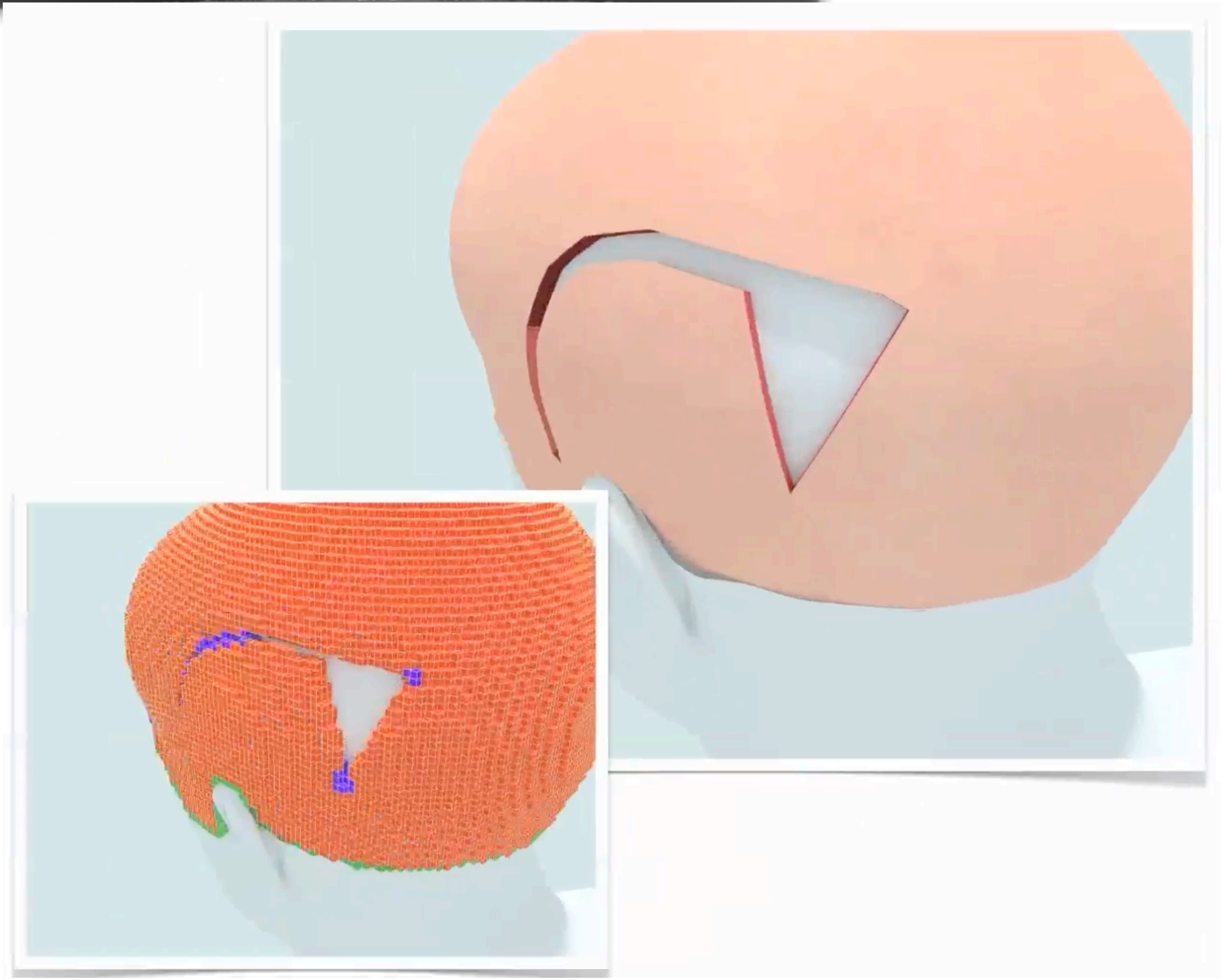
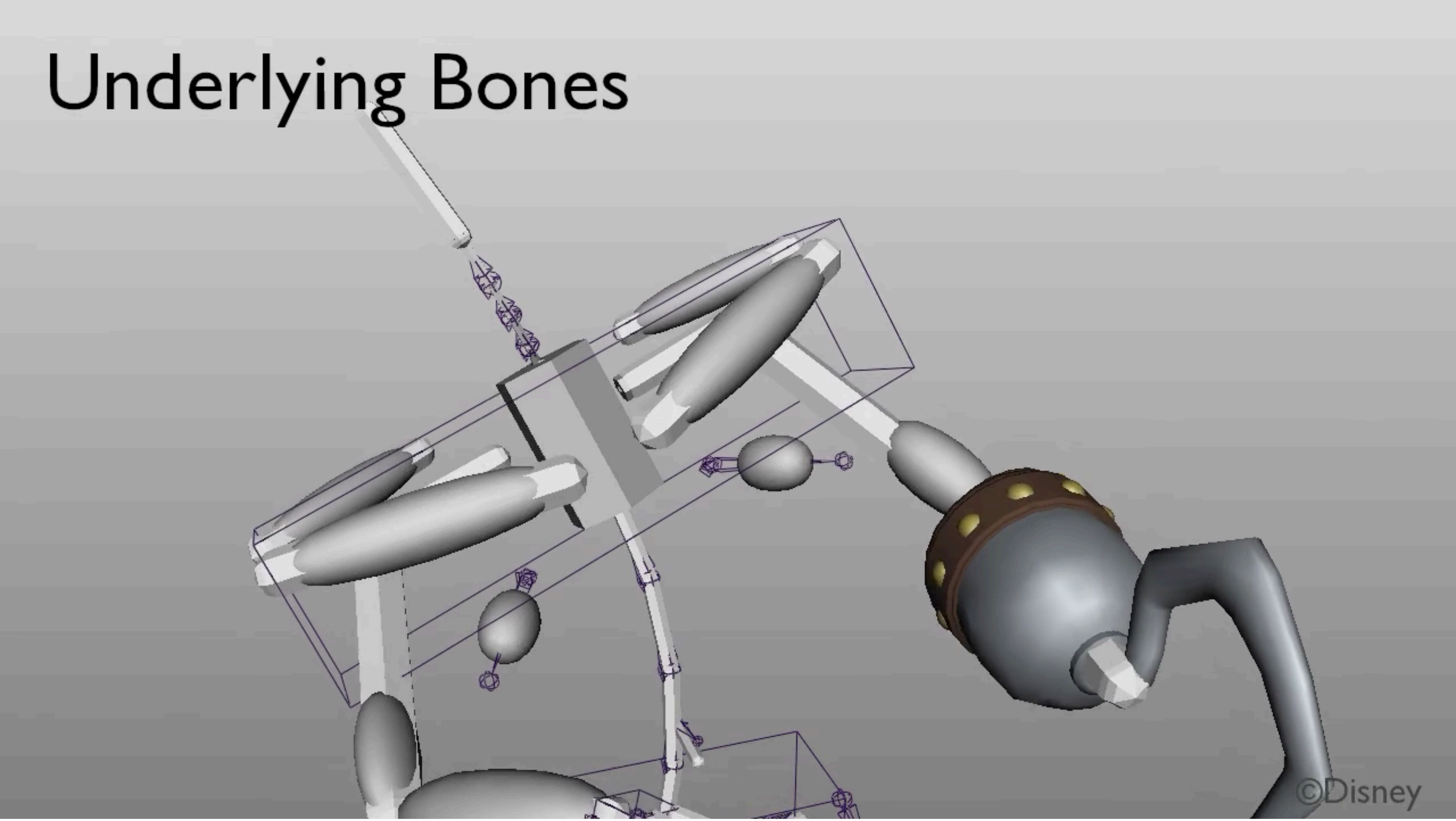
Collision detection (for simulated objects)

- Cannot (easily and efficiently) convert into levelsets to facilitate O(1) collision queries
    - Sometimes we seek collisions between open surfaces, which do not have an "*interior*" to describe as a levelset
- If simulation contains N primitives (particles, segments, triangles, etc) there is a potential for O(N^2) "*candidate*" intersection pairs
    - Brute force check would require O(N^2) cost
    - Every simulation step ideally requires O(N) effort
      (e.g. with Forward Euler, or BE with fixed CG iterations)
    - Ideally the detection cost should not exceed O(N) by much
- Popular approach : Using axis-aligned bounding box (AABB) queries to accelerate collision detection

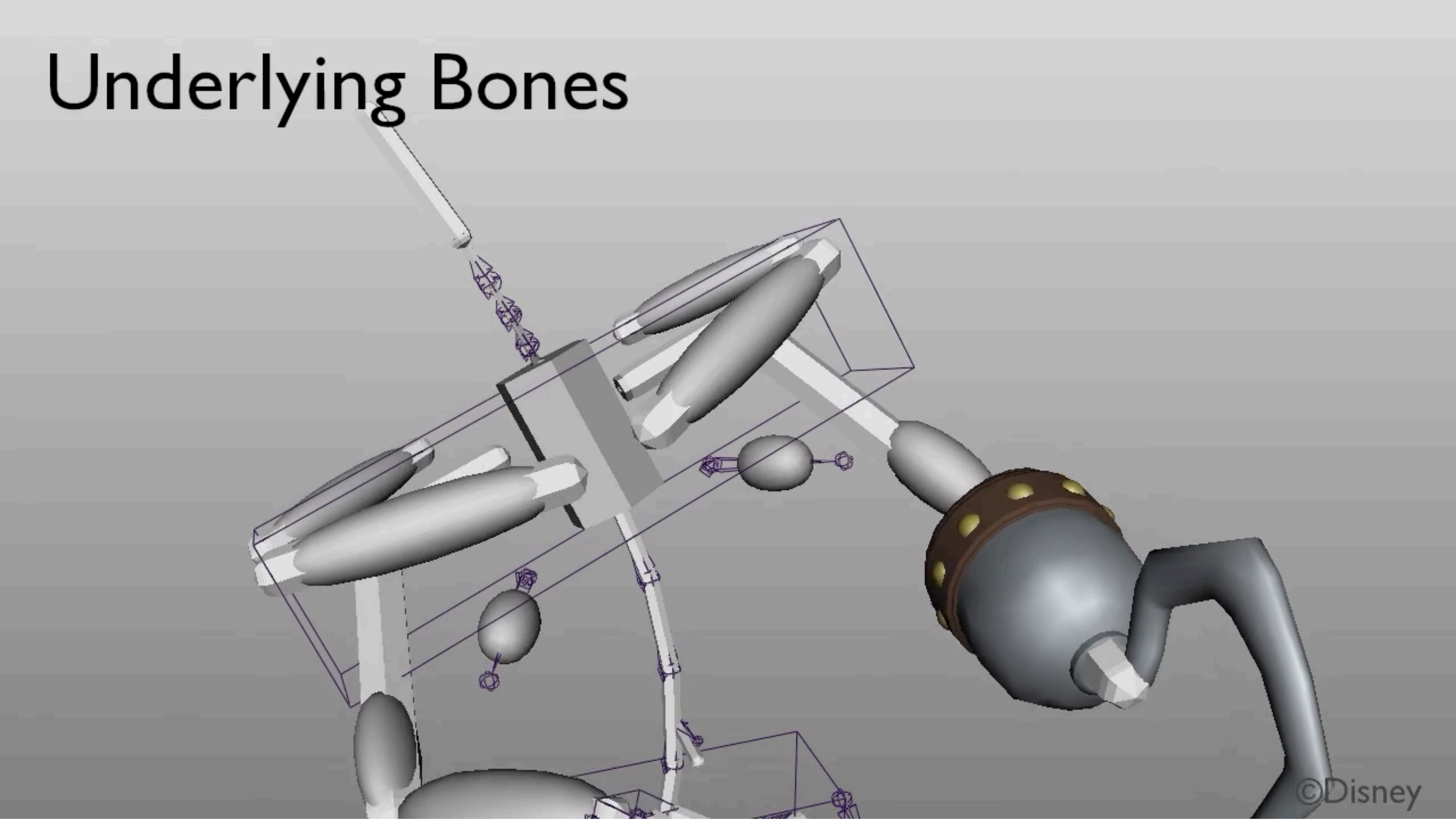
Embedded collisions (w/penalty forces)



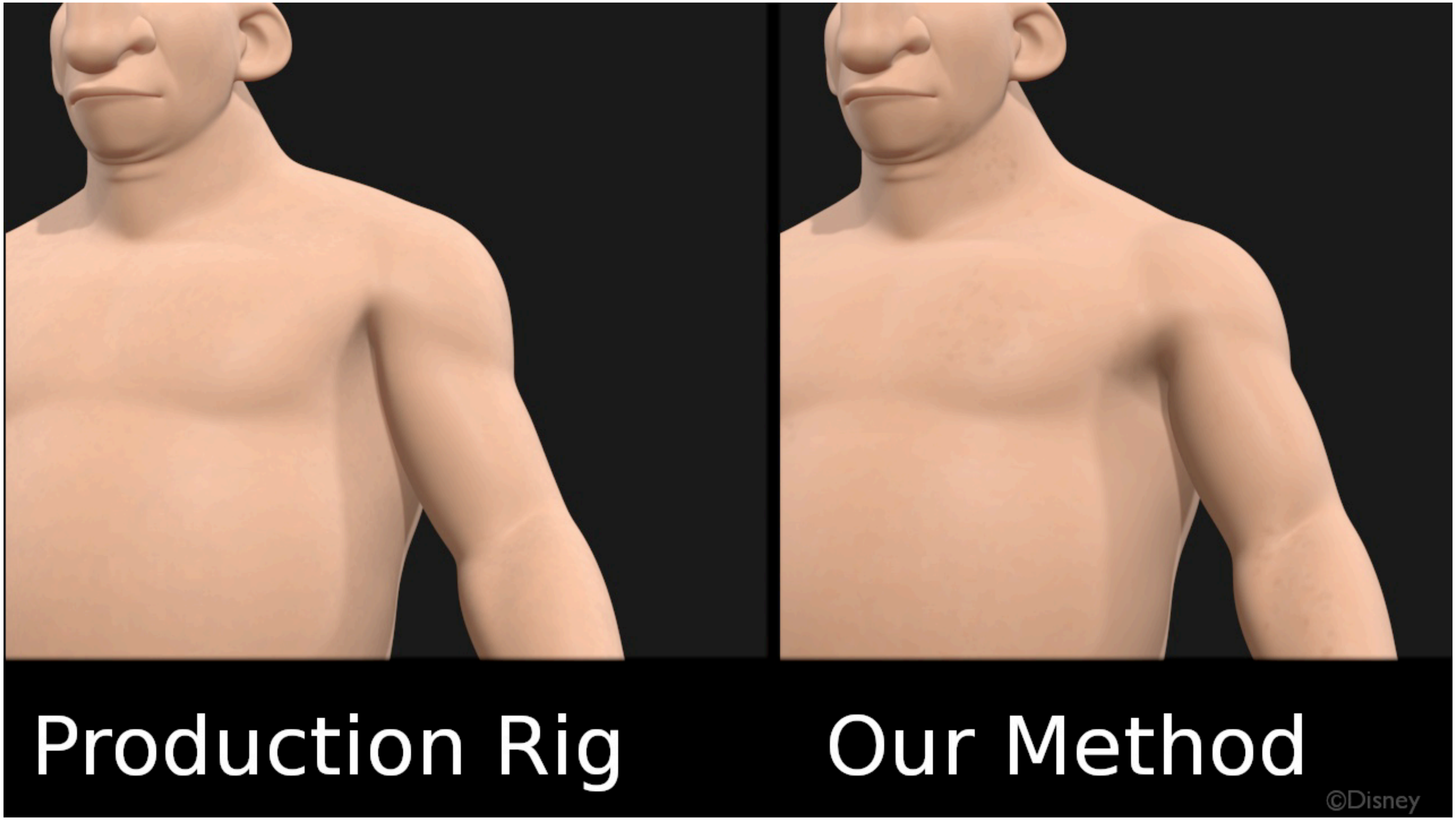
Embedded collisions (w/penalty forces)



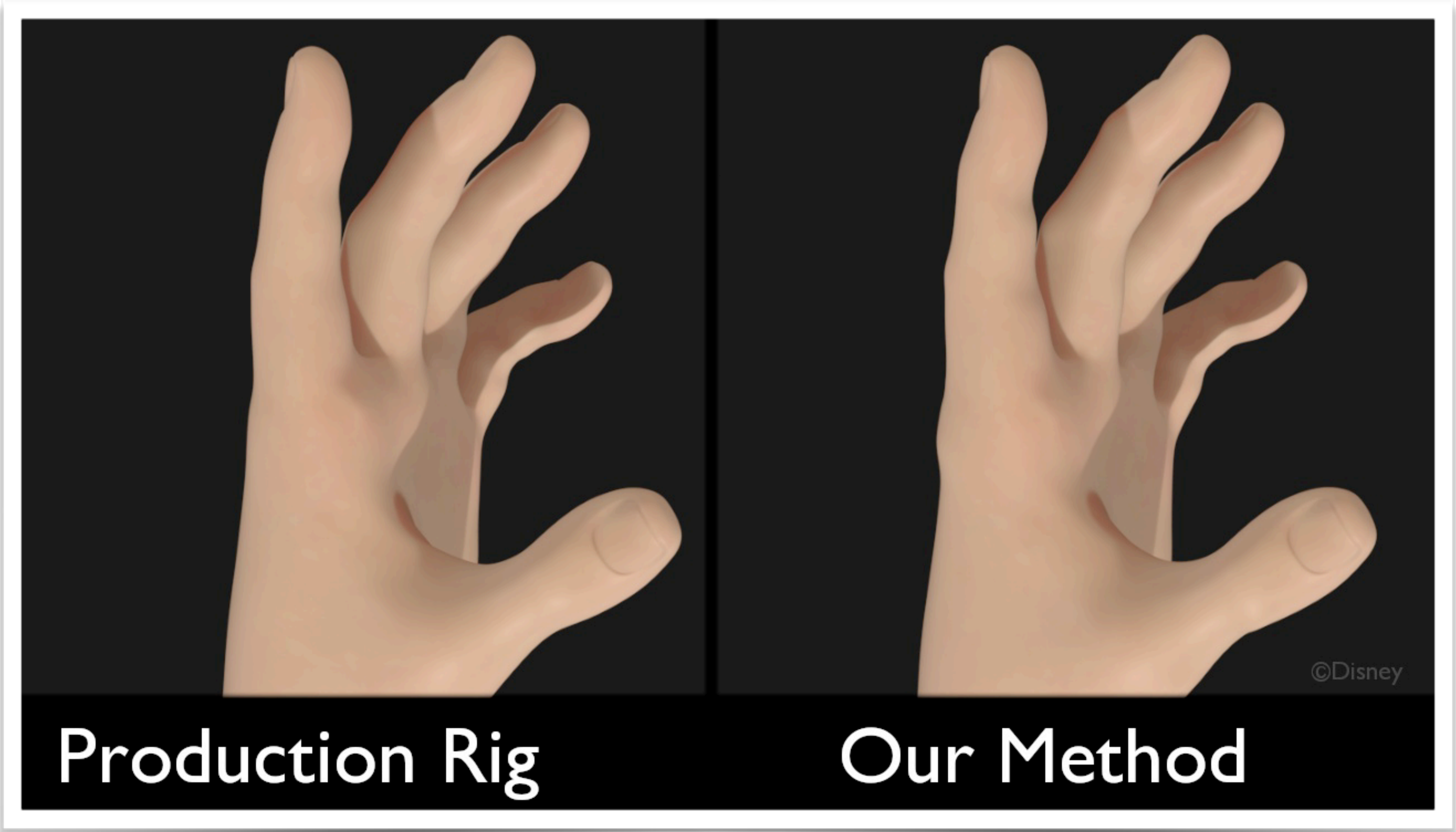
Embedded collisions (w/penalty forces)

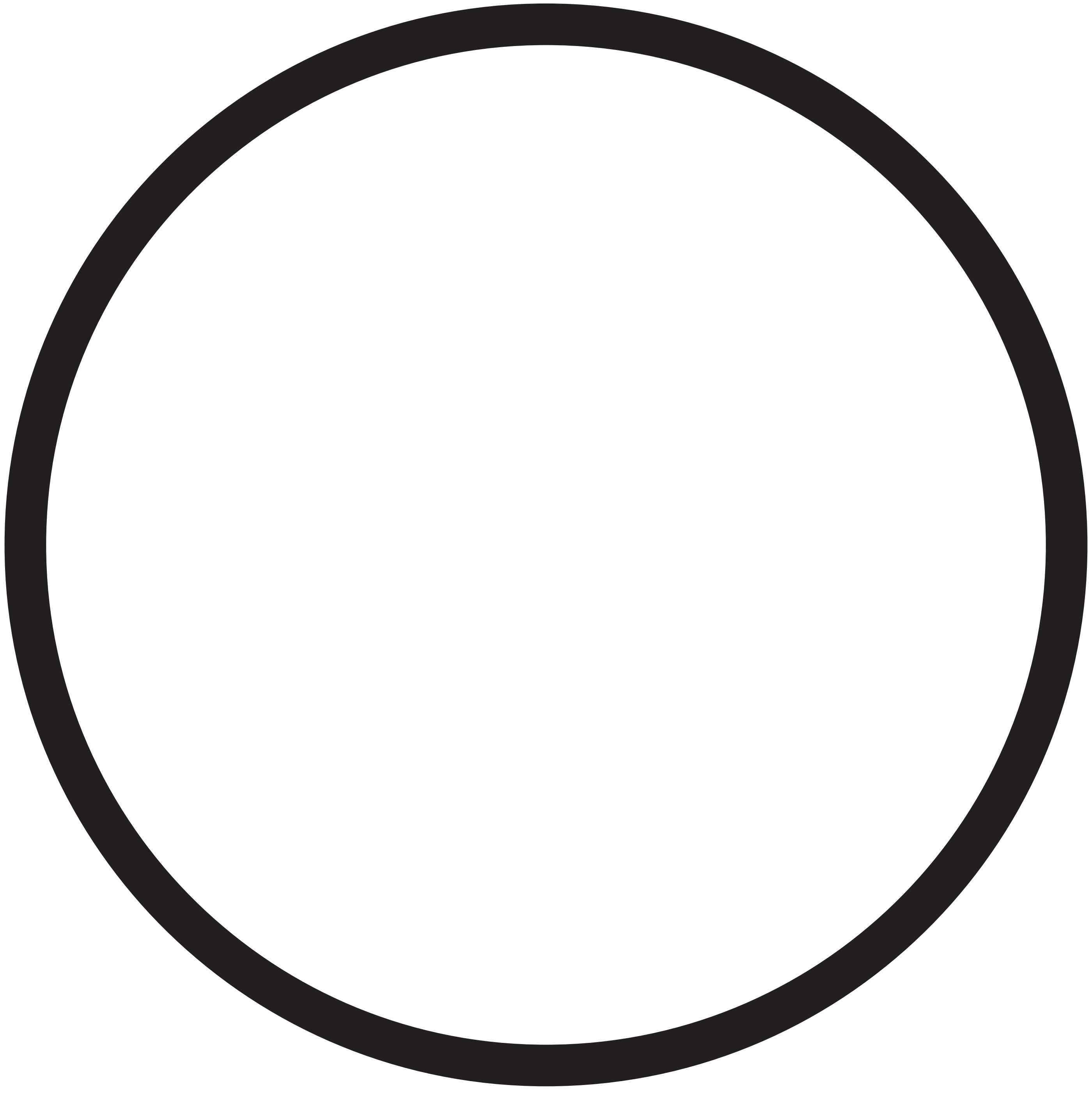


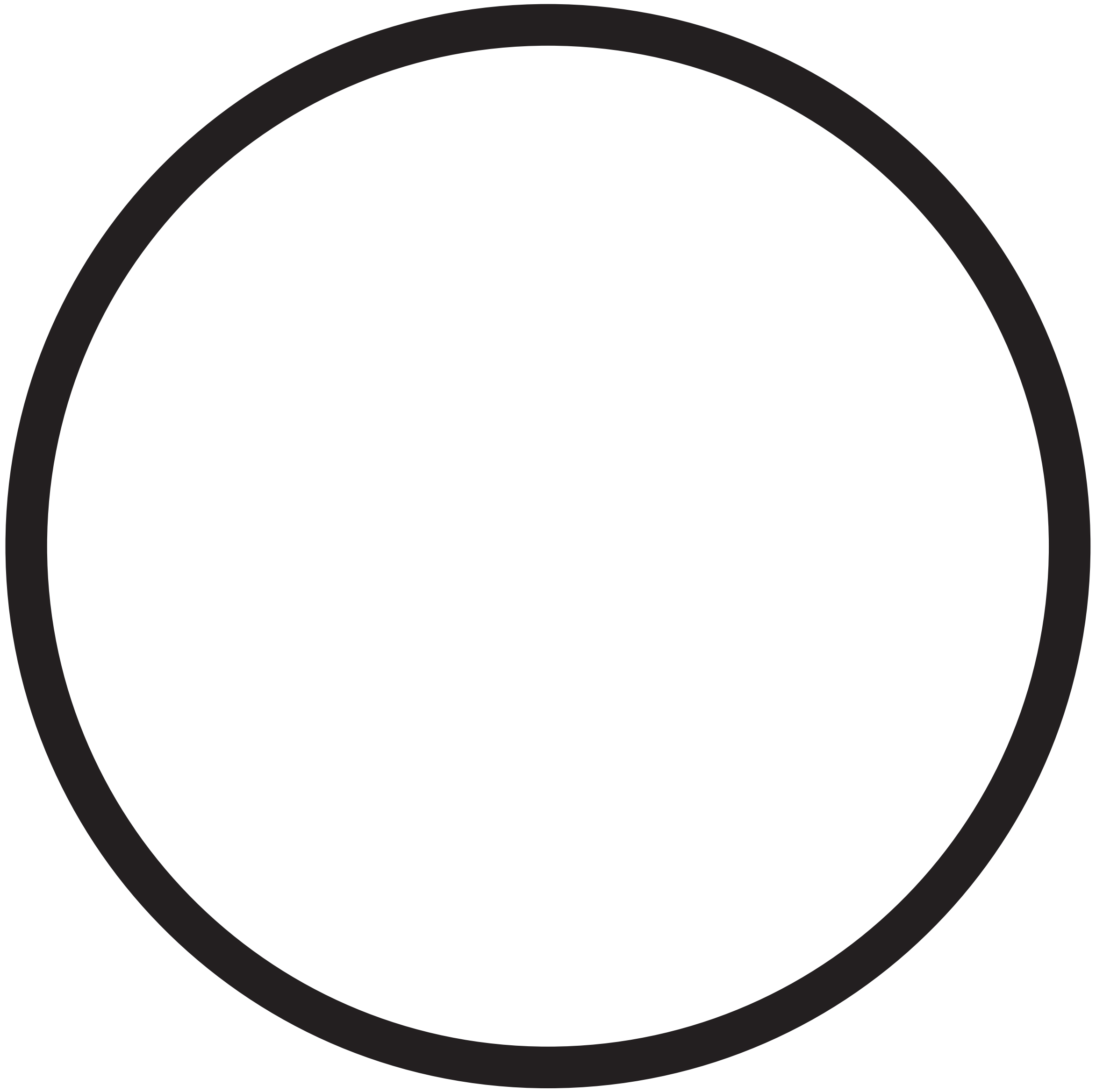
Self collisions : Can we use level sets?



Self collisions : Can we use level sets?





