

**Macoun**

# Animation ist schwierig

Frank Illenberger

# Warum Animation?

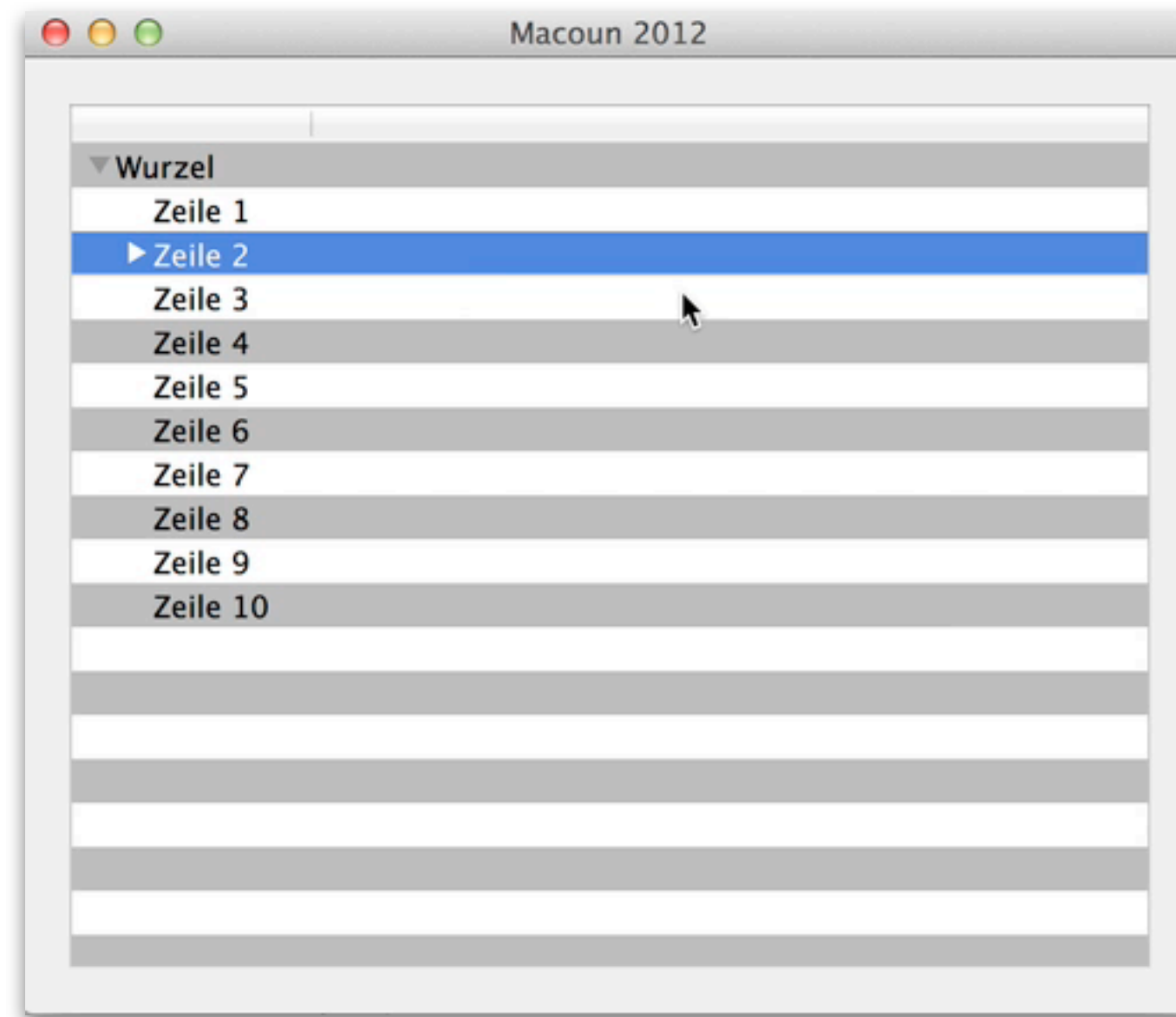
# Warum Animation?

Sanft interpolierte Übergänge zwischen grafischen Zuständen einer Anwendung

- erhöhen die Physikalität von Objekten
- erzeugen Ortsgedächtnis
- veranschaulichen komplexe Zusammenhänge

# Erhöhte Physikalität?

NSOutlineView bewegt Hintergrundstreifen

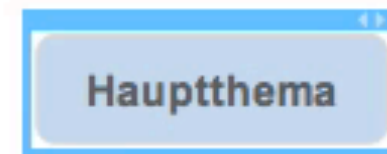






# Ortsgedächtnis

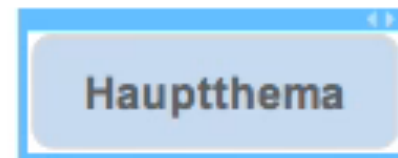
Seltsame Reihenfolge beim Anlegen von Mind-Map-Zweigen?





# Ortsgedächtnis

Mit Animation wäre das nicht passiert...



# Goldene Regel

für die Animation einer Ansicht

Gibt es in einer Ansicht, deren Übergänge sanft animiert sind, einen Bestandteil, der einen sprunghaften Übergang vollzieht, wird die Aufmerksamkeit des Betrachters stets auf diesen Punkt gezogen.

Man könnte sich die Animationen also auch sparen.

