



Painter's Algorithm

© 1995-2015 Josef Pelikán & Alexander Wilkie
CGG MFF UK Praha

pepca@cgg.mff.cuni.cz

<http://cgg.mff.cuni.cz/~pepca/>



Painter's Algorithm

- ◆ **Drawing to a buffer**
 - Video RAM, raster printer with a buffer
- ◆ **Area filling**
 - Even with patterns
- ➔ **Drawing is back to front**
 - **Overdrawing** of earlier objects
- ➔ **Determines visibility**



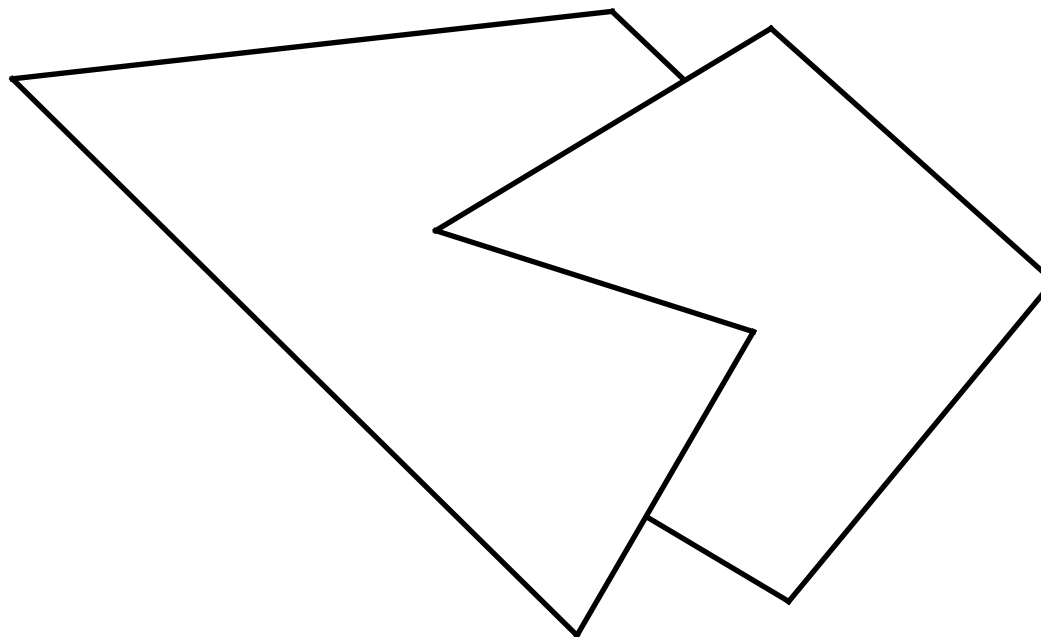
Simplified Versions

- ◆ **Explicit drawing order**
 - E.g. as function of two variables: $z = f(x,y)$
- ◆ **Depth-sort**
 - **Sorting of objects (polygons) by z coordinate (center)**
 - Works well for large amounts of small objects
 - Does not work correctly for mixtures of large and small polygons (table-top with small objects on them)



