

4 Overview on Approaches to Multimedia Programming

4.1 History of Multimedia Programming

4.2 Squeak and Smalltalk: An Alternative Vision

4.3 Director and Lingo: Advanced Multimedia Authoring

4.4 Frameworks for Multimedia Programming

Literature:

<http://xsrv.mm.cs.sunysb.edu/364/historyofMM/historyofMM.html>

Mark Guzdial: History of Squeak

Lecture notes at <http://coweb.cc.gatech.edu/cs2340/3608>

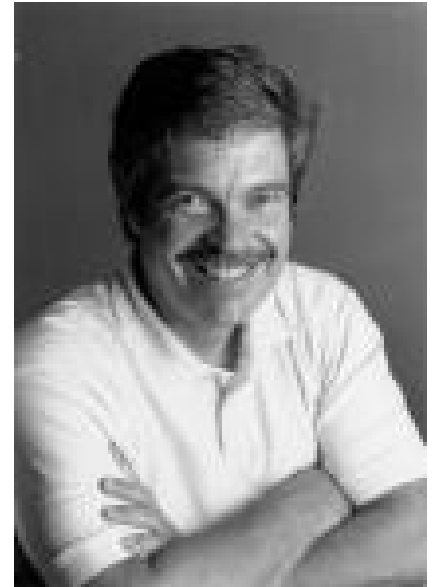
<http://minnow.cc.gatech.edu/squeak/3139>

Ivan Sutherland's Sketchpad, 1963



First object-oriented
drawing program
Master and instance
drawings
Rubber bands
Simple animations

Alan C. Kay



- U. Utah PhD student in 1966
 - Read Sketchpad, Ported Simula
- Saw “objects” as the future of computer science
- His dissertation:
Flex, an object-oriented *personal* computer
 - A *personal* computer was a radical idea then
 - How radical?

"There is no reason anyone would want a computer in their home."
(Ken Olsen, Digital Equipment Corp, **1977**)

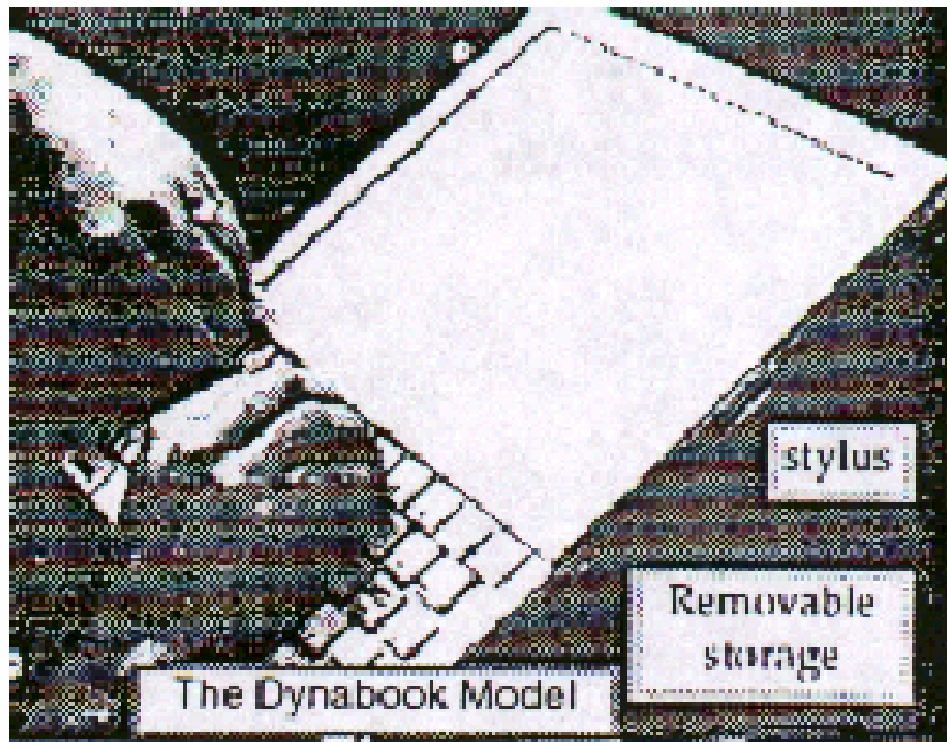
Further stations of Alan Kay's life:

- Stanford Artificial Intelligence Laboratory
- **Xerox PARC**
- Apple, Atari
- Disney Interactive
- Viewpoints Research Institute
- Hewlett-Packard

from M. Guzdial

The Dynabook Vision

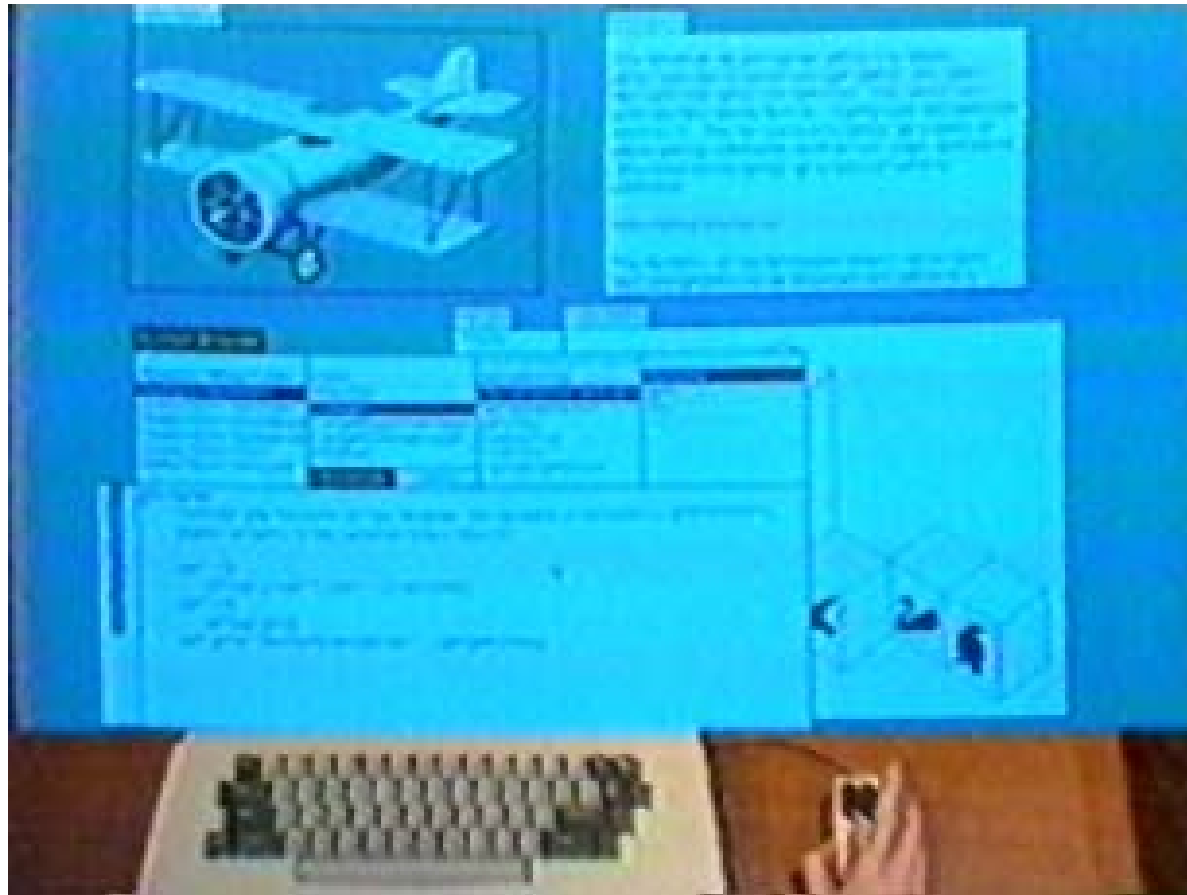
- Small, handheld, wireless(!) device – a new *medium*
- Can be used creatively by everybody, in particular children, for learning
- Xerox PARC Learning Research Group, early 70s



