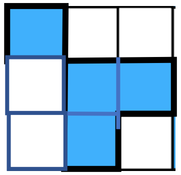
There exists 4 distinct points u,v,s,t in S with v-u=t-s

YES: for any parallelogram-free shape, we have B-complexity(S)=n-1



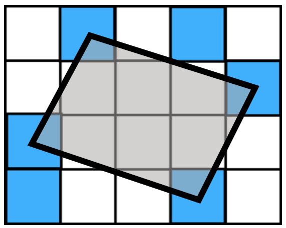
Non Parallelogram-free shape

There exists 4 distinct points u,v,s,t in S with v-u=t-s



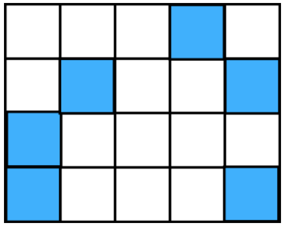
Parallelogram-free shape

YES: for any parallelogram-free shape, we have B-complexity(S)=n-1



Non Parallelogram-free shape

There exists 4 distinct points u,v,s,t in S with v-u=t-s

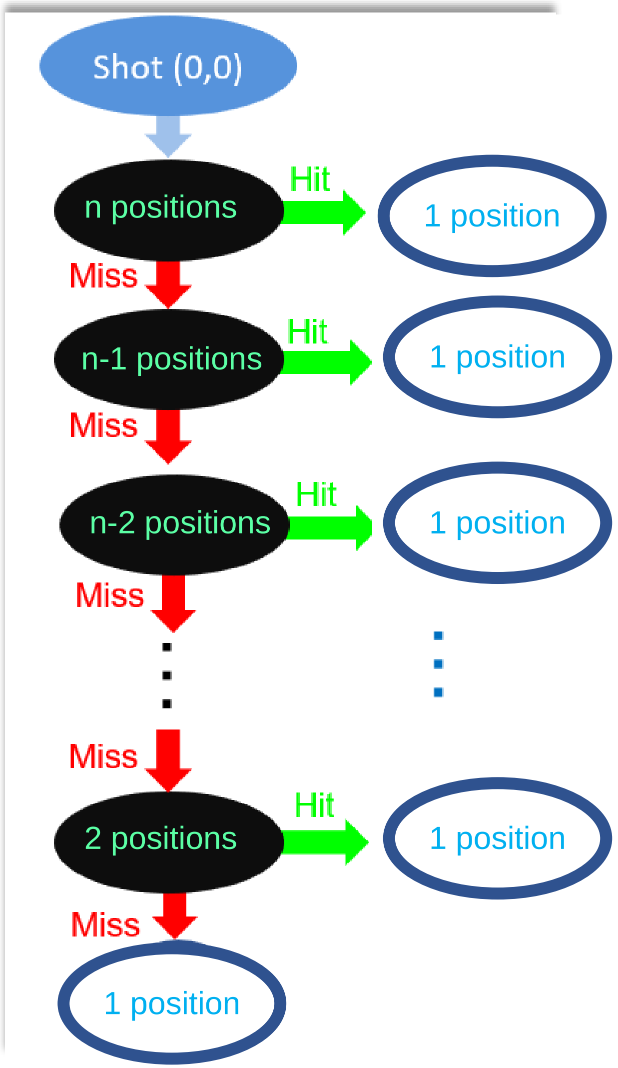


Parallelogram-free shape

YES: for any parallelogram-free shape, we have B-complexity(S)=n-1

What happens with parallelogram-free shapes ?

Bound 1
For any shape
B-complexity(S) ≤ n-1



For parallelogram-free shapes
B-complexity(S) = n -1

Bound 1

For any shape

B-complexity(S) ≤ n-1

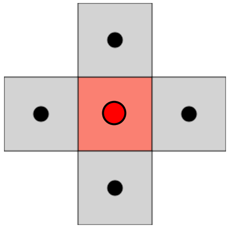
Bound 1
For any shape
B-complexity(S) ≤ n-1

Bound 2
For HV-convex polyominoes
B-complexity(S) = ….

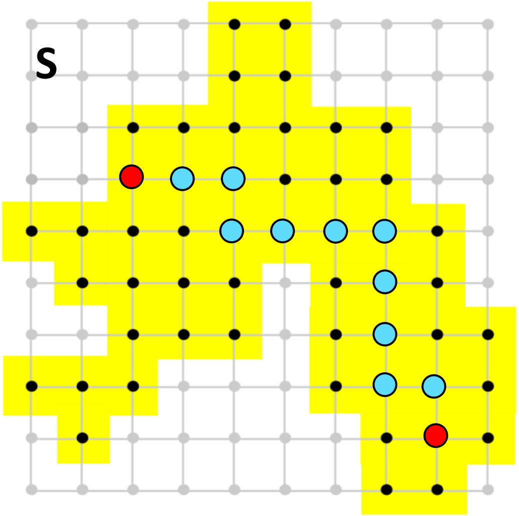
Bound 1
For any shape
B-complexity(S) ≤ n-1

Bound 2
For HV-convex polyominoes
B-complexity(S) = ….

What's this ?

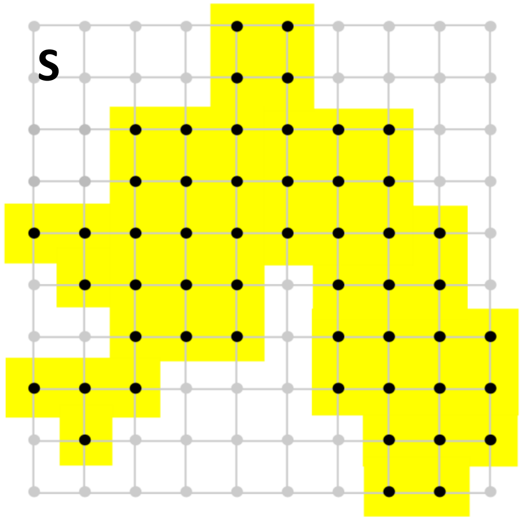


The four 4-neighbors
of a red point

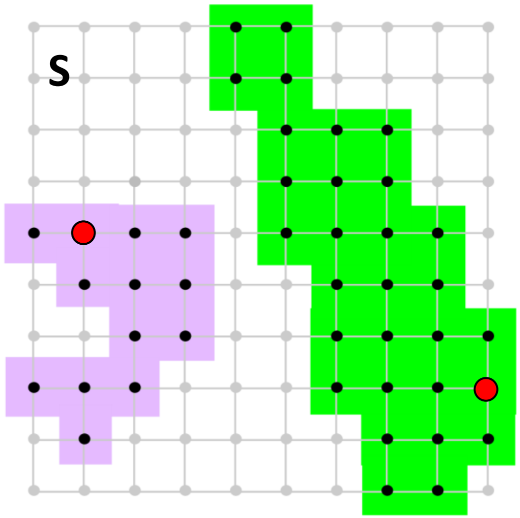


4-connected

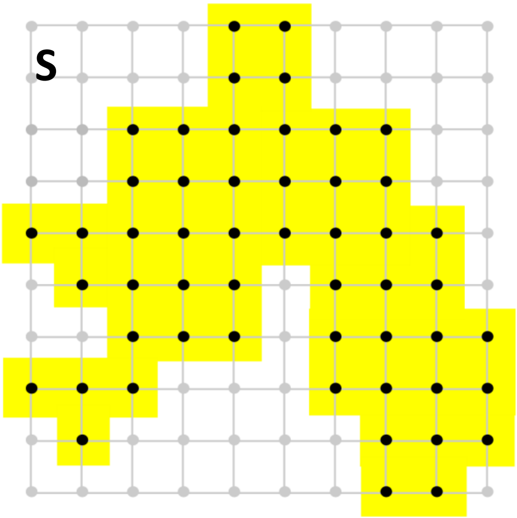
Any pair of squares/points is related
by a sequence of neighbors.