Correct Algorithm

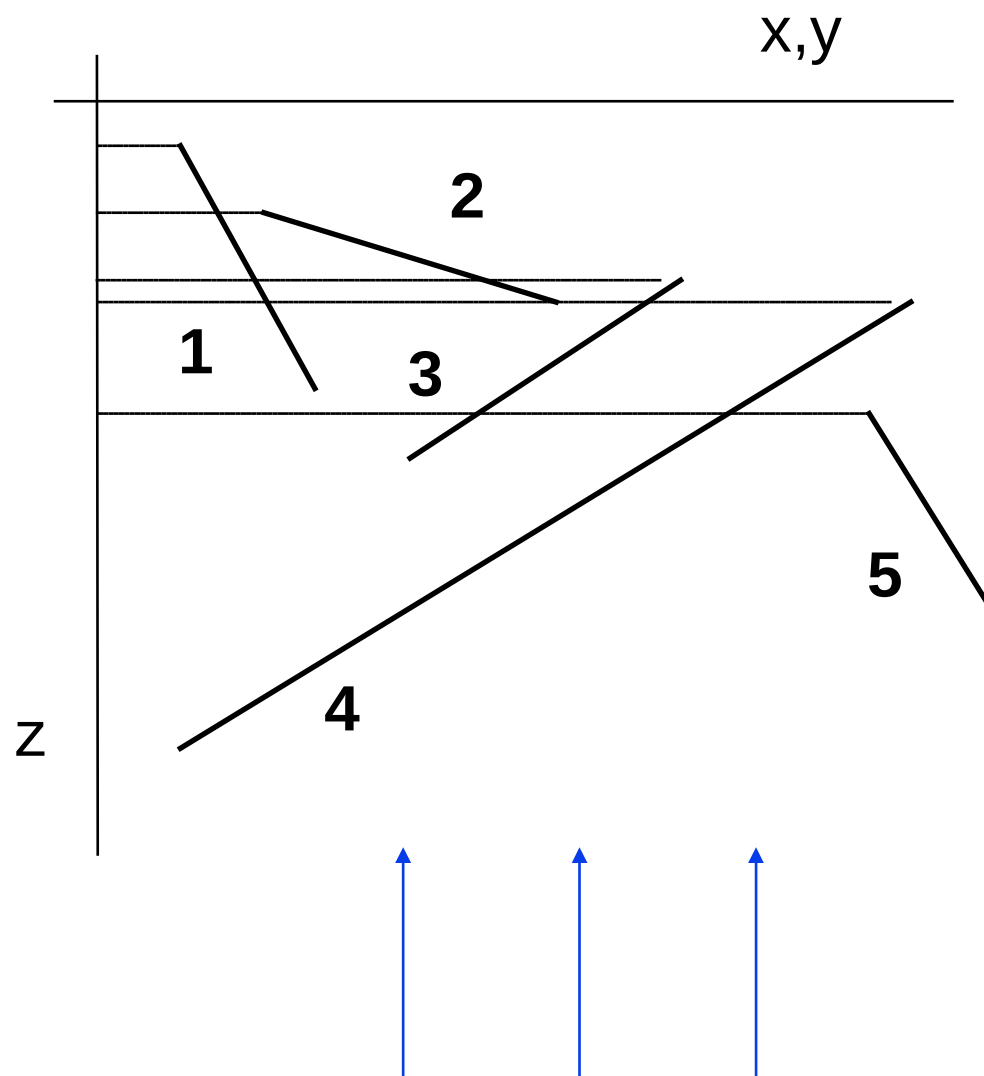
- ◆ Scene is made up of **planar geometry**
- ◆ Faces may have common points **only along the border** (no intersections)





