APRiL: a close-to-natural language for geometric proofs

Mikhail Koroteev, Ilya Krukov, Artem Smirnov
Horis International Ltd

Games that teach a love of math

Euclidea

Geometry brainteaser

★★★★★ 4.8

Pythagorea

Magic of the square grid

★★★★★ 4.8

XSection

Solid geometry

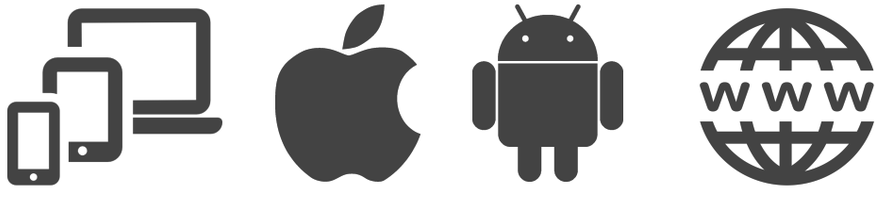
★★★★★ 4.9

Tchisla

Counting numbers is fun

★★★★★ 4.8

math apps for everyone



www.euclidea.xyz

Euclidea

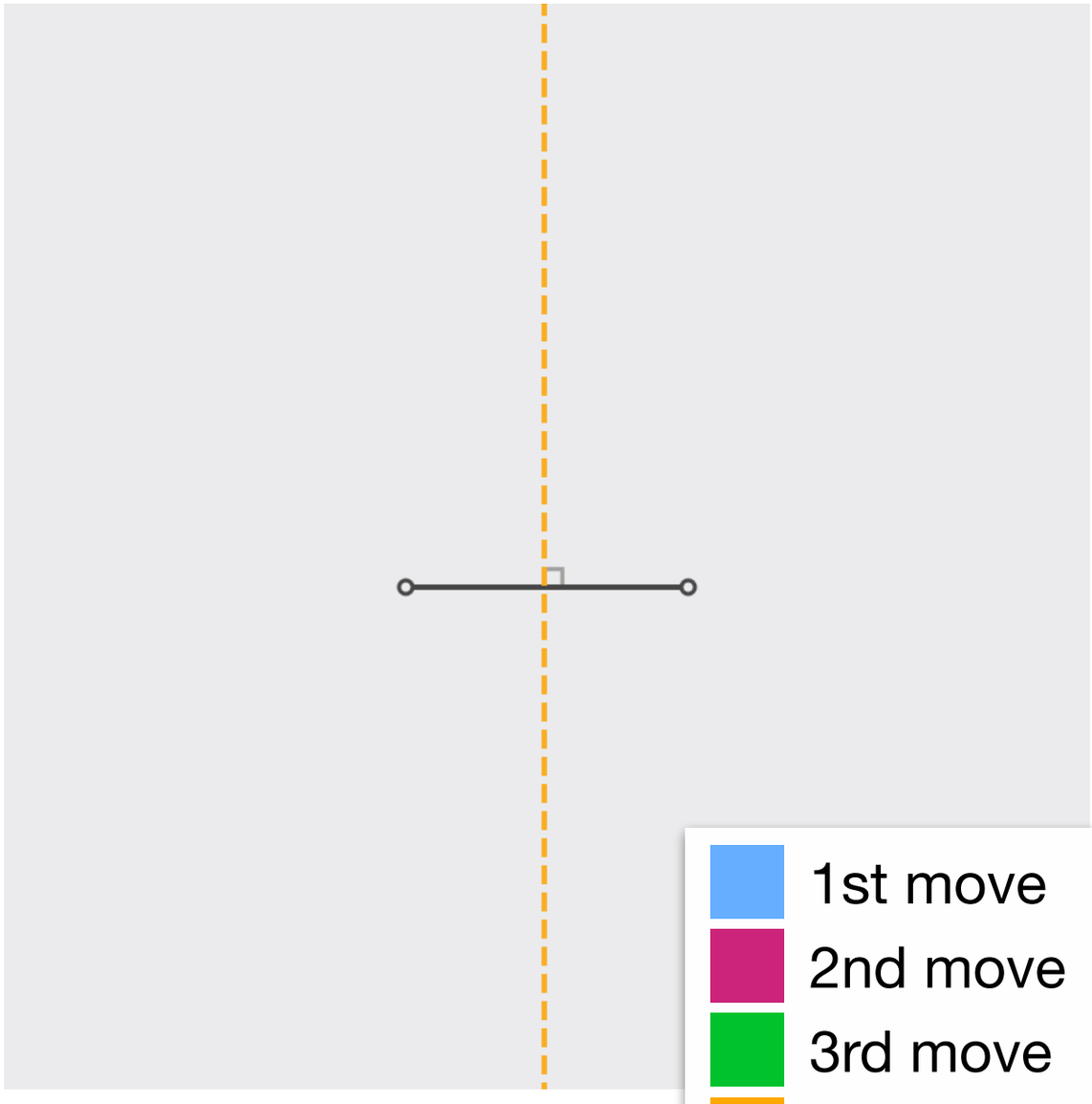
Geometric constructions with compass and straightedge

