Georg Lambert *12.Mai.98*

# Das ZEUS-System für Algorithmen-Animation und Multi-View-Editing

- *Algorithmen-Animation* ist eine Form der Programmvisualisierung, bei der grundlegende Operationen des Programms durch dynamische, interaktive Animationen dargestellt werden.

- *Zeus* ist ein System für Algorithmen-Animation.

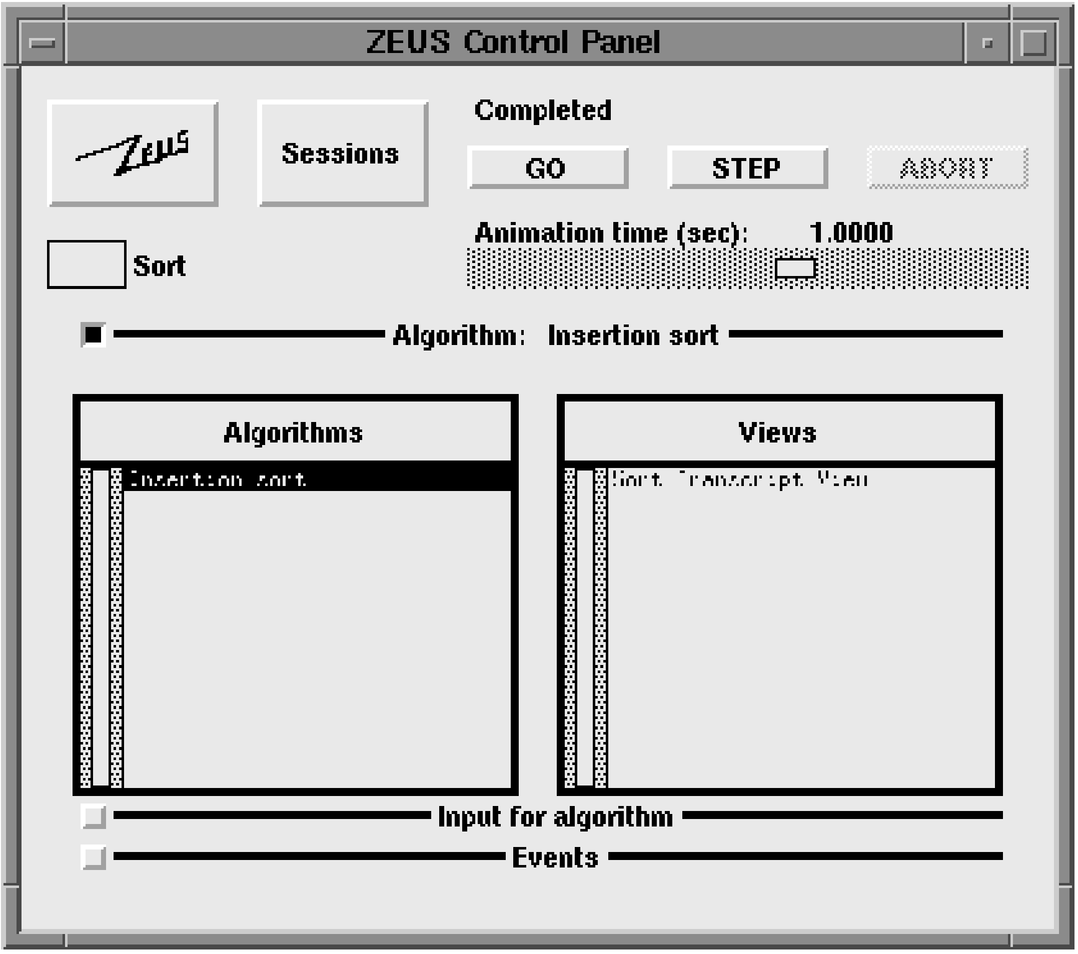
# Inhalt des Vortrags

1. Die User-Perspektive
2. Die Programmierer-Perspektive
3. Implementierung von ZEUS
4. Multi-View-Editoren auf ZEUS-Basis

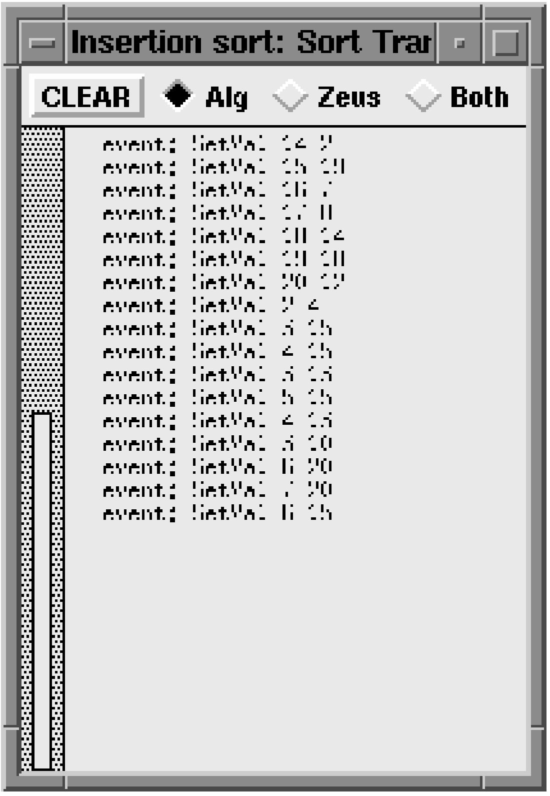
# 1. Die User-Perspektive

- *Central Control Panel*
  - Konfigurationskontrollen (Auswahl des Algorithmus, Auswahl der Anzeigendisplays, Eingabe der Daten des Algorithmus, Snapshot...)
  - Wiedergabekontrollen (Start, Stop, Single step, Geschwindigkeit,...)

