Integration von Algorithmus-Animationen in eine Visuelle Programmiersprache

nach einem Paper von

Paul Carlson, Margaret Burnett
und Jonathan Cadiz

Inhalt des Vortrags

- **Forms/3:** Deklarative VPL, die One-Way-Constraints benutzt.
- 2 Animationsbeispiele:
  - 1. Bsp. zeigt, wie Animationen mit Hilfe von One Way Constraints programmiert werden.
  - 2. Bsp. zeigt, wie man einen in dieser VPL programmierten Algorithmus animieren kann.
- Ergebnisse und Folgerungen.

One Way Constraints in Forms/3

- Programmieren in Forms/3 nach dem Spreadsheet-Paradigma:
  - direkte Manipulation, um Zellen zu plazieren.
  - Definition von Formeln für jede Zelle.
  - Formeln ergeben "Netz von One-Way-Constraints".

    $\Longrightarrow$ System stellt sicher, daß Werte, die auf dem Schirm dargestellt werden, diese Constraints erfüllen.

Bsp. für Constraints: Digitale Uhr

- **Cell counterSec:**

```
0 fby (earlier (counterSec + 1))
```

- **Cell Sec:**

```
0 fby
(if ((counterSec mod 60)=0)
 then 0
 else (earlier (sec + 1)))
```

- **Cell counterMin:**

```
0 fby
(if ((sec mod 60)=0)
 then (earlier (counterMin + 1))
 else (earlier counterMin))
```

- **Cell Min:**

```
0 fby
(if ((counterMin mod 60)=0)
 then 0
 else (earlier (Min + 1)))
```

- **Cell Hour**.....

Grundlagen der Programmierung in Forms/3

- Forms/3 ist "deklarativ" und "responsiv":
  - **Deklarativ:** Programmierung besteht darin, die Beziehung zwischen Inputs und benötigten Outputs zu definieren.
  - **Responsiv:** Immer wenn "neue Information" hinzukommt, werden die Änderungen direkt und automatisch angezeigt.

$\Longrightarrow$ Programmierer muß sich nur um die Spezifizierung der Animation kümmern und nicht etwa um Details wie Update der Anzeige oder Synchronisation mit Algorithmus.

- Konzept des Path/Transition-Paradigmas: Objekt wird entlang eines Pfades ”animiert”.

- **Transition:** Bewegung eines Objekts vom Startpunkt des Pfades zum Endpunkt.