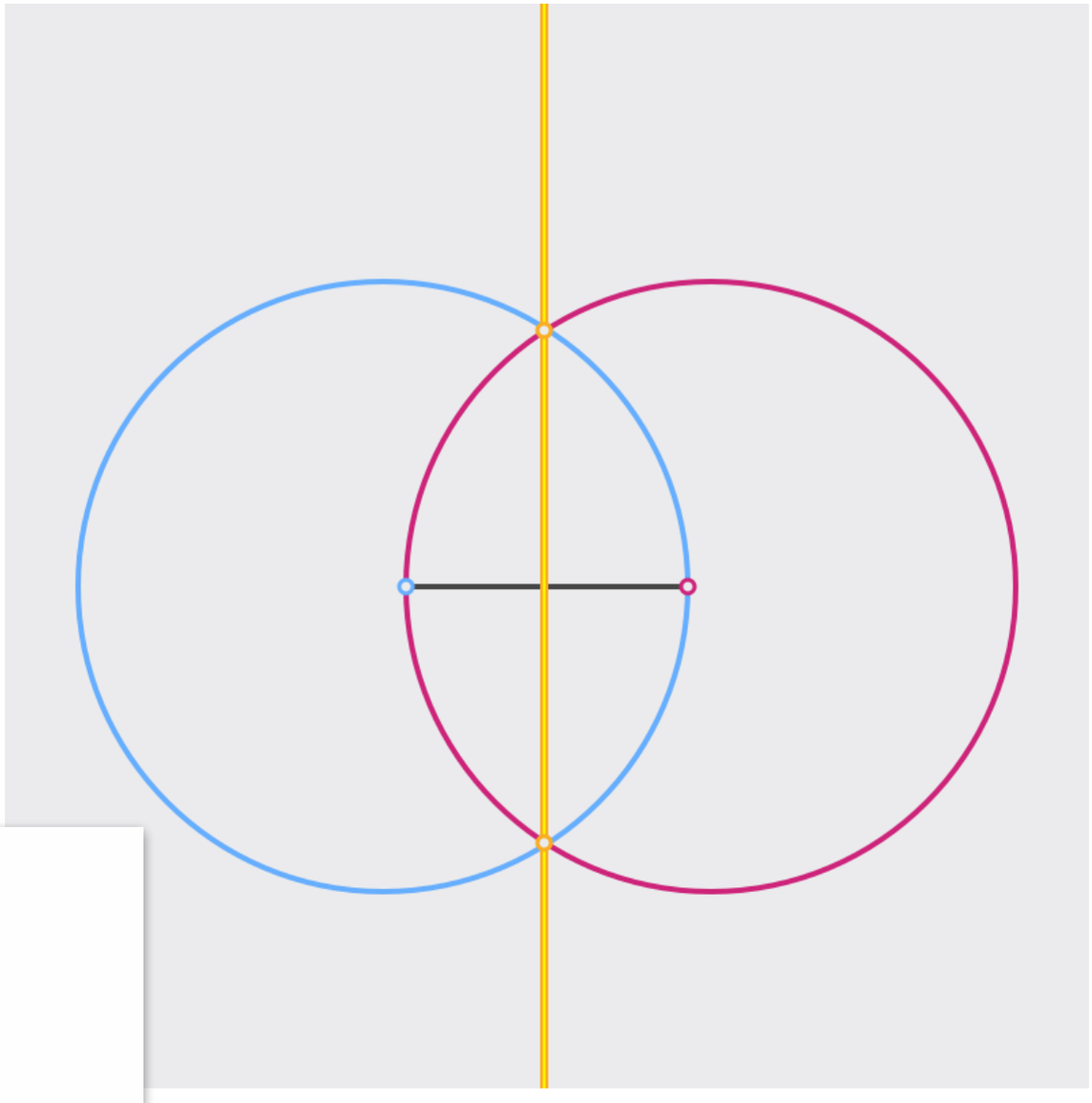
https://www.euclidea.xyz

3 million downloads

Euclidea, problem 1.2

Construct the perpendicular bisector of the segment.