Tablet PC, 2004



Xerox PARC Learning Research Group: Smalltalk-72



- Object-oriented programming system
 - Mouse
 - Windows
 - Icons
 - Pop-up menus
- Uses simple object-oriented language “Smalltalk”
- Idea of user interface: Make computers easy to use for everybody
- Idea of language: make programming both more simple and more powerful (e.g. include *multimedia: sound*)

The Alto

- The machine the prototype of which impressed Steve Jobs so much that he decided to produce the Lisa/Macintosh kind of computers for the mass market (1979)
 - Graphical user interface
 - Networked via Ethernet
 - Programming language Smalltalk



Commodore Amiga



- Erscheinungsjahr: Mitte 1985 (Deutschland 1986)
- Arbeitsspeicher 256KByte Chip-Ram
- CPU: Motorola 68000, 7.16Mhz NTSC, 7.09Mhz PAL
- Grafik (u.a.):
320*200/256 (32/4096 Farben)
640*200/256 (16 Farben)
- Sound: 8 Bit 4 Kanal Stereo, 29Khz
- Massenspeicher: 1 Diskettenlaufwerk 880KByte
- Betriebssystem: Kickstart 1.0, 1.1, 1.2, 1.3 (Rom auf Diskette)
- Einführungspreis: 6000,- DM

Atari Mega ST



- 1988
- CPU Motorola 68000 8 MHz
- Ram 1 bis 4 MByte
- Grafik
 - 640 x 200 (4 Farben)
 - 320 x 200 (16 Farben)
- Tongenerator: 3 Stimmen
 - MIDI Interface
- Ca. 2000 DM
- Typische Anwendungen:
 - Spiele
 - Musik
 - Ausbildung (Schulen)

4 Overview on Approaches to Multimedia Programming

4.1 History of Multimedia Programming

4.2 Squeak and Smalltalk: An Alternative Vision

EToys: Visual Programming in Squeak

Introduction to Smalltalk

Multimedia in Squeak

4.3 Director and Lingo: Advanced Multimedia Authoring

4.4 Frameworks for Multimedia Programming

Literature:

<http://www.squeakland.org>

Back to the Future: Squeak

- Smalltalk:
 - Developed 1972
 - Commercial versions from 1980 on
- 1995: Alan Kay, Dan Ingalls, Ted Kaehler at Apple
 - Build on Open Source Software strengths
 - » Use the distributed power of Internet-based programmers
 - Available Smalltalk versions had lost many media capabilities
- Later on, the Squeak team moves to Disney
 - “Its all about media”
- Multimedia in Squeak:
 - 16 voice music synthesis
 - 3-D graphics, MIDI, Flash, sound recording
 - Network: Web, POP/SMTP, zip compression/decompress

Basics of Squeak Interaction (1)

- Squeak assumes a three-button mouse
- Menus are invoked by clicking on objects
 - clicking on surface opens “world” menus
- “Red”
 - Windows: left-button click
 - MacOS: simple click
- “Yellow”
 - Windows: middle-button click
 - MacOS: option + click
- “Blue”
 - Windows: right-button click
 - MacOS: + click



(A different colour mapping...)

Basics of Squeak Interaction (2)

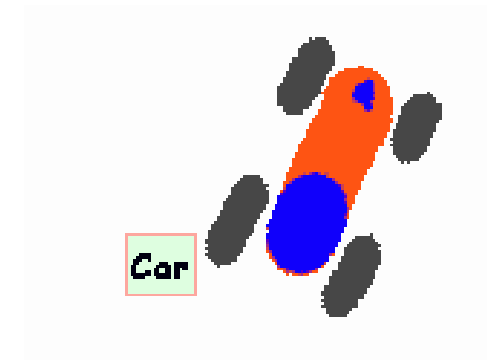
- Flaps:
 - Areas which can be opened or closed in a drawer-style
 - Often used as repositories (“parts-bins”)



