# Vorlesung
# Computergrafik

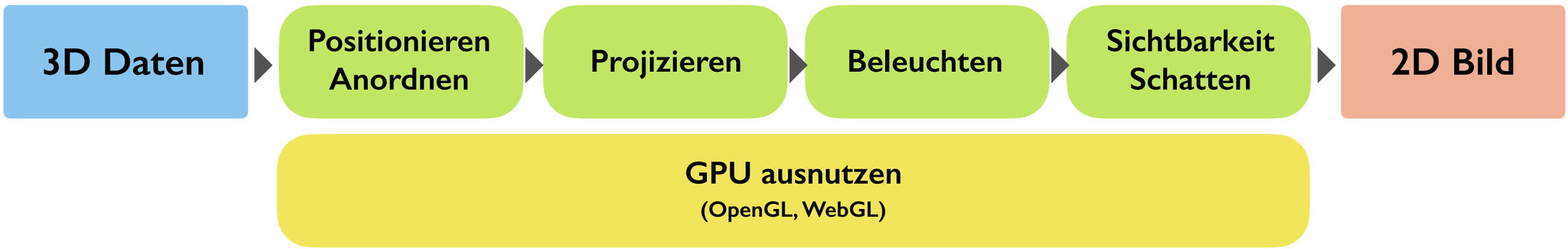
Media Informatics
**Graphics & Geometric Computing**
**Prof. Dr. M. Campen**

# Übersicht

**Tricks & Effekte**
Texturing
Environment Mapping
Displacement Mapping
Anti-Aliasing

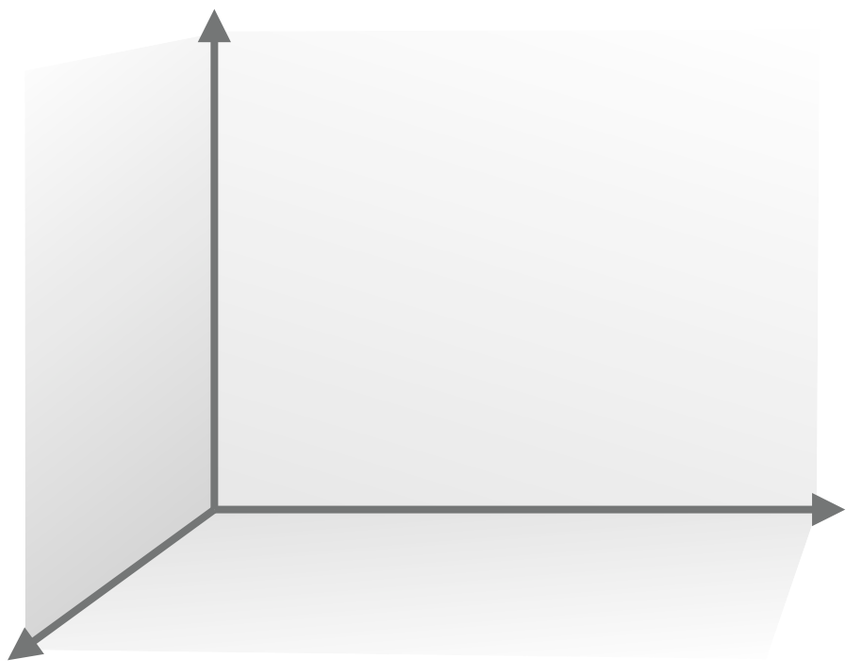
**Repräsentation**
Polygonnetze
Bézier-Flächen
Splines/NURBS
Volumendaten

**Globale Beleuchtung**
Raytracing
Radiosity

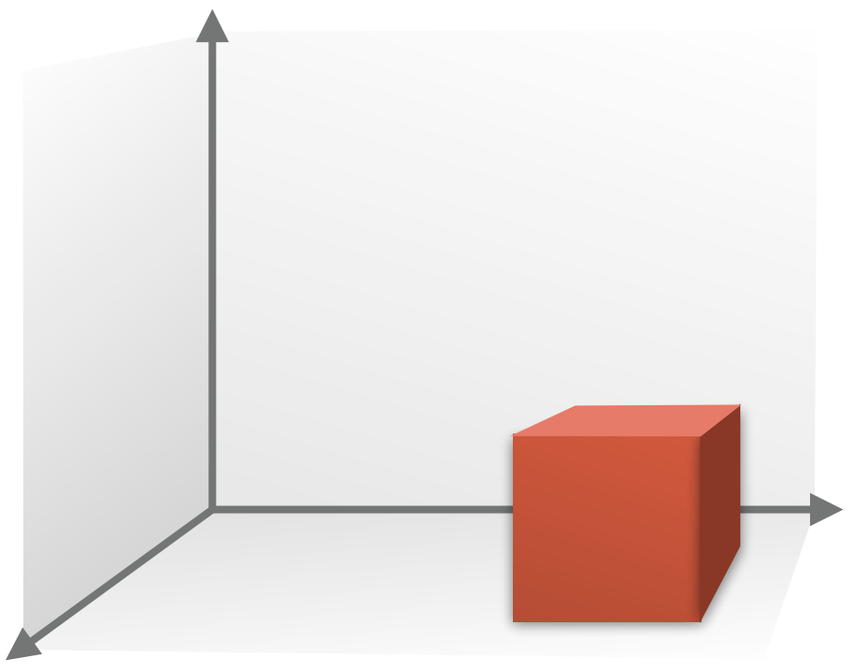


# Sichtbarkeit / Verdeckung

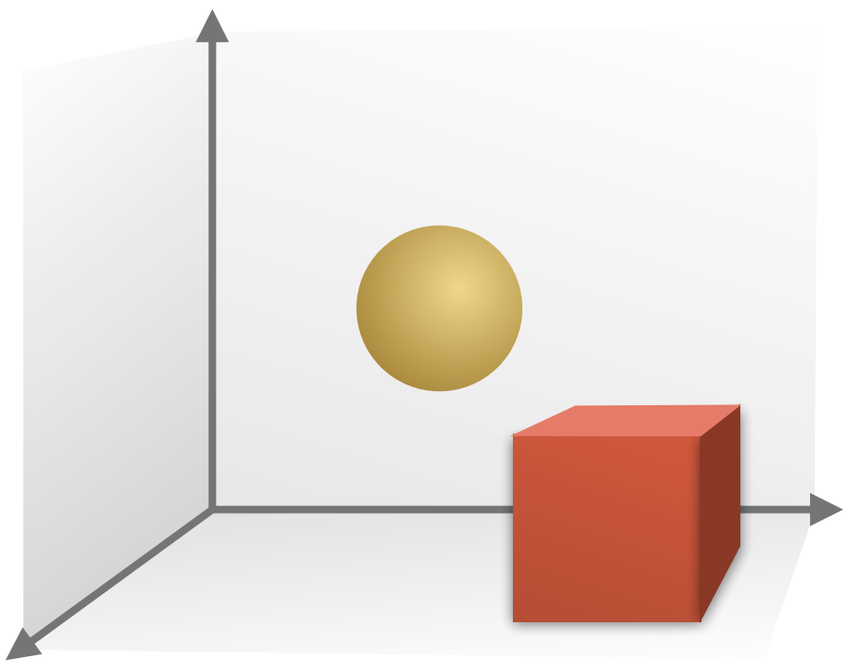
# Sichtbarkeit / Verdeckung



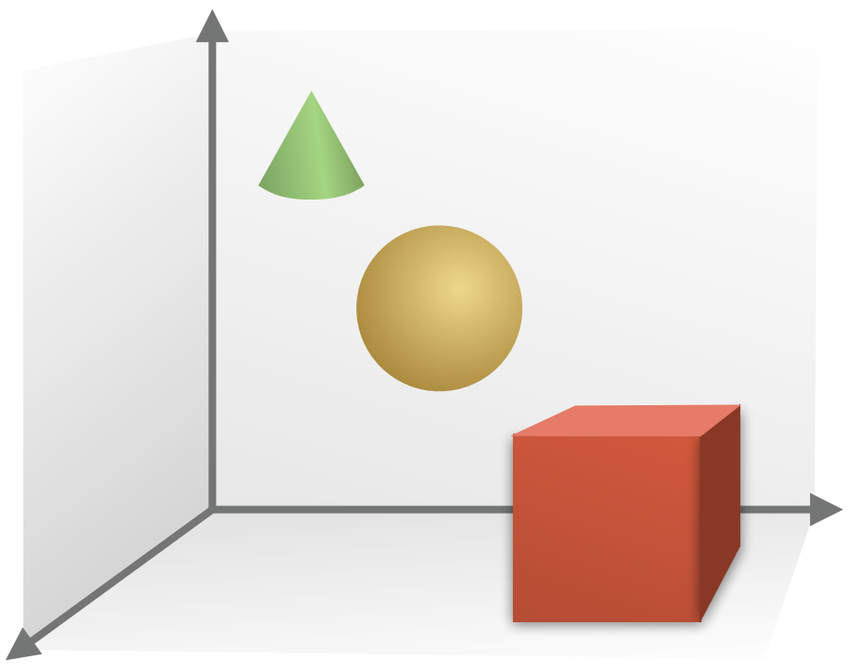
# Sichtbarkeit / Verdeckung



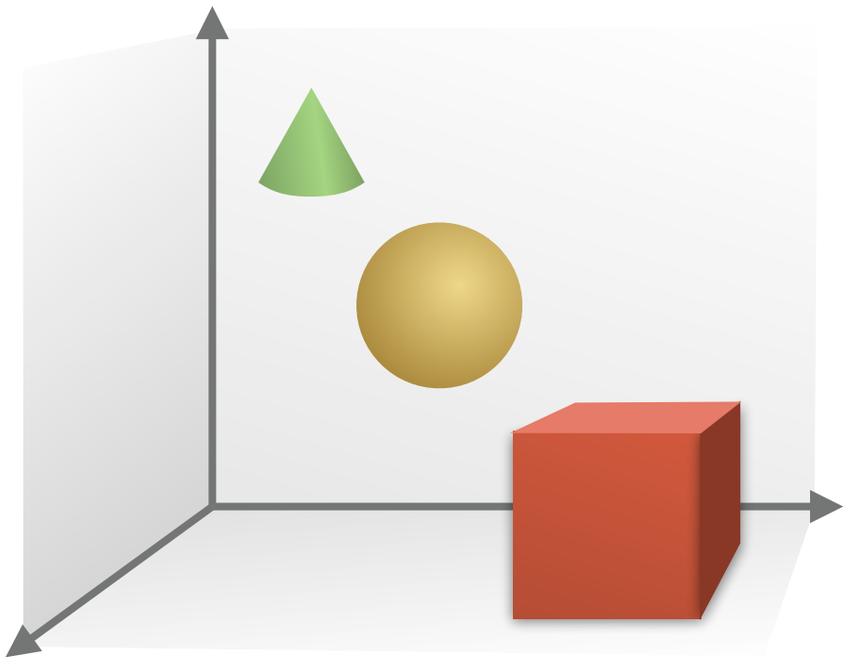
# Sichtbarkeit / Verdeckung

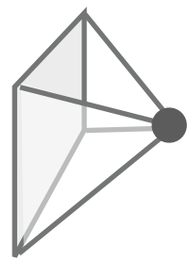


# Sichtbarkeit / Verdeckung

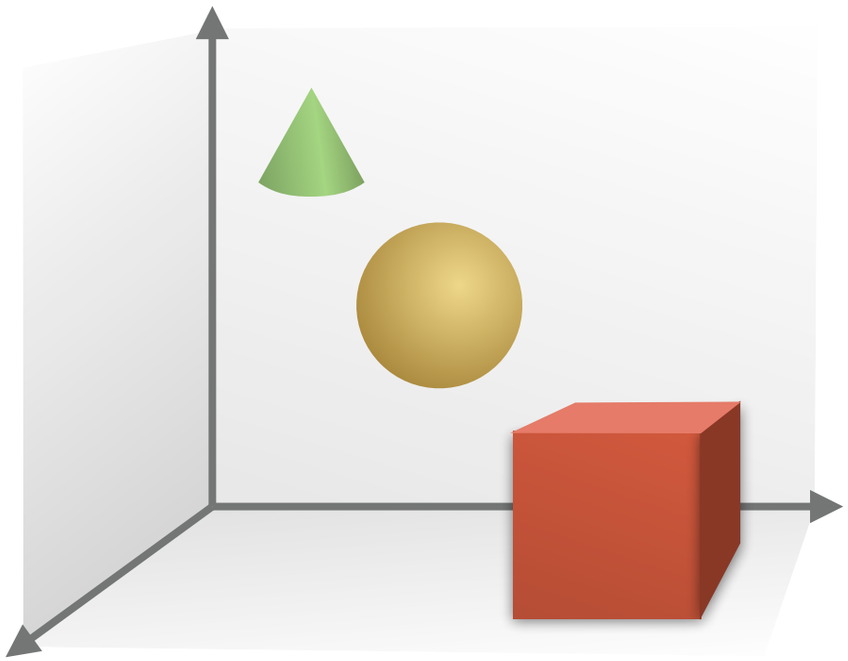


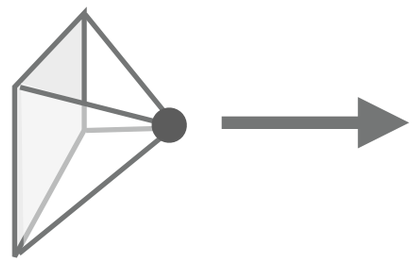
# Sichtbarkeit / Verdeckung



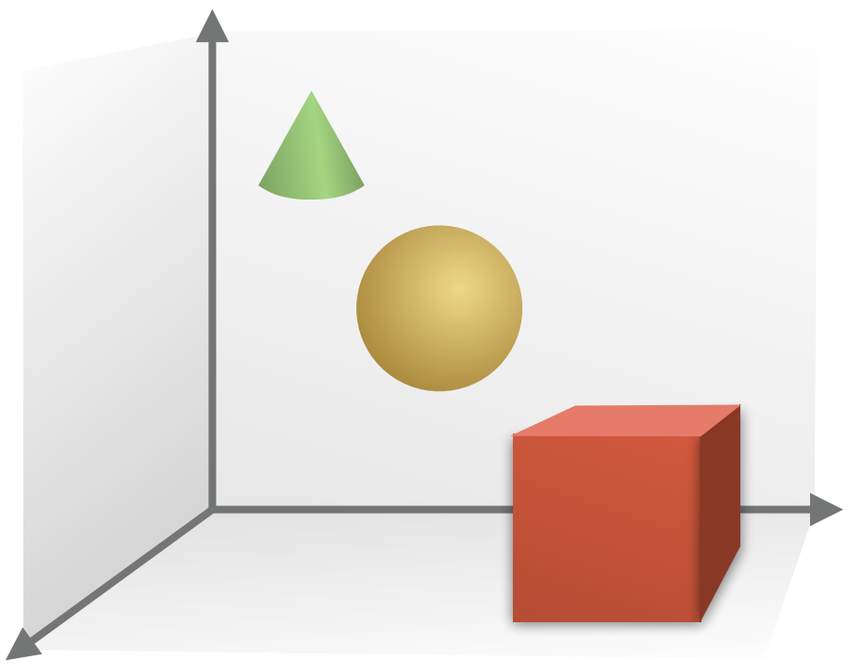


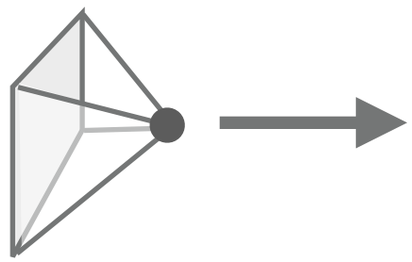
# Sichtbarkeit / Verdeckung

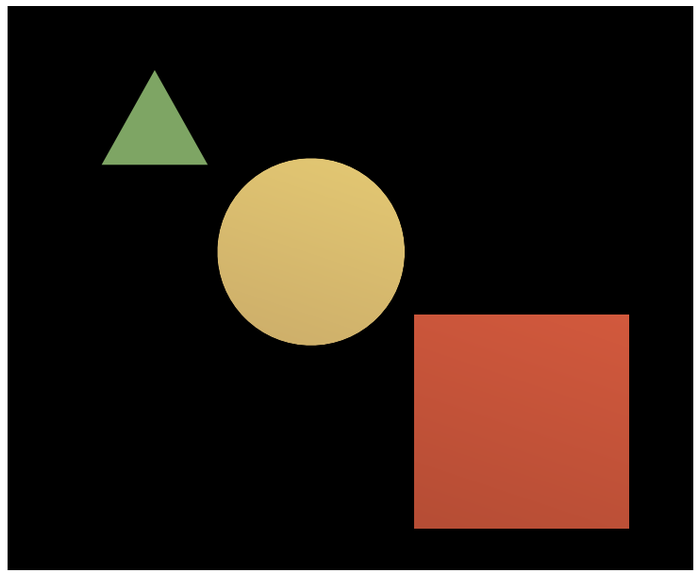




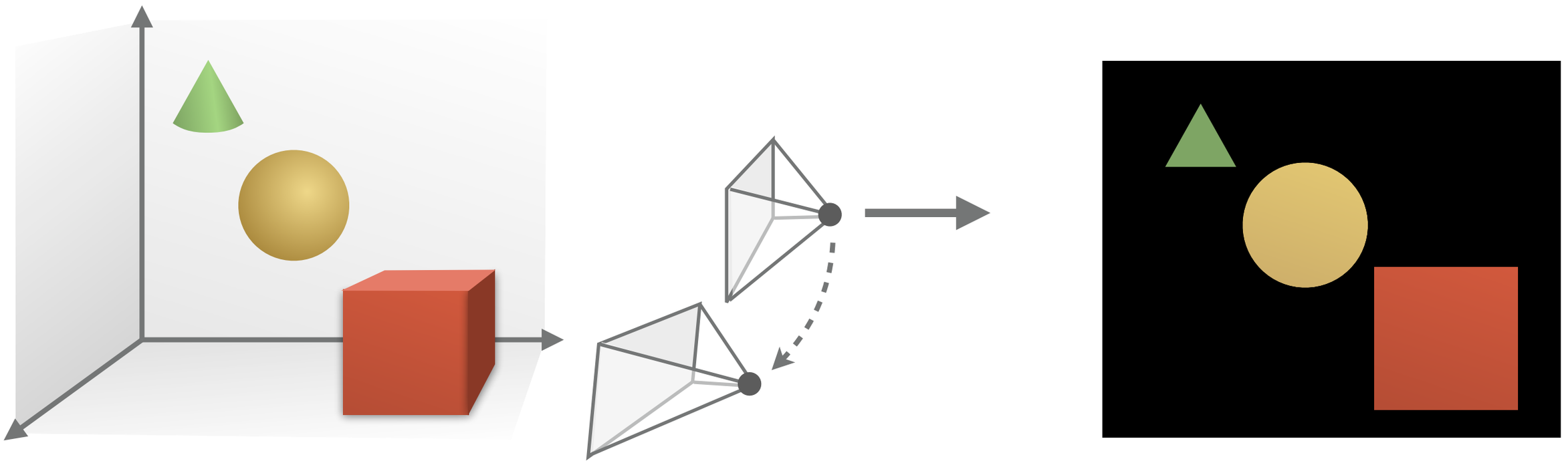
# Sichtbarkeit / Verdeckung



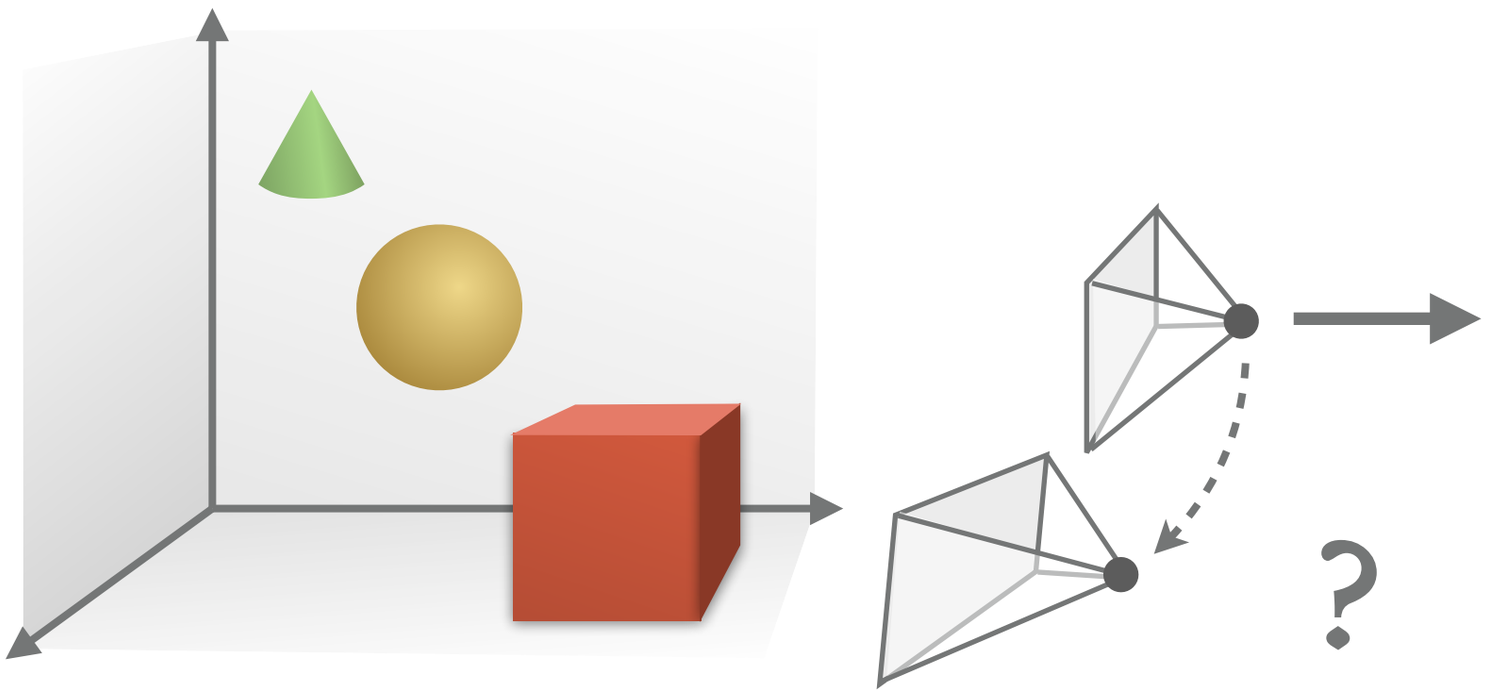




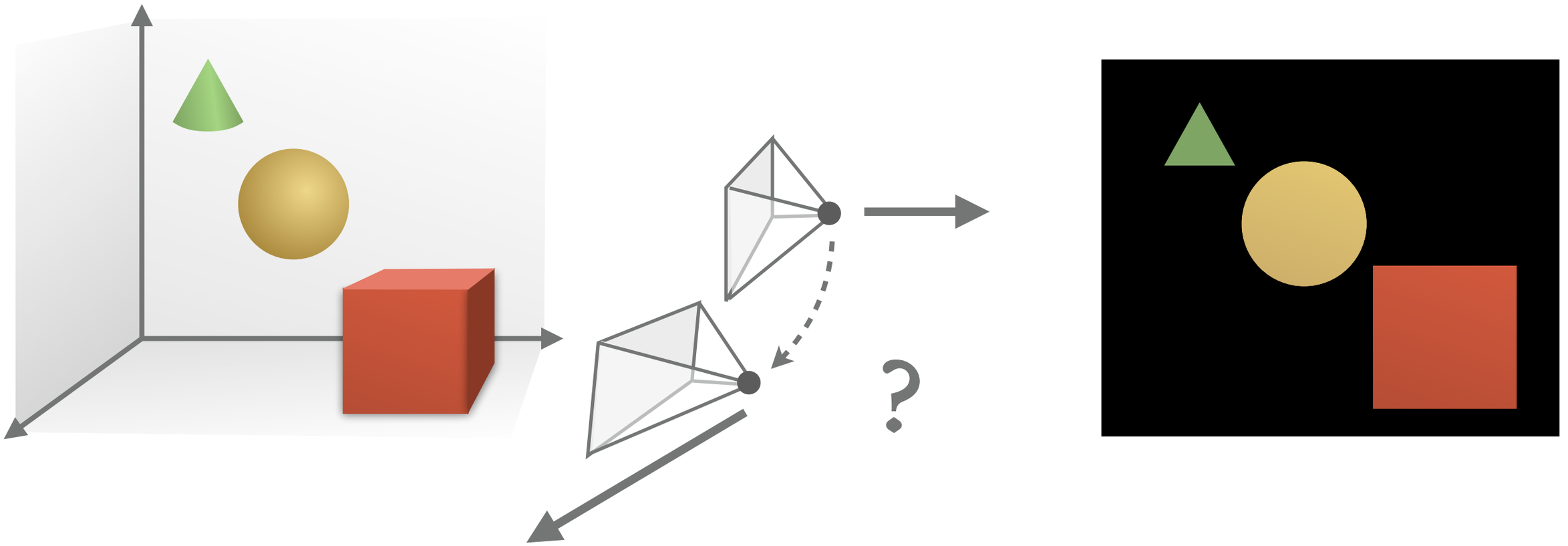
# Sichtbarkeit / Verdeckung



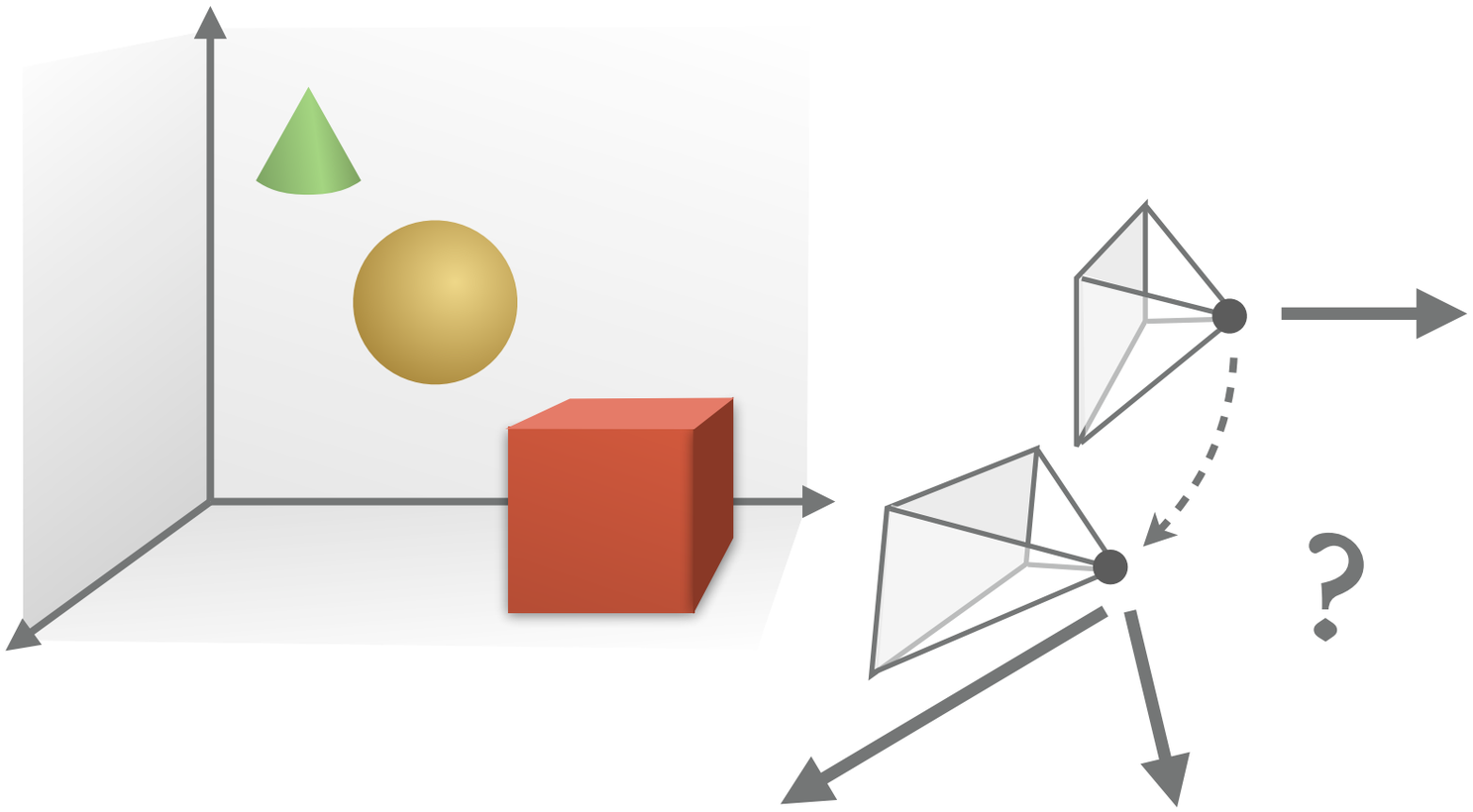
# Sichtbarkeit / Verdeckung

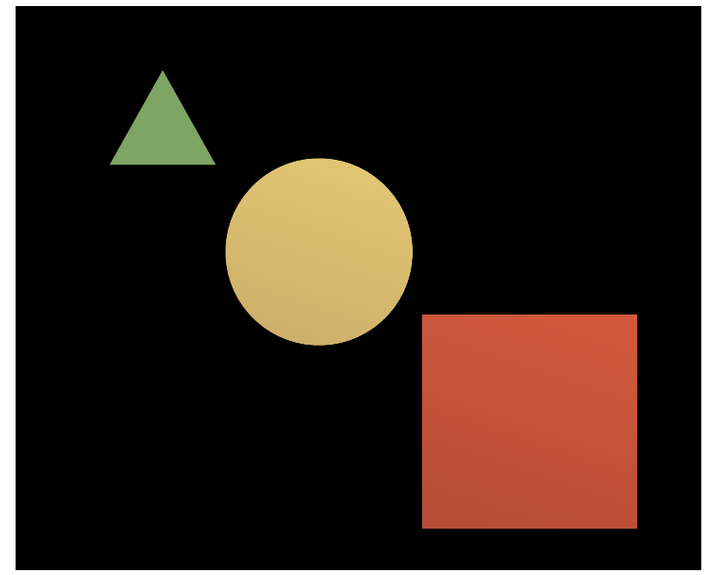


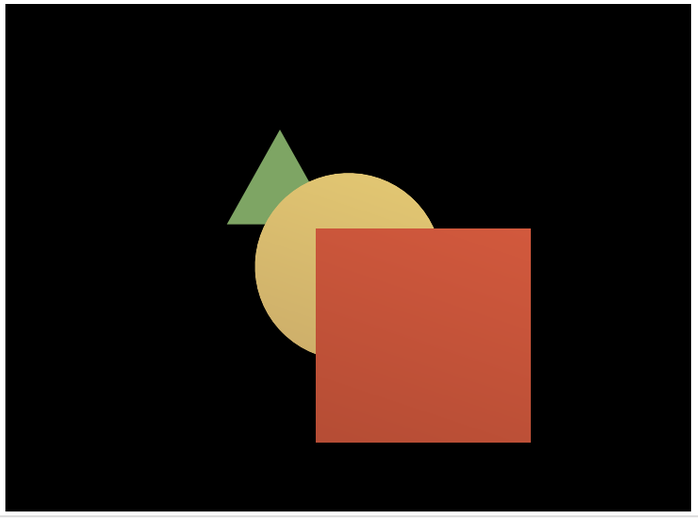
# Sichtbarkeit / Verdeckung



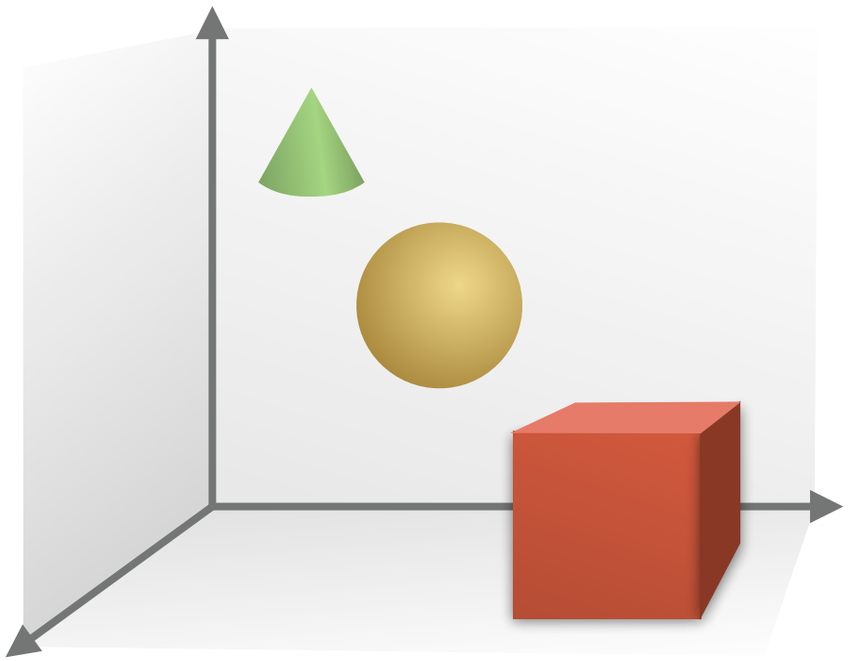

# Sichtbarkeit / Verdeckung

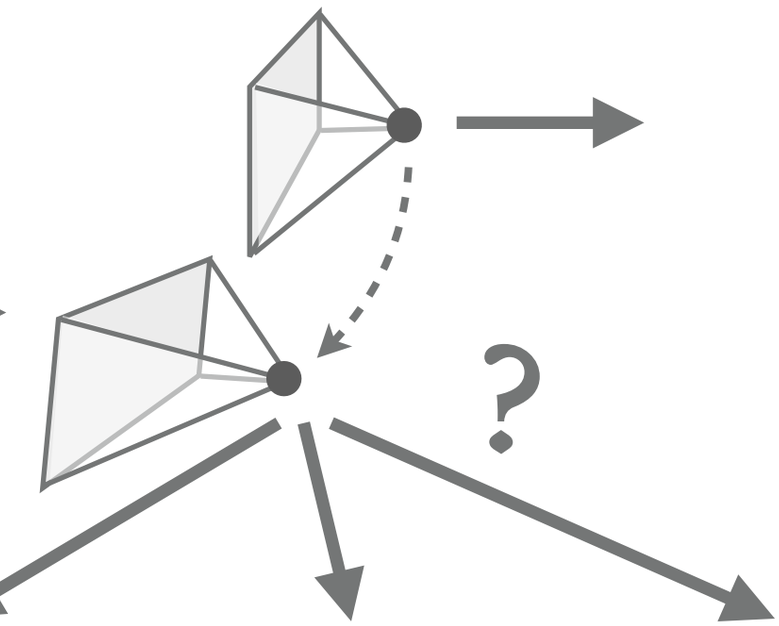


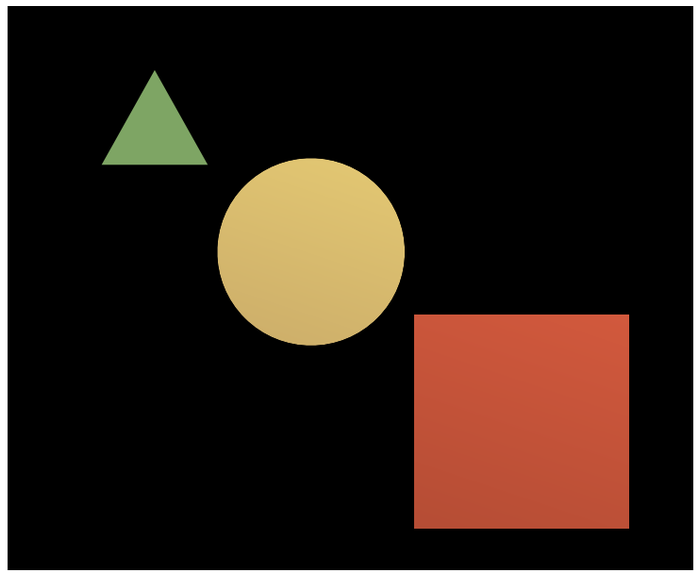


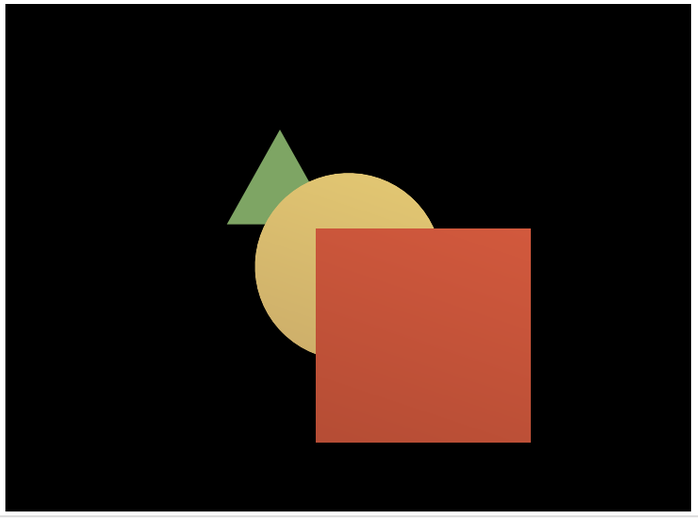


# Sichtbarkeit / Verdeckung





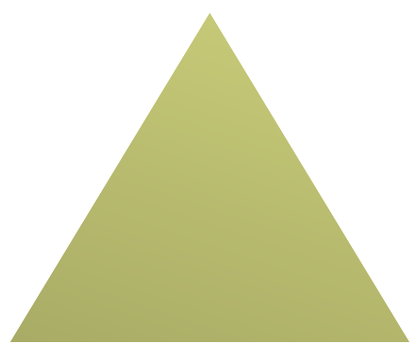


# Was Sehen Wir?

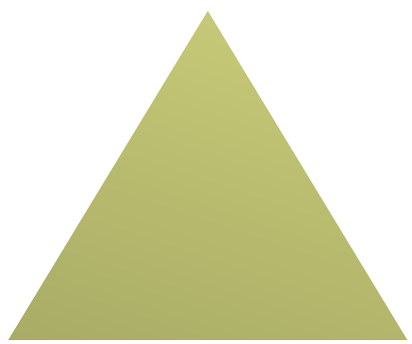
# Was Sehen Wir?

# Was Sehen Wir?

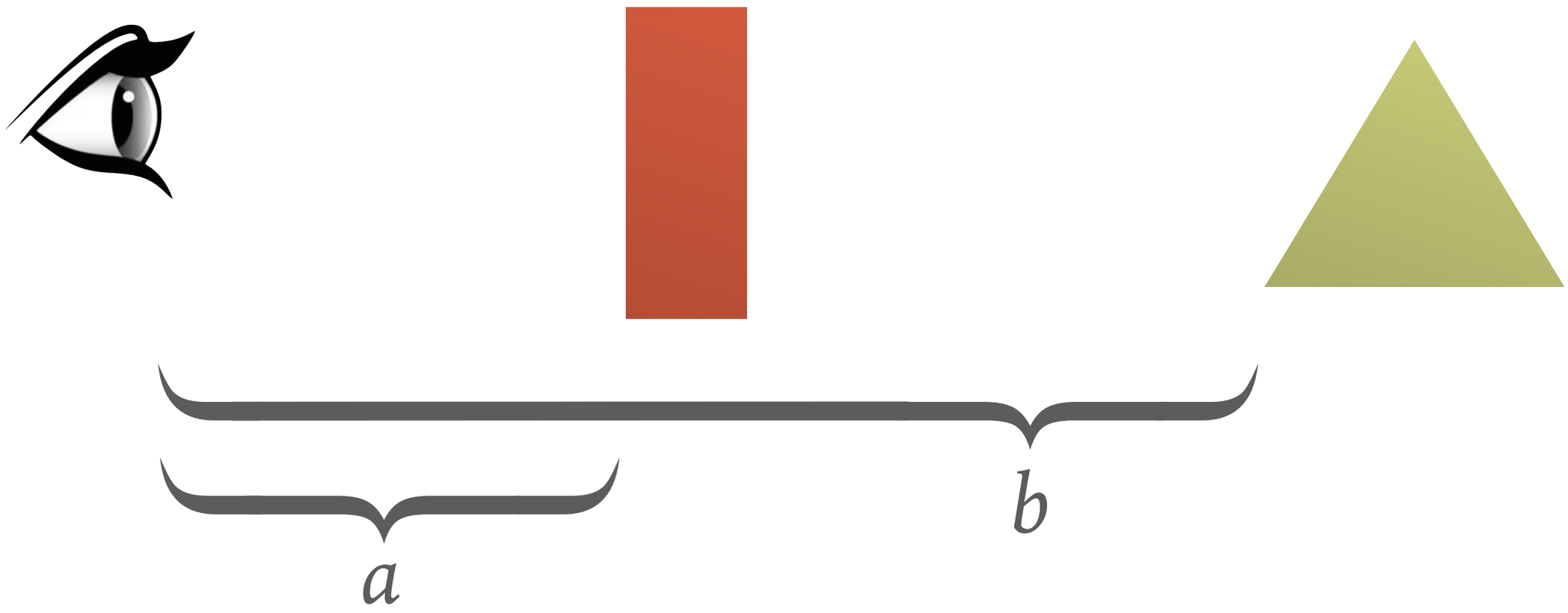
# Was Sehen Wir?



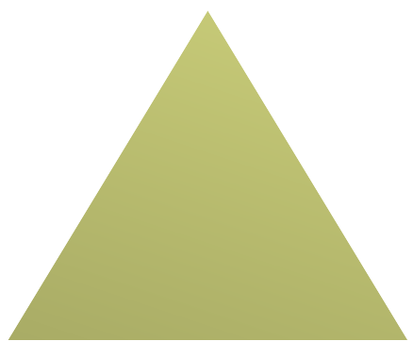
# Was Sehen Wir?



# Was Sehen Wir?

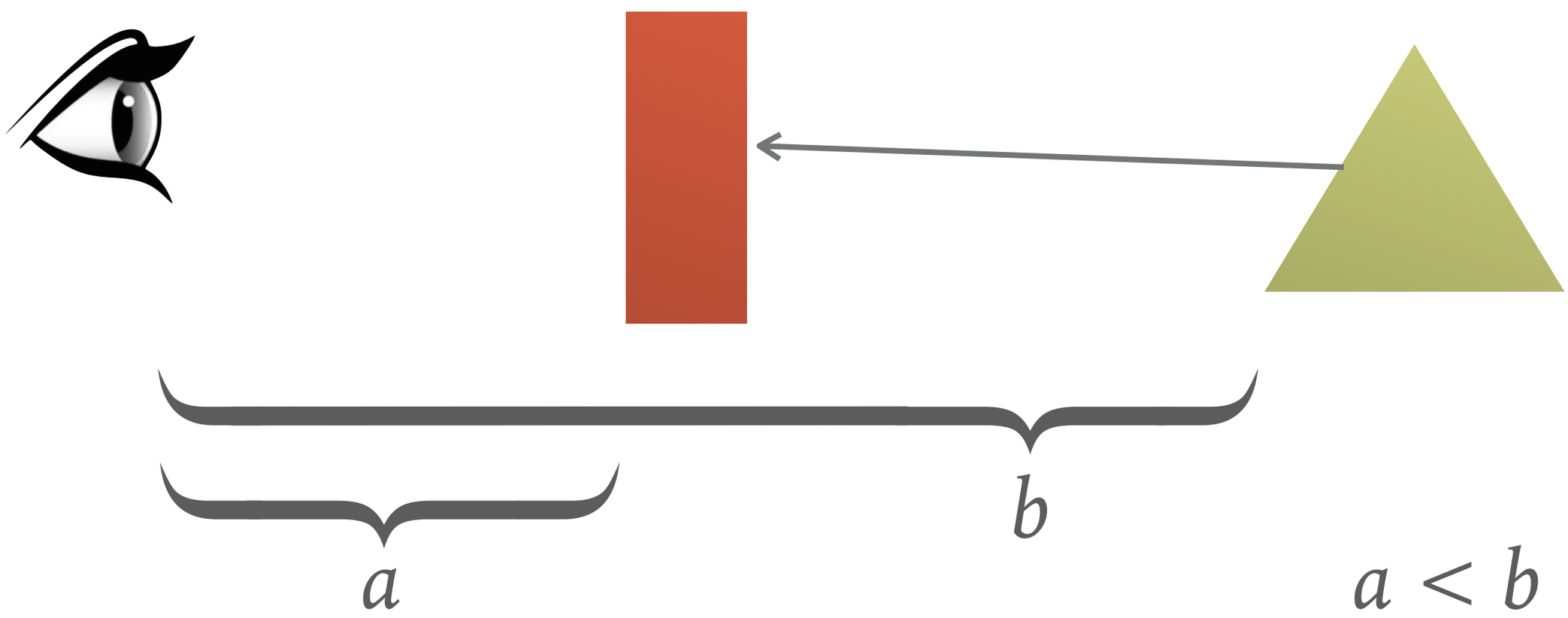


# Was Sehen Wir?



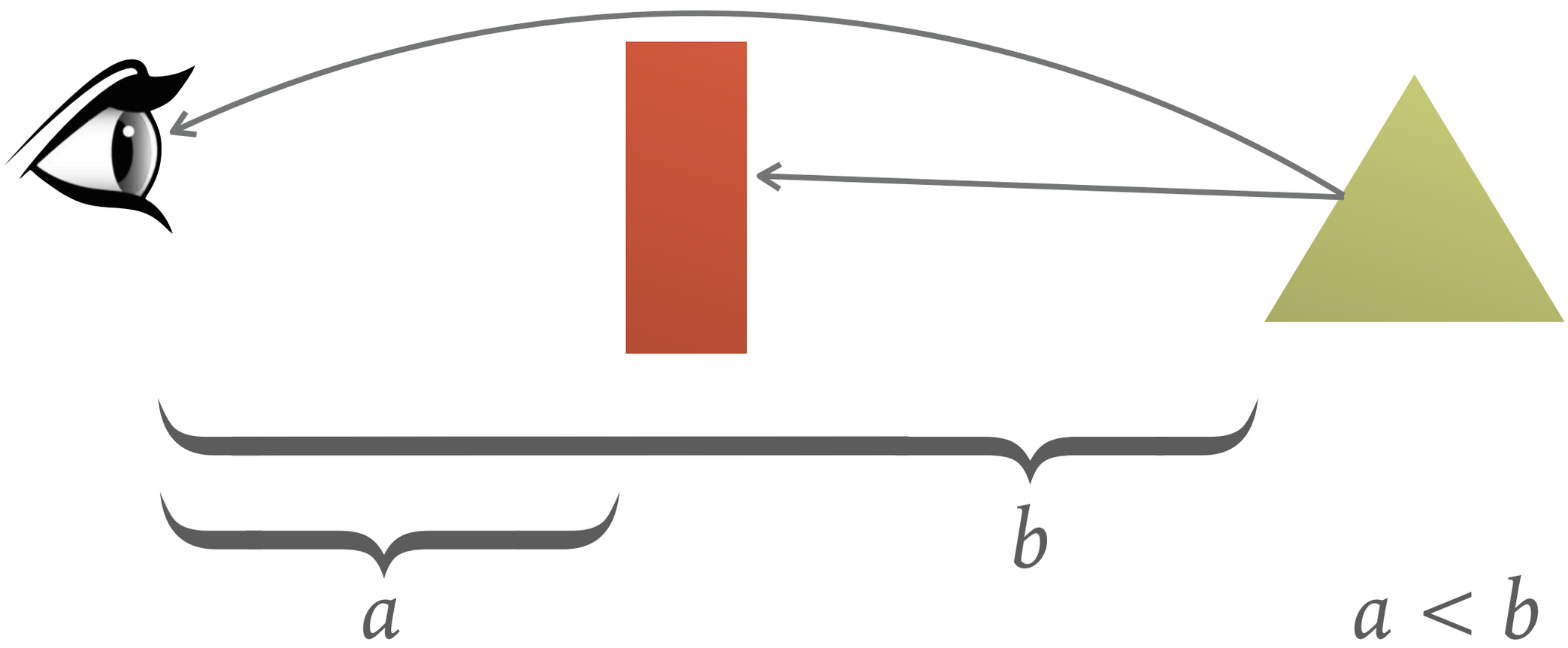

$a < b$

# Was Sehen Wir?



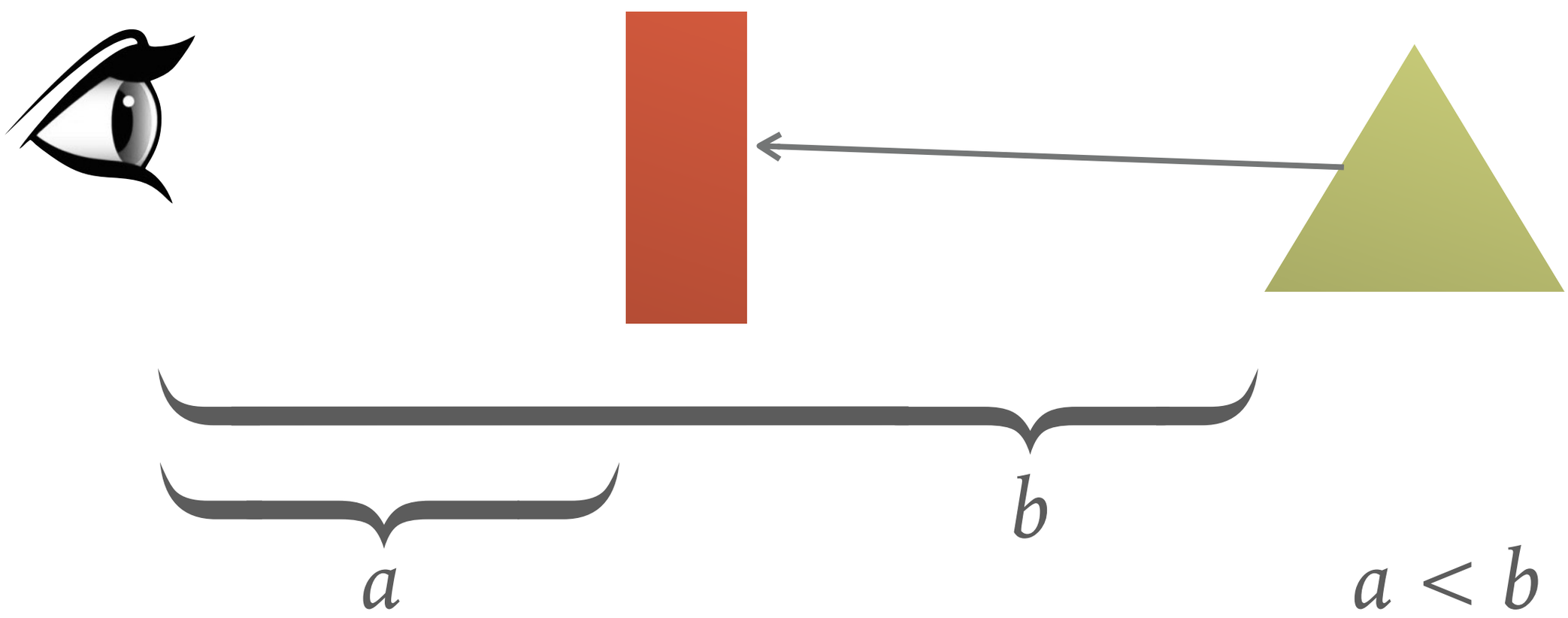

$a < b$

# Was Sehen Wir?



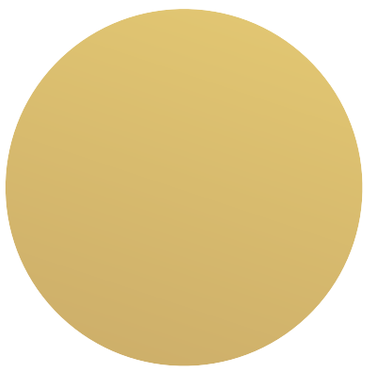

$$a < b$$

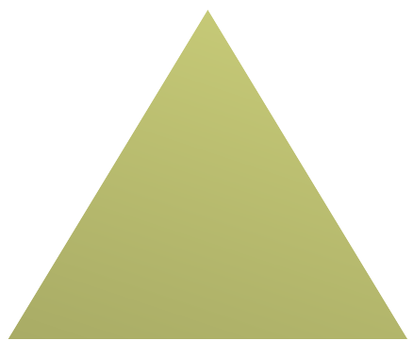
# Was Sehen Wir?