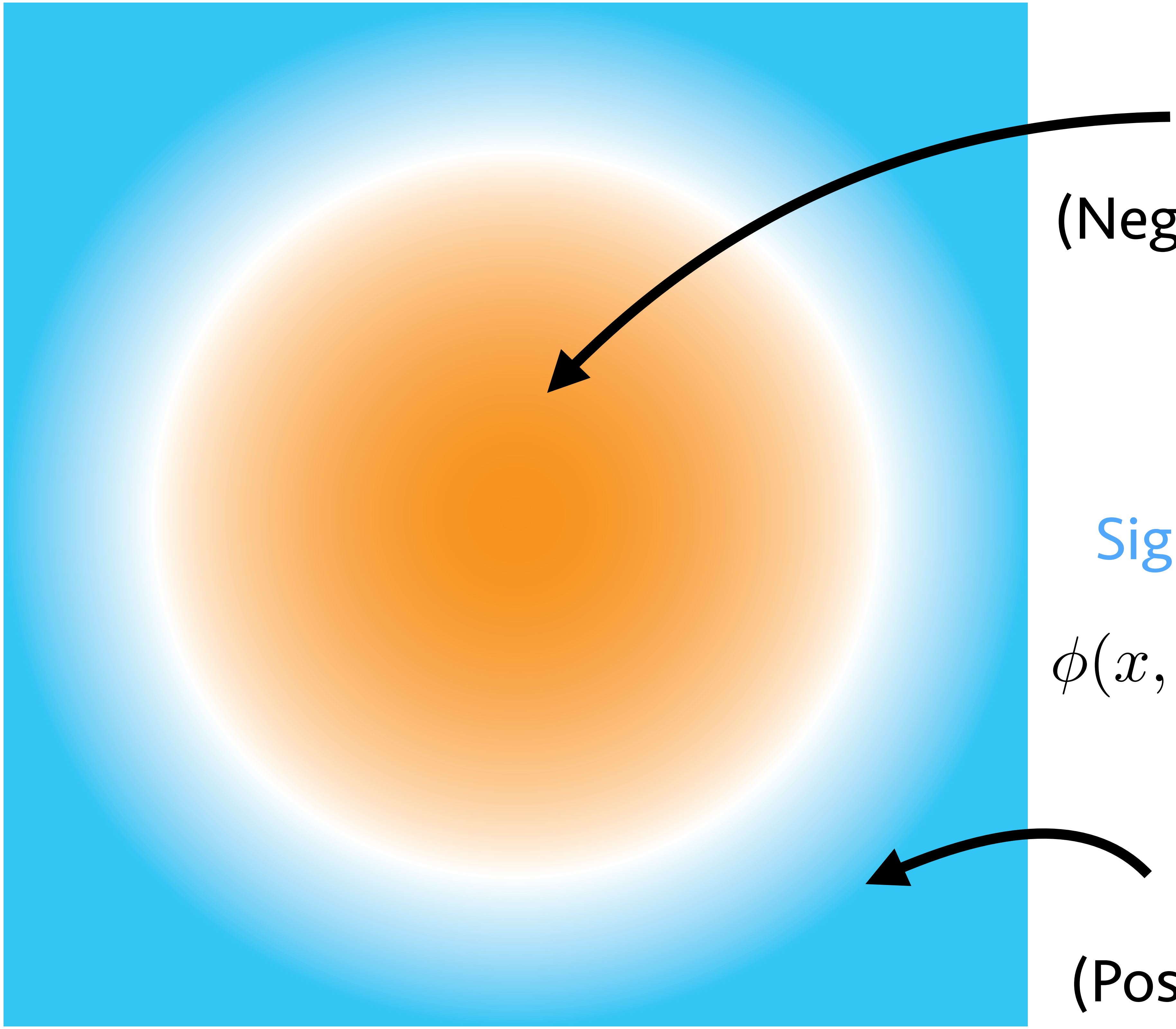


Implicit Surface

$$\mathcal{C} = \{(x,y)|\phi(x,y) = 0\}$$

Signed Distance Field

$$\phi(x,y) = \sqrt{x^2 + y^2} - 1$$



Inside

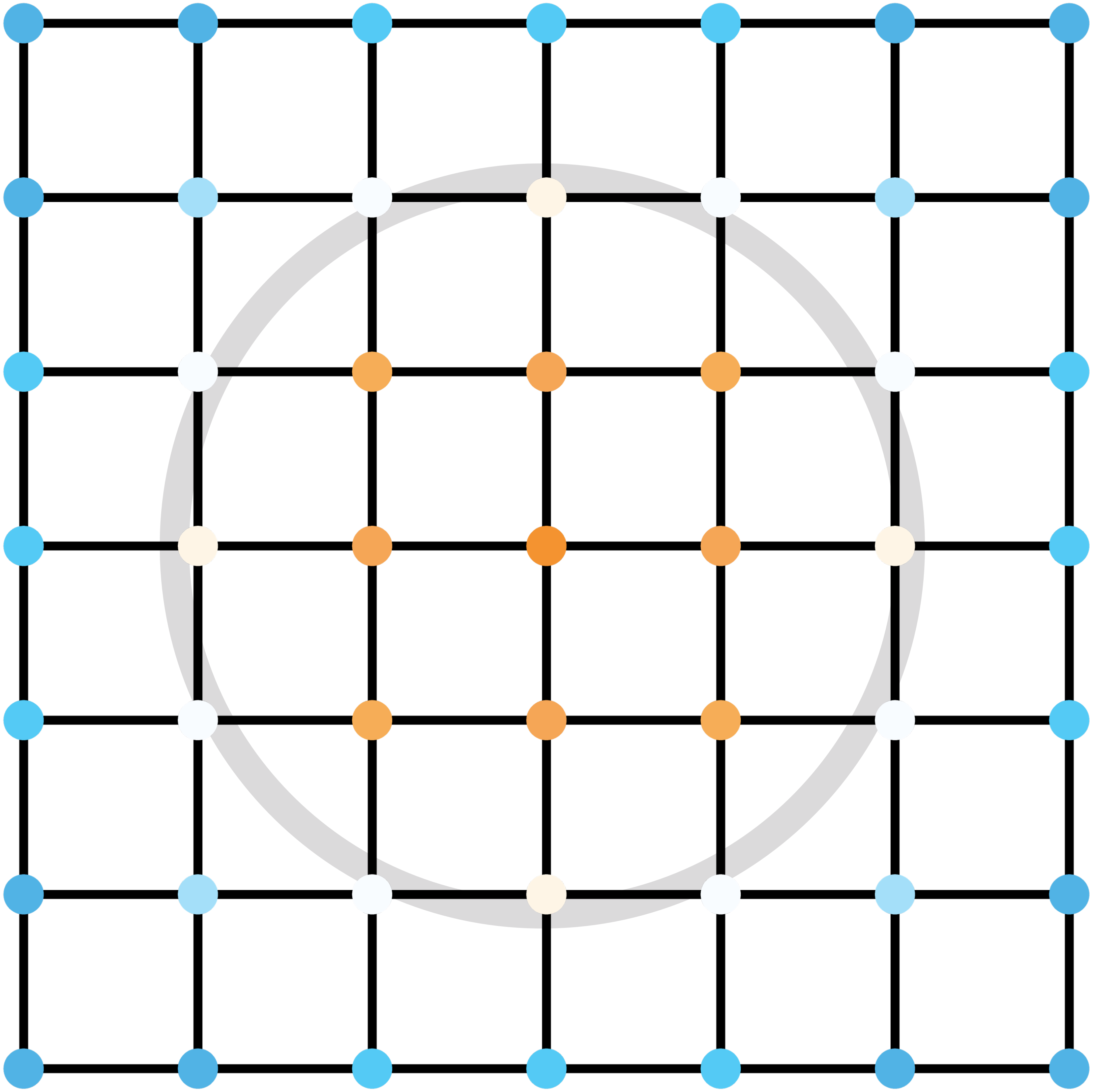
(Negative Φ Values)

Signed Distance Field

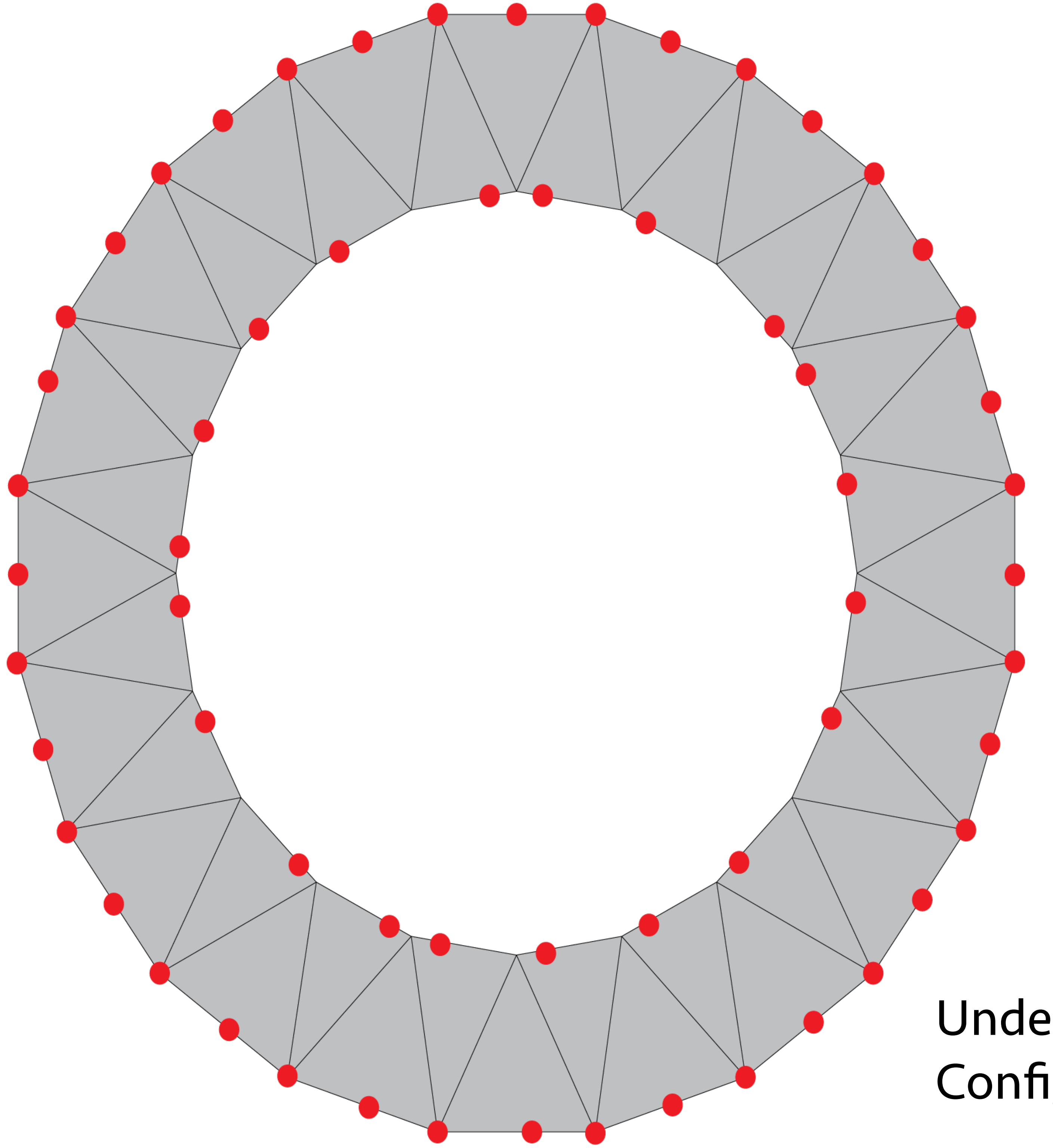
$$\phi(x, y) = \sqrt{x^2 + y^2} - 1$$

Outside

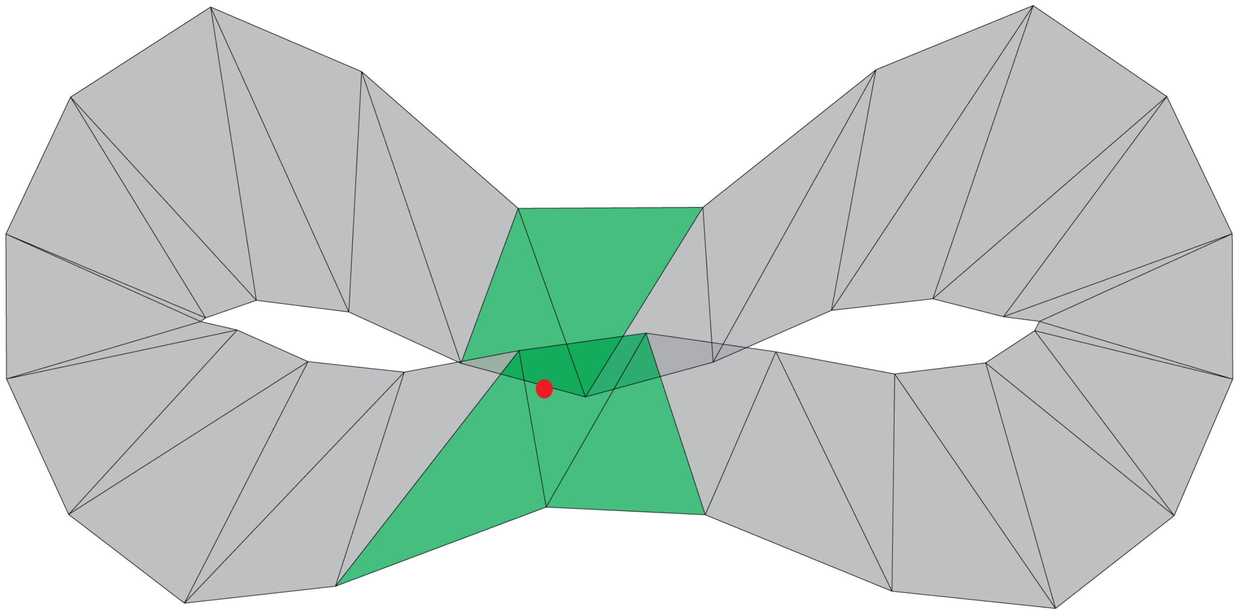
(Positive Φ Values)