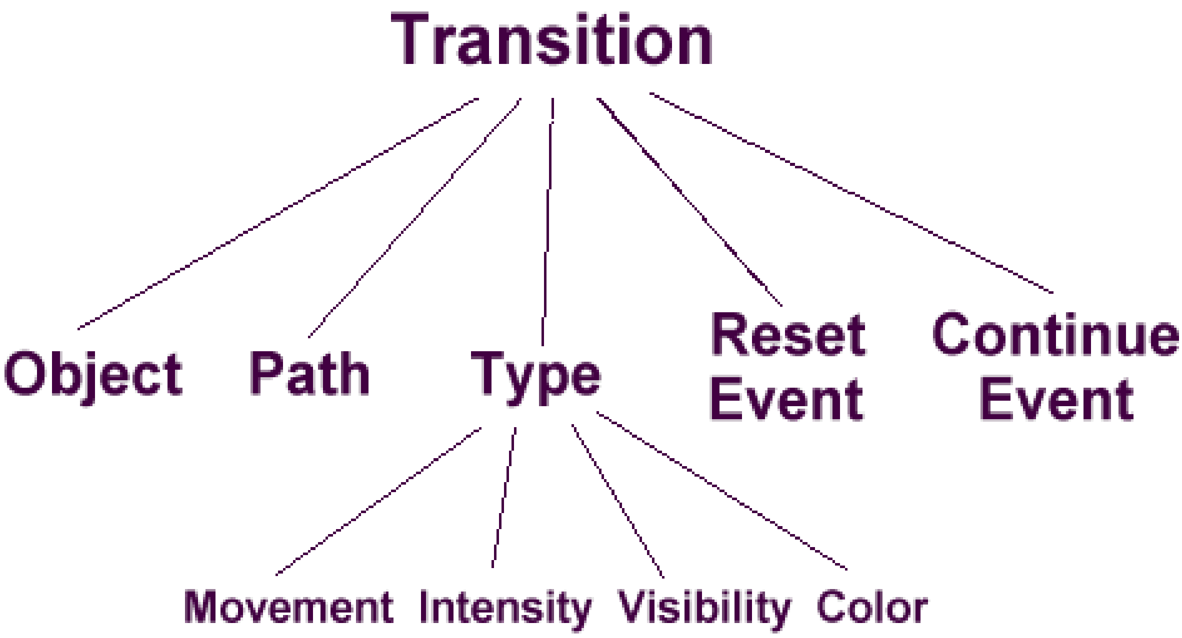

Abbildung 1: Transition in Forms/3

- **Pfad:** Endliche Sequenz von x-y Koordinatenpaaren.
- **Objekt:** Dieses wird vom Start- zum Endpunkt des Pfades bewegt.
- **Typen:** Für unterschiedliche Typen müssen verschiedene Pfade definiert werden.
- **ResetEvent:** Constraint, der zu Beginn einer Transition erfüllt sein muß.
- **ContinueEvent:** Constraint, der erfüllt sein muss, um nächsten "Schritt" entlang des Pfades zu gehen.