- *Standard Displays* basieren auf den $IE$ (*interesting events*) und werden automatisch generiert.
  - Transcript View (textuelle Ausgabe der $IE$'s mit den aktuellen Parametern)
  - Control Panel (Button für jedes $IE$ mit veränderbaren Parametern, erlaubt Aufruf des entsprechenden $IE$'s)




Das Central Control Panel




Die Transcript View

# 2. Die Programmierer-Perspektive

- Für Programmierer bietet ZEUS
  - eine plattformunabhängige Umgebung
  - Steuerung mehrerer *Views* gleichzeitig
  - die Möglichkeit, mehrere $IE$'s pro *View* zuzulassen, gekoppelt mit einer entsprechenden Animation

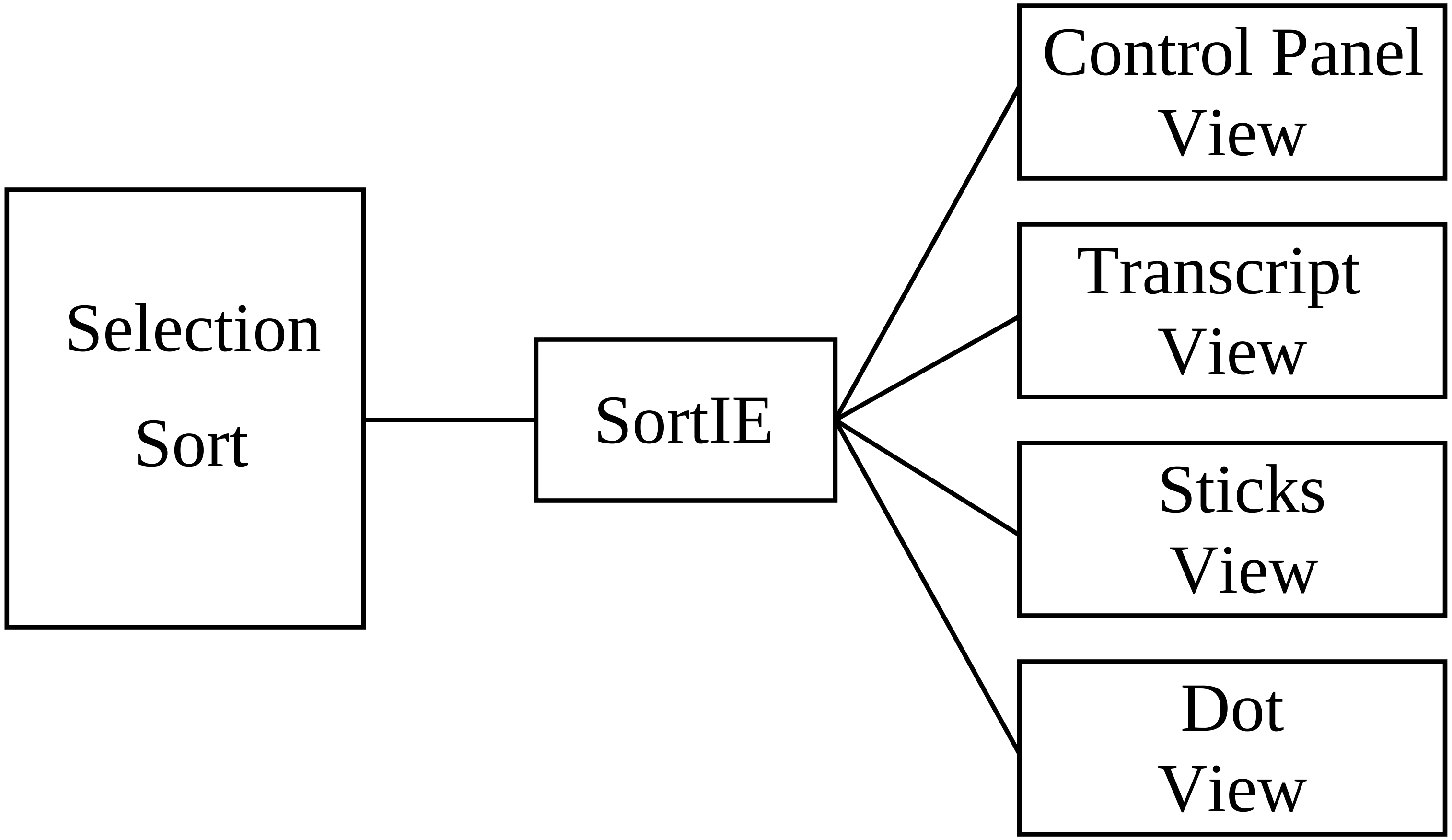
- *Zeus* basiert auf Modula3 und ist daher
  - objektorientiert
  - *type-checking* (Typprüfend)
  - leicht erweiterbar

## Grundlegendes Prinzip von ZEUS:

- Der Programmierer legt interessante Events ($IE$) im Algorithmus fest (*Annotation*).