Phase 1: Sorting

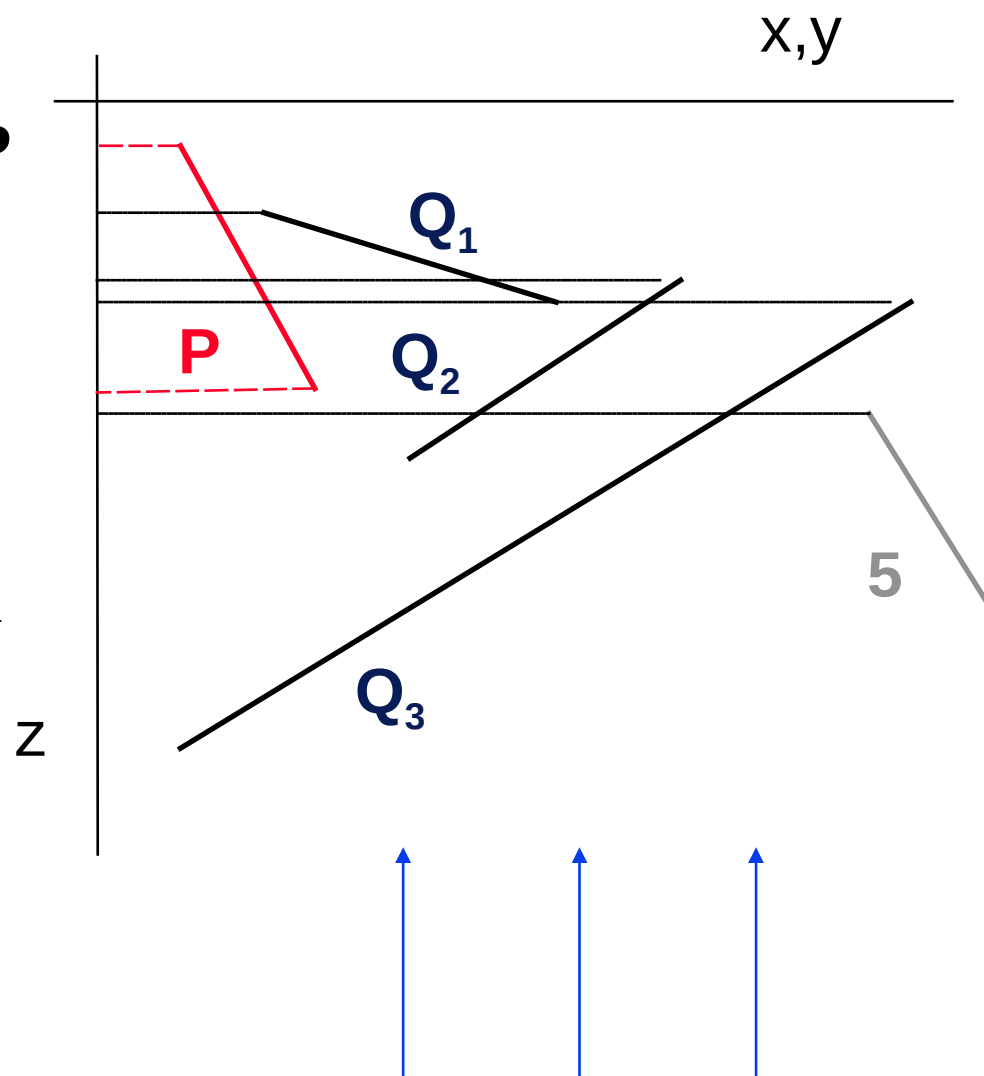
- ① Polygons are sorted by minimal **z** coordinate in **ascending** order – back to front – which generates an **input list *S***