Ein Animations-Beispiel aus der Sicht des Benutzers

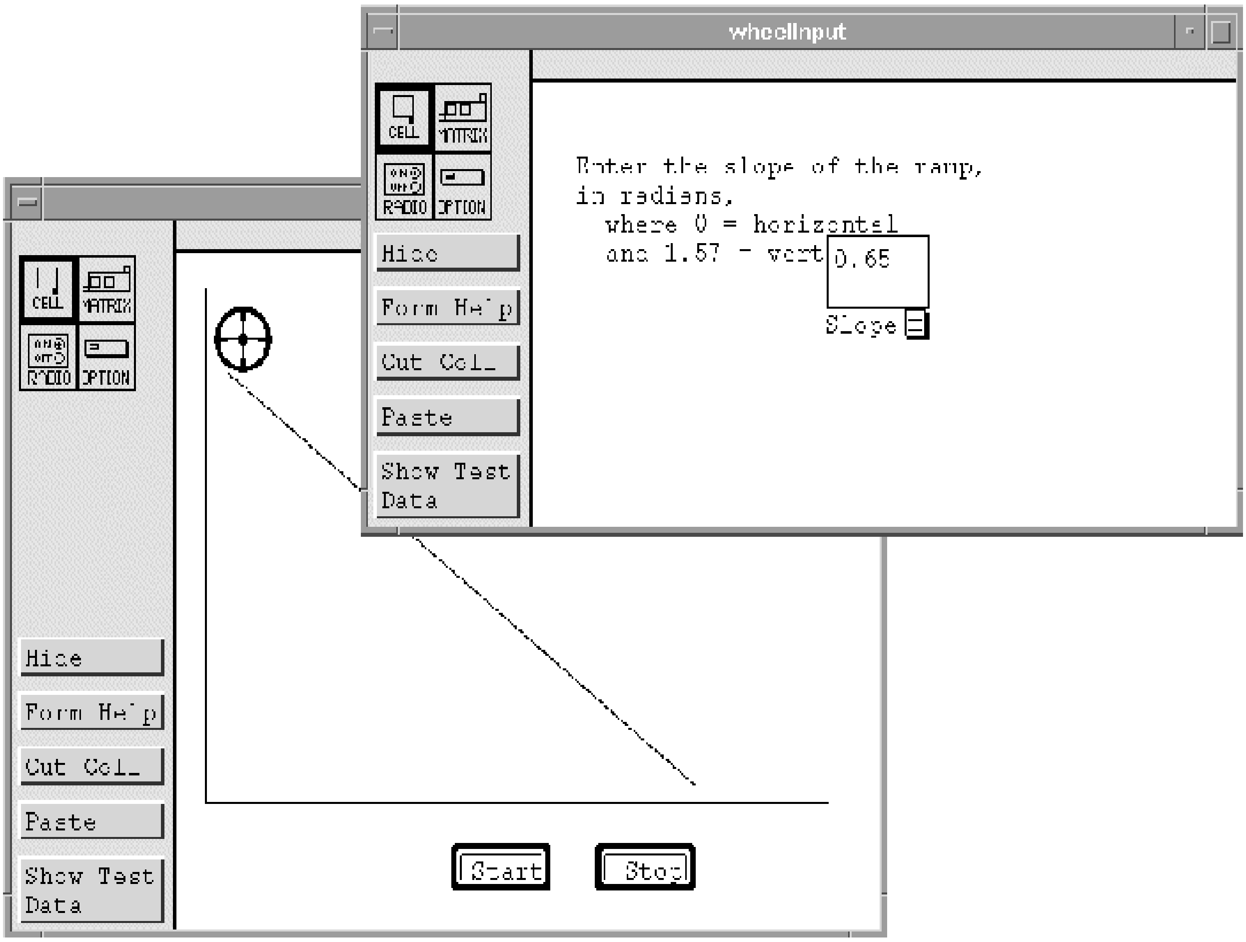

Abbildung 2: Rollendes Rad aus der Sicht des Users

Das Beispiel aus der Sicht des Programmierers

- Animationsparameter werden über sog. "Animation-Form" spezifiziert.
- **Objektmatrix:** Definiert die Objekte, die animiert werden.
  - Objekte:
    1. primitive Objekte (boxes, text strings...).
    2. benutzerdefinierte graphische Objekte (people...)
    3. komplexe Objekte, die aus anderen Berechnungen resultieren.
  - Bei mehreren Objekten in Objektmatrix werden diese zyklisch durchlaufen und angezeigt.

- **Pfad:** Drawn Path oder Computed Path
  Zum Beispiel:
  - **Startpunkt:** (0,0)
    Endpunkt: Abhängig von Steigung der Rampe
    $\Longrightarrow$ Constraint setzt Endpunkte gleich:
    ```
    end:  (wheelInput:rampBotton-x,
          wheelInput:rampBotton-y)
    ```
  - Zelle **Steps** enthält Anzahl der Punkte des Pfades.
    ```
    steps:  round(40-(wheelInput:slope * 10))
    ```
    $\Longrightarrow$ Rad wird in Animation mit wachsender Steigung schneller.

- **Type:** Wird durch Radio-Buttons ausgewählt.

- **resetEvent:** Muss zu *true* evaluieren, um Animation zu starten.

```
resetEvent:  true fby false
```

$\Longrightarrow$ Bei diesem Constraint kein Reset möglich!

- **continueEvent:** Muss zu *true* evaluieren, um Animation auszuführen.

```
continueEvent = false  fby
  (if (start:whatEvent?=''ButtonPress'')
    then  true
   else (if (stop:whatEvent?=''ButtonPress'')
    then false))
```