- Er verbindet diese $IE$'s mit den Animationseinheiten wie graphisches Display oder Sound.

- Beim Durchlaufen des Algorithmus werden beim Auftreten eines $IE$ die entsprechenden Funktionen ausgeführt.




Output Events laufen von links nach rechts,
Feedback Events von rechts nach links.

## Implementierung einer Algorithmenanimation

1. Festlegen der $IE$'s des Algorithmus und Schreiben einer *Event Specification*

2. Implementierung des Algorithmus und Einfügen der Aufrufe der Funktionen, die die $IE$'s generieren, an den entsprechenden Stellen

3. Implementierung der Displays, welche die Events des Algorithmus empfangen, und Darstellung der Events als Animation.

Beispiel: *Insertion Sort Algorithmus* mit *Stick View*.

## Festlegen der $IE$'s und Schreiben der Event Spezifikation

File *Sort.evt*:

```
OUTPUT SetN (N:INTEGER [ZFmt.Int]);
OUTPUT SetVal (i:INTEGER [ZFmt.Int];
                val:INTEGER [ZFmt.Int]);
```

- `OUTPUT` legt fest, daß das Event ein Output-Event ist
- `SetN` und `SetVal` sind die Namen der Funktionen
- `INTEGER` Schlüsselwort für Variablentyp
- `ZFmt.Int` ist Name der Funktion, die Integer in Text konvertiert

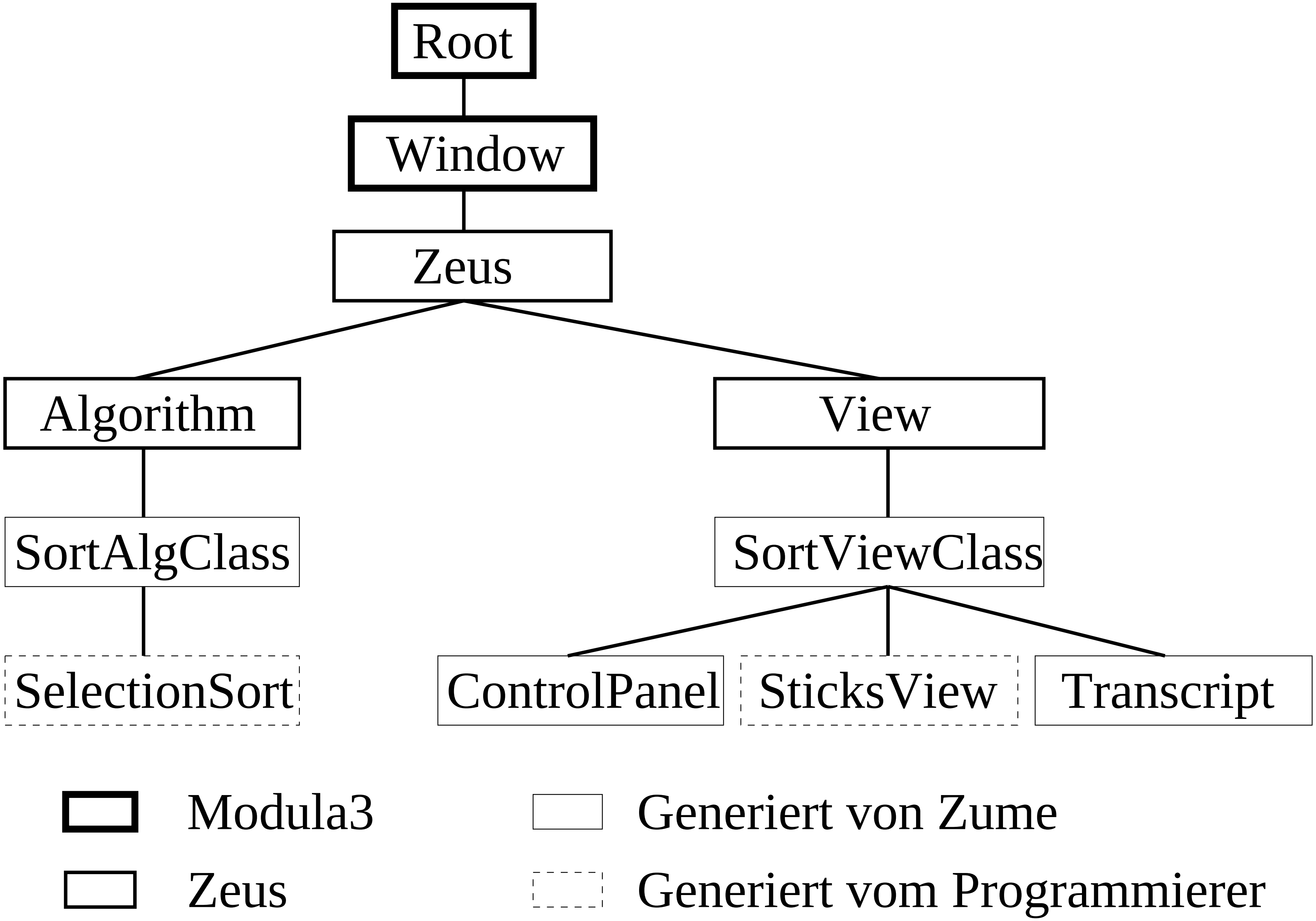
Vorverarbeitung mit $m3zume$ auf *Sort*.evt liefert folgende Modula3 files:

- *Sort*IE.i3 bzw. *Sort*IE.m3 mit dem Interface bzw. der Implementierung des neuen Moduls *Sort*IE

- *Sort*AlgClass.i3 und *Sort*AlgClass.m3 mit der Basisalgorithmenklasse *Sort*AlgClass.

- *Sort*ViewClass.i3 und *Sort*ViewClass.m3 mit der Basisviewklasse *Sort*ViewClass.

- weitere Files, die z.B. Standardviews enthalten.


Root
Window
Zeus
Algorithm
View
SortAlgClass
SortViewClass
SelectionSort
ControlPanel
SticksView
Transcript
Modula3
Zeus
Generiert von Zume
Generiert vom Programmierer

## Implementierung des Algorithmus

- $m3zume$ hat ein entprechende Algorithmenklasse generiert (*Sort*AlgClass)

- Das Interface dieses Moduls stellt Datentyp $T$ bereit (Typ eines Sortier-Algorithmus)

- Ableiten des neuen Moduls (Algorithmus) von dieser Klasse

- Überschreiben der abgeleiteten Funktion $run$ mit dem Algorithmus

```
(***** Begin of file Algs.i3 *******************)

 INTERFACE Algs;
 END Algs.

(***** End of file Algs.i3 *********************)

(***** Begin of file Algs.m3 *******************)

 MODULE Algs;

 IMPORT SortAlgClass, SortIE;
 IMPORT Algorithm, Thread, ZeusPanel;

 TYPE
  InsertionSort = SortAlgClass.T BRANDED OBJECT
   a : ARRAY[0..101] OF INTEGER;
   N : INTEGER;
  OVERRIDES
   run := Run;
  END;

 PROCEDURE GetData(...)
           RAISES {Thread.Alerted} =
  BEGIN
   ...
  END GetData;
```

```
PROCEDURE Run (self:InsertionSort)
          RAISES {Thread.Alerted} =
 VAR j, v : INTEGER;
 BEGIN
   GetData (self);
   WITH a = self.a, N = self.N DO
     FOR i := 2 TO N DO
       v := a[i];
       j := i;
       WHILE a[j - 1] > v DO
         a[j] := a[j - 1];
         SortIE.SetVal (self, j, a[j]);
         DEC (j);
       END;
       a[j] := v;
       SortIE.SetVal (self, j, v);
     END
   END
 END Run;
```

*SortIE* ist ein automatisch von $m3zume$ generiertes Modul. Der Aufruf einer Funktion generiert ein entsprechendes Event, daß an alle *Views* weitergeleitet wird.

```
 PROCEDURE New() : Algorithm.T =                    (1)
  BEGIN
   RETURN NEW(InsertionSort).init();
  END New;

 BEGIN                                              (2)
  ZeusPanel.RegisterAlg (New, "Insertion sort",
                         "Sort");
 END Algs;

(**** End of file Algs.m3 *************************)
```

(1) liefert einen neuen *InsertionSort*.

(2) registriert den neuen Algorithmus im zentralen Kontroller. Der String *Sort* muß das Prefix des Zume Files *Sort.evt* sein.

Zusammenfassung

- Schreiben der Eventspezifikation *[Name].evt* und Anwenden des Präprozessors $m3zume$ liefert uns eine Basisklasse für Algorithmen, die diese Eventspezifikationen nutzen (*[Name]*AlgClass).

- Ebenso liefert dieser Schritt ein Modul (*[Name]*IE) mit aufrufbaren Funktionen, die die entsprechenden Events generieren.

- Man leitet den neuen Algorithmus von der Basisklasse ab und ruft an den entsprechenden Stellen die Eventfunktionen auf.

## Implementierung der grafischen Schnittstellen des Algorithmus

- Implementierung eines Control Panels
- Implementierung einer Grafik
- Implementierung einer Code View
- Implementierung einer Datenübersicht

# Implementierung eines Control Panels

- Control Panels dienen zur Eingabe bzw. Veränderung von Parametern (z.B. Anzahl der zu sortierenden Elemente).