$a < b$

# Tiefen-Sortierung

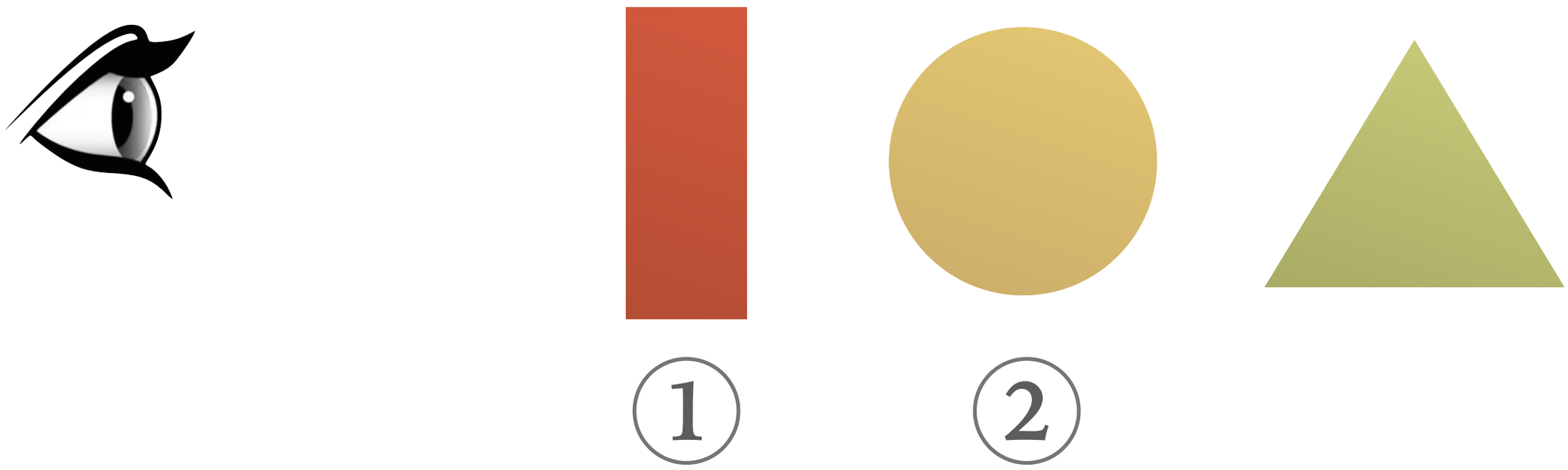




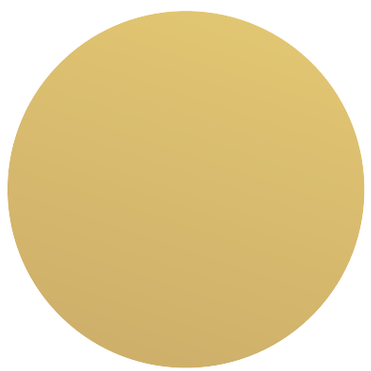
# Tiefen-Sortierung

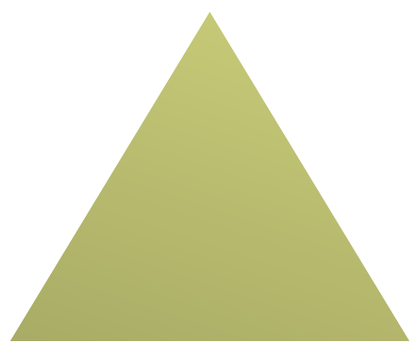
1

# Tiefen-Sortierung



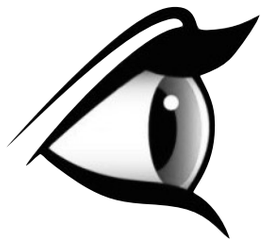
# Tiefen-Sortierung

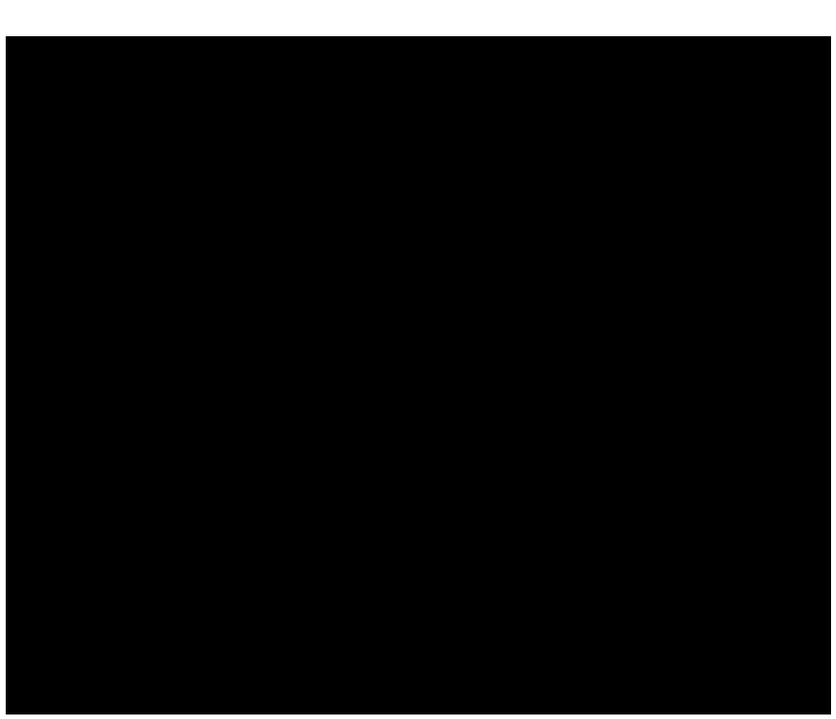


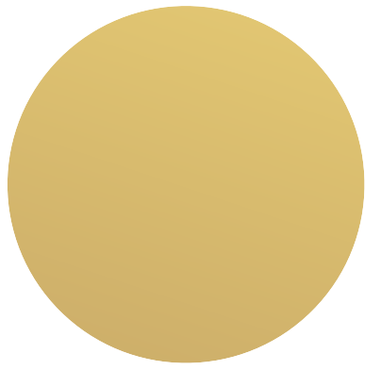


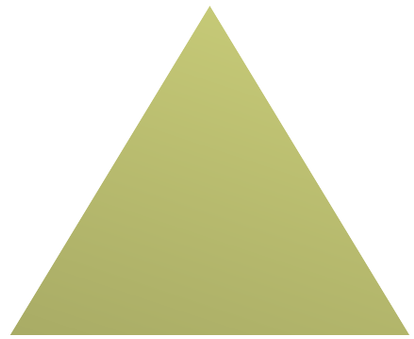
1 2 3

# Tiefen-Sortierung







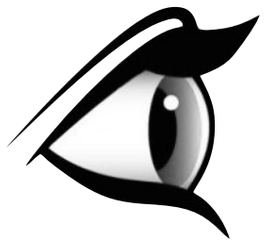


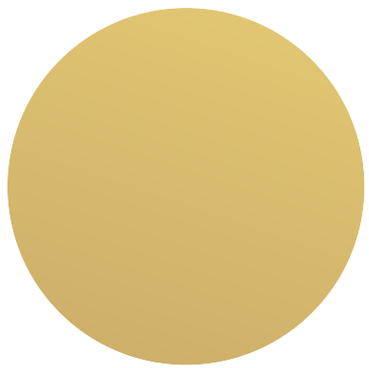
1

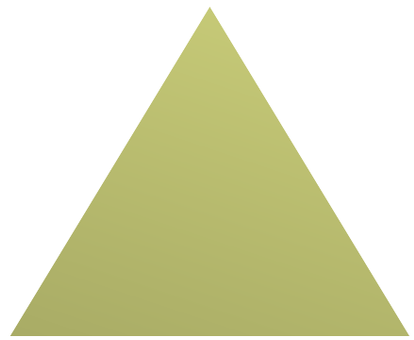
2

3

# Tiefen-Sortierung





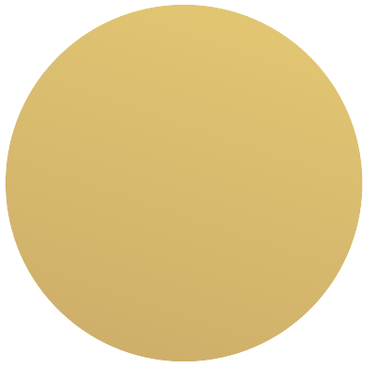


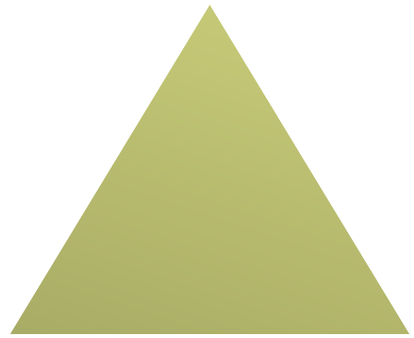
1

2

3

# Tiefen-Sortierung



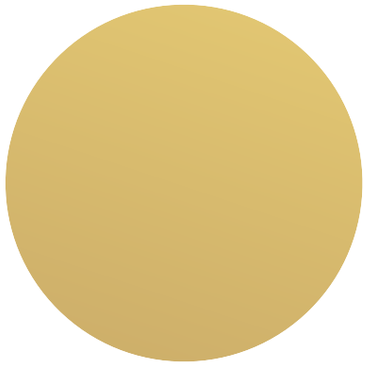


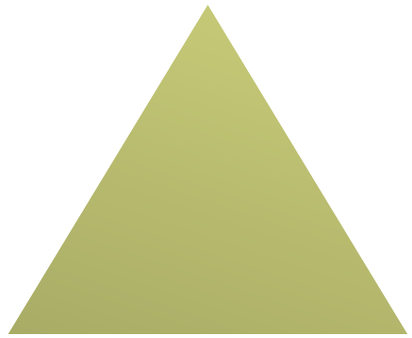
1

2

3

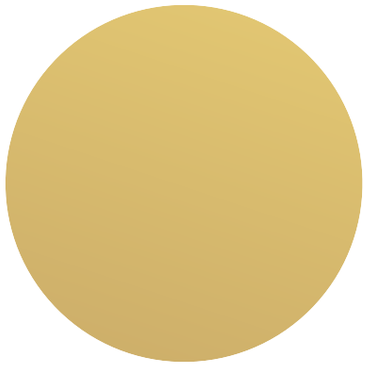
# Tiefen-Sortierung

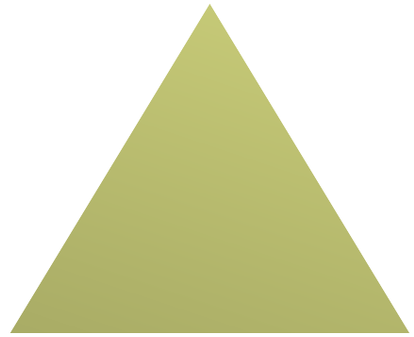




1 2 3

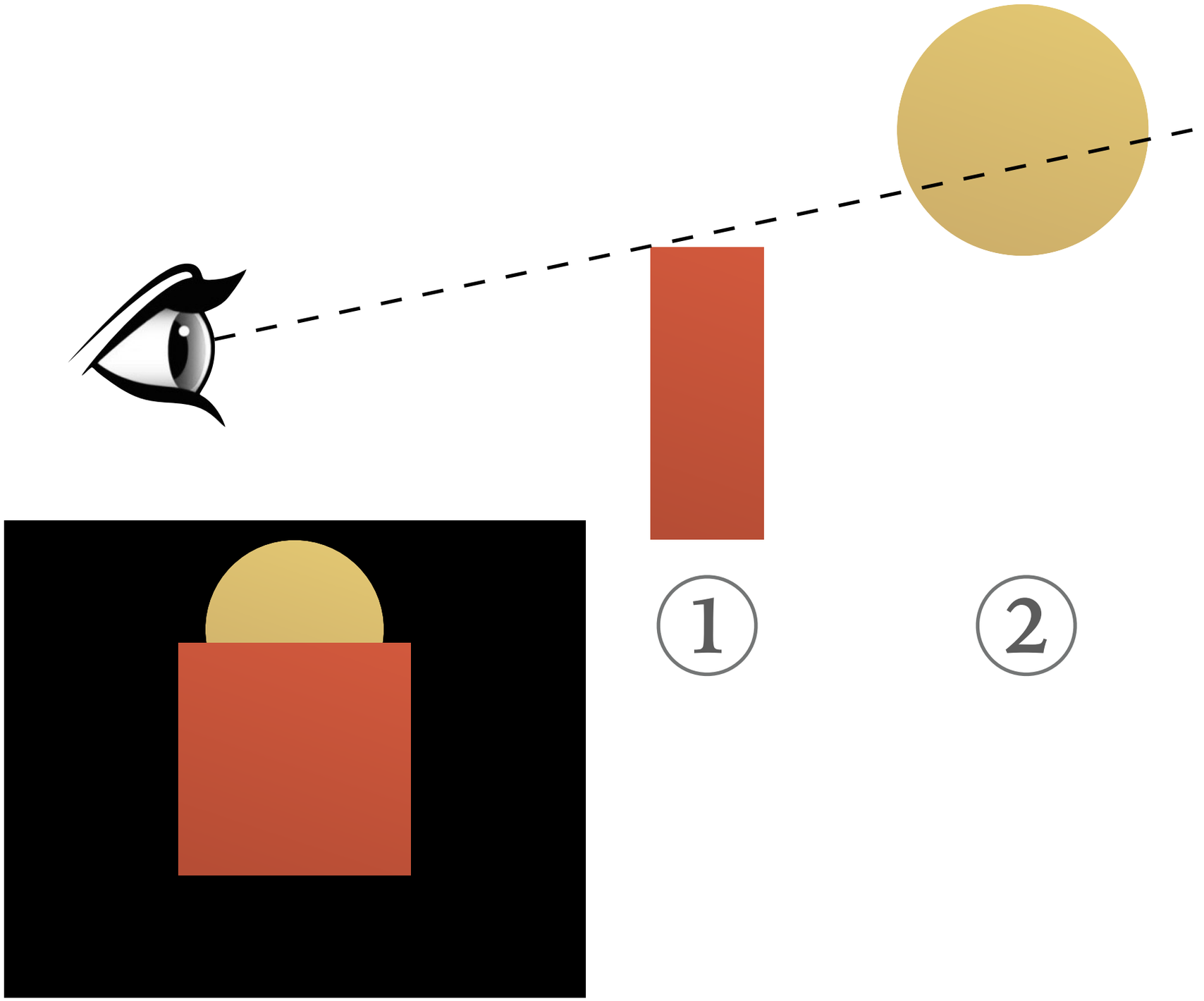
# Tiefen-Sortierung



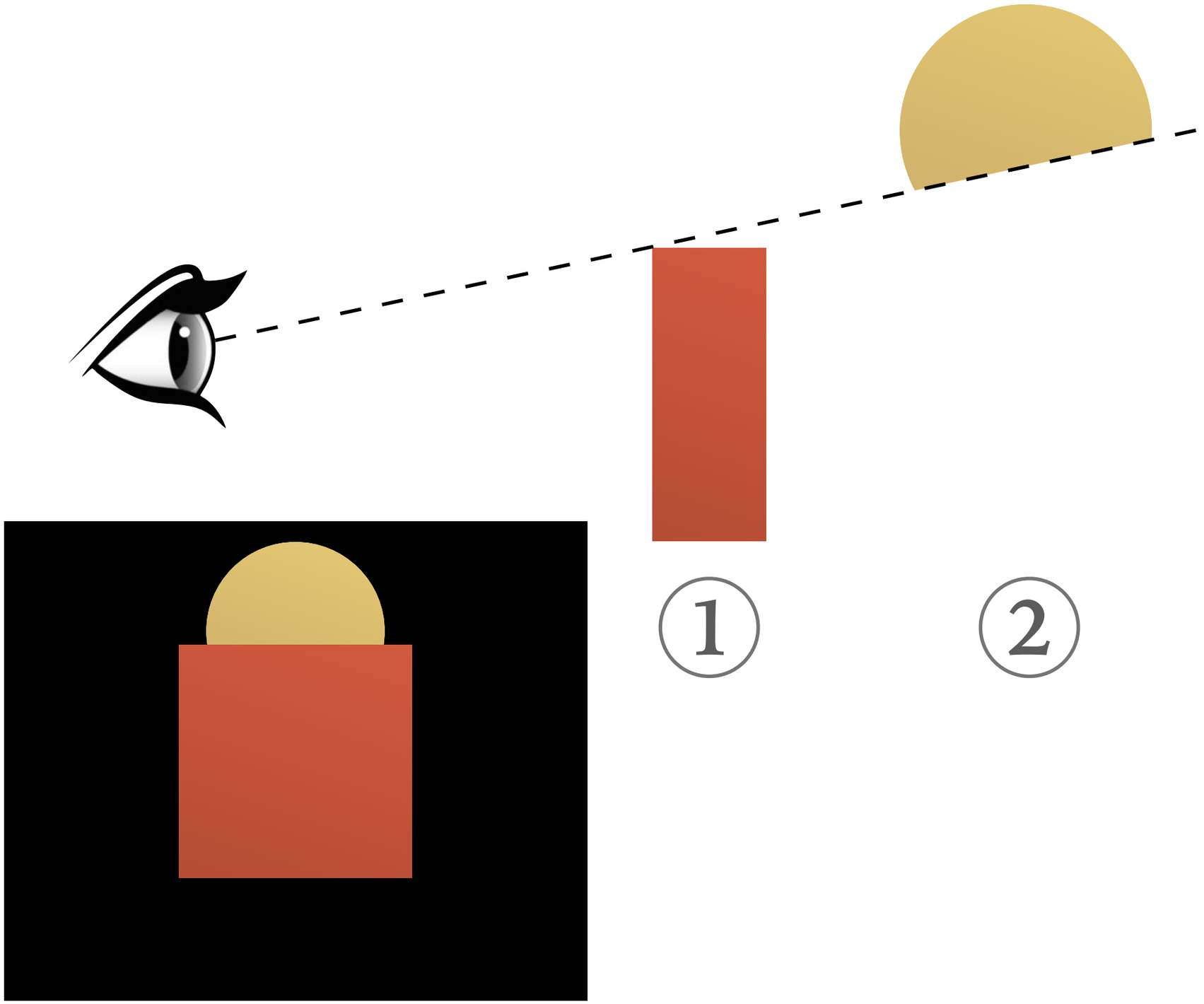


1 2 3

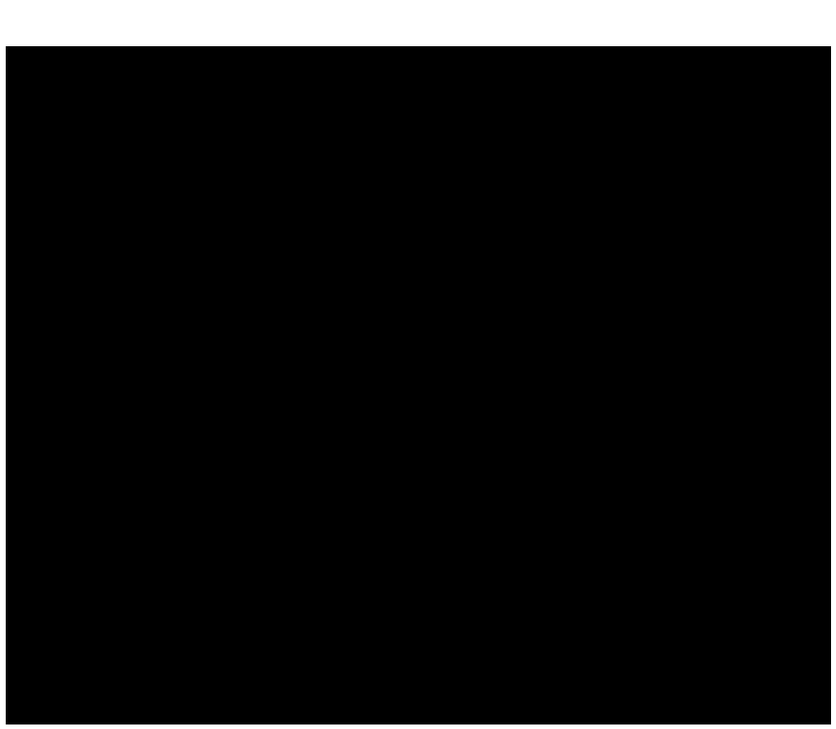
# Tiefen-Sortierung

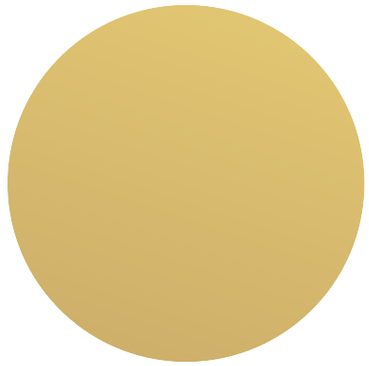


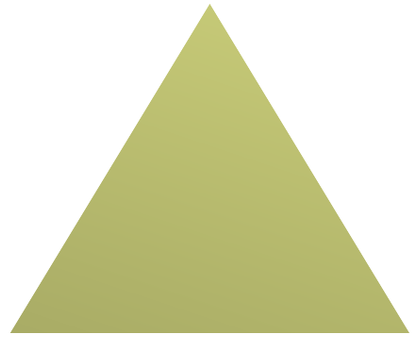
# Tiefen-Sortierung



# Painter's Algorithm

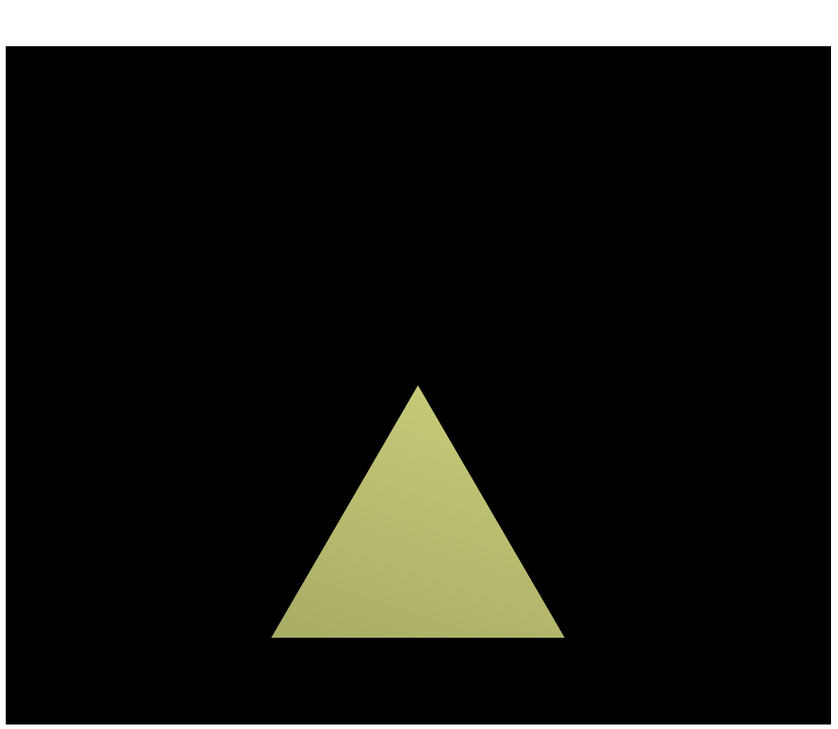


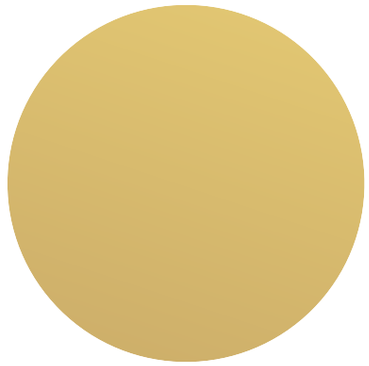


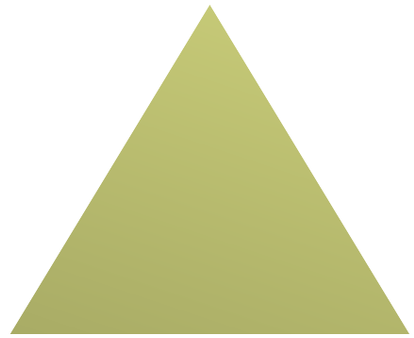


1 2 3

# Painter's Algorithm

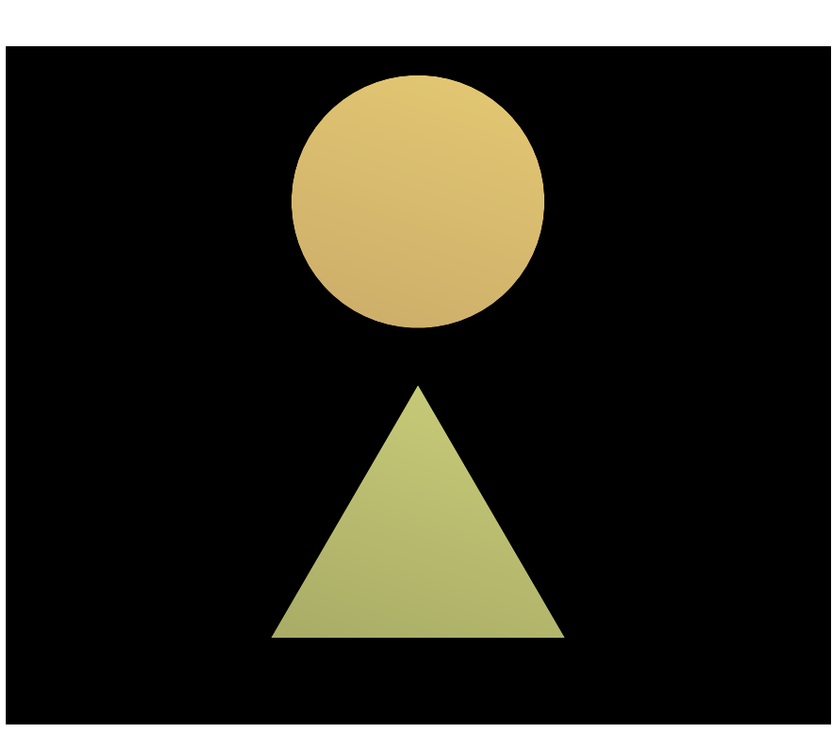


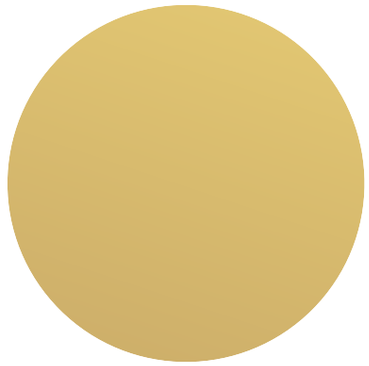


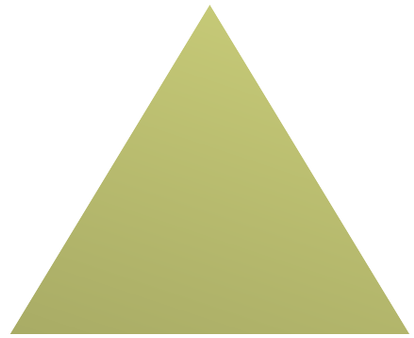


1 2 3

# Painter's Algorithm





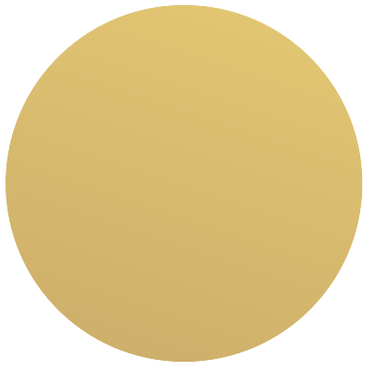


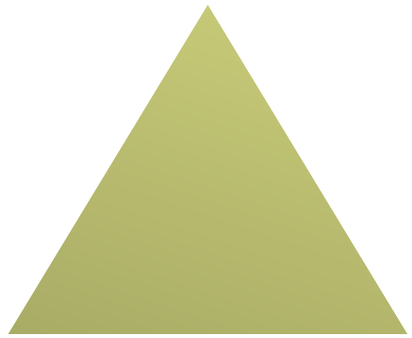