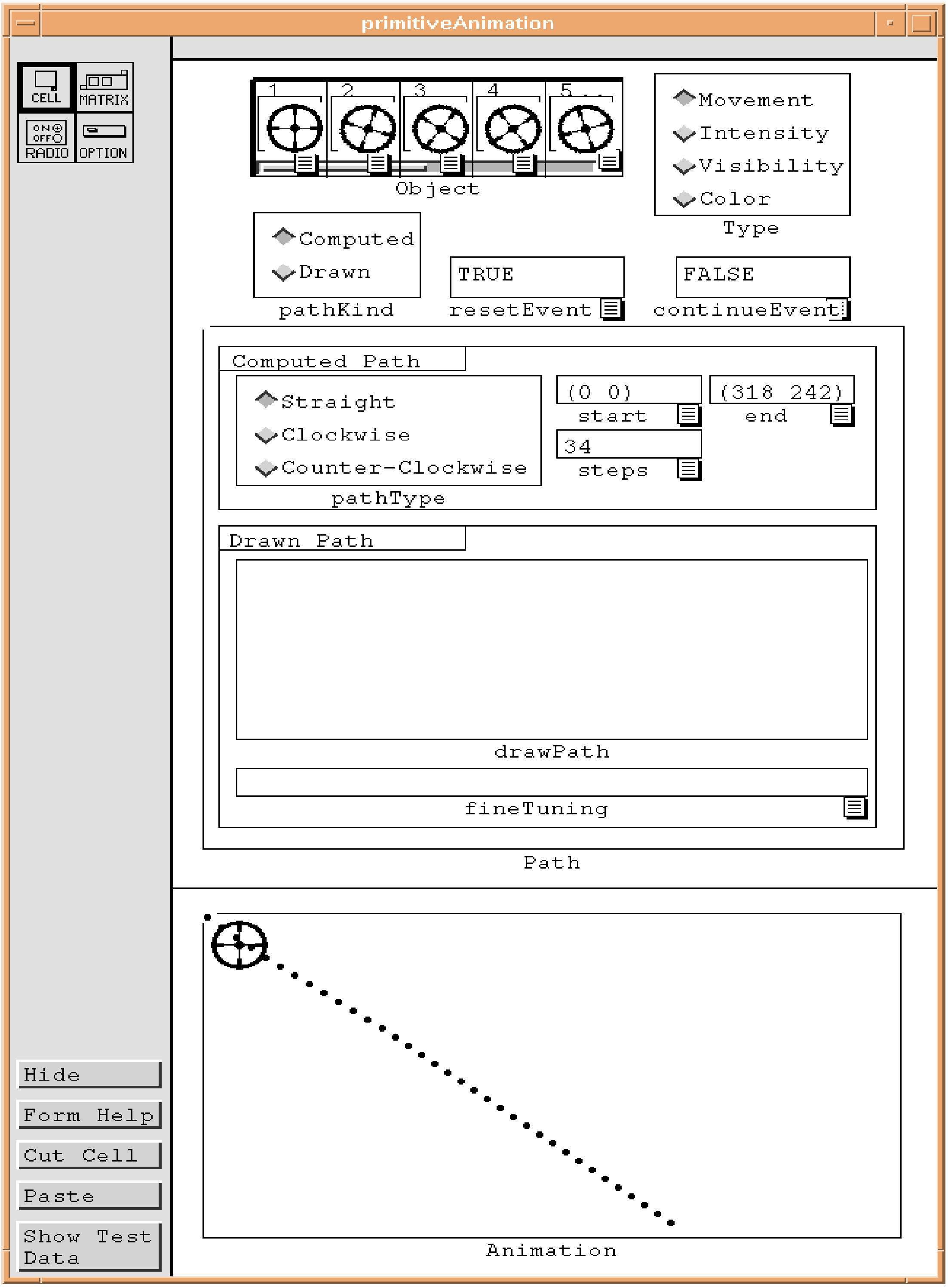

Abbildung 3: Animation-Form

2.Bsp: Sortieren

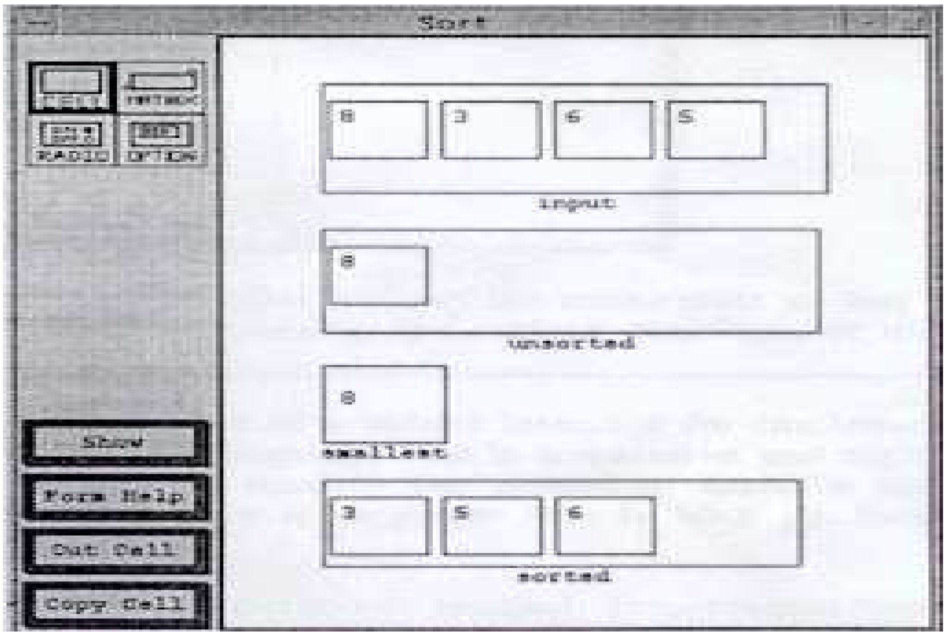

Abbildung 4: Users View

- Einfaches Selection Sort.
- 4 Matrizen: Input, Unsorted, Smallest und Sorted.
- Constraint des Sortieralgoritmus: Jedes neue Element der Liste "sorted" ist das kleinste der Liste "unsorted".

Die Constraints im Sortieralgorithmus

- **Cell Counter:**

```
0 fby (earlier (counter + 1))
```

- **Matrix Cell Unsorted:**

```
input fby
(if ((counter mod 11)=0)
 then (earlier (removeOne unsorted smallest))
 else (earlier unsorted))
```