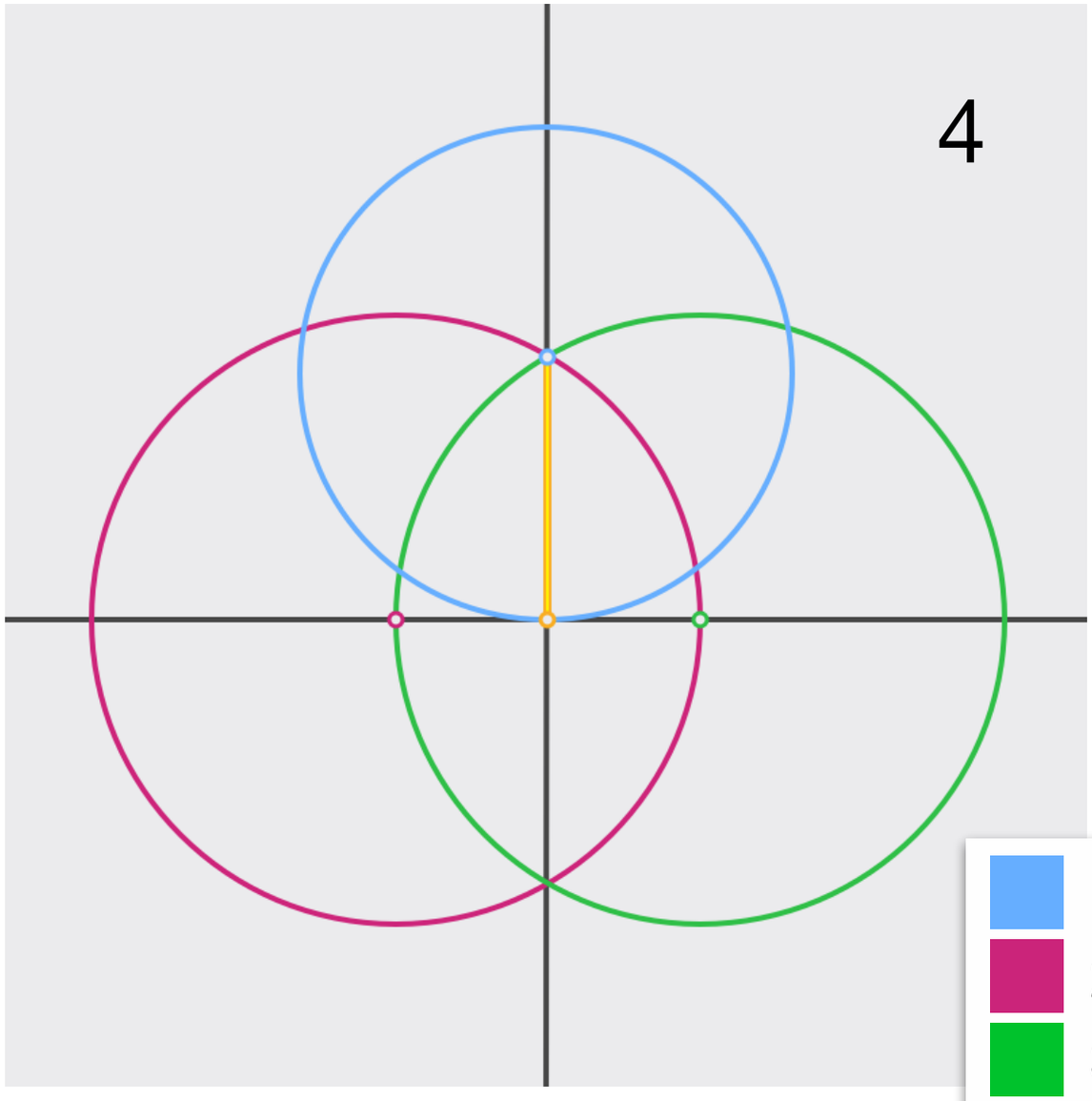


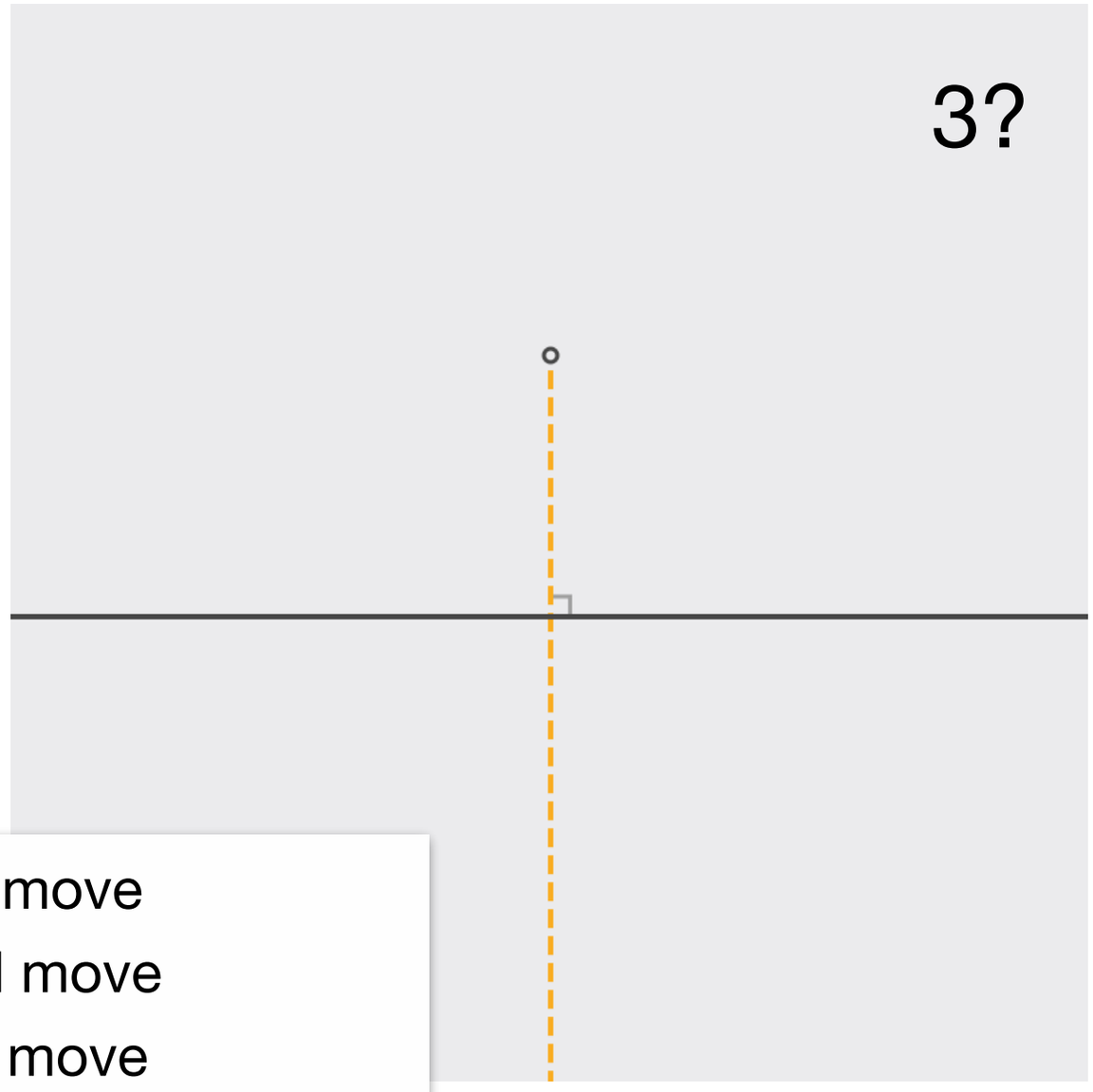


1st move
2nd move
3rd move
answer (last move)

Euclidea, problem 2.6

Drop a perpendicular from the point to the line.