Phase 2: Checking the Ordering

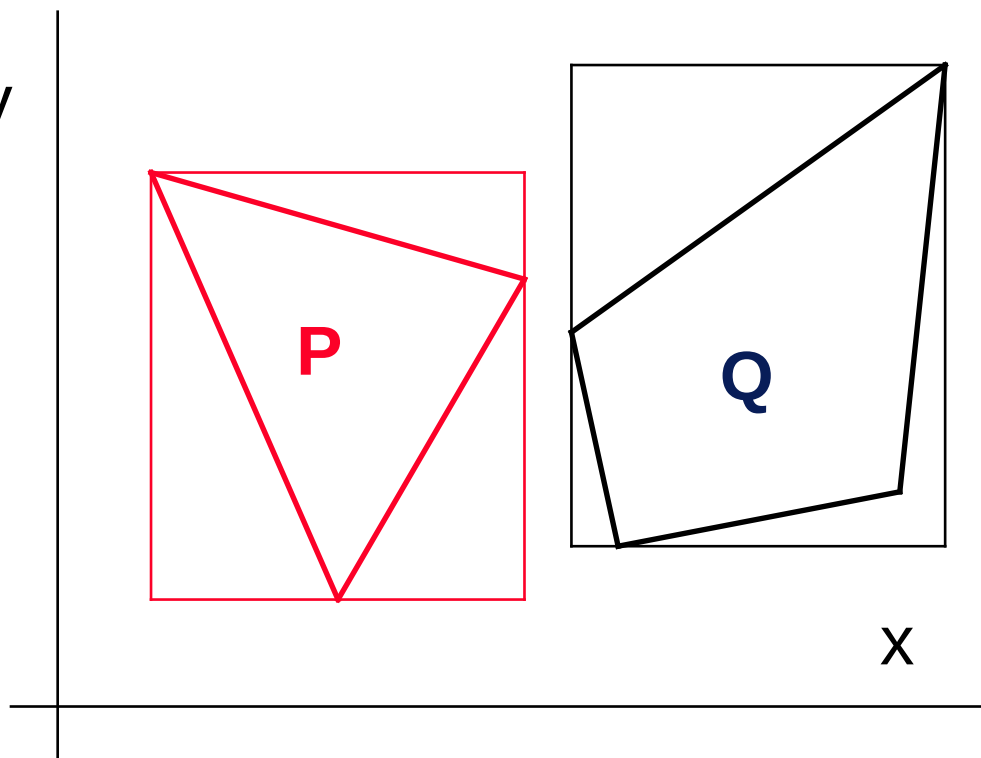
- ② From the beginning of the list S we take the polygon P – a **candidate** for drawing. We have to test other polygons against P whether there is a collision. The tested polygons are denoted Q





Phase 2A: “minimax test”

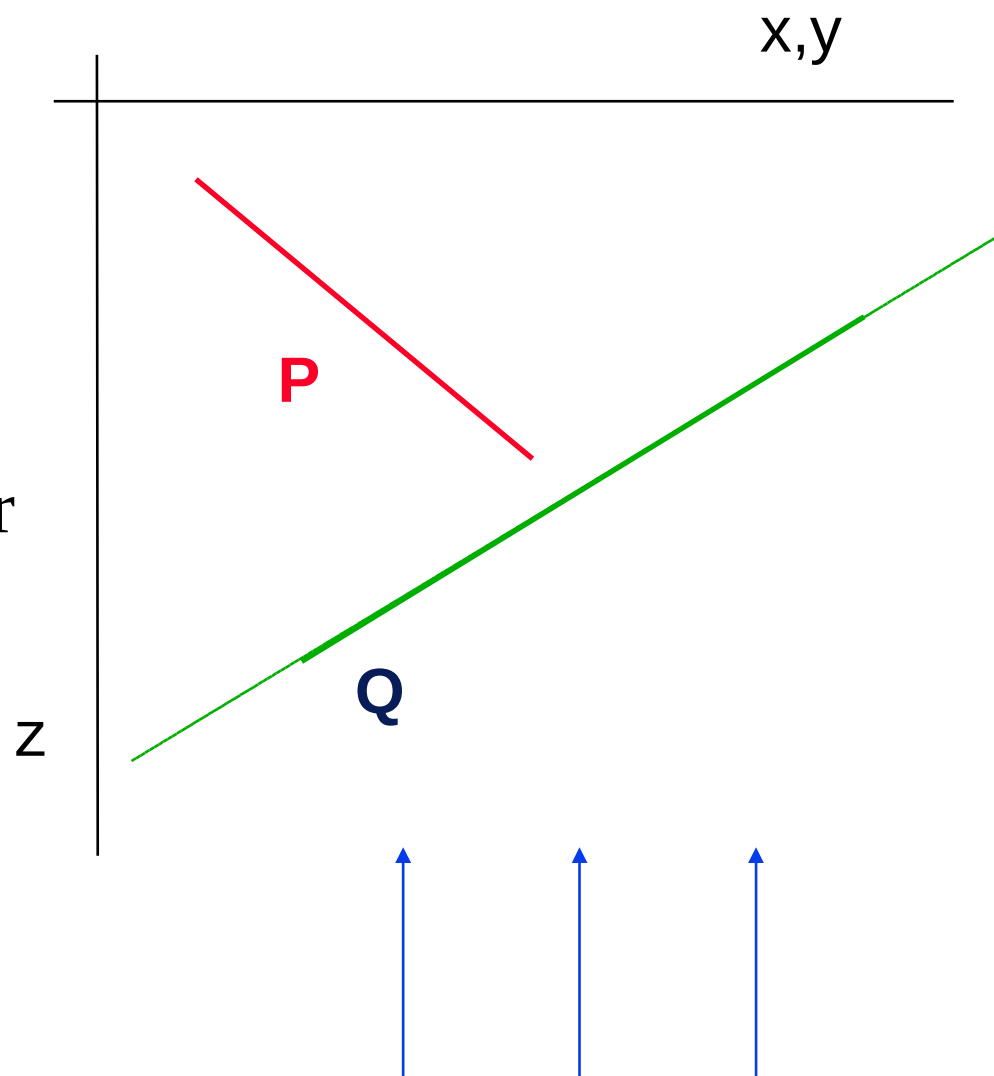
- 1 First we perform a **very easy test** – we **compare** the bounding boxes of the two polygons. If there are no overlapping points, the testing of **Q** ends. If not, we go on with further tests of **P** and **Q**.