- **Matrix Cell Smallest:**

```
if ((counter mod 11)=0)
 then \"\"
 else (min unsorted)
```

- **Matrix Cell Sorted:**

```
empty fby
(if ((counter mod 11)=0)
 then ( earlier (append sorted smallest))
 else (earlier sorted))
```

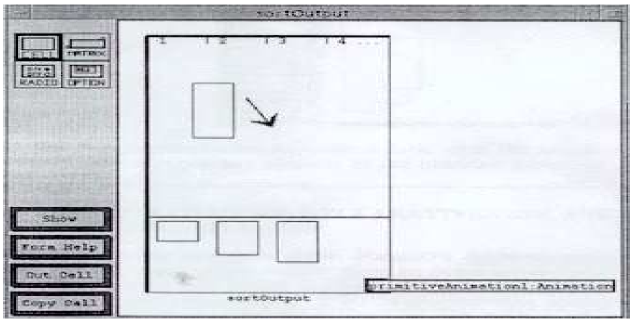

Abbildung 5: Animation

- Jeder "Kasten" gehört zu einem Element der Input-Matrix.
- **Erzeugen der Animation:**
  - Programmierung der Animations-Form für ein Element der Matrix.
  - Automatische Generalisierung, d.h. Animation Form wird auf alle Elemente der Input-Matrix übertragen.
    $\Longrightarrow$ Animation für bel. viele Elemente.