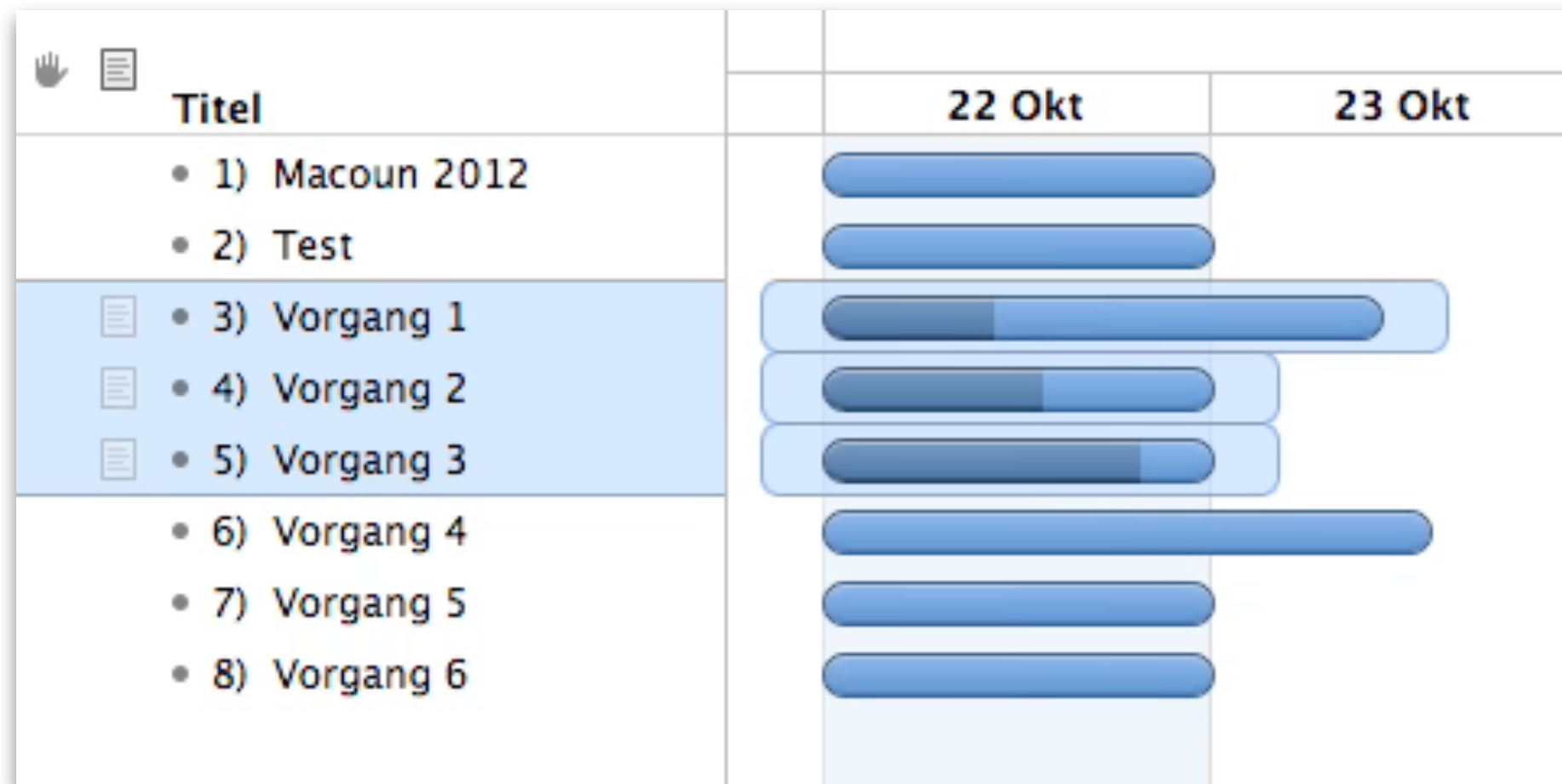
# Goldene Regel

für die Animation einer Ansicht

**Alles oder nichts!**

# Goldene Regel

Nicht so schön



# Goldene Regel

# Besser

[illegible]

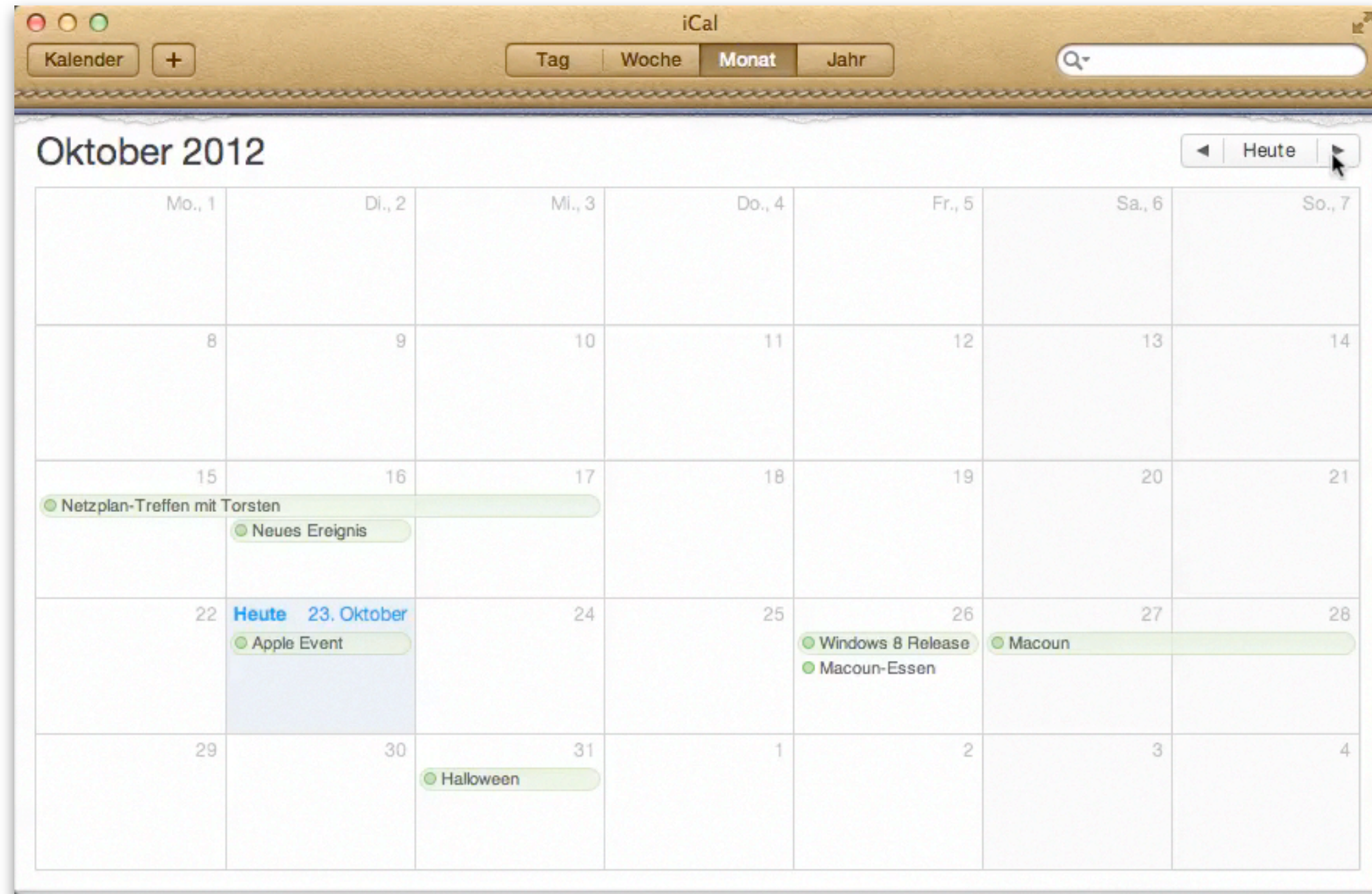
**Was soll Animation nicht?**

# Was soll Animation nicht?

- den Arbeitsfluss behindern

# Was soll Animation nicht?

## Behinderter Arbeitsfluss



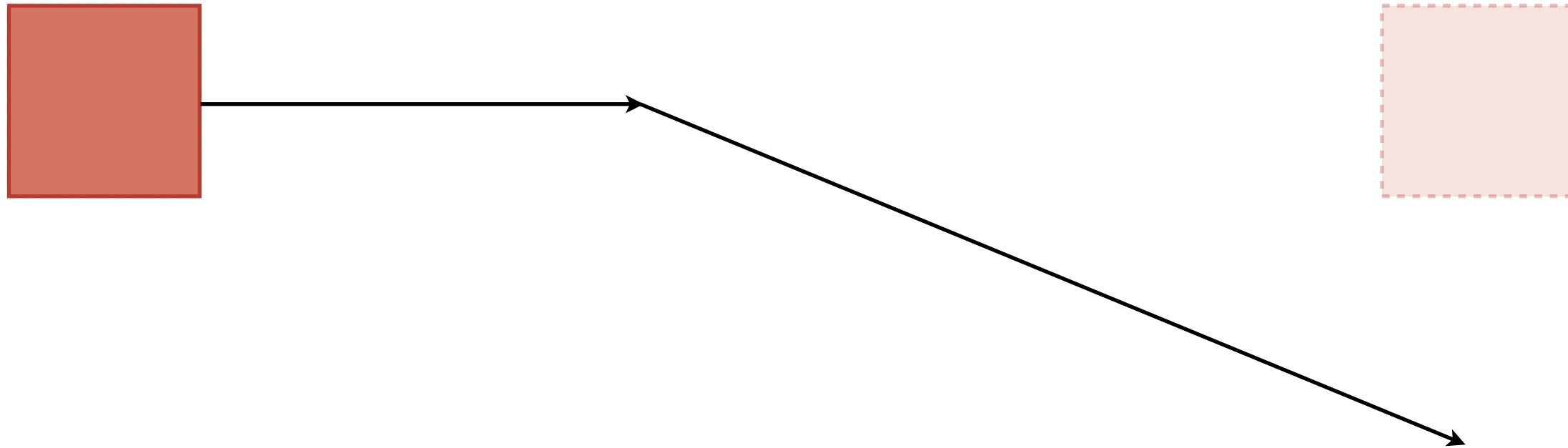


# Was soll Animation nicht?

- Im Arbeitsfluss behindern
  - Der Benutzer darf nicht gezwungen sein, auf das Ende einer Animation zu warten.