1

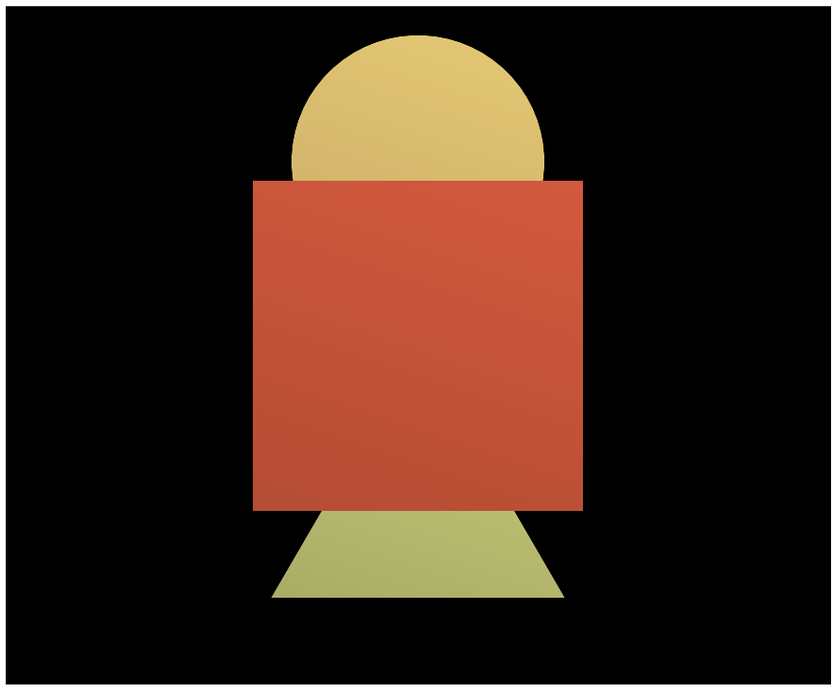
2

3

# Painter's Algorithm

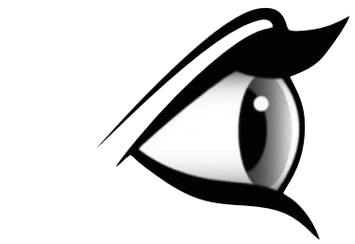


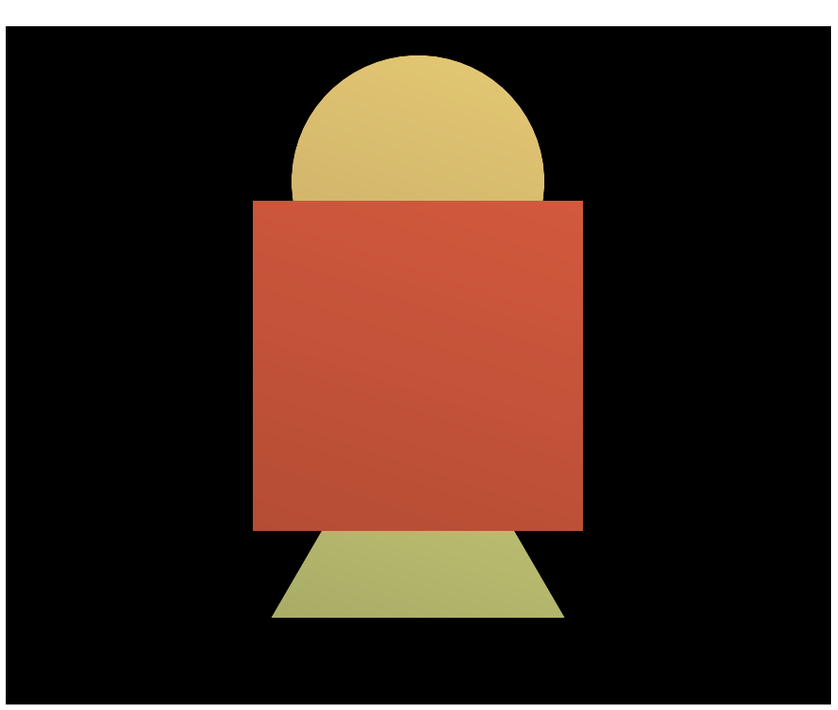


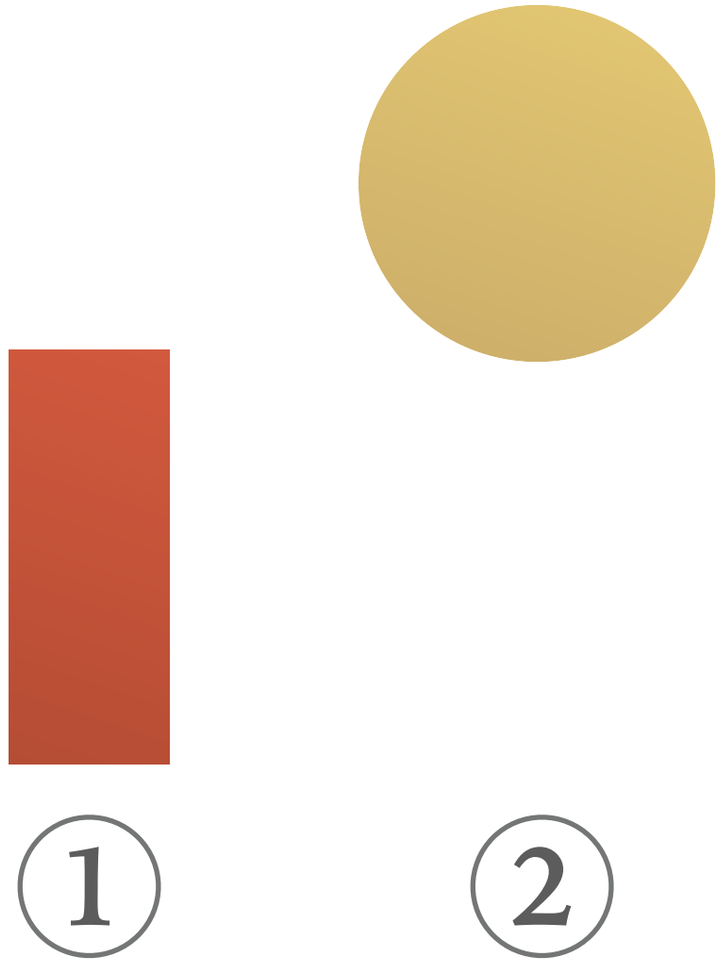


1 2 3

# Painter's Algorithm



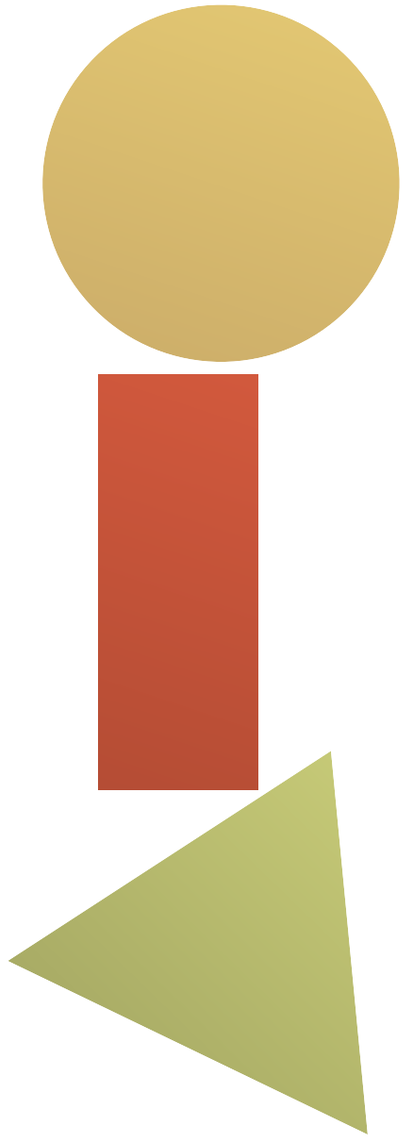




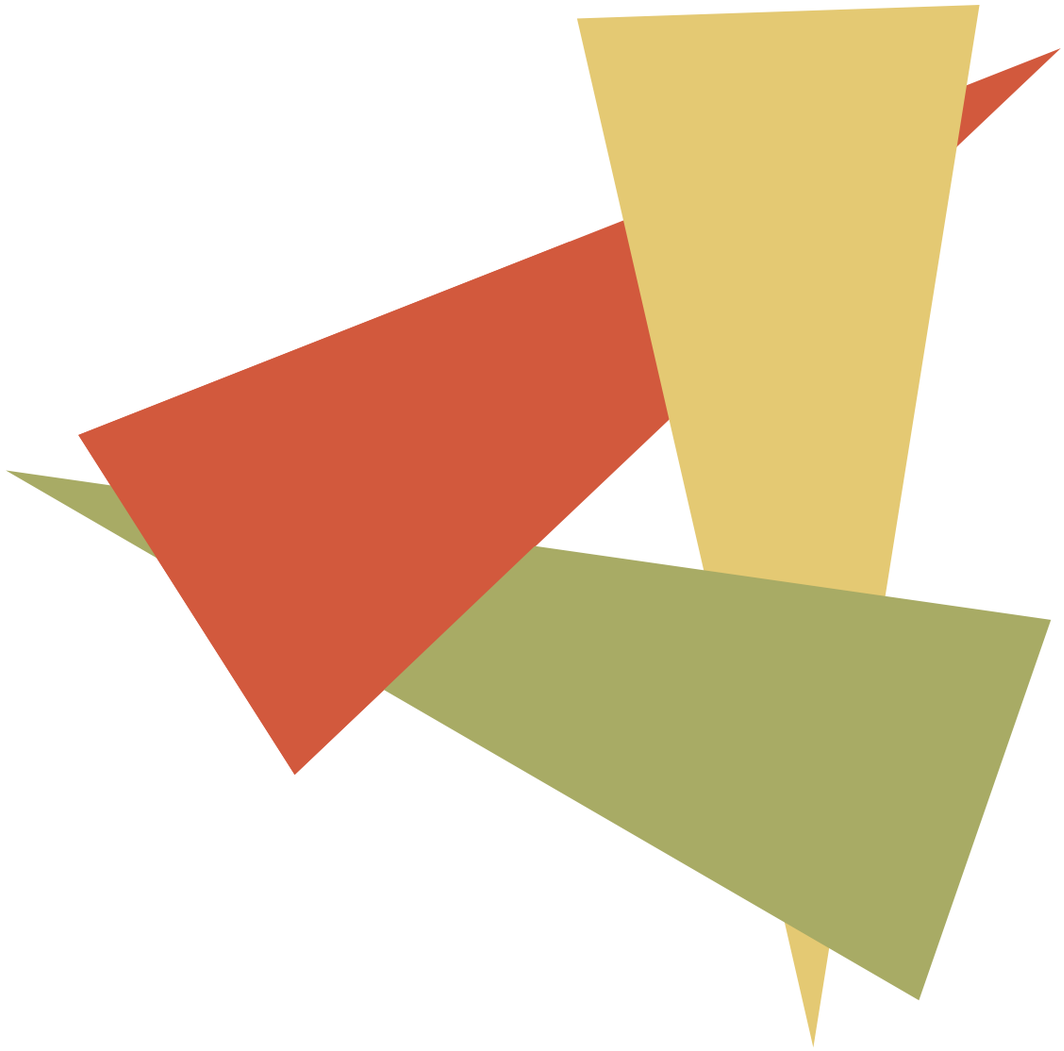
1 2 3

von hinten nach vorne zeichnen,
dabei immer überschreiben

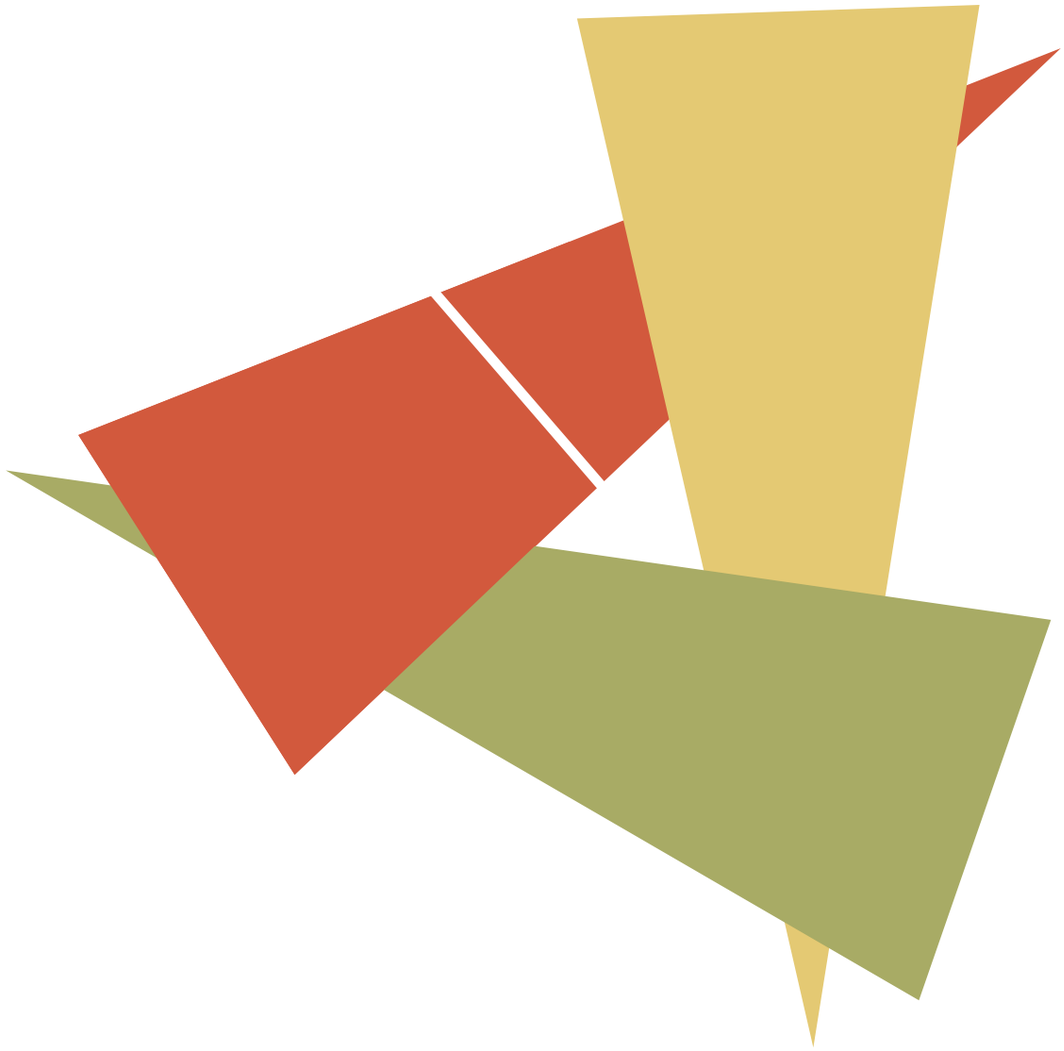
# Painter's Algorithm



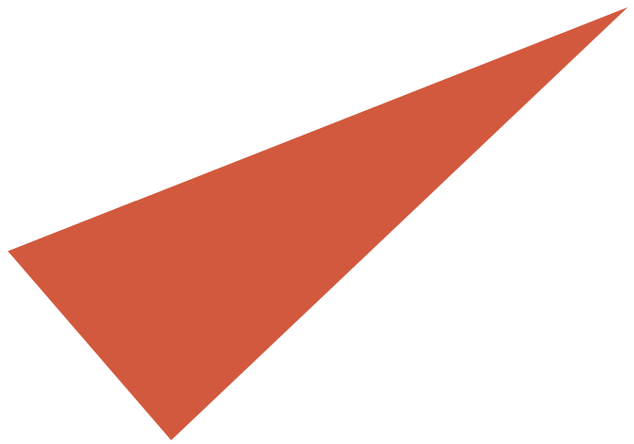
# Painter's Algorithm



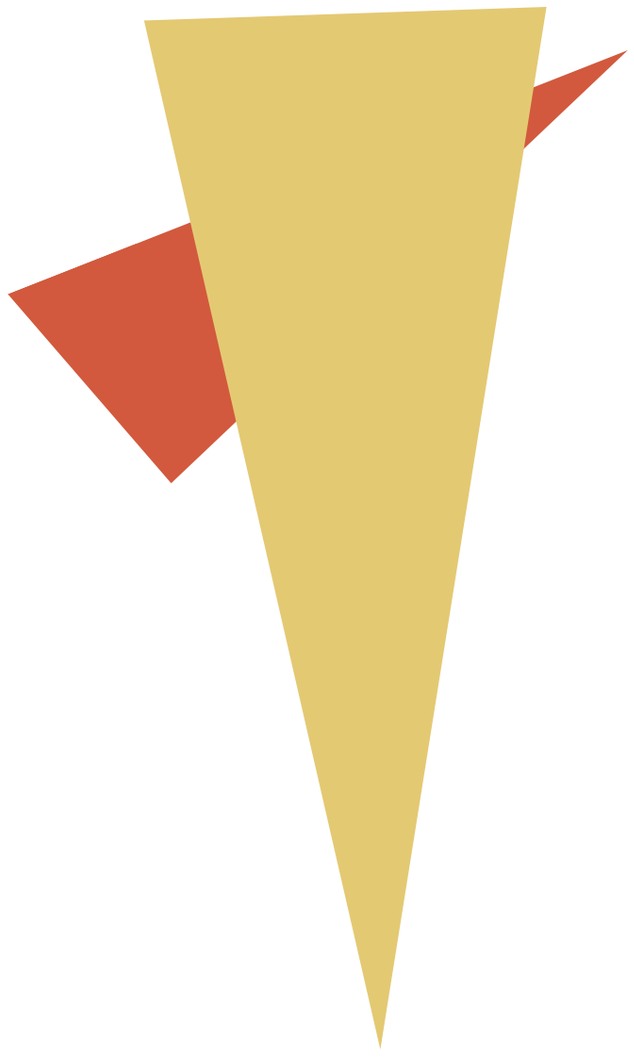
# Painter's Algorithm



# Painter’s Algorithm

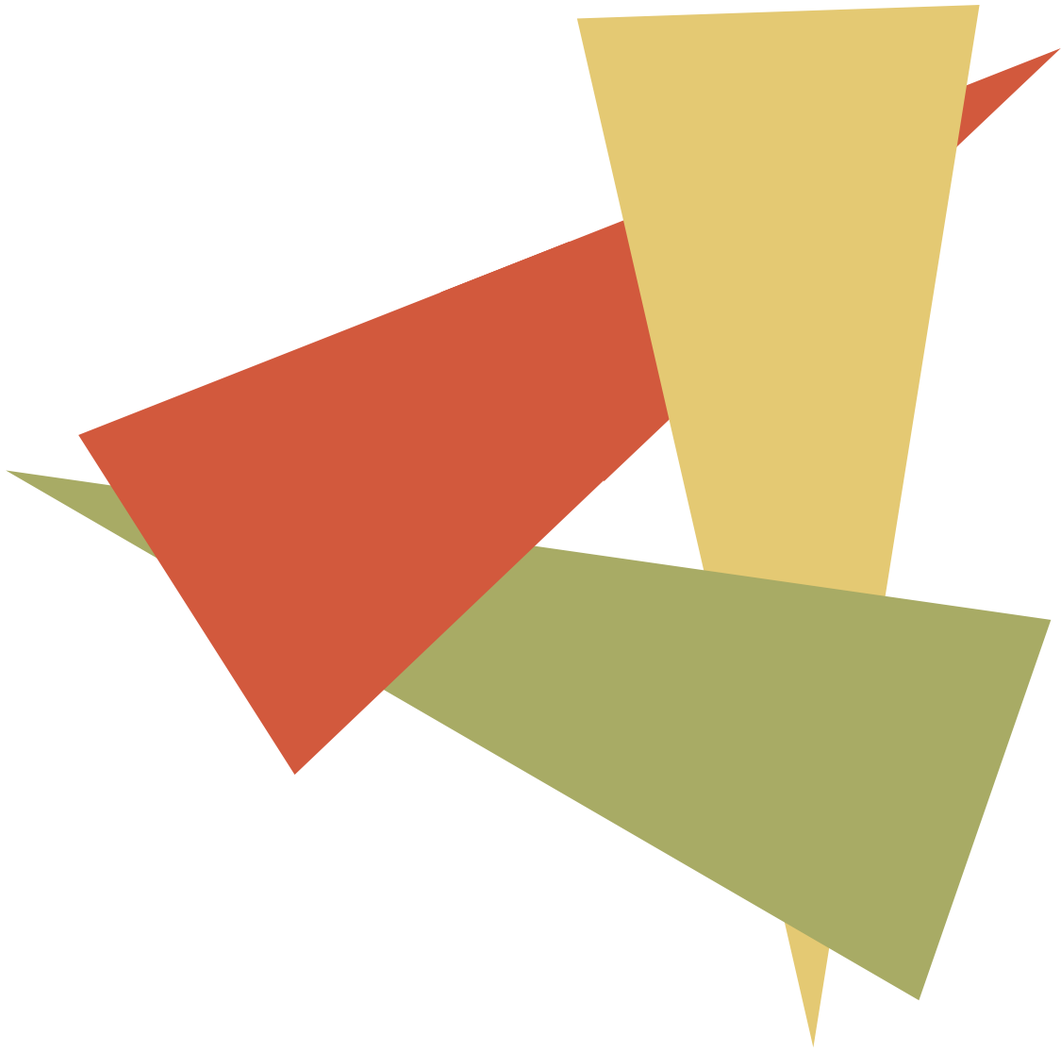


# Painter's Algorithm

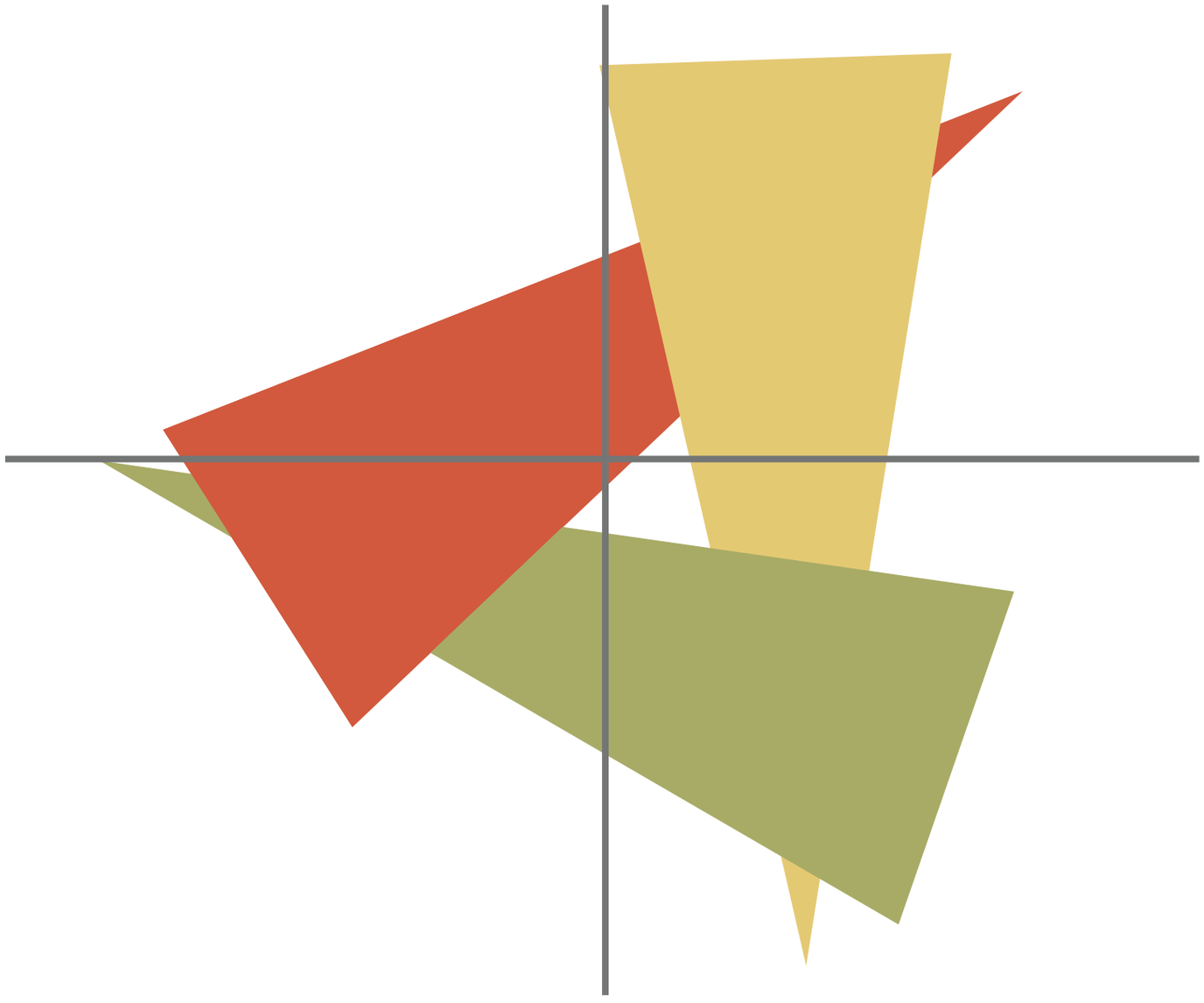


# Painter's Algorithm

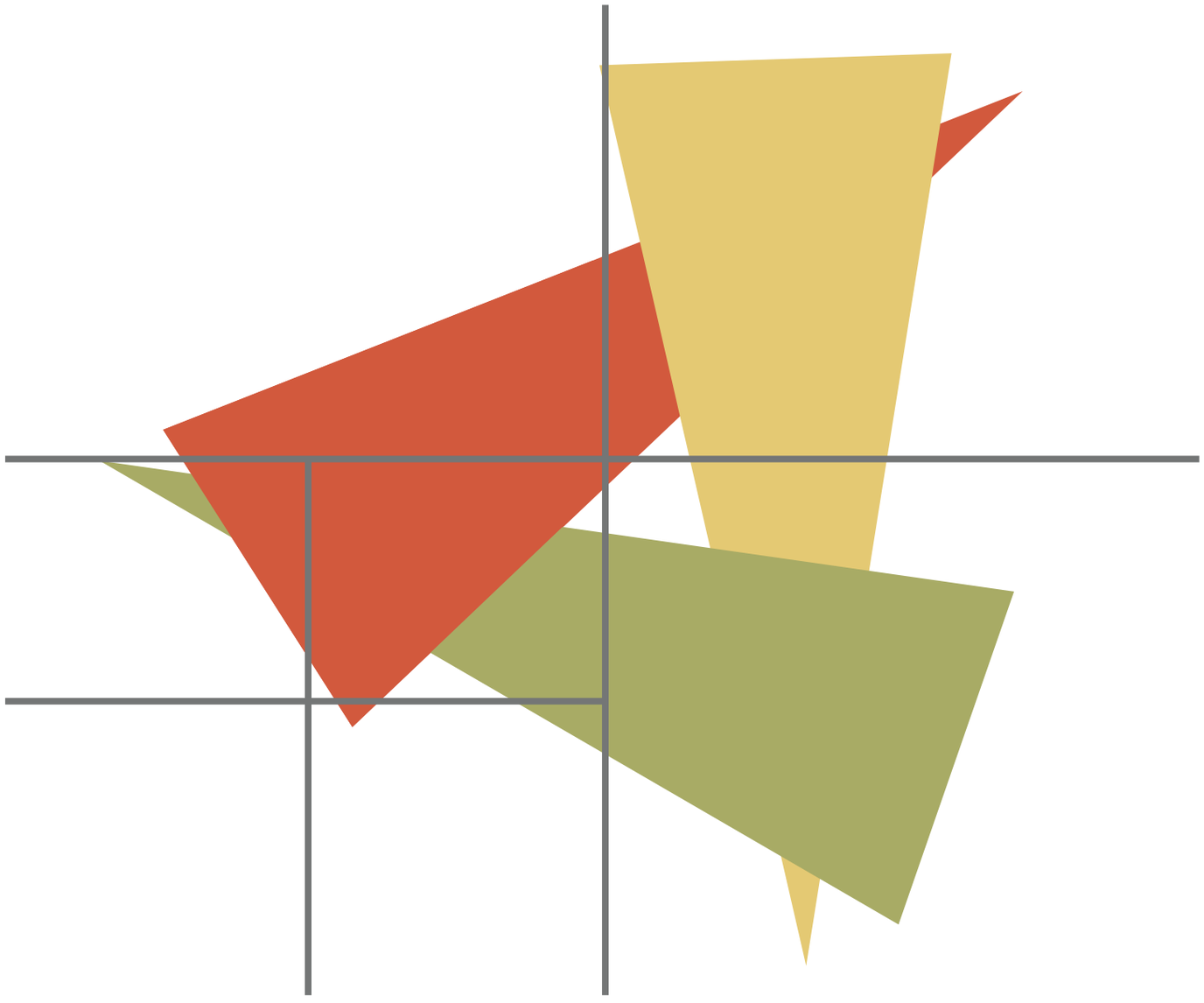
# Painter's Algorithm



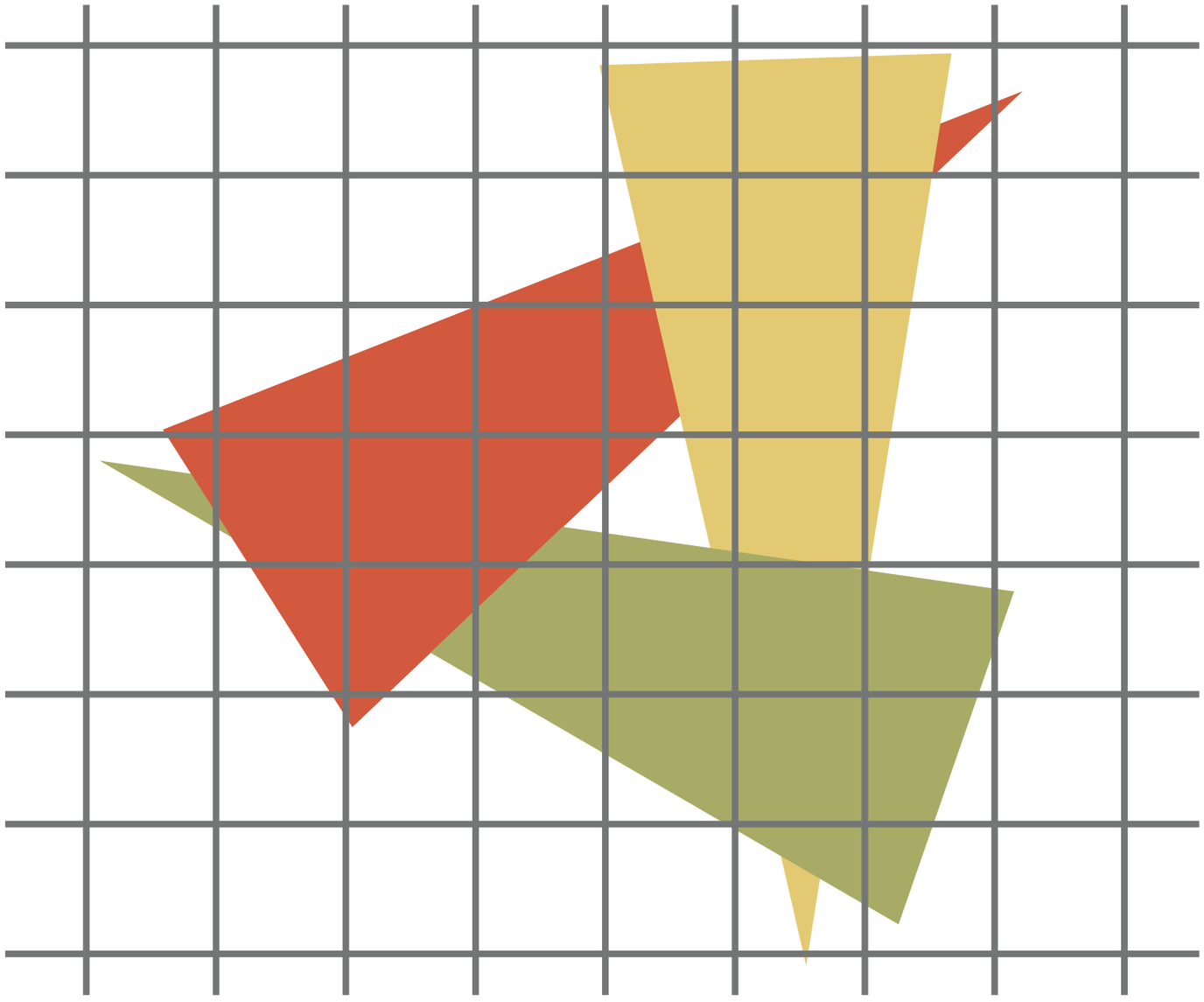
# Painter's Algorithm mit Screen Subdivision



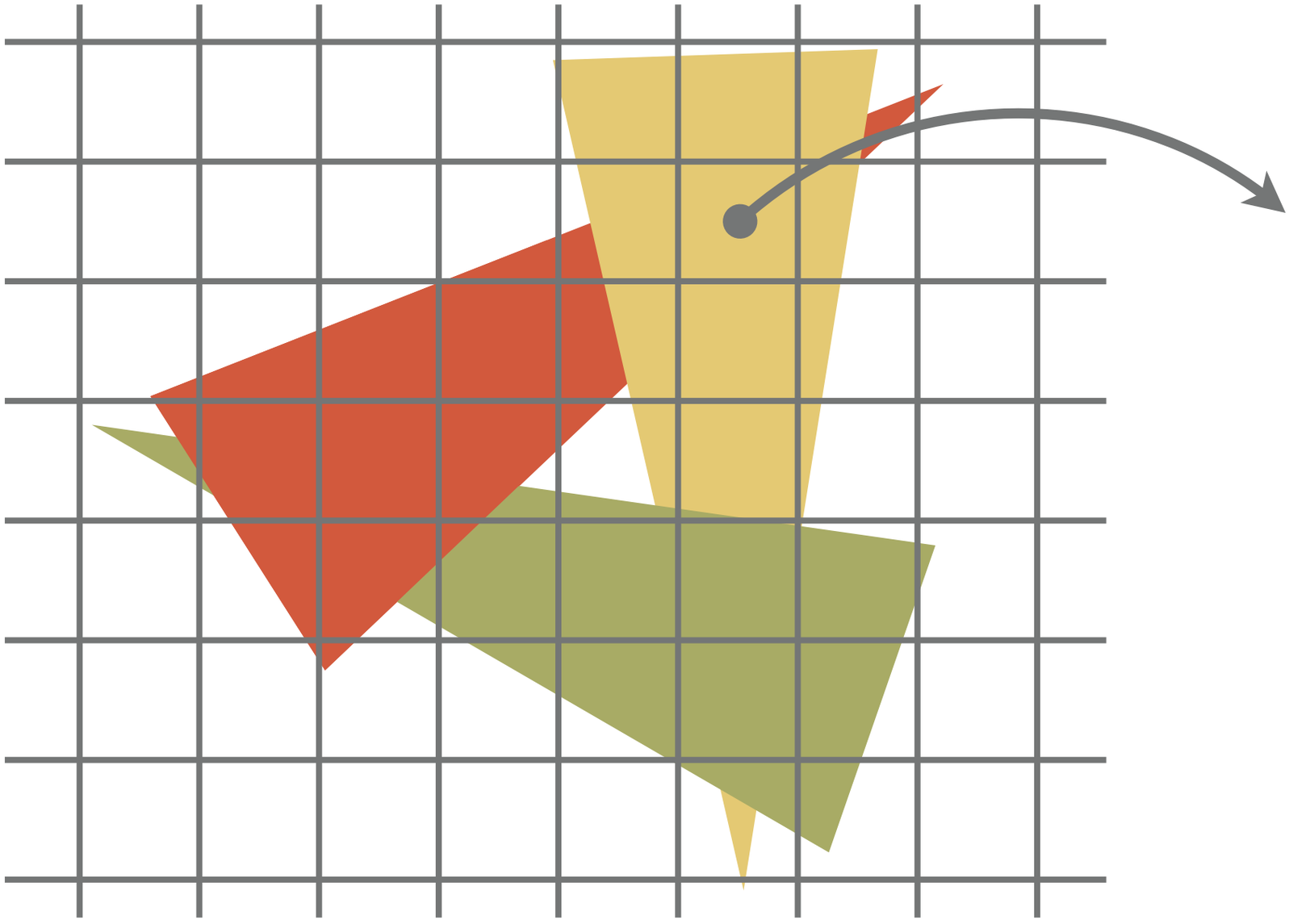
# Painter’s Algorithm mit Screen Subdivision



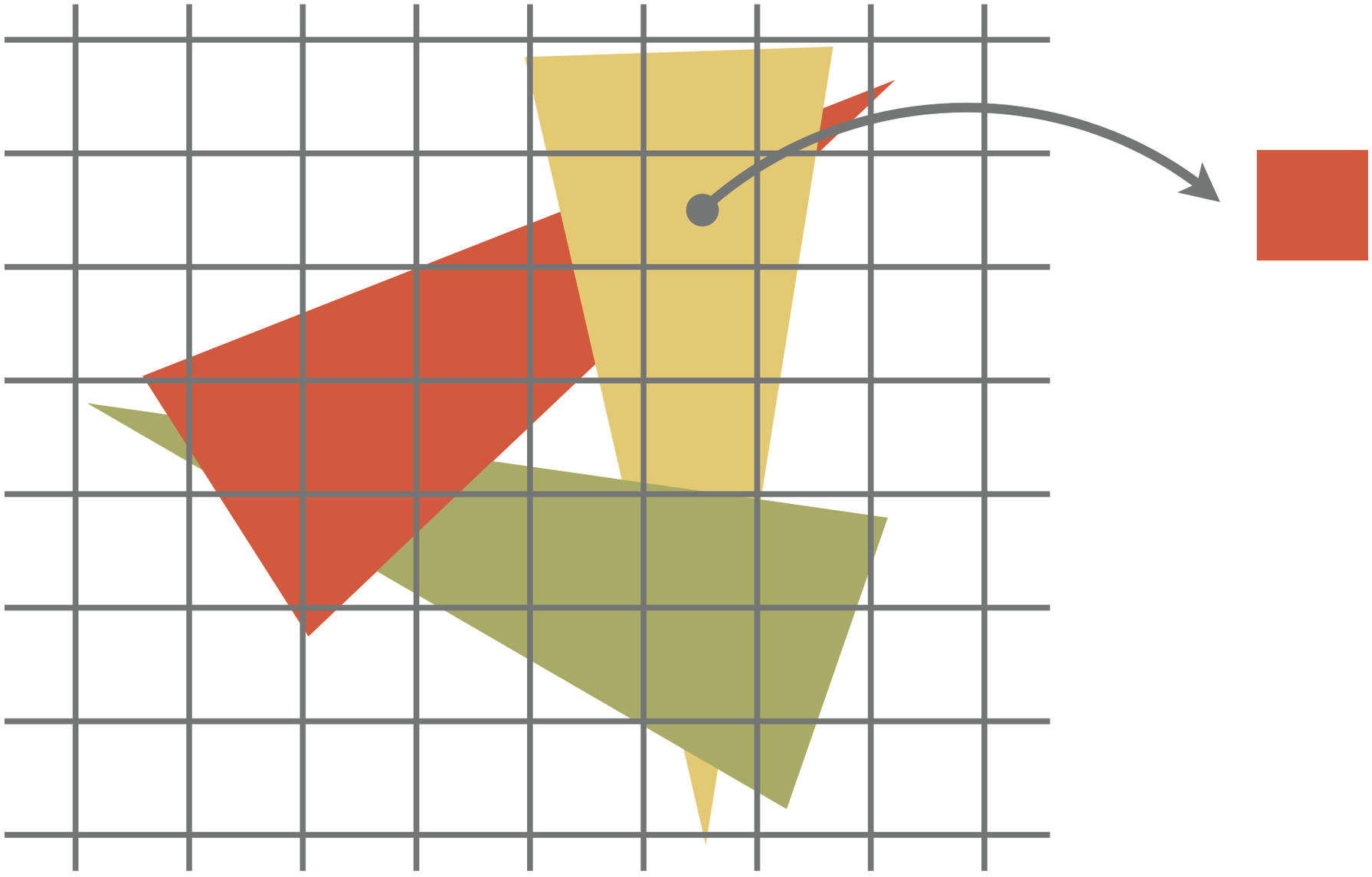
# Painter's Algorithm mit Screen Subdivision



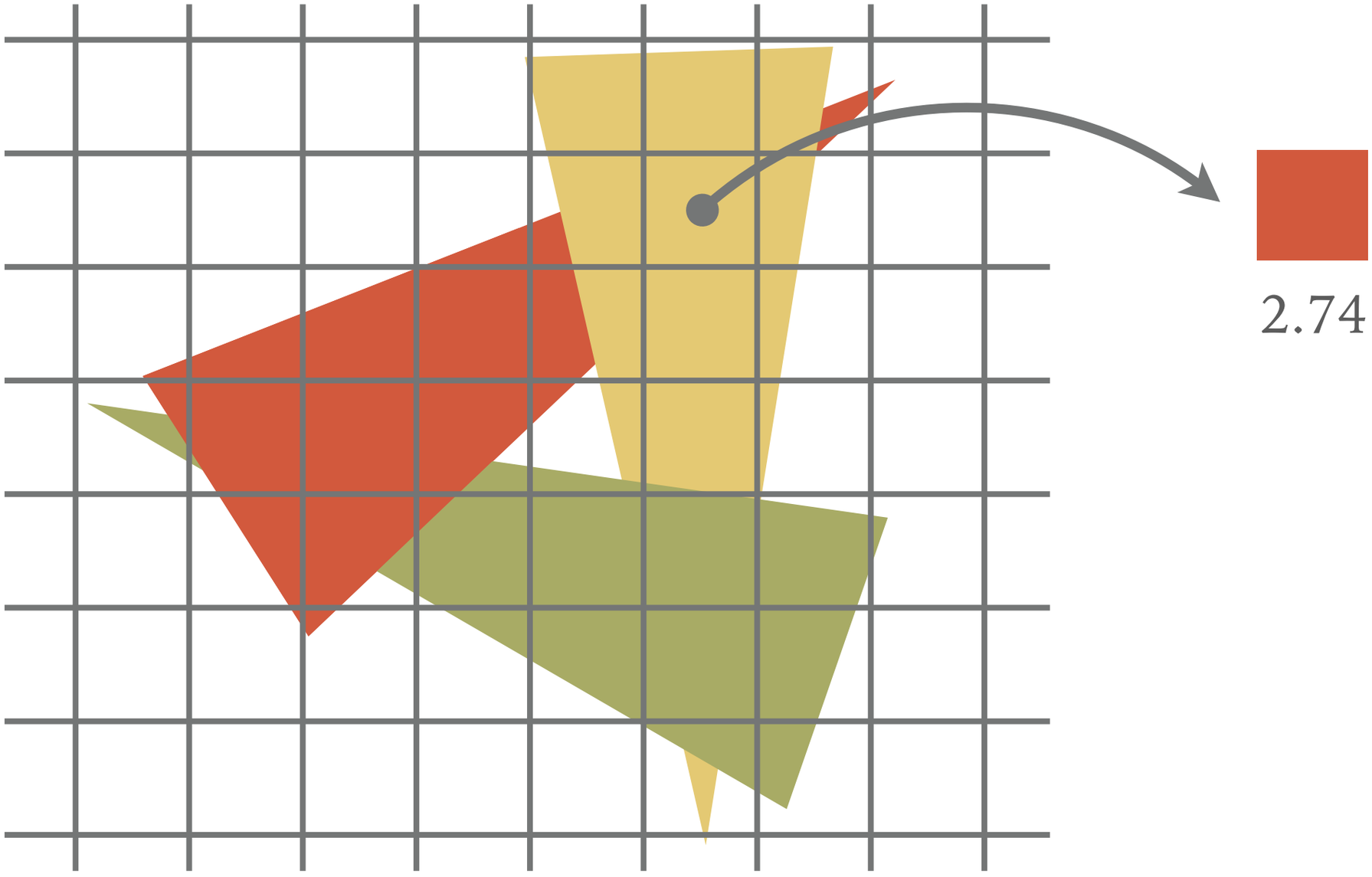
# Painter's Algorithm mit Screen Subdivision



# Painter's Algorithm mit Screen Subdivision

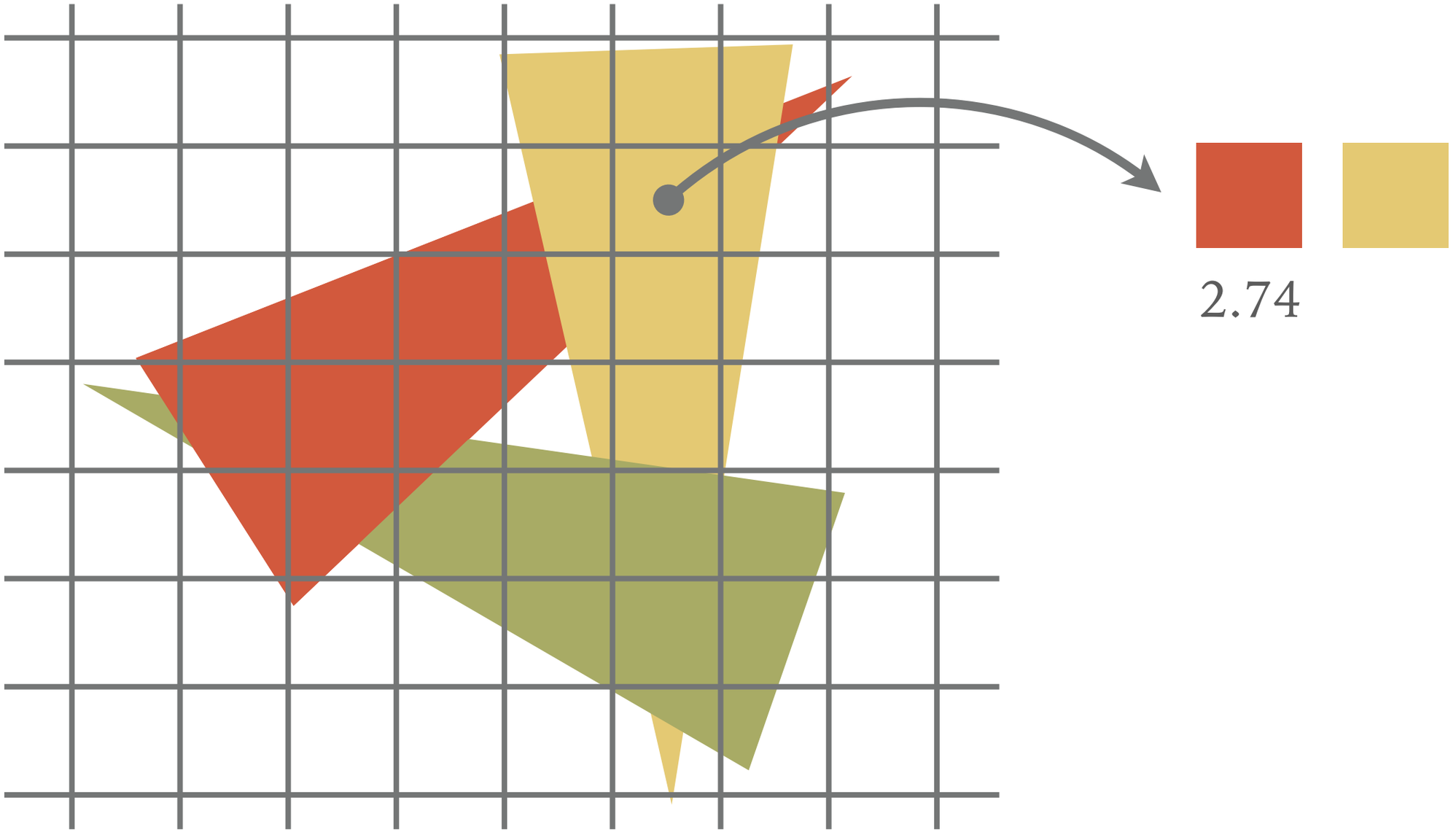


# Painter's Algorithm mit Screen Subdivision

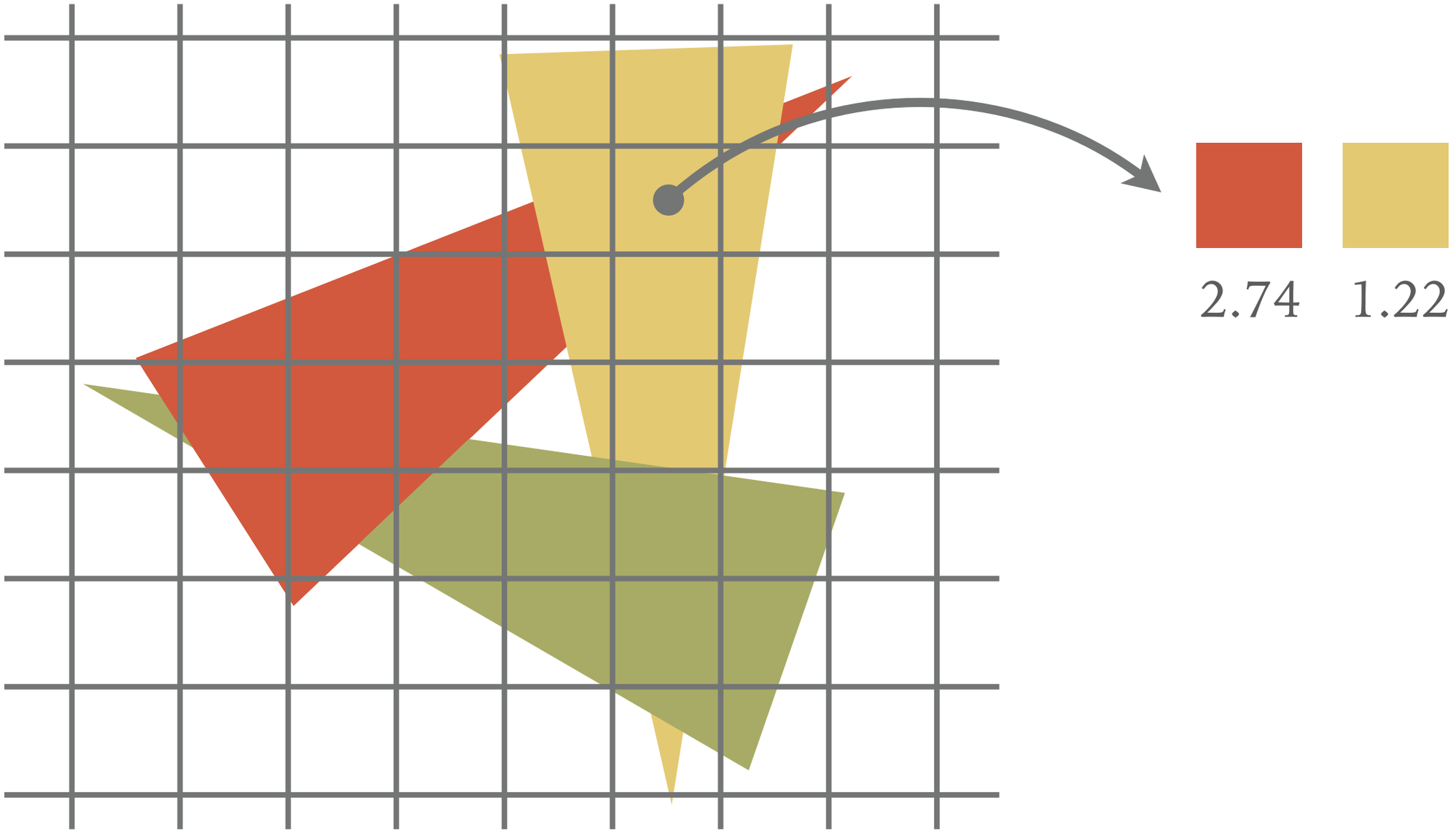


2.74

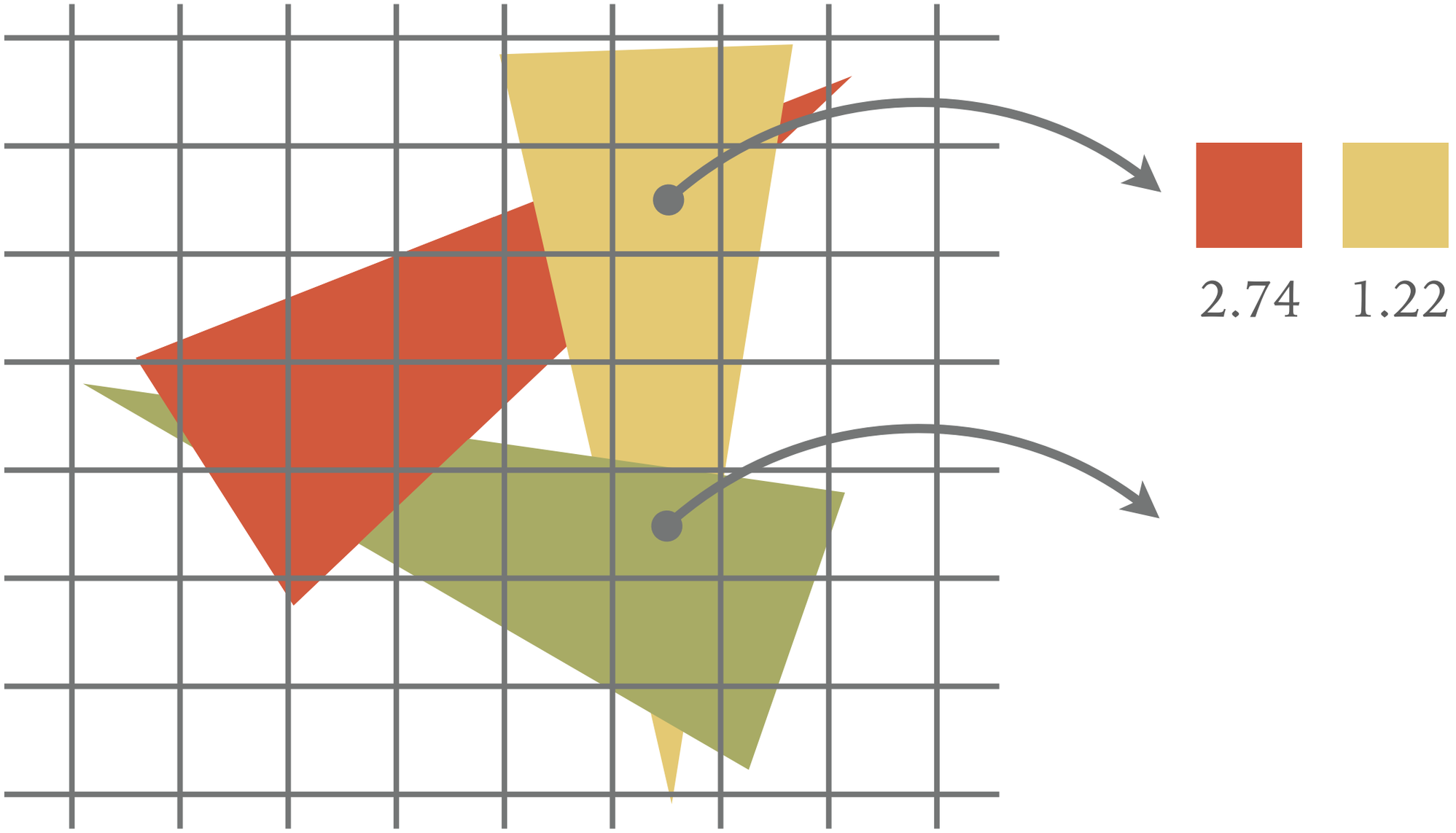
# Painter's Algorithm mit Screen Subdivision



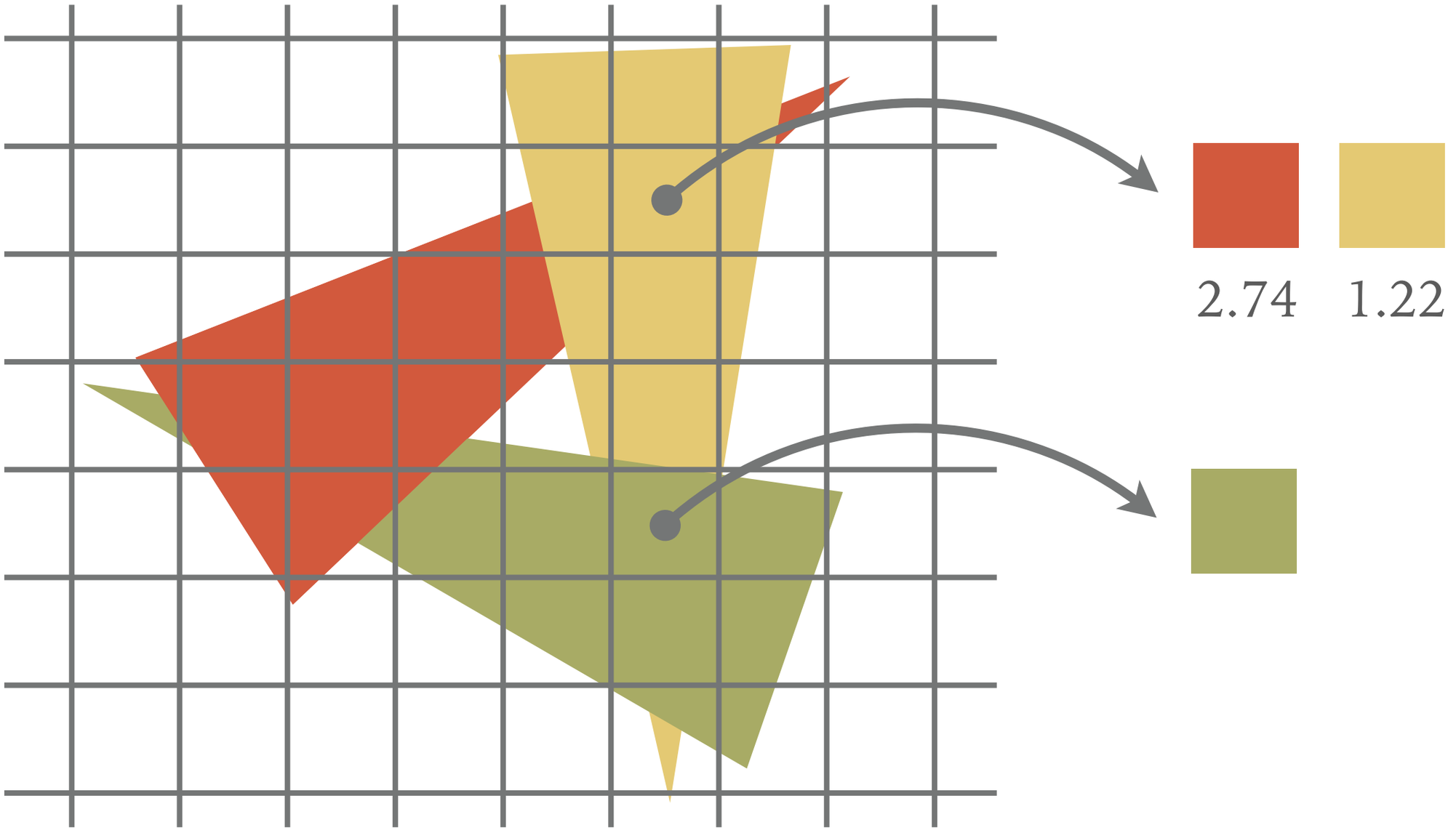
# Painter's Algorithm mit Screen Subdivision



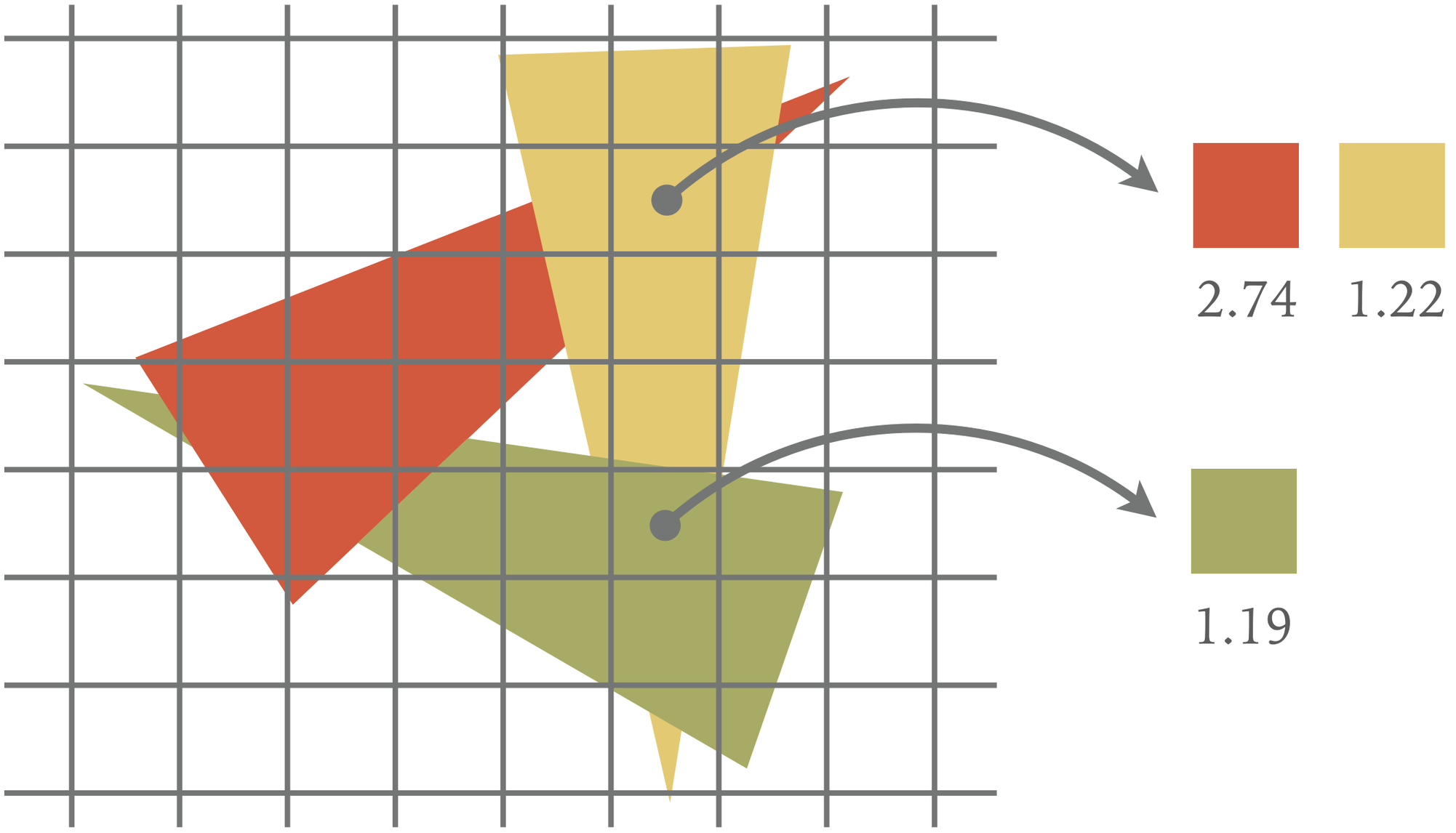
# Painter's Algorithm mit Screen Subdivision