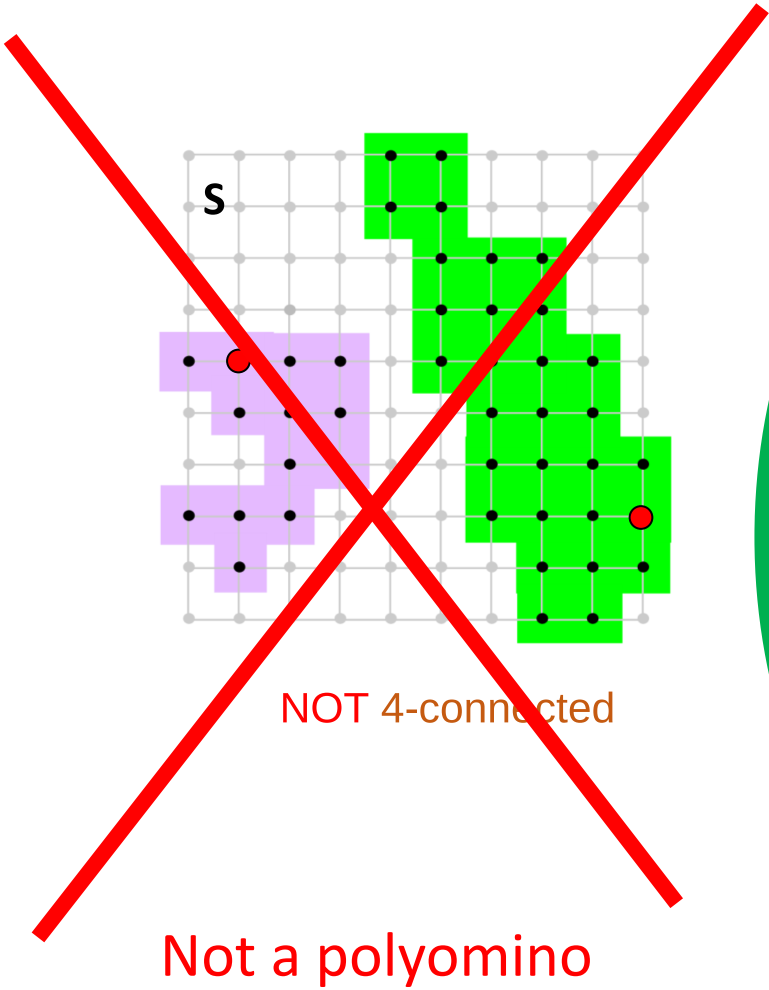
4-connected

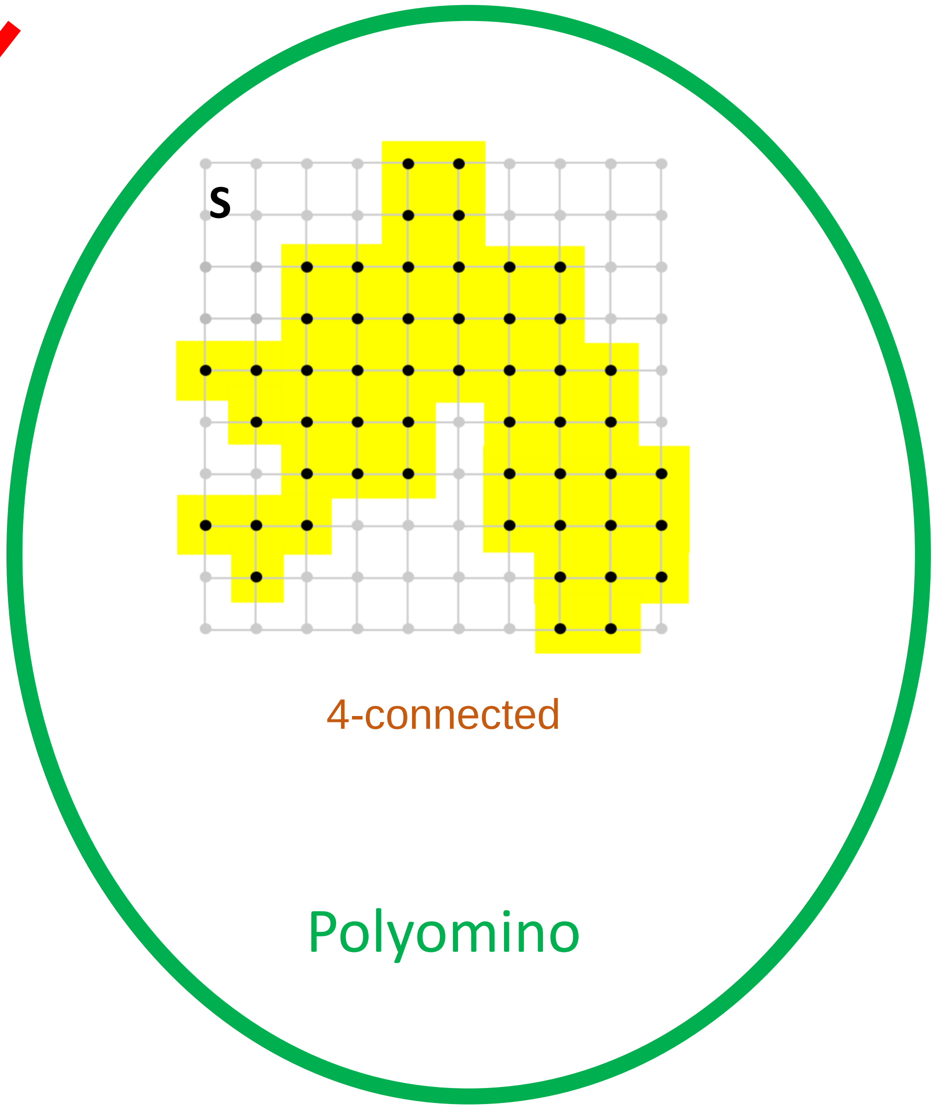


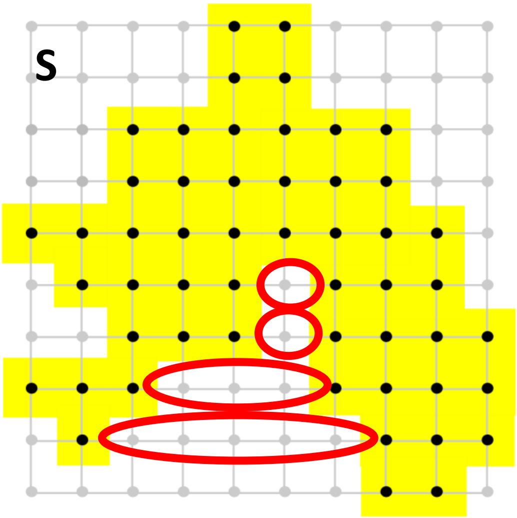
NOT 4-connected



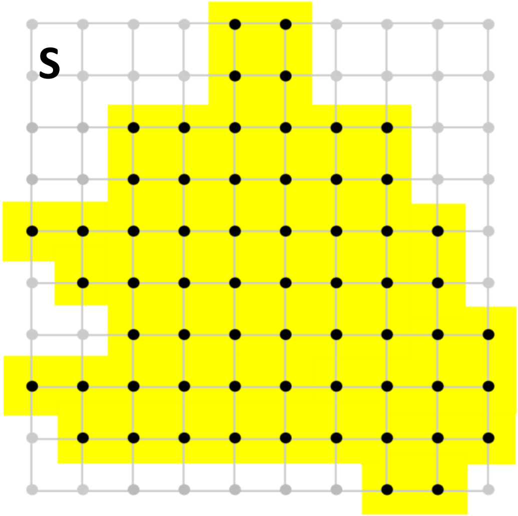
4-connected


S
NOT 4-connected
Not a polyomino

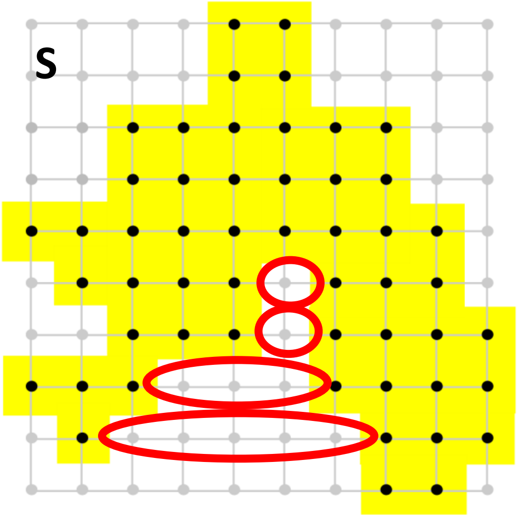

S
4-connected
Polyomino



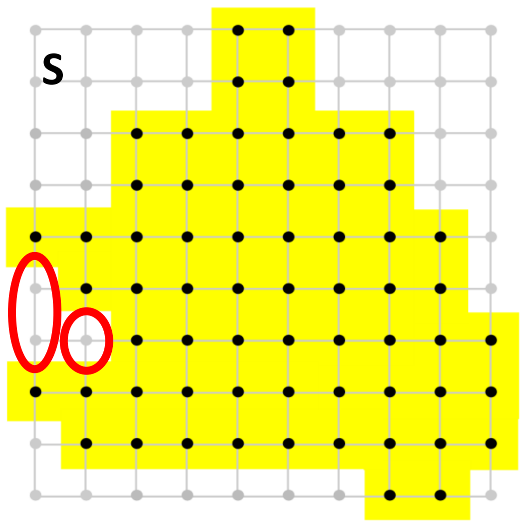
NOT *horizontally convex*



horizontally convex

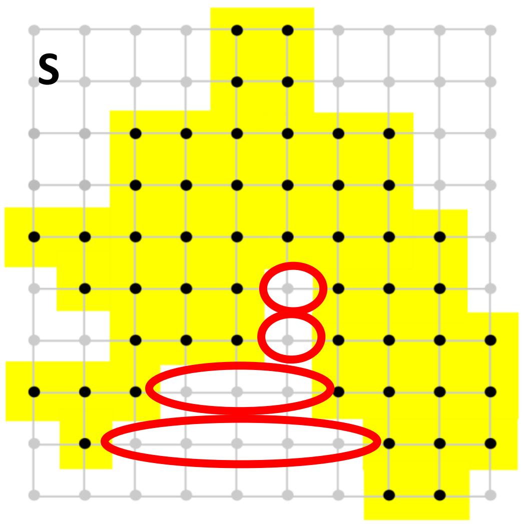


NOT *horizontally convex*