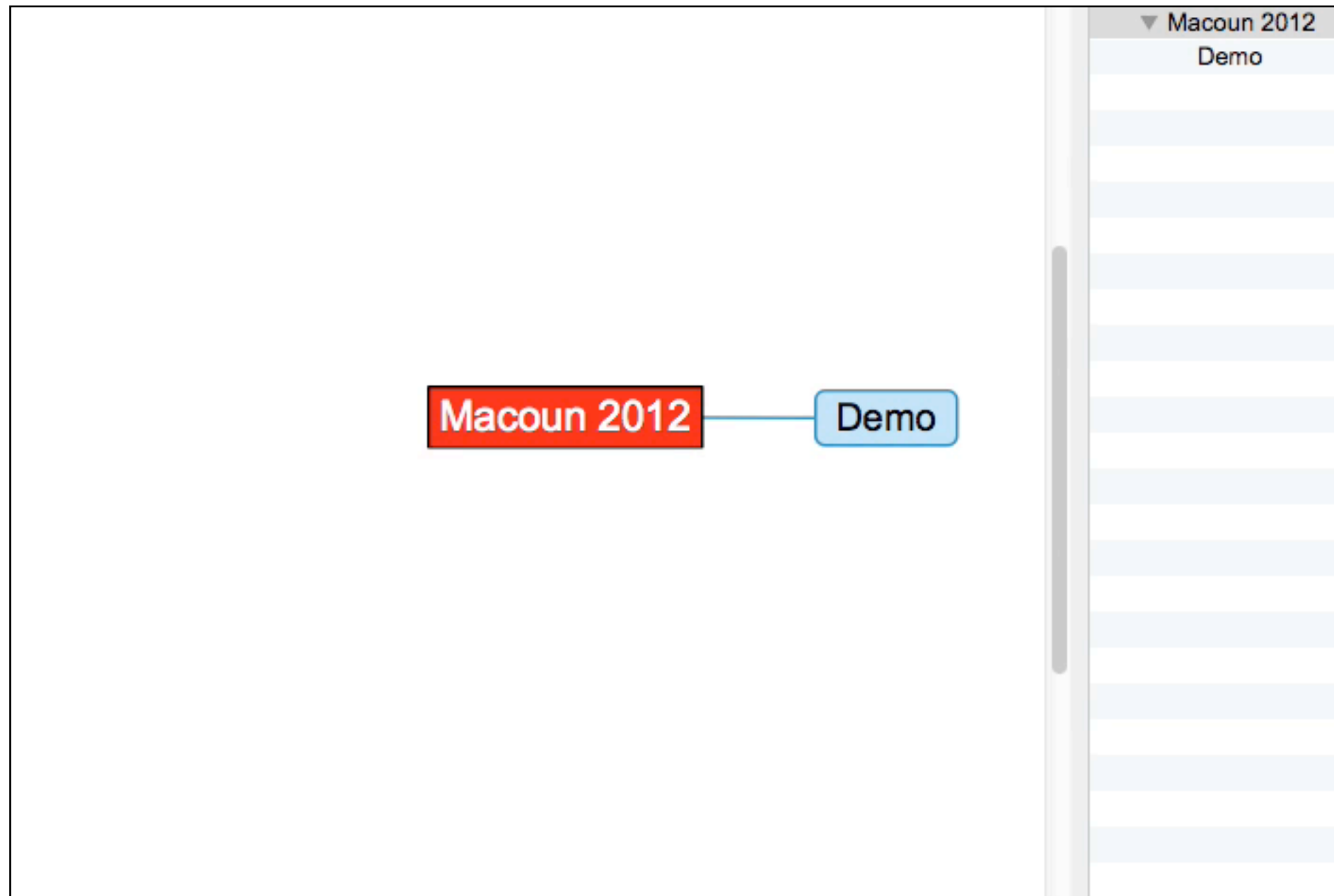
# Was soll Animation nicht?

Besserer Arbeitsfluss - Starten aus Zwischenzuständen



# Was soll Animation nicht?

Besserer Arbeitsfluss - Starten aus Zwischenzuständen?



# Was soll Animation nicht?

Besserer Arbeitsfluss - Starten aus Zwischenzuständen

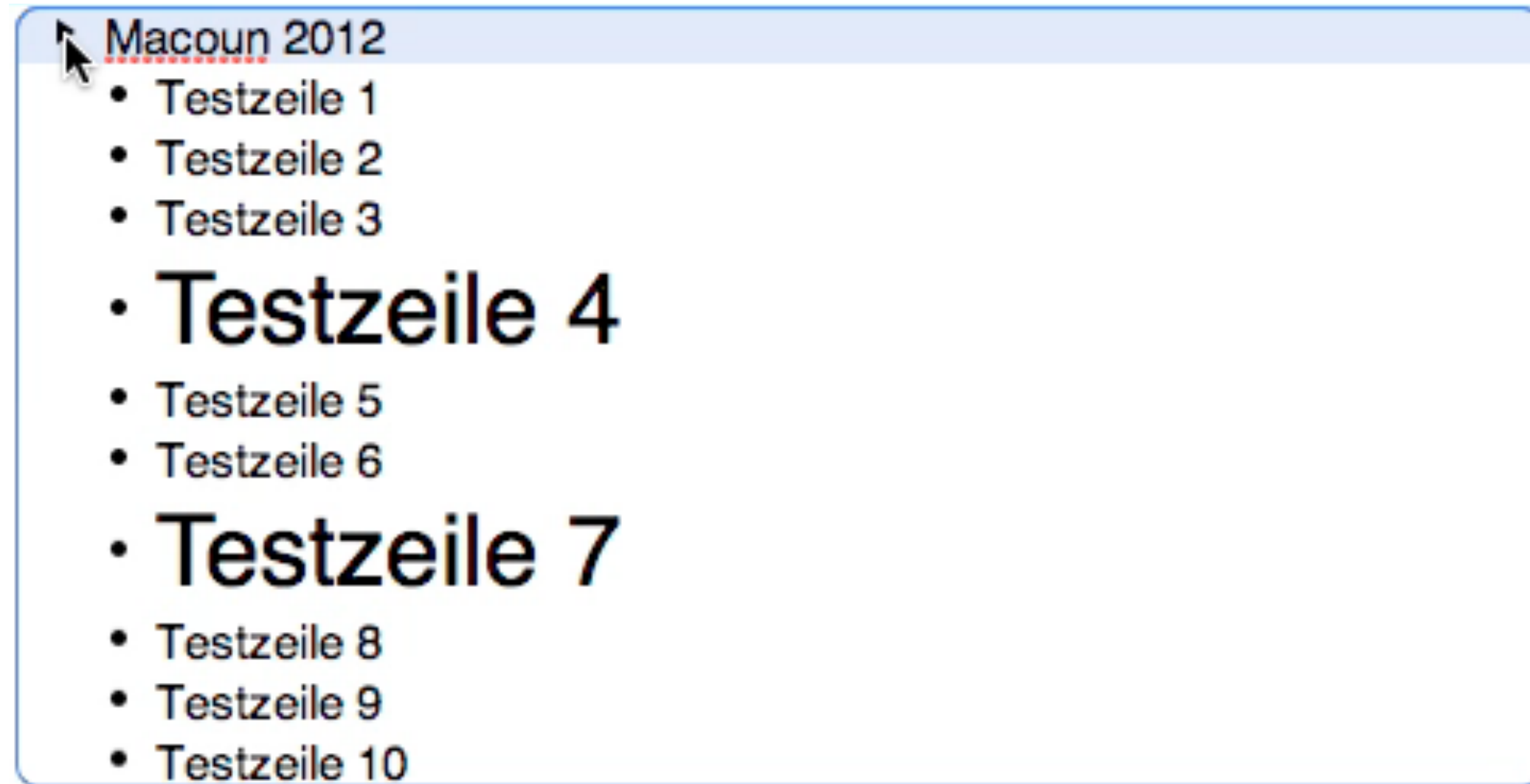
Titel
▼ <b>Macoun 2012</b>
Testvorgang 1
Testvorgang 2
Testvorgang 3
Testvorgang 4
Testvorgang 5
Testvorgang 6