- Collapsing windows:
 - A window can be collapsed or expanded

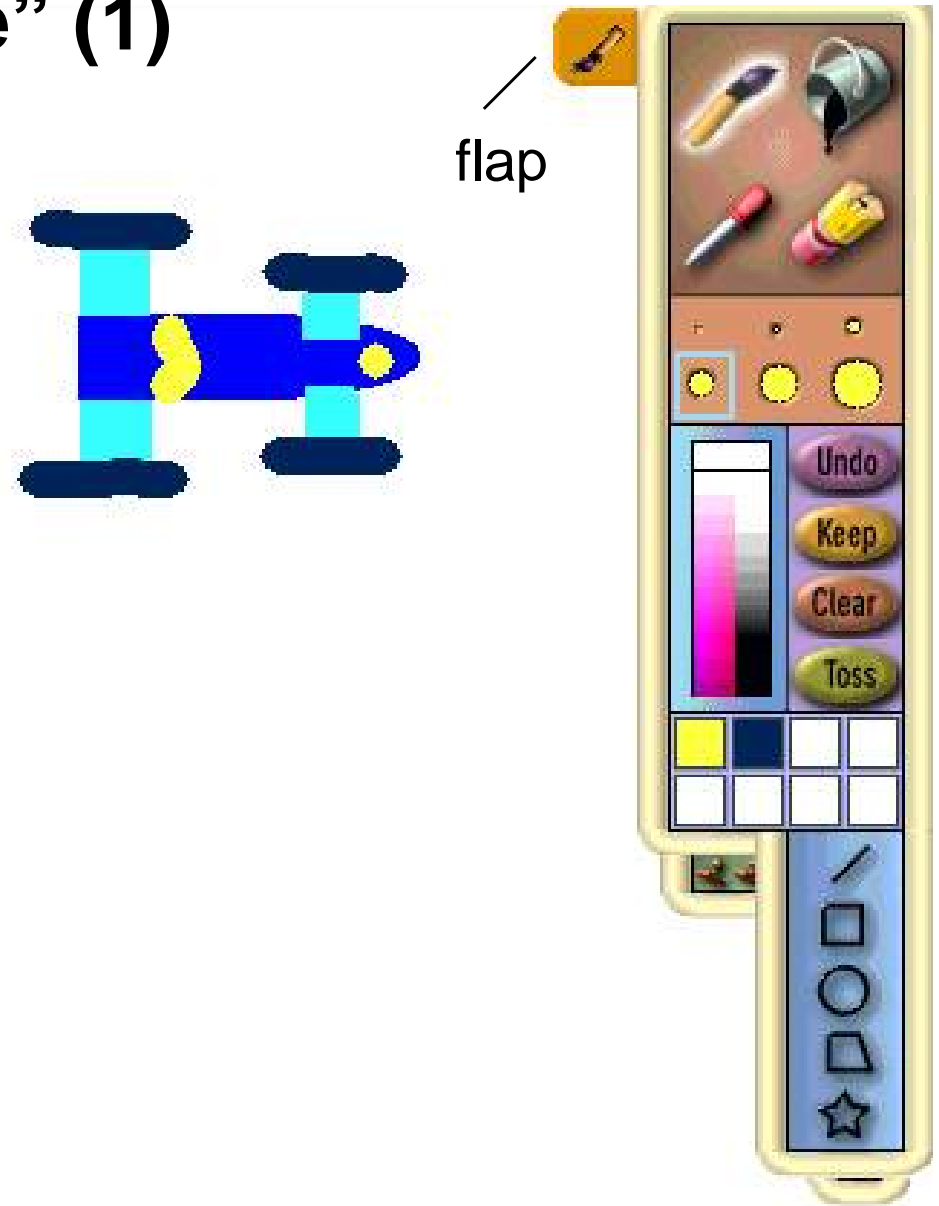


- Tiles:
 - Objects can be represented by “tiles”



Etoys: Example “Car Race” (1)

- Step 0:
Create a new empty project
 - world menu -> open...
 - > morphic project
 - enter new project by double-click
- Step 1: Draw the things with which we want to play
 - Very simplistic
bitmap-oriented painting tool
- Step 2: “Keep” the drawing
 - We get a Squeak object
 - » Free form, not square
 - Can be moved around