Phase 2B: testing of P against Q

- ② We then test whether P **completely lies behind** the plane of the polygon Q . If this is the case, testing of Q ends. If not, we go on with further tests of P and Q .

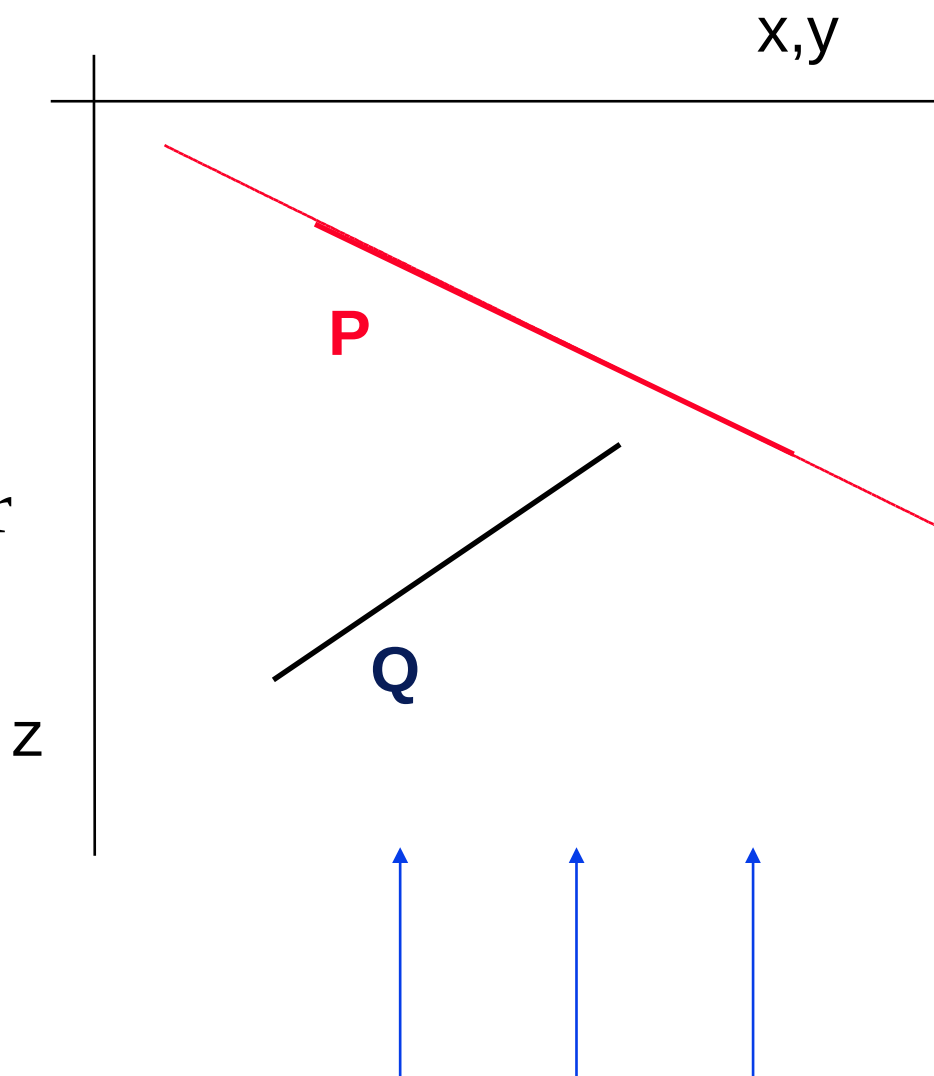


$$a \cdot x + b \cdot y + c \cdot z + d < 0$$



Phase 2C: testing of Q against P

- ② We then test whether Q **completely lies before** the plane of the polygon P . If this is the case, testing of Q ends. If not, we go on with further tests of P and Q .