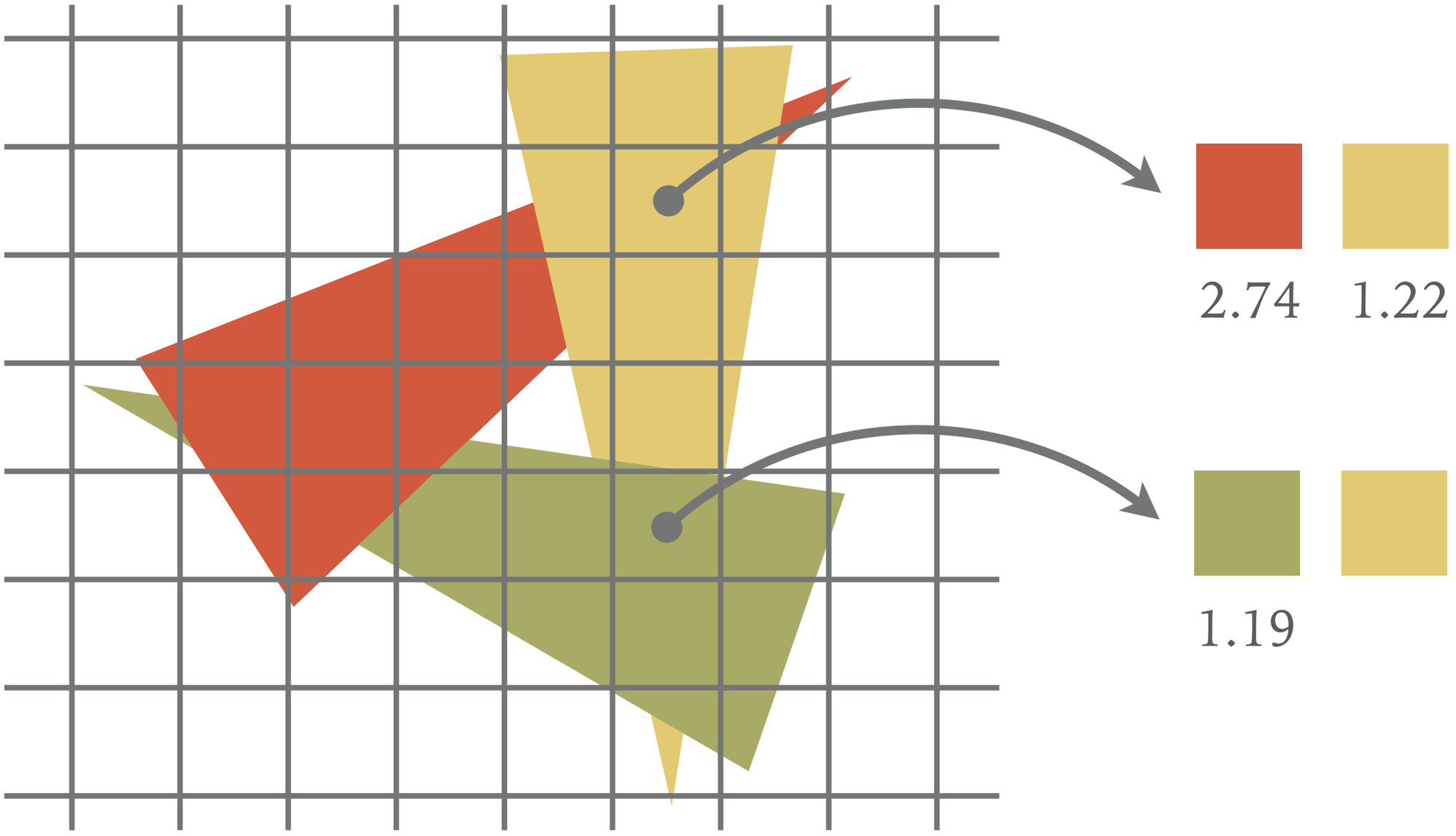
# Painter's Algorithm mit Screen Subdivision



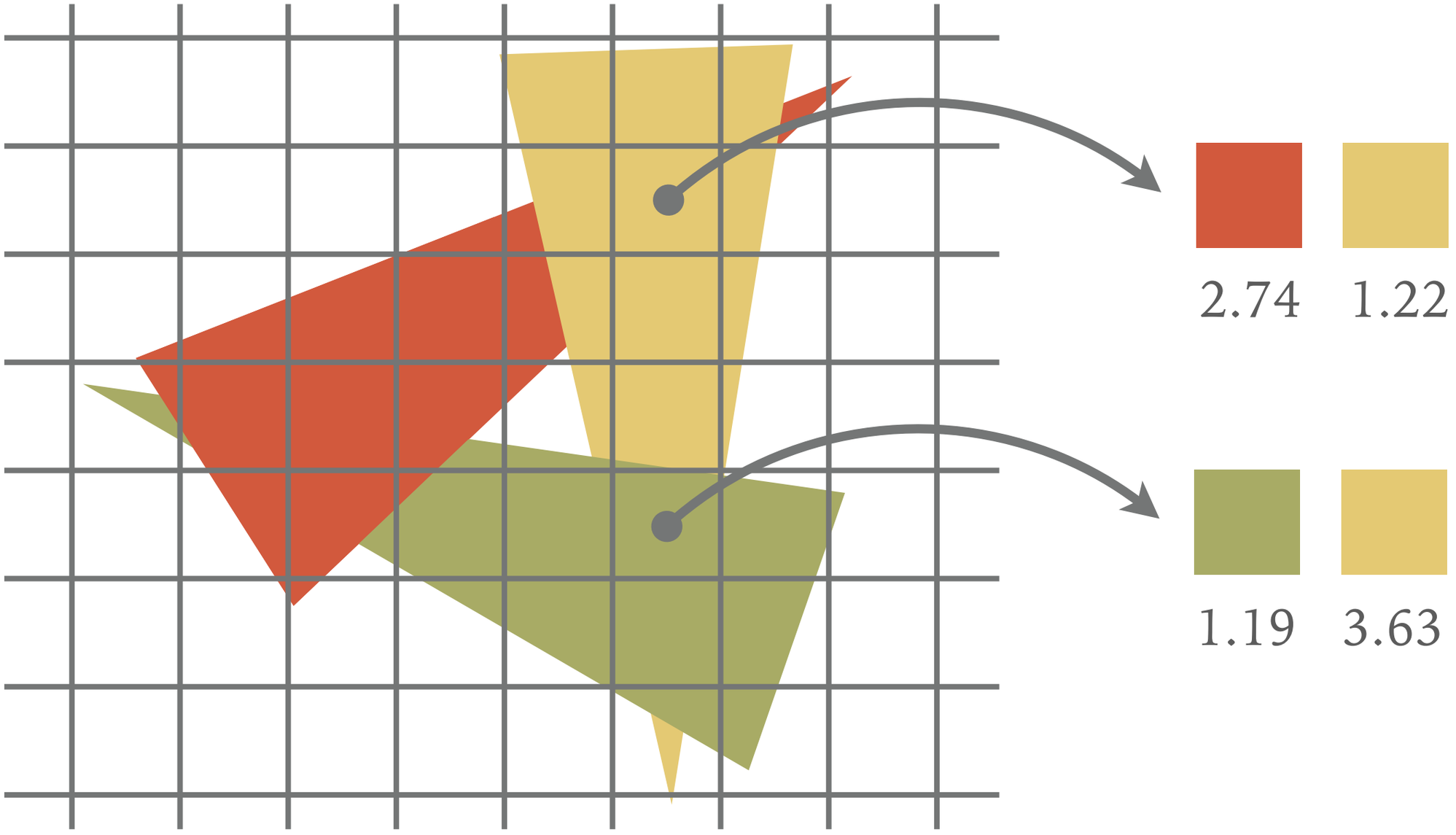
# Painter's Algorithm mit Screen Subdivision



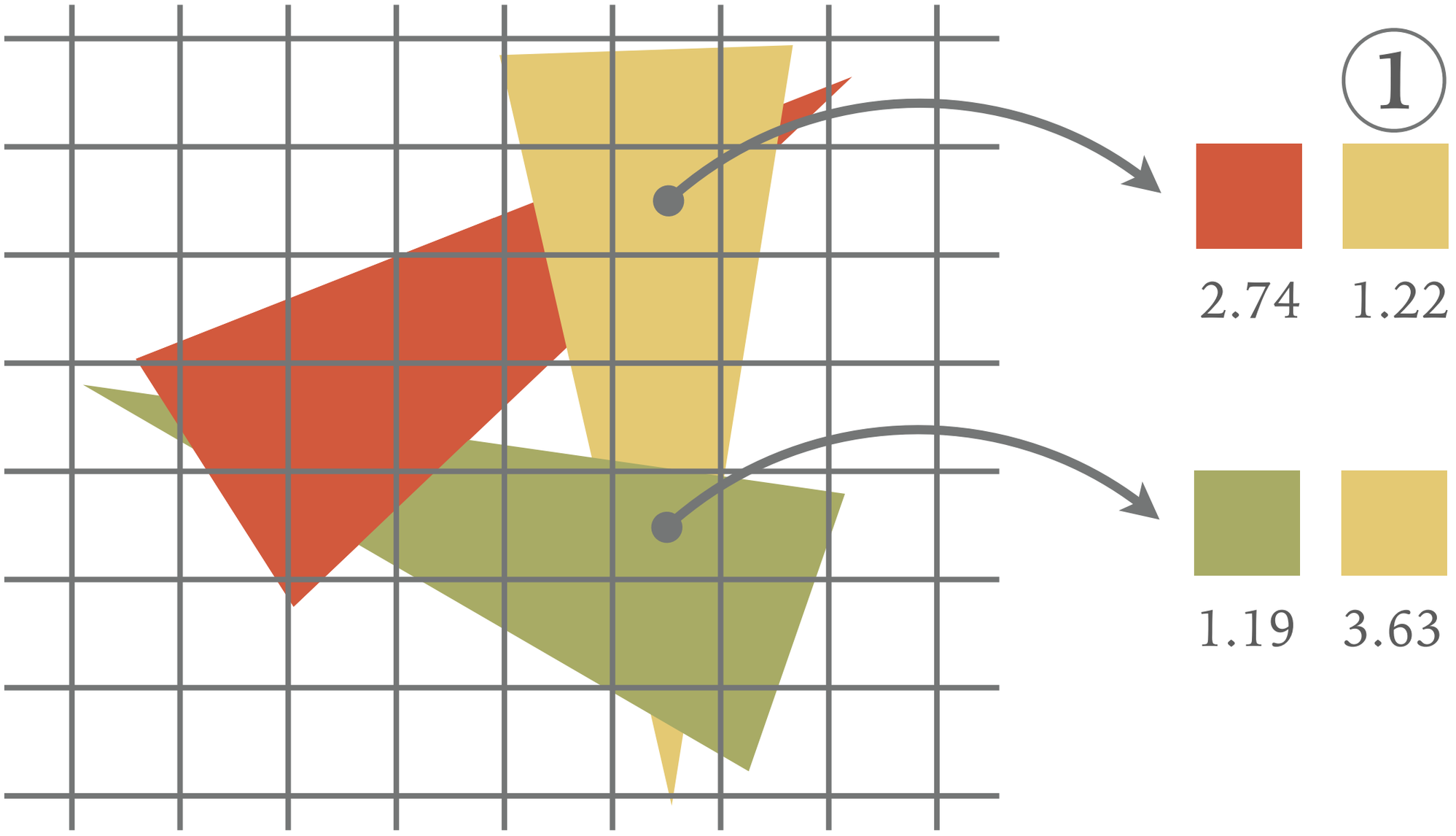
# Painter's Algorithm mit Screen Subdivision



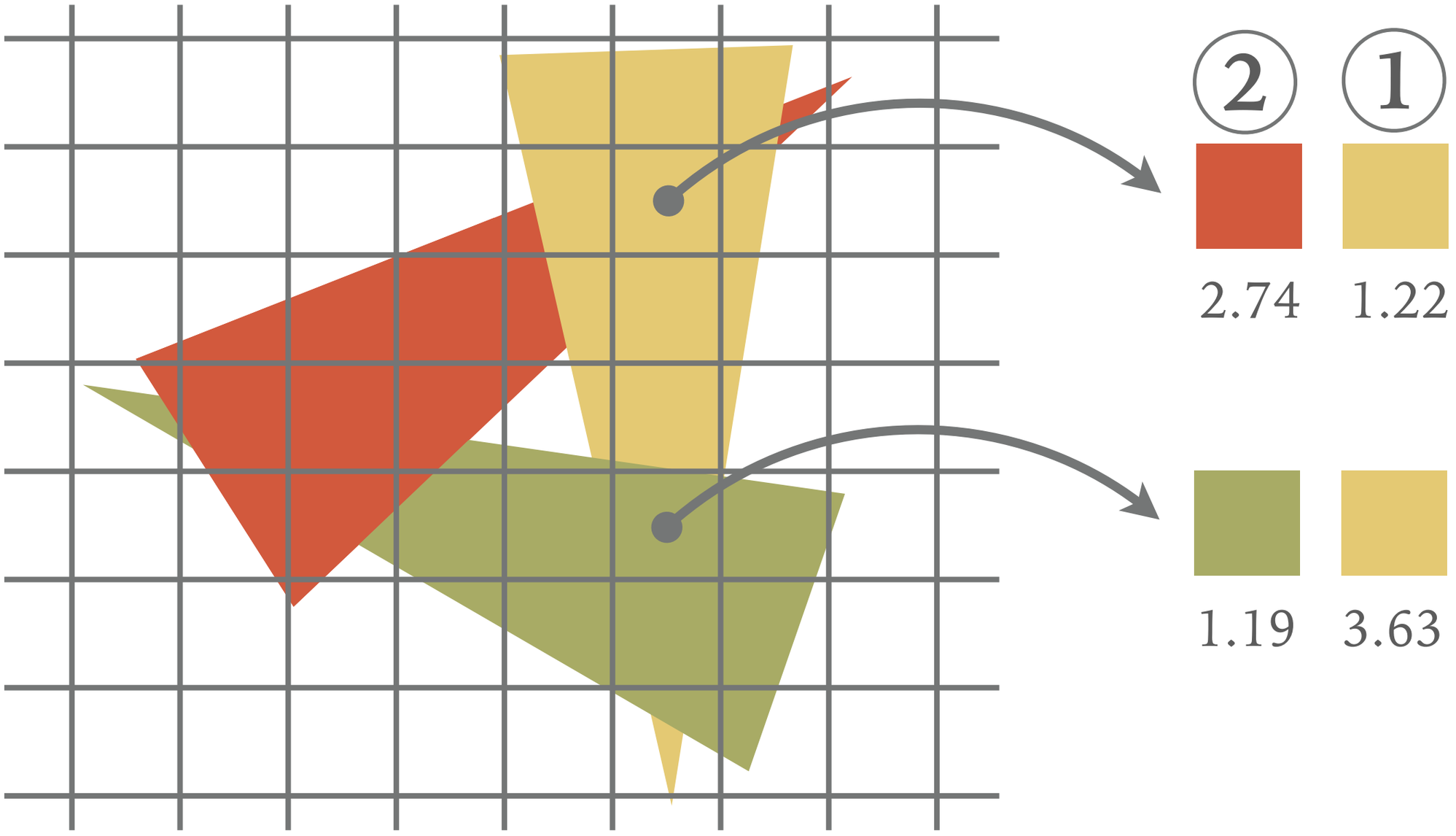
# Painter's Algorithm mit Screen Subdivision



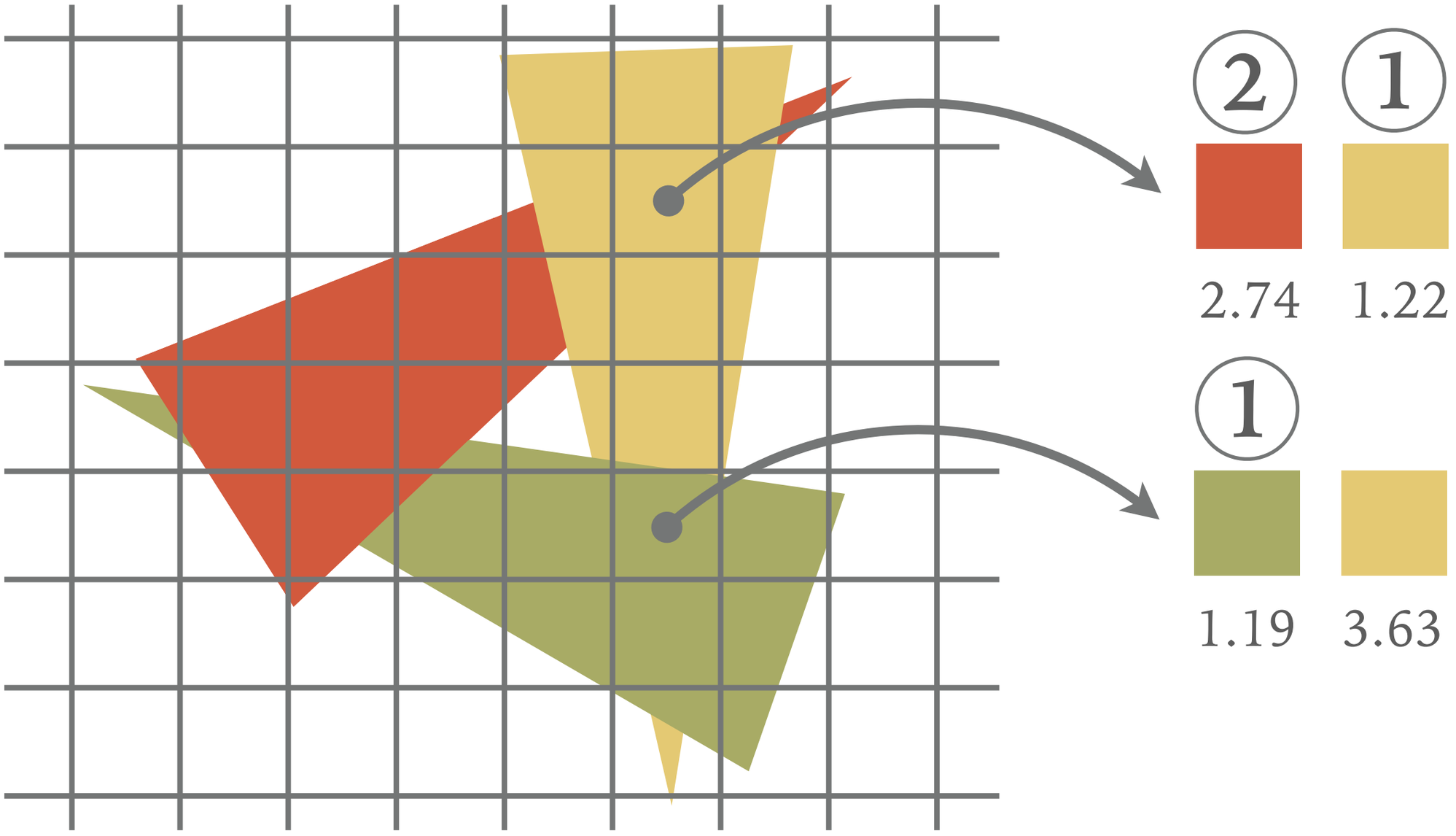
# Painter's Algorithm mit Screen Subdivision



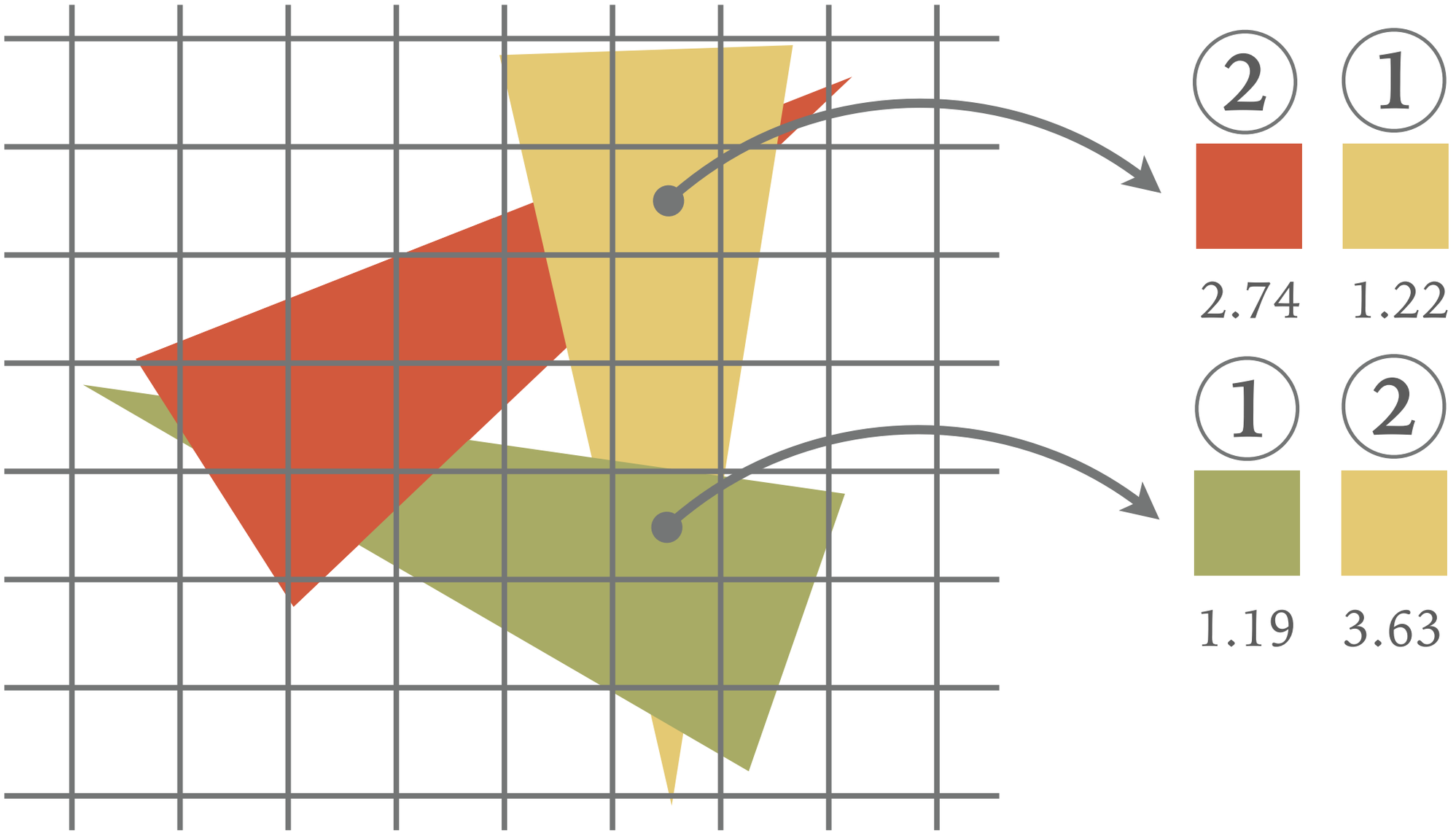
# Painter's Algorithm mit Screen Subdivision



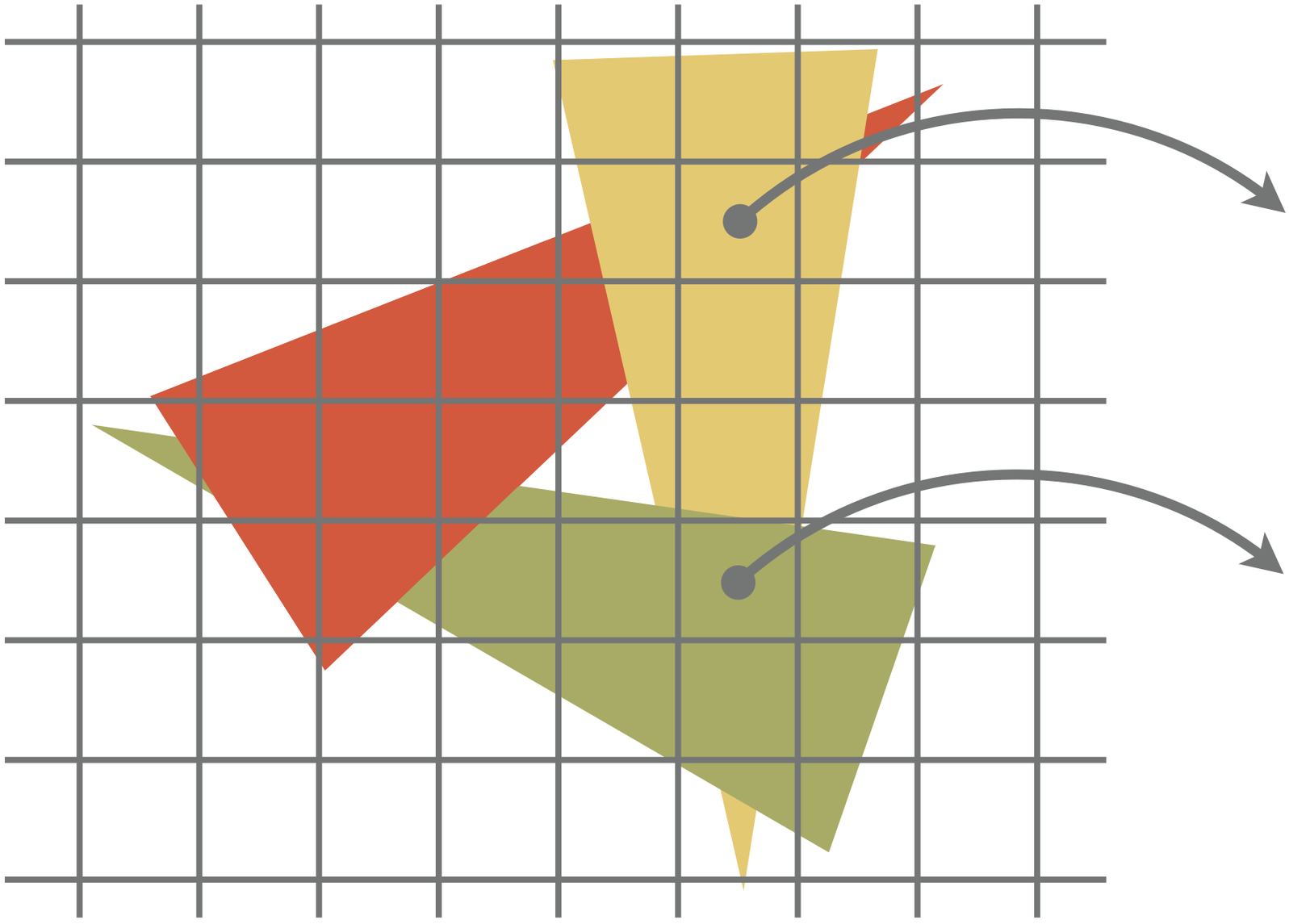
# Painter's Algorithm mit Screen Subdivision



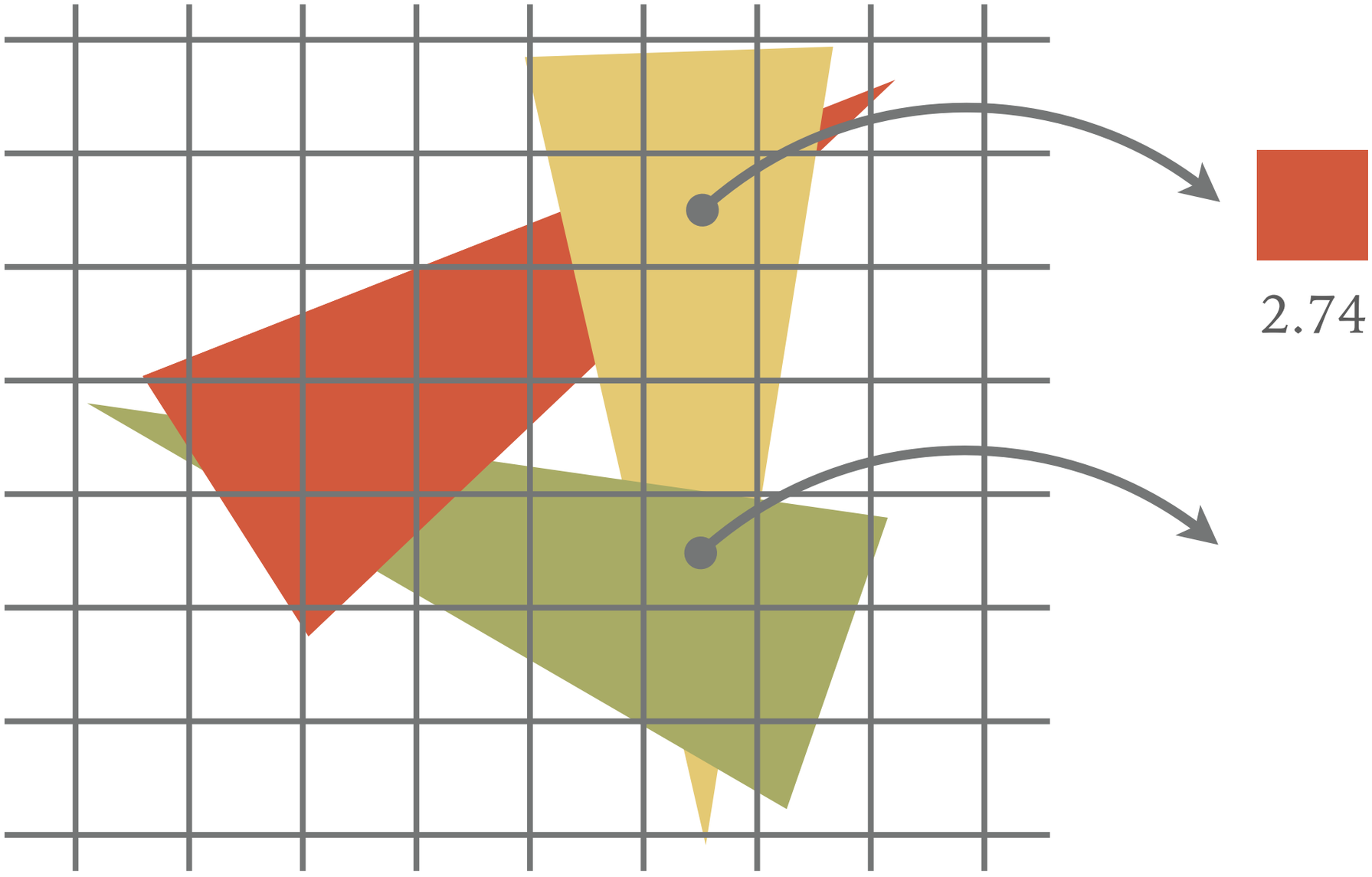
# Painter's Algorithm mit Screen Subdivision



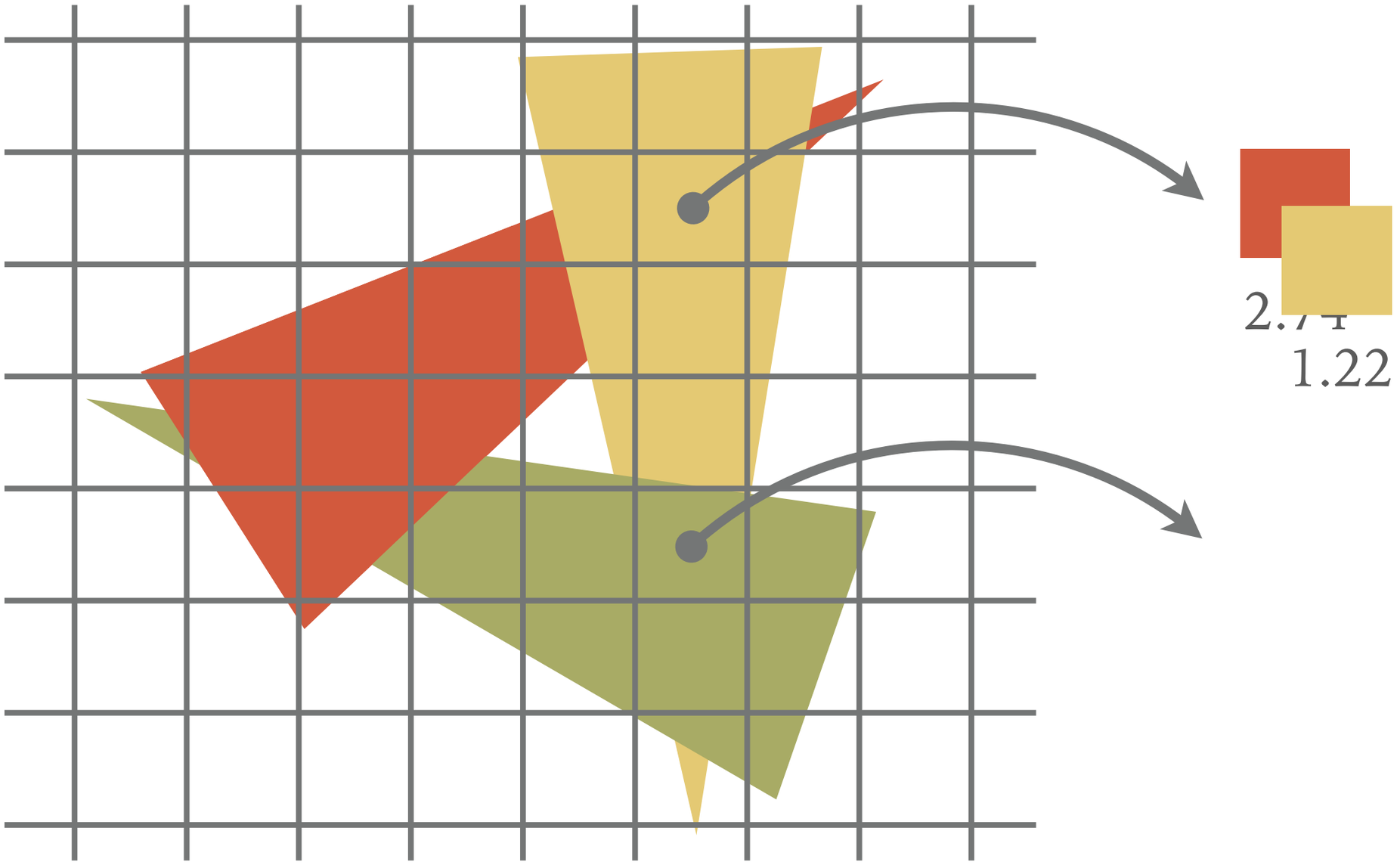
# Painter's Algorithm mit Screen Subdivision



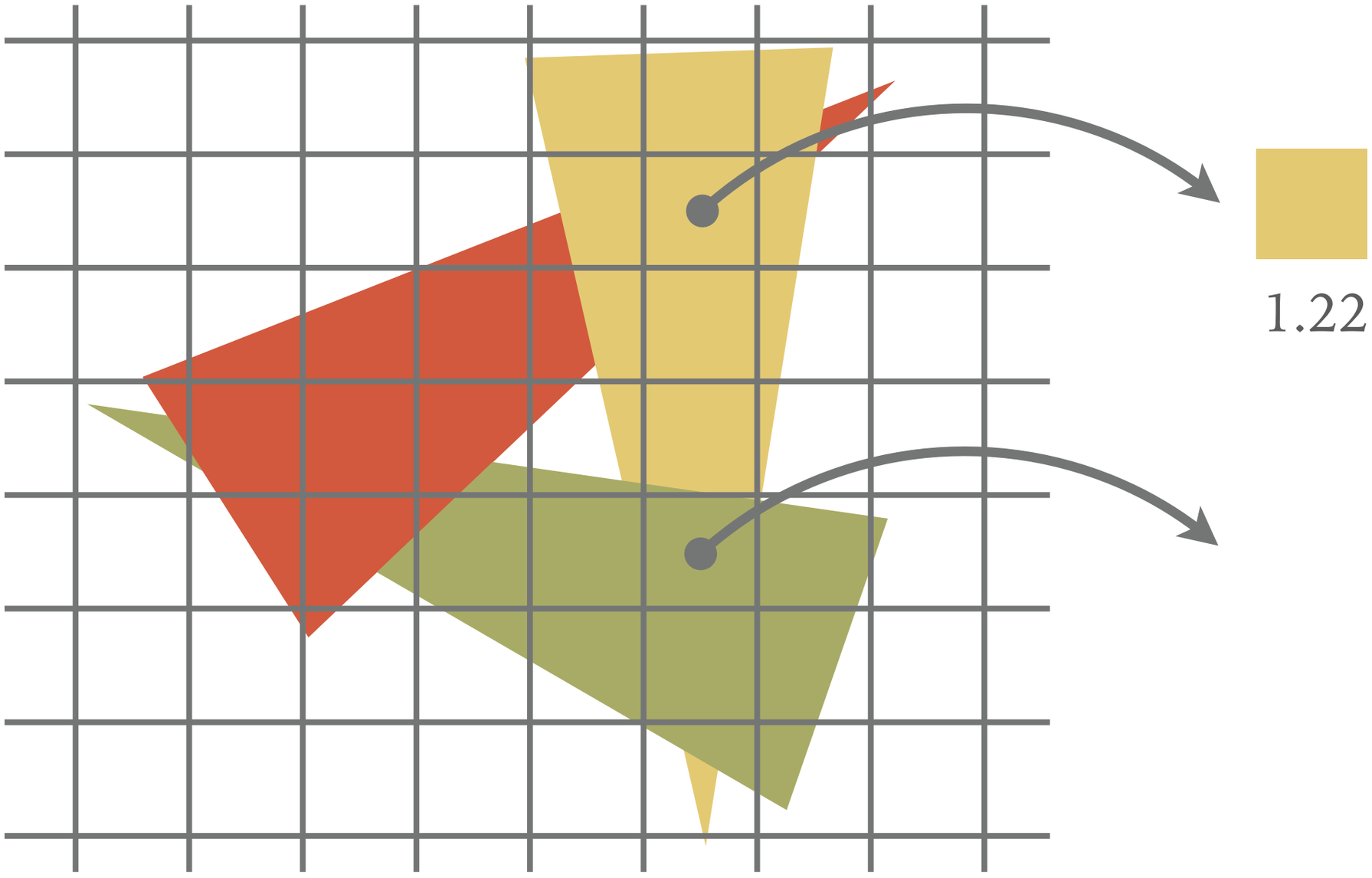
# Painter's Algorithm mit Screen Subdivision



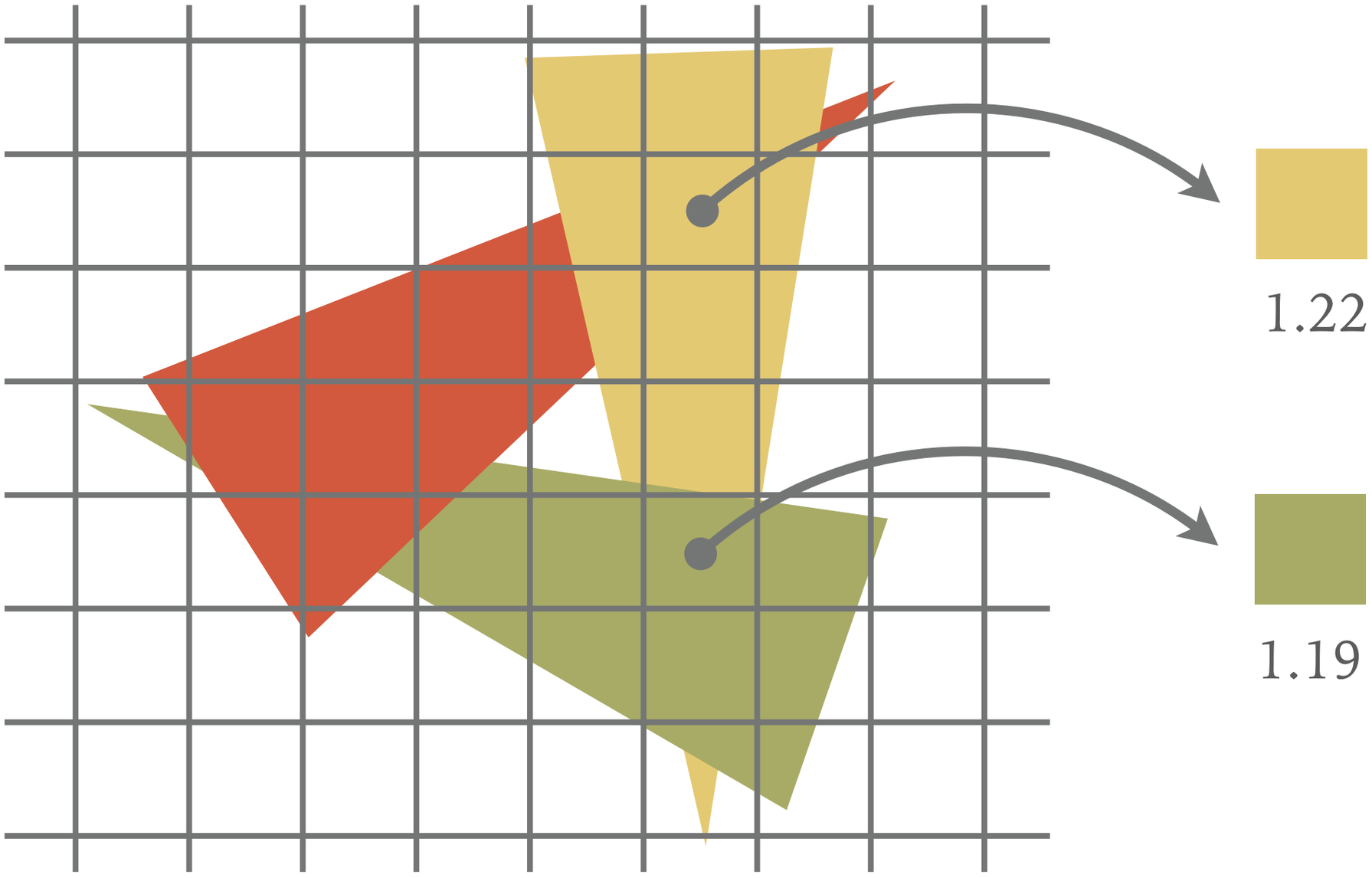
# Painter's Algorithm mit Screen Subdivision



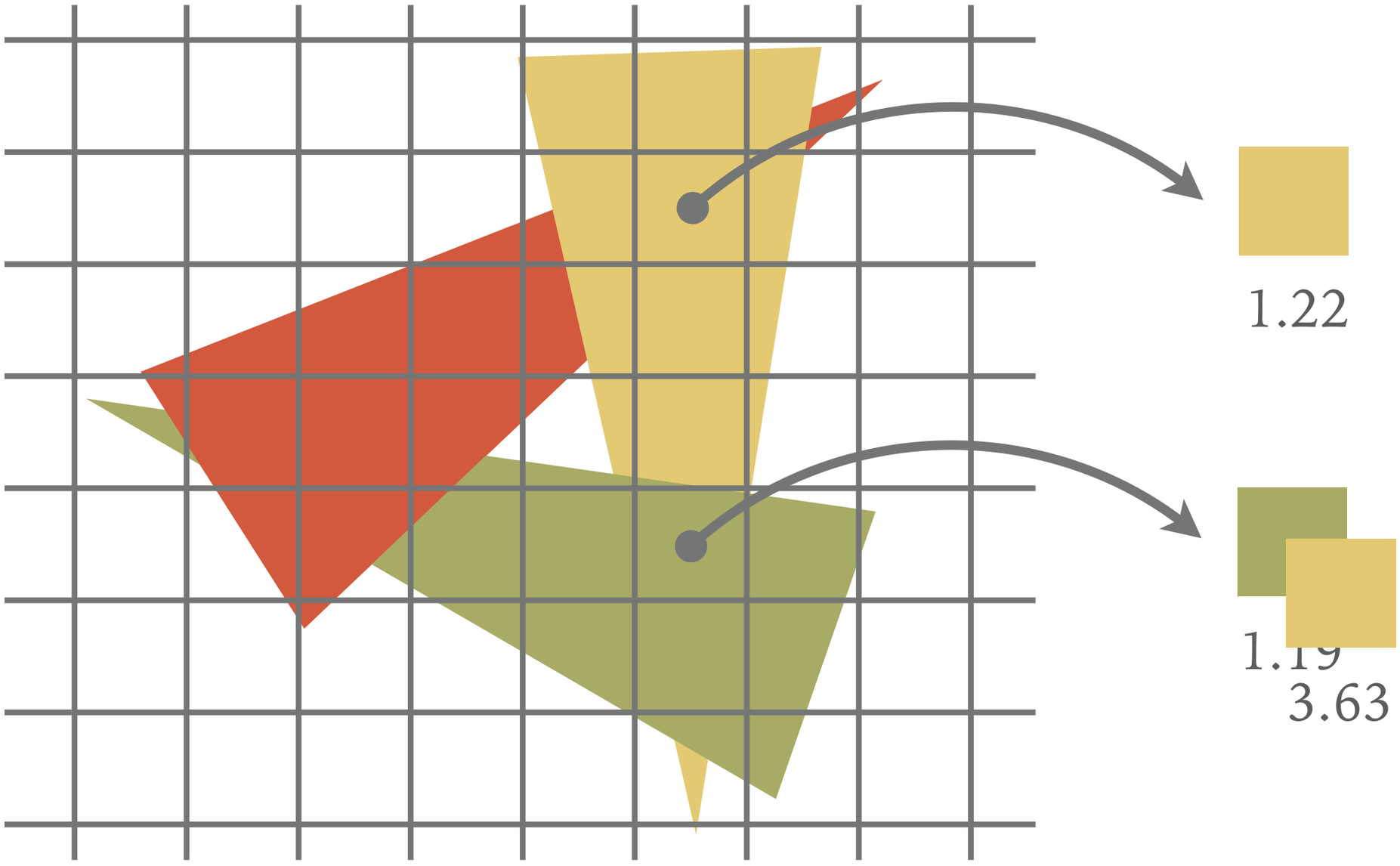
# Painter's Algorithm mit Screen Subdivision



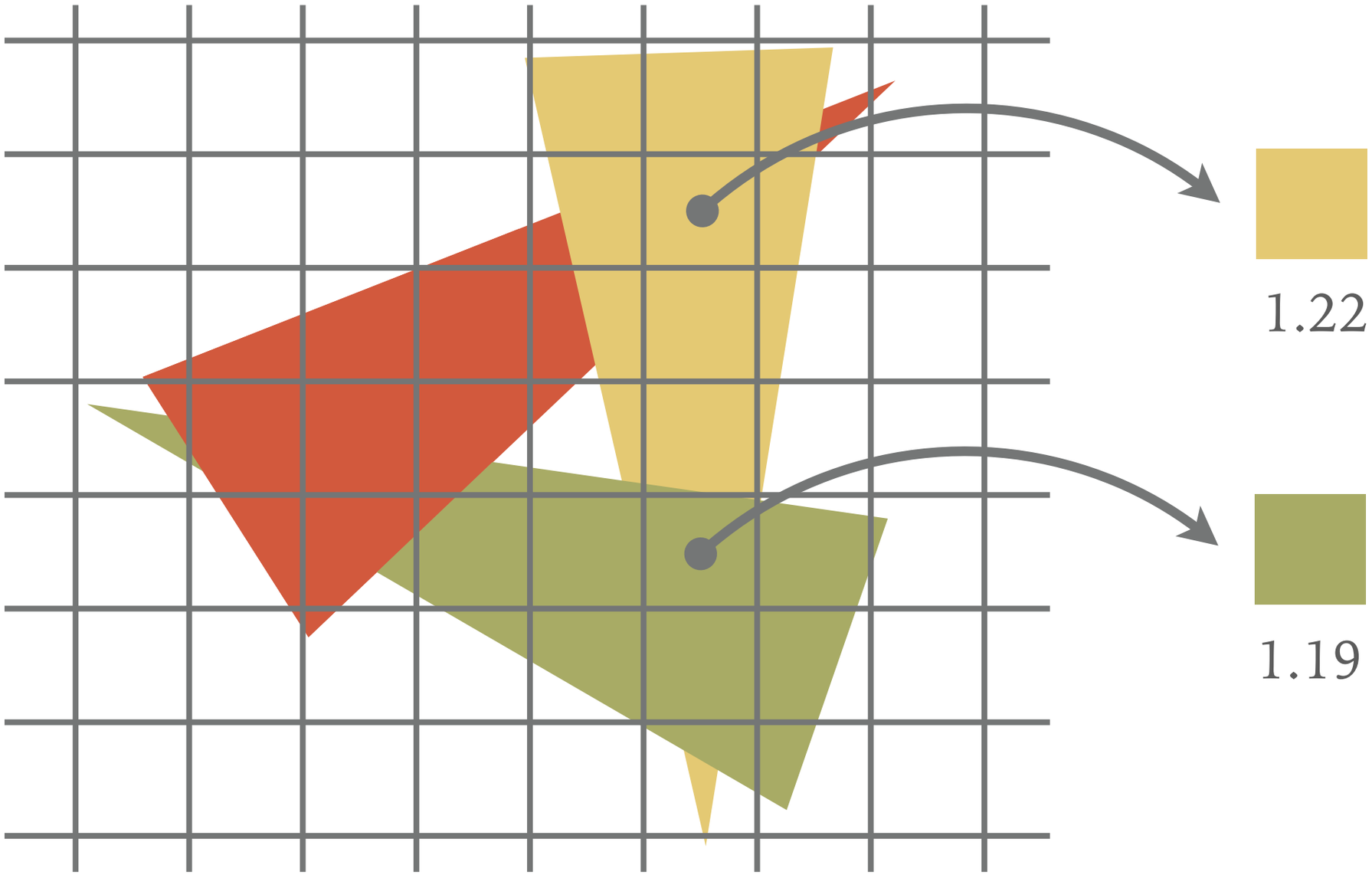
# Painter's Algorithm mit Screen Subdivision