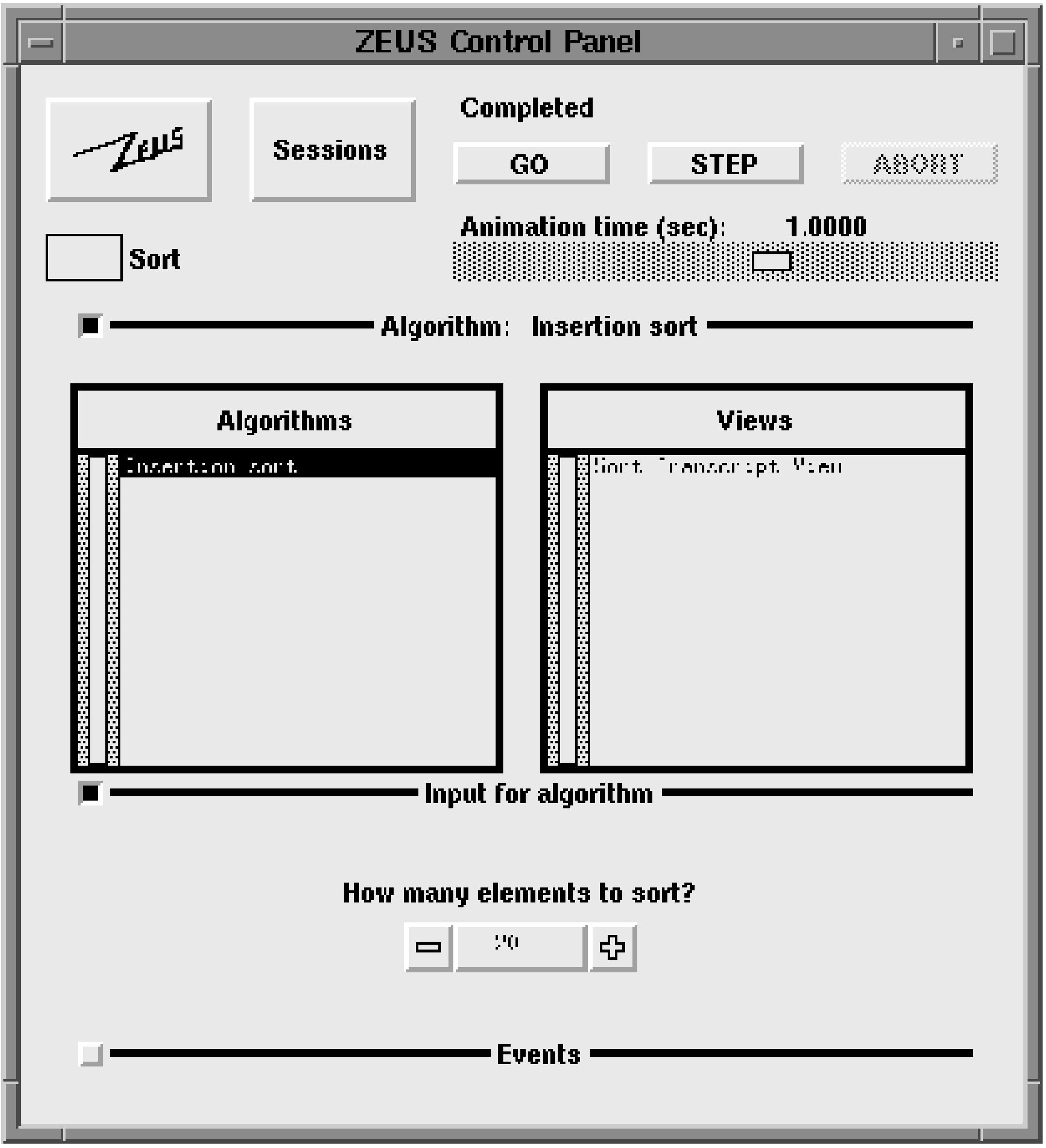
- Festlegen des Aussehens des Panels und der *Interaktoren* (für den Algorithmus lesbare Variablen) in einem *FormsVBT.T* Objekt.

- Verändern des Algorithmus Moduls

  – Es muß ein *FormsVBT.T* Objekt enthalten (i.d.R. *data* von *Algorithm.T*)

  – In *New* muß das entsprechende Objekt kreiert werden

  – Die Variablen des Algorithmus müssen den aktuellen Wert des Control Panels erhalten (mittels der *Interaktoren*)