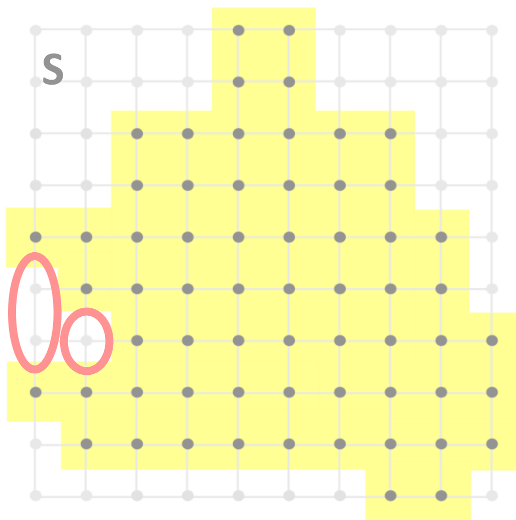


horizontally convex

NOT *vertically convex*

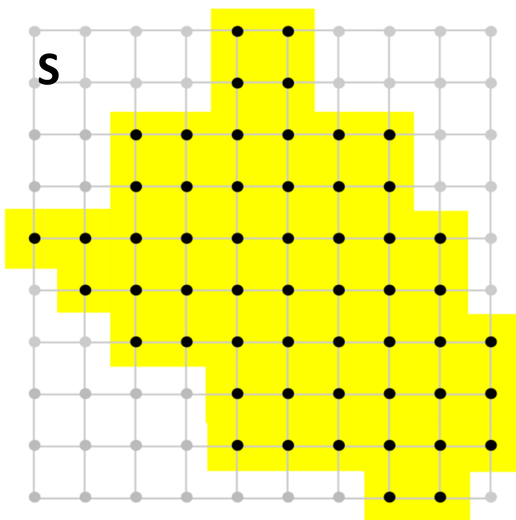


NOT *horizontally convex*



horizontally convex

NOT vertically convex



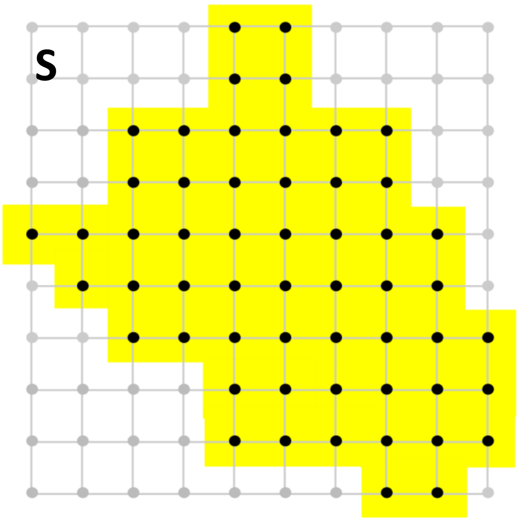
S is *vertically convex*.

S is *horizontally convex*.

S is *HV-convex*.

Bound 1
For any shape
B-complexity(S) ≤ n-1

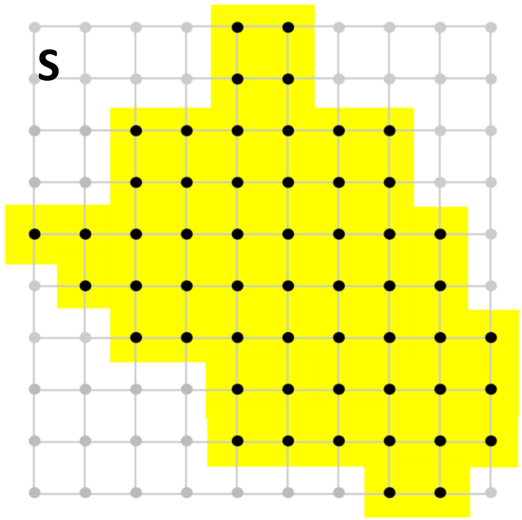
Bound 2
For HV-convex polyominoes
B-complexity(S) = ….