# Was soll Animation nicht?

- Im Arbeitsfluss behindern
  - Der Benutzer darf nicht gezwungen sein, auf das Ende einer Animation zu warten.
- Durcheinander erzeugen

# Was soll Animation nicht?

Durcheinander erzeugen



# Was soll Animation nicht?

Durcheinander erzeugen

Titel
▼ Macoun 2012
Testzeile 1
Testzeile 2
Testzeile 3
Testzeile 4
Testzeile 5
Testzeile 6
Testzeile 7
Testzeile 8

# Was soll Animation nicht?

Durcheinander vermeiden durch Phasen





# Was soll Animation nicht?

Durcheinander vermeiden durch Phasen

Ausblenden

—————→ Zeit

# Was soll Animation nicht?

Durcheinander vermeiden durch Phasen

Ausblenden

Bewegen

→ Zeit

# Was soll Animation nicht?

Durcheinander vermeiden durch Phasen

Ausblenden

Bewegen

Einblenden

→ Zeit

# Was soll Animation nicht?

Durcheinander vermeiden durch Phasen

Ausblenden

Bewegen

Einblenden

Überblenden

Zeit



The diagram illustrates the sequence of animation phases over time. It consists of four horizontal bars of different colors, each containing a phase name. The bars are arranged in a staggered, overlapping fashion from top to bottom: a yellow bar for 'Ausblenden', a blue bar for 'Bewegen', a green bar for 'Einblenden', and a red bar for 'Überblenden'. Below these bars is a horizontal black arrow pointing to the right, with the word 'Zeit' (Time) at its tip, indicating the progression of time.

# Was soll Animation nicht?

Durcheinander vermeiden durch Phasen

Titel
▼ Macoun 2012
Testzeile 1
Testzeile 2
Testzeile 3
Testzeile 4
Testzeile 5
Testzeile 6
Testzeile 7
Testzeile 8

# CoreAnimation

# CoreAnimation

- Hierarchische, View-artige Abstraktion
  - Render-Tree aus CALayer-Objekten
- Implizite Interpolation von Parametern
- Hardware-beschleunigtes Compositing
- Hocheffizientes Cachen auf Bitmap-Ebene

# CoreAnimation

Cache-Ebenen z.B. für darzustellenden Text



Bitmap als Textur auf Grafikhardware (OpenGL)
Bitmap
Text Layout (Anordnung von Glyphen, z.B. CTFrame)
NSAttributedString



# CoreAnimation

Wann ist es ungeeignet?

- Wenn eine View-artige Abstraktion das Problem nicht gut abbildet
- Wenn sich Bestandteile der Ansicht nicht gut durch Rechtecke repräsentieren lassen
- Wenn man abstrakte Koordinaten benötigt
- Wenn Bestandteile der Ansicht viel größer sind als der Bildschirm
- Wenn die Animationen nicht auf Compositing-Ebene ausgeführt werden können