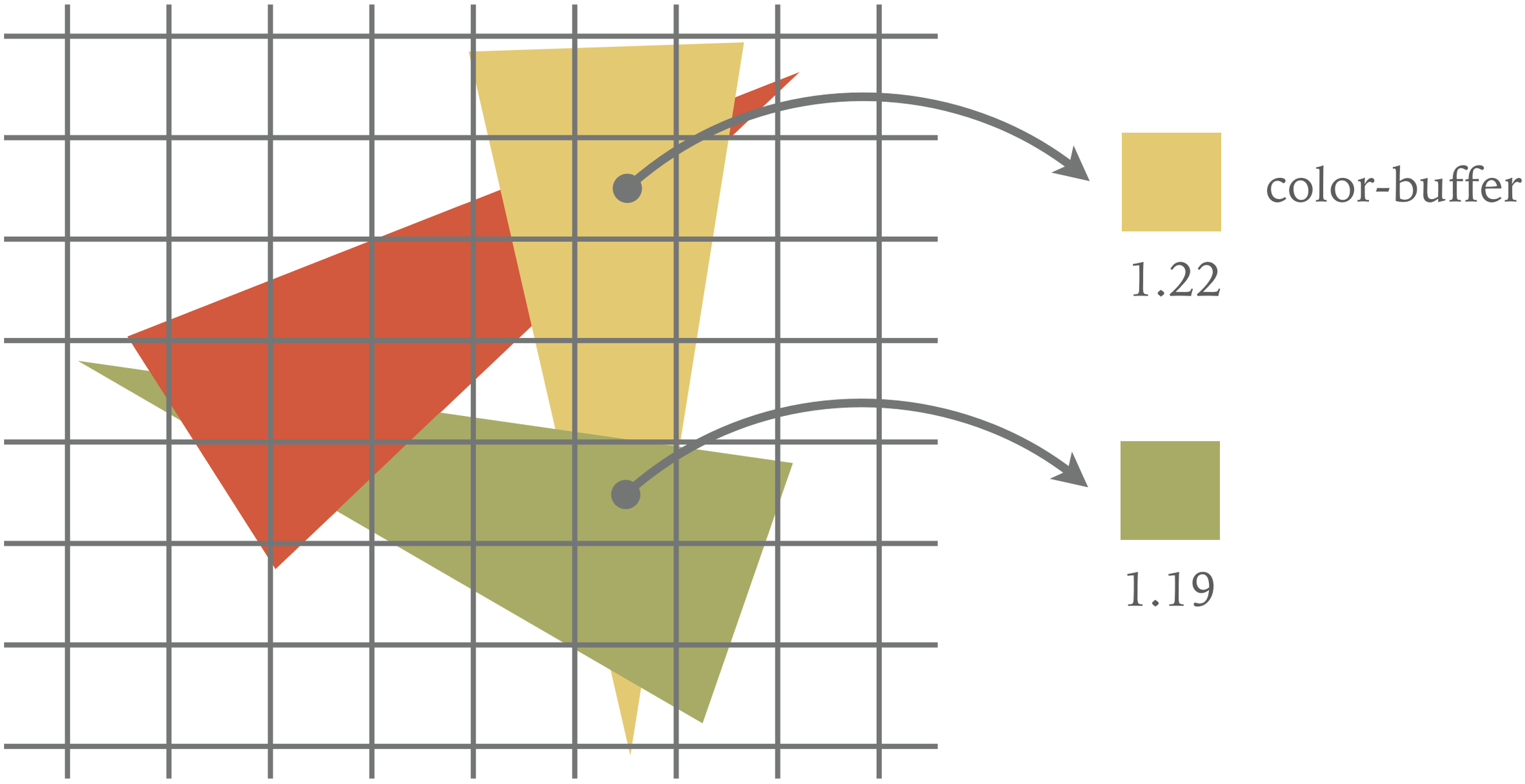
# Painter's Algorithm mit Screen Subdivision



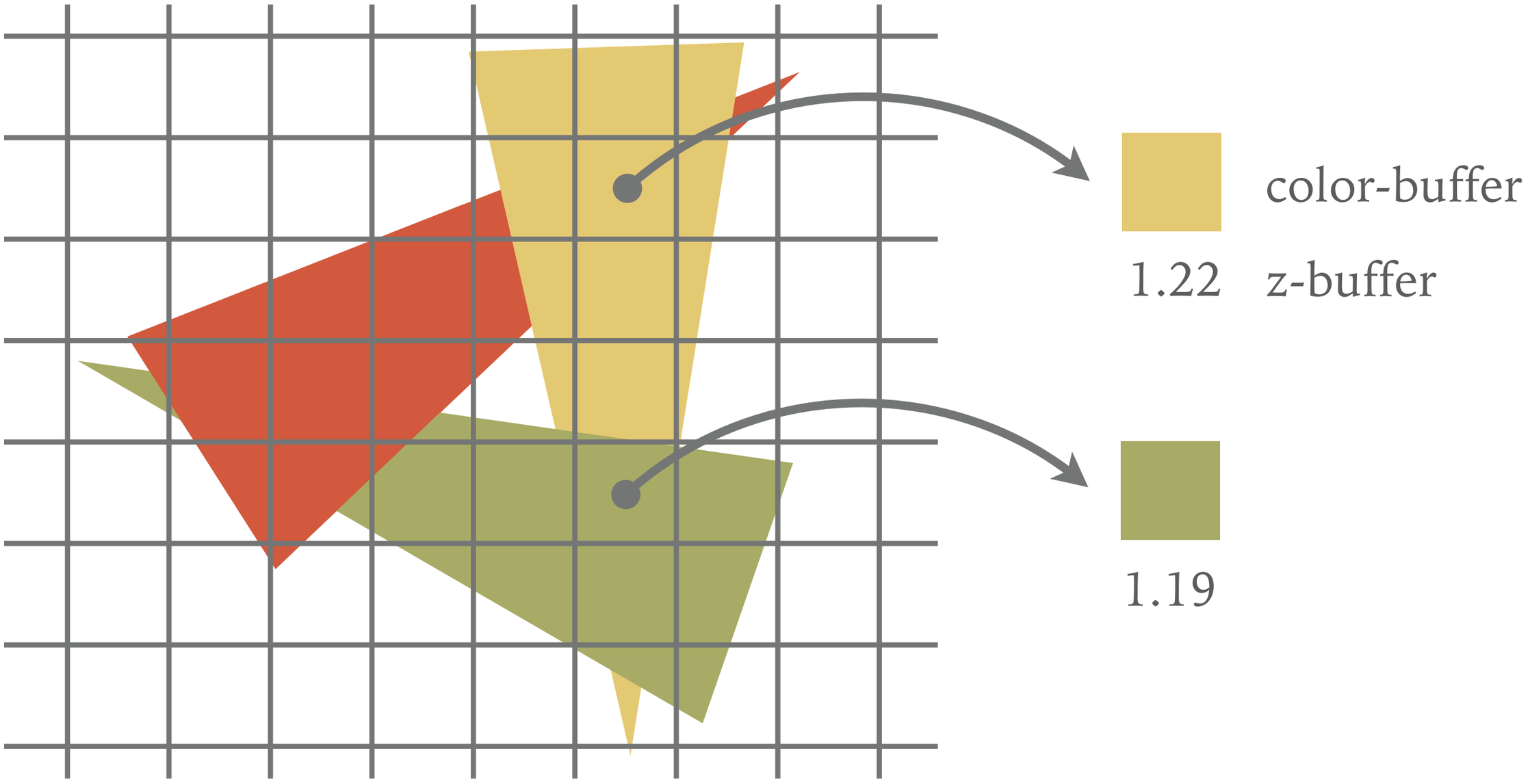
# Painter's Algorithm mit Screen Subdivision



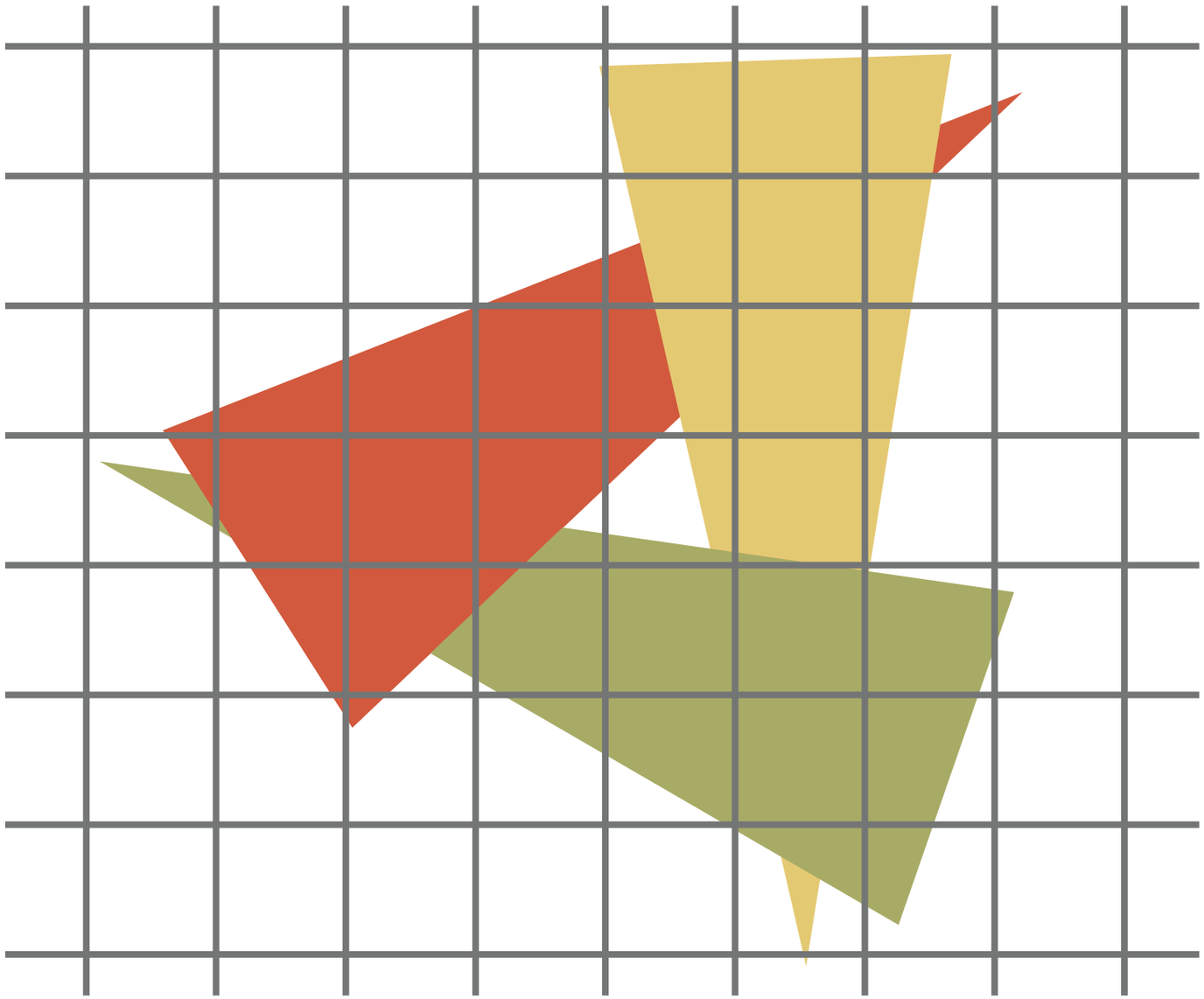
# Painter's Algorithm mit Screen Subdivision