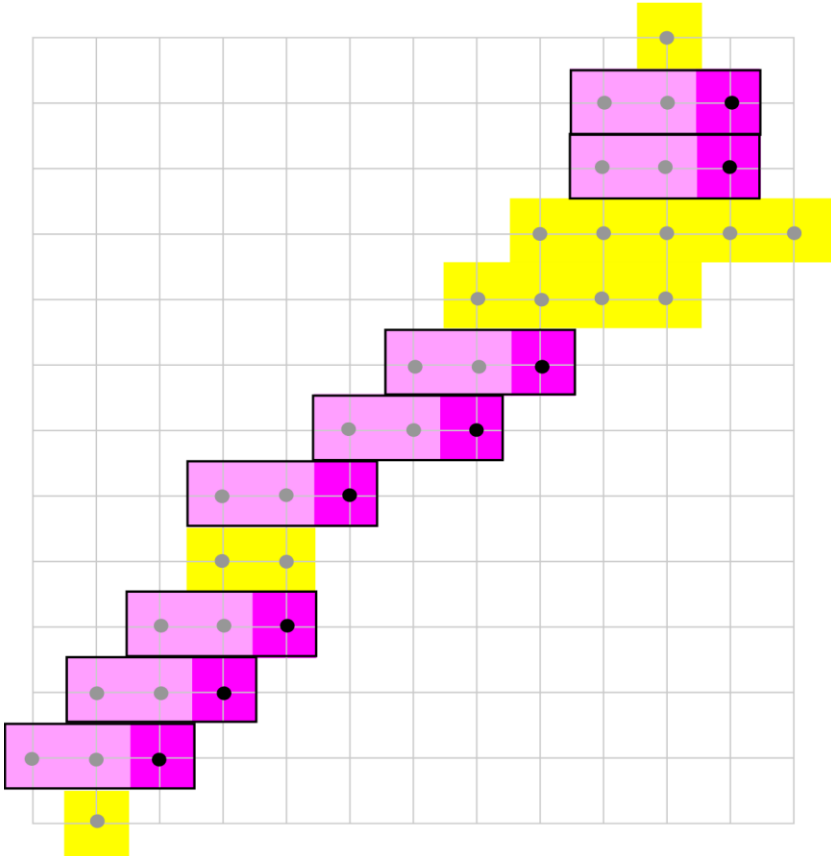
HV-convex polyomino

Bound 1
For any shape
B-complexity(S) ≤ n-1

Bound 2
For HV-convex polyominoes
B-complexity(S) = O(log(n))