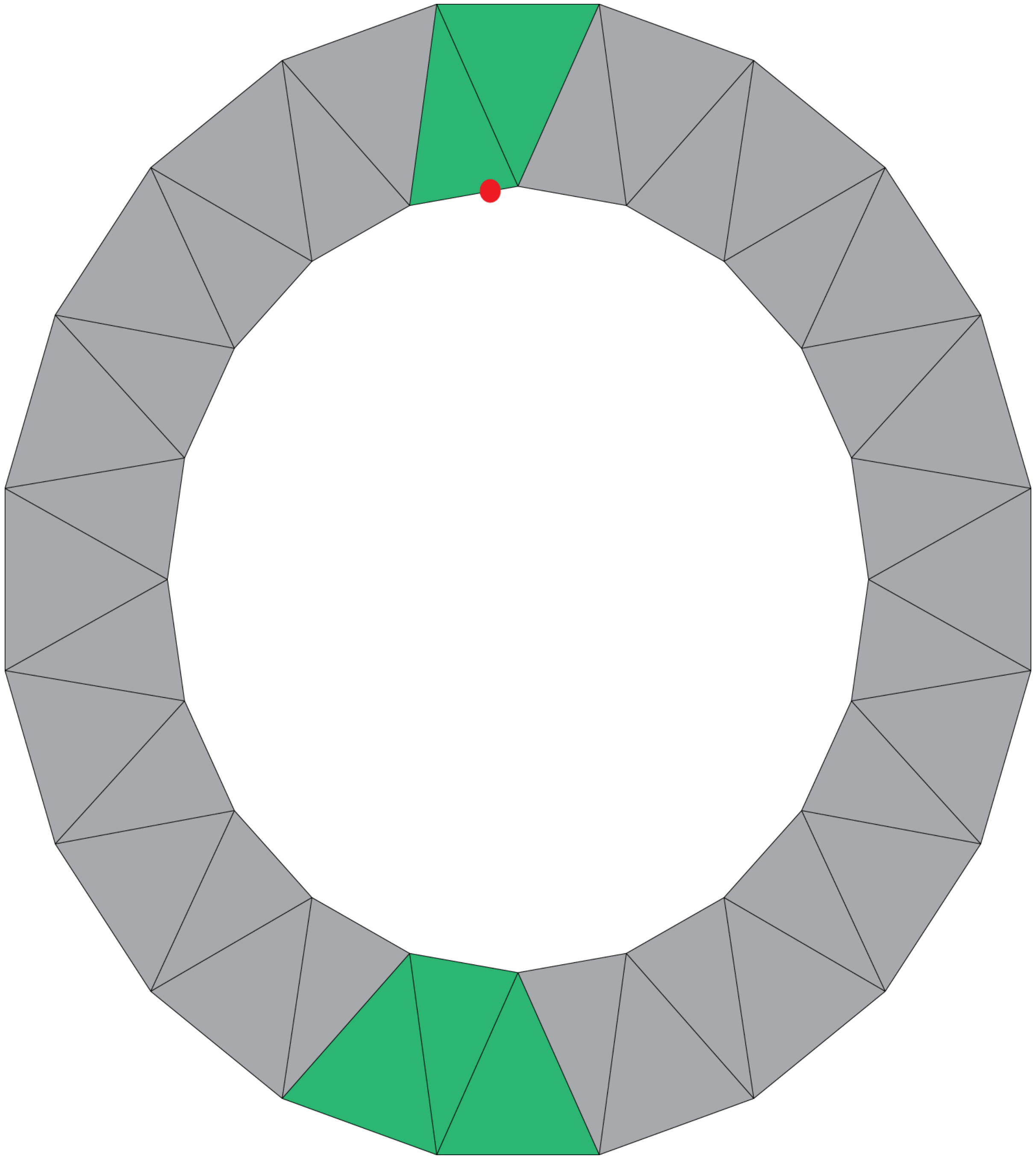
Discrete
Signed Distance Field



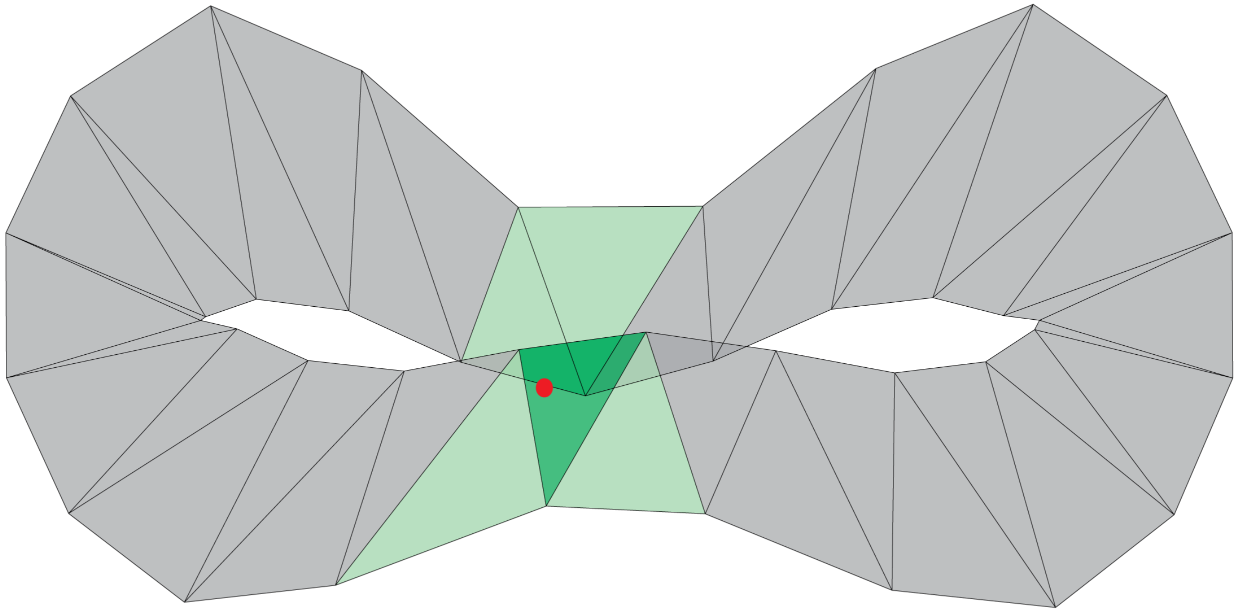
Undeformed
Configuration



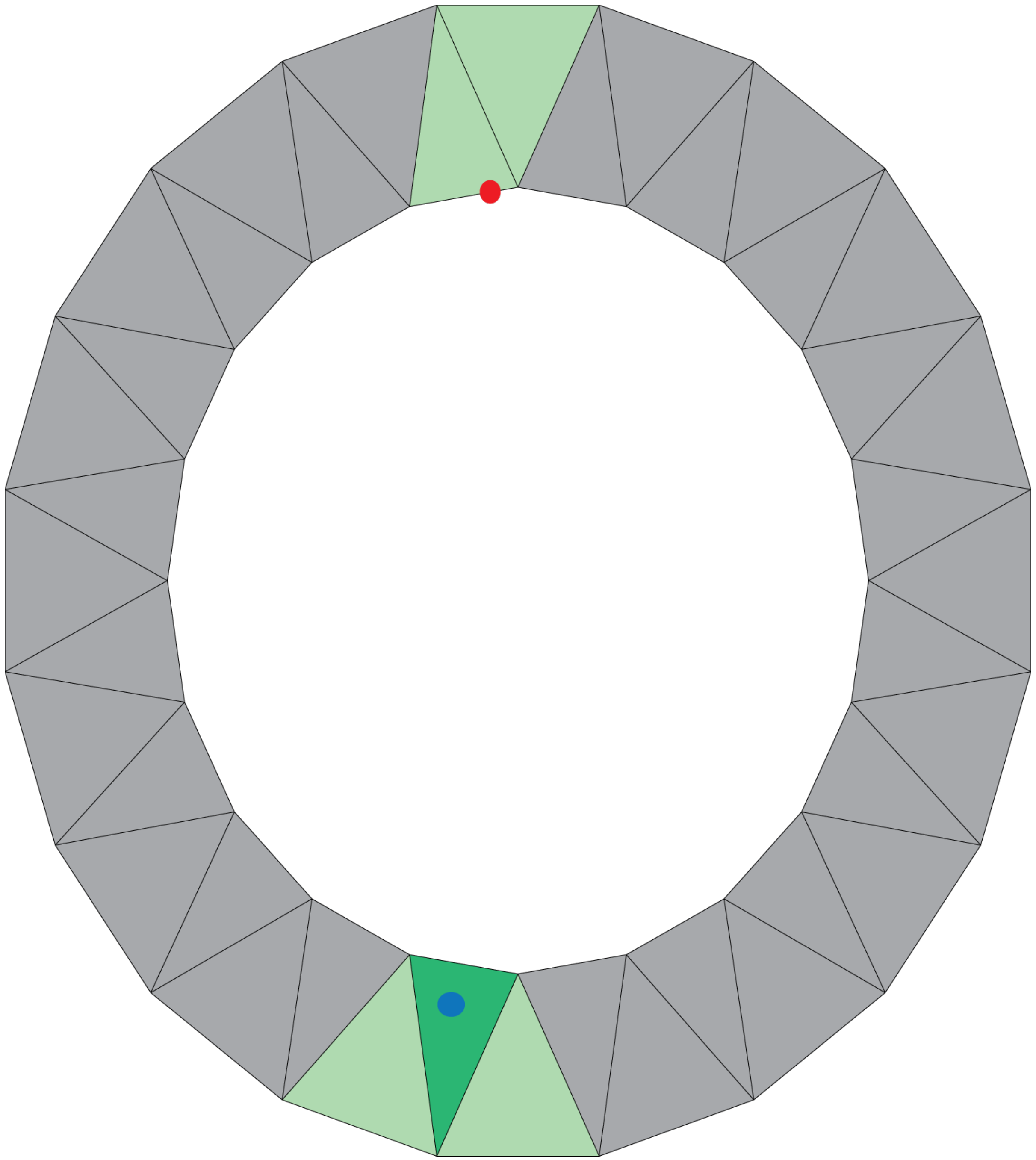
Deformed Configuration



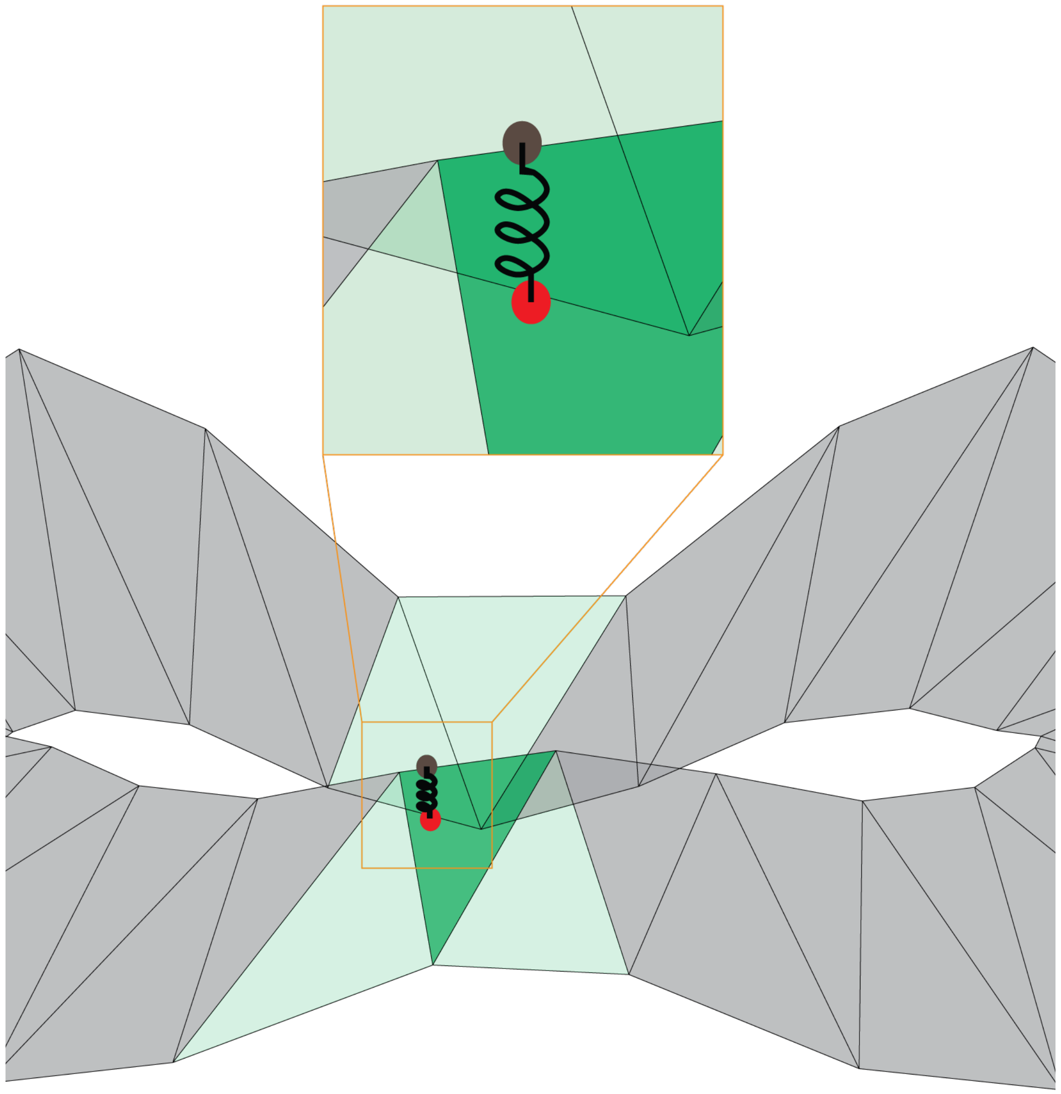
Undeformed Configuration



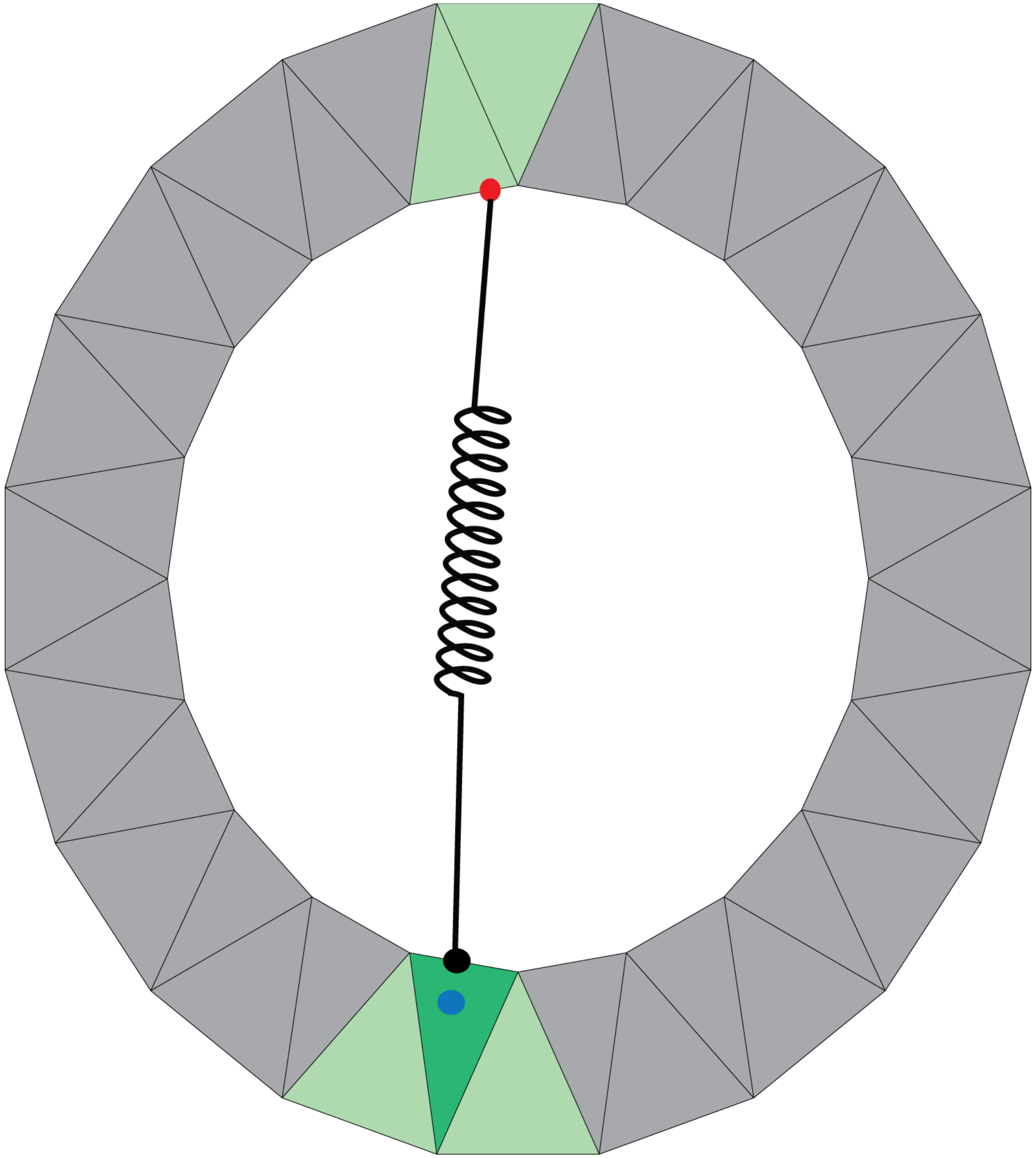
Deformed
Configuration



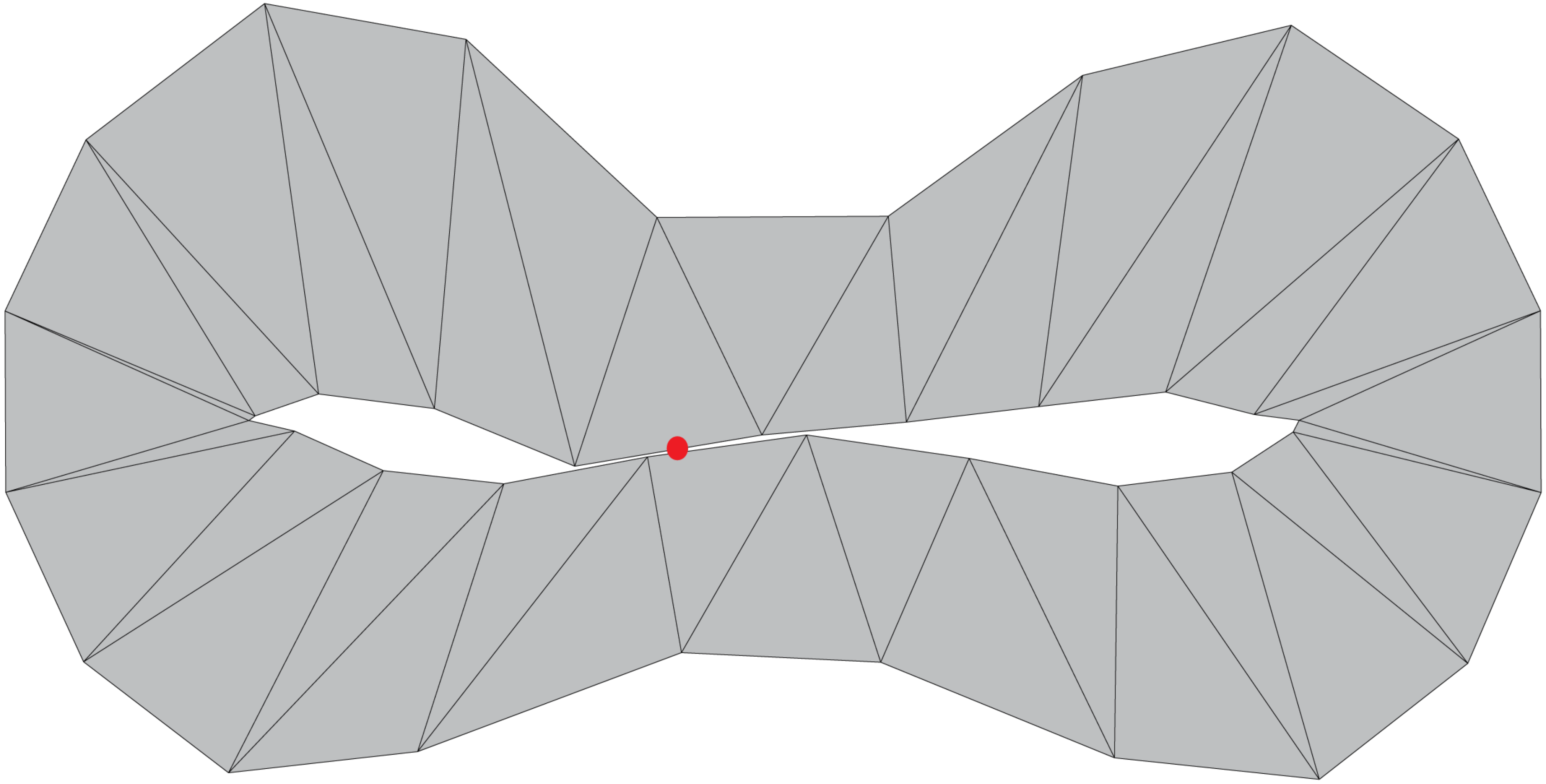
Undeformed
Configuration



Deformed Configuration

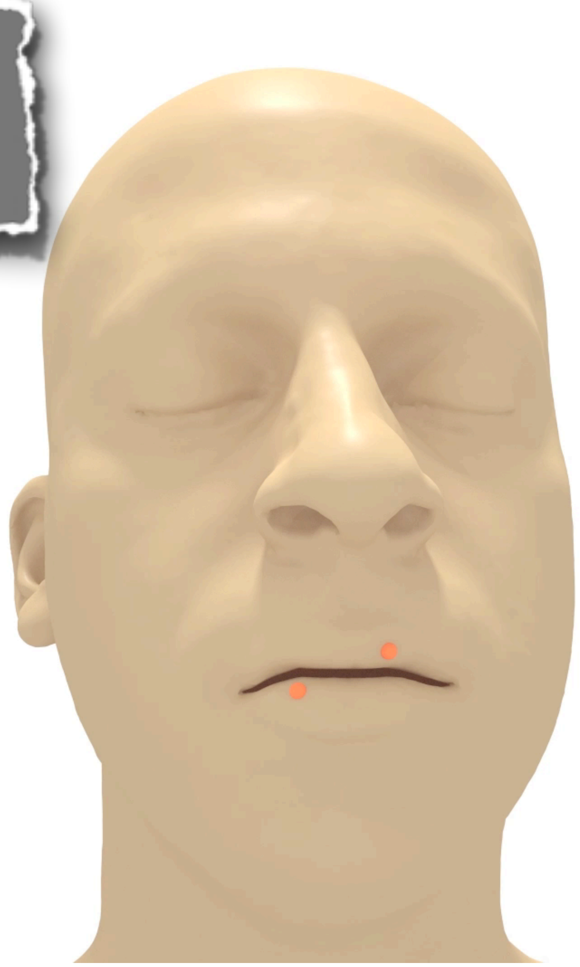


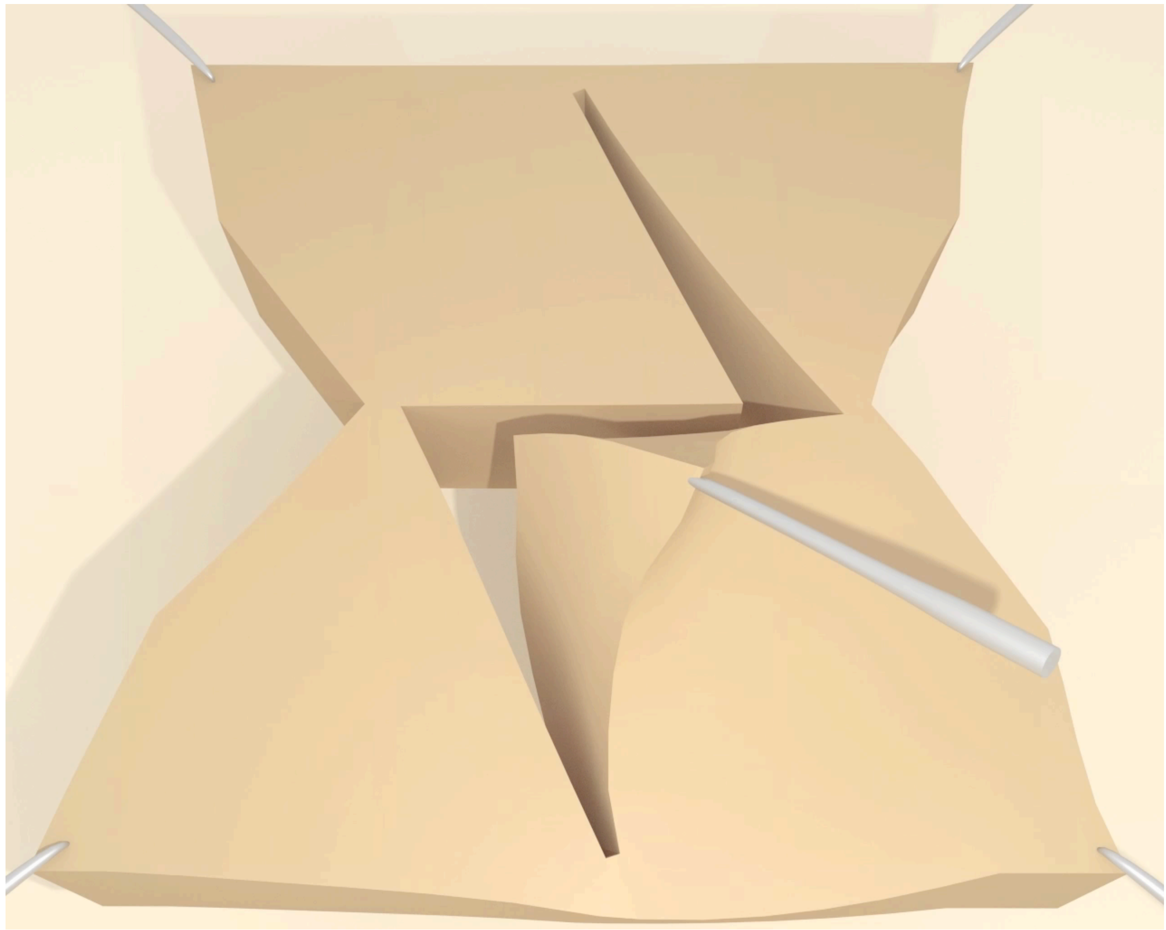