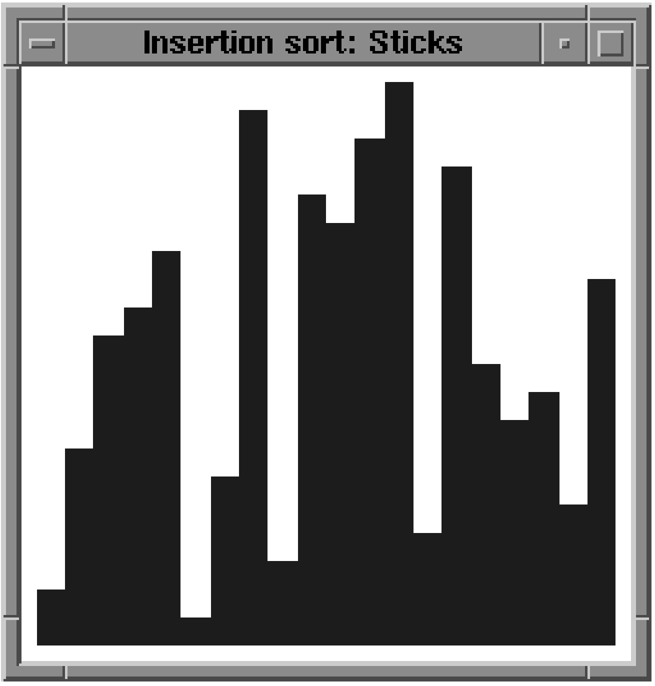

ZEUS Control Panel
Sessions
Completed
GO
STEP
ABORT
Animation time (sec): 1.0000
Sort
Algorithm: Insertion sort
Algorithms
Views
Input for algorithm
How many elements to sort?
Events

## Implementierung einer Grafik

- Festlegen der Art der Grafik und Animation

- Nutzen schon vorhandener Grafikmodule oder Tools (*VBT*, *Anim3D*)

- Programmieren der Grafik unter Beachtung der Events




Beispiel für eine animierte Grafik

Beispiel: *SticksView* in Modula3

- Implementierung des Interfaces für *SticksView*

- Ableiten des neuen Moduls von *SortViewClass.T*, das zwei Methoden *oeSetN* und *oeSetVal* enthält.

- Überschreiben dieser Methoden mit Funktionen, die eine entsprechende Animation erzeugen.

- Implementierung der *New* Funktion

- Registrieren von *SticksView* im *ZeusPanel*

```
(***** begin SticksView.m3 ******************)

MODULE SticksView;

TYPE
  T = SortViewClass.T BRANDED OBJECT
        rects : RectsVBT.T;
      OVERRIDES
        oeSetN   := SetN;
        oeSetVal := SetVal;
      END;

PROCEDURE SetN (self: T; N: INTEGER) =
  BEGIN
    ....
  END SetN;

PROCEDURE SetVal (self: T; i: INTEGER; val: INTEGER) =
  BEGIN
    RectsVBT.Erase (self.rects, i);
    RectsVBT.Position(self.rects, i,
                      FLOAT(i) - 0.5, 0.5,
                      FLOAT(i) + 0.5, FLOAT(val) + 0.5);
    RectsVBT.Draw (self.rects, i);
  END SetVal;
```

```
PROCEDURE New (): View.T =
  VAR
    view := NEW(T, rects := NEW(RectsVBT.T).init());
  BEGIN
    RETURN SortViewClass.T.init (view, view.rects);
  END New;

BEGIN
  ZeusPanel.RegisterView (New, "Sticks", "Sort");
END SticksView.

(***** end SticksView.m3 *************************)
```

Implementierung einer Code View

- Code View läuft während des Algorithmus durch einen Pseudocode und hebt die Operation hervor, die gerade abgearbeitet wird.

- Vorgehensweise der Implementierung:
  - Bilden eines Files, welches den Pseudocode enthält
  - Modifikation des echten Codes, so daß entsprechende Befehle mit den entsprechenden Zeilen des Pseudocodes verbunden werden
  - Modifikationen des Algorithmenmoduls für die Kreation der Code View

Insertion sort: Modula-3 Code View

```
PROCEDURE InsertionSort (a : ARRAY OF INTEGER) =
  VAR
    j, v : INTEGER;
  BEGIN
    FOR i := 2 TO LAST(a) DO
      v := a[i];
      j := i;
      WHILE a[j - 1] > v DO
        a[j] := a[j - 1];
        DEC (j);
      END;
      a[j] := v;
    END;
  END InsertionSort;
```

Beispiel für eine Code View

## Implementierung einer Datenübersicht

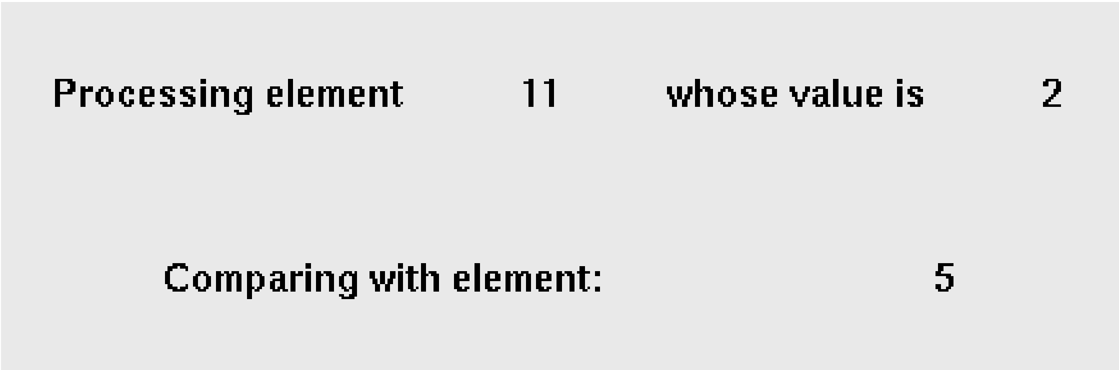
- Eine *Data View* zeigt den aktuellen Wert der Variablen an