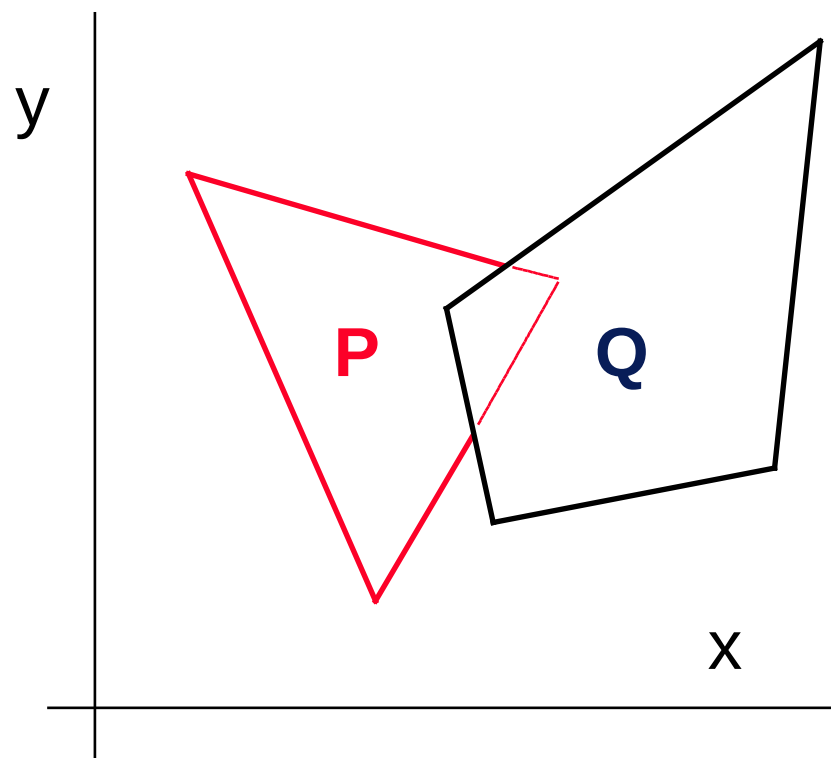


$$a \cdot x + b \cdot y + c \cdot z + d > 0$$



Phase 2D: Complete Projection

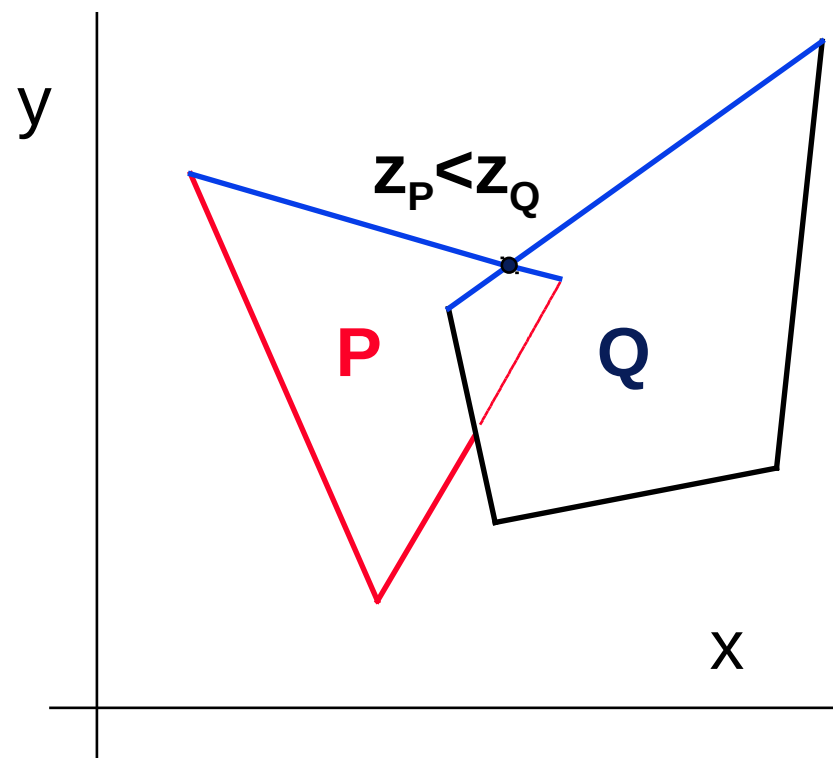
- 4 If the previous tests all failed, we have to run a **complete intersection test** of the **polygons P and Q in projection**. It is necessary to determine whether Q covers any part of P . In this case, P has to be drawn before Q !