Undeformed Configuration

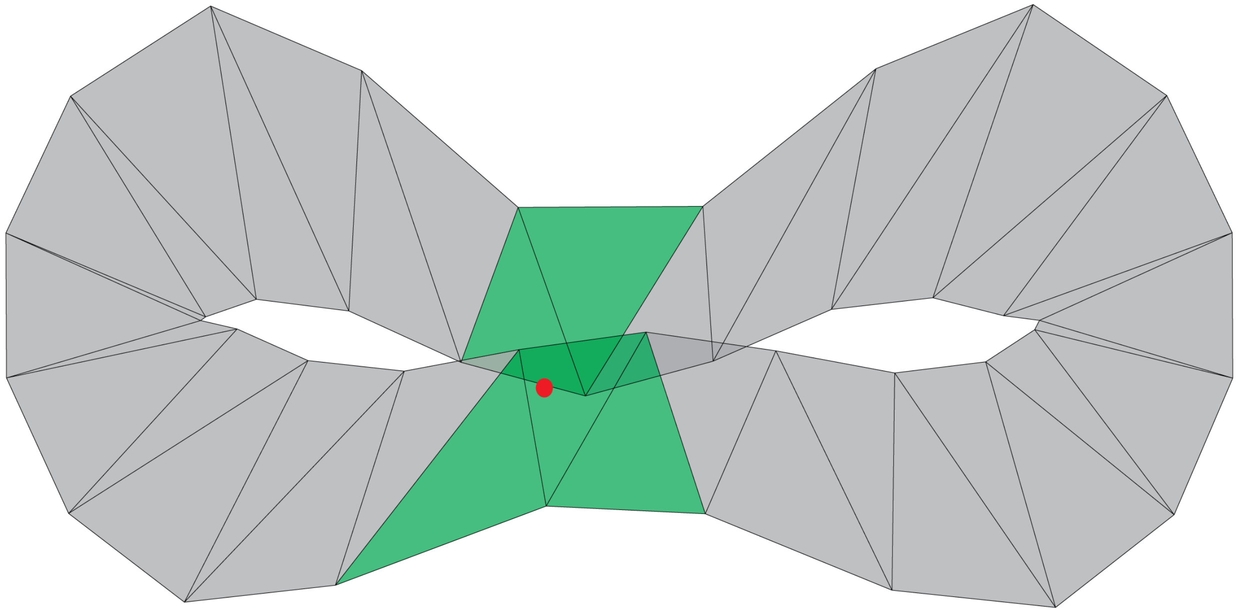


Deformed
Configuration

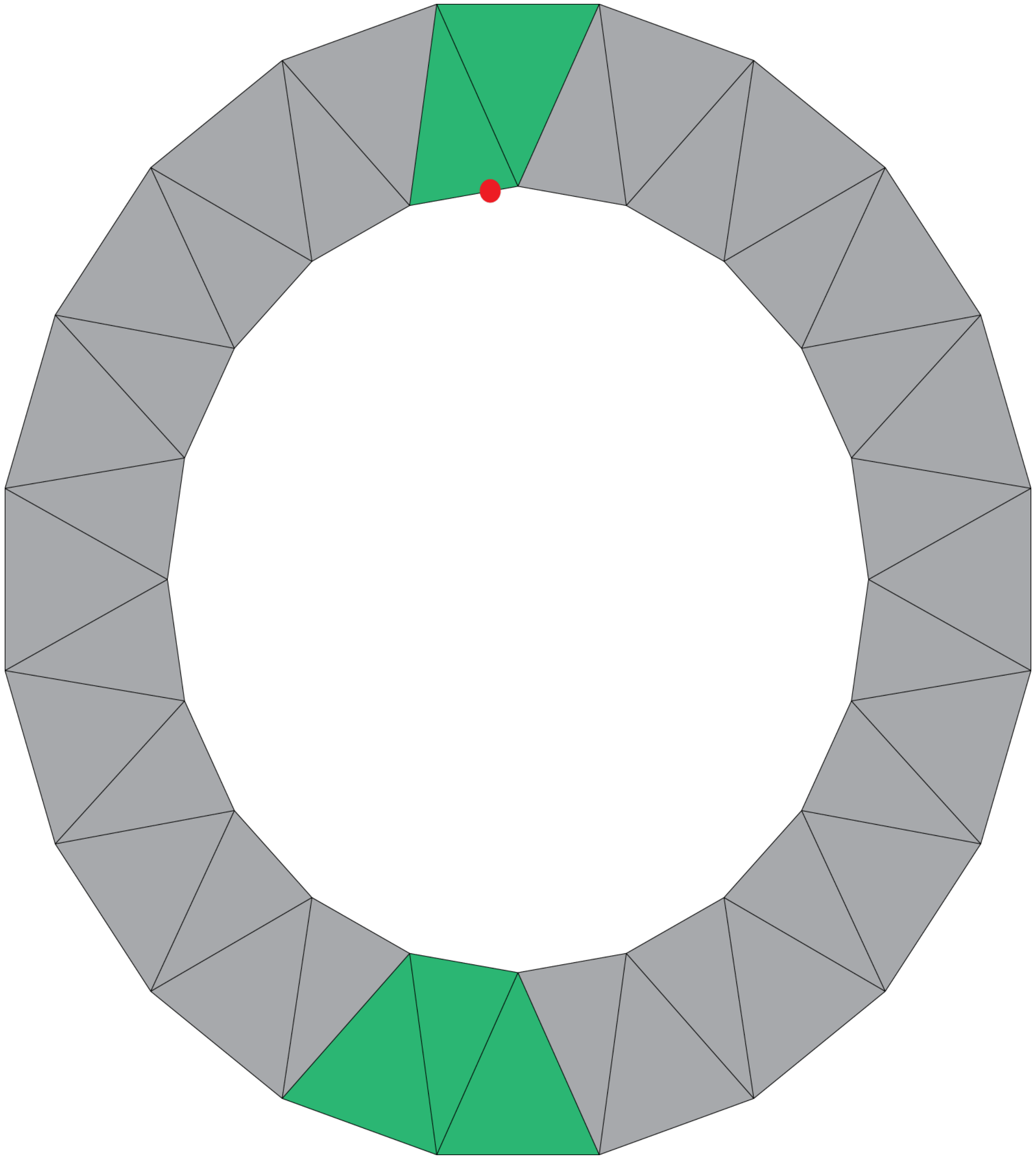
Problem cases



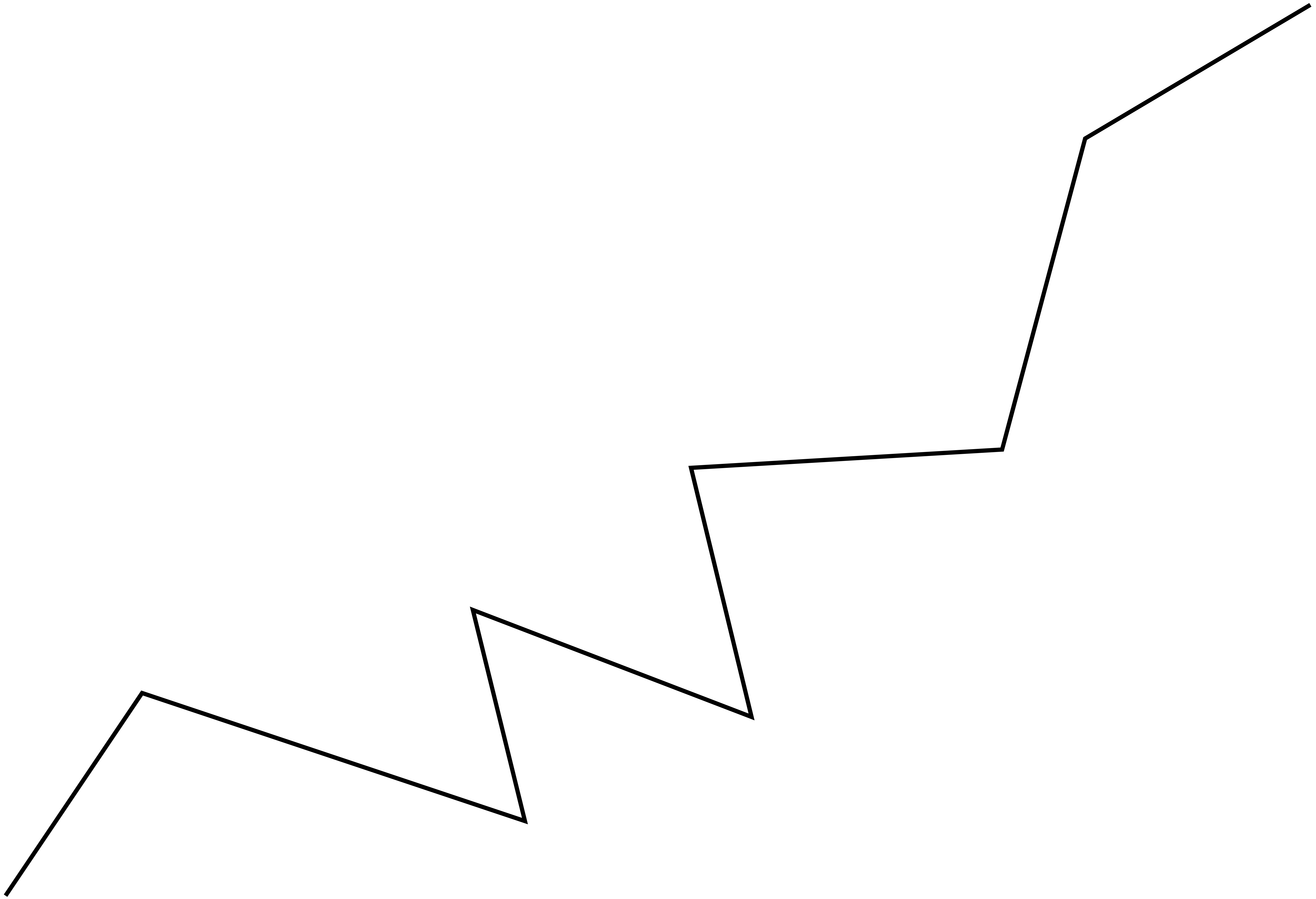


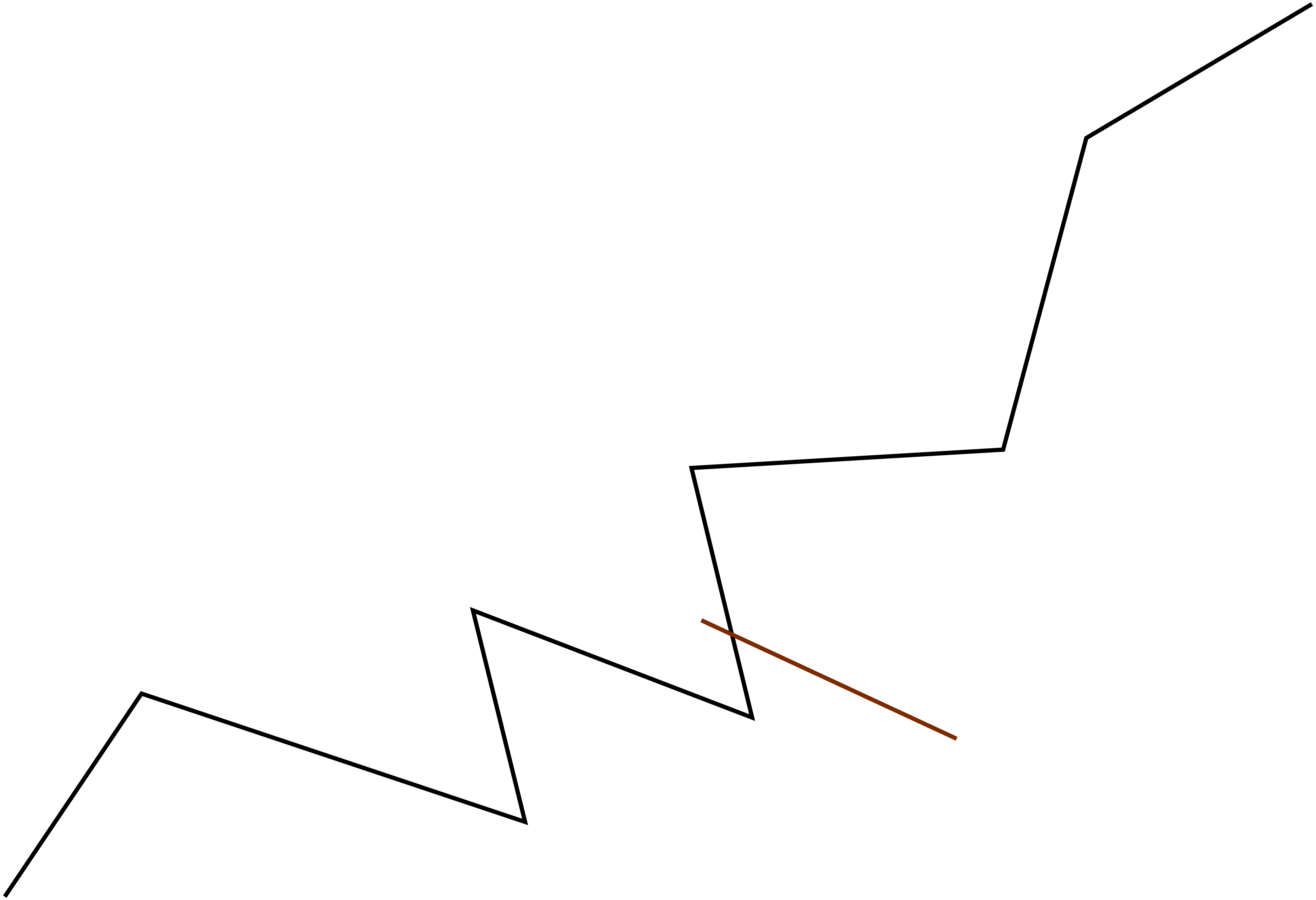


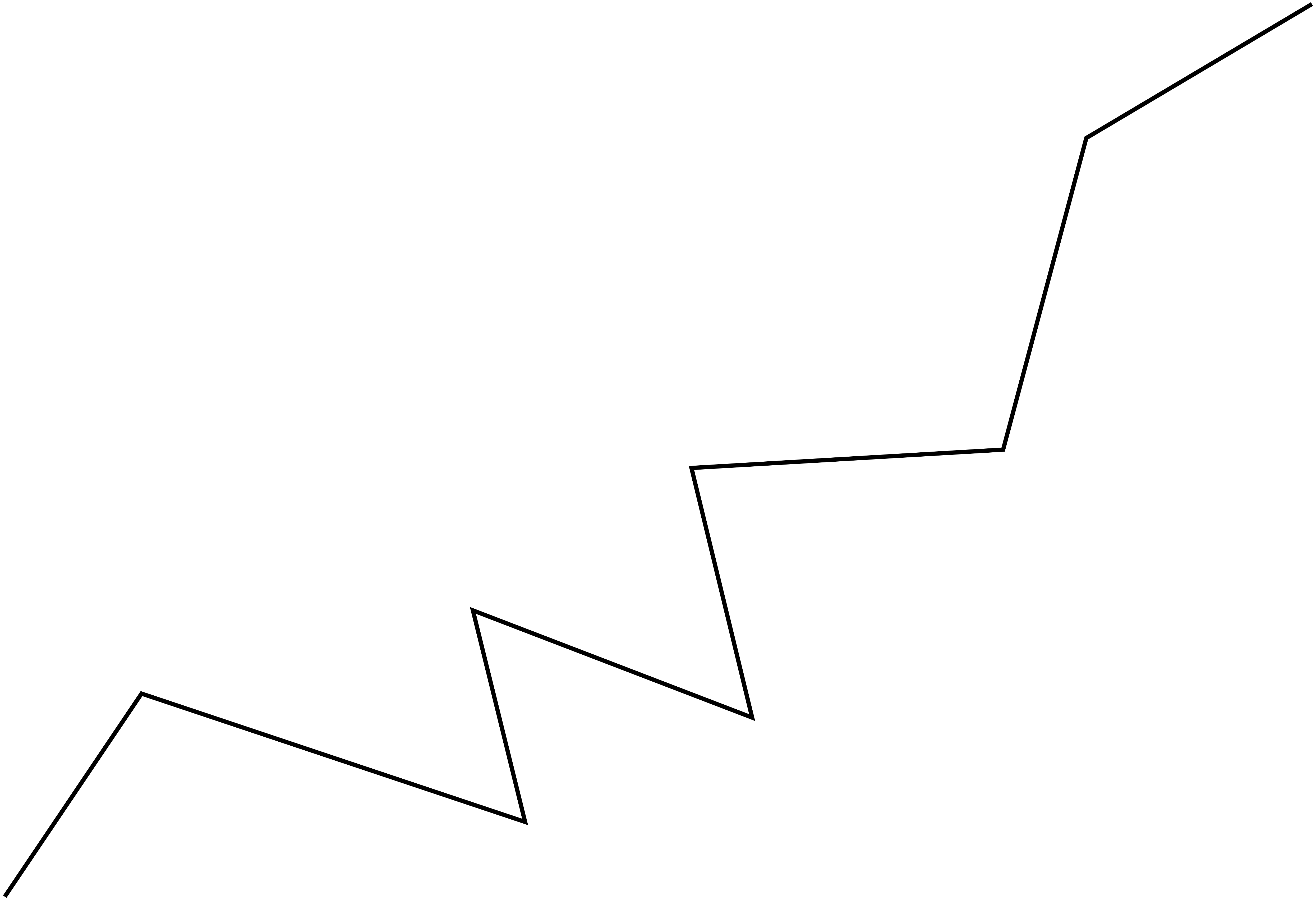
Deformed
Configuration

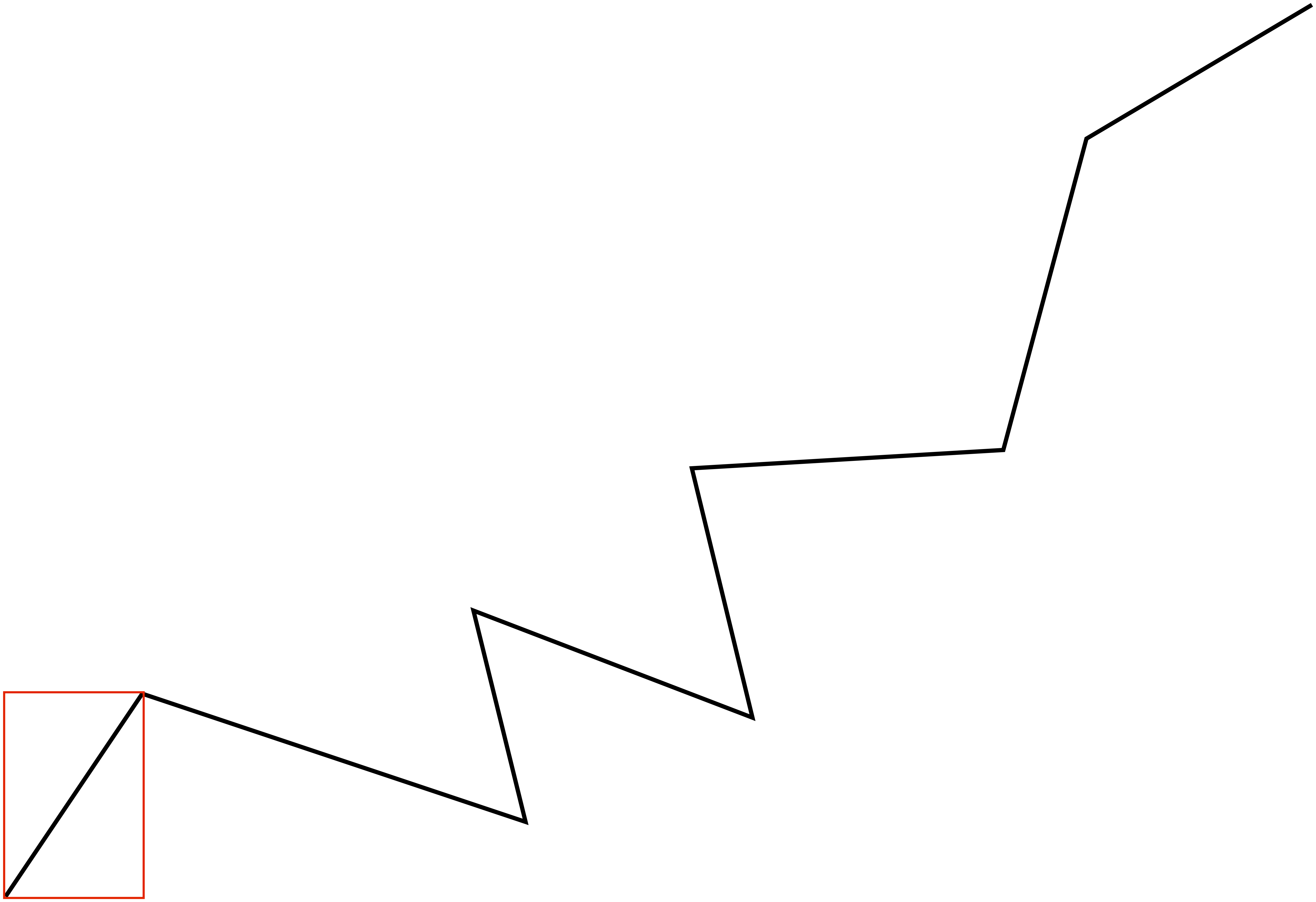


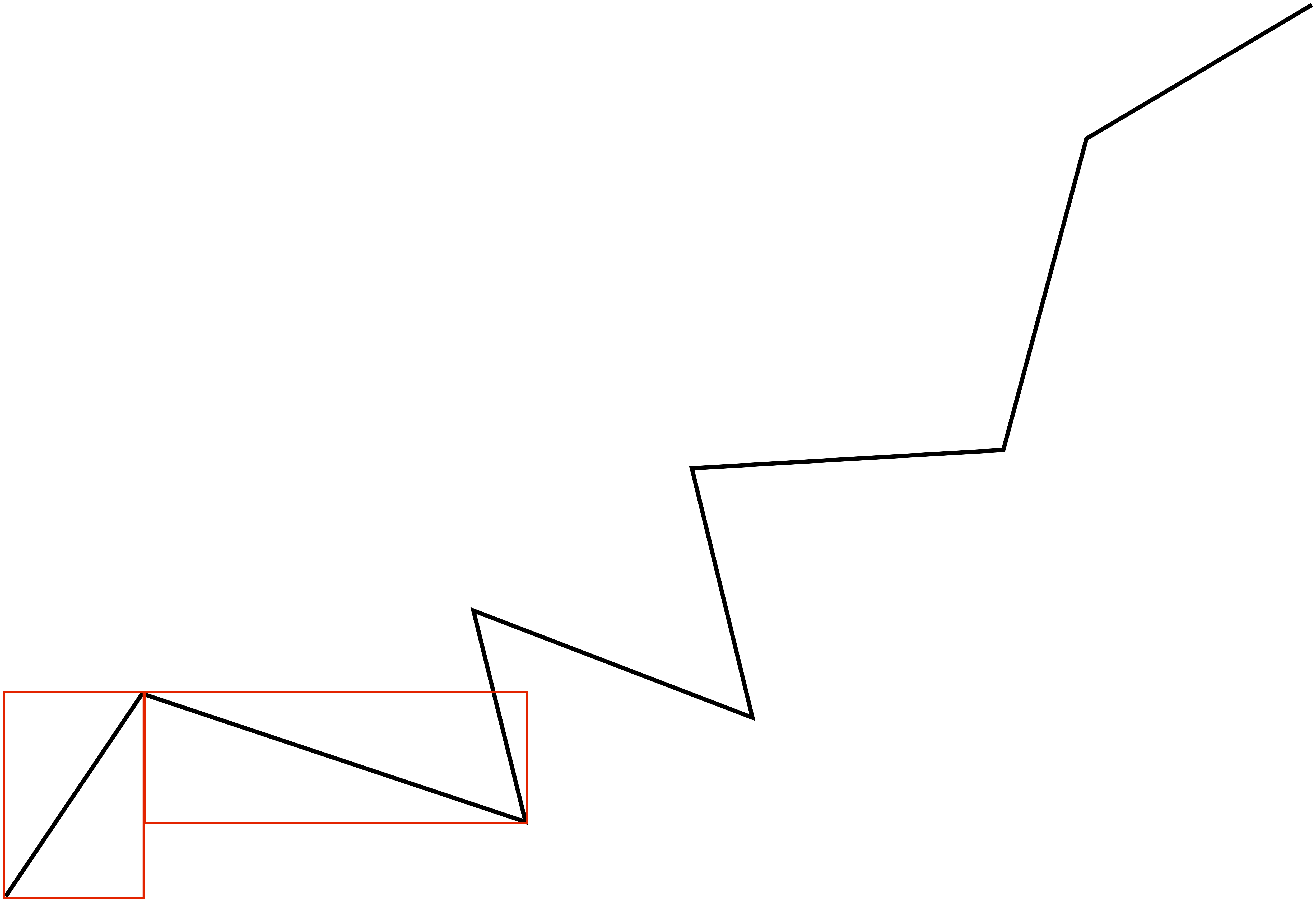
Undeformed
Configuration

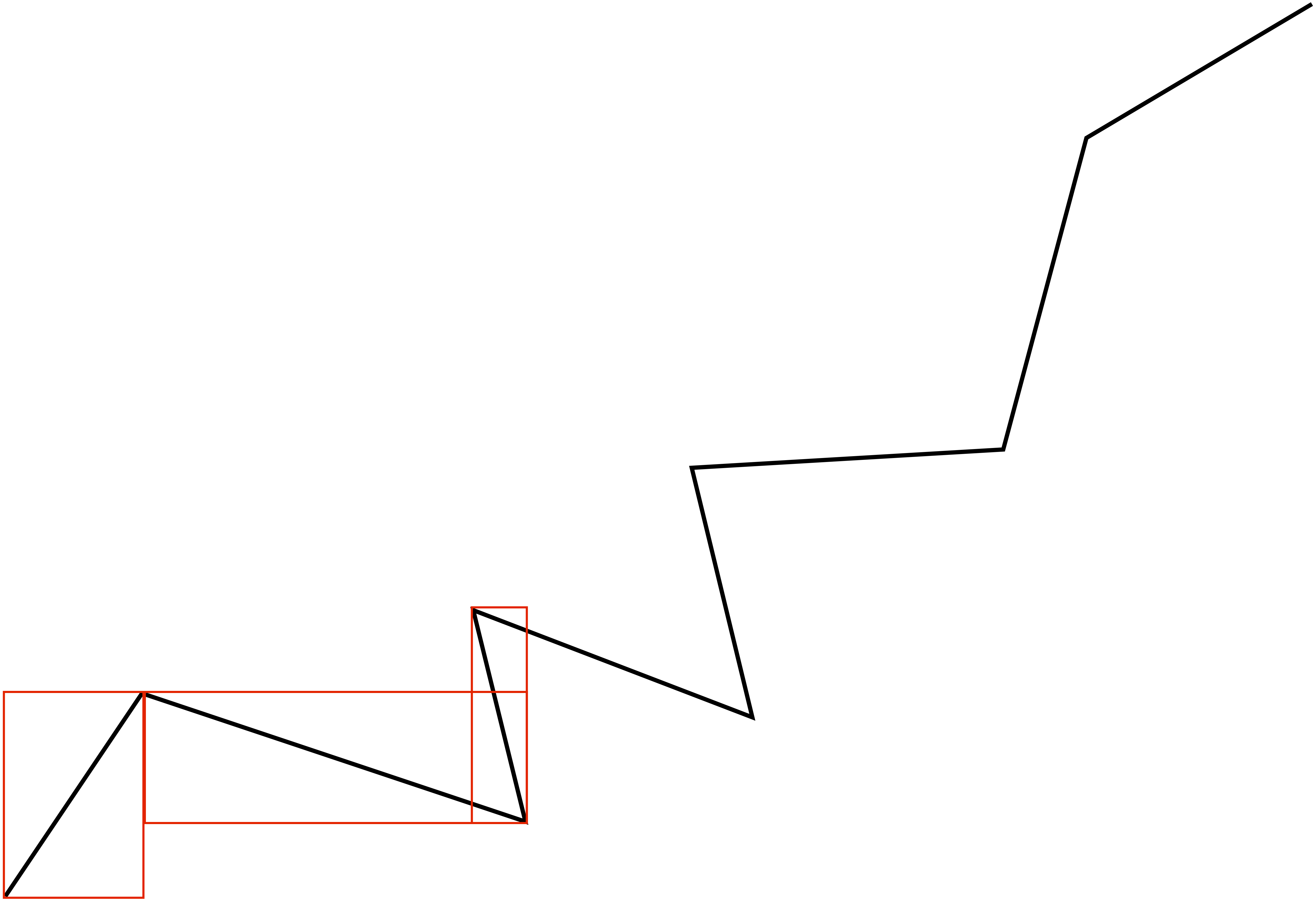


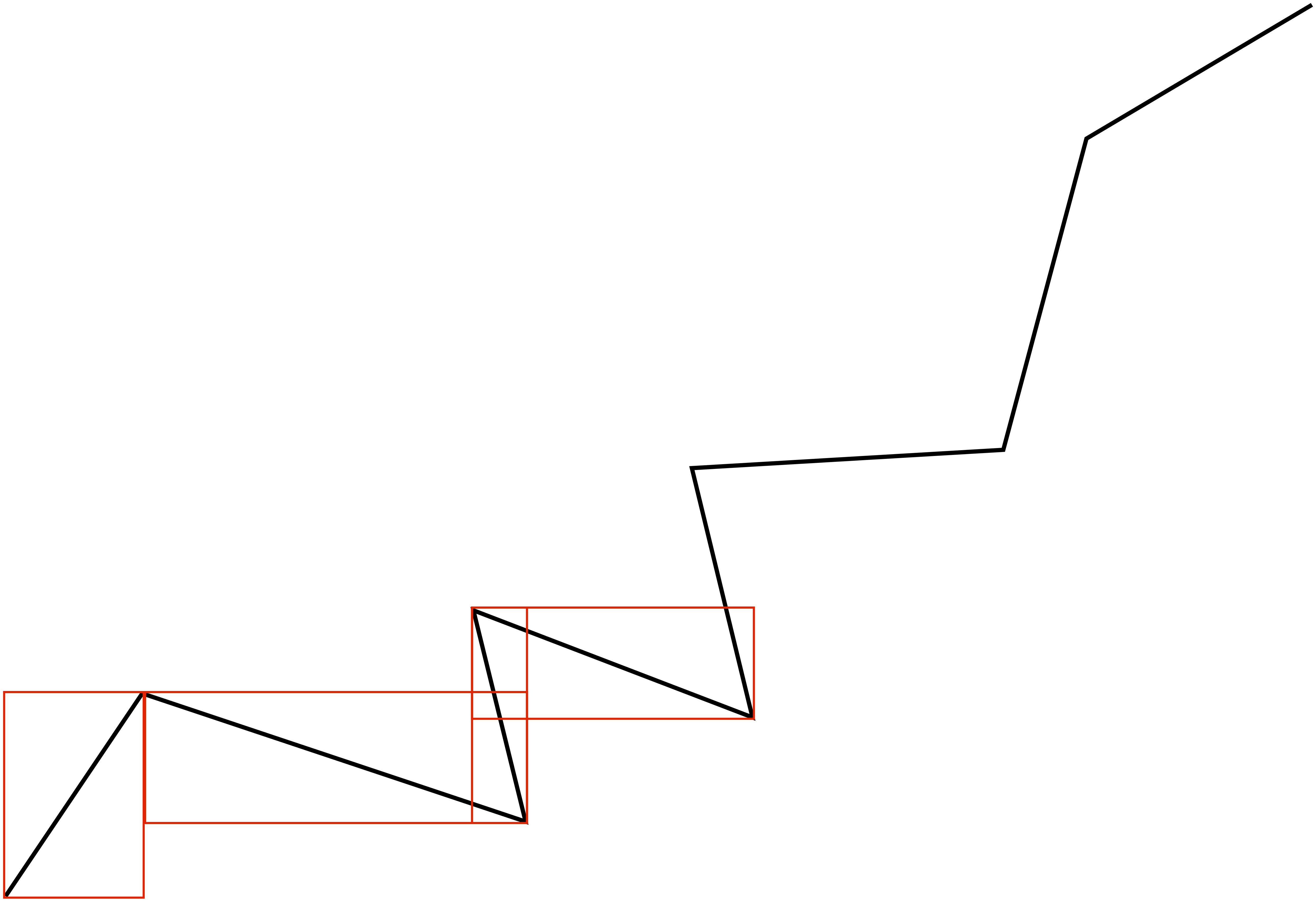