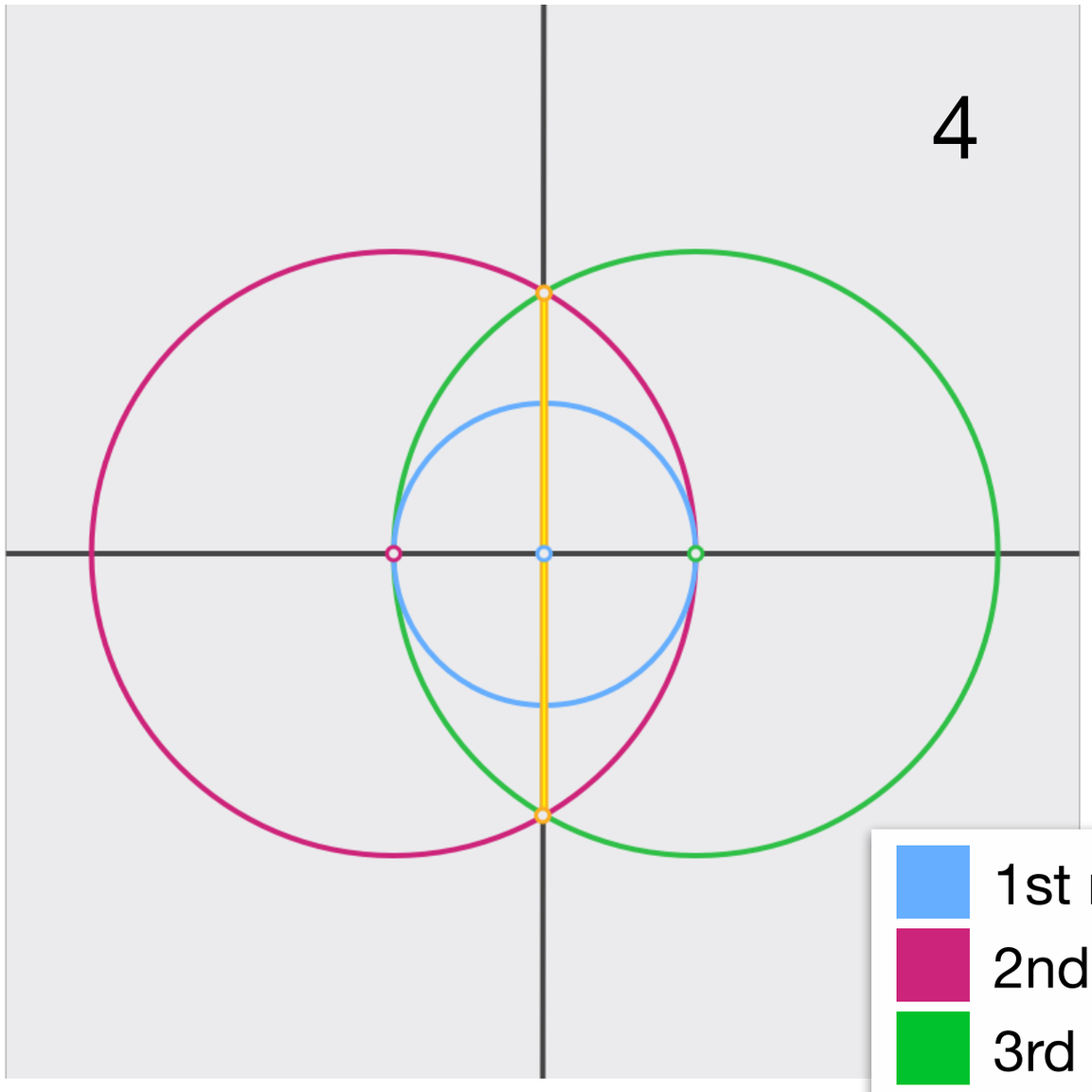


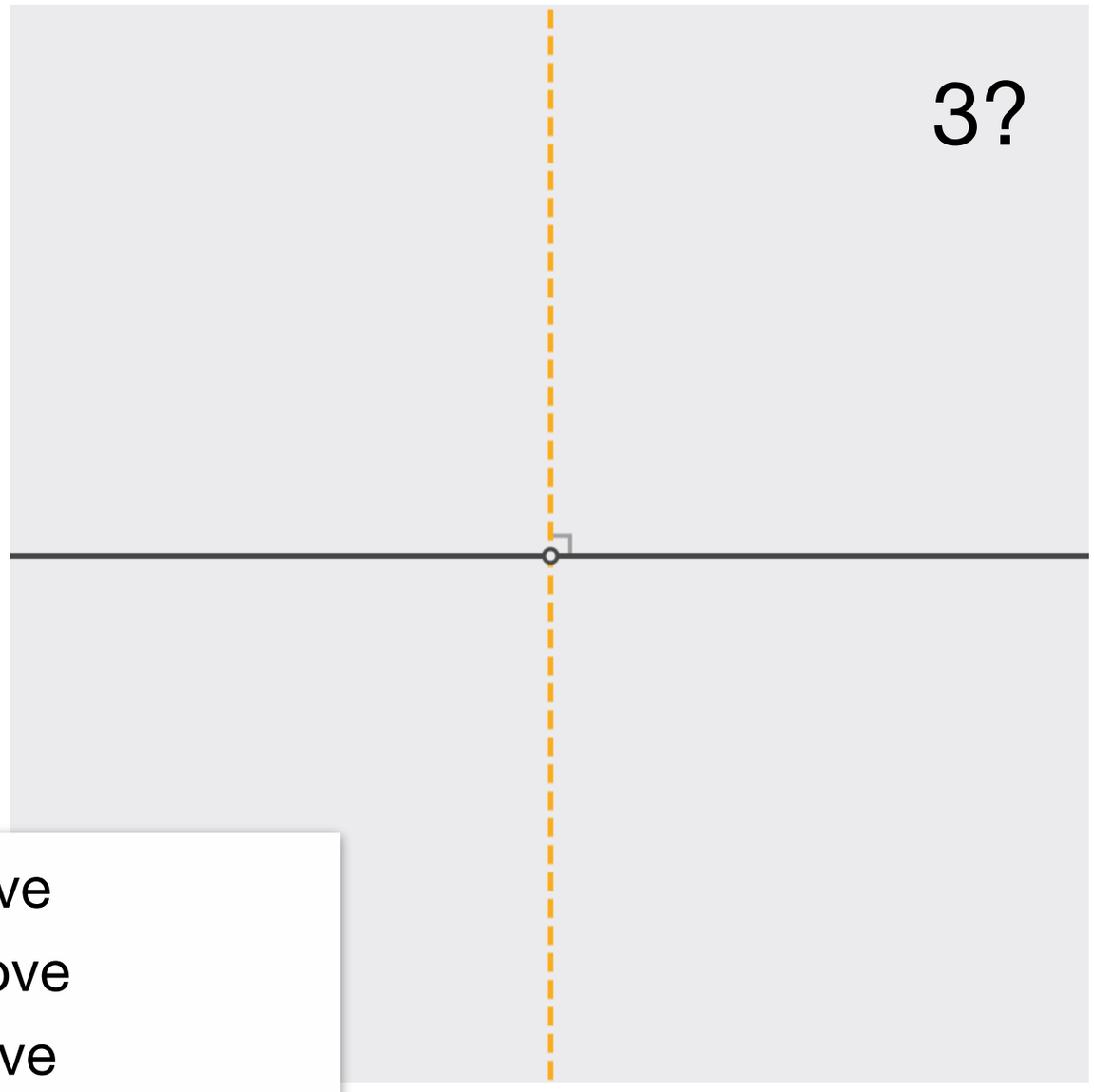


1st move
2nd move
3rd move
answer (last move)

Euclidea, problem 2.7

Erect a perpendicular from the point on the line.