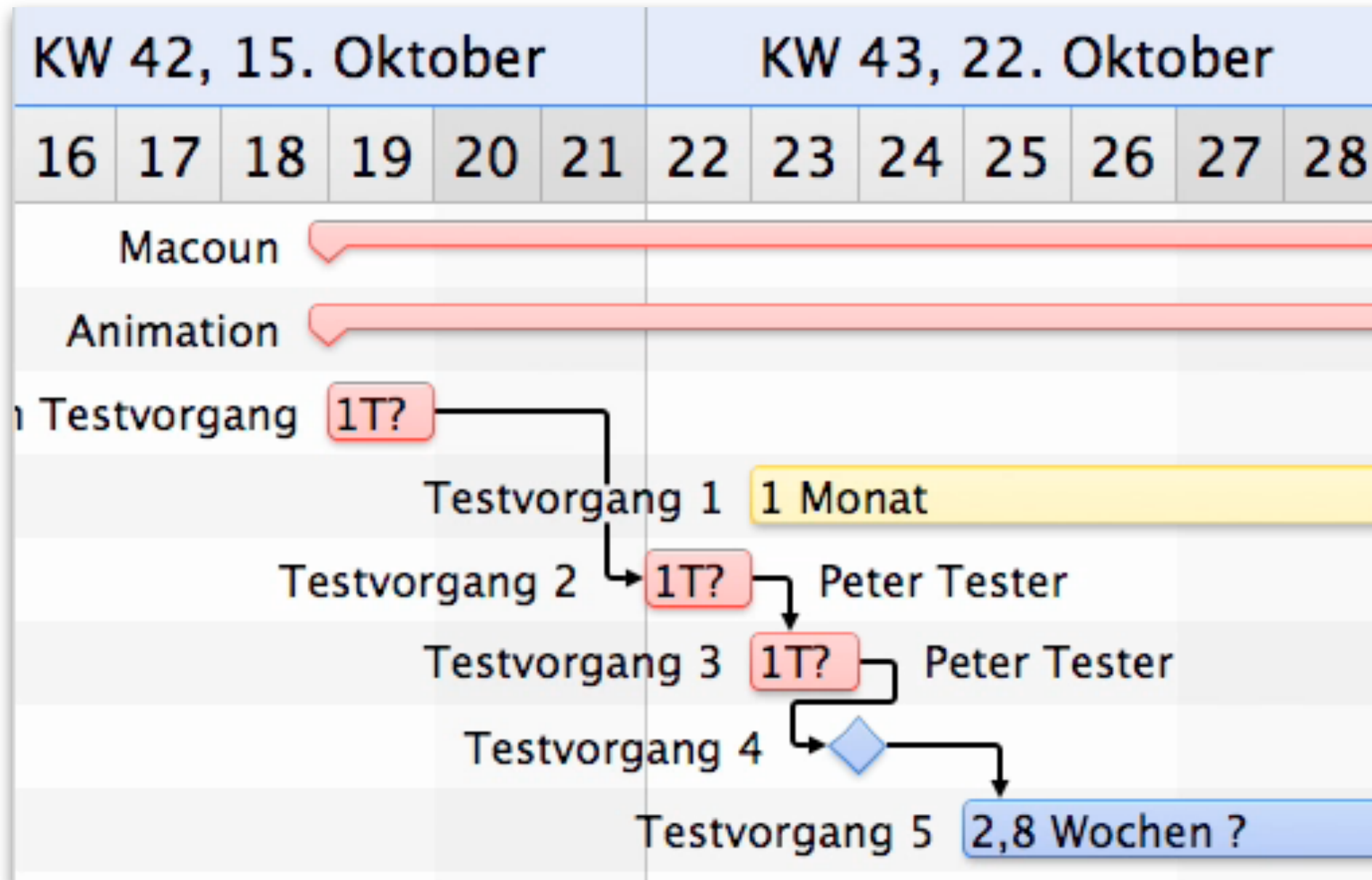
# CoreAnimation

# Wann ist es ungeeignet?

#	Titel	KW 42, 15. Oktober						KW 43, 22. Oktober								
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	
0 ▾	Macoun	Macoun														
1	Animation	Animation 4 Tage ?														
6	Testvorgang 5	Testvorgang 5 2 Tage ?														

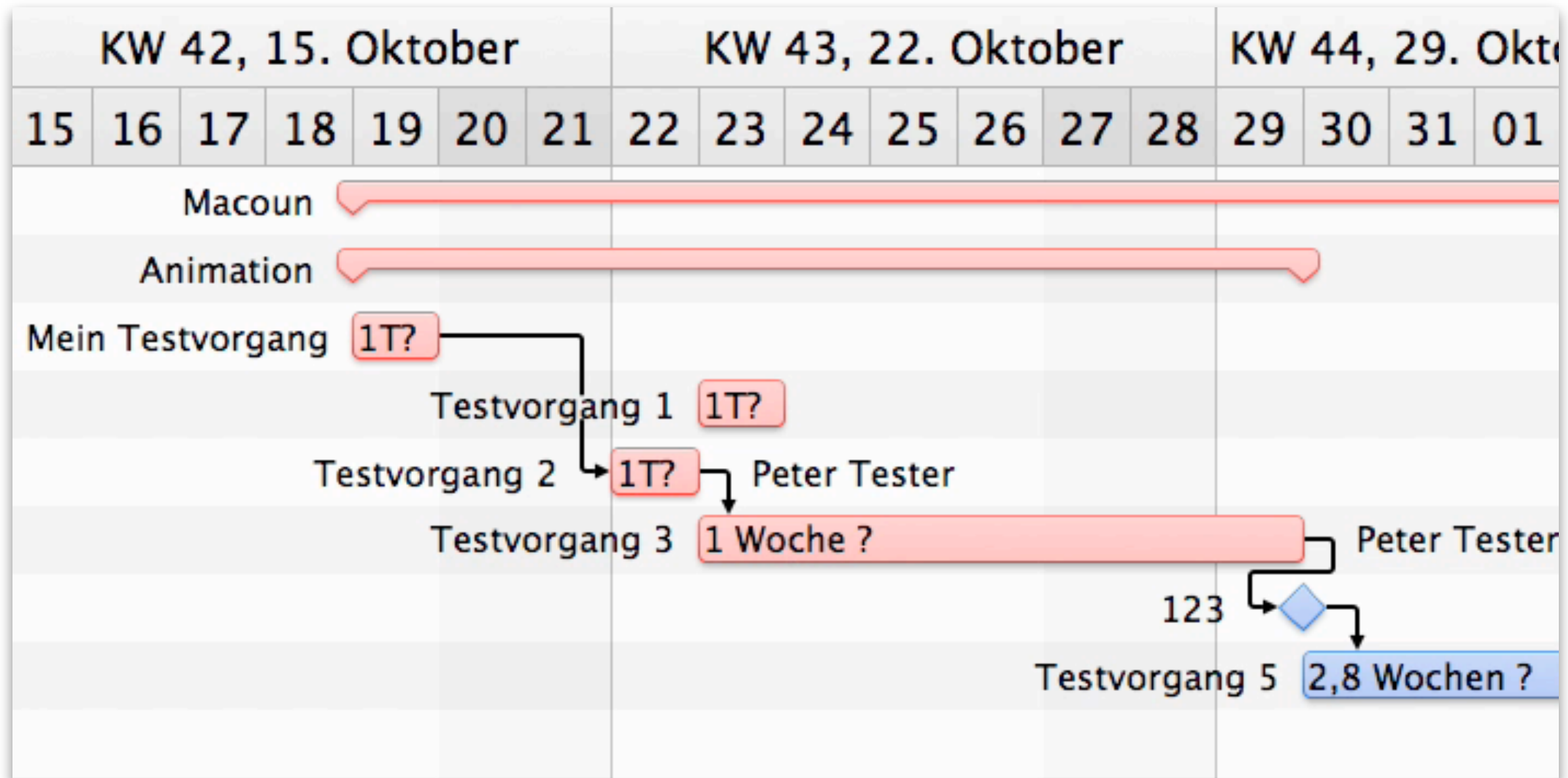
# CoreAnimation

Wann ist es ungeeignet?