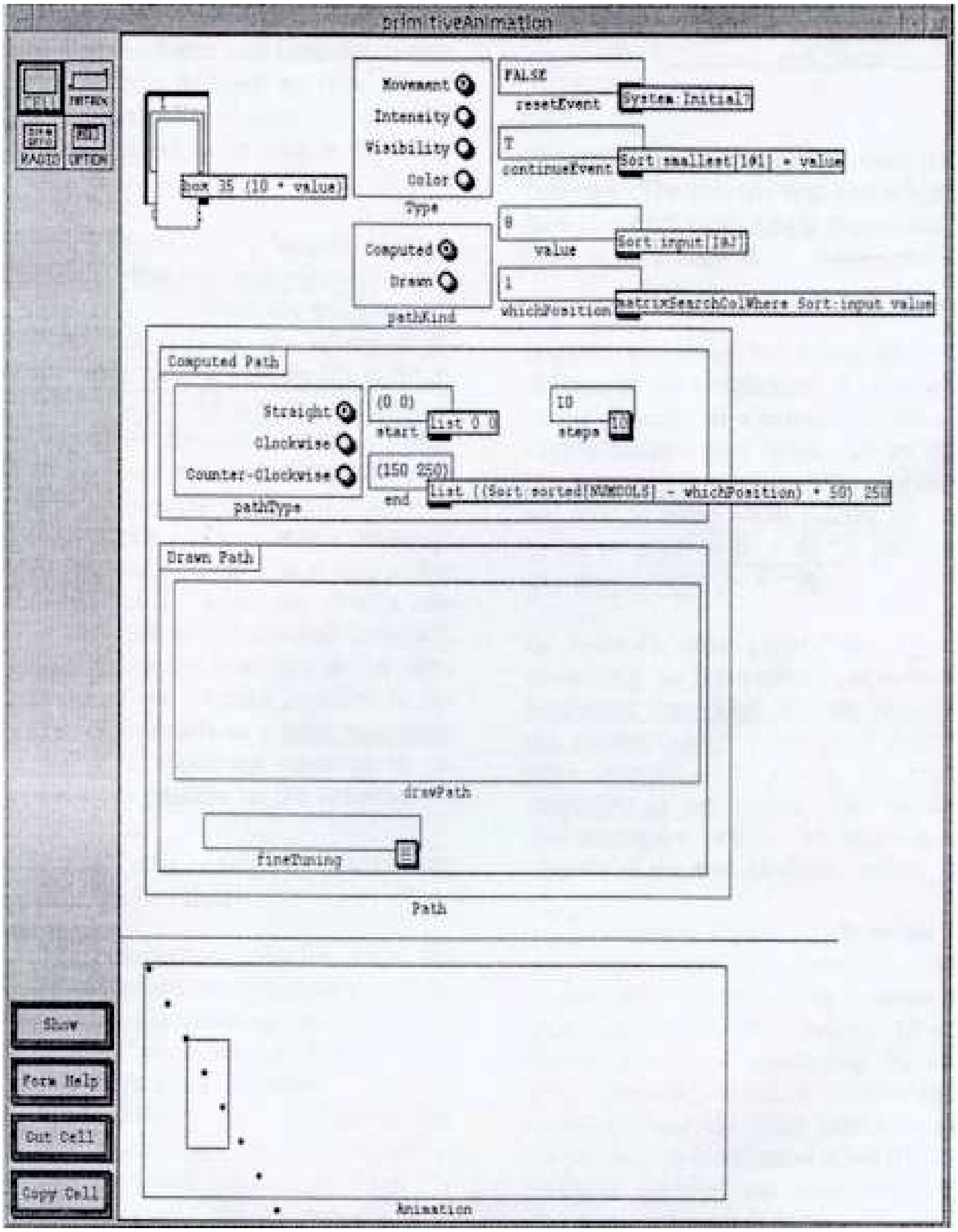

Abbildung 6: Animation

Ergebnisse und Folgerungen

- Bisher gesehene Animationssysteme basieren auf der Festlegung von **"interessanten Ereignissen"**. Das Äquivalent hierzu ist in Forms/3 die **resetEvent-Zelle**. Dieser Constraint wird kontinuierlich evaluiert.
  $\Longrightarrow$ Nicht nur Ereignis, welches Start definiert, sondern feste Bedingung des Programms.
  $\Longrightarrow$ Spezifikation ausreichend, d.h. Programmierer muss nicht festlegen, wann oder wie Bedingung erfüllt ist.
  $\Longrightarrow$ **Trennung von Animation und Algorithmus.**

- Im P/T Paradigma wird die Transition nur durch 3 Parameter festgelegt:
  type, image und path.
  Zum Ausführen der Transition wird der Befehl **"perform"** benuzt.
  In Forms/3 wird dieser Befehl durch resetEvent und continueEvent ersetzt.

- Modifikation von Constraints während der Ausführung möglich; dabei sofort visuelles Feedback.

- Ausführung von Programmen sowohl "vorwärts" als auch "rückwärts".

Informationen über Forms/3 und VPL:

http://www.cs.orst.edu/∼burnett/Forms3