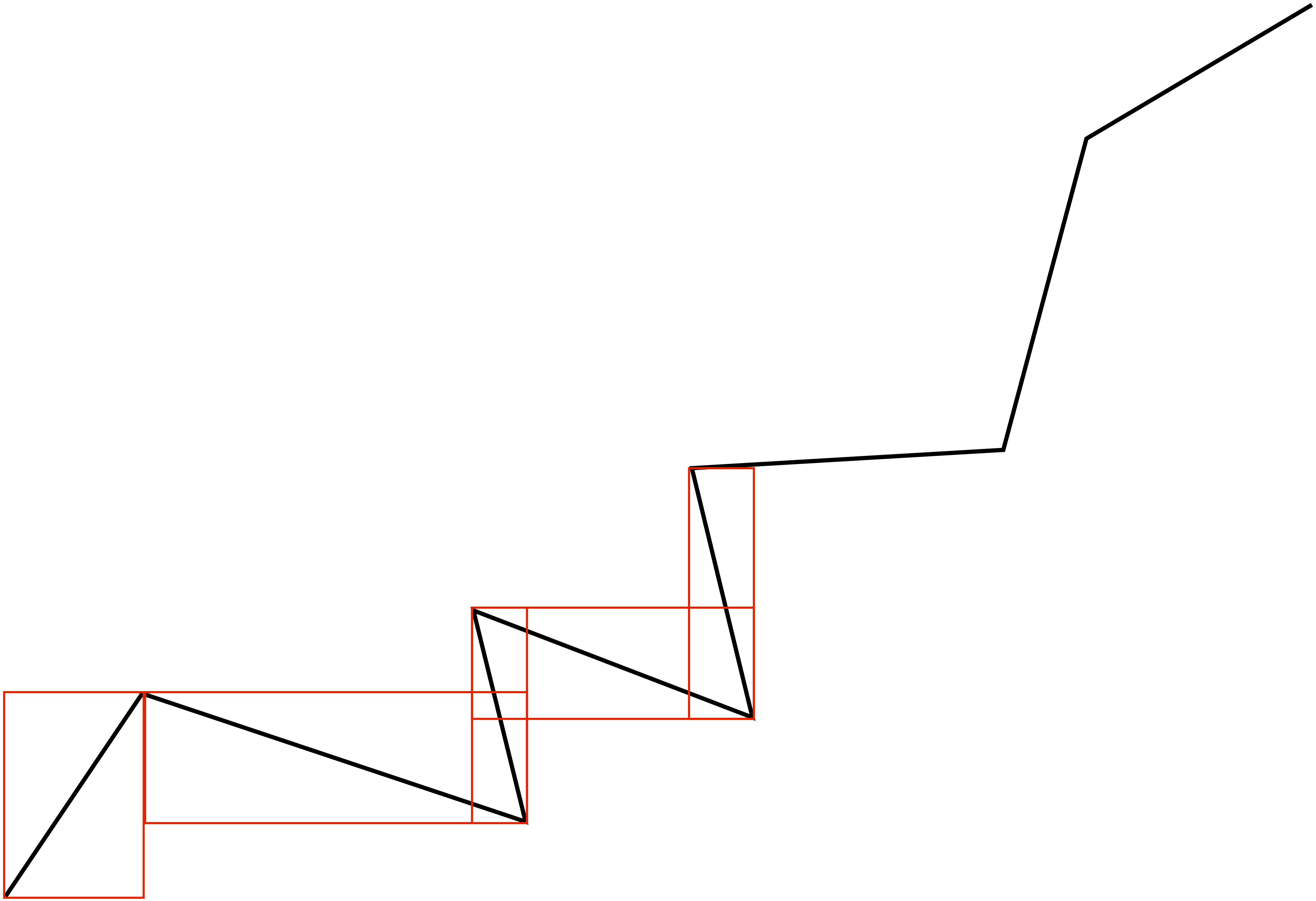


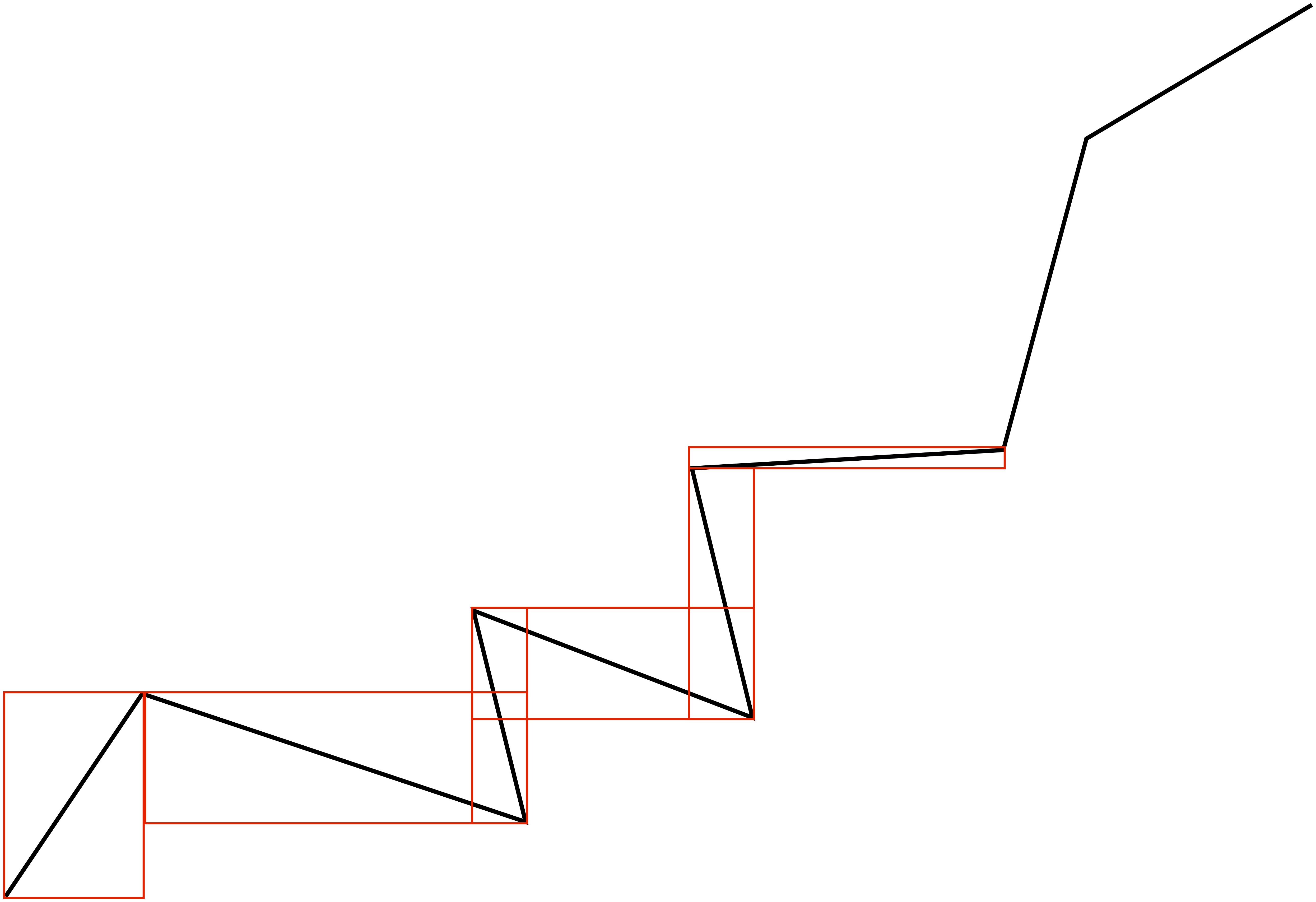


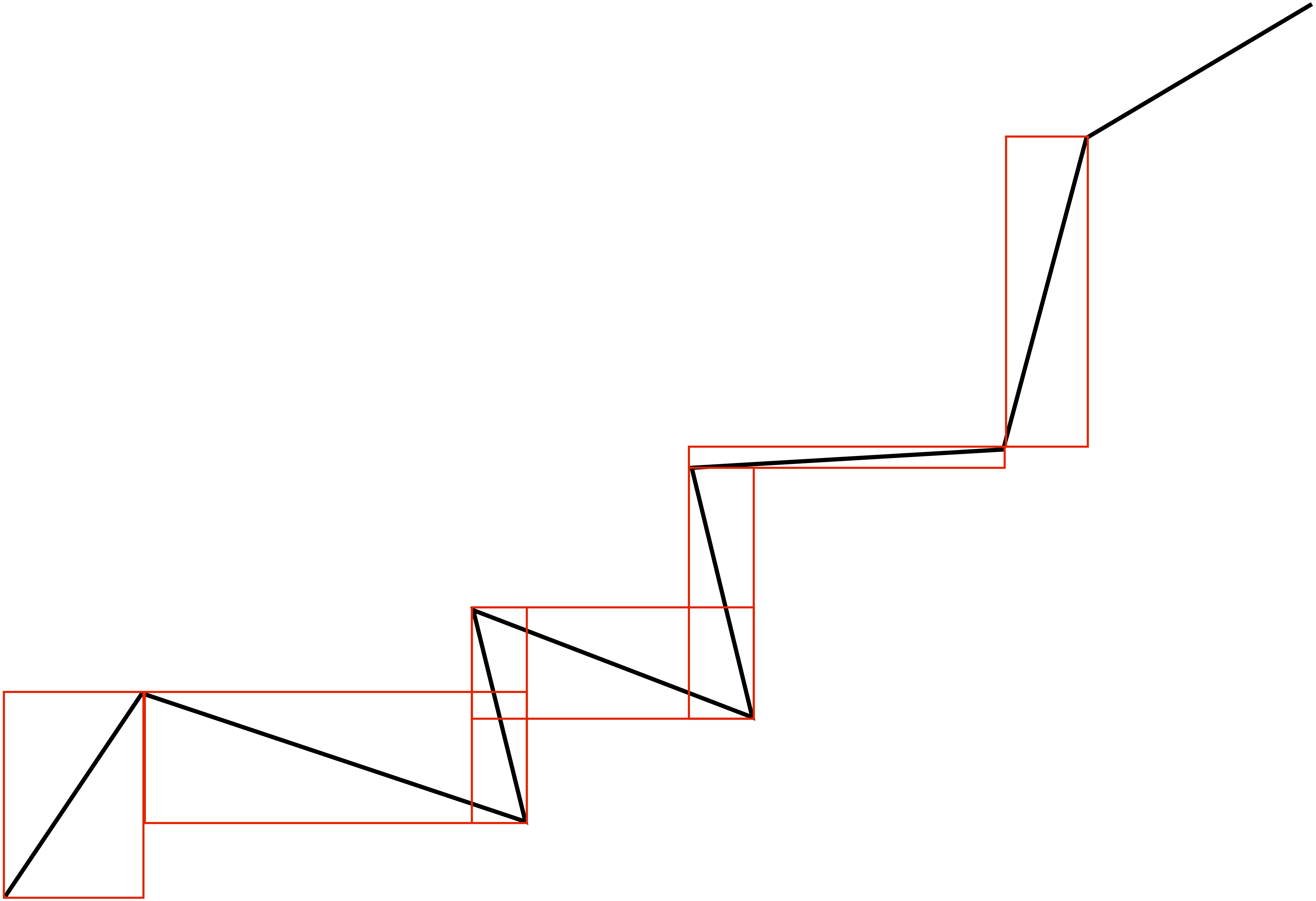


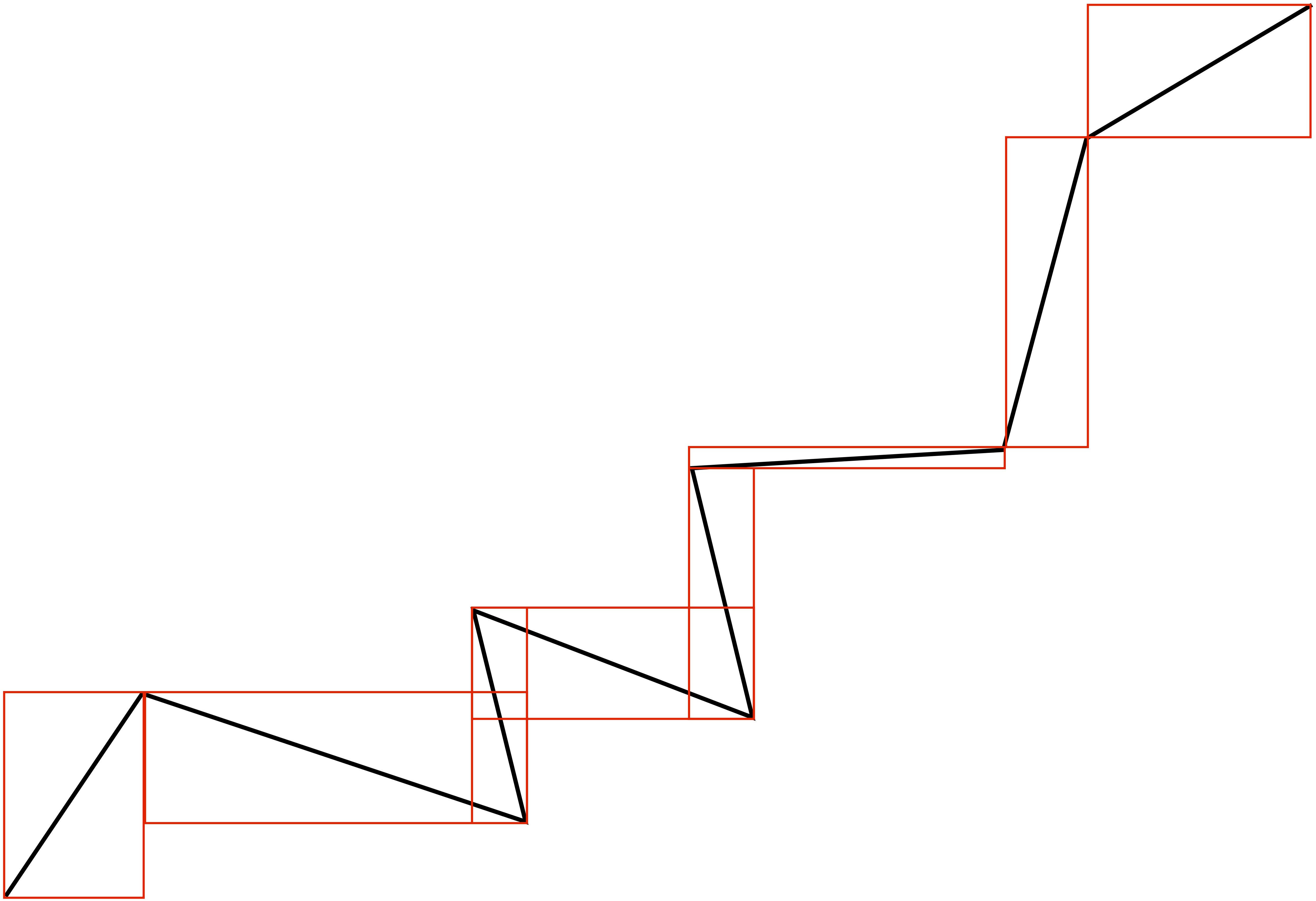


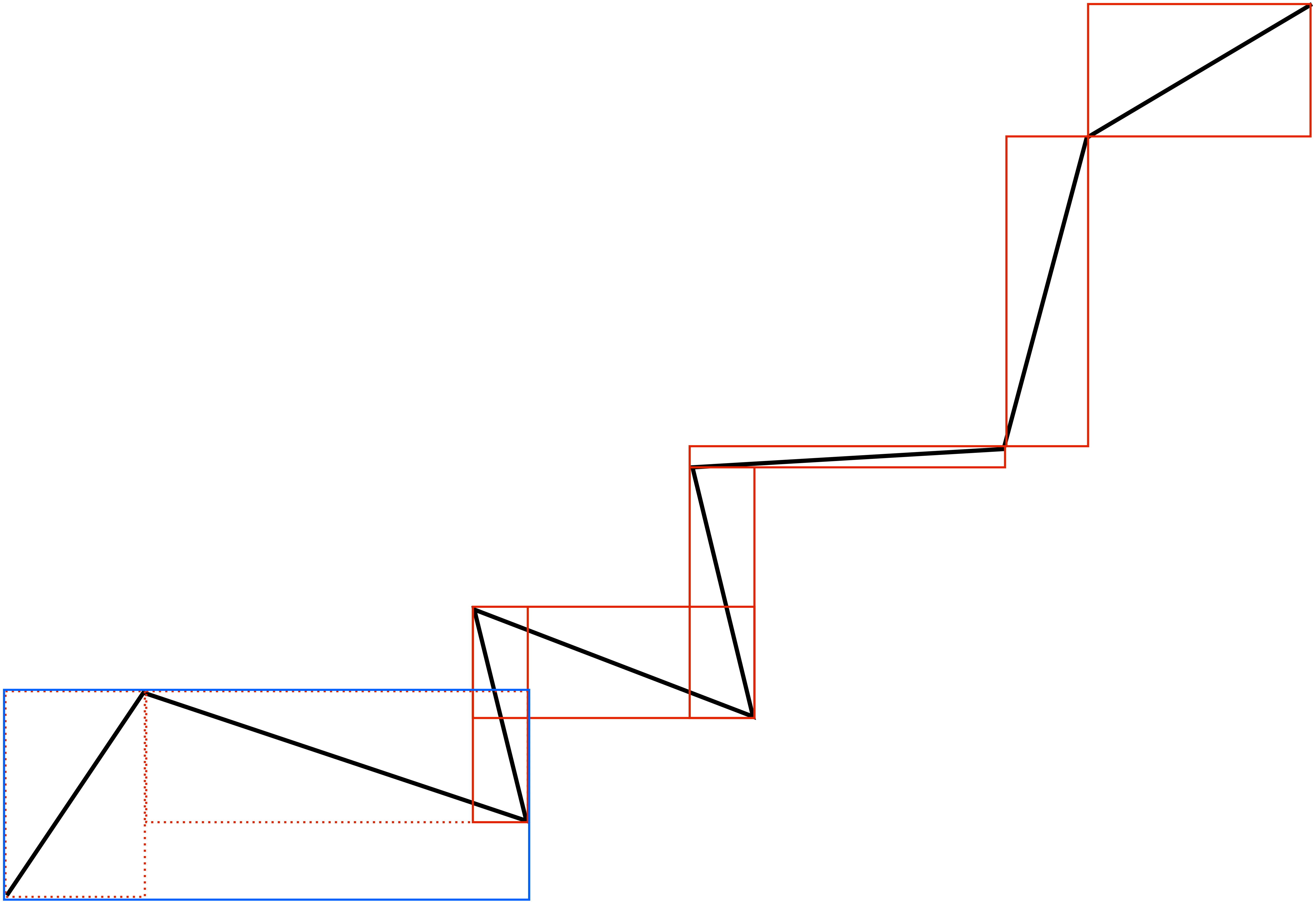


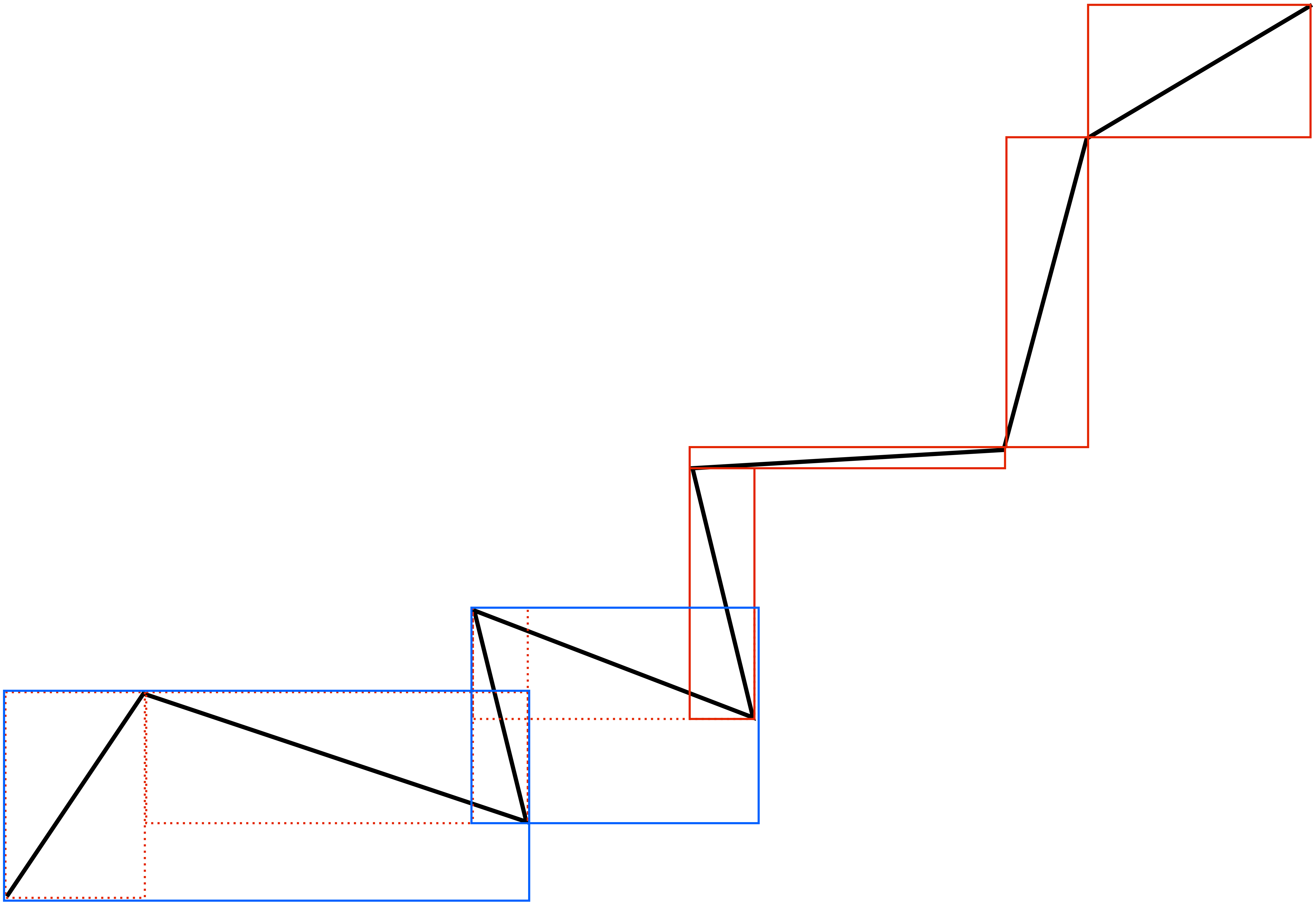


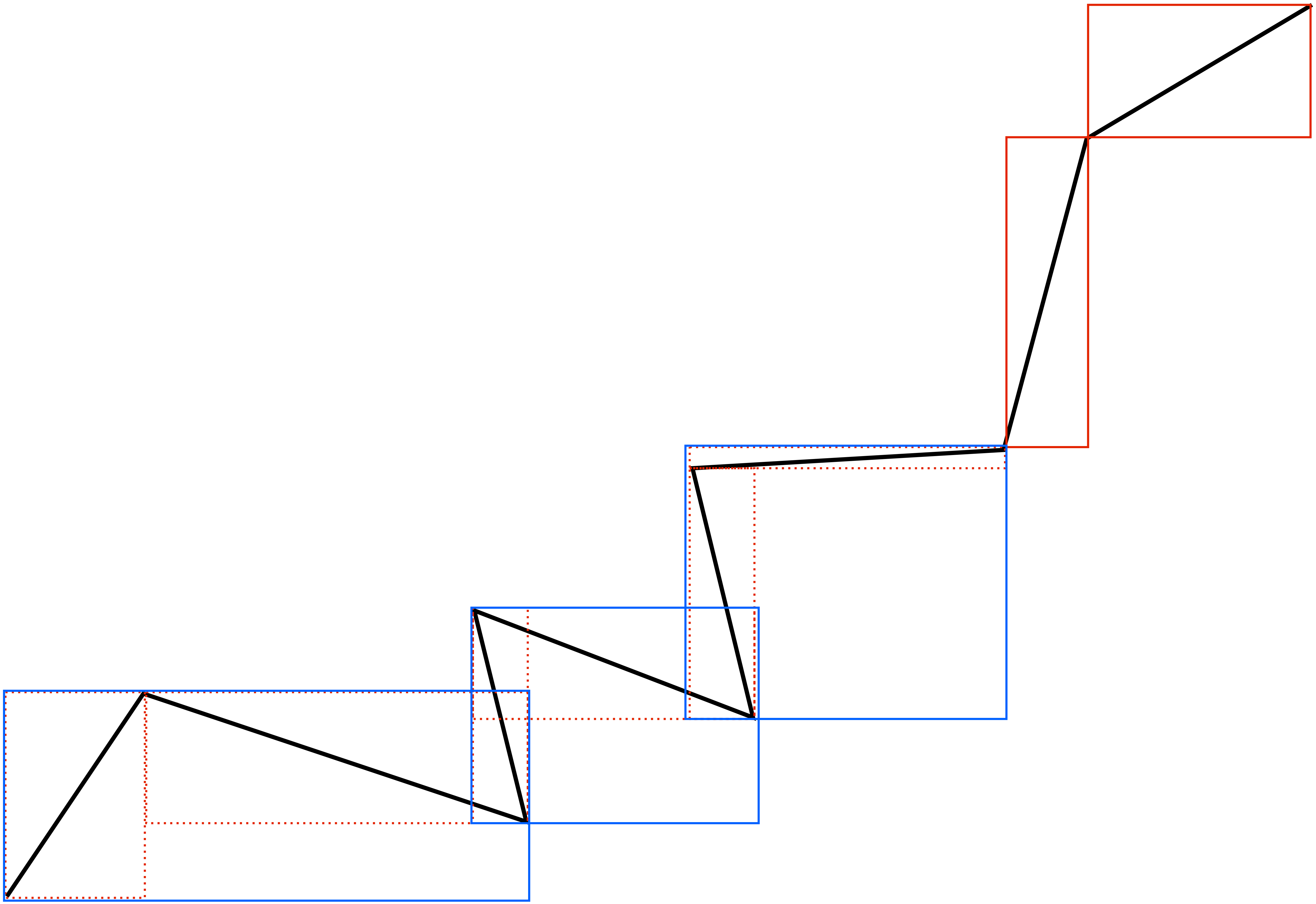


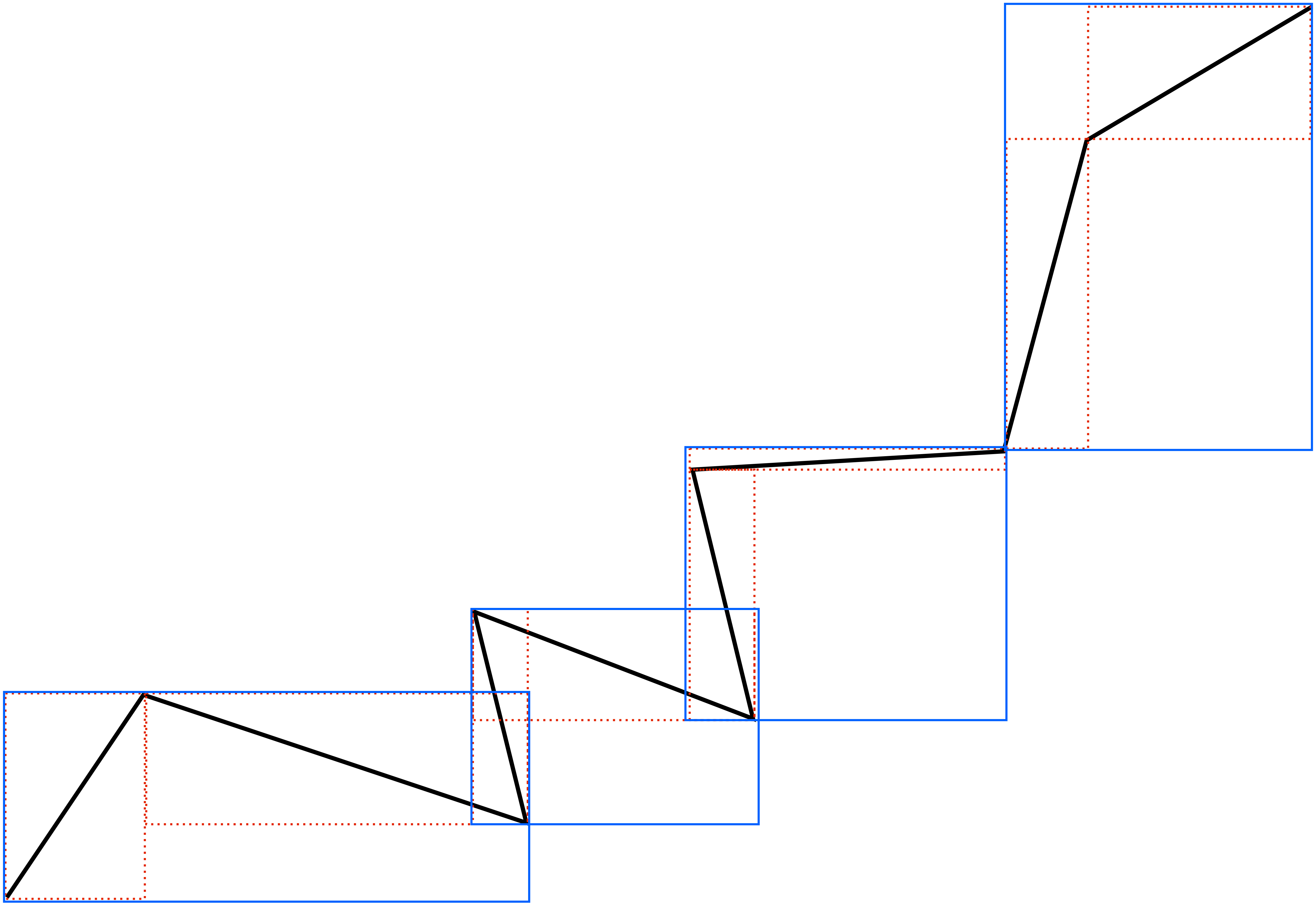


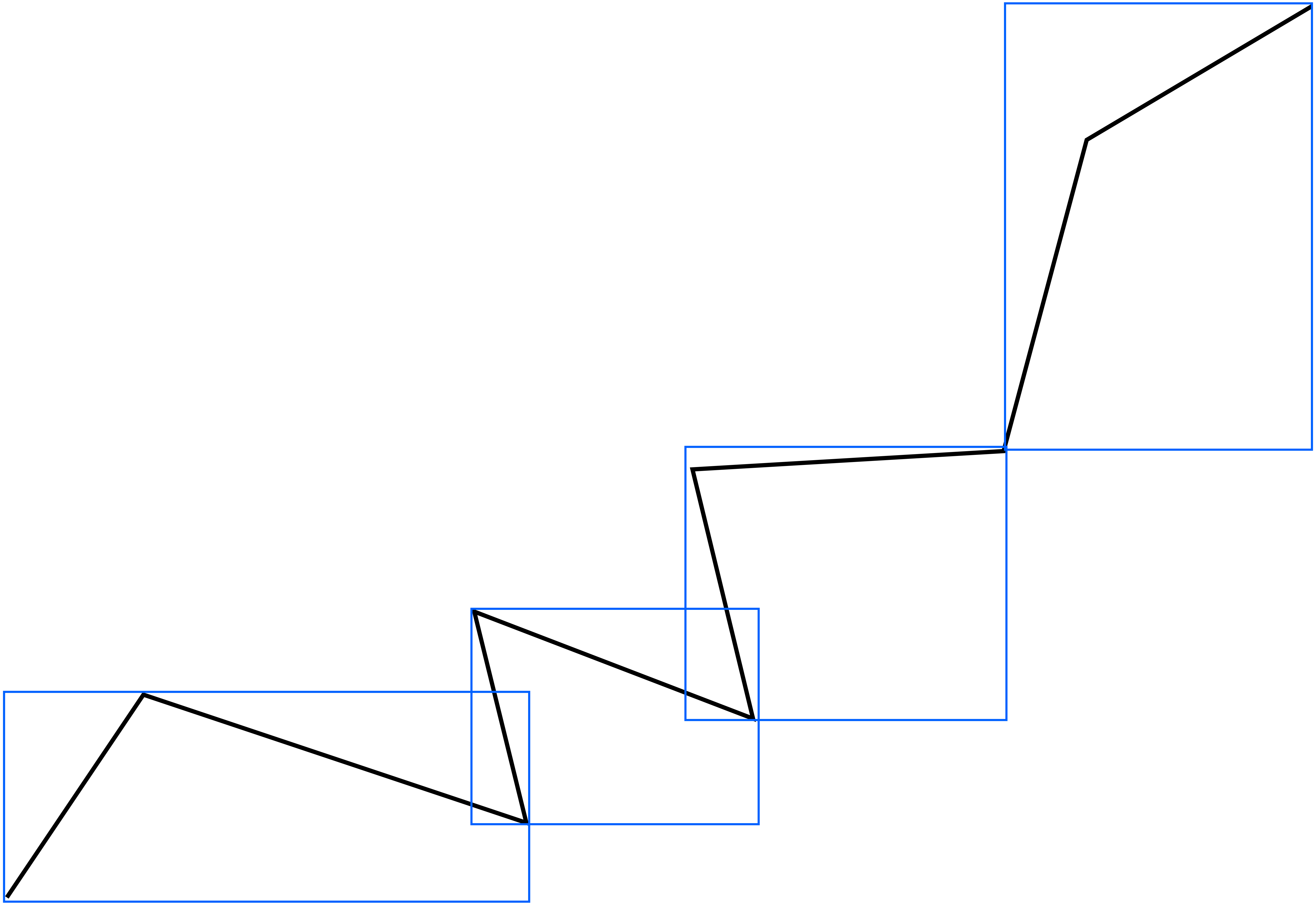


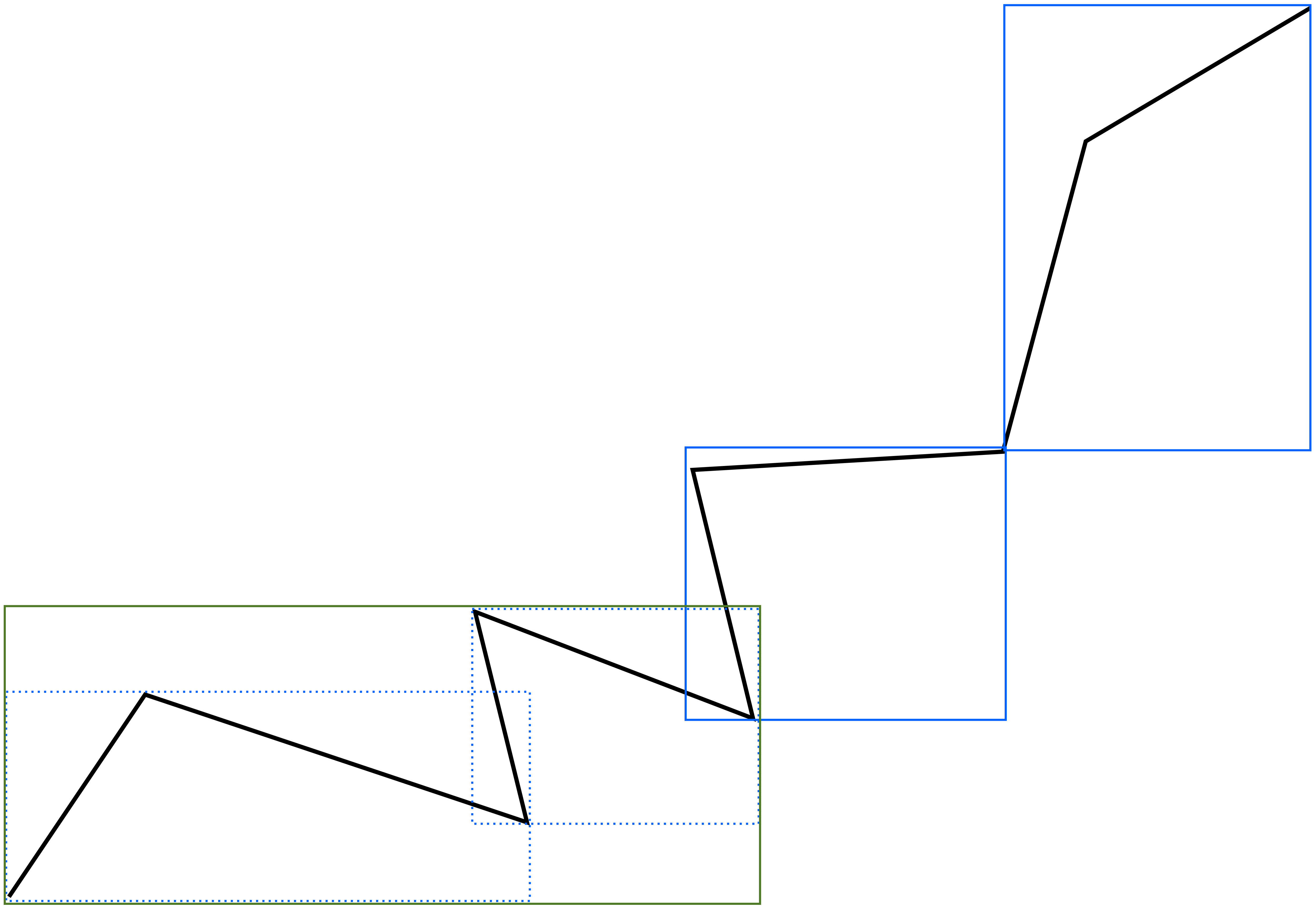


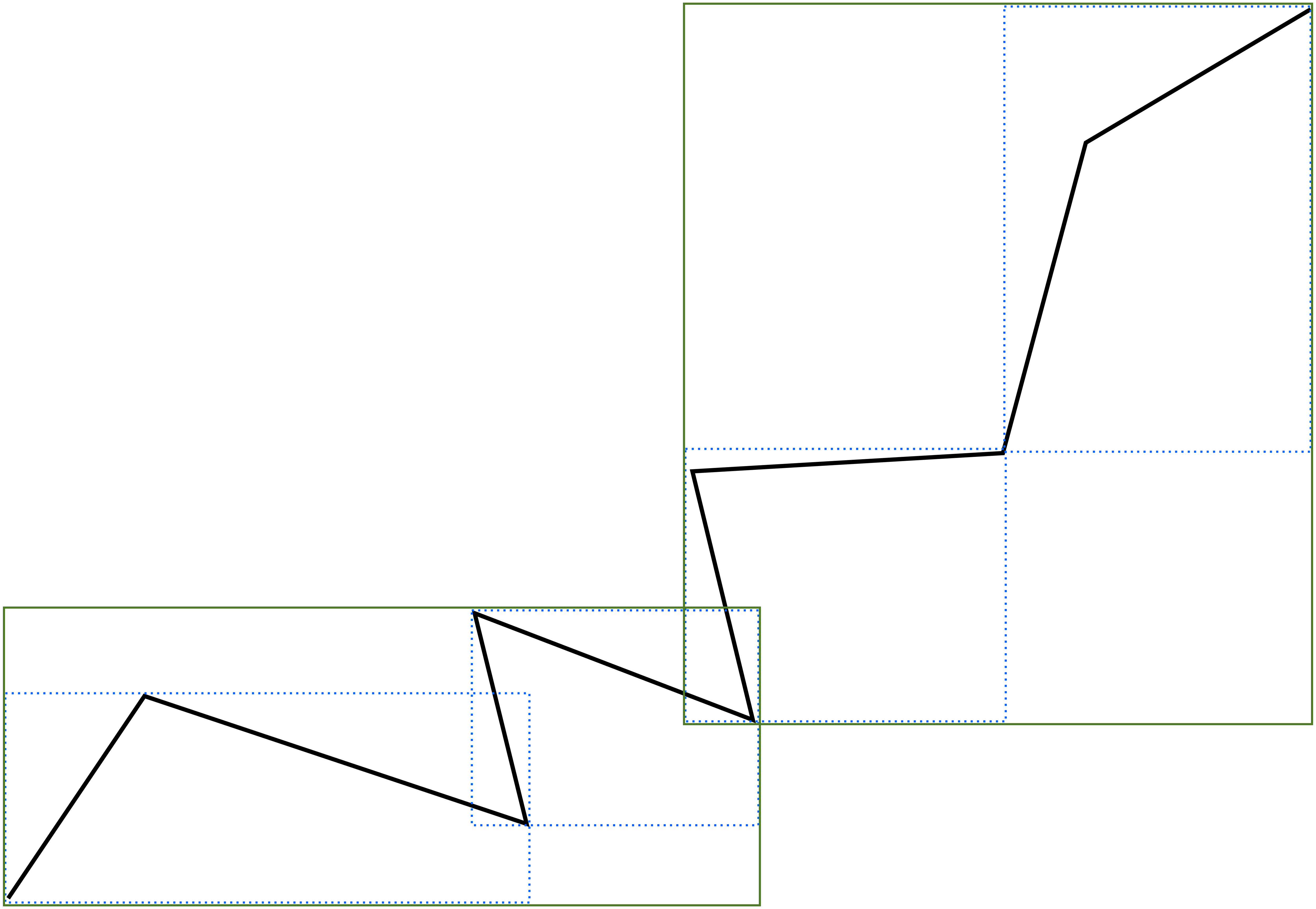


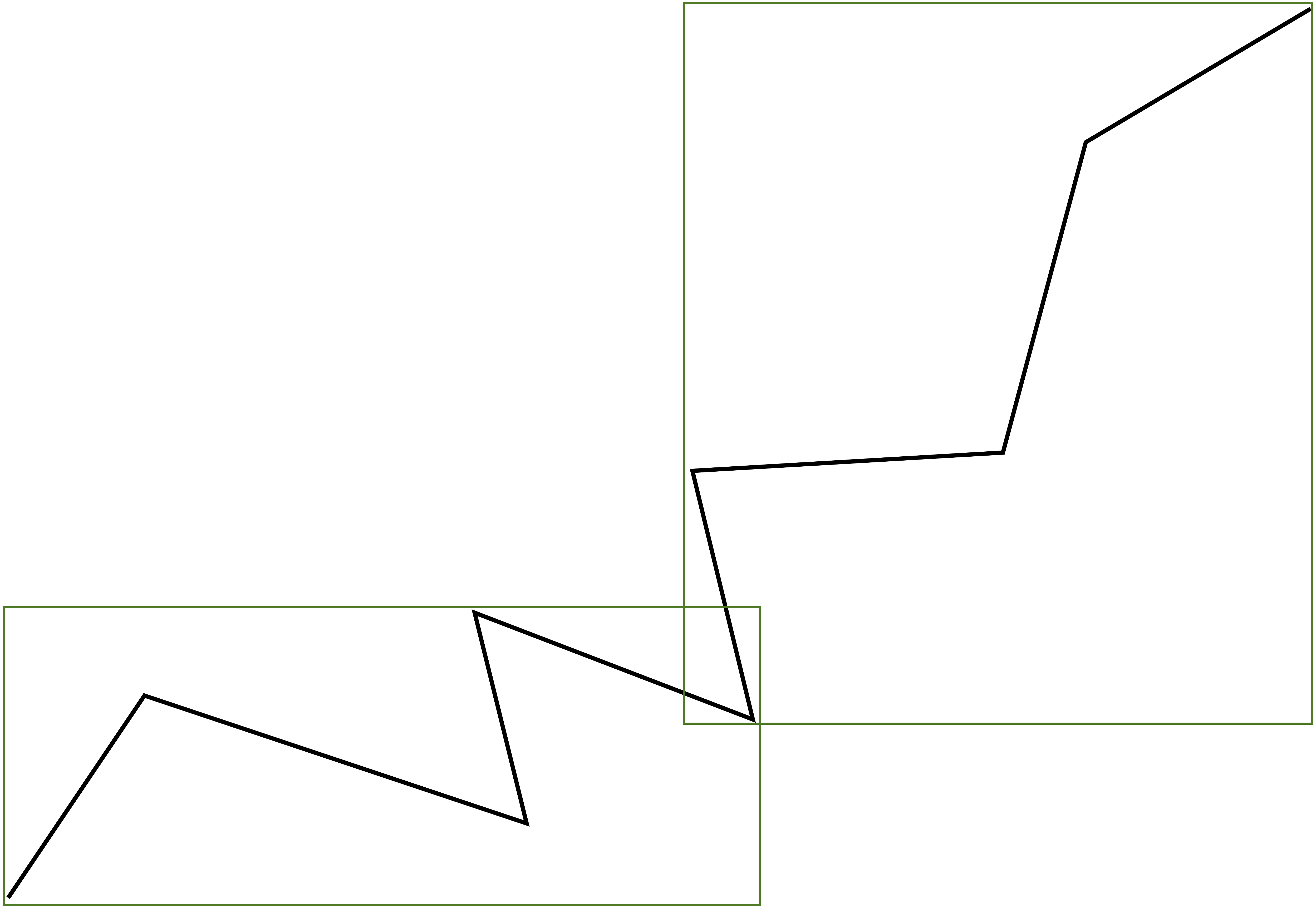