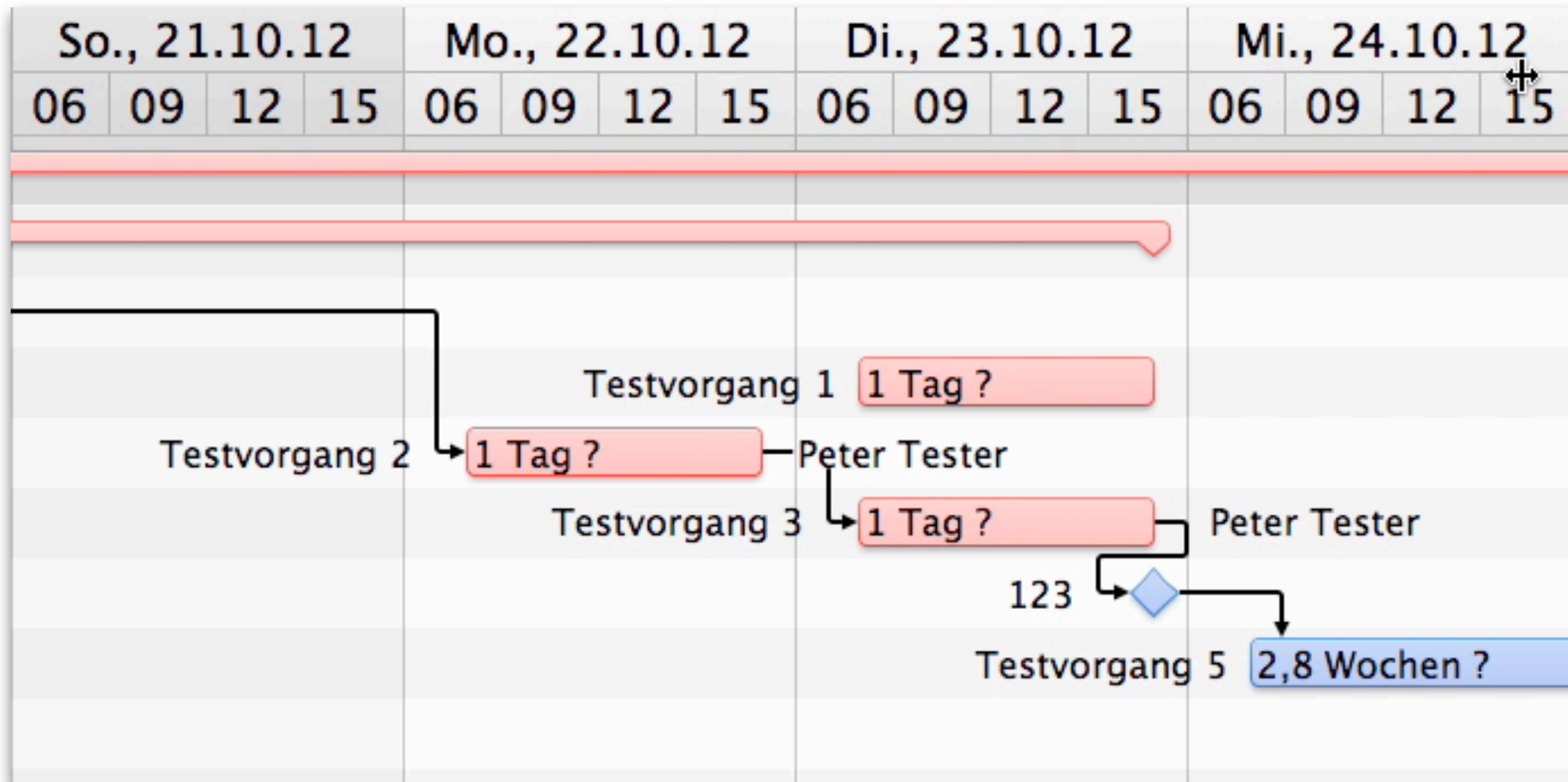
# CoreAnimation

Wann ist es ungeeignet?



# CoreAnimation

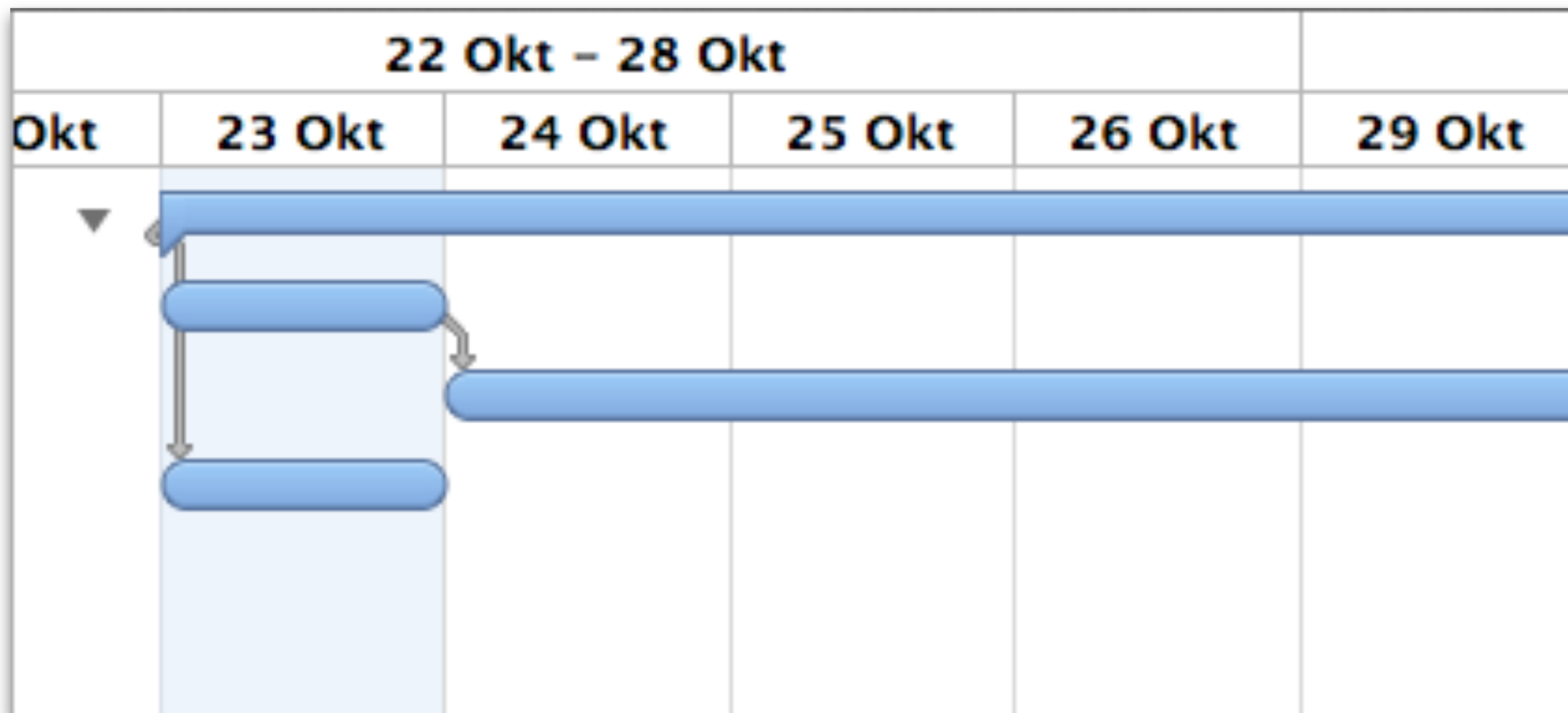
Wann ist es ungeeignet?