Phase 2D: Complete Projection

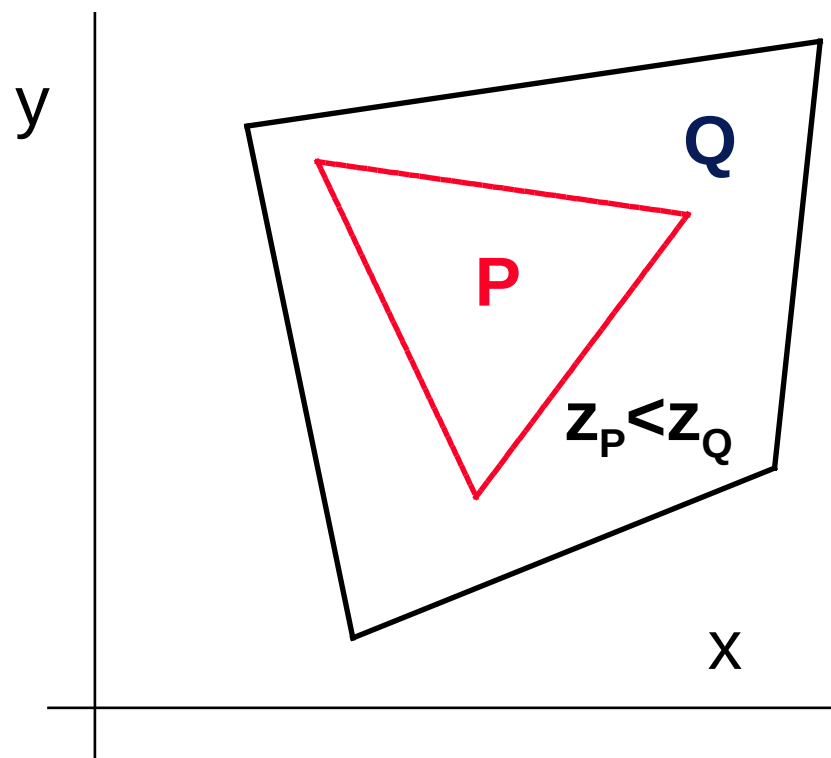
- ➔ We test **all edges** of ***P*** and ***Q*** against each other. If we find intersections, we compare the ***z*** coordinate. If any part of ***P*** is before ***Q***, the test of ***Q*** ends. In this case, it would not be possible to draw ***P*** before ***Q***!