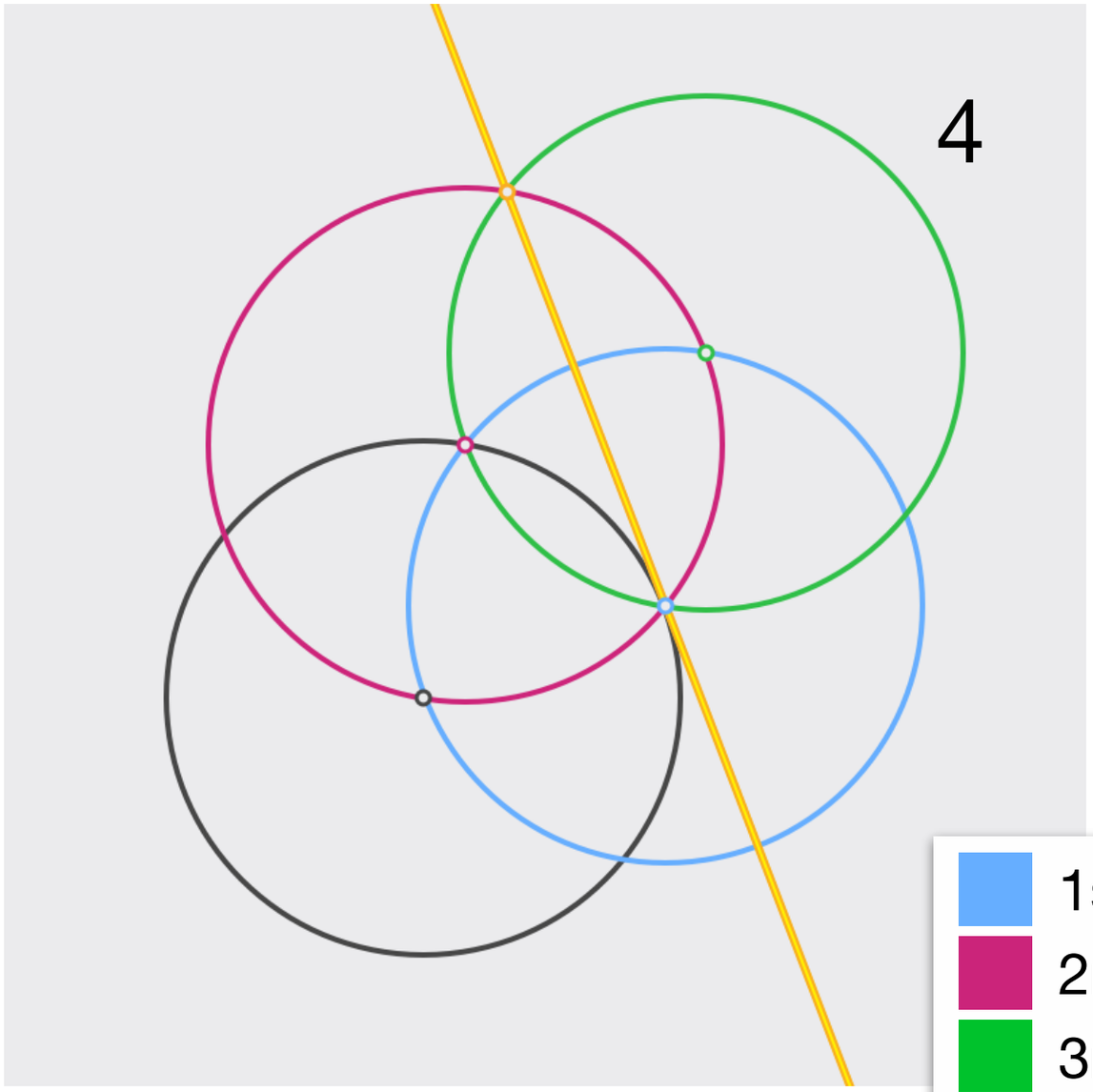


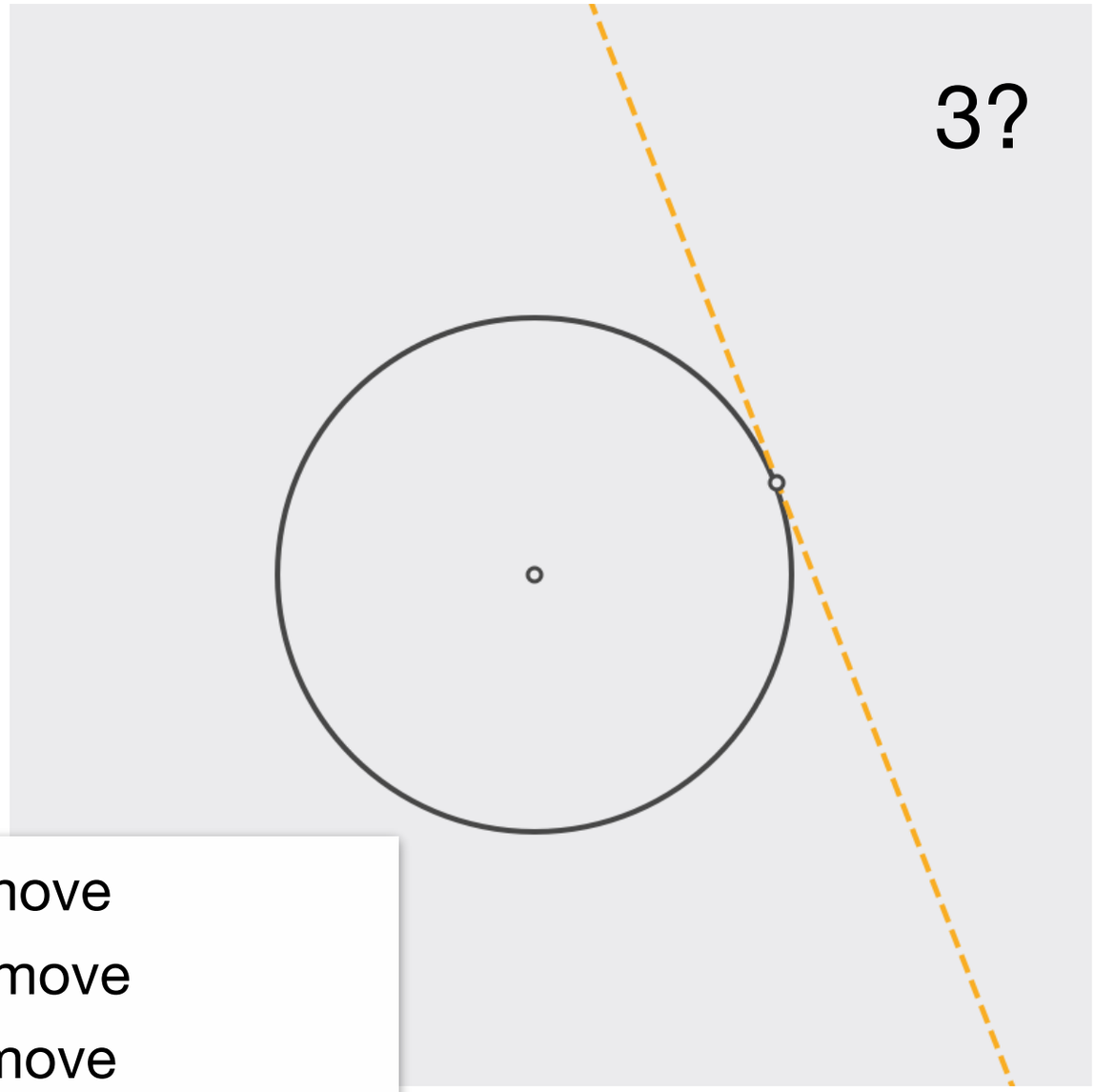


1st move
2nd move
3rd move
answer (last move)

Euclidea, problem 2.8

Construct a tangent to the circle at the given point.





1st move
2nd move
3rd move
answer (last move)

Geometry starts here

Why does it work?

Where is a proof?

Reasoning and proofs develop skills and habits of lifelong importance

- make and investigate conjectures
- justify ideas
- develop or refute arguments
- refine thinking

Our idea

to make proofs accessible for everyone
(like our "serious" games engage people in math)

Our mission

to preserve the spirit of math for the future
and to resist mathematical degradation

APRiL (A PRoof Language)

- close to natural language
- flexible enough to create proofs in Euclidean geometry
- formal enough to satisfy any desired level of rigor
- allowing users to skip boring details if necessary

APRiL (A PRoof Language)

- first-order logic
- no explicit quantifiers
- no sets
- no plane transformations
- ...

APRiL (A PRoof Language)

- high-level notions
- basic theorems and properties as axioms
- graphic user interface

Euc language

Proposition 05
IS A B C ==> EA A B C A C B

Prop05.prf
ISABC
ANTRABC+EEABAC defn:isosceles
TRABC
EEABAC
EEACAB lemma:congruencesymmetric
NCABC defn:triangle
COCAB assumption
** COABC lemma:collinearorder**
NCCAB reductio