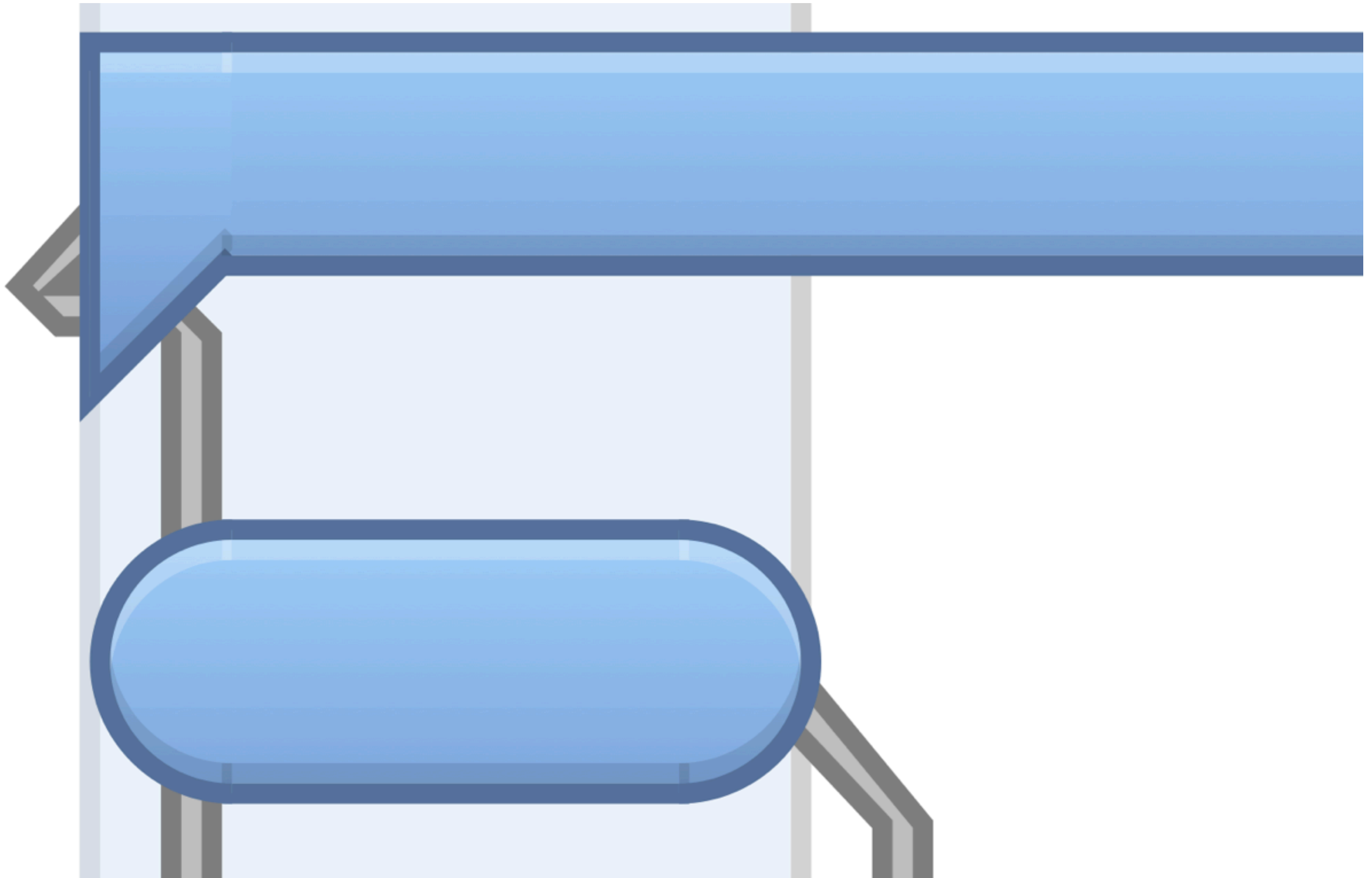
# CoreAnimation

Wann ist es ungeeignet?

Beispiel: Gantt-Diagramm mit CoreAnimation implementiert



# Vergrößerte Druckausgabe



**Animation ohne „Core“**



# Animation ohne „Core“

- Model-View-Controller-Architektur
  - Model darf nichts von Animation wissen.
  - Darstellung soll zwischen Representation der Modellzustände interpolieren.
  - Zur Interpolation benötigt man die vorhergehende Representation.
- Ergo: Controller dürfen Views nicht direkt aus Modelldaten präparieren.

# Render Tree

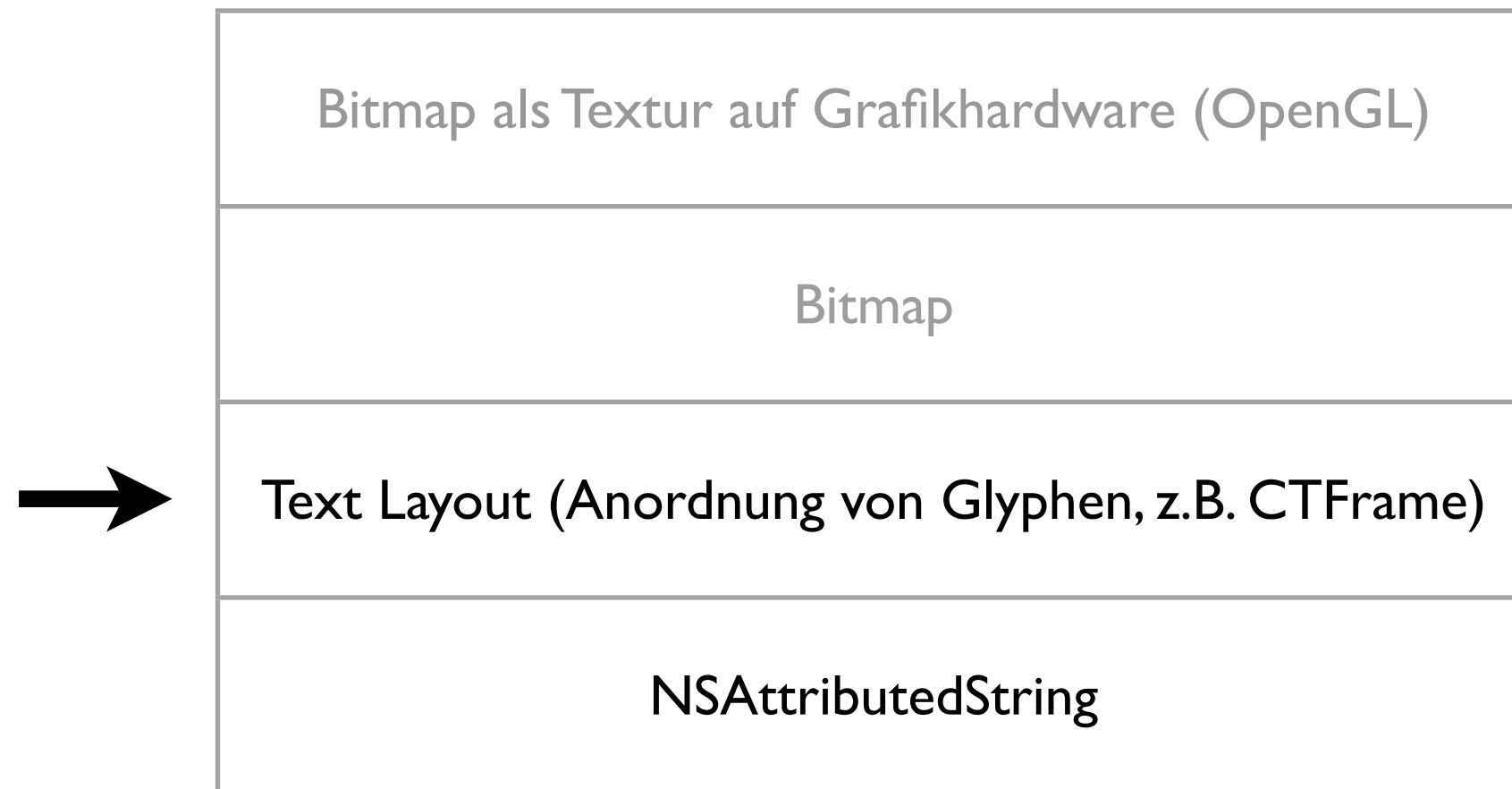
- Neue Zwischenschicht
- Analog zu CALayers jedoch mit beliebiger Abstraktion
- Bekannte Beispiele
  - NSTreeNode
  - Web-Browser

# Render Tree

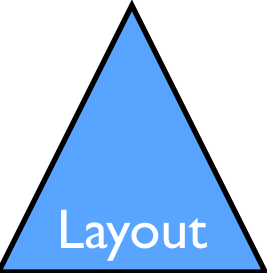
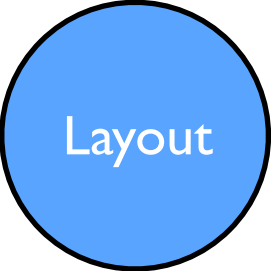
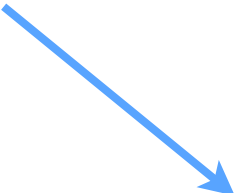
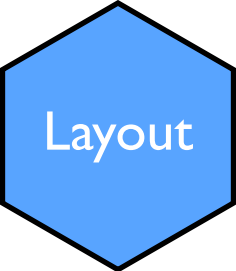
- Knoten werden auf Modellobjekte gemappt (uniqued)
- Knoten übernehmen Layout-Aufgaben
- Knoten observieren Modellobjekte
- Knoten sind kopierbar
  - Referenzknoten repräsentieren vorherigen Zustand
  - Mapping von Knoten auf Referenzknoten
  - wie CALayer: -presentationLayer, -modelLayer