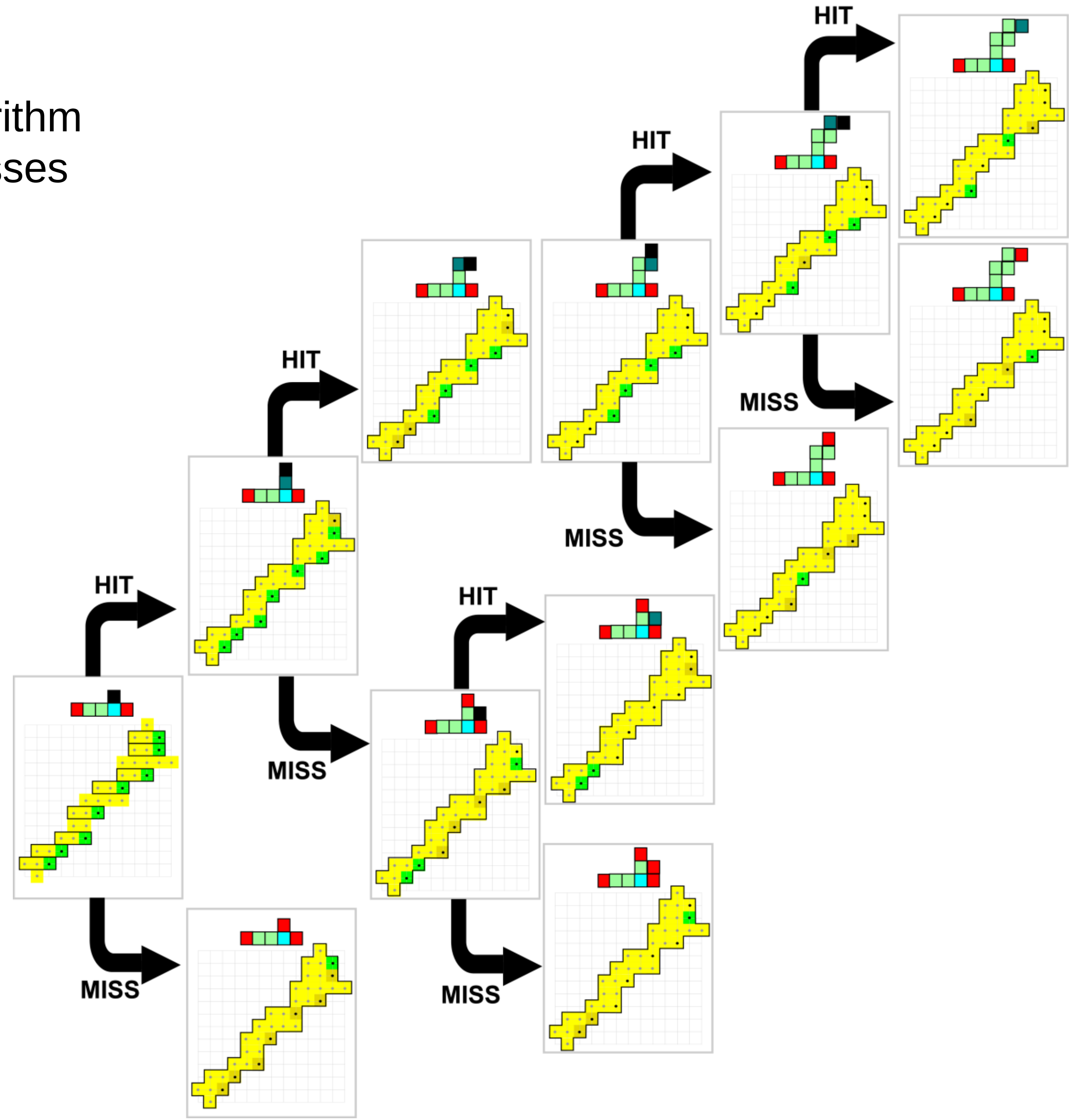


HV-convex polyomino



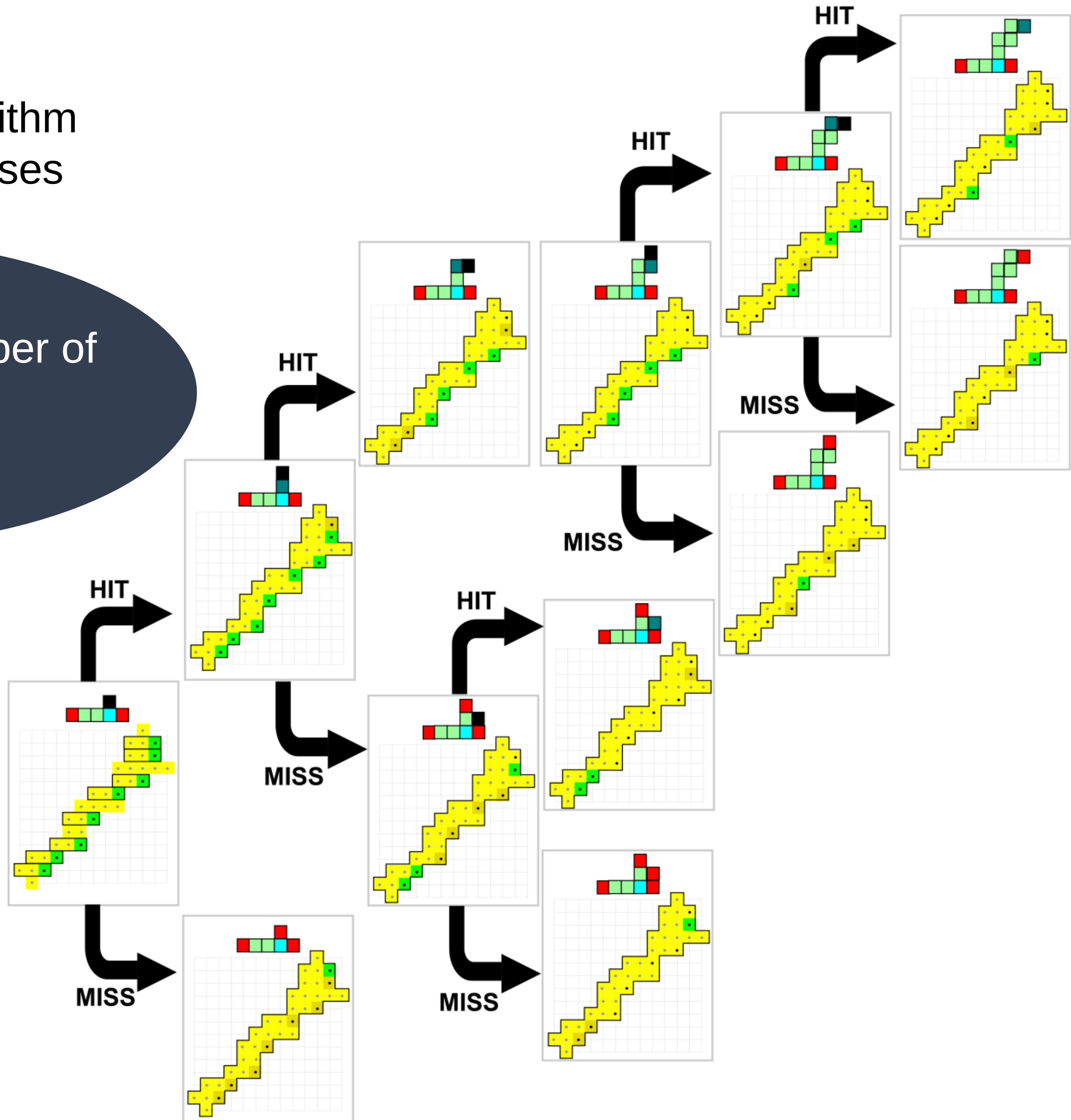
We use a property of monotonicity of HV-convex polyominoes…

A staircase shooting algorithm
with at most *O(log(n))* misses



A staircase shooting algorithm with at most *O(log(n))* misses

At each miss, the number of feasible positions is divided by 2.



Bound 1
For any shape
B-complexity(S) ≤ n-1

Bound 2
For HV-convex polyominoes
B-complexity(S) = O(log(n))

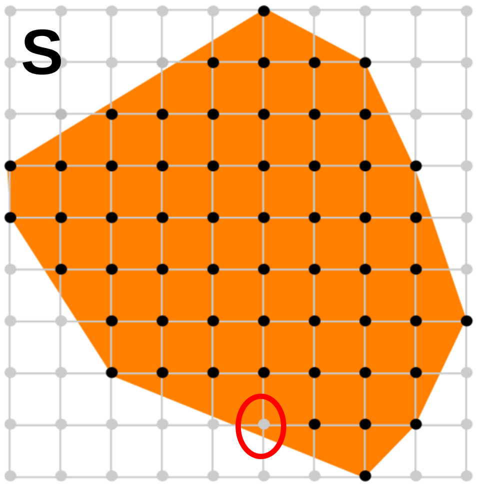
Bound 3
For digital convex sets
B-complexity(S) = …

Bound 1
For any shape
B-complexity(S) ≤ n-1

Bound 2
For HV-convex polyominoes
B-complexity(S) = O(log(n))

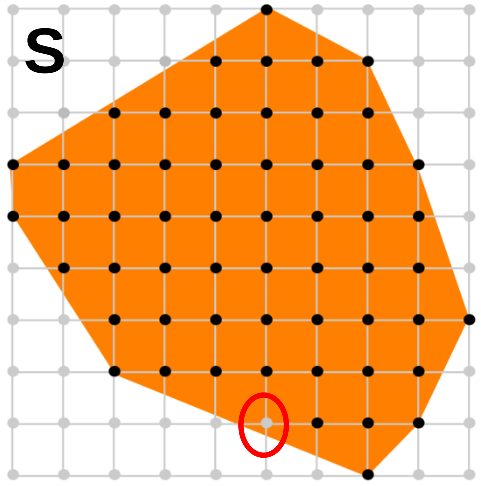
Bound 3
For digital convex sets
B-complexity(S) = …

And what’s this ?

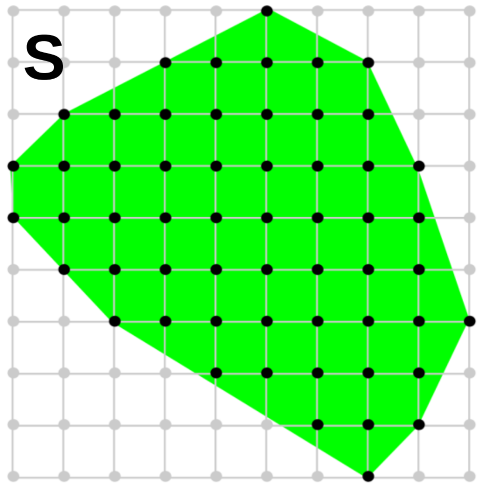


S is *not digital convex*.

Digital convexity means
S is equal to its intersection with its (real) *convex hull*.



S is *not digital convex*.



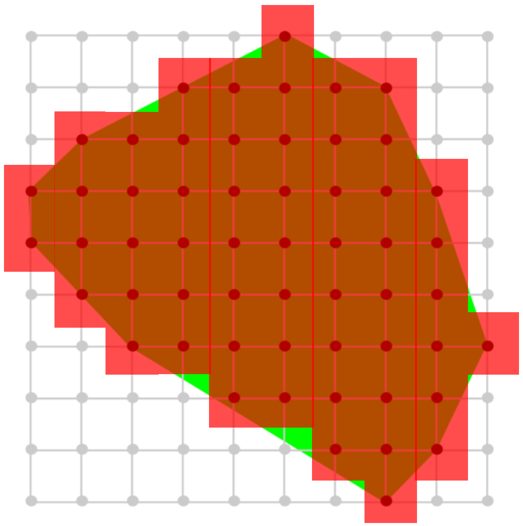
S is *digital convex*.

Digital convexity means
S is equal to its intersection with its (real) *convex hull*.