# Painter's Algorithm mit Screen Subdivision



# Z-Buffer Algorithm



**for each** pixel $p$

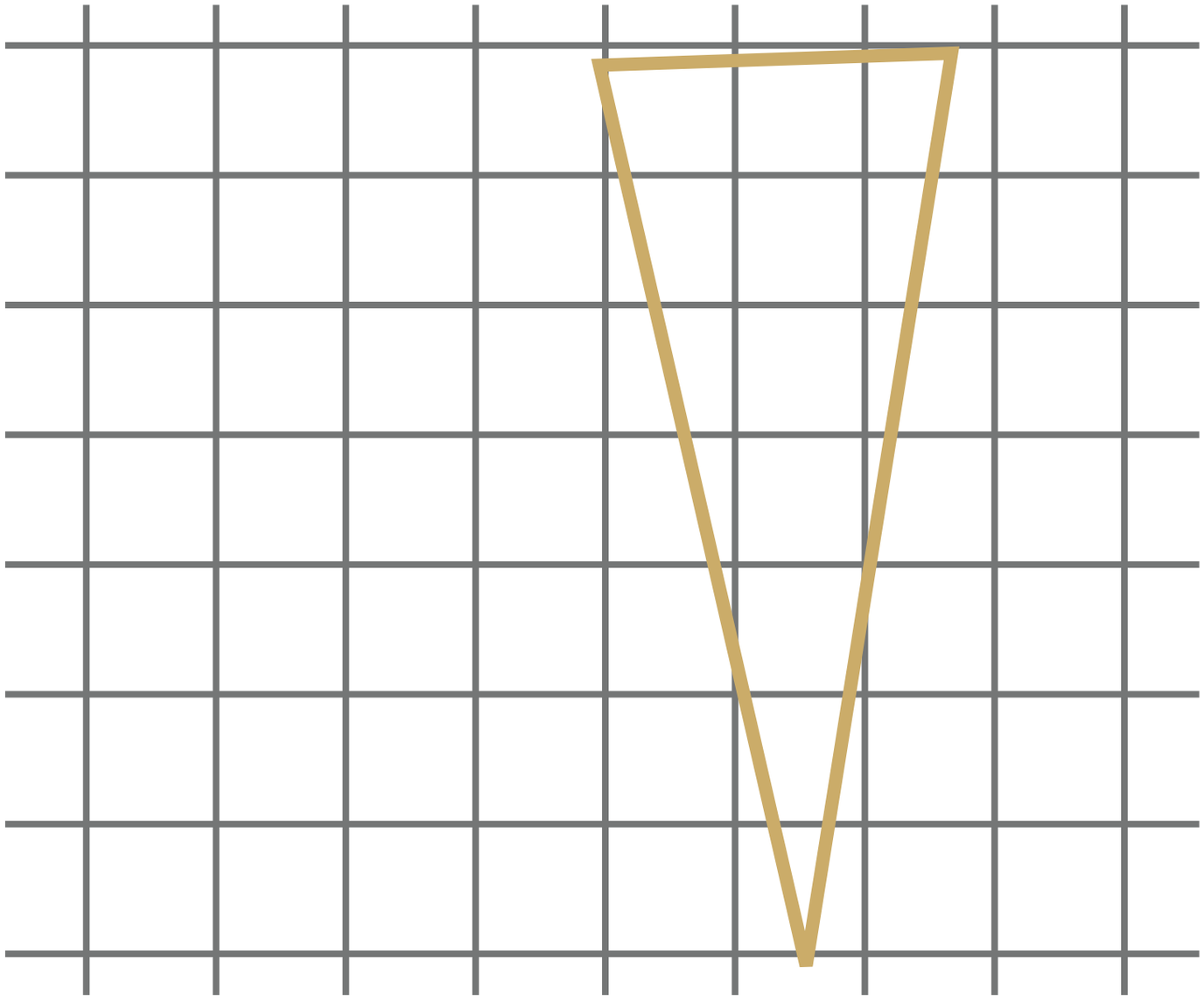
  **for each** object $o$

    **if** $z(o,p) < z\text{-buffer}[p]$

      $z\text{-buffer}[p] = z(o,p)$

      $\text{color}[p] = o.\text{color}$

# Z-Buffer Algorithm



**for each** object $o$

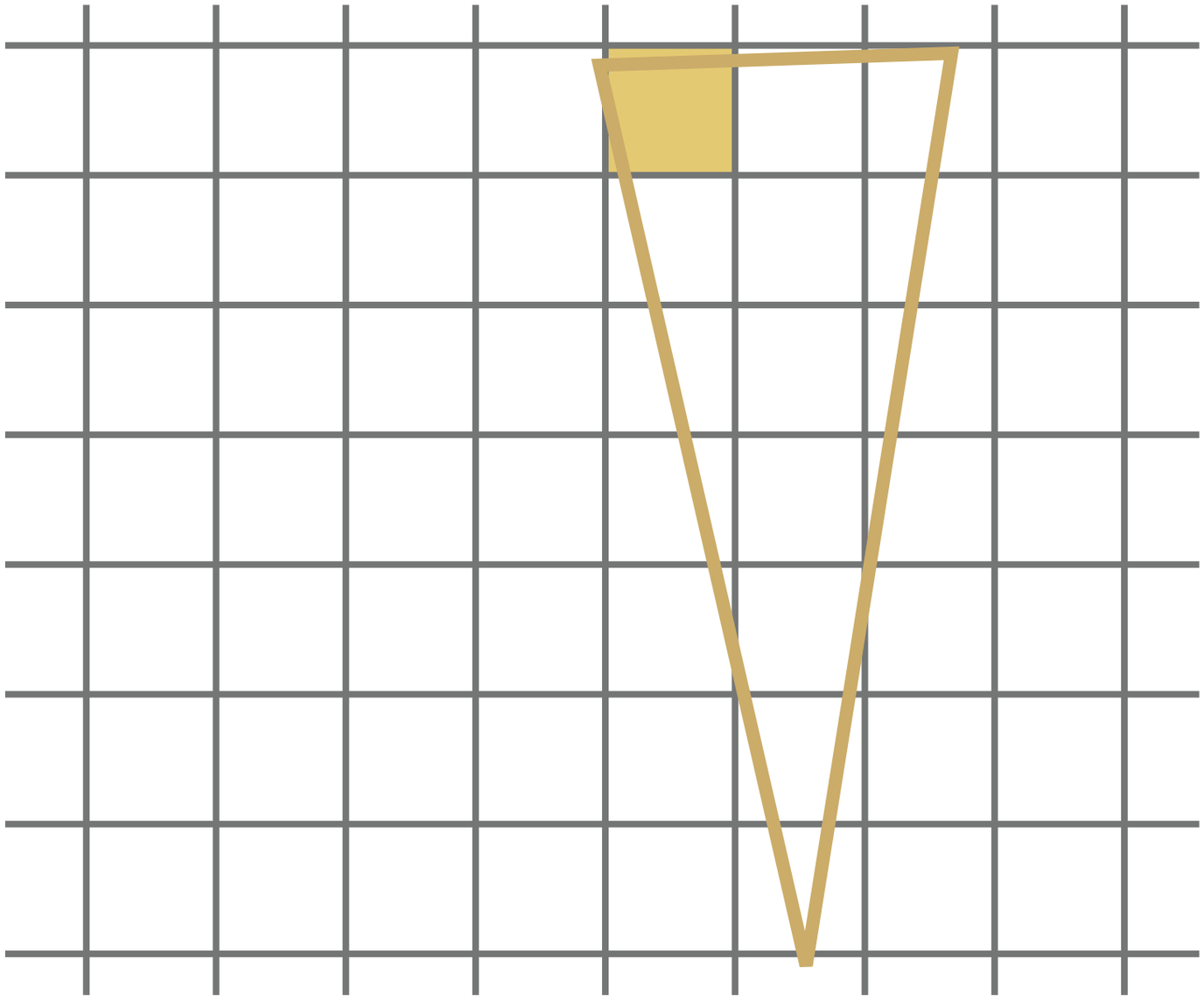
  **for each** pixel $p$

    **if** $z(o,p) < \text{z-buffer}[p]$

      $\text{z-buffer}[p] = z(o,p)$

      $\text{color}[p] = o.\text{color}$

# Z-Buffer Algorithm



**for each** object $o$

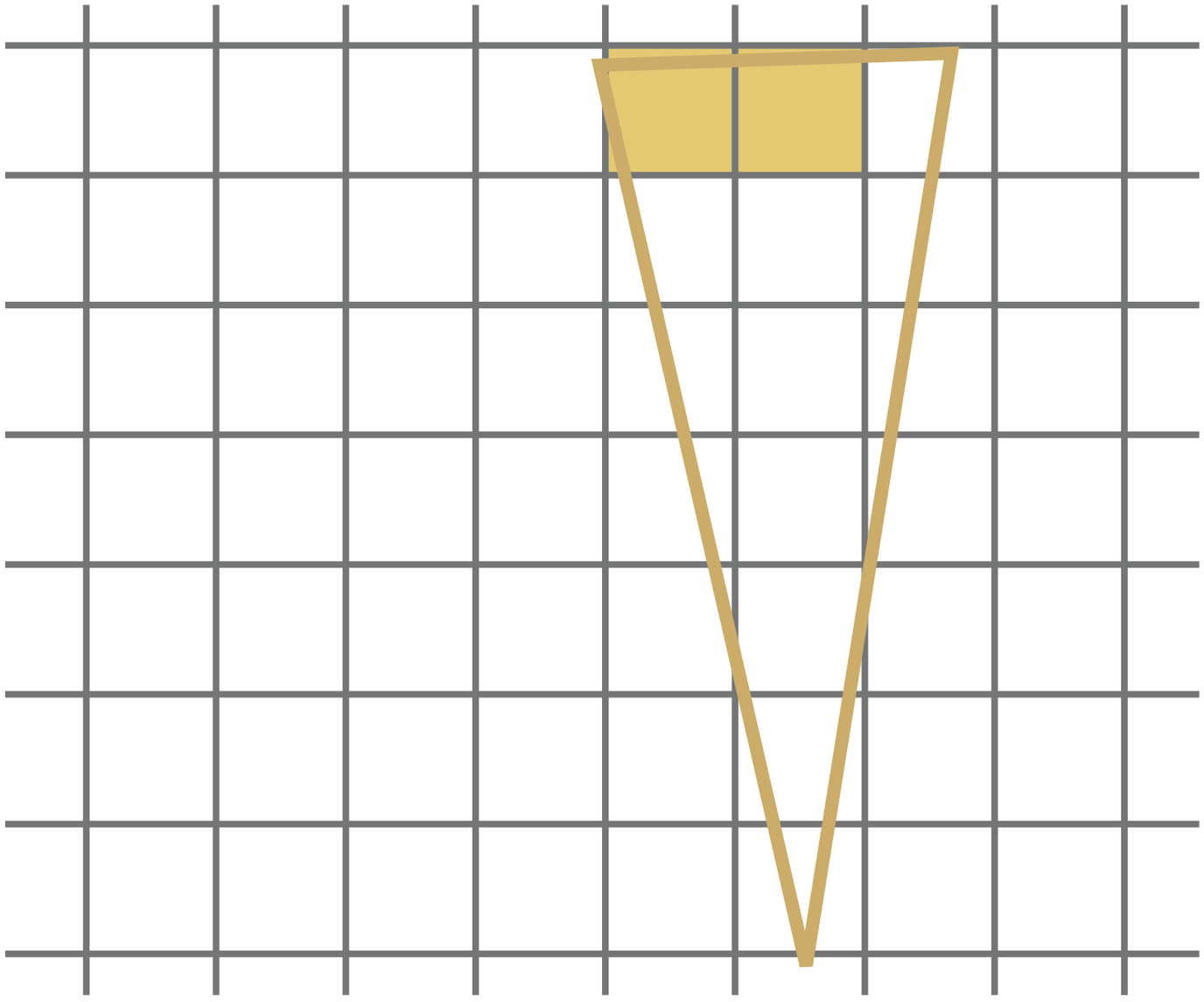
  **for each** pixel $p$

    **if** $z(o,p) < \text{z-buffer}[p]$

      $\text{z-buffer}[p] = z(o,p)$

      $\text{color}[p] = o.\text{color}$

# Z-Buffer Algorithm



**for each** object $o$

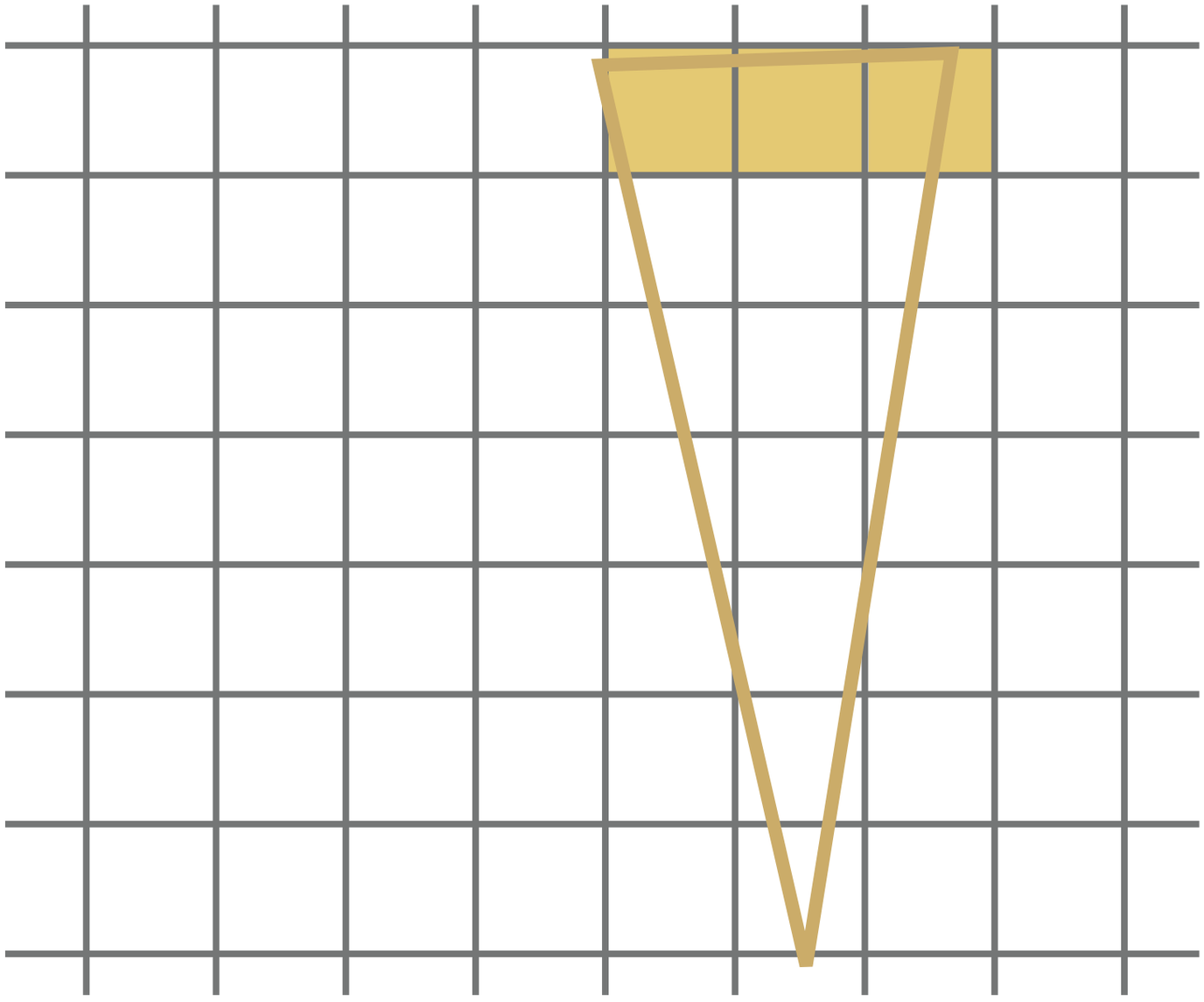
  **for each** pixel $p$

    **if** $z(o,p) < \text{z-buffer}[p]$

      $\text{z-buffer}[p] = z(o,p)$

      $\text{color}[p] = o.\text{color}$

# Z-Buffer Algorithm



**for each** object $o$

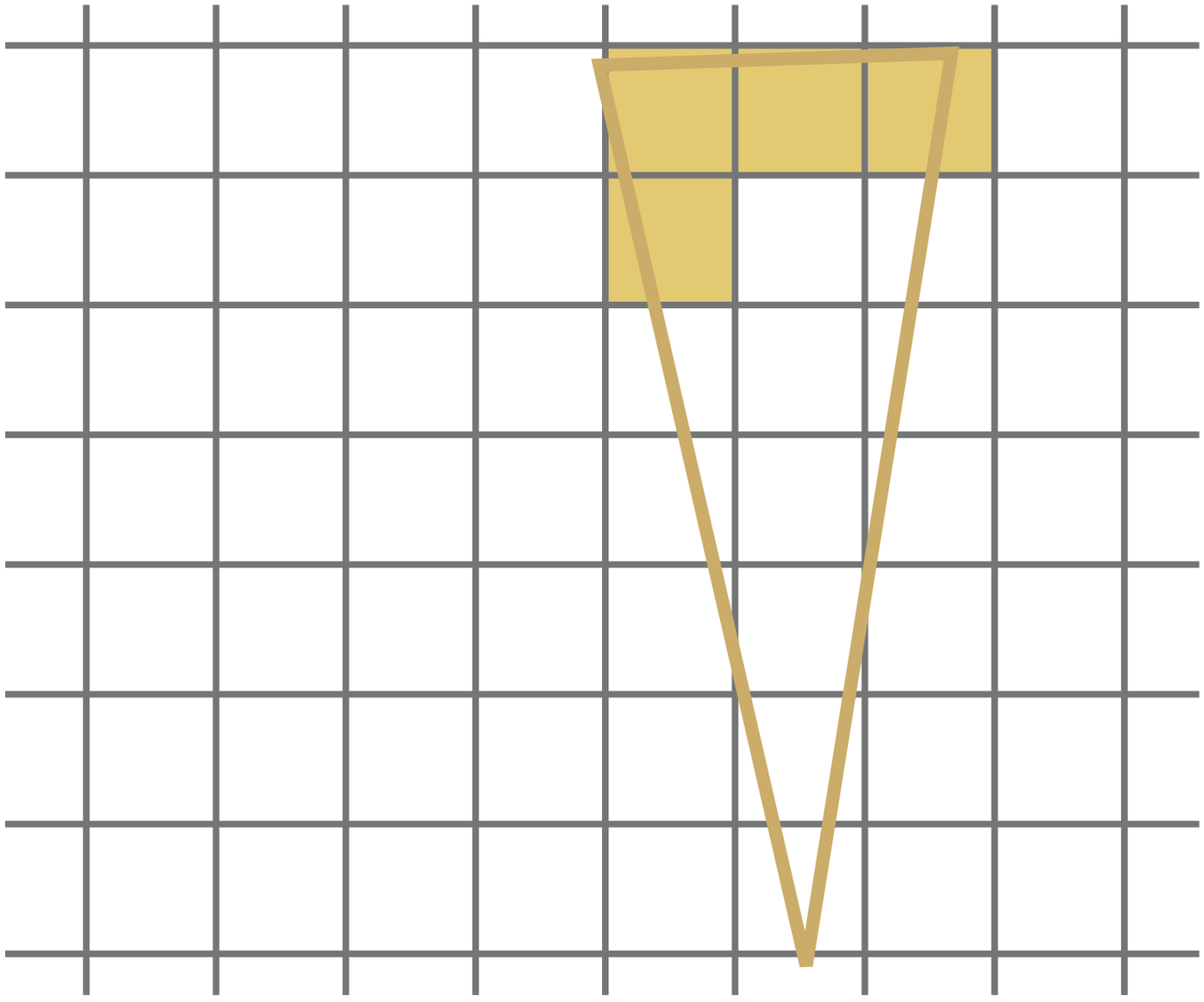
  **for each** pixel $p$

    **if** $z(o,p) < \text{z-buffer}[p]$

      $\text{z-buffer}[p] = z(o,p)$

      $\text{color}[p] = o.\text{color}$

# Z-Buffer Algorithm



**for each** object $o$

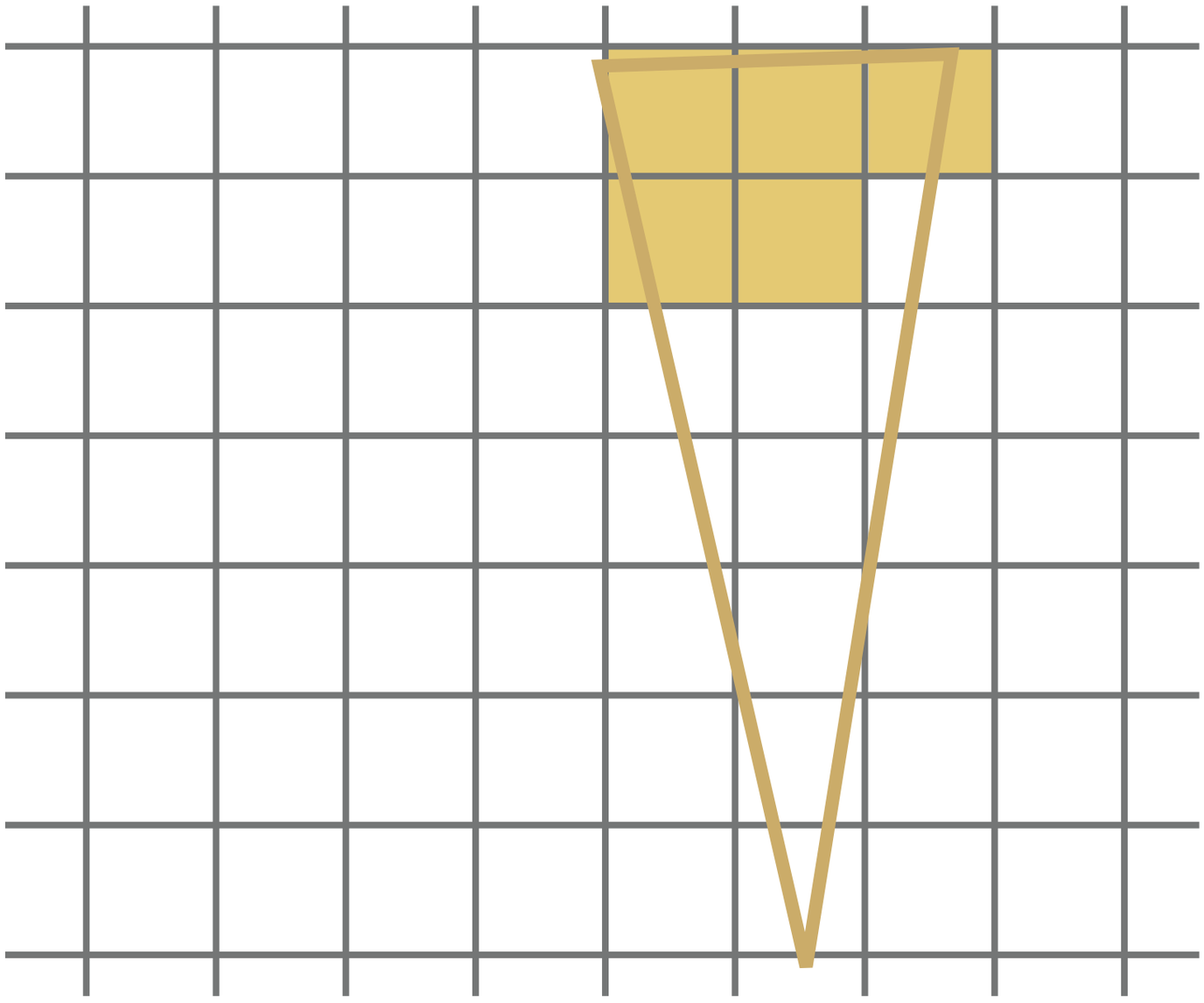
  **for each** pixel $p$

    **if** $z(o,p) < \text{z-buffer}[p]$

      $\text{z-buffer}[p] = z(o,p)$

      $\text{color}[p] = o.\text{color}$

# Z-Buffer Algorithm



**for each** object $o$

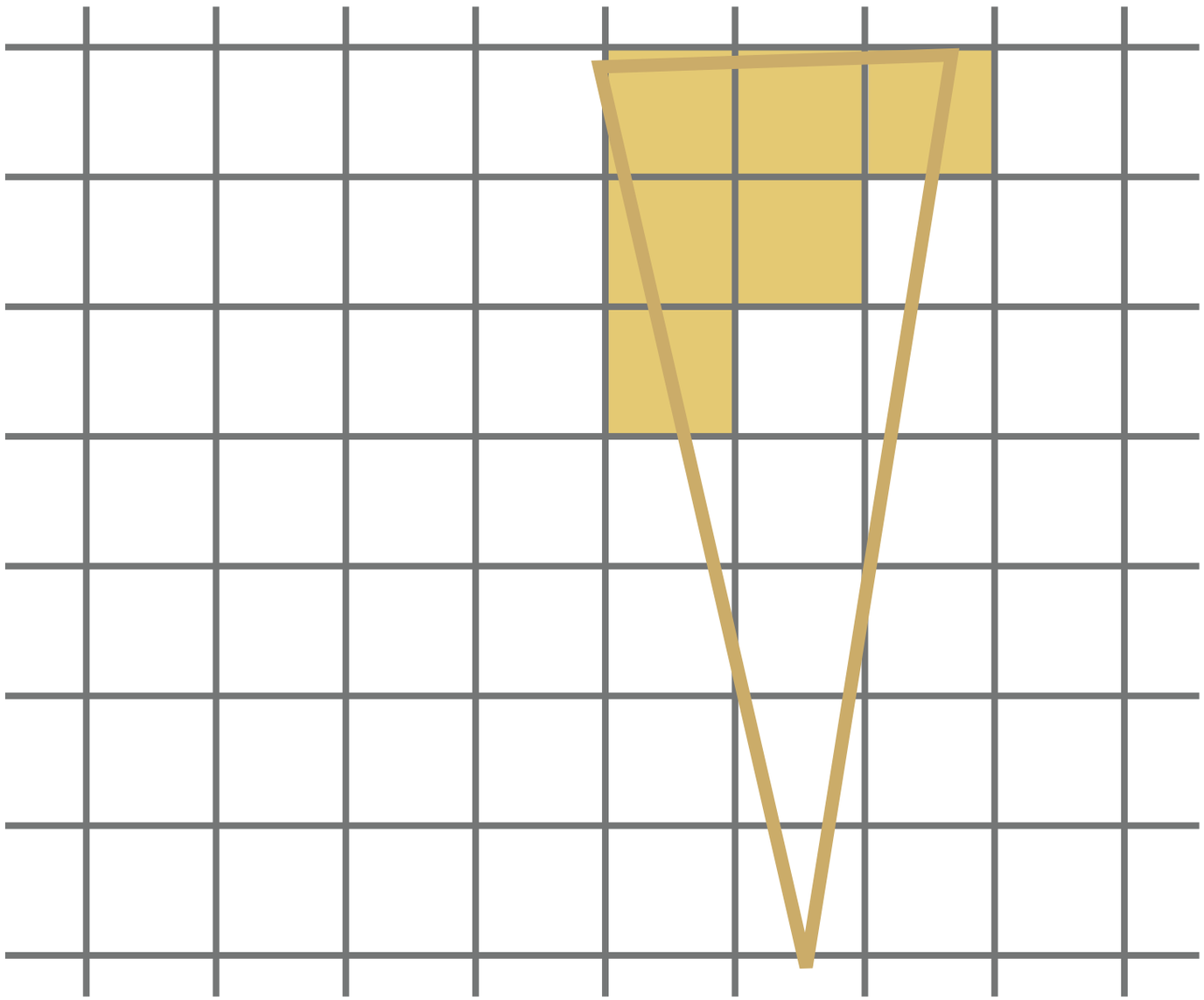
  **for each** pixel $p$

    **if** $z(o,p) < \text{z-buffer}[p]$

      $\text{z-buffer}[p] = z(o,p)$

      $\text{color}[p] = o.\text{color}$

# Z-Buffer Algorithm



**for each** object $o$

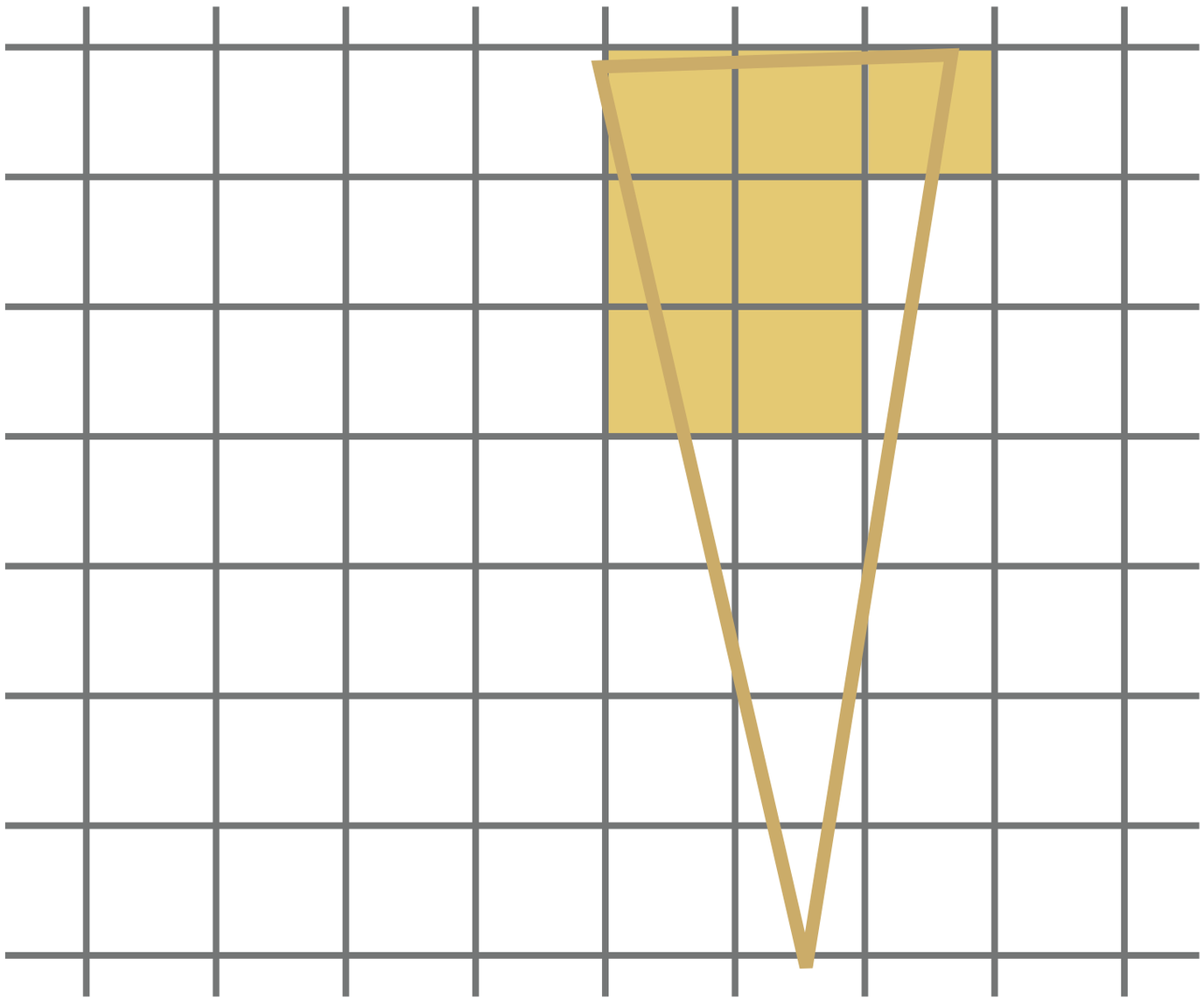
  **for each** pixel $p$

    **if** $z(o,p) < z\text{-buffer}[p]$

      $z\text{-buffer}[p] = z(o,p)$

      $\text{color}[p] = o.\text{color}$

# Z-Buffer Algorithm



**for each** object $o$

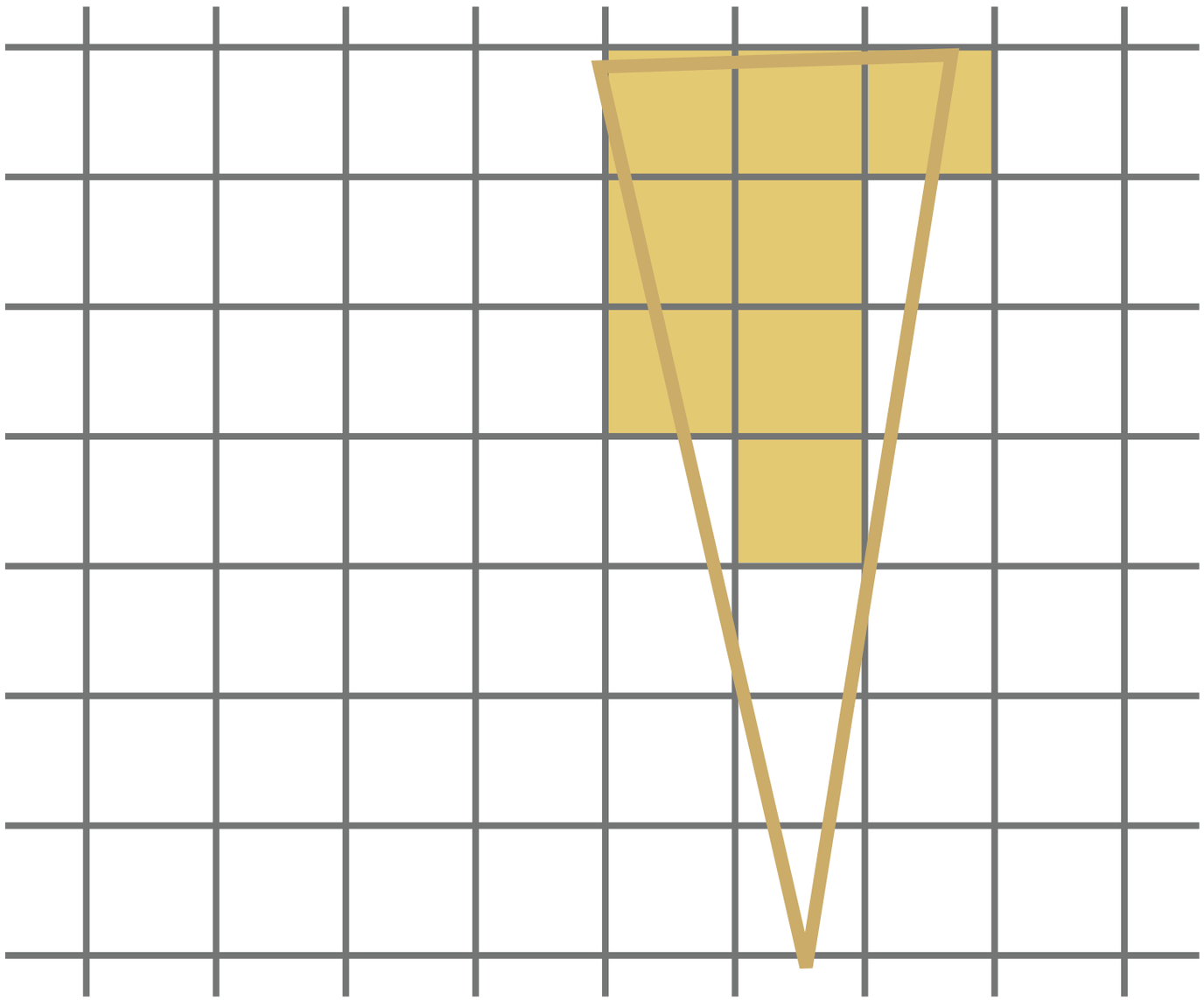
  **for each** pixel $p$

    **if** $z(o,p) < z\text{-buffer}[p]$

      $z\text{-buffer}[p] = z(o,p)$

      $\text{color}[p] = o.\text{color}$

# Z-Buffer Algorithm



**for each** object $o$

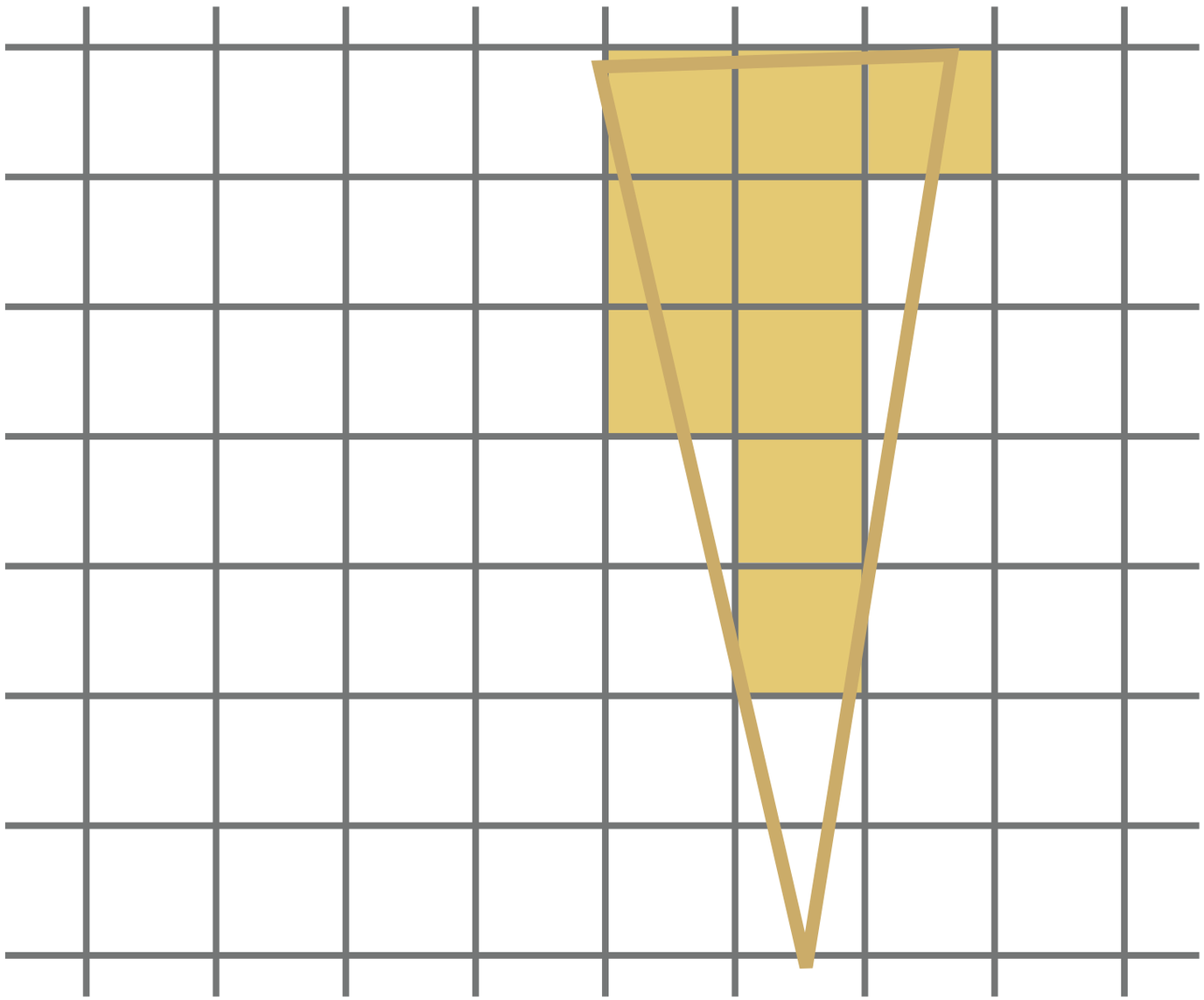
  **for each** pixel $p$

    **if** $z(o,p) < z\text{-buffer}[p]$

      $z\text{-buffer}[p] = z(o,p)$

      $\text{color}[p] = o.\text{color}$

# Z-Buffer Algorithm



**for each** object $o$

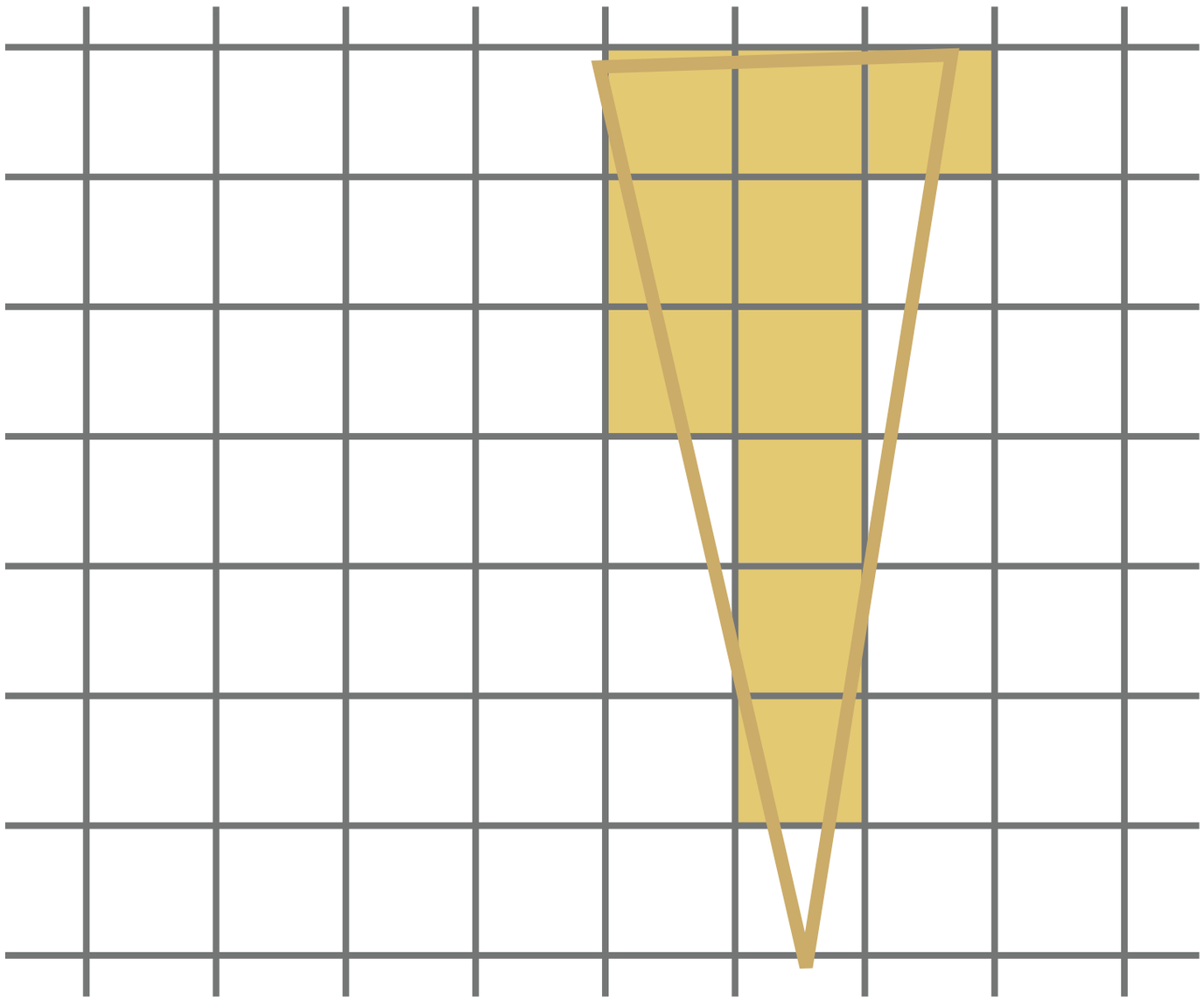
  **for each** pixel $p$

    **if** $z(o,p) < z\text{-buffer}[p]$

      $z\text{-buffer}[p] = z(o,p)$

      $\text{color}[p] = o.\text{color}$

# Z-Buffer Algorithm



**for each** object $o$

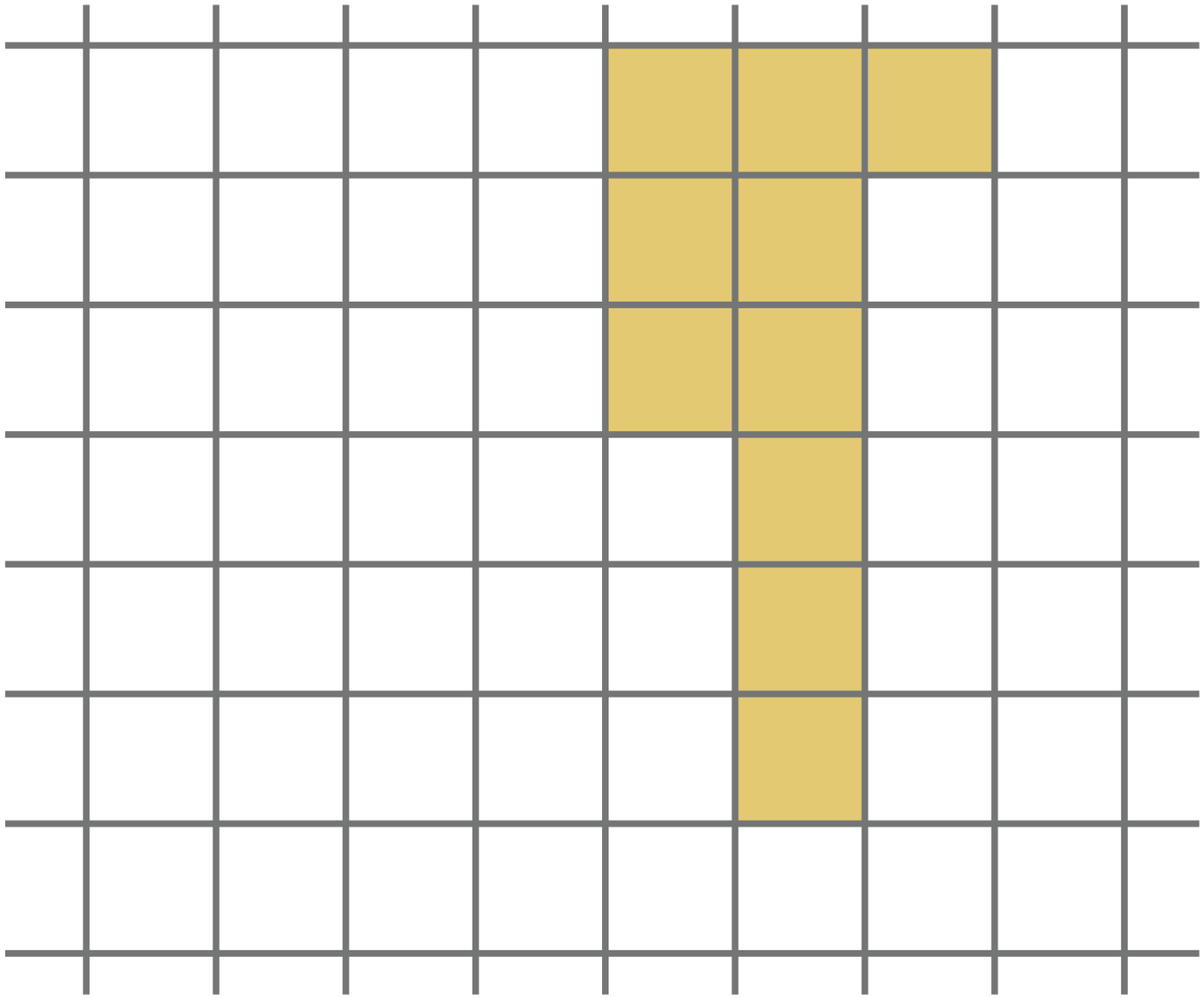
  **for each** pixel $p$

    **if** $z(o,p) < z\text{-buffer}[p]$

      $z\text{-buffer}[p] = z(o,p)$

      $\text{color}[p] = o.\text{color}$

# Z-Buffer Algorithm



**for each** object $o$

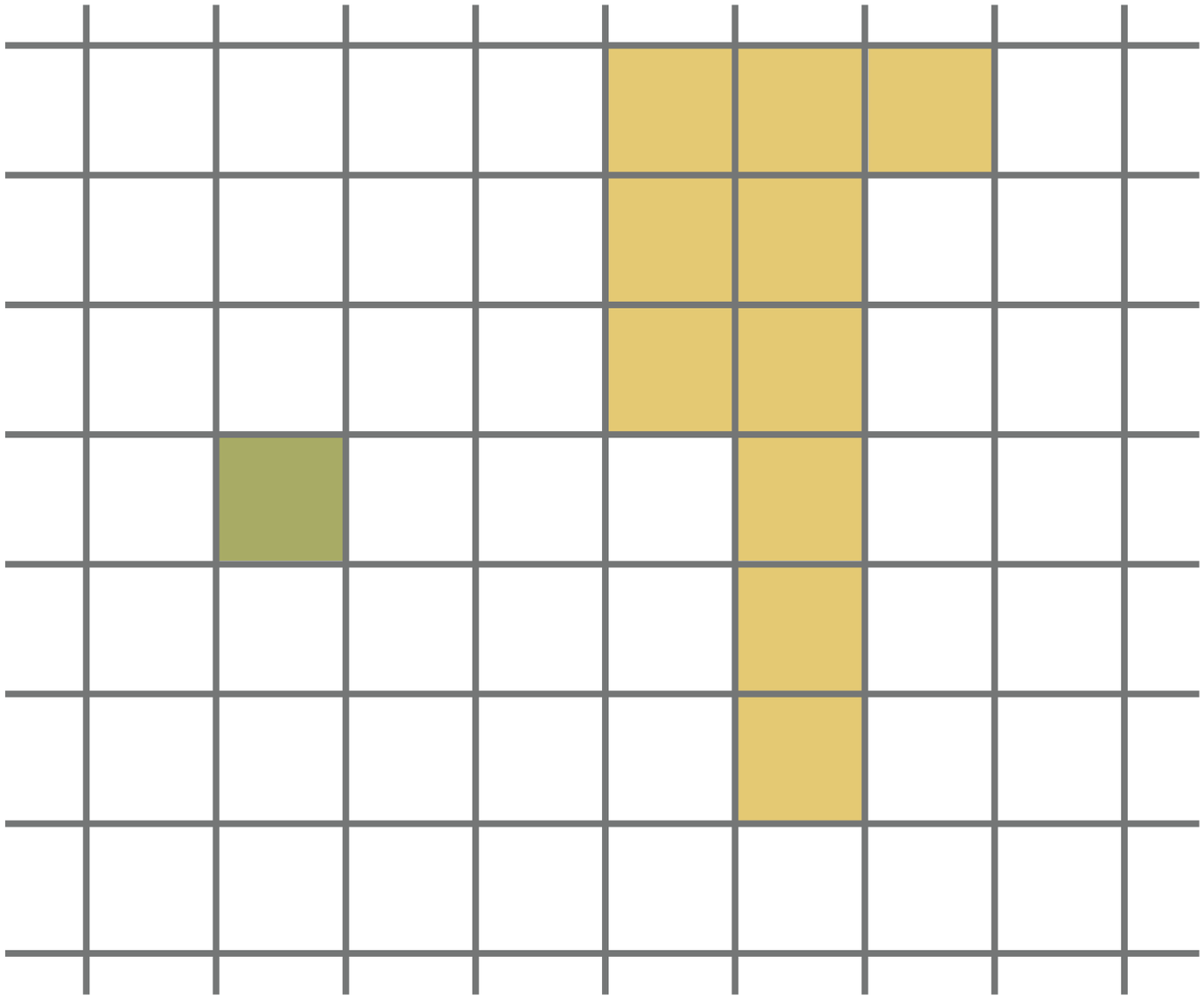
  **for each** pixel $p$

    **if** $z(o,p) < z\text{-buffer}[p]$

      $z\text{-buffer}[p] = z(o,p)$

      $\text{color}[p] = o.\text{color}$

# Z-Buffer Algorithm



**for each** object $o$

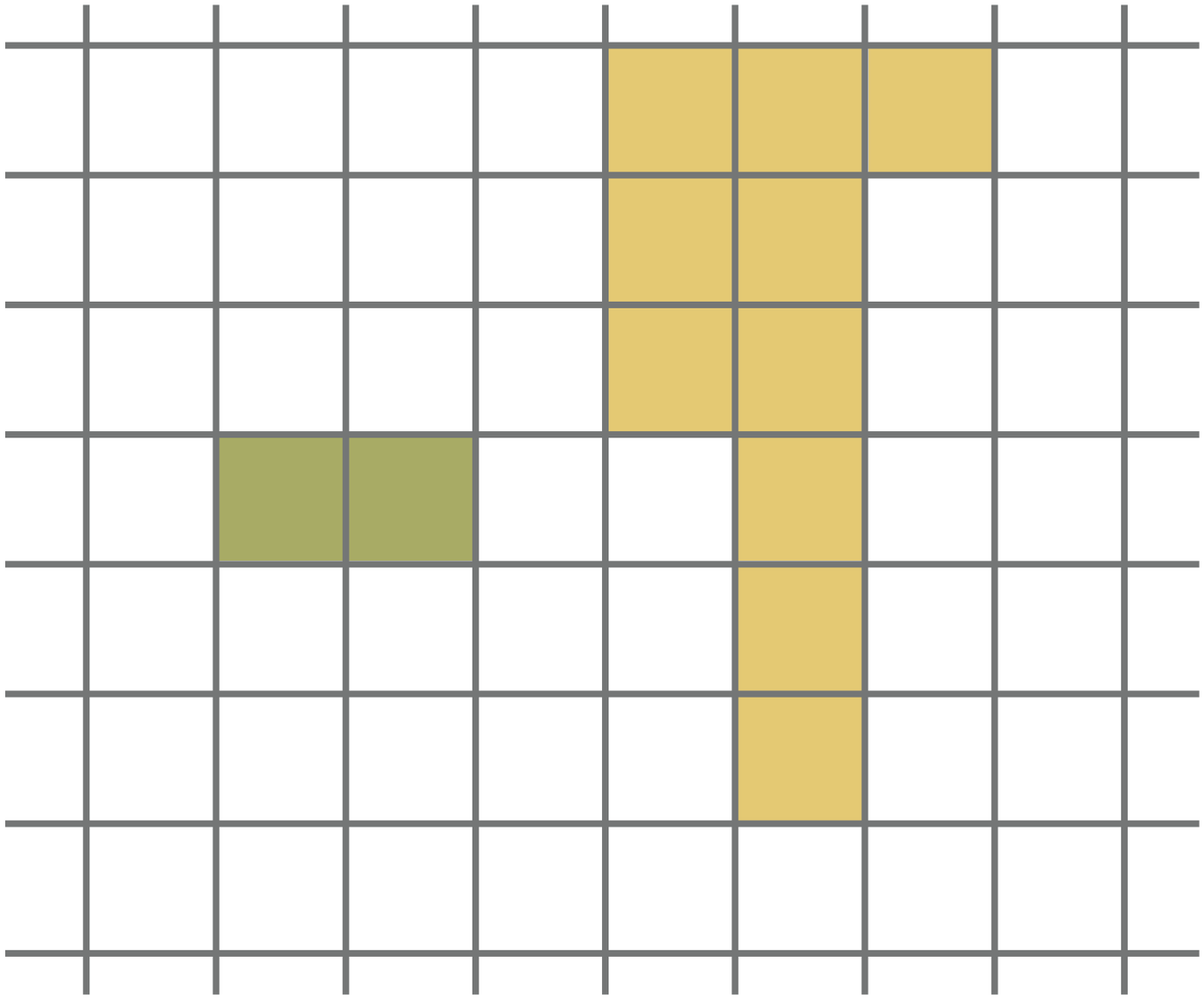
  **for each** pixel $p$

    **if** $z(o,p) < z\text{-buffer}[p]$

      $z\text{-buffer}[p] = z(o,p)$

      $\text{color}[p] = o.\text{color}$

# Z-Buffer Algorithm



**for each** object $o$

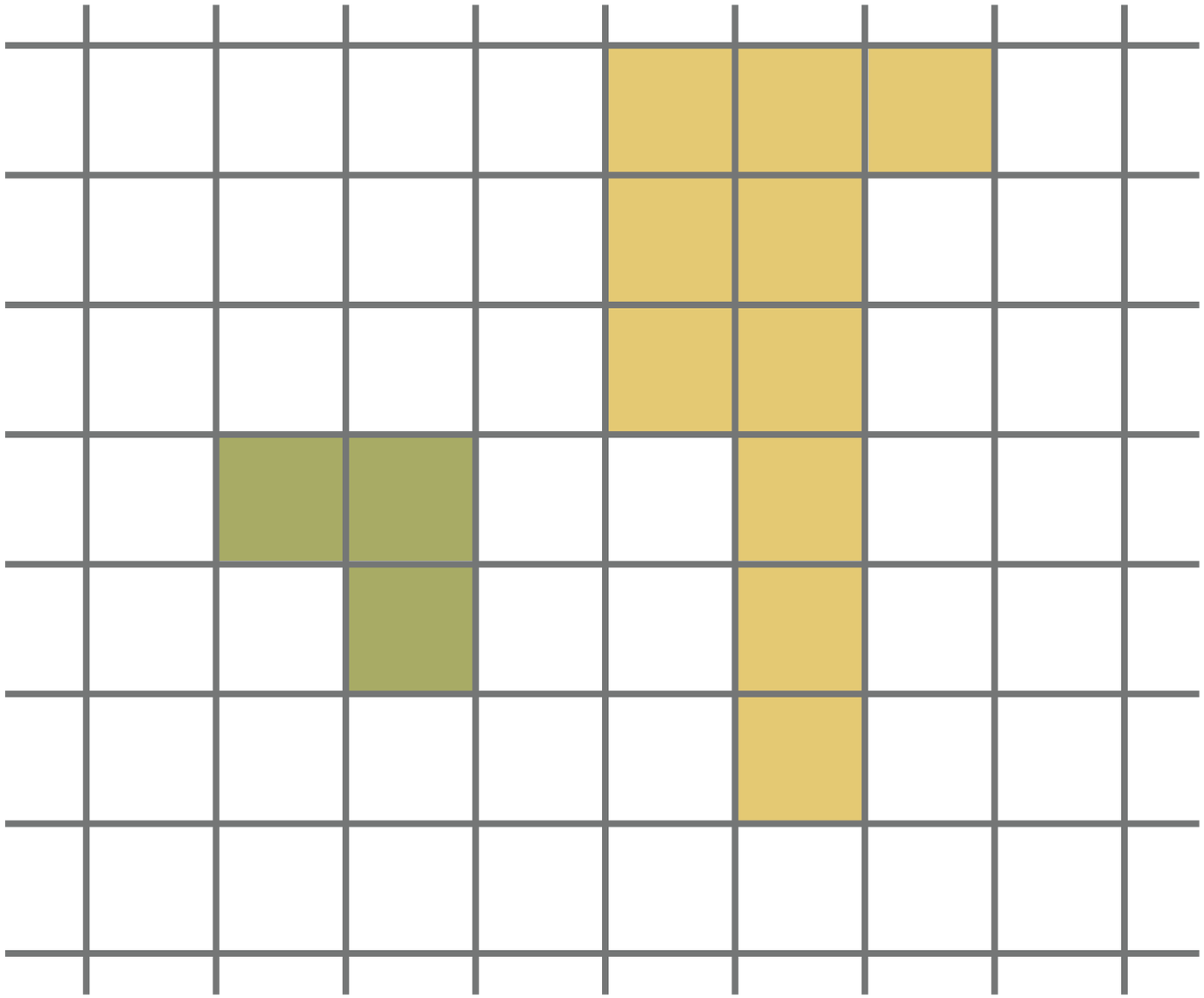
  **for each** pixel $p$

    **if** $z(o,p) < z\text{-buffer}[p]$

      $z\text{-buffer}[p] = z(o,p)$

      $\text{color}[p] = o.\text{color}$

# Z-Buffer Algorithm



**for each** object $o$

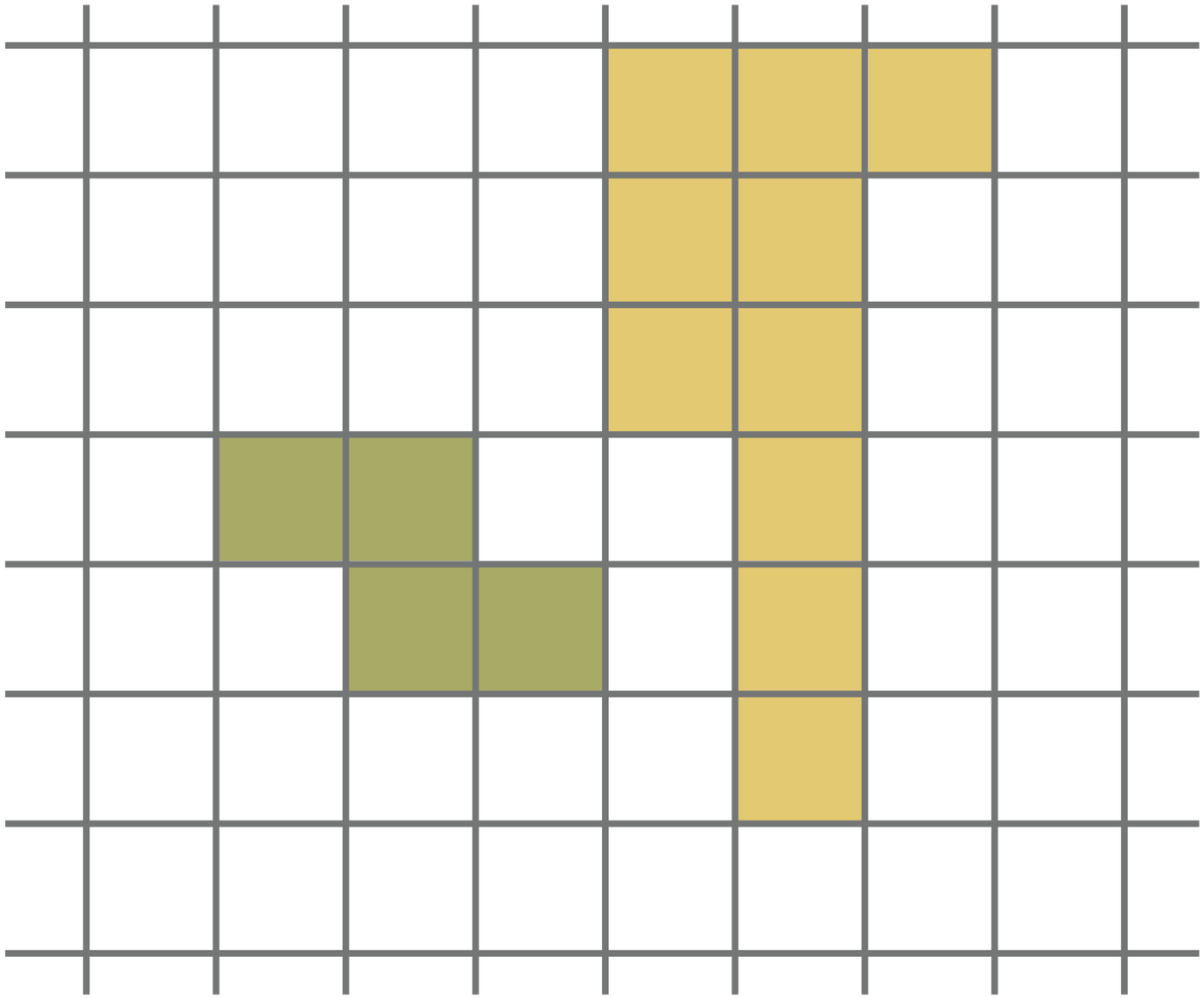
  **for each** pixel $p$

    **if** $z(o,p) < z\text{-buffer}[p]$

      $z\text{-buffer}[p] = z(o,p)$

      $\text{color}[p] = o.\text{color}$

# Z-Buffer Algorithm



**for each** object $o$

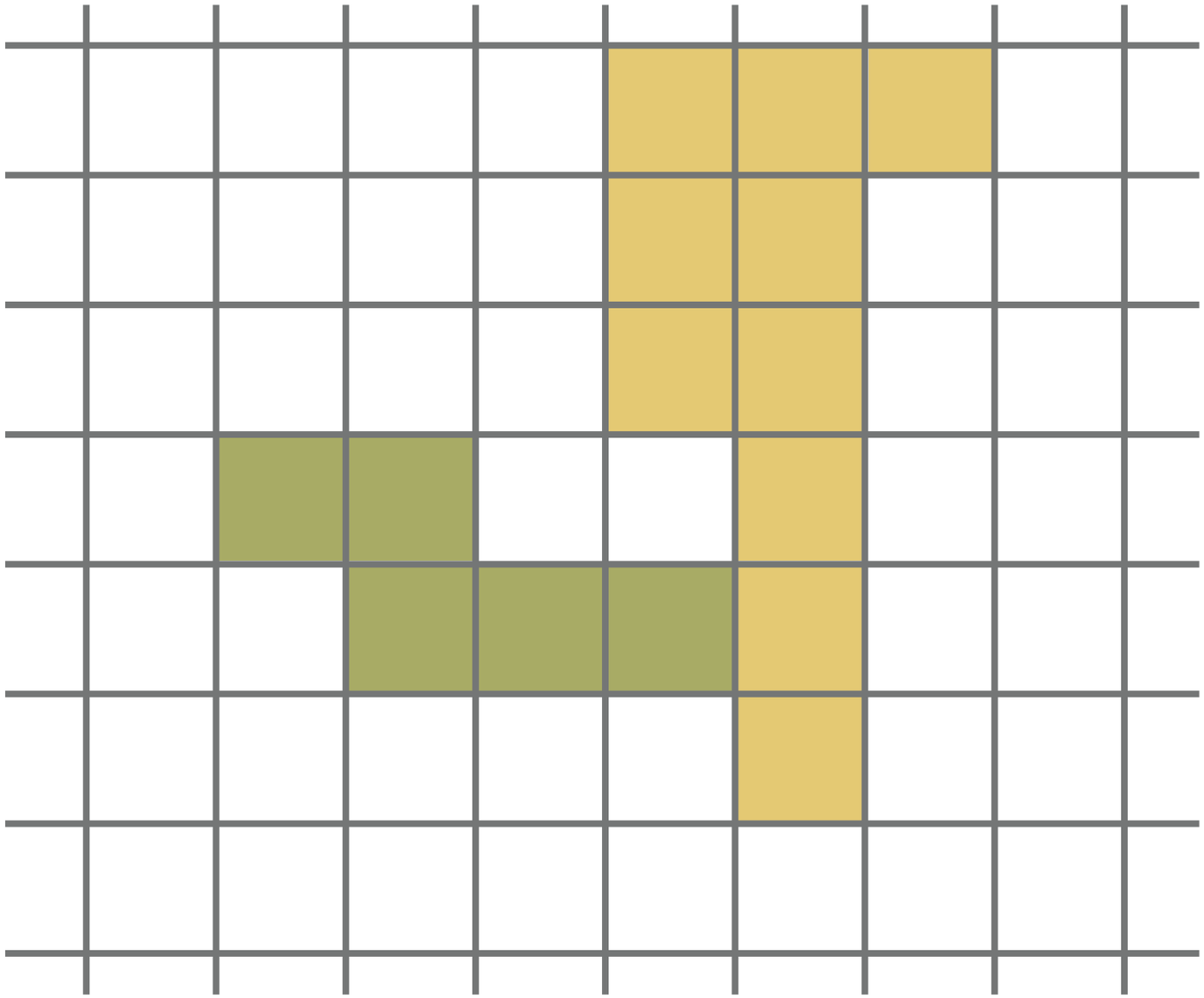
  **for each** pixel $p$

    **if** $z(o,p) < z\text{-buffer}[p]$

      $z\text{-buffer}[p] = z(o,p)$

      $\text{color}[p] = o.\text{color}$

# Z-Buffer Algorithm



**for each** object $o$

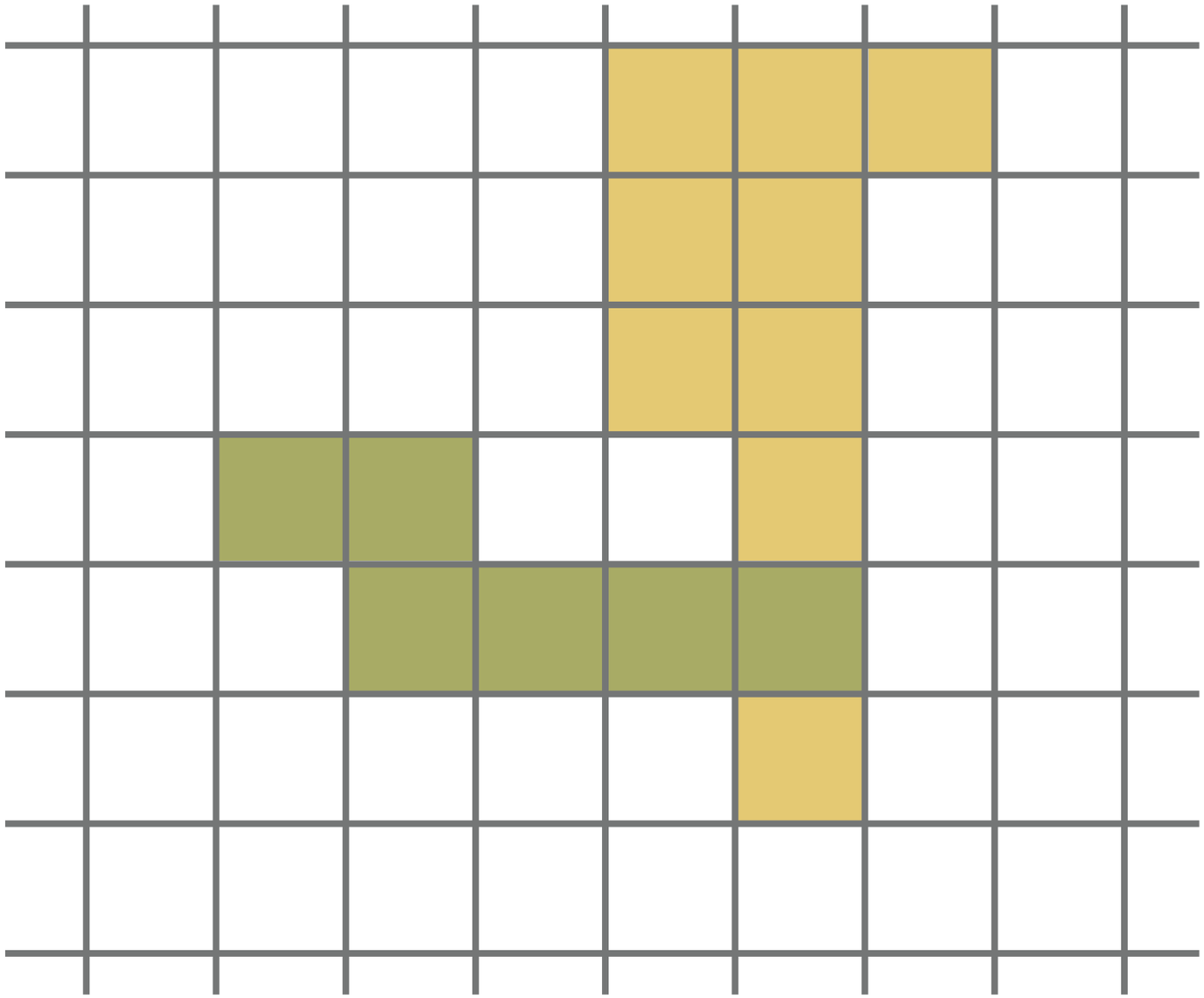
  **for each** pixel $p$

    **if** $z(o,p) < z\text{-buffer}[p]$

      $z\text{-buffer}[p] = z(o,p)$

      $\text{color}[p] = o.\text{color}$

# Z-Buffer Algorithm



**for each** object $o$

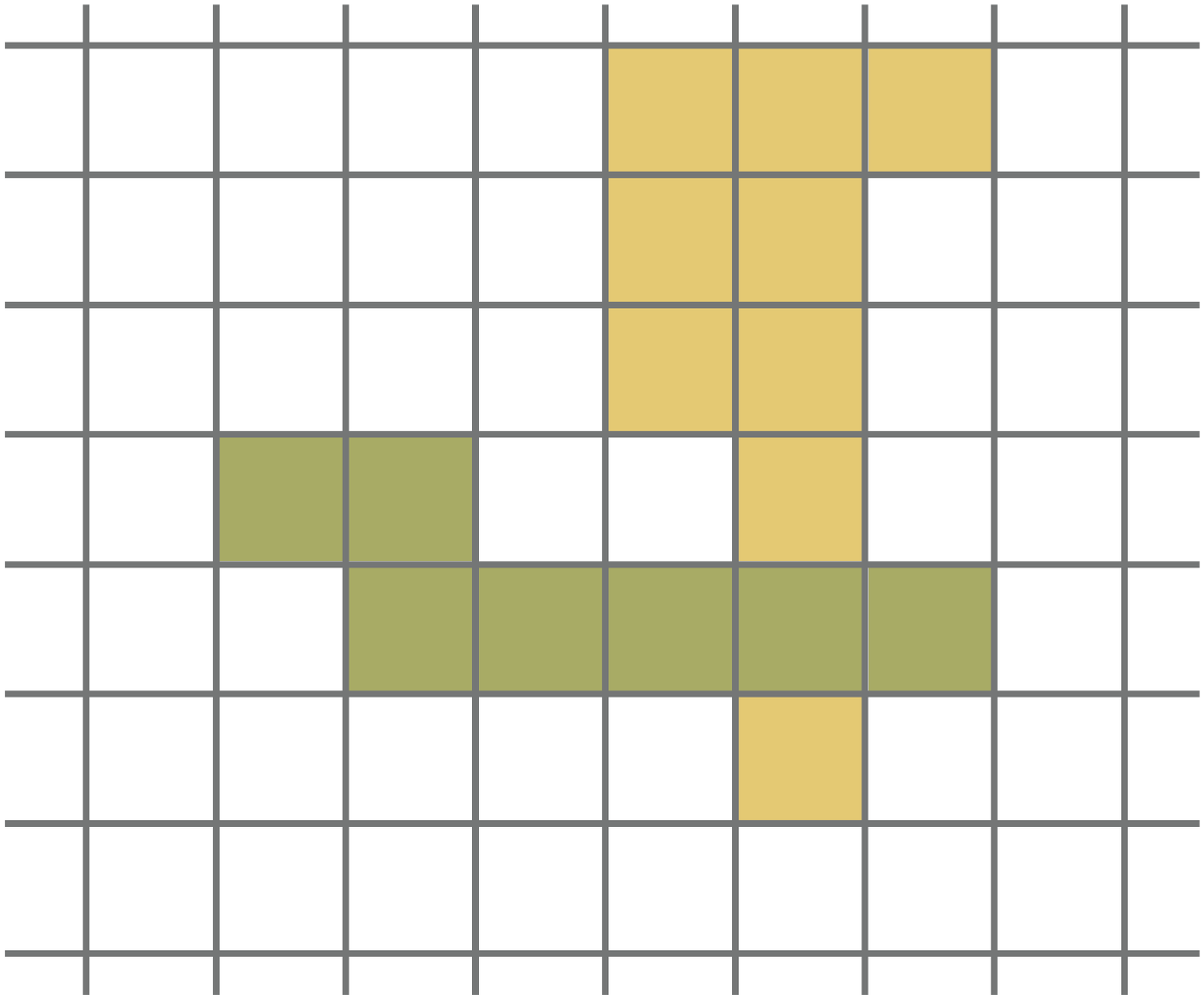
  **for each** pixel $p$

    **if** $z(o,p) < z\text{-buffer}[p]$

      $z\text{-buffer}[p] = z(o,p)$