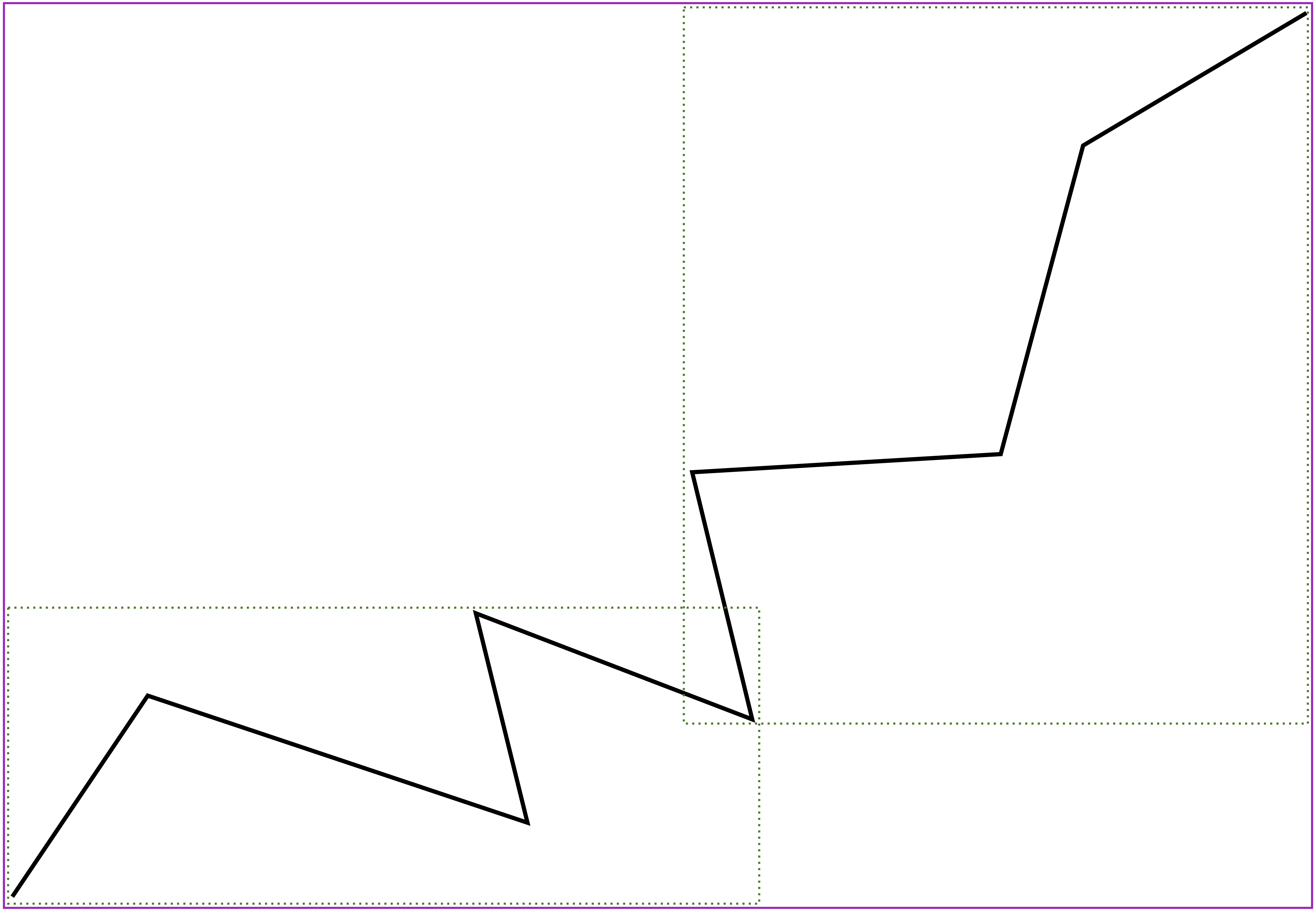


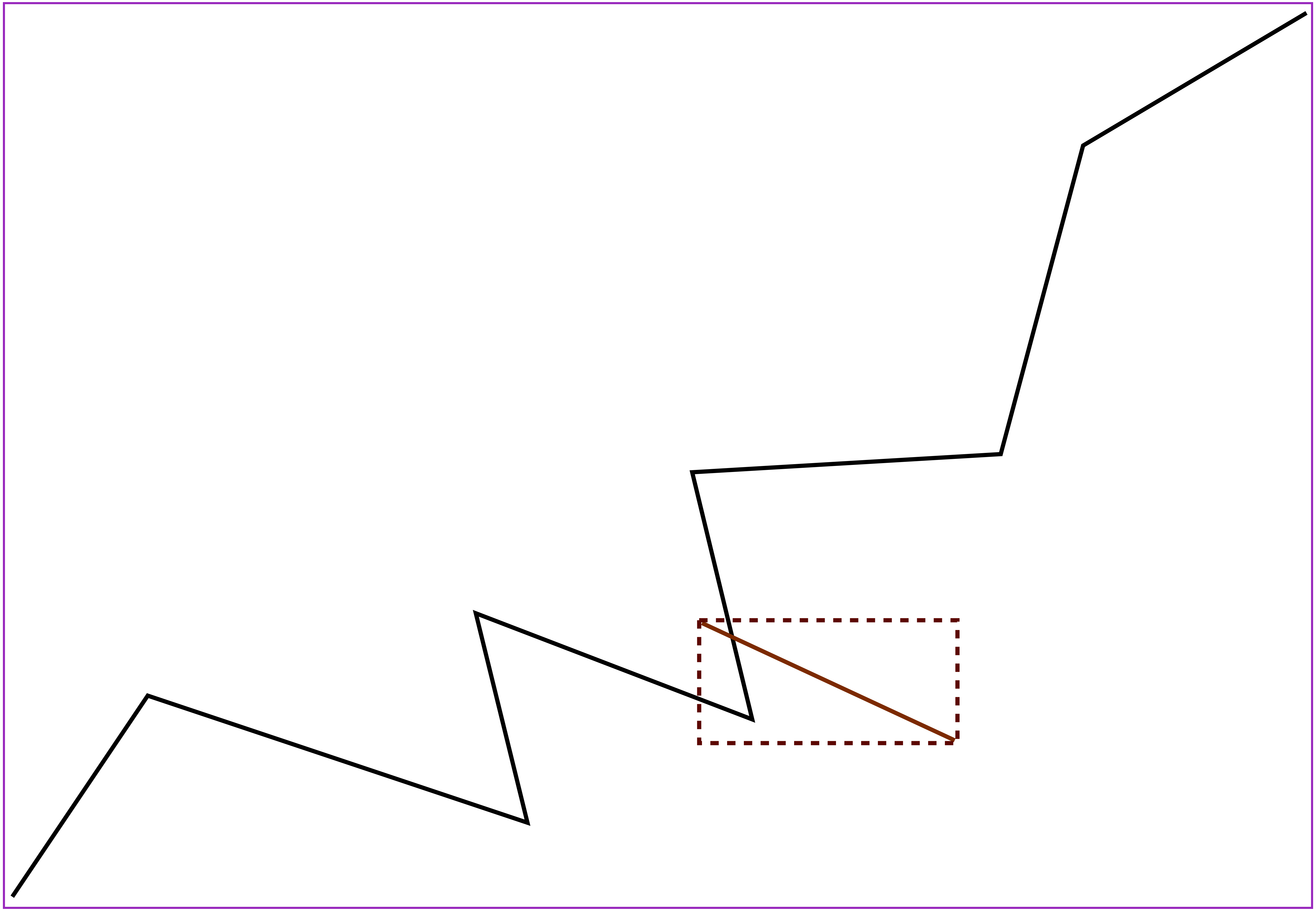


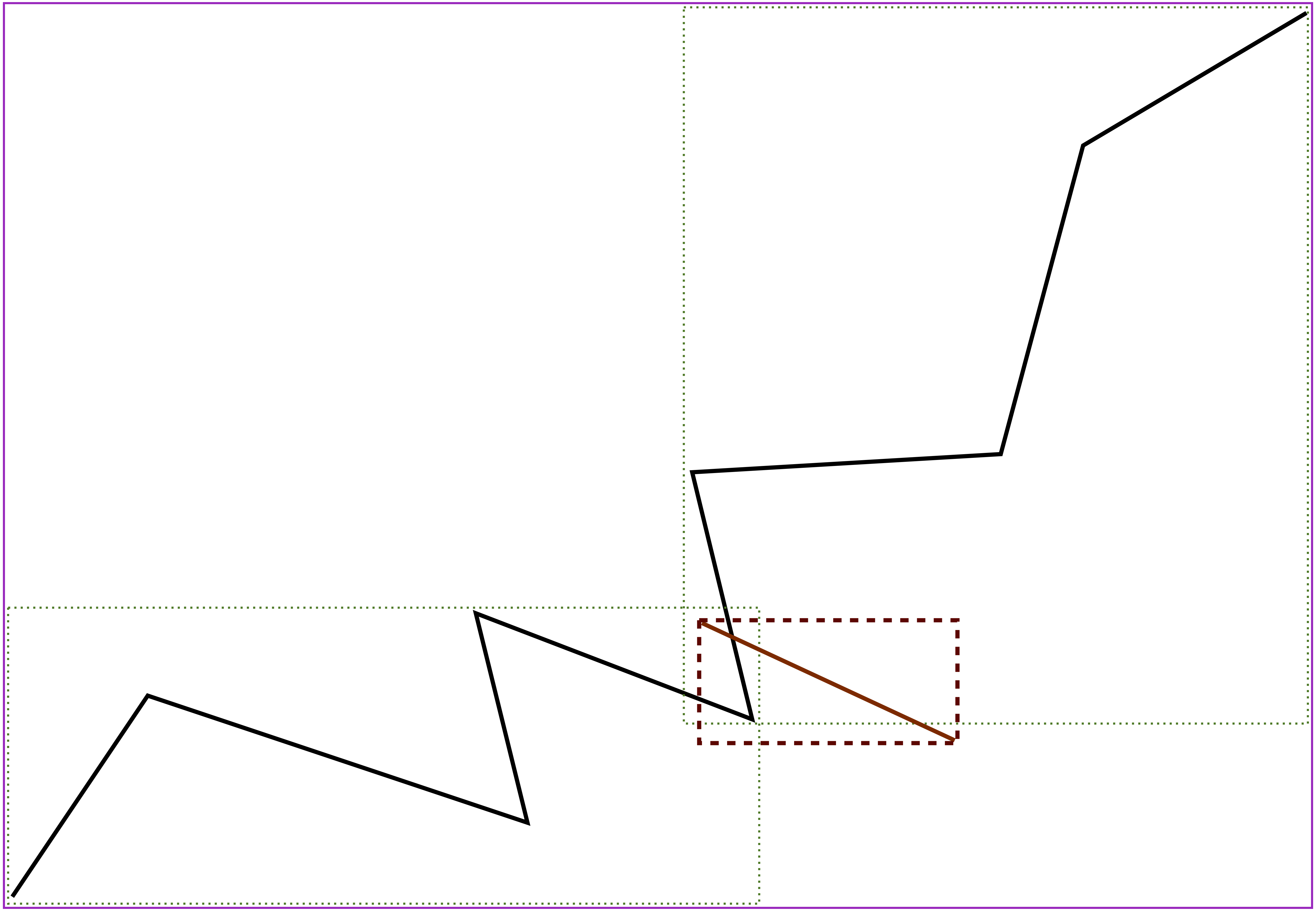


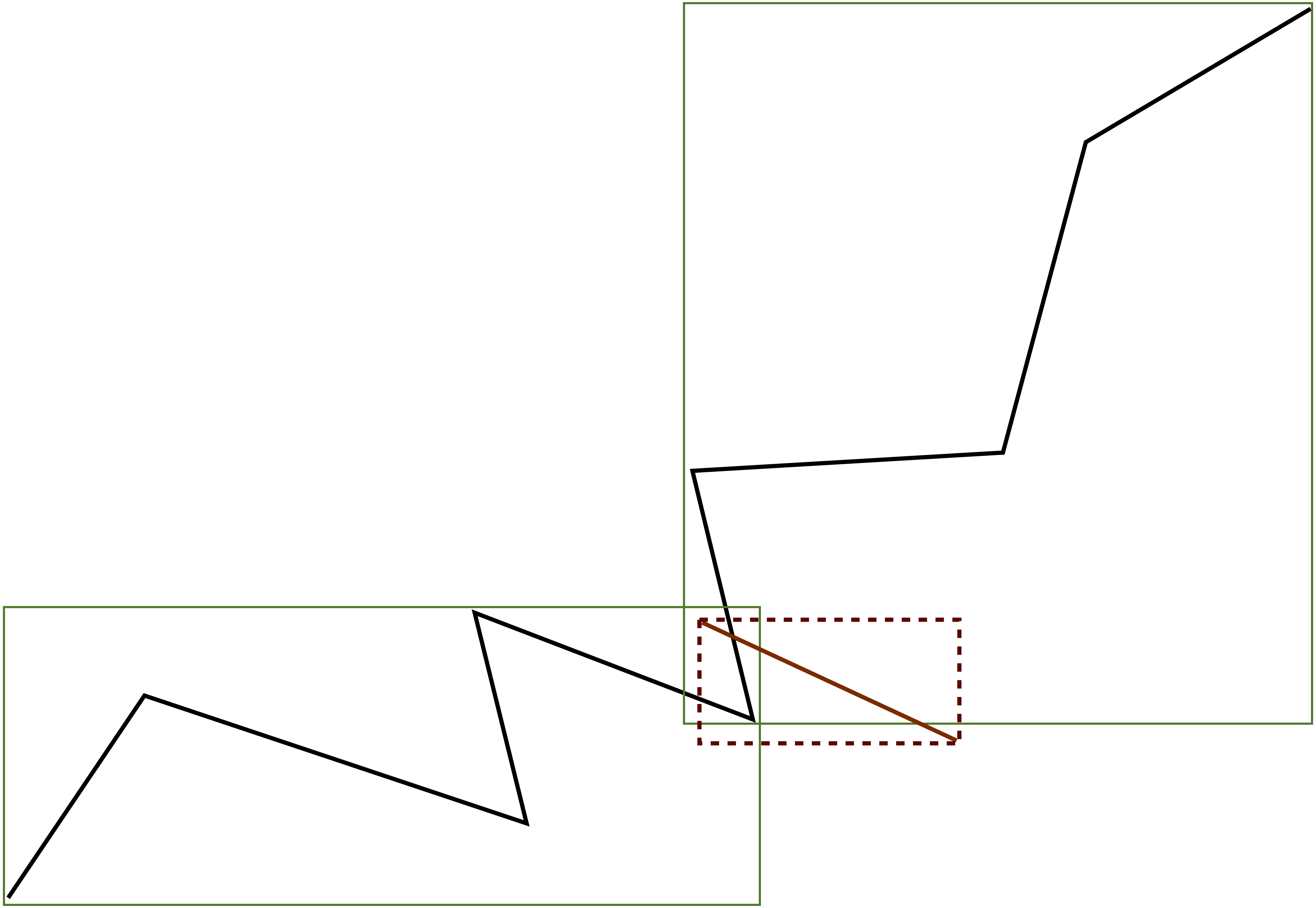


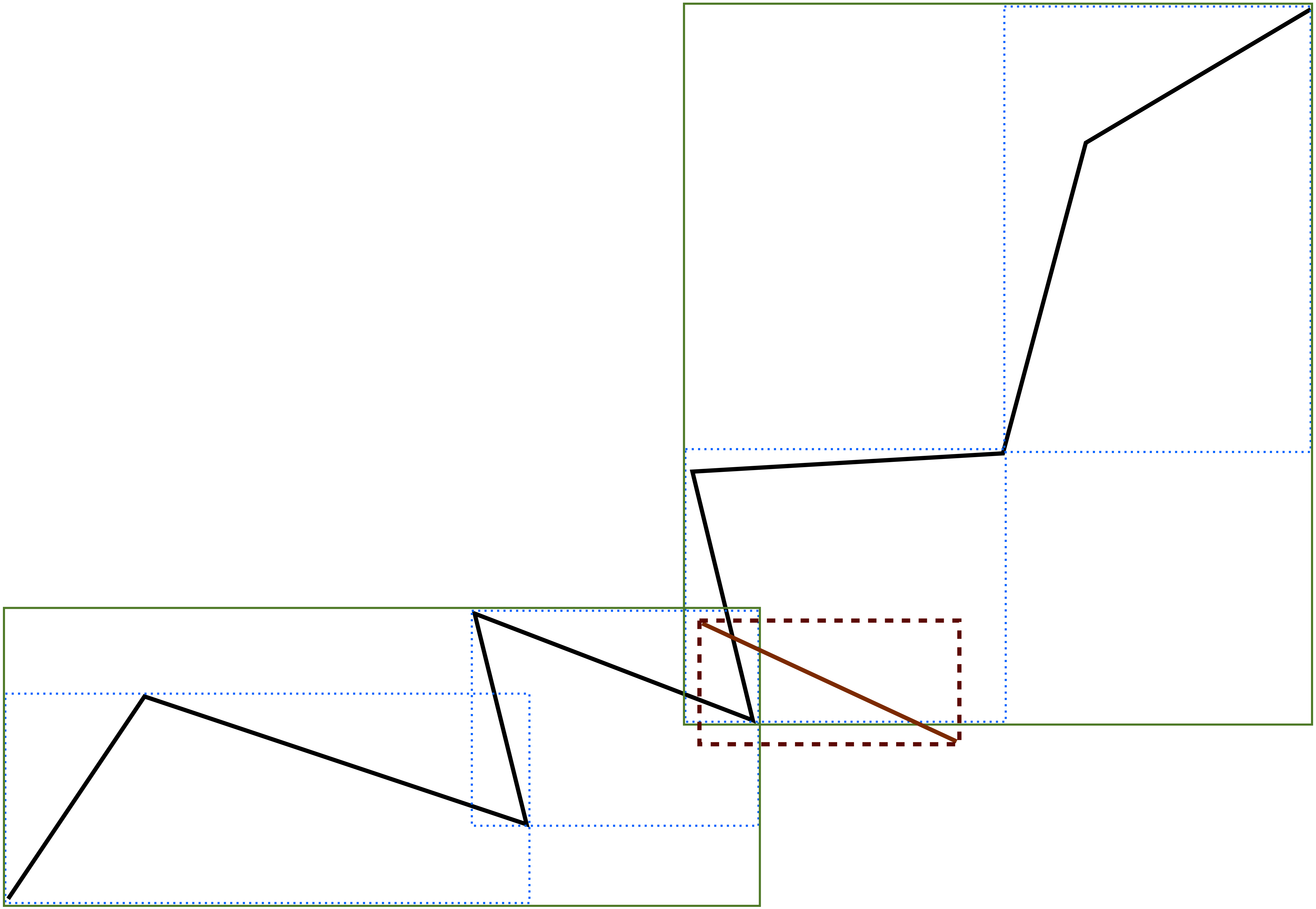


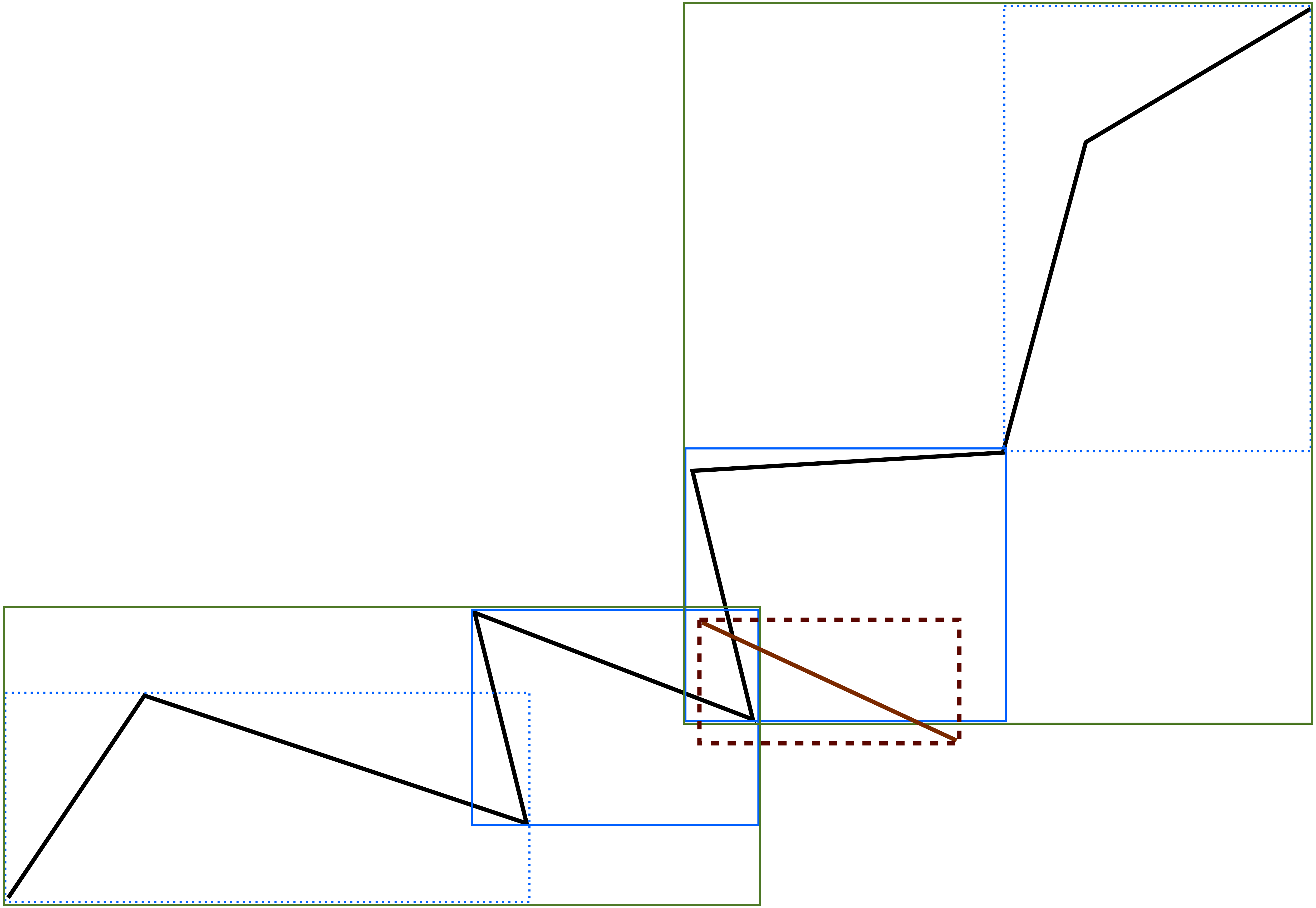


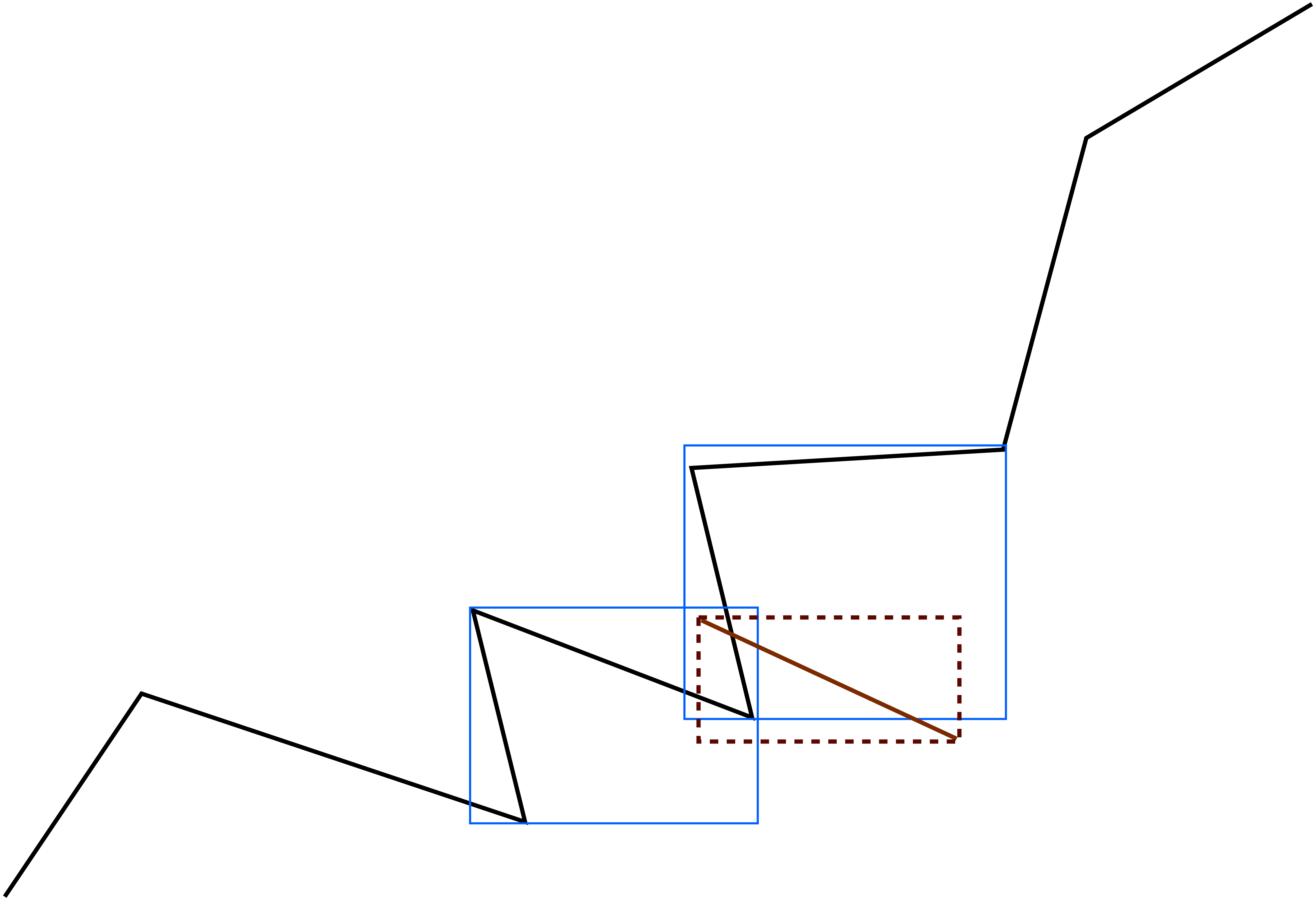


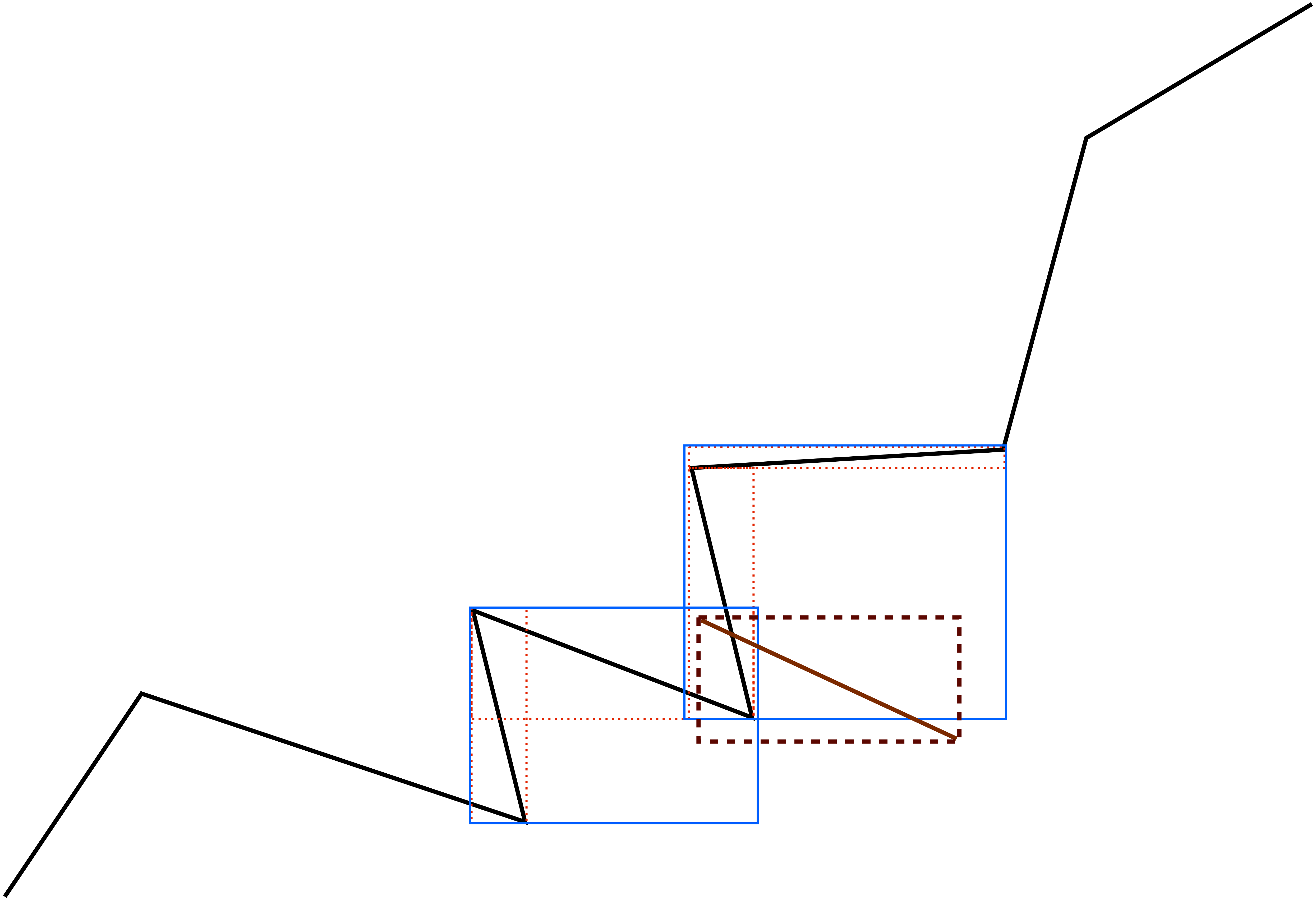


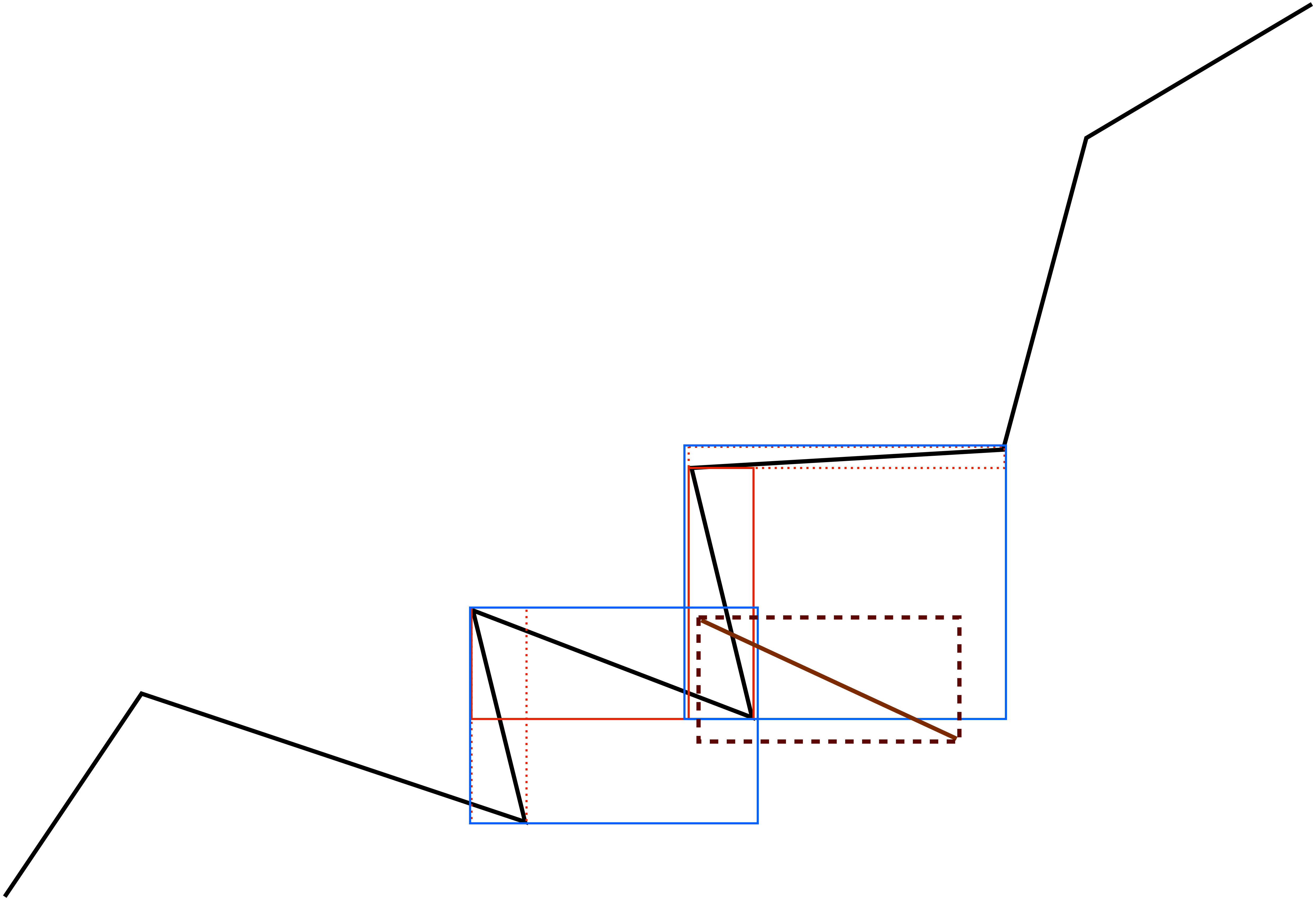


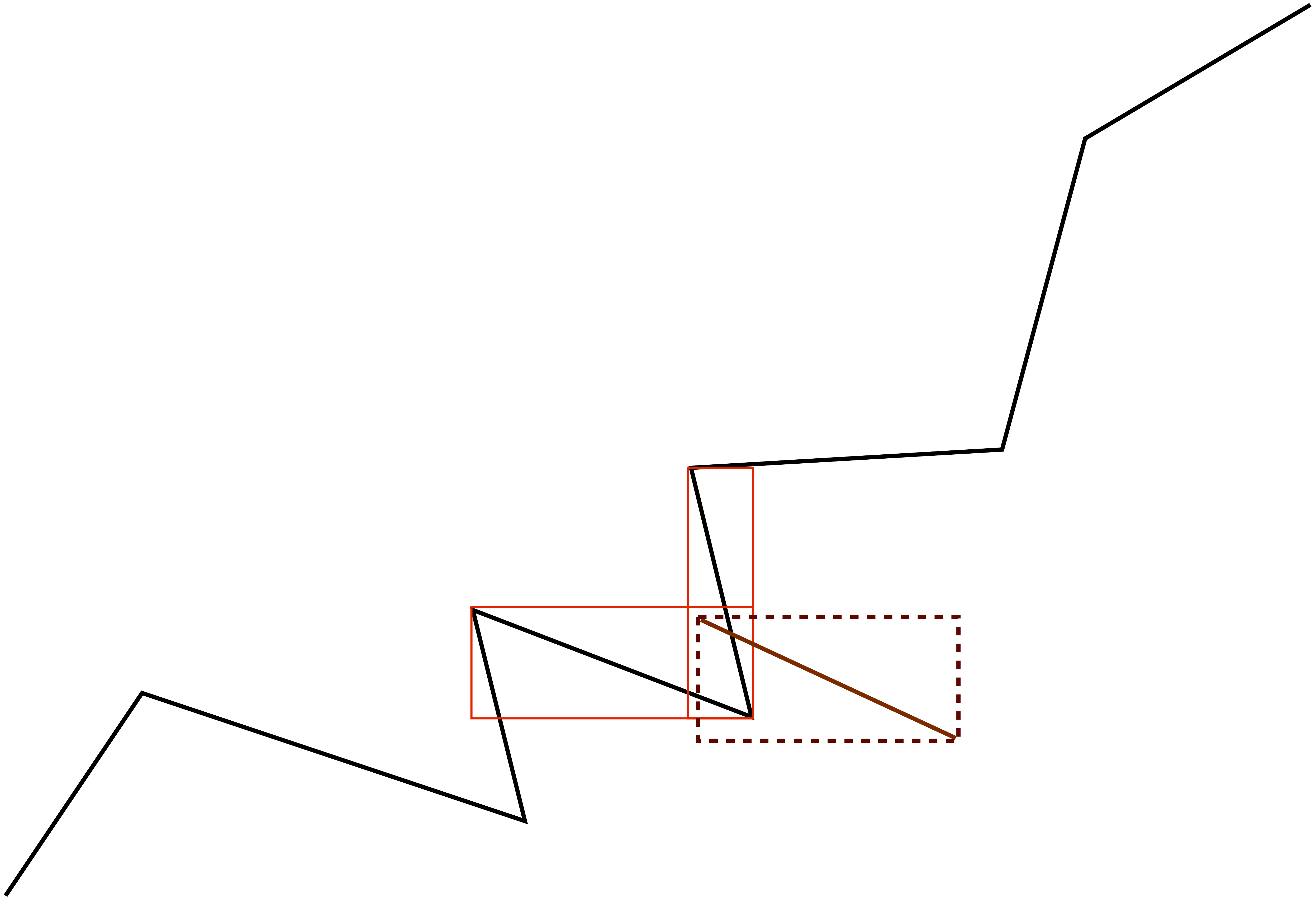


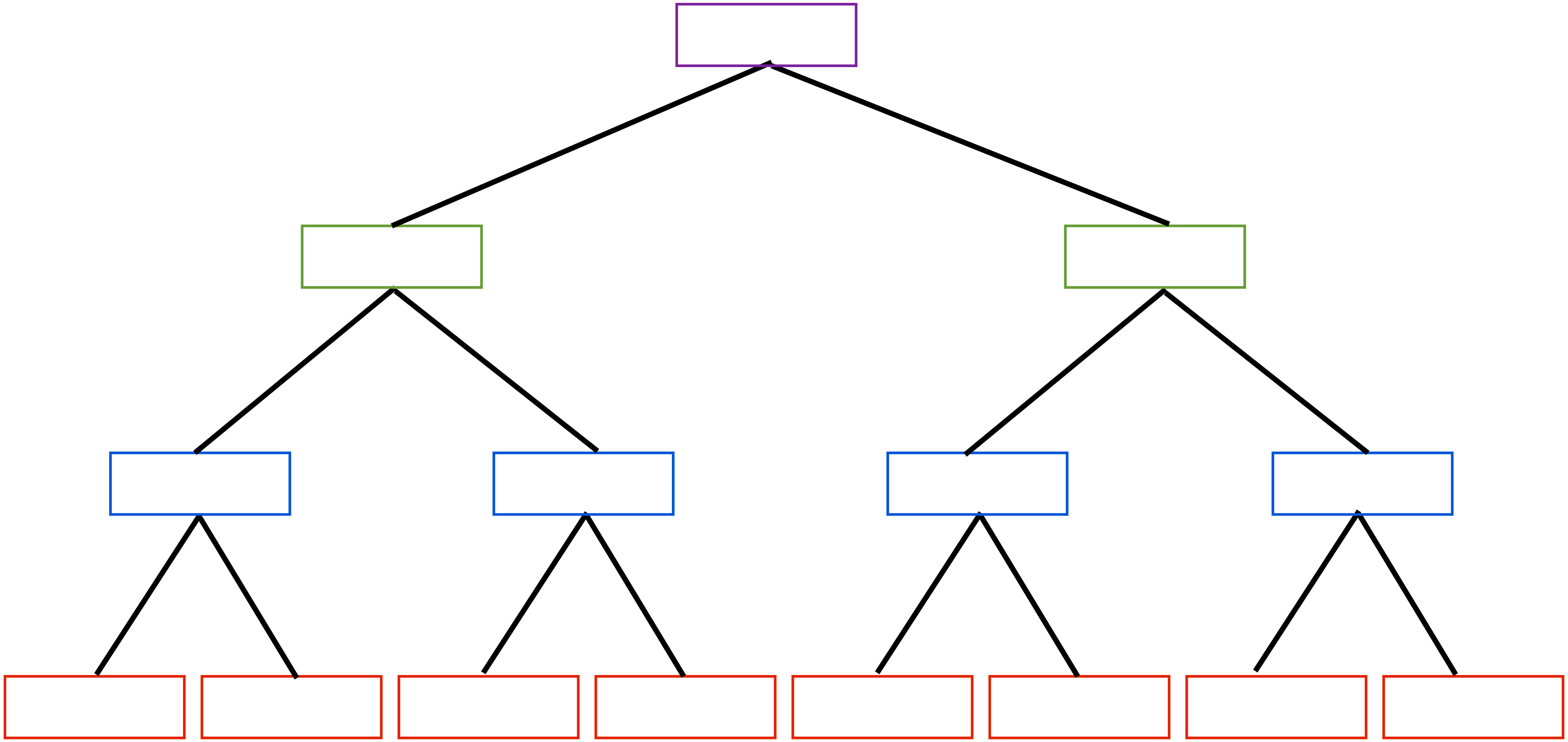


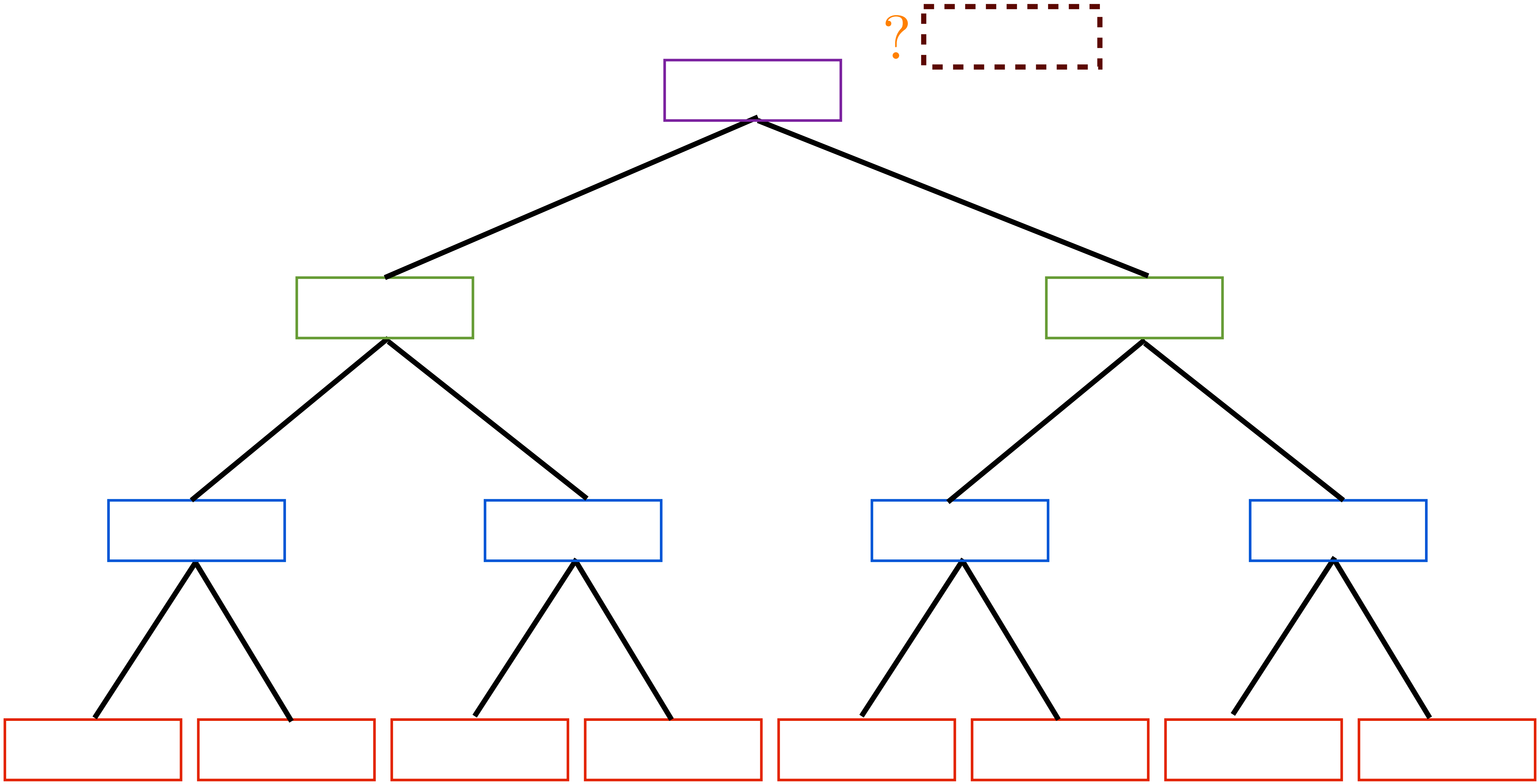