- Vorgehensweise der Implementierung

  - Verbinden der darzustellenden Varibalen mit dem $< *TRACE* >$ Befehl von Modula3 $\Rightarrow$ Aufruf einer *changed*-Funktion bei jeder Änderung des Variablenwertes

  - Bilden eines Ausgabedisplays (z.B. wieder auf Basis von *VBT*)

  - Modifikation des Algorithmenmoduls für die Kreation der Data View




Beispiel für eine Datenübersicht

## Zusammenfassung

Man kann folgendes Schema erkennen, um grafische Schnittstellen für den Algorithmus zu implementieren.

- Verändern des Algorithmus, so daß entsprechende Werte an die *Views* exportiert bzw. von diesen importiert werden können

- Design des grafischen Displays

- Eventuell Registrierung des Displays im zentralen *Control Panel* (bei einer Animation)

- Eventuell Aufnahme des Displays als *member* in das Algorithmenmodul (bei *Control Panel*, *Code View* und einer Datenübersicht)

# 3. Implementierung von ZEUS

Zeus umfaßt

- das zentrale Kontroll-Panel
- das Event-Handling
- die default-Funktionen für Algorithmen und Views
- default Views und Panels (*TranscriptView*)

# Das zentrale Kontroll-Panel

- Die Konfigurationskontrollen rufen die entsprechenden *Zeus*-Funktionen des Algorithmus bzw. der Views auf (z.B. *snapshot*)

- Wiedergabekontrollen:

  – *Go*-Button ruft *run* des Algorithmus in eigenem *thread* auf

  – *Abort* beendet diesen *thread*

  – *Step* setzt eine *flag*, die vom Event-Handling beachtet wird und setzt den *thread* fort, wenn er ruht.

  – *Resume* setzt den *thread* fort ohne die *stepFlag* zu setzen.

## Das Event-handling

- Implementiert durch die Event Prozeduren, die vom *Zume* generiert werden.

- Ein Output-Event bildet für jede View einen *thread*, der die entsprechende Funktion aufruft.

- Wenn all diese Funktionen abgearbeitet sind, wird der Algorithmus fortgesetzt, wenn *stepFlag* nicht gesetzt ist oder der *Resume* oder *Step* Button gedrückt wird. *stepFlag* wird disabled.

# Merkmale des ZEUS Systems

- Das System erzeugt Basisklassen, die durch Überschreibung der relevanten Funktionen spezialisiert werden

- Algorithmen werden in Objekten verwaltet, und sind somit *first-class citizens* (können wie Werte behandelt werden)

- Der Programmcode wird unterteilt durch *Interesting Events*

- Alle $IE$'s werden an alle *Views* weitergeleitet, diese filtern die für sie interessanten Events heraus und stellen sie entsprechend dar.

## 4. Multi-View-Editing

- Zeus kann benutzt werden, um Multi-View-Editoren zu bilden.

- Multi-View Editoren beinhalten mehrere editierbare Views, wobei Modifikation einer View durch den User sofortiges Updaten der Daten und der anderen Views zu Folge hat.

- Ein Beispiel ist *FormsEdit*.