      $\text{color}[p] = o.\text{color}$

# Z-Buffer Algorithm



**for each** object $o$

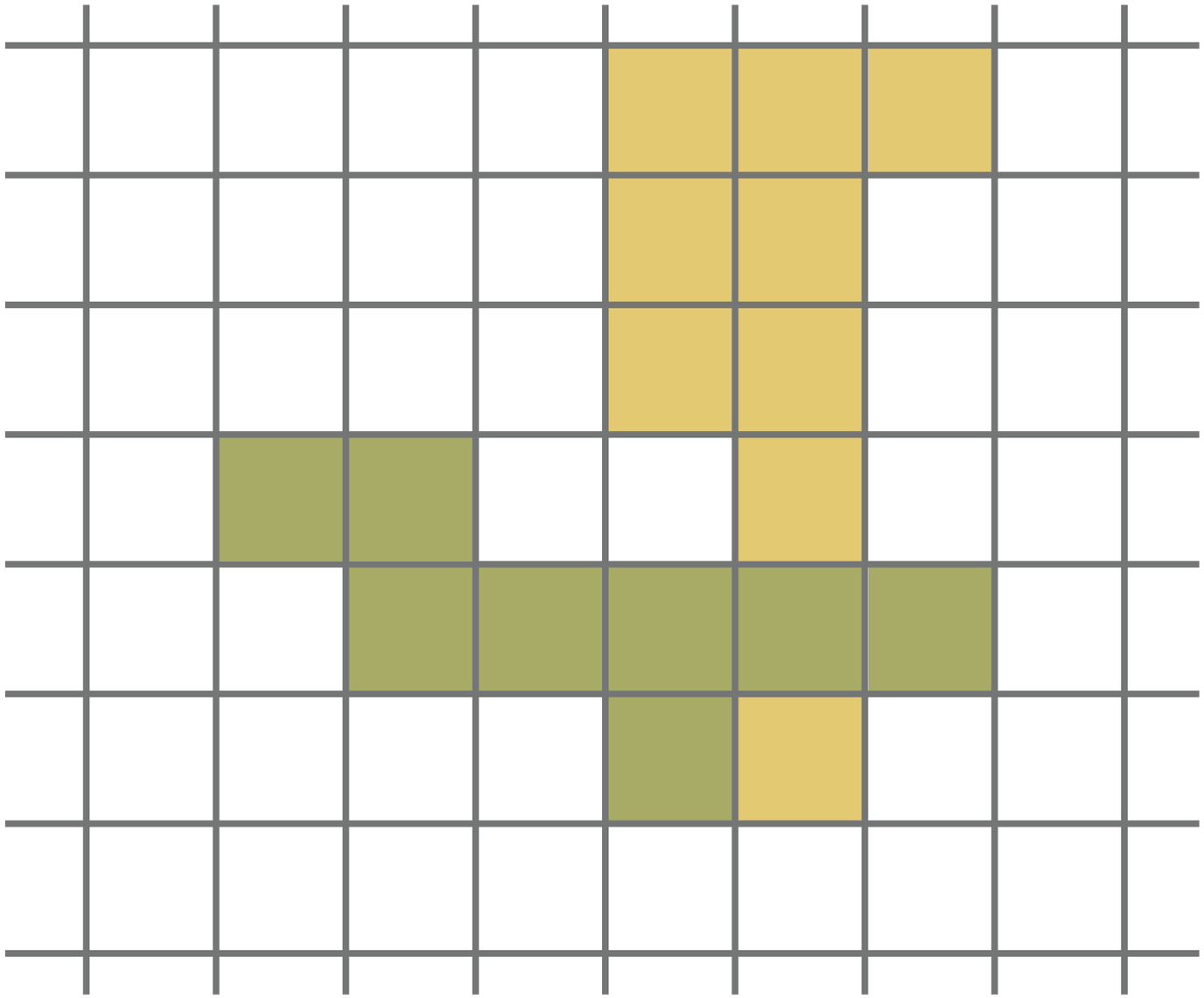
  **for each** pixel $p$

    **if** $z(o,p) < z\text{-buffer}[p]$

      $z\text{-buffer}[p] = z(o,p)$

      $\text{color}[p] = o.\text{color}$

# Z-Buffer Algorithm



**for each** object $o$

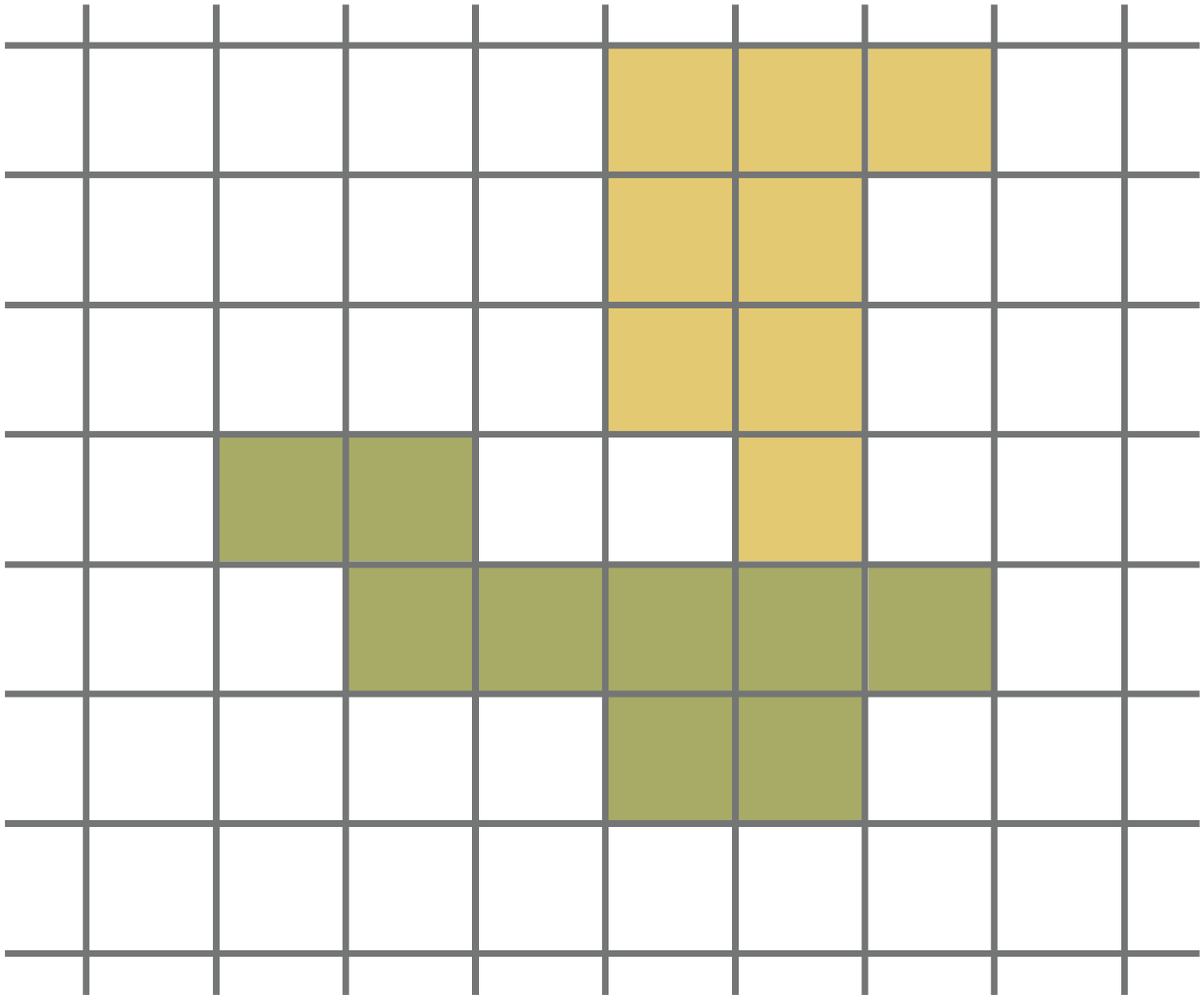
  **for each** pixel $p$

    **if** $z(o,p) < z\text{-buffer}[p]$

      $z\text{-buffer}[p] = z(o,p)$

      $\text{color}[p] = o.\text{color}$

# Z-Buffer Algorithm



**for each** object $o$

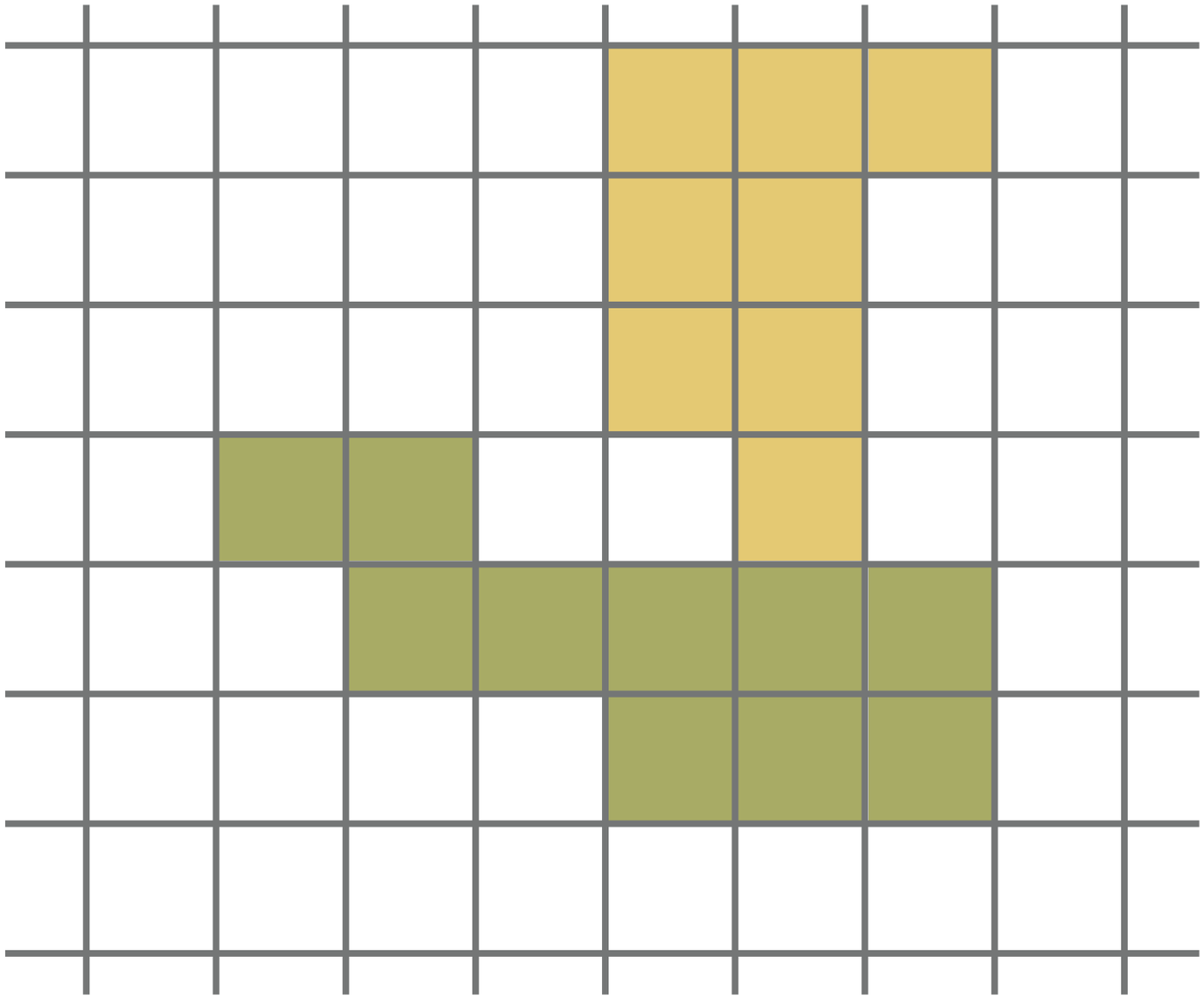
  **for each** pixel $p$

    **if** $z(o,p) < z\text{-buffer}[p]$

      $z\text{-buffer}[p] = z(o,p)$

      $\text{color}[p] = o.\text{color}$

# Z-Buffer Algorithm



**for each** object $o$

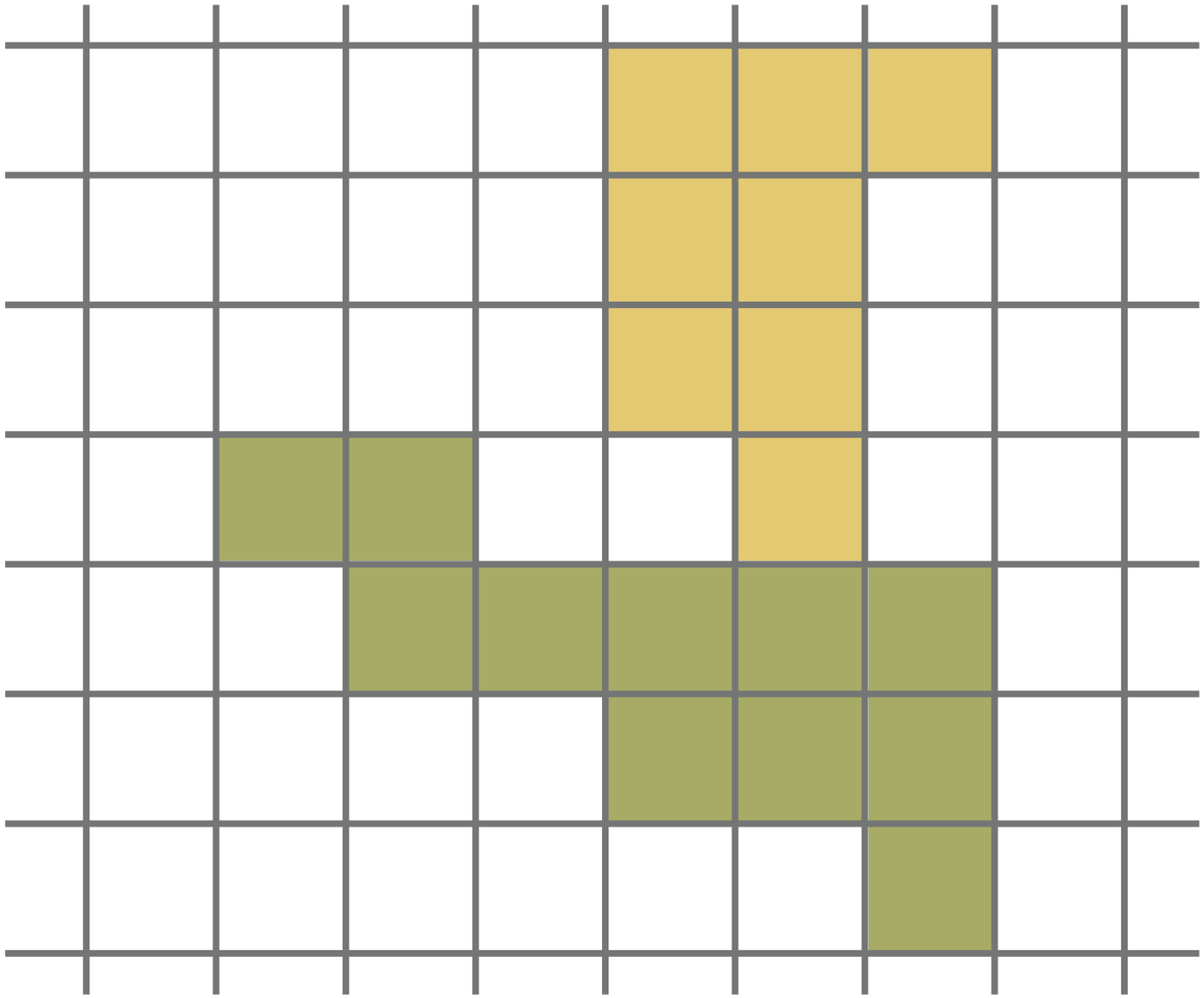
  **for each** pixel $p$

    **if** $z(o,p) < z\text{-buffer}[p]$

      $z\text{-buffer}[p] = z(o,p)$

      $\text{color}[p] = o.\text{color}$

# Z-Buffer Algorithm



**for each** object $o$

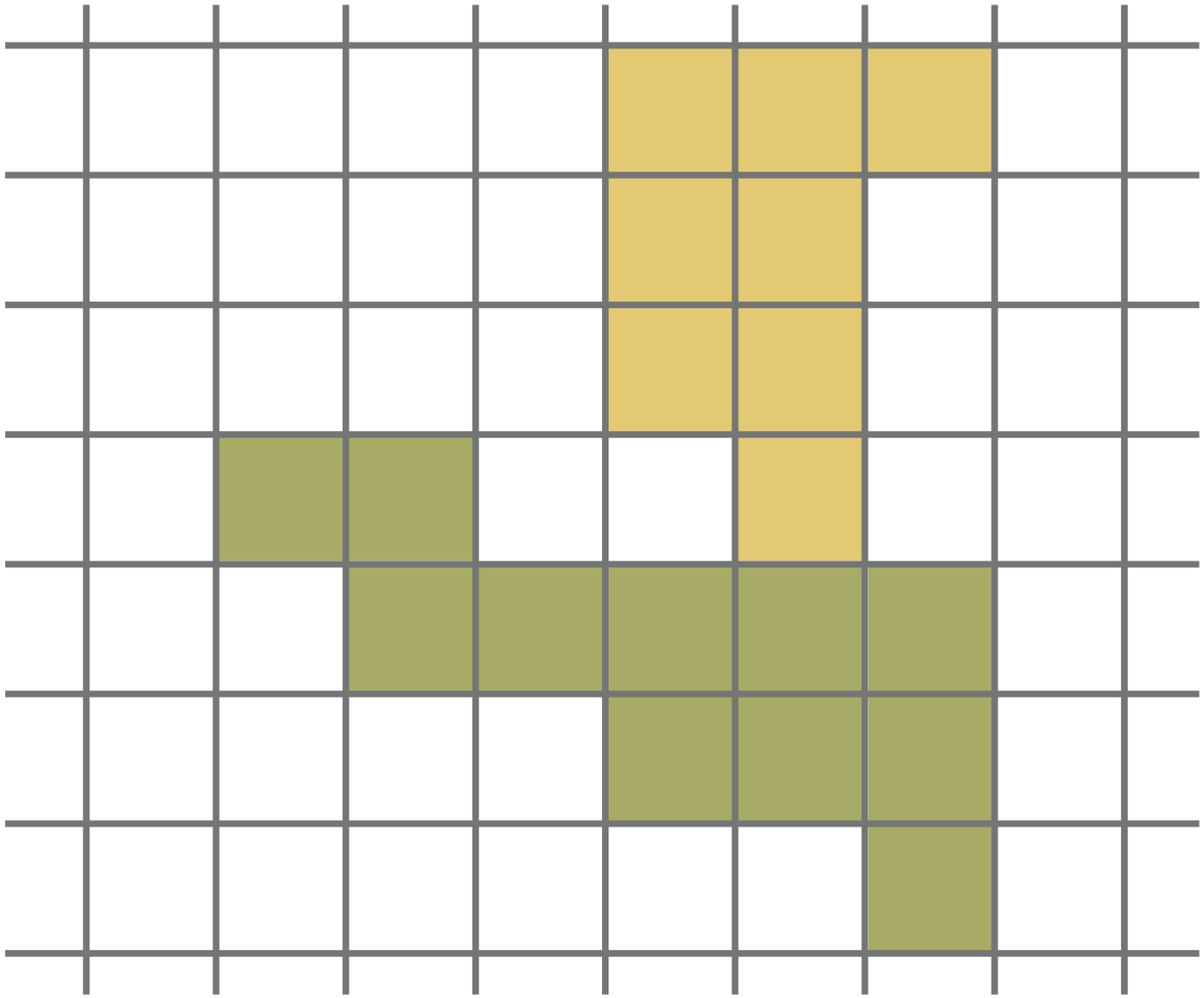
  **for each** pixel $p$

    **if** $z(o,p) < z\text{-buffer}[p]$

      $z\text{-buffer}[p] = z(o,p)$

      $\text{color}[p] = o.\text{color}$

# Z-Buffer Algorithm



**for each** object $o$

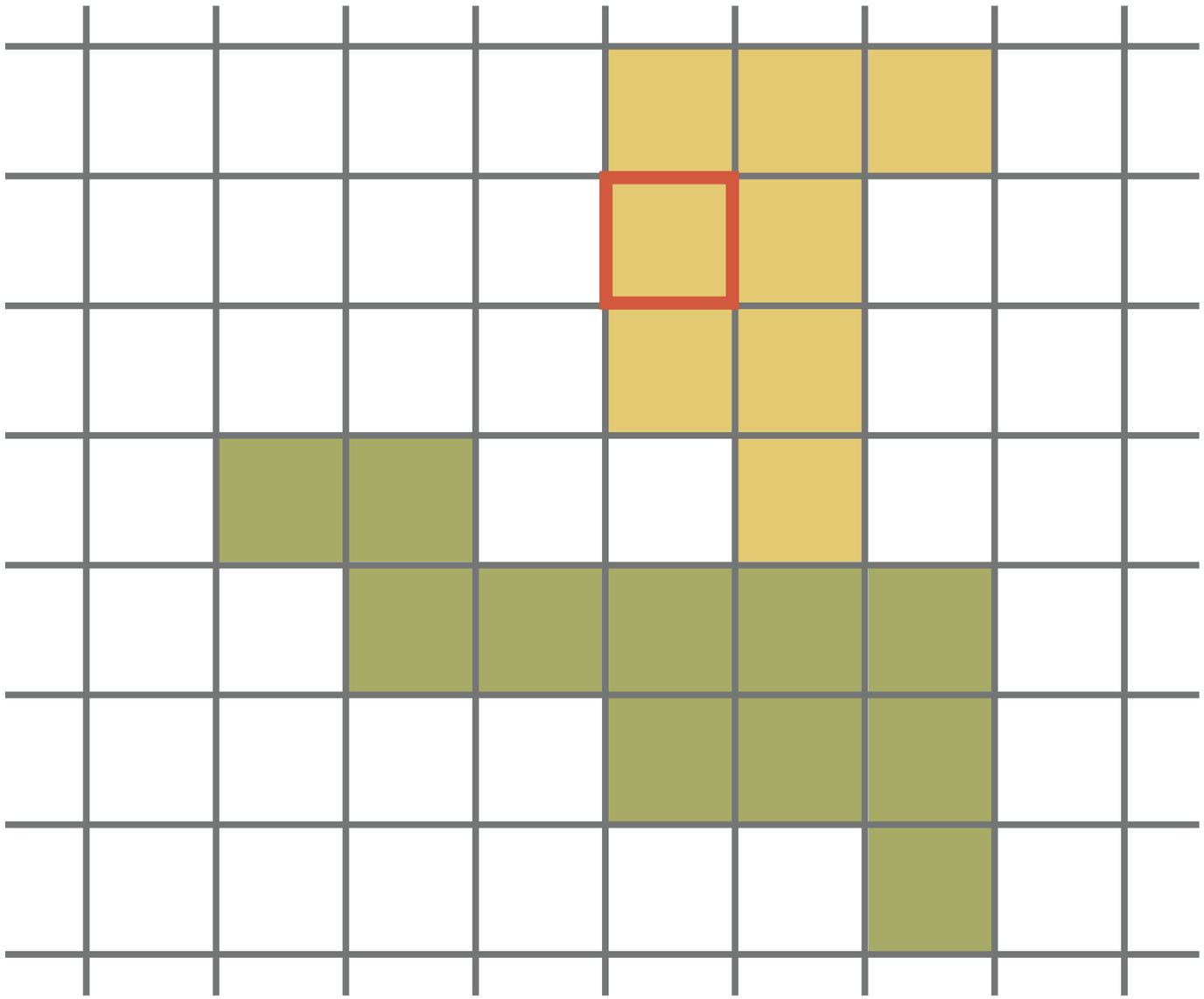
  **for each** pixel $p$

    **if** $z(o,p) < z\text{-buffer}[p]$

      $z\text{-buffer}[p] = z(o,p)$

      $\text{color}[p] = o.\text{color}$

# Z-Buffer Algorithm



**for each** object $o$

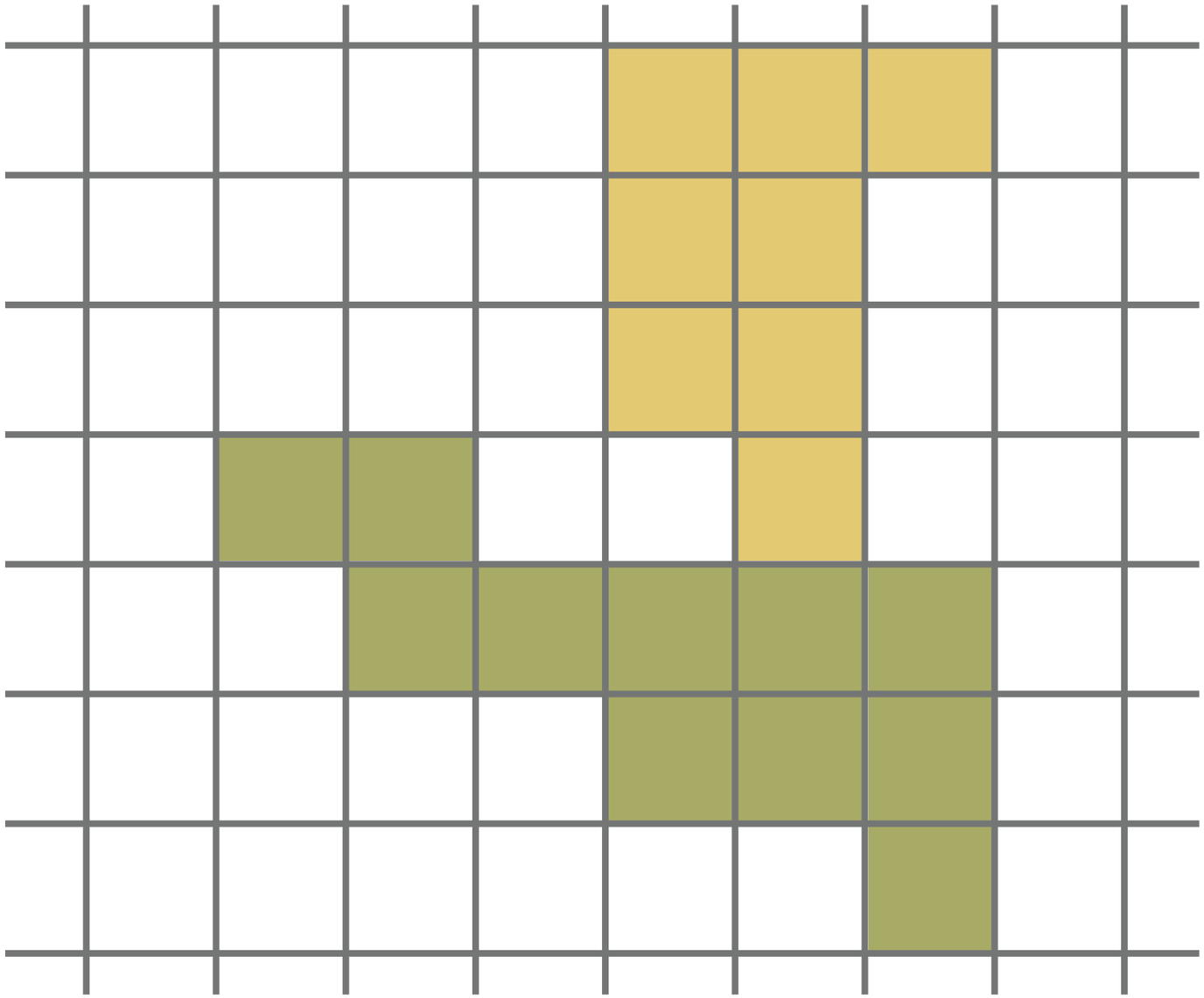
  **for each** pixel $p$

    **if** $z(o,p) < z\text{-buffer}[p]$

      $z\text{-buffer}[p] = z(o,p)$

      $\text{color}[p] = o.\text{color}$

# Z-Buffer Algorithm



**for each** object $o$

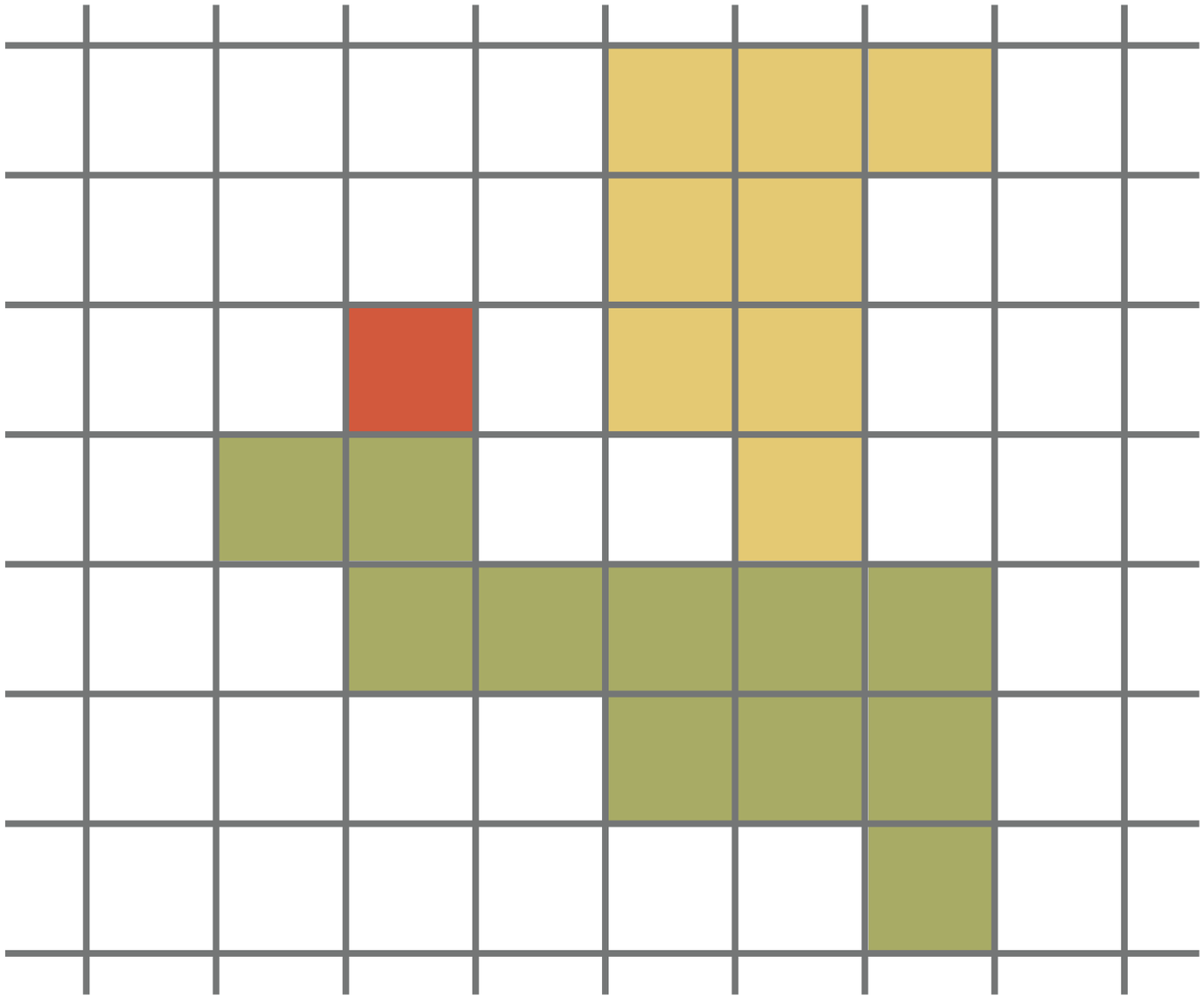
  **for each** pixel $p$

    **if** $z(o,p) < z\text{-buffer}[p]$

      $z\text{-buffer}[p] = z(o,p)$

      $\text{color}[p] = o.\text{color}$

# Z-Buffer Algorithm



**for each** object $o$

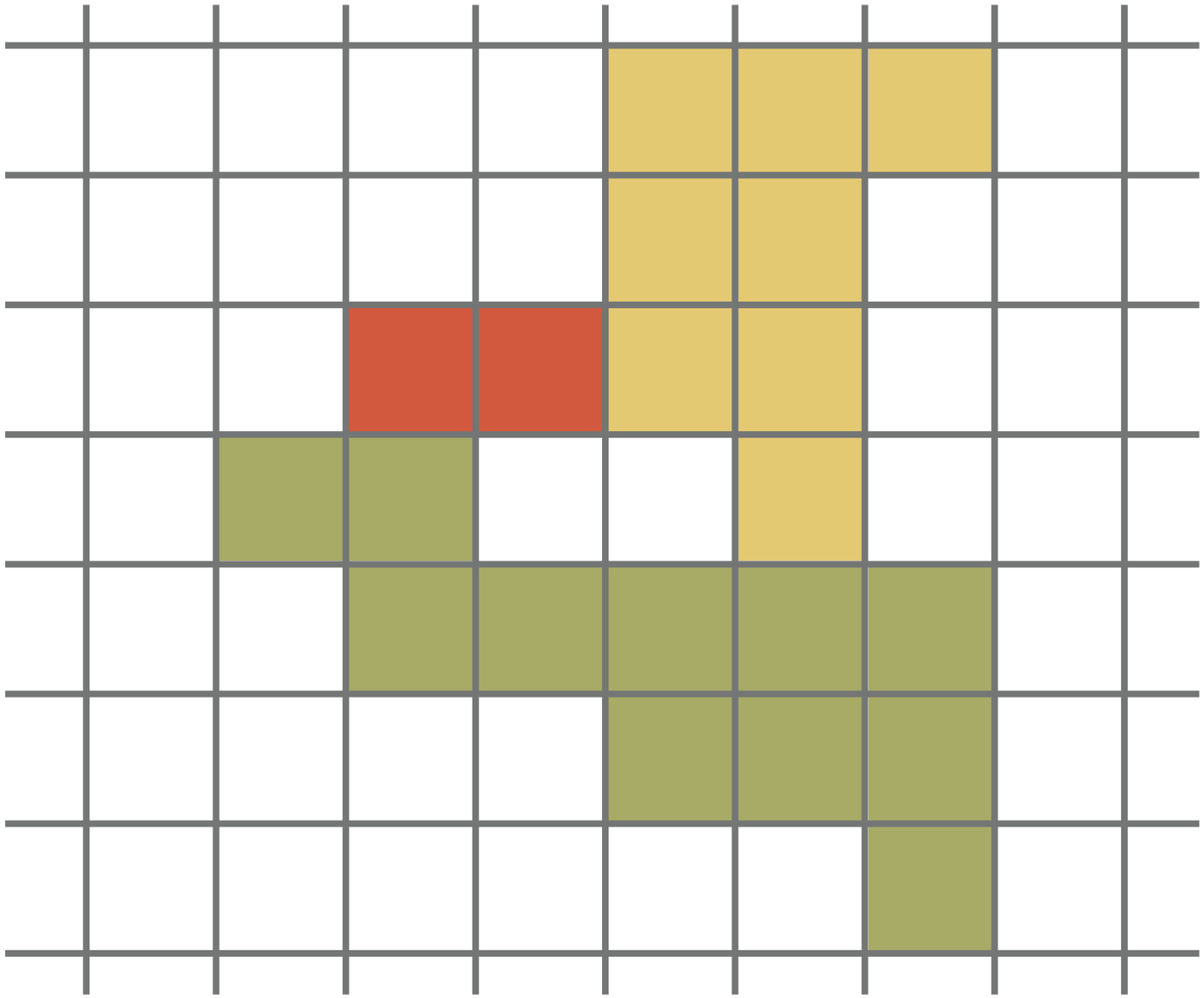
  **for each** pixel $p$

    **if** $z(o,p) < z\text{-buffer}[p]$

      $z\text{-buffer}[p] = z(o,p)$

      $\text{color}[p] = o.\text{color}$

# Z-Buffer Algorithm



**for each** object $o$

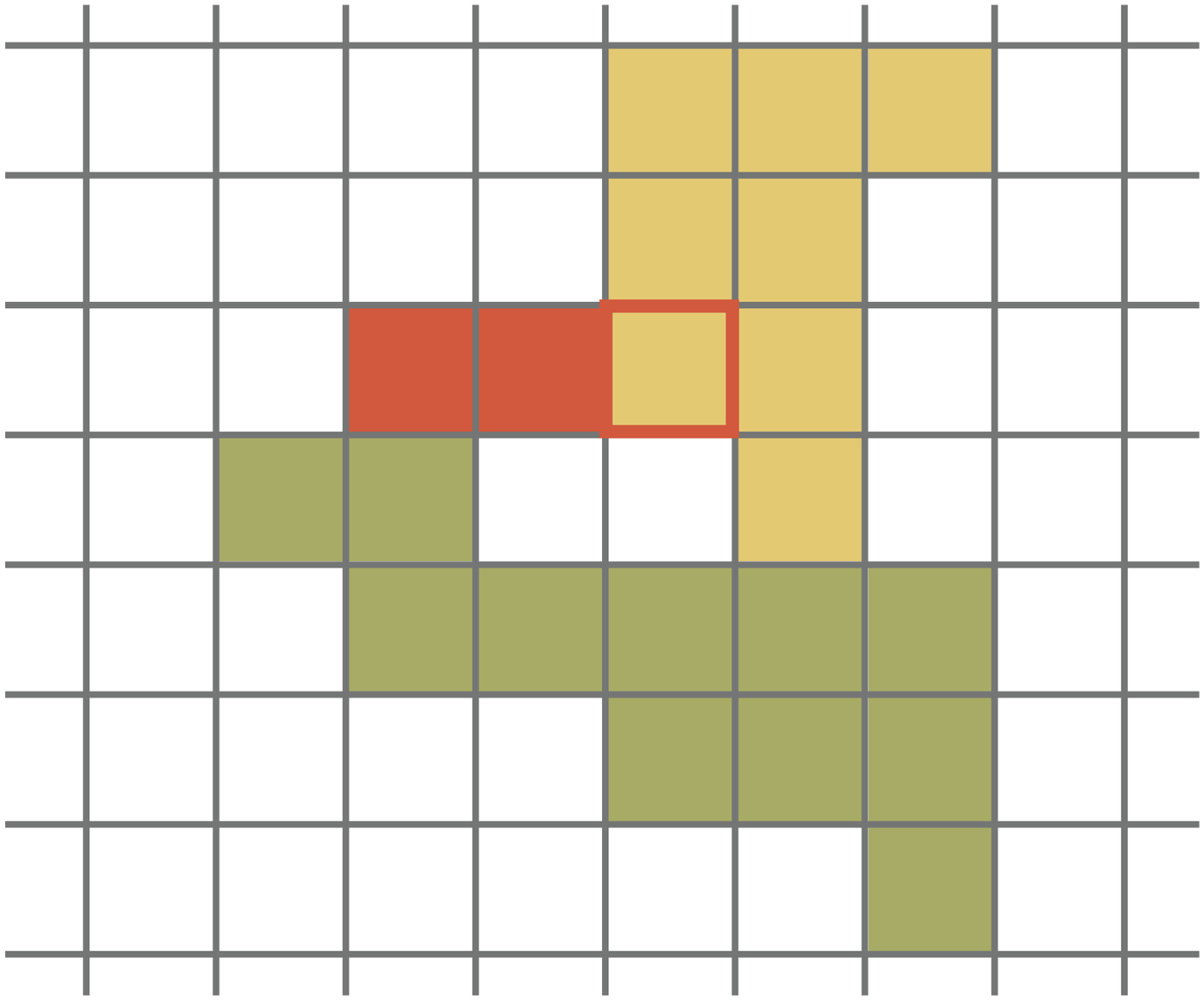
  **for each** pixel $p$

    **if** $z(o,p) < z\text{-buffer}[p]$

      $z\text{-buffer}[p] = z(o,p)$

      $\text{color}[p] = o.\text{color}$

# Z-Buffer Algorithm



**for each** object $o$

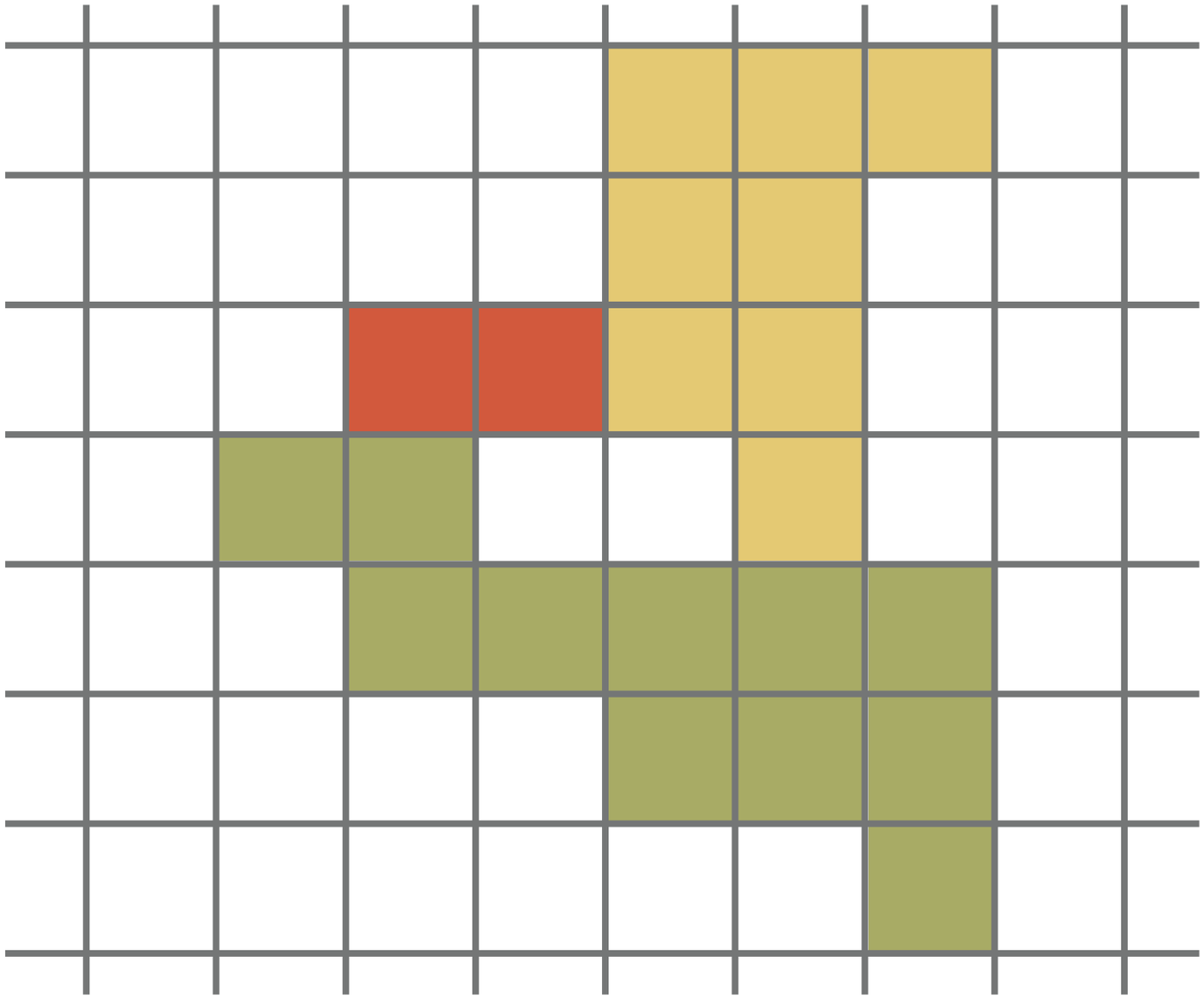
  **for each** pixel $p$

    **if** $z(o,p) < \text{z-buffer}[p]$

      $\text{z-buffer}[p] = z(o,p)$

      $\text{color}[p] = o.\text{color}$

# Z-Buffer Algorithm



**for each** object $o$

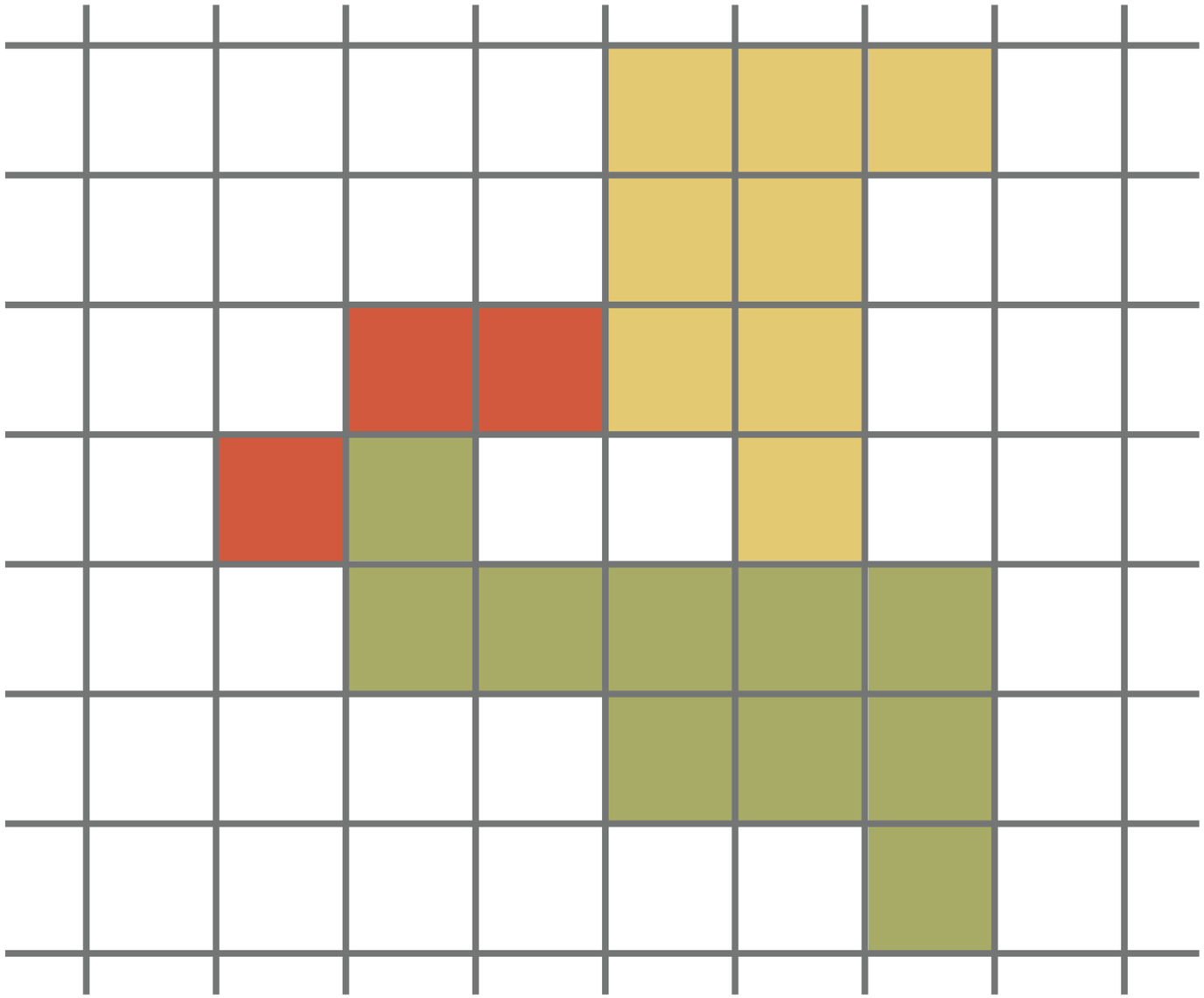
  **for each** pixel $p$

    **if** $z(o,p) < z\text{-buffer}[p]$

      $z\text{-buffer}[p] = z(o,p)$

      $\text{color}[p] = o.\text{color}$

# Z-Buffer Algorithm



**for each** object $o$

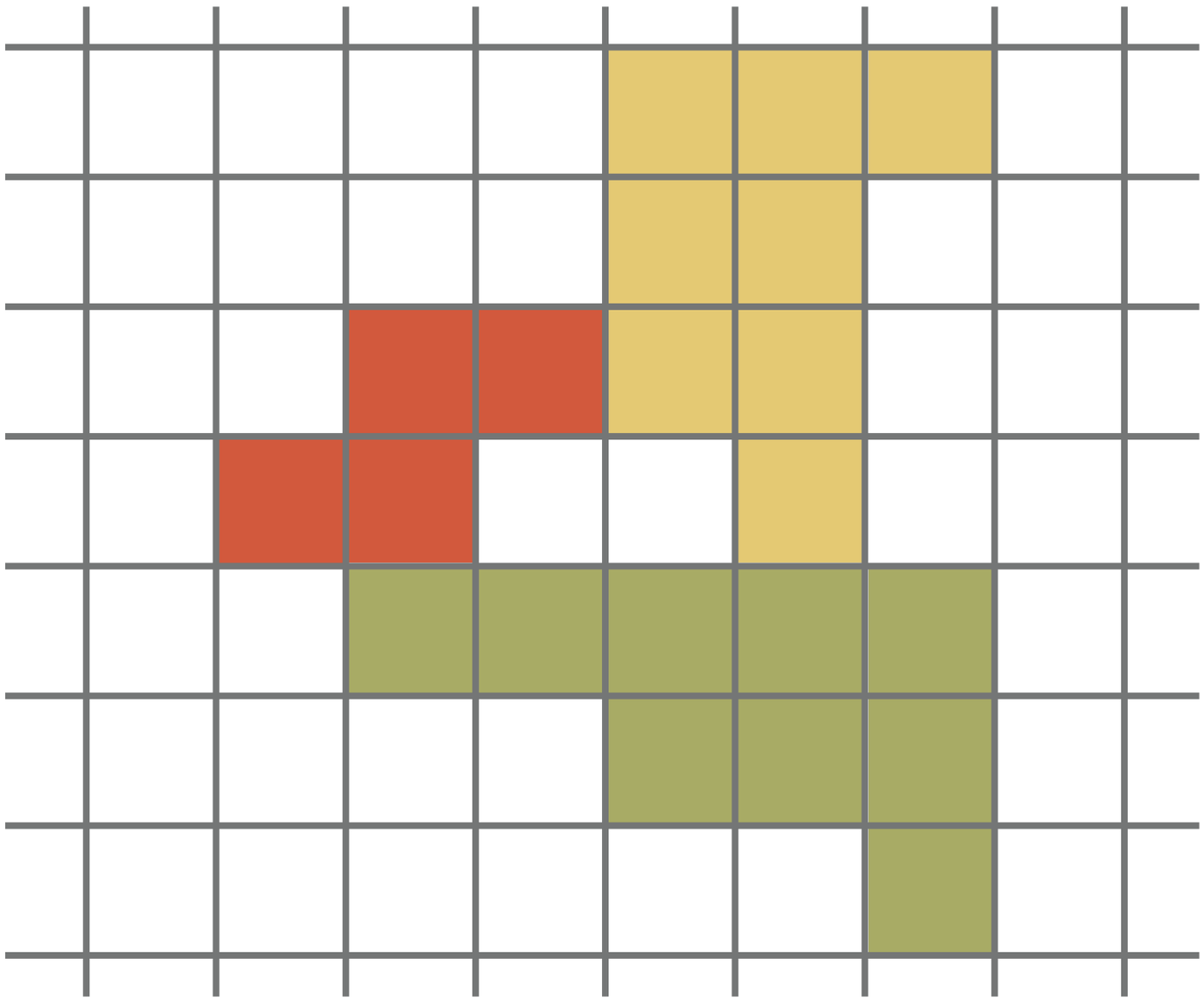
  **for each** pixel $p$

    **if** $z(o,p) < z\text{-buffer}[p]$

      $z\text{-buffer}[p] = z(o,p)$

      $\text{color}[p] = o.\text{color}$

# Z-Buffer Algorithm



**for each** object $o$

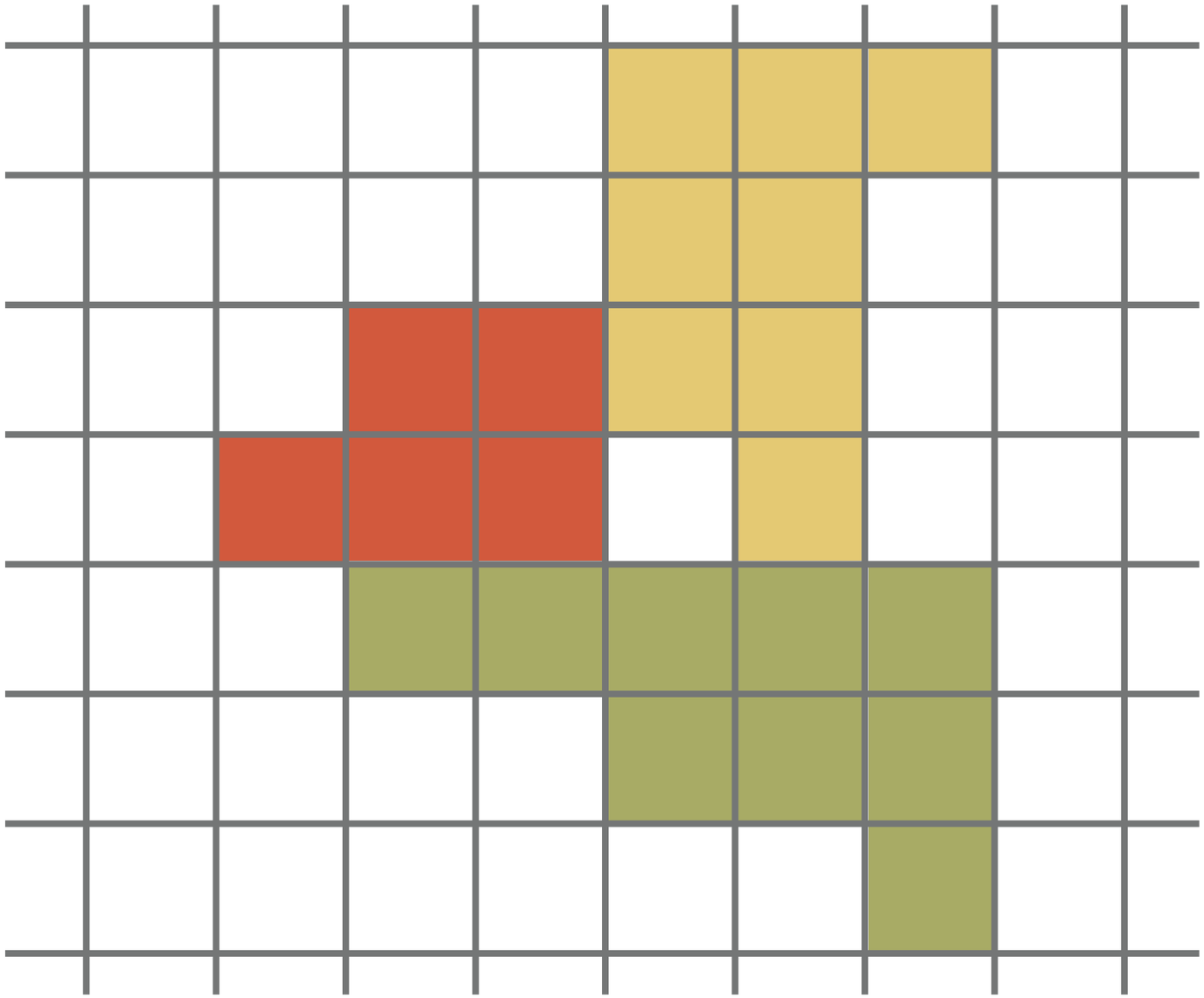
  **for each** pixel $p$

    **if** $z(o,p) < z\text{-buffer}[p]$

      $z\text{-buffer}[p] = z(o,p)$

      $\text{color}[p] = o.\text{color}$

# Z-Buffer Algorithm



**for each** object $o$

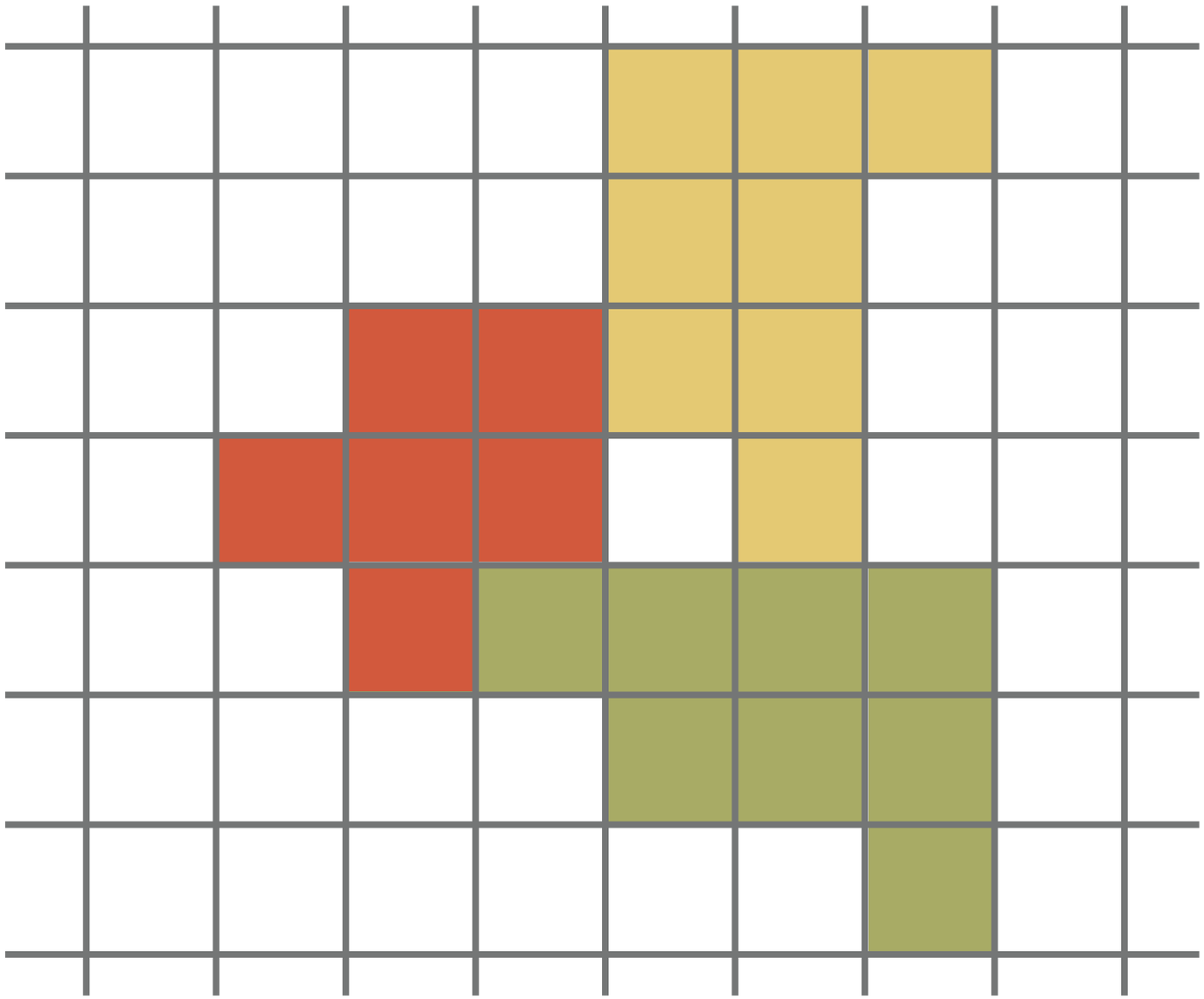
  **for each** pixel $p$

    **if** $z(o,p) < z\text{-buffer}[p]$

      $z\text{-buffer}[p] = z(o,p)$

      $\text{color}[p] = o.\text{color}$

# Z-Buffer Algorithm



**for each** object $o$

  **for each** pixel $p$

    **if** $z(o,p) < z\text{-buffer}[p]$

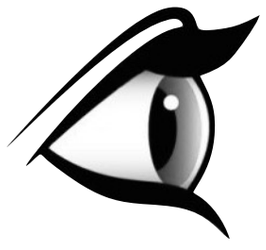
      $z\text{-buffer}[p] = z(o,p)$

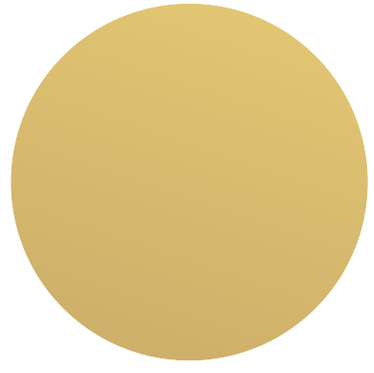
      $\text{color}[p] = o.\text{color}$

# A-Buffer

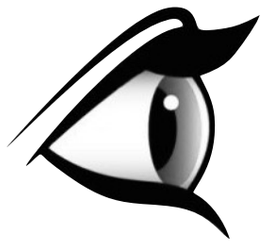
- Zur korrekten Darstellung von Szenen mit teiltransparenten Objekten
- Alle Fragmente, die auf ein Pixel treffen in pro-Pixel-Liste speichern (mit c (Farbe), z (Tiefe), a (Opazität))
- Zum Schluss für jedes Pixel:
  - Liste nach z sortieren
  - Fragmentfarben von hinten nach vorne übereinander zeichnen:
    - $C_i = a_i\, c_i + (1\text{-}a_i)\, C_{i\text{-}1}$
    - wobei $C_{\text{-}1}$ mit einer “Hintergrundfarbe” (z.B. schwarz) initialisiert ist (die nur relevant wird, wenn es Bereiche gibt in denen Pixel ausschließlich von nicht-opaken Fragmenten bedeckt werden.)

