Part 5: 4DScript

- Overview
- Syntax
- Syntax examples
- Mathematical statements
- Most used words
  - multiple statements
  - conditional statements
  - repeat statements
- 4DScript editor

4DScript Overview (1)

- 4DScript is a functional language:
  - example: 'if a then b else c' is written '`if(a,b,c)`'
  - Logical and calculation operators can also be written like `2+3` instead of `+(2,3)`
- 4DScript for everything
  - model logic
  - create atoms
  - control ED from outside applications
- 4DScript is auto compiled during run-time
- In logistic suite mostly used to manipulate labels in triggers and conditional statements

4DScript Overview (2)

- 4D script-words overview
- Invoke with menu option

  <window | 4D script overview>

  or

  shortcut F2 when 4DScript editor is open

4DScript Syntax (1)

- 4D Script words have 0..25 parameters
- Parameters between ( )
- Parameters separated by a comma: " , "
- Parameters can be:
  - values
  - strings
  - expressions
- (4DScript) Strings always between square brackets " [..] "
- Comments between `{ }`

4DScript Syntax (2)

- Multiple lines and spaces: OK
- Lower and/or upper case
  - NOT OK in label
- All brackets must match!

4DScript Syntax Examples

- `msg([hello])`
  displays window with text hello
- `/(3,7)`
  divides 3 by 7
- `setchannels(3,2,animatom)`
  set number of in- and output channels of the selected atom in animation window
- `set(color(animatom),0)`
  changes the color of the selected atom in the animation window into black (=0)

Mathematical statements

- Shortcut evaluation order is: * / + - (right to left within same operator type)

Function Call	Shortcut
+(2,5)	2+5
/(9,3)	9/3
-(+(/(*(2,4),2),3),4)	2*4/2+3-4
>(6,2)	6>2

4D Script: Multiple Statements

- "DO" – statement
  - for executing multiple 4dscript commands at once.
  - do(e1,do(e1..e25),..e25)
  - maximum is 25 parameters
- Example for "trigger on exit":

```
do(
 set(color(i),colorred),
 set(icon(i),2)
)
```

shortcut syntax

```
do(
 color(i):=colorred,
 icon(i):=2
)
```


4D Script: Conditional Statements

- if(e1,e2,{e3})
  - e1 = condition
  - e2 = true logic
  - e3 = false logic, not mandatory => 0 is returned
- and/or allowed in condition
- Example:

```
if(
  comparetext(label([ok],i),[yes]),
  negexp(10),
  negexp(50)
)
```


Repeat Statements (1)

- **repeat(e1,e2)**
  - executes e2, e1 times
  - 'count' returns the current iteration number
- Example for "4dScript Interact":

```
repeat(
 content(animatom),
 msg(
  concat(
   [The product ranked number ],
   string(count),
   [ is named ],
   name(rank(count,animatom))
  )
 )
)
```


Repeat Statements (2)

- **loopuntil(e1,e2,e3)**
  - e1 = condition
  - e2 = statements
  - e3 = maximum repetitions
- use '**count**' for the current number of loops made
- omitting e3 might result in endless loop

sum statement

- **sum(e1,e2)**
  - executes e2, e1 times and returns the sum
- Example for total output across several atoms:

```
sum(nric(c),output(in(count,c)))
```

Similar words:

- **maximum**
- **minimum**
- **mean**
- **stdev**

indexmin statement

- **indexmin(e1,e2)**
  - executes e2, e1 times and returns the count at which e2 had the lowest value.
- Example for sending to channel connected to atom with the lowest content:

```
indexmin(nroc(c),content(out(count,c)))
```

Similar words:

- **indexmax**
- **indexmatch**
- **indexmatchclosest**
- **indexmatchrank**

case statement

- **case(e1,e2,e3 {,e4...e25})**
  - e1 = defines case number to be executed (result must be 1 to number of cases)
  - e2 = case 1 statement
  - e3 = case 2 statement
  - etc.
- Example for user interface:

```
case(
  popupmenu(200,100,[happy/sad/mad]),
  msg([This is case 1],1),
  msg([This is case 2],2),
  msg([This is case 3],3)
)
```

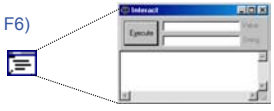

Atom Statistics

Is available for every atom at any time:

- **age** - time from creation or reset
- **content** - current number of atoms contained in an atom
- **avgcontent** - average number of atoms contained in an atom since reset
- **avgstay** - average time (sec.) atoms have stayed in an atom since reset
- **input** - the number of atoms which have entered
- **output** - the number of atoms which have exited
- **status** - the state of an atom (see table of atom T029-Statuslist)
- **entrytime** - time (sec.) at which an atom has entered

4DScript Editor

- To delete code make block
  - backspace
  - delete
- Caution!! When “Auto Brackets” is on:
  - press backspace to select code between matching ( )
  - press backspace twice to delete code between matching ()
- Interact window (shift F6)
  - Apply 4DScript directly on model



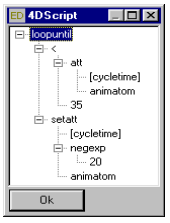
4DScript Editor Functionality

Right mouse click for functionality list:

- Auto Brackets
- Auto Select
- 4Dscript View
- Select Word (F2)
- Previous Word (F3)
- Next Word (F4)
- Get Word (F8)
- Check Syntax (F10)