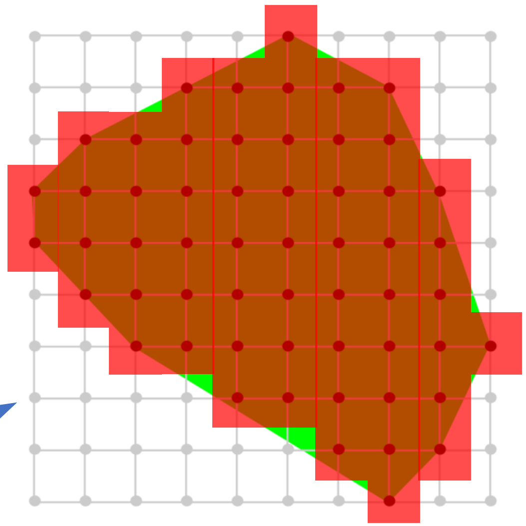
Bound 1
For any shape
B-complexity(S) ≤ n-1

Bound 2
For HV-convex polyominoes
B-complexity(S) = O(log(n))

Bound 3
For digital convex sets
B-complexity(S)=O(…)



Digital convex shape



Digital convex shape

We need a property of these convex shapes…

Zimbra: Réception (3632) | Quand l'Ivermectine attire le | Dashboard | Cours : Mathématiques DUT | Efficient Algorithms for Batt | Blaschke–Lebesgue theorem

https://en.wikipedia.org/wiki/Blaschke–Lebesgue_theorem 130%

Google Qwant Fip Translate Maps Facebook Instagram Mail Météo News Maison Shops wiki Fun words synonymes UCA Odin Theillard Blockly Games

Not logged in Talk Contributions Create account Log in

WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Current events
Random article
About Wikipedia
Contact us
Donate

Contribute

Help
Learn to edit
Community portal
Recent changes
Upload file

Tools

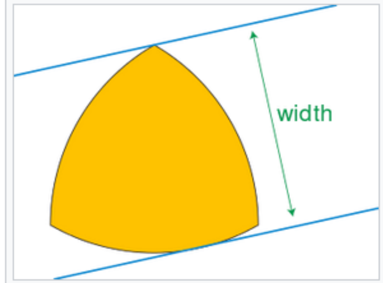
What links here
Related changes
Special pages
Permanent link
Page information
Cite this page

Article Talk Read Edit View history Search Wikipedia

Blaschke–Lebesgue theorem

From Wikipedia, the free encyclopedia

In plane geometry the **Blaschke–Lebesgue theorem** states that the Reuleaux triangle has the least area of all curves of given constant width.[1] In the form that every curve of a given width has area at least as large as the Reuleaux triangle, it is also known as the **Blaschke–Lebesgue inequality**.[2] It is named after Wilhelm Blaschke and Henri Lebesgue, who published it separately in the early 20th century.

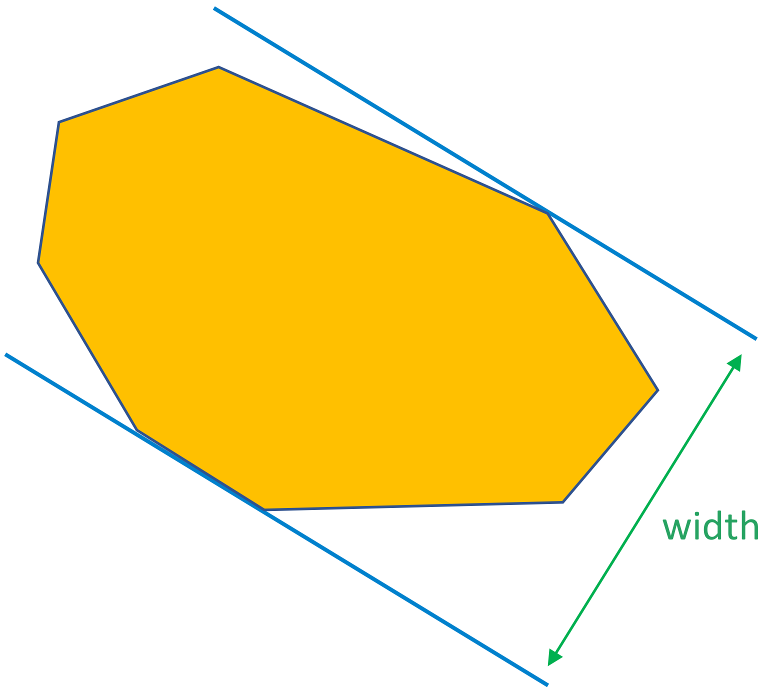


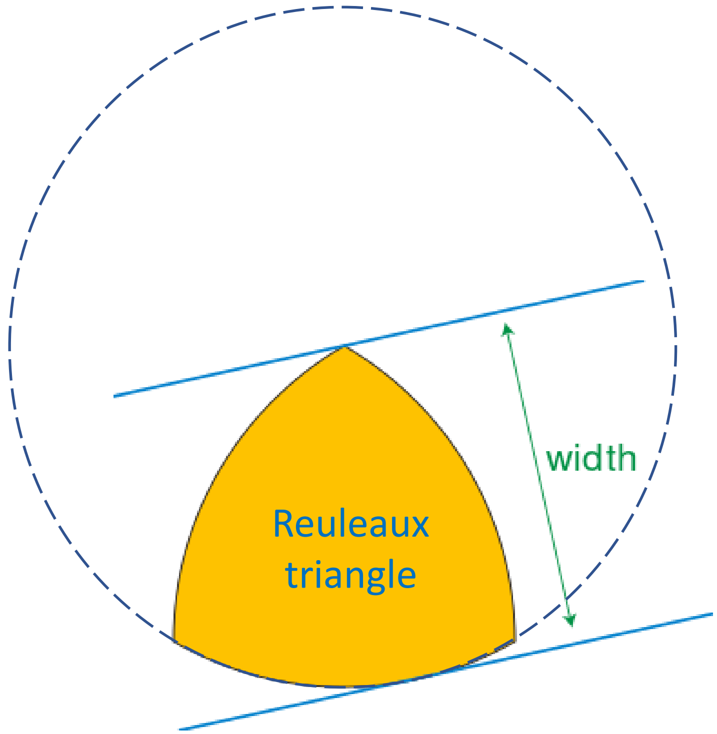
A Reuleaux triangle, a curve of constant width whose area is minimum among all convex sets with the same width

Contents [hide]

Statement [edit]

The width of a convex set K in the Euclidean plane is defined as the minimum distance between any two parallel lines that enclose it. The two minimum-distance lines are both necessarily tangent lines to K, on opposite sides. A curve of constant width is the boundary of a convex set with the property that, for every direction of parallel lines, the two tangent lines with that direction that are tangent to opposite sides of the curve are at a distance equal to the width. These curves include both the circle and the Reuleaux triangle, a curved triangle formed from arcs of three equal-radius