Phase 2D: Complete Projection

- But even if no intersections of P and Q exist, we have to check whether P does not lie **completely inside** Q , or vice versa. We do this by comparing z coordinates.



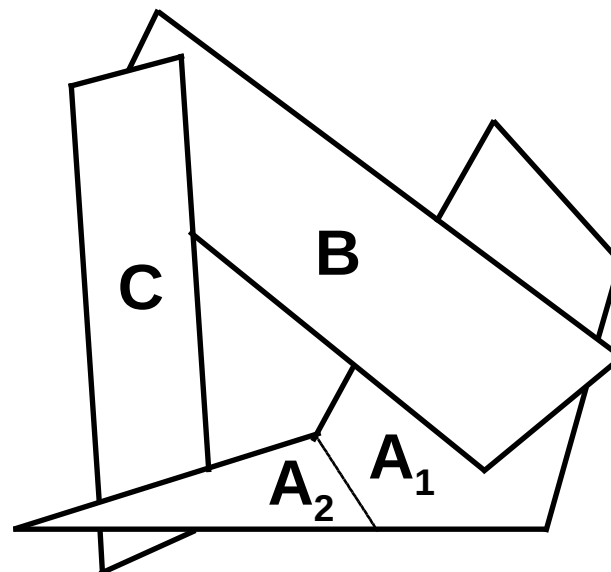
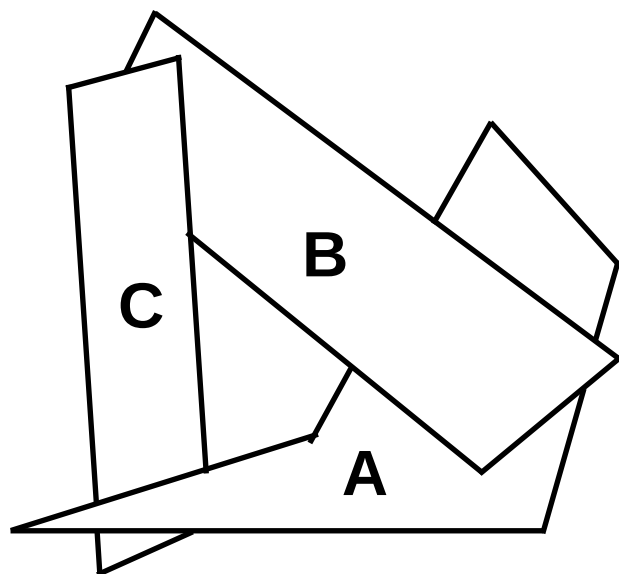


Phase 2: Re-Ordering

- ➔ If P cannot be drawn in front of Q , we try to move Q to the beginning of the list S (even before P)
 - Q will again undergo all tests of the 2nd phase (described for P)
 - Tests between Q and P have already been done, you only need to do an inverted test on B and C
- ➔ During **loops** each candidate has to be evaluated separately



Phase 2: Cycle Removal



- ➔ If any candidate is tested more than once, there is a **cycle**
- ➔ A cycle can be eliminated by **splitting** some polygons (correct order is A_1 , **B**, **C**, A_2)

End



Further information:

- **J. Foley, A. van Dam, S. Feiner, J. Hughes:**
Computer Graphics, Principles and Practice, 672-675
- **Jiří Žára a kol.:** *Počítačová grafika*, principy a algoritmy, 302-304