EACABBAC lemma:ABCequalsCBA
ANEECBBC+EAACBABC+EAABCACB
proposition:04
EAABCACB

Coq language

```
Proposition_05
forall A B C,
isosceles A B C ->
CongA A B C A C B.

Proof
intros.
assert ((Triangle A B C /\ Cong A B A C)) by (conclude_def isosceles).
assert (Cong A C A B) by (conclude lemma_congruencesymmetric).
assert (nCol A B C) by (conclude_def Triangle).
assert (~ Col C A B).
{
  intro.
  assert (Col A B C) by (forward_using lemma_collinearorder).
  contradict.
}
assert (CongA C A B B A C) by (conclude lemma_ABCequalsCBA).
assert ((Cong C B B C /\ CongA A C B A B C /\ CongA A B C A C B)) by (conclude
proposition_04).
close.
```

APRiL

```
theorem SAS:
[AB] = [A'B'],
[BC] = [B'C'],
Angle(ABC) = Angle(A'B'C')
==>
Triangle(ABC) = Triangle(A'B'C')
end

def IsoscelesTriangle(ABC):
is Triangle(ABC),
[AB] = [BC]
end
```

APRiL

```
theorem Euclid_p5:
is IsoscelesTriangle(ABC)
==>
Angle(BAC) = Angle(BCA)
end

proof Euclid_p5:
abis: let [BD] is AngleBisector for (Triangle(ABC)).
Triangle(ABD) = Triangle(CBD) where {
    st1: [AB] = [BC] due to is IsoscelesTriangle(ABC).
    st2: [BD] = [BD].
    st3: Angle(ABD) = Angle(CBD) by abis.
    Triangle(ABD) = Triangle(CBD) by theorem SAS(st1, st2, st3).
}
Angle(BAD) = Angle(BCD) by property Triangle(ABD) = Triangle(CBD).
Angle(BAC) = Angle(BCA).
end
```

Thank you for your attention

The End