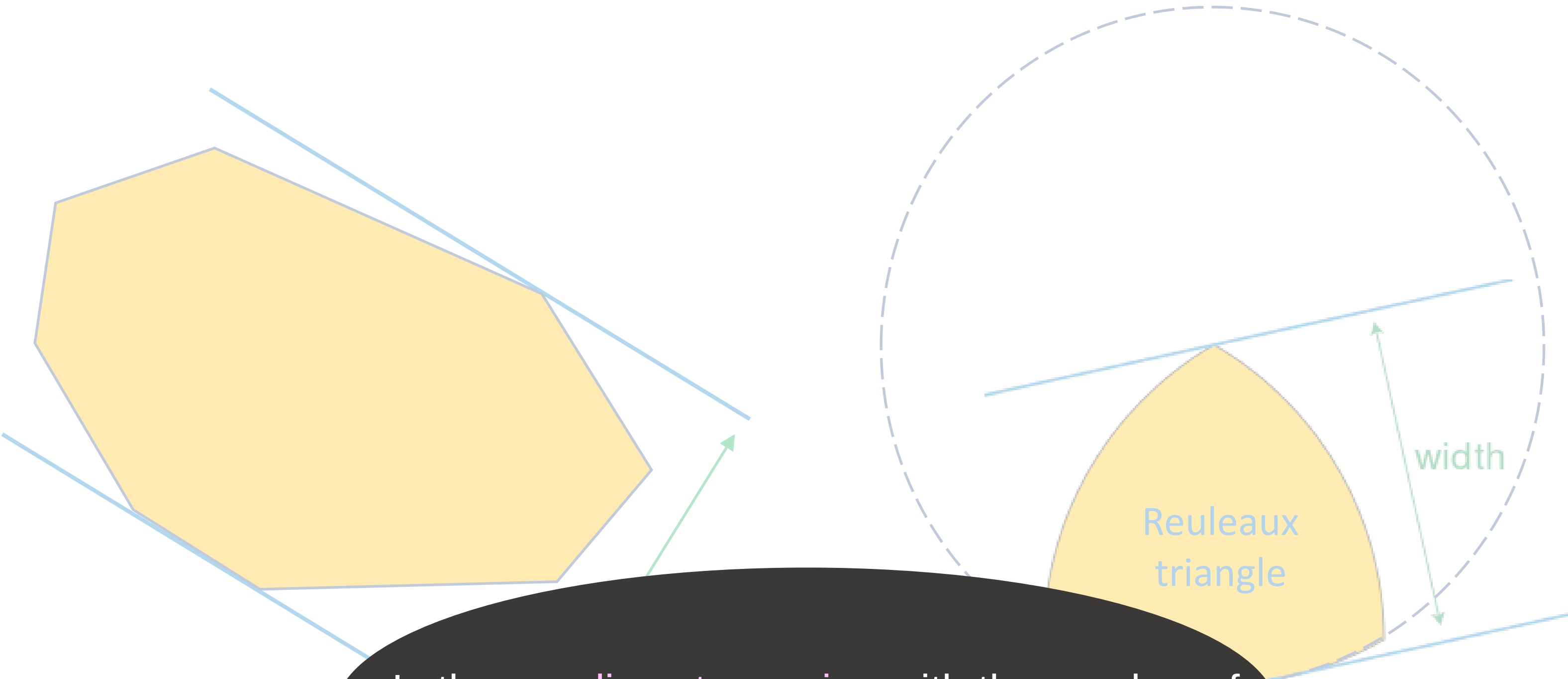


Blaschke-Lebesgue Inequality *(1914)*

The area of a convex shape is bounded by

$$area \geq \frac{1}{2}\left(\pi - \sqrt{3}\right) width^2$$

(Equality for Reuleaux triangles).



Is there a discrete version with the number of points instead of the area ?

Blaschke-Lebesgue Inequality *(1914)*

The area of a convex shape is bounded by

(Equality for Reuleaux triangles).

$$area \geq \frac{1}{2}\left(\pi - \sqrt{3}\right) width^2$$

Barany-Füredi Inequality *(2001)*

The area of a discrete shape *S* is bounded by $width \leq \lfloor 4/3\ diam \rfloor + 1$

where $diam$ is the maximum number of points of *S* on a line and $width$ is the arithmetic width (tight bound).

Is there a discrete version with the number of points instead of the area ?

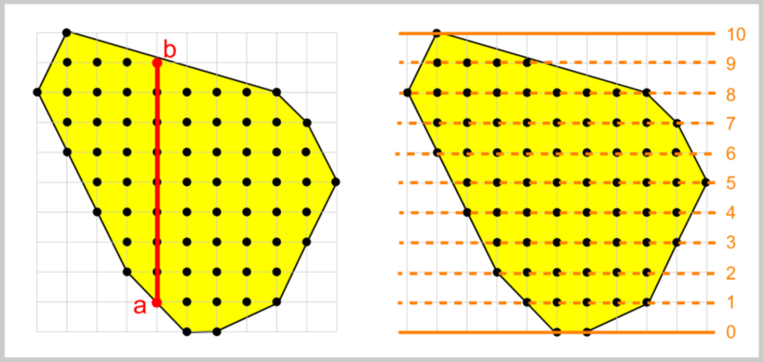
Blaschke-Lebesgue Inequality *(1914)*

The area of a convex shape is bounded by $area \geq \frac{1}{2}\left(\pi - \sqrt{3}\right) width^2$

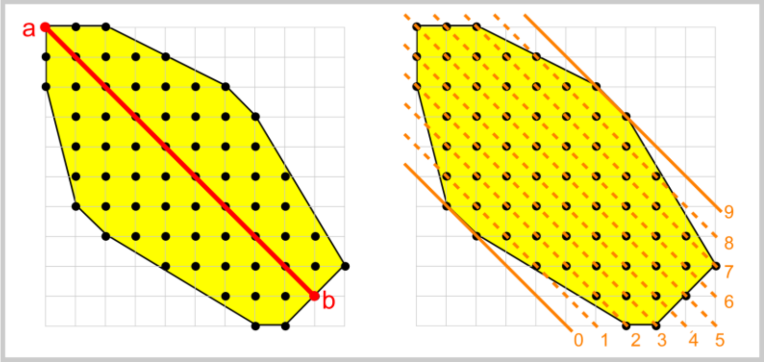
(Equality for Reuleaux triangles).

Barany-Füredi Inequality *(2001)*

The area of a discrete shape *S* is bounded by $width \leq |4/3\ diam| + 1$
where $diam$ is the maximum number of points of *S* on a line
and $width$ is the arithmetic width (tight bound).



$diam = 9$ $width = 10$



$diam = 10$ $width = 9$

Blaschke-Lebesgue Inequality *(1914)*

The area of a convex shape is bounded by $area \geq \frac{1}{2}(\pi - \sqrt{3})width^2$

(Equality for Reuleaux triangles).

Barany-Füredi Inequality *(2001)*

The area of a discrete shape *S* is bounded by $width \leq |4/3\ diam| + 1$
where $diam$ is the maximum number of points of *S* on a line
and $width$ is the arithmetic width (tight bound).

Pick formula & …

***Discrete* Blaschke-Lebesgue Inequality (L. Crombez, G. Da Fonseca, Y.G)**

The number n of a digital convex shape *S* is bounded by $n \geq \frac{1}{4} width^2 + 2$

Blaschke-Lebesgue Inequality *(1914)*

The area of a convex shape is bounded by $area \geq \frac{1}{2}\left(\pi - \sqrt{3}\right) width^2$
(Equality for Reuleaux triangles).

Discrete Blaschke-Lebesgue Inequality (L. Crombez, G. Da Fonseca, Y.G)

The number n of a digital convex shape S is bounded by $n \geq \frac{1}{4} width^2 + 2$

(not tight)

Blaschke-Lebesgue Inequality *(1914)*

The area of a convex shape is bounded by $area \geq \frac{1}{2}(\pi - \sqrt{3}) width^2$

(Equality for Reuleaux triangles).

Discrete Blaschke-Lebesgue Inequality (L. Crombez, G. Da Fonseca, Y.G)

The number n of a digital convex shape S is bounded by $n \geq \frac{1}{4} width^2 + 2$

(not tight)

We use it for bounding the maximum number of misses of our algorithm…

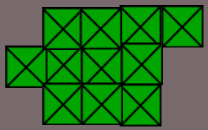
Discrete Blaschke-Lebesgue Inequality (L. Crombez, G. Da Fonseca, Y.G)

The number n of a digital convex shape *S* is bounded by $n \geq \frac{1}{4} width^2 + 2$

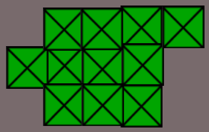
(not tight)

At each step of the algorithm

At each step of the algorithm

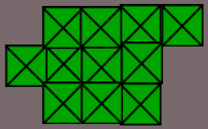
The set of the feasible positions  is digital convex….

At each step of the algorithm

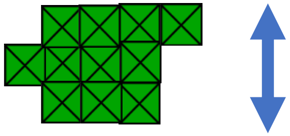
The set of the feasible positions  is digital convex….

We compute its arithmetic width….and shot in the direction of the lines…

At each step of the algorithm

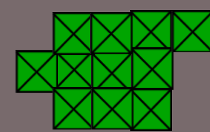
The set of the feasible positions  is digital convex….