# Render Tree

Cache-Ebene z.B. für darzustellenden Text



PWTreeController



Observing



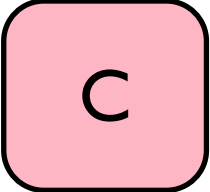
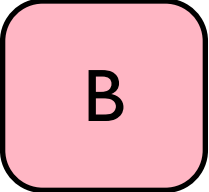
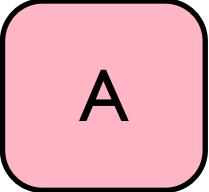
Observing



Observing



Modellschicht

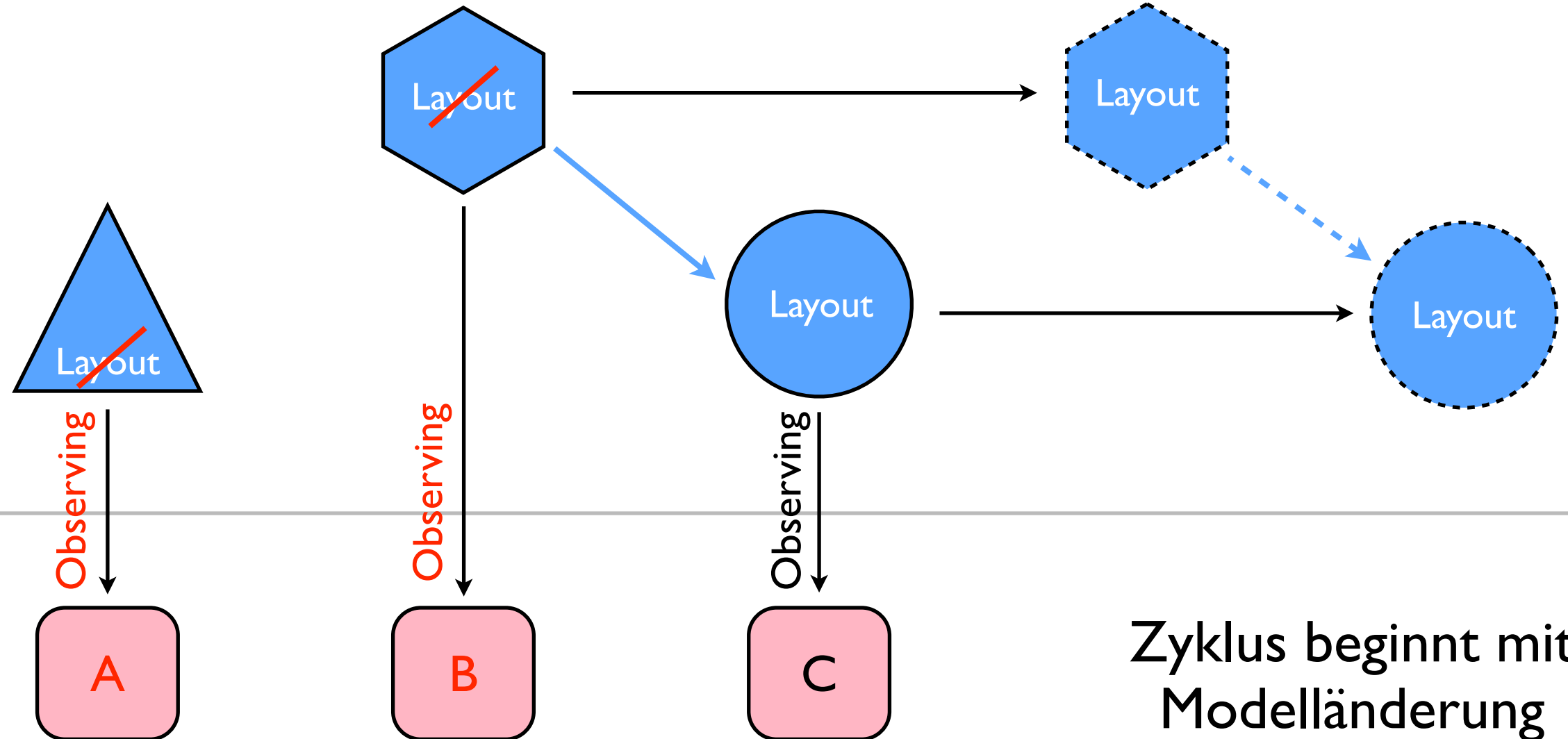


Aktion für Zyklusende: -preflightAnimations

PWAnimationPreparer  
(Mapping)

kopiert sichtbare Knoten

PWTreeController



# Preflight Animations

- Am Ende des Event-Aktions-Zyklus
- Alle Modelländerungen sind passiert
- Alle sichtbaren Render-Tree-Knoten sind kopiert
- Alle benötigten Layout-Caches sind invalidiert
- Nun können wir analysieren, welche Animationen benötigt werden.
  - Neue Animationen werden gestartet, bestehende verändert.  
ViewController besitzt Animationen.
- Animation preparer wird weggeworfen

# Preflight Animations

- PWAnimation
  - sehr leichtgewichtig, volle Kontrolle
  - Teilen sich gemeinsamen Timer PWHeartBeat
  - Animationsparameter  $0 \rightarrow 1$ 
    - Kurvenunterstützung (ease in/out)
- Basisklasse für konkrete Animationsarten, die die dafür benötigten Informationen enthalten, z.B. Referenzen auf Knoten.



# Animationsablauf

- PWHeartBeat feuert
- Alle Animationen durchgehen
  - Für Animation im aktuellen Schritt veränderte View-Bereiche grob identifizieren
  - → -setNeedsDisplayInRect:

# Animationsablauf

-drawRect:

- Zu zeichnende Render-Tree-Knoten identifizieren:
  - alle Knoten, die Teil einer Positionsanimation sind
  - zusätzlich alle Knoten, die im sichtbaren Bereich der Ansicht liegen
- In umgekehrter Reihenfolge zeichnen, damit die sich bewegendenden Objekte obenauf liegen
- Aus Performancegründen sollte die Menge der zu betrachtenden Knoten nicht mit der Gesamtzahl an Objekten skalieren.

# Animationsablauf

-drawRect:

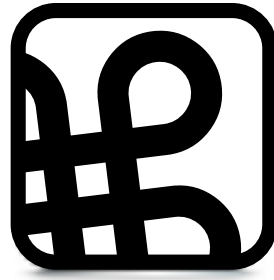
- nutzt Animationsobjekte, um interpolierte Werte zu ermitteln
- über -current...-Methoden in Render-Tree-Objekten
- zieht an On-Demand-Layout-Methoden, die wenn benötigt ihre Caches neu befüllen
- Typischerweise ist an allen Layout-Caches jedoch schon während der Vorbereitung gezogen worden.

# Zusammenfassung

- Animation ist schwierig!
- Wenn es ausreicht, auf CoreAnimation setzen.
- Falls es nicht ausreicht:
  - T-Shirts drucken
  - früh aufstehen
  - warm anziehen
  - hohen Entwicklungsaufwand einplanen
  - über schöne Animationen freuen!

Fragen?

**Vielen Dank**



**Macoun**