4DScript Editor Functionality

- Color View (F11)
- Tree View (F12)
- Comment View (Ctrl+O)
- Select All (Ctrl+A)
- Find Replace
- Print
- Save to file
- Load from file



Part 6: Atom Referencing

- About atom referencing
- In Entry and Exit triggers
- Direct referencing
- Relative referencing

About atom referencing

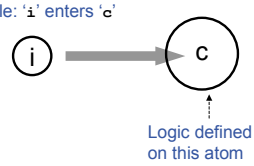
- Referencing in ED is like using a pointer
- You need to reference other atoms to:
  - get information or data from that atom
  - send atoms or messages to other atoms
- To refer to an atom is in general always relative
- Relative referencing is used because:
  - it is needed in an object oriented environment !!
  - everything is an atom (model, product, machine…)
  - it is fast

In entry/exit triggers

- Current “c” & Involved “i”
- ‘**c**’ is current atom or the atom where the statement is written on
- ‘**i**’ is the involved atom (= the atom that triggered an eventhandler)

Entry/Exit Triggers

- Example: ‘**i**’ enters ‘**c**’



- ‘**c**’ refers to atom which is being entered/exited atom by other atom
- ‘**i**’ refers to entering/exiting atom

Labels and entry/exit triggers

- On entry/exit triggers, the following shortcuts may be used:

querying

```
c.labelname
i.labelname
```

updating

```
c.labelname:=value/[string]
i.labelname:=value/[string]
```

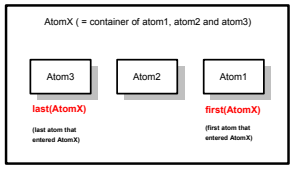
- Examples
  - `i.producttype:=[halfproduct]`
  - `c.setuptime:=30`
  - `i.producttype:=0`

Direct Atom Referencing

- Sometimes atoms can be referenced directly:
  - **`library`** = library atom
  - **`model`** = model atom
  - **`treeatom`** = currently selected atom in treeview
  - **`animatom`** = currently selected atom in 2D animation window
  - **`atombyname([e1],e2)`** = atom with name e1 in container e2
  - if tables have aliases you can use the table name direct

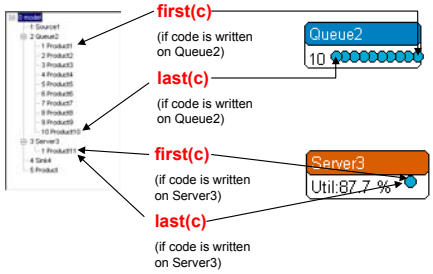
Relative Atom Referencing (1)

- All other referencing is relative: start from the atom where statement is written (=c):
  - **`first(e1)`** = first atom inside atom e1
  - **`last(e1)`** = last atom inside atom e1



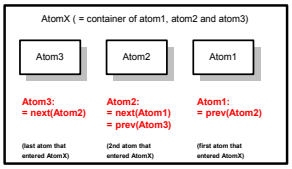
Relative Atom Referencing (2)

- Example



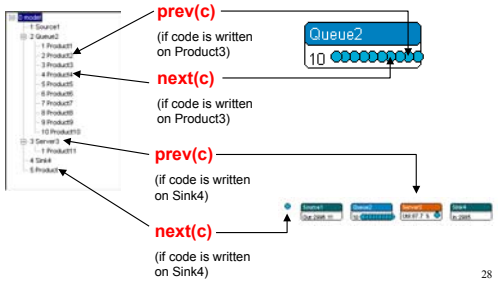
Relative Atom Referencing (3)

- **`next(e1)`** = the atom next of atom e1 in same container
- **`prev(e1)`** = the atom previous of atom e1 in same container



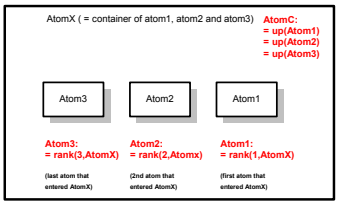
Relative Atom Referencing (4)

- Example



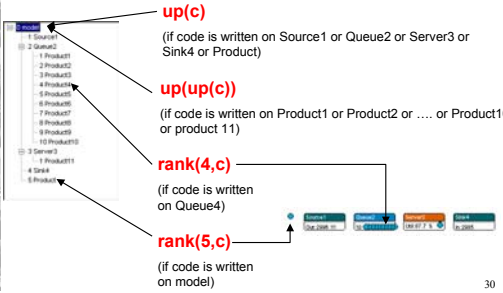
Relative Atom Referencing (5)

- **`up(e1)`** = container of atom e1
- **`rank(e1,e2)`** = atom at position e1 in queue of atom e2



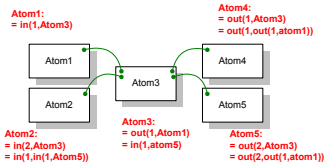
Relative Atom Referencing (6)

- Example



Relative Atom Referencing (7)

- `in(e1,e2)` = atom connected to input channel e1 of atom e2
- `out(e1,e2)` = atom connected to output channel e1 of atom e2



Atom reference (Summary)

- "`c`" will always refer to the atom you write the code on!
- "`i`" is only used to refer to the product that caused a trigger to execute.
- Do not use "`i`" on other places than triggers, for example: `first(c)` is used in Cycletime field to refer to the first product.
- Use animatom, treeatom only for debugging purposes (Interactive window) and not in code on atoms.