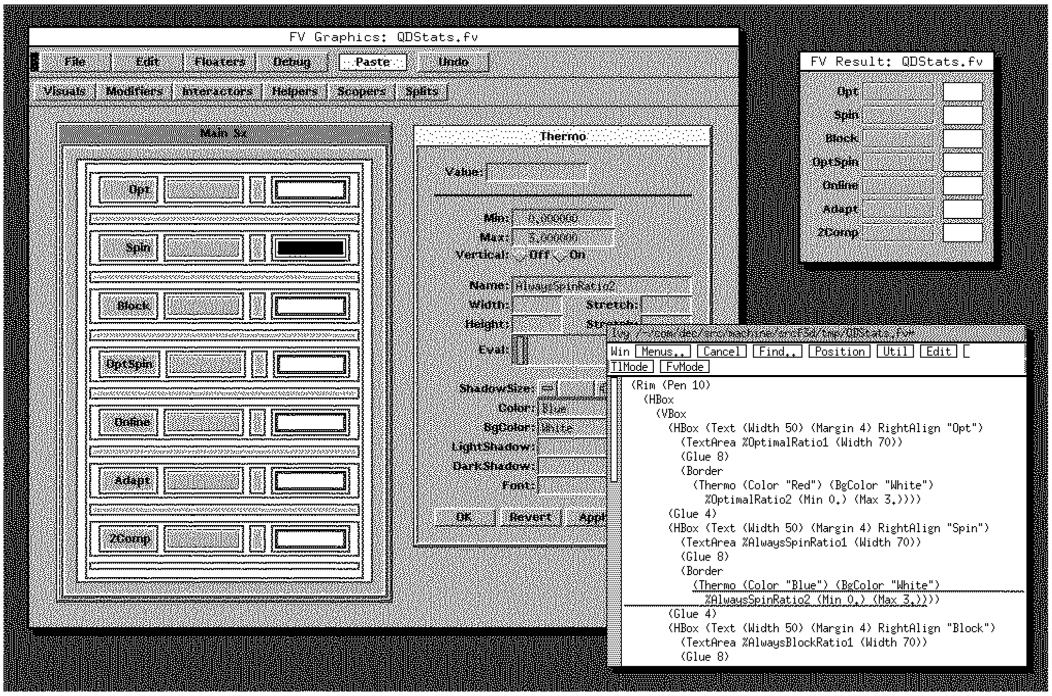
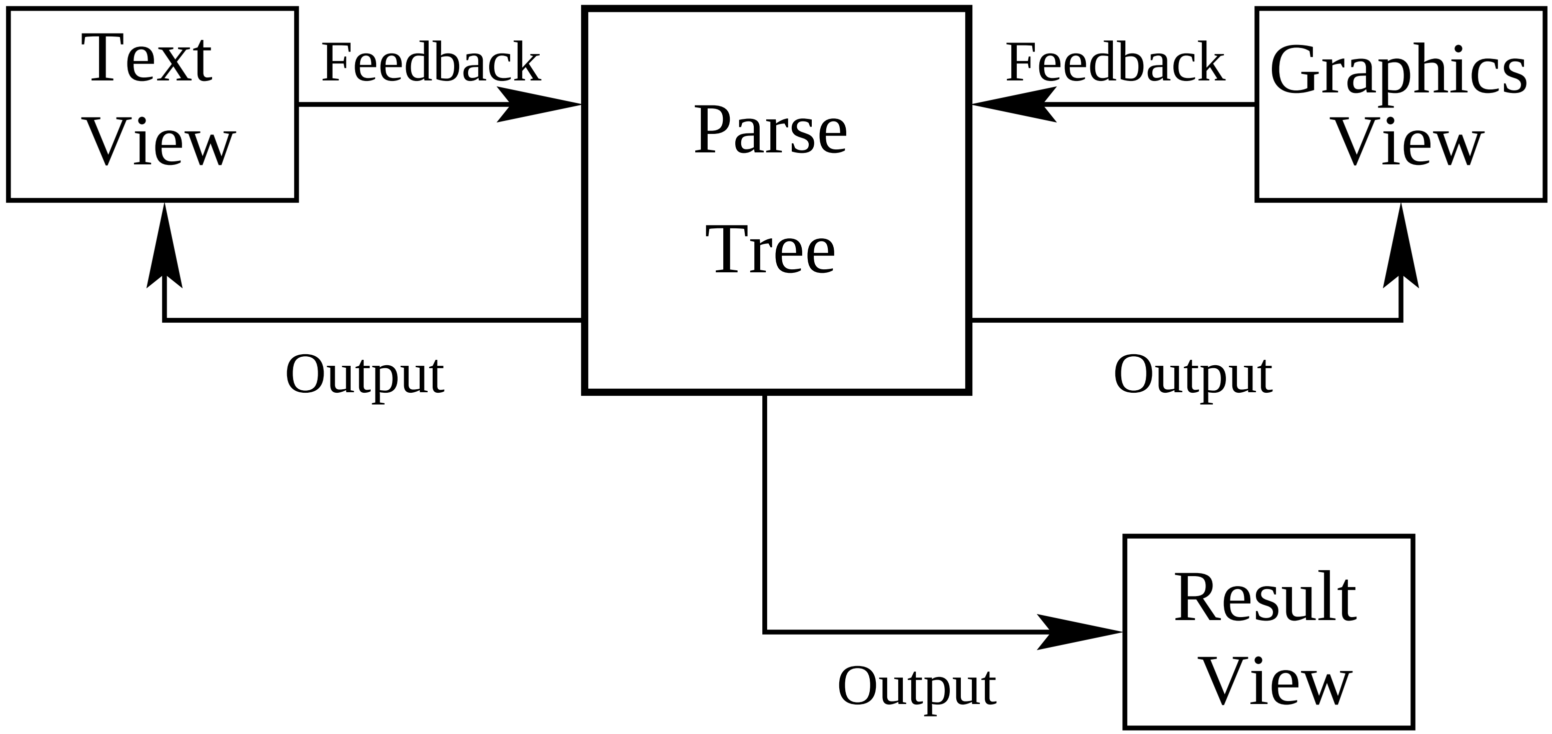



Figure 3: Constructing a view graphically using a multi-view editor.




Informationsaustausch zwischen den Modulen in FormsEdit

# Quellen


- Marc H.Brown:
  Zeus: A System for Algorithm Animation and Multi-View Editing (1992)
  (http://gatekeeper.dec.com/pub/DEC/SRC/research-reports/abstracts/src-rr-075.html)

- Marc A. Najork, Marc H. Brown:
  Building Algorithm Animations with Zeus (1994)

  (http://www.research.digital.com/SRC/zeus/home.html (Zeustutorial))

- Bunte Beispiele:

  (http://www.research.digital.com/SRC/zeus/gallery.html)