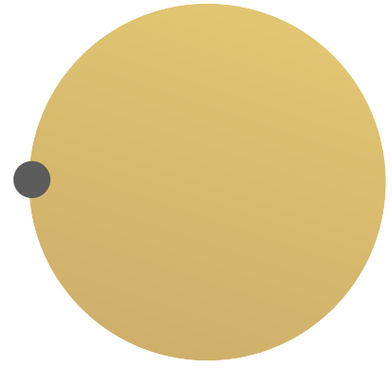
# Sichtbarkeit



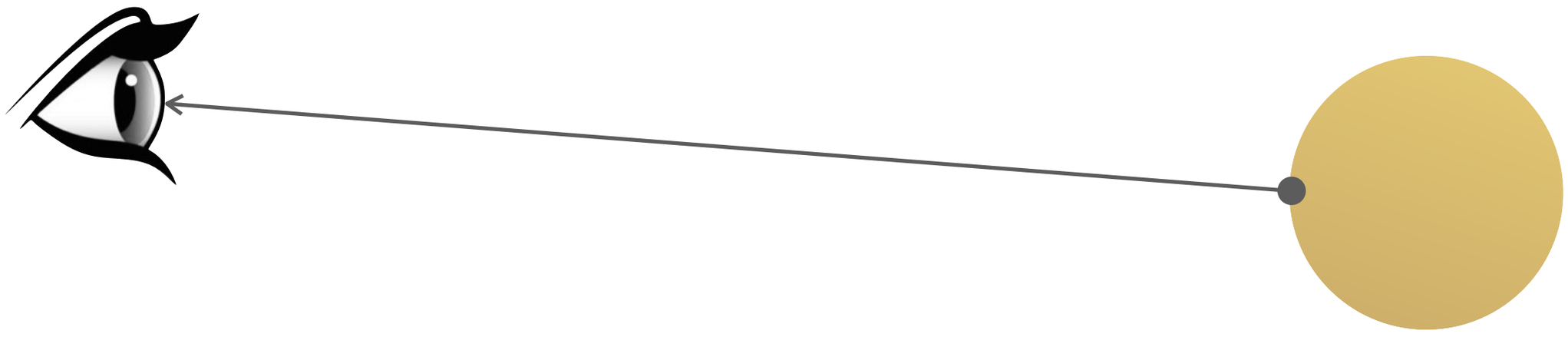


# Sichtbarkeit

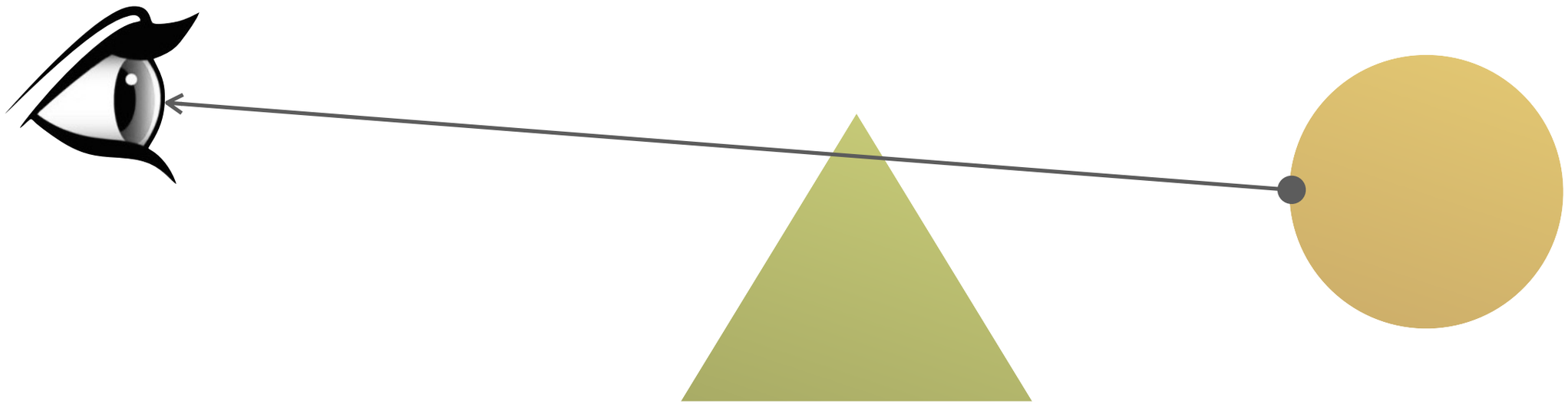




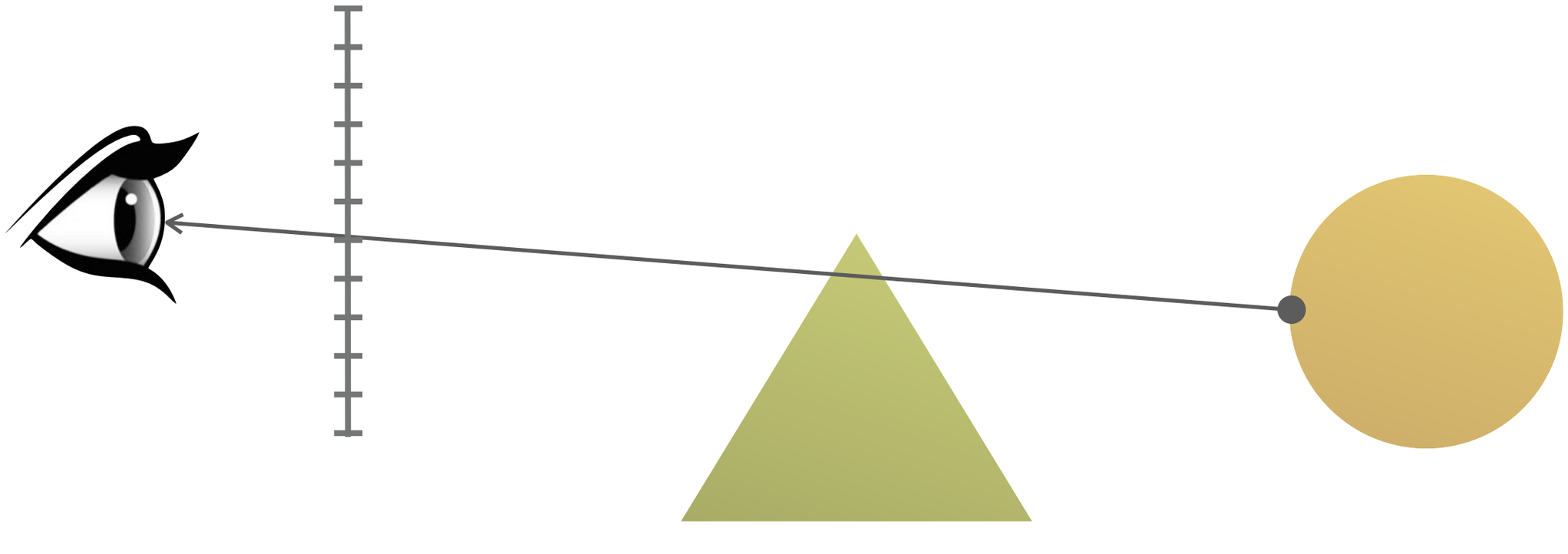
# Sichtbarkeit



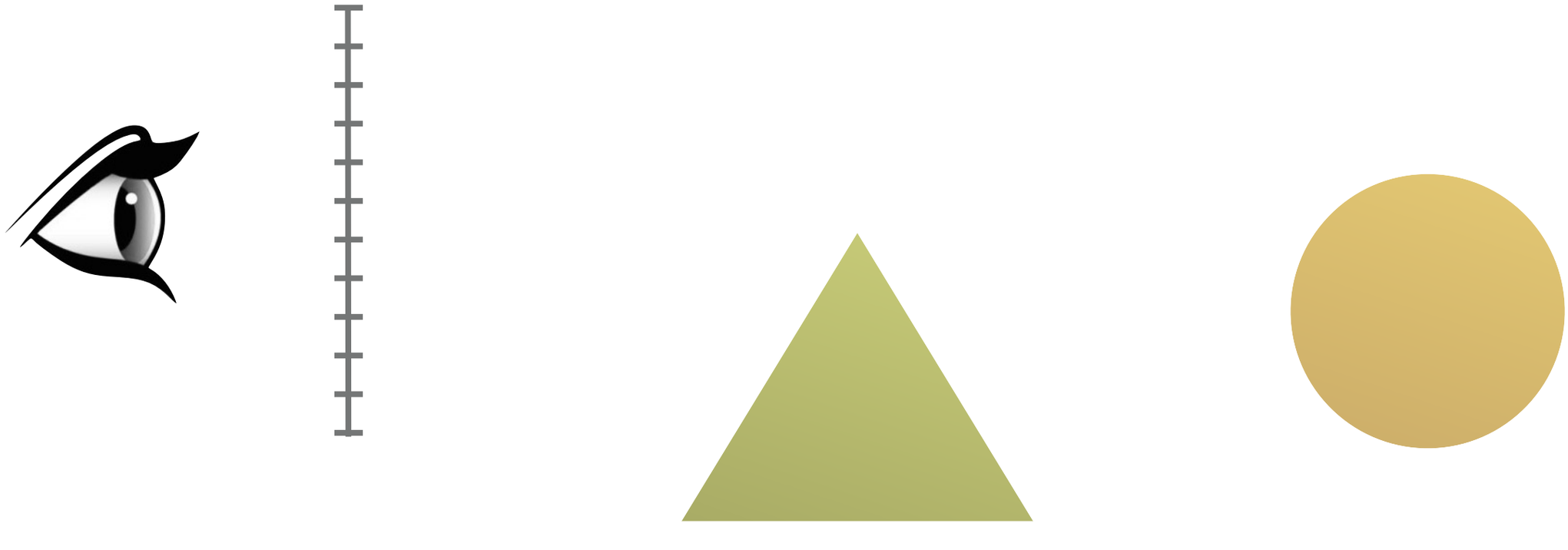
# Sichtbarkeit



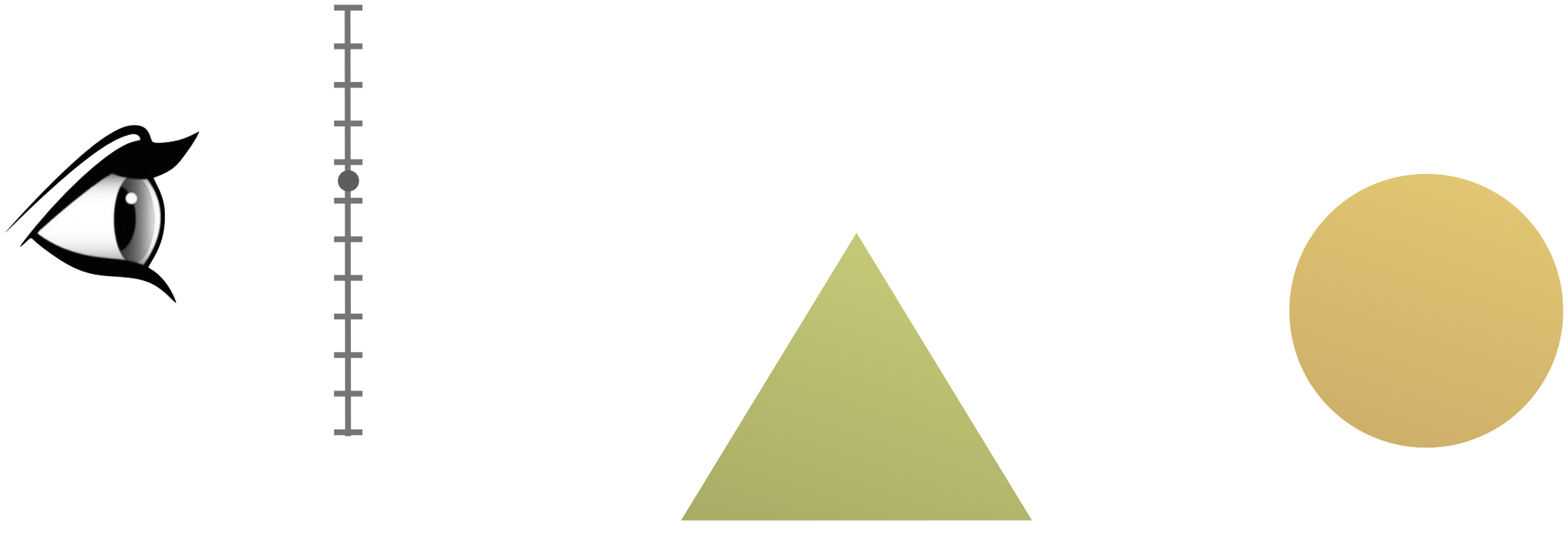
# Sichtbarkeit



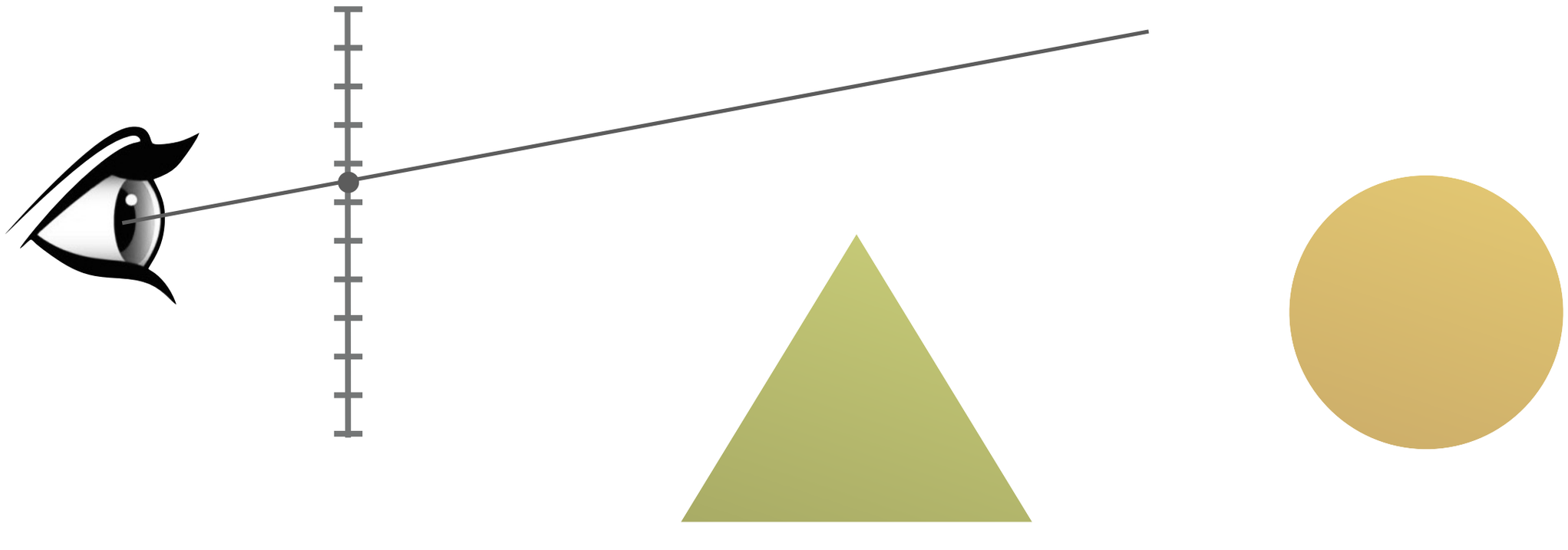
# Sichtbarkeit



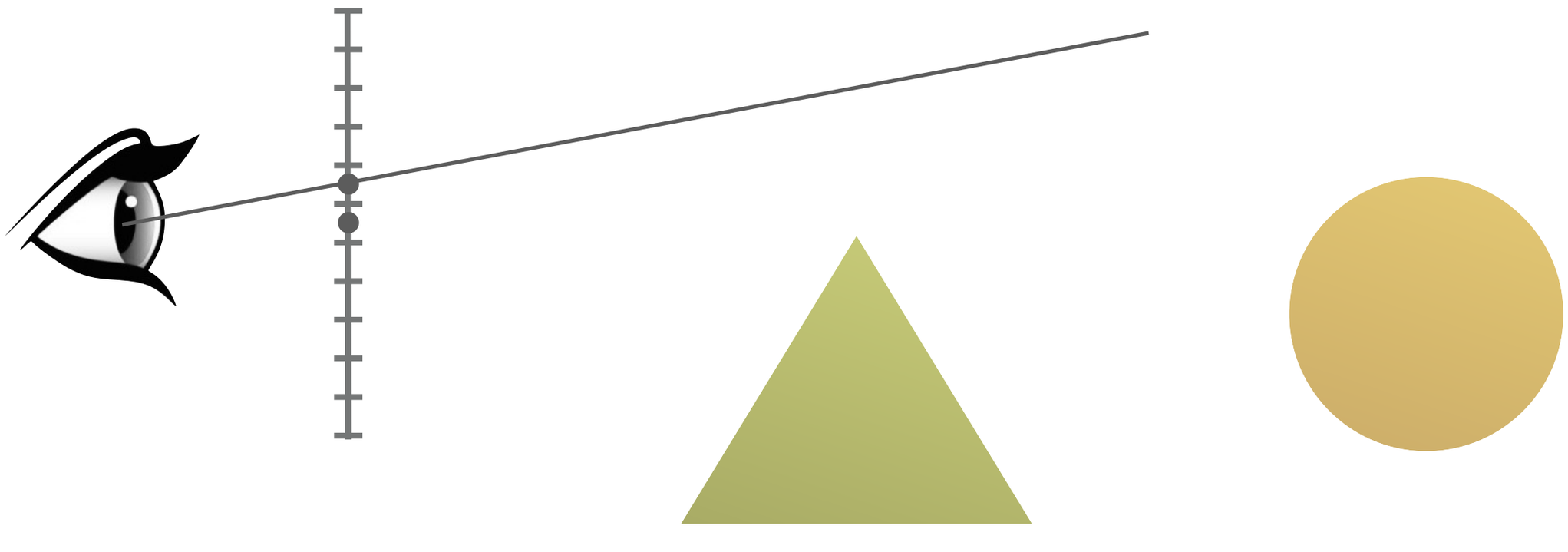
# Sichtbarkeit



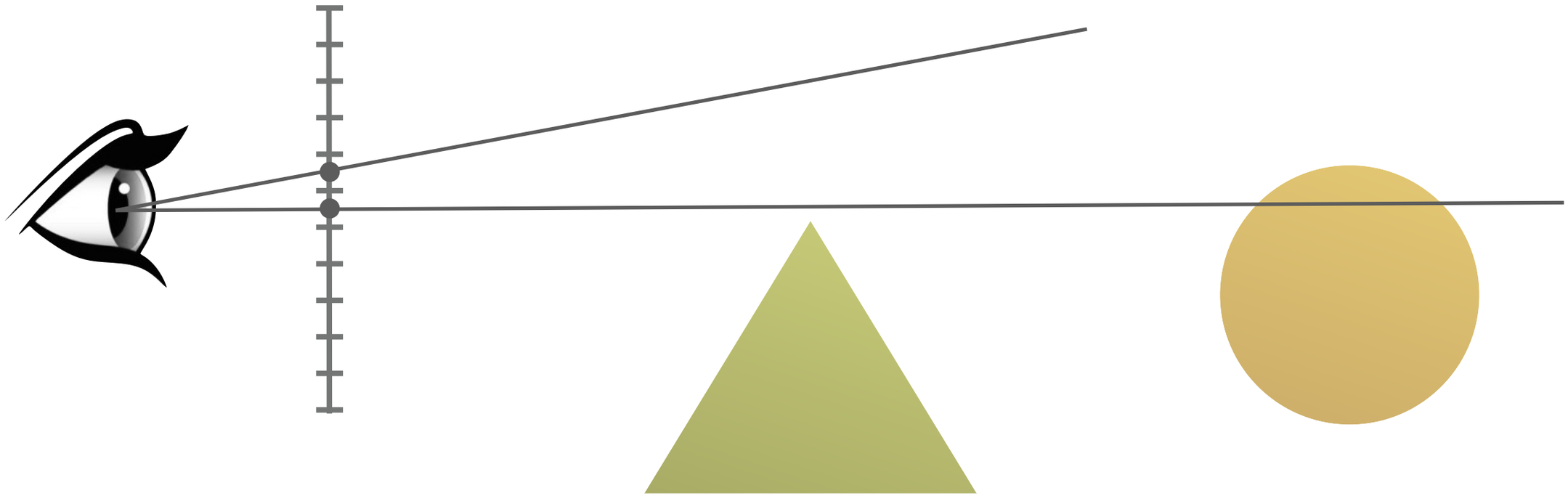
# Sichtbarkeit



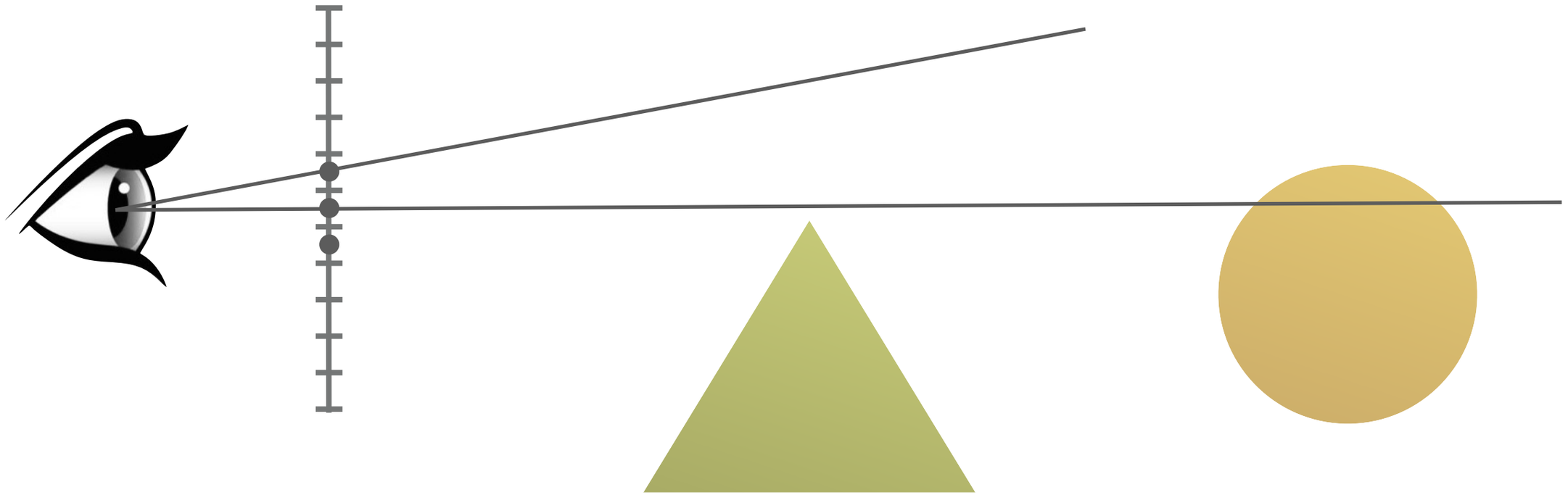
# Sichtbarkeit



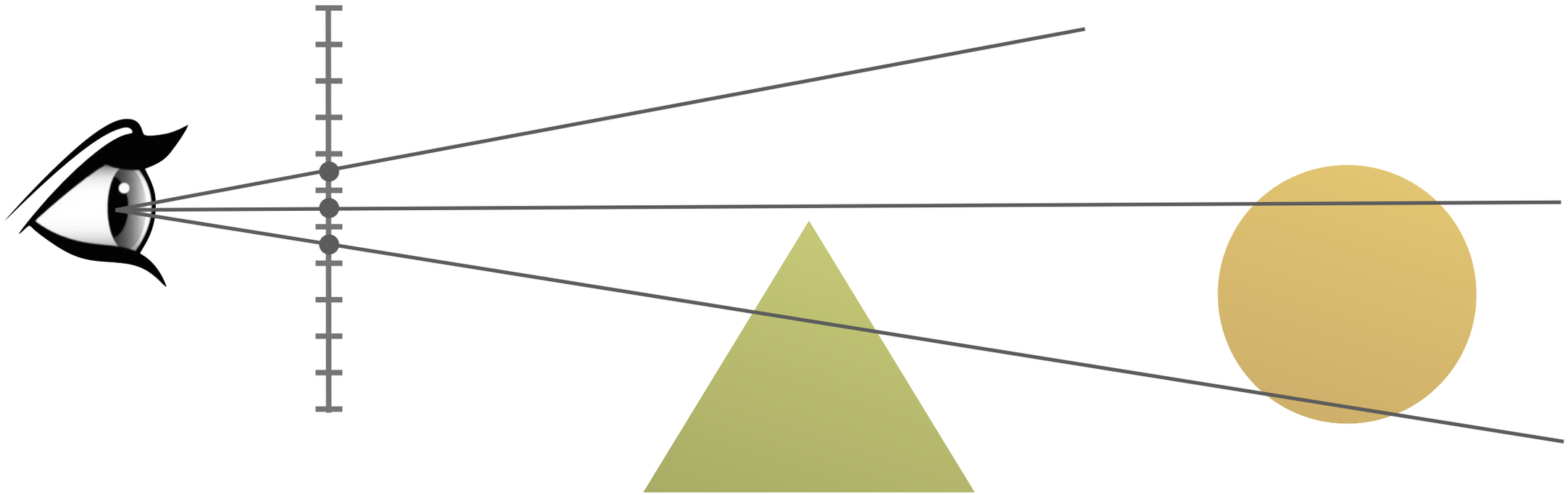
# Sichtbarkeit



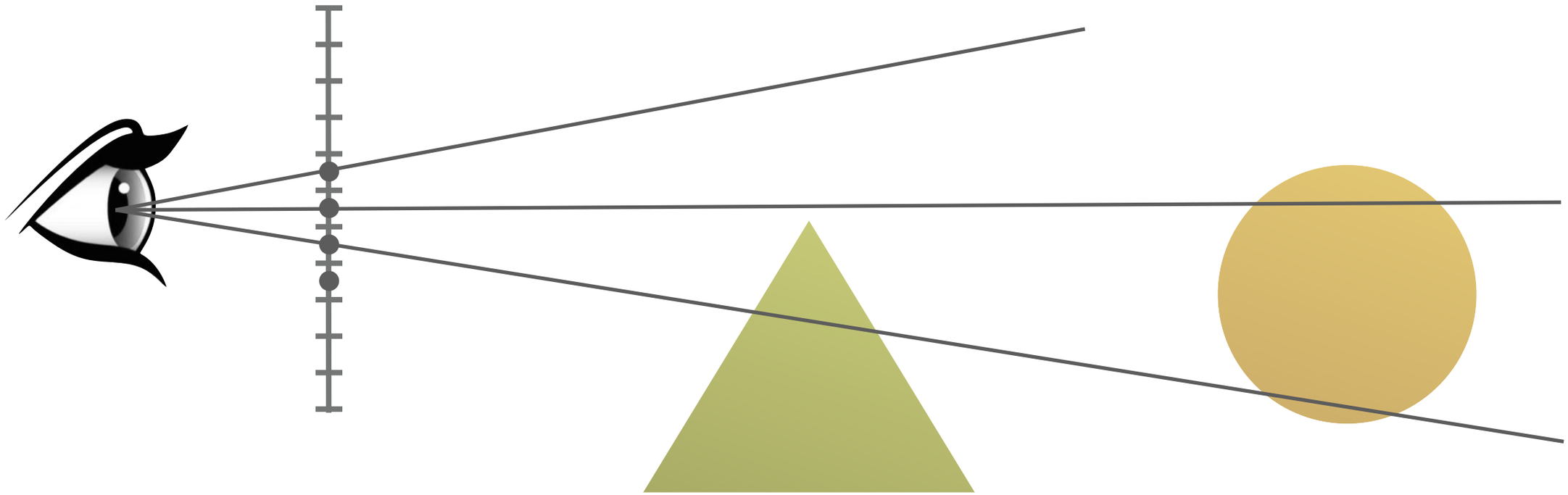
# Sichtbarkeit



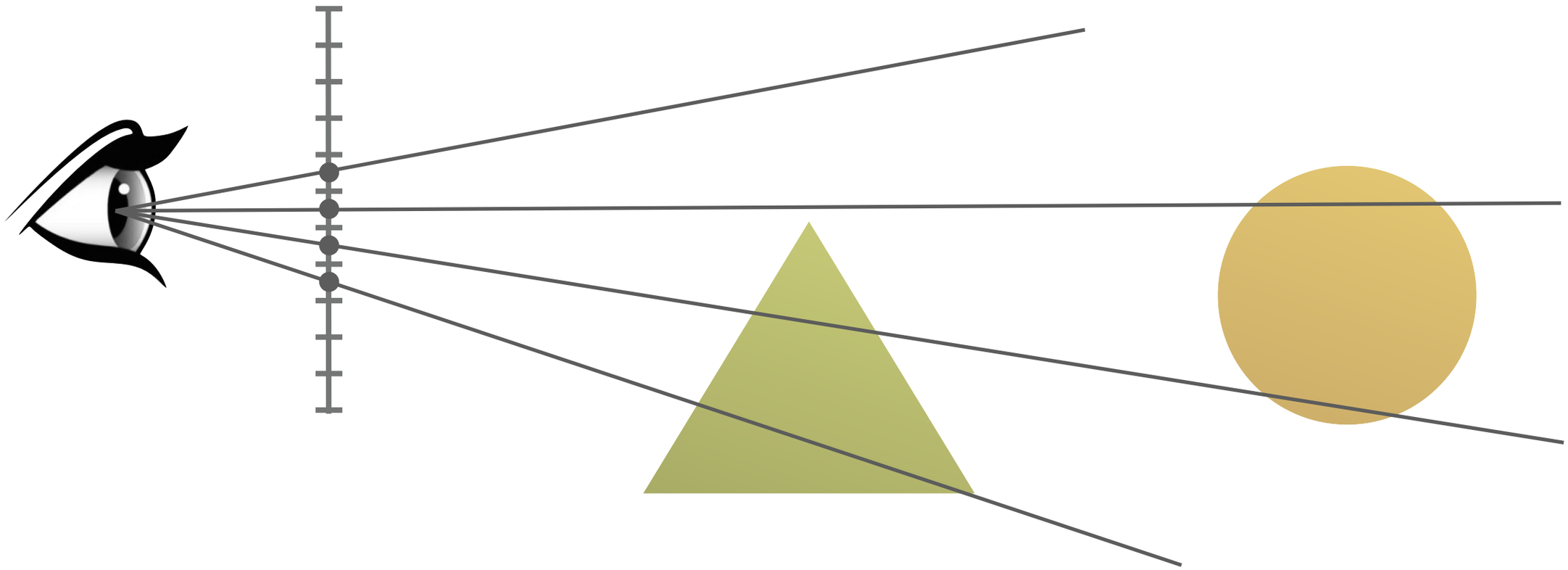
# Sichtbarkeit



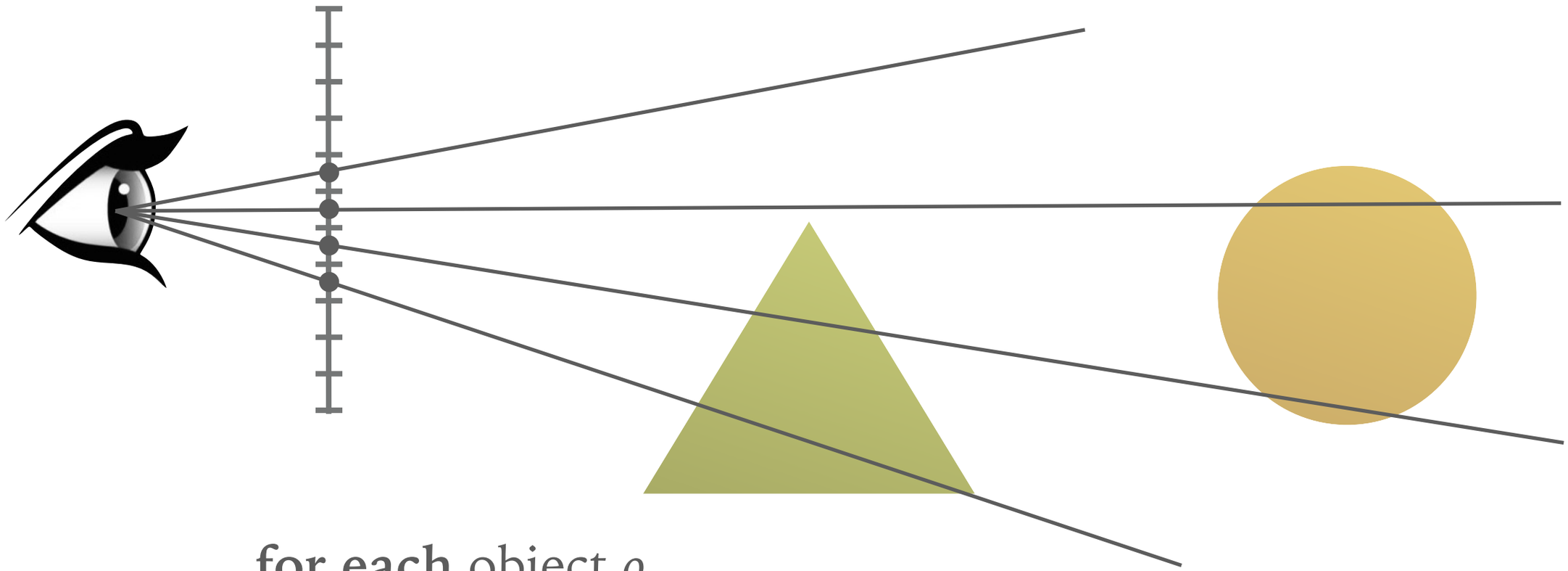
# Sichtbarkeit



# Sichtbarkeit



# Sichtbarkeit



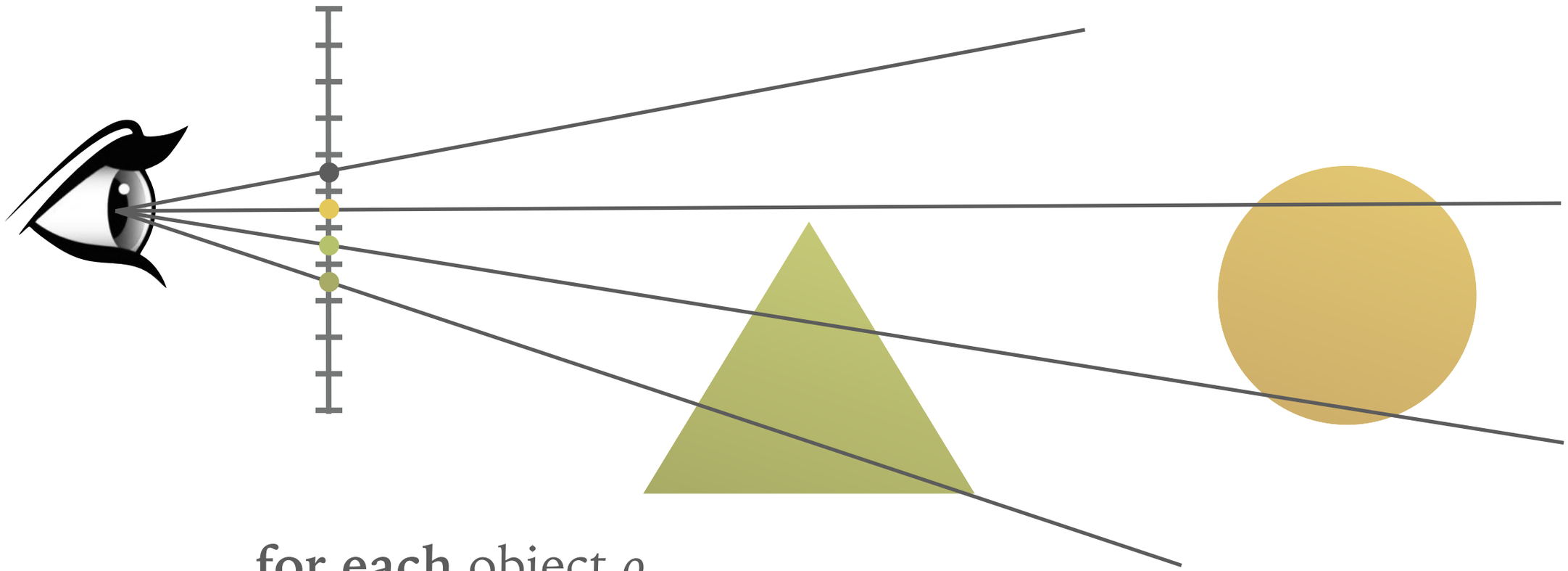
**for each** object $o$

  **for each** pixel $p$

    **if** line through $p$ intersects $o$ and nothing else between $p$ and $o$

      $\text{color}[p] = o.\text{color}$

# Sichtbarkeit



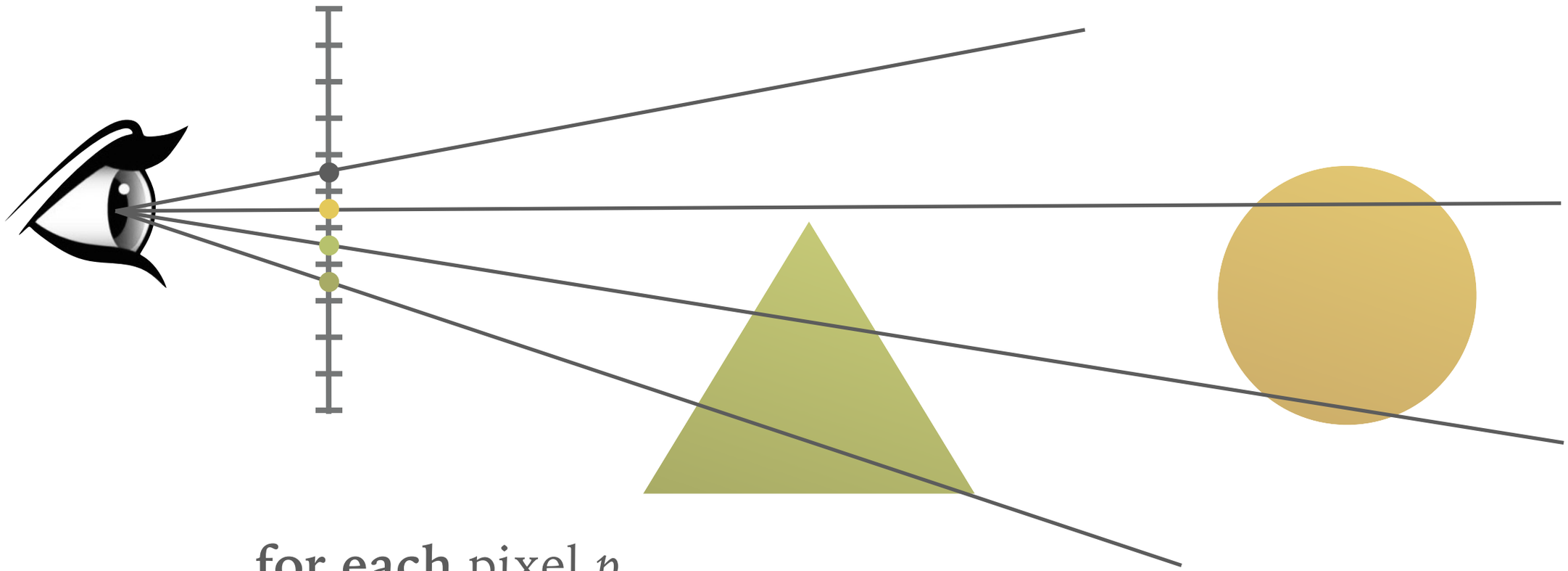
**for each** object $o$

  **for each** pixel $p$

    **if** line through $p$ intersects $o$ and nothing else between $p$ and $o$

      $\text{color}[p] = o.\text{color}$

# Ray-Casting



**for each** pixel $p$

find first object $o$ intersected by the ray through $p$

$\text{color}[p] = o.\text{color}$

# Vergleich

- Z-BUFFER

**for each** object $o$ $O(n_p\, n_o)$

**for each** pixel $p$

**if** $z(o,p) < \text{z-buffer}[p]$

$\text{z-buffer}[p] = z(o,p)$

$\text{color}[p] = o.\text{color}$

- RAYCASTING

**for each** pixel $p$ $O(n_p \log n_o)$

find first object $o$ intersected by ray

$\text{color}[p] = o.\text{color}$

# Vergleich

- Z-BUFFER

**for each** object $o$

**for each** pixel $p$

**if** $z(o,p) < \text{z-buffer}[p]$

$\text{z-buffer}[p] = z(o,p)$

$\text{color}[p] = o.\text{color}$

} primitive Operationen

- RAYCASTING

**for each** pixel $p$

find first object $o$ intersected by ray

$\text{color}[p] = o.\text{color}$

# Vergleich

- Z-BUFFER

**for each** object $o$

**for each** pixel $p$

**if** $z(o,p) < \text{z-buffer}[p]$

$\text{z-buffer}[p] = z(o,p)$

$\text{color}[p] = o.\text{color}$

} primitive Operationen

- RAYCASTING

**for each** pixel $p$

find first object $o$ intersected by ray ⟵ komplexe Operation

$\text{color}[p] = o.\text{color}$

# Vergleich

- Z-BUFFER

**for each** object $o$

**for each** pixel $p$

**if** $z(o,p) < z\text{-buffer}[p]$

$z\text{-buffer}[p] = z(o,p)$

$color[p] = o.color$

} primitive Operationen

unabhängig von anderen $o$ und $p$

- RAYCASTING

**for each** pixel $p$

find first object $o$ intersected by ray ⟵ komplexe Operation

$color[p] = o.color$

# Vergleich

- Z-BUFFER

**for each** object $o$

**for each** pixel $p$

**if** $z(o,p) < z\text{-buffer}[p]$

$z\text{-buffer}[p] = z(o,p)$

$\text{color}[p] = o.\text{color}$

} primitive Operationen

unabhängig von anderen $o$ und $p$

- RAYCASTING

**for each** pixel $p$

find first object $o$ intersected by ray ← komplexe Operation

abhängig von anderen $o$

$\text{color}[p] = o.\text{color}$


# Bis zum nächsten Mal!

Media Informatics
**Graphics & Geometric Computing**
**Prof. Dr. M. Campen**