We compute its arithmetic width….and shot in the direction of the lines…

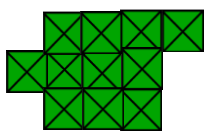


width=3 (only 3 lines in direction (1,0))

At each step of the algorithm

The set of the feasible positions  is digital convex….

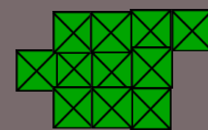
We compute its arithmetic width….and shot in the direction of the lines…



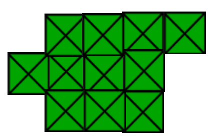
width=3 (only 3 lines in direction (1,0))

Shot k(1,0) until finding the two first misses with a positive and negative integer k…

At each step of the algorithm

The set of the feasible positions  is digital convex….

We compute its arithmetic width….and shot in the direction of the lines…



width=3 (only 3 lines in direction (1,0))

Shot k(1,0) until finding the two first misses with a positive and negative integer k…

With these 2 new misses, we go from Q feasible positions
to at most $Q^{3/4}$ feasible positions

provides the *O(log(log(n)))* bound

With these 2 new misses, we go from Q feasible positions
to at most $Q^{3/4}$ feasible positions

Plan

I The Game / Shooting Algorithms

II Examples

III Results

Conclusion

Plan

I The Game / Shooting Algorithms

II Examples

III Results

Conclusion

Conclusion

Bound 1
For any shape
B-complexity(S) $\leq$ n-1

Bound 2
For HV-convex polyominoes
B-complexity(S) =O(log(n))

Bound 3
For digital convex polyominoes
B-complexity(S) =O(log(log(n)))

Conclusion

Bound 1
For any shape
B-complexity(S) ≤ n-1

Bound 2
For HV-convex polyominoes
B-complexity(S) =O(log(n))

Bound 3
For digital convex polyominoes
B-complexity(S) =O(log(log(n)))

A lot of open questions…

Complexity
of computing the B-complexity
of a given shape ?

A lot of open questions…

Conclusion

B-complexity(A U B) ≤ B-complexity(A) + B-complexity(B) ?

Complexity
of computing the B-complexity
of a given shape ?

A lot of open questions…

Conclusion

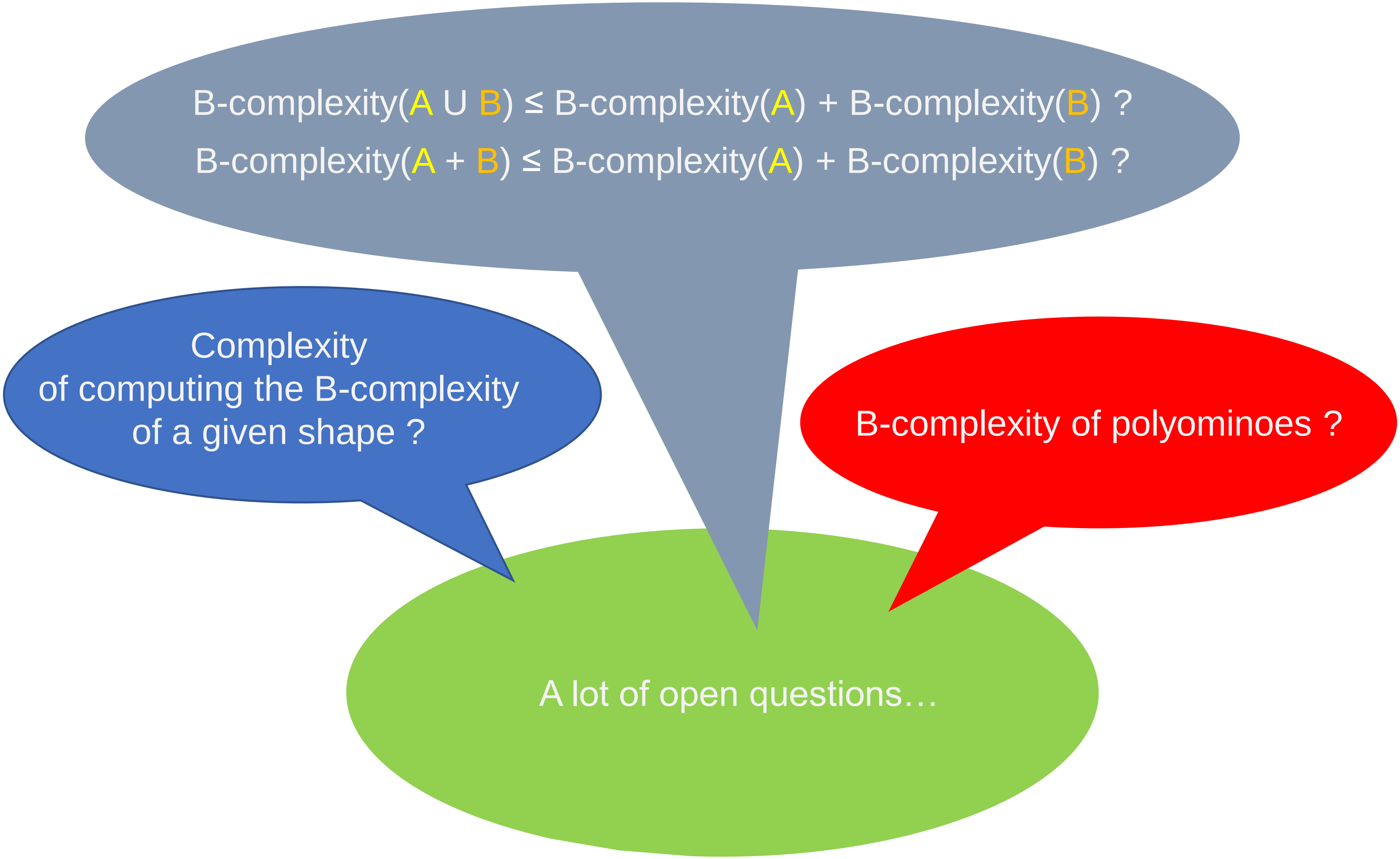
B-complexity(A U B) ≤ B-complexity(A) + B-complexity(B) ?

B-complexity(A + B) ≤ B-complexity(A) + B-complexity(B) ?

Complexity
of computing the B-complexity
of a given shape ?

A lot of open questions…

Conclusion


B-complexity(A U B) ≤ B-complexity(A) + B-complexity(B) ?
B-complexity(A + B) ≤ B-complexity(A) + B-complexity(B) ?
Complexity of computing the B-complexity of a given shape ?
B-complexity of polyominoes ?
A lot of open questions…

Conclusion

B-complexity(A U B) ≤ B-complexity(A) + B-complexity(B) ?

B-complexity(A + B) ≤ B-complexity(A) + B-complexity(B) ?

What I like with the B-complexity ?

What I like with the B-complexity ?

This algorithmic game can be played in any group!!!!!!

Z Z^2 …. Z^d Z/nZ $GL_n(R)$

What I like with the B-complexity ?

This algorithmic game can be played in any group!!!!!!

Z $\quad$ Z^2 $\quad$ .... $\quad$ Z^d $\quad$ Z/nZ $\quad$ $GL_n(R)$

with any shape i.e finite set

Last word

What I like with the B-complexity ?

This algorithmic game can be played in any group!!!!!!

Z Z^2 …. Z^d Z/nZ $GL_n(R)$

with any shape i.e finite set

It is invariant by morphisms (injective* on S-S)

$* \ for\ any\ u, v, u', v'\ in\ S:\ \ f(u-v) = f(u'-v')\ implies\ u-v = u'-v'$

Thanks a lot for your attention…

Some questions ?