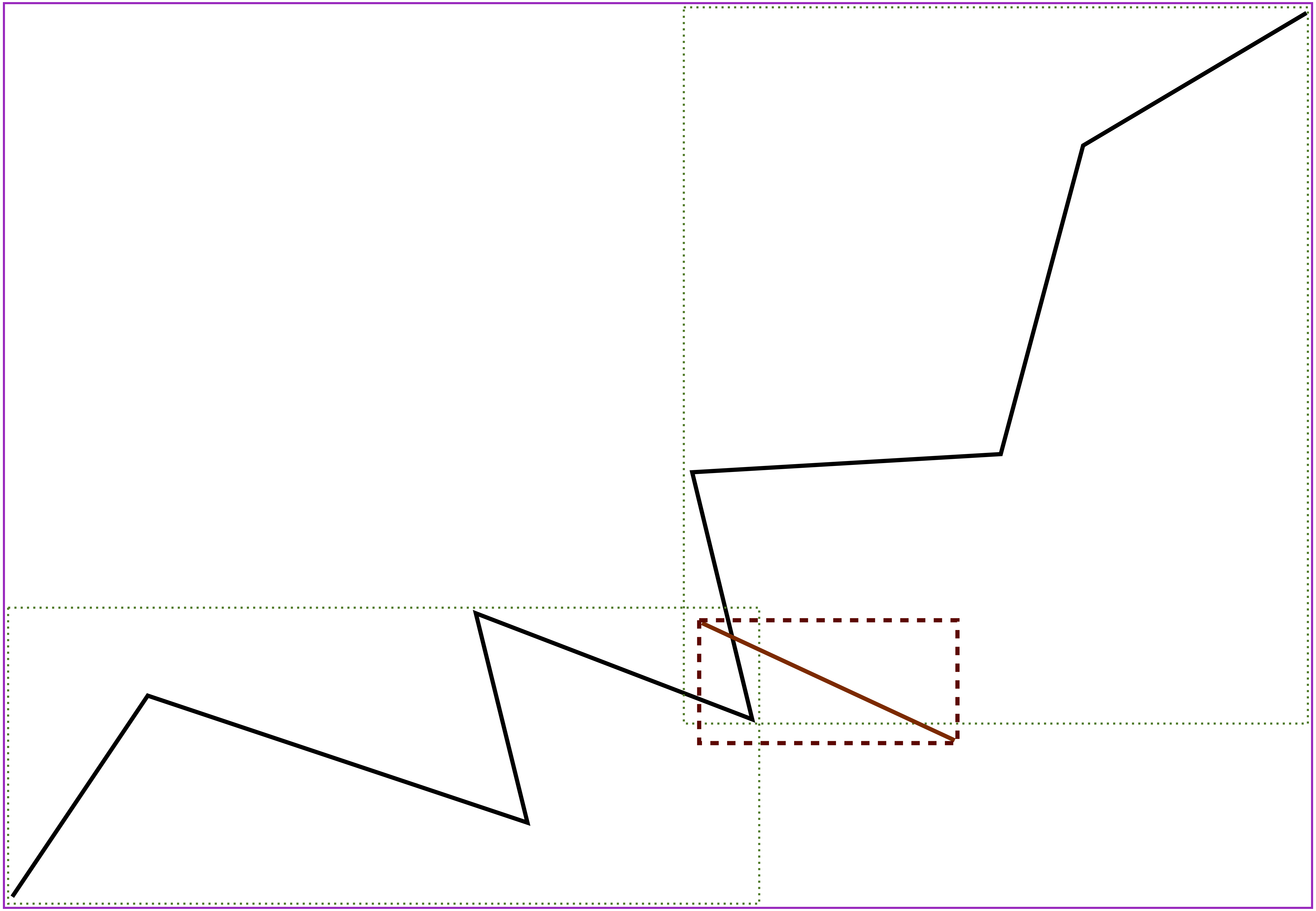


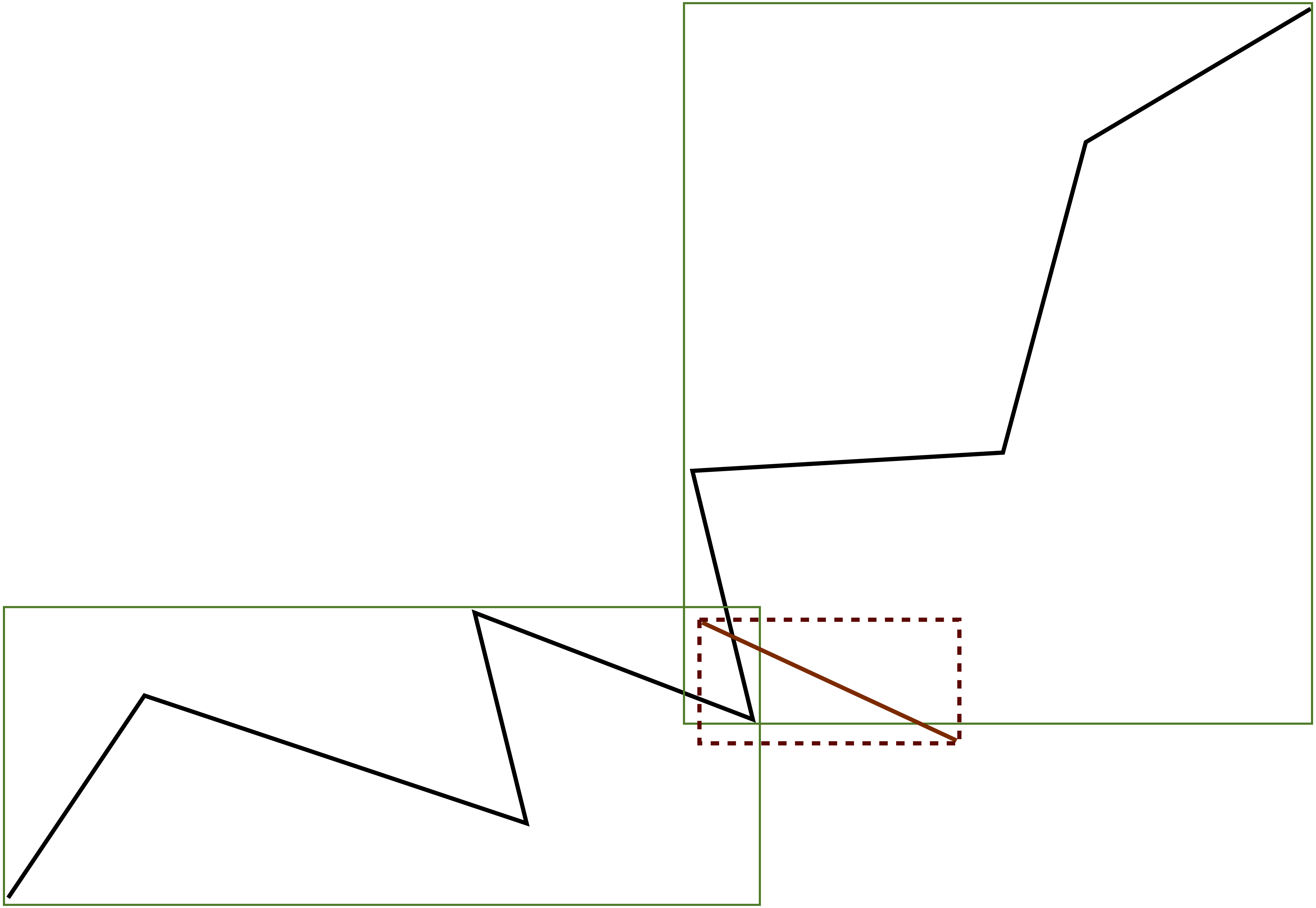


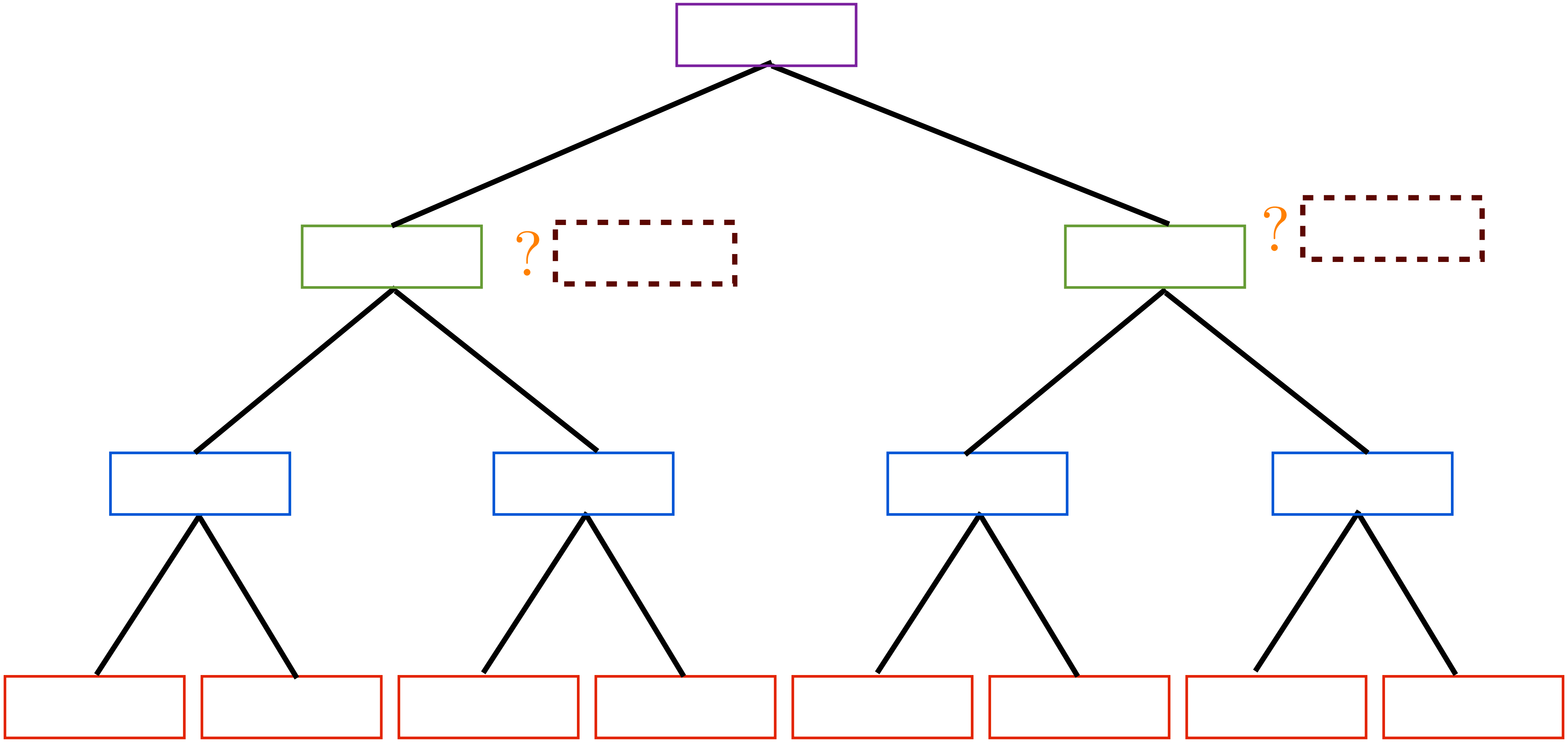


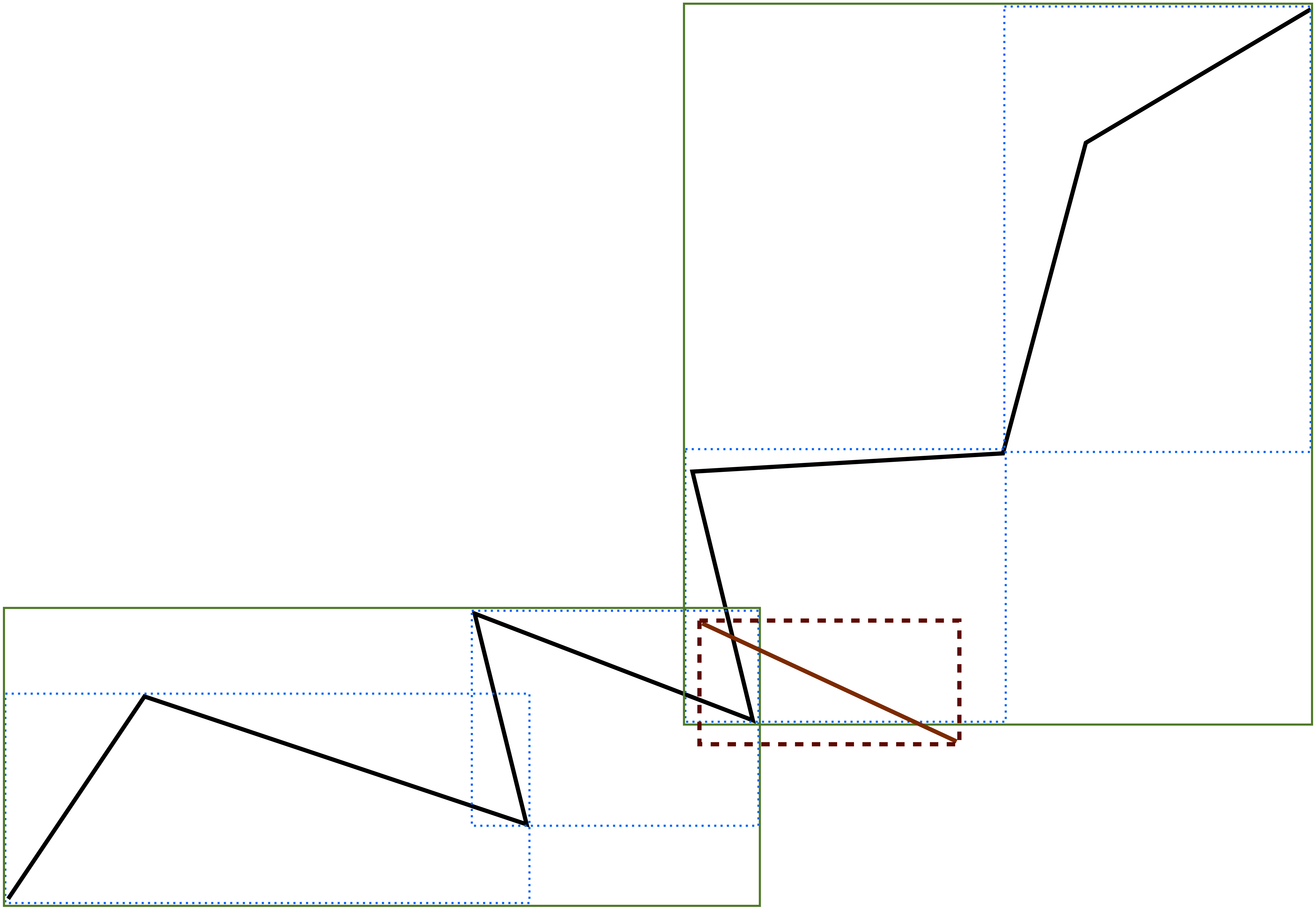


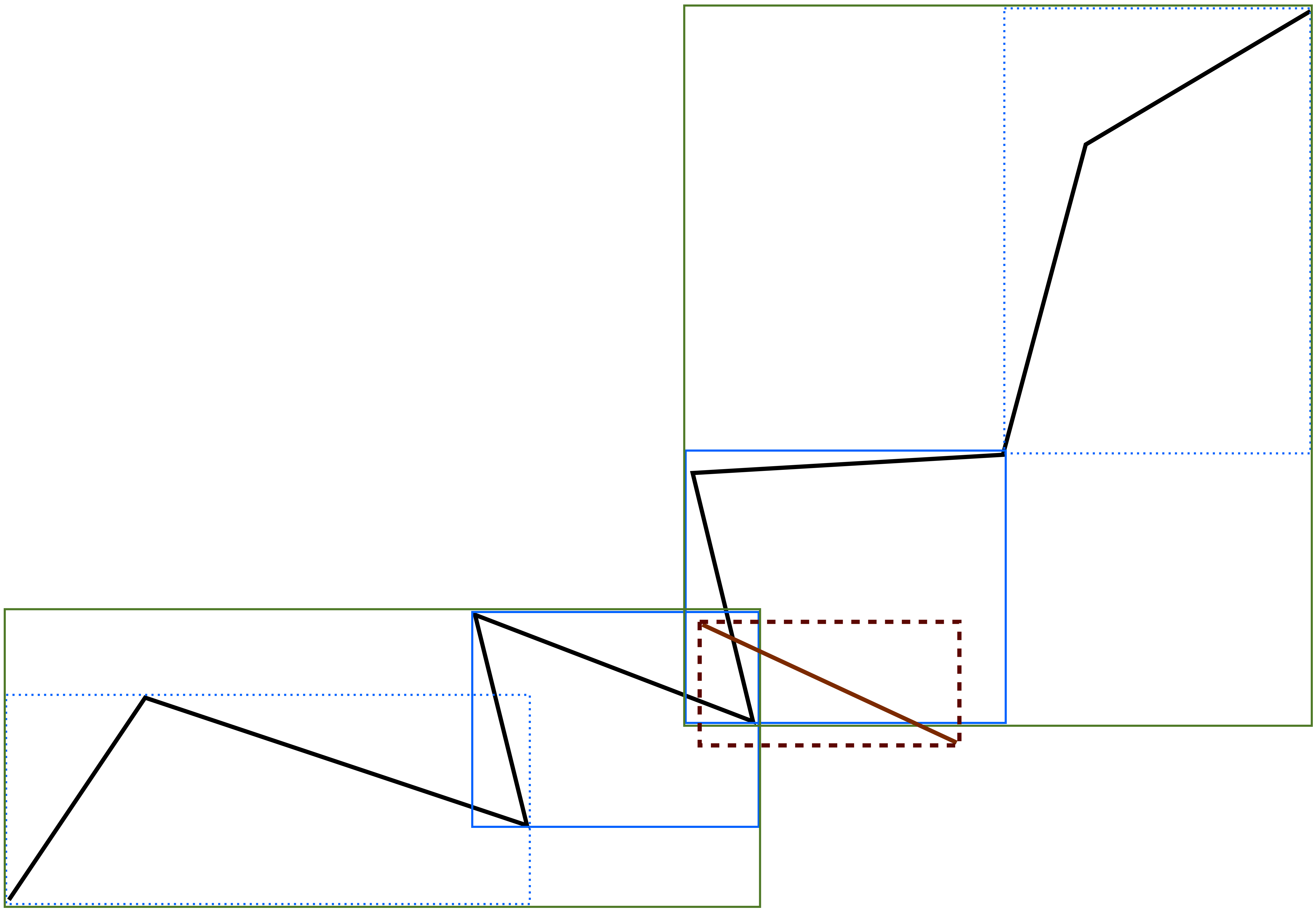


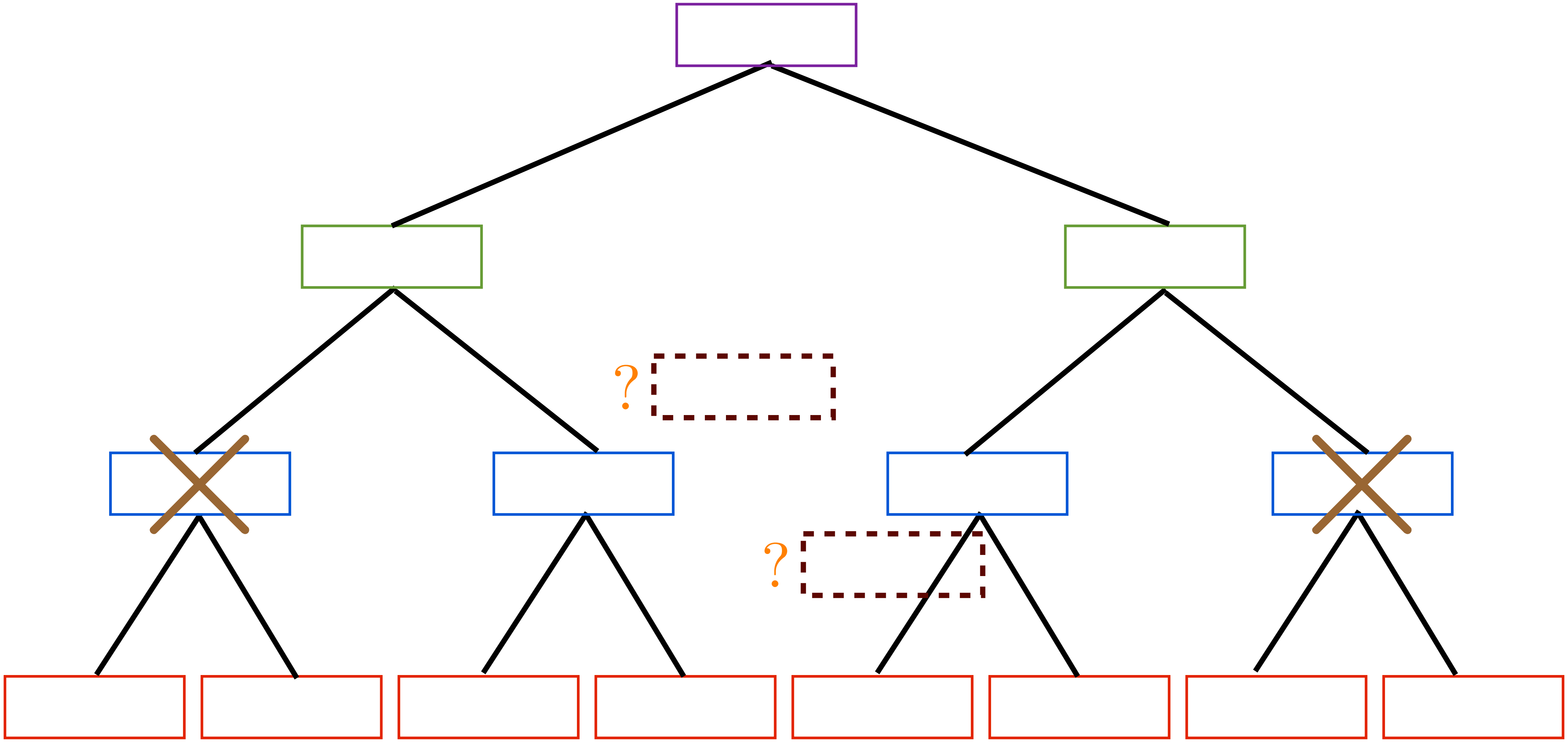


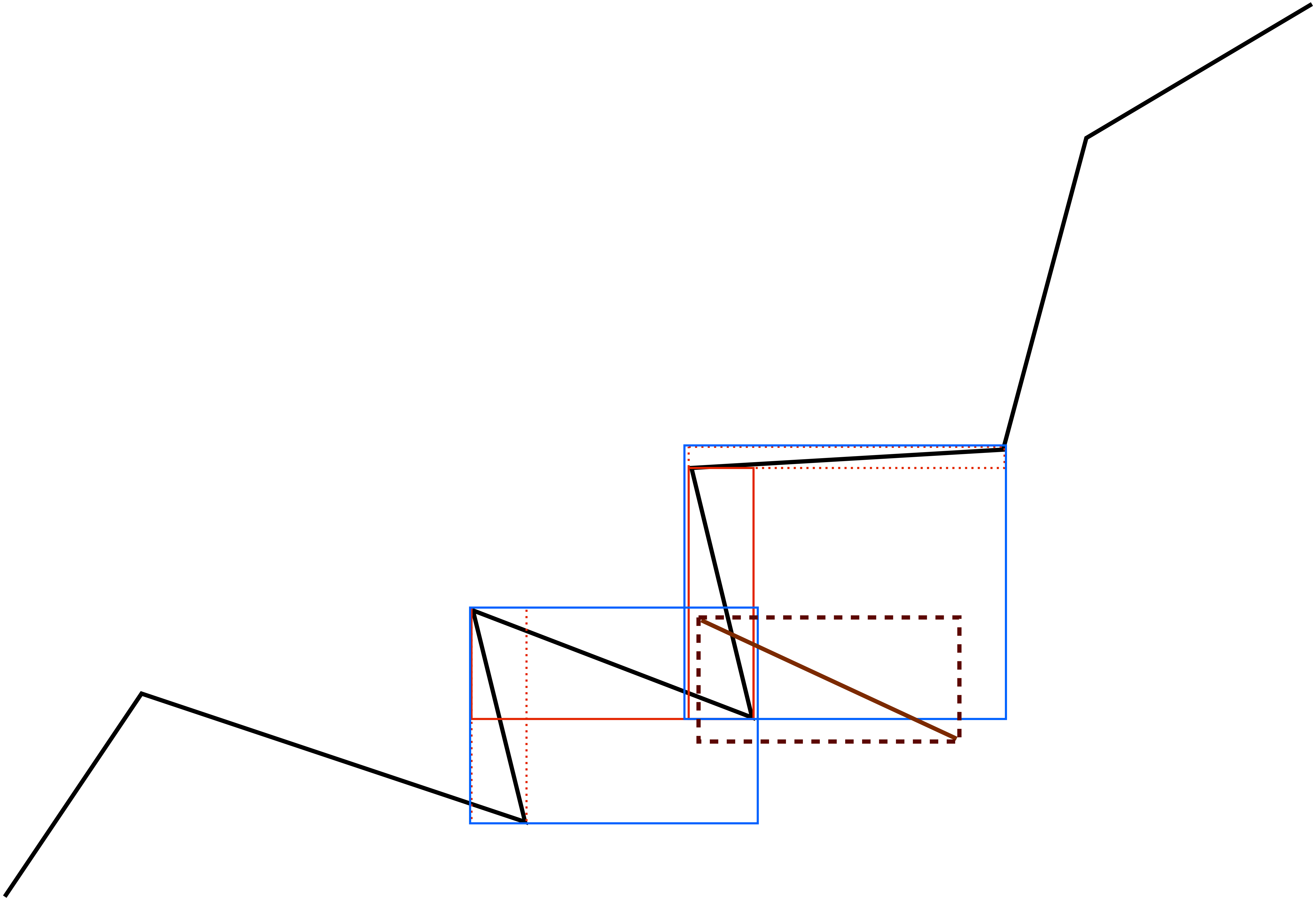


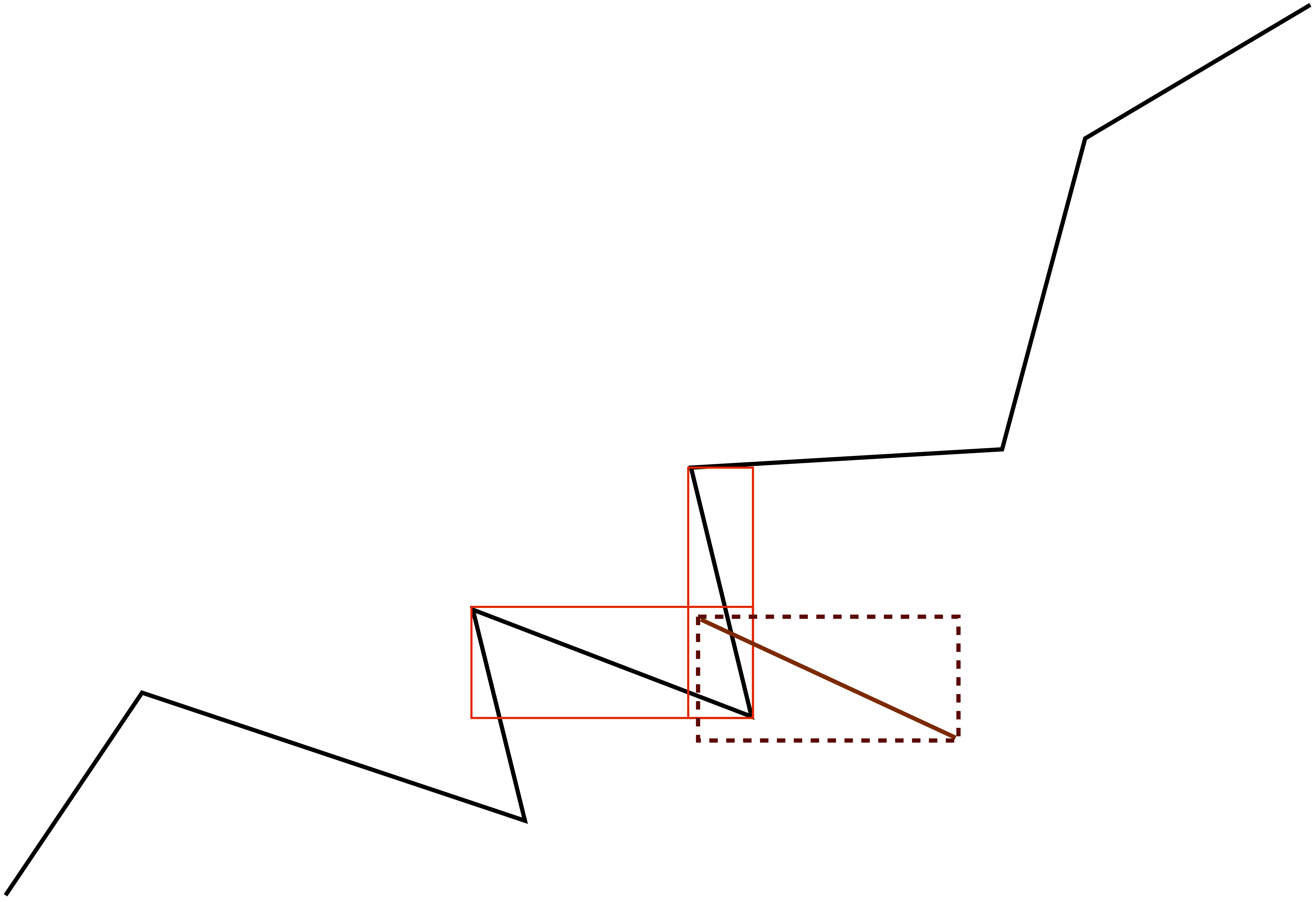


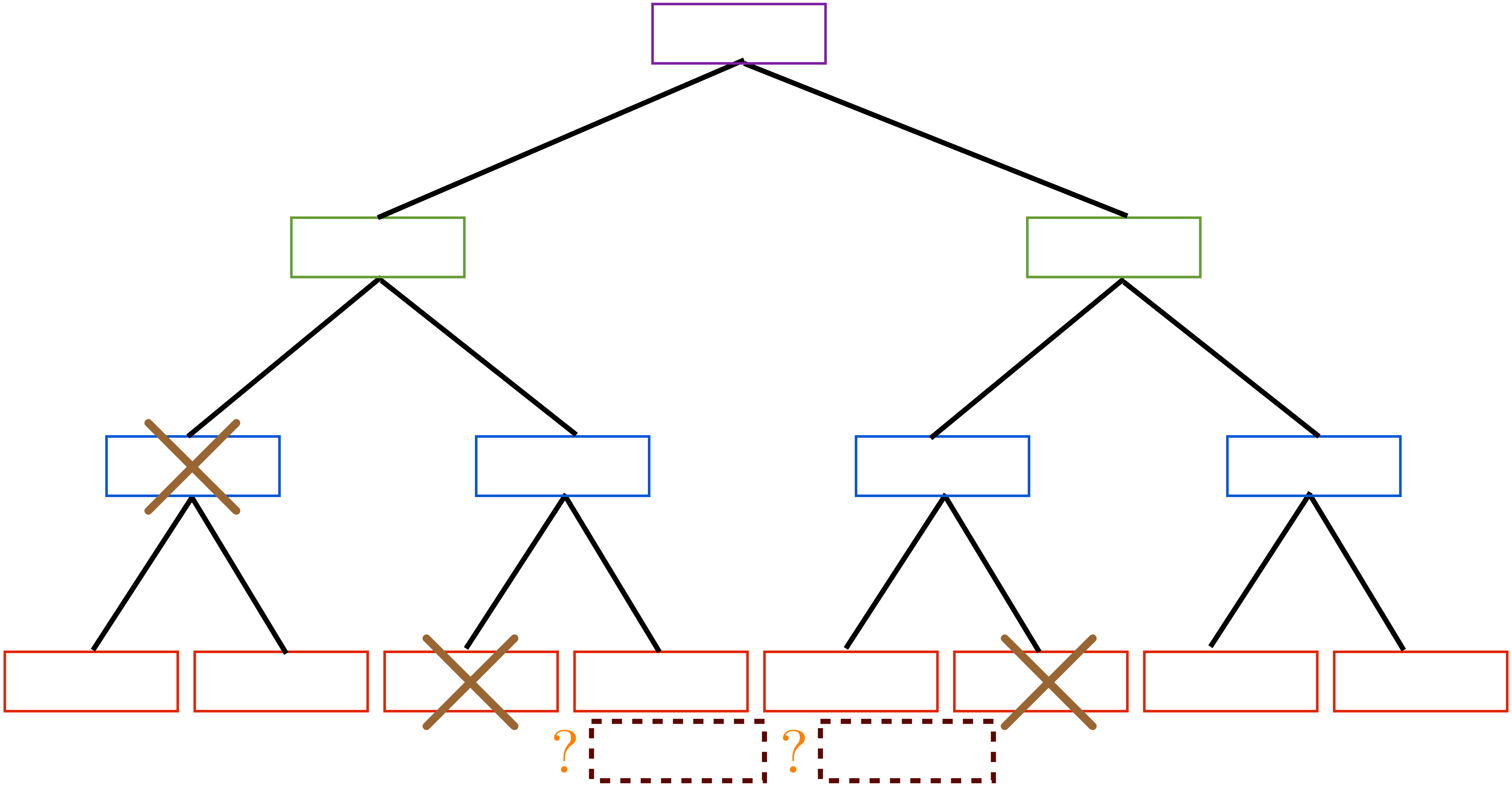












Collision detection (for simulated objects)

- Popular approach : Using axis-aligned bounding box (AABB) queries to accelerate collision detection
  - Prunes away most of the *"faraway"* collisions
  - Cost to check one primitive, against a box B-tree hierarchy with k leaves : O(logk) in the best case
    - Cost will increase if the box hierarchy is not optimally constructed (i.e. if we chose to merge faraway boxes)
    - Quality of hierarchy will degrade as object moves : May choose to re-build the hierarchy from scratch every few time steps
    - KD-Tree or Quad-/Oct-trees can be used to generate box hierarchies