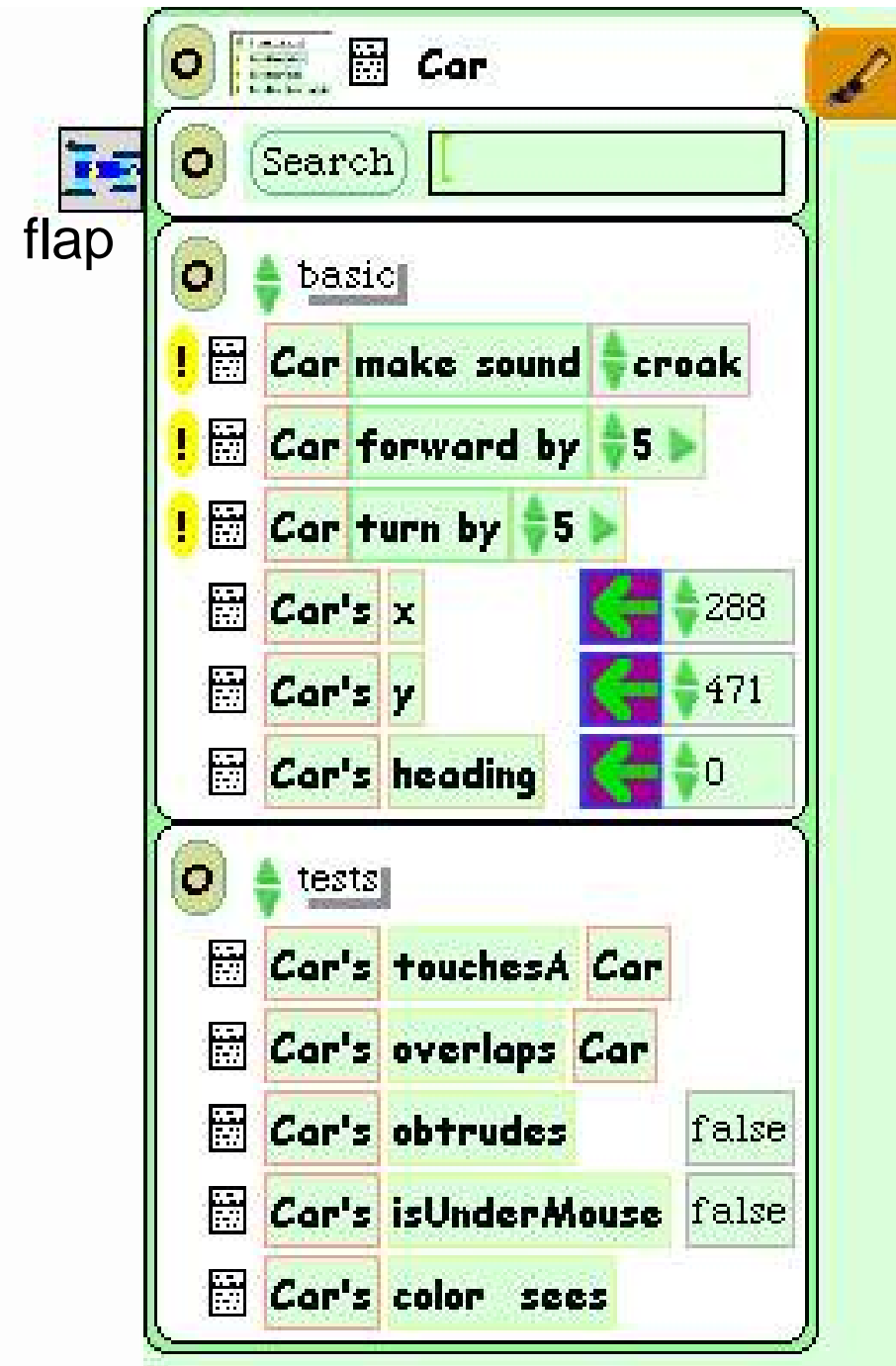
“Halo” of a Squeak Object

- The “halo” is a circular graphic menu which can be invoked on any object by a mouse click
 - “blue” click
 - special “playfield configuration” (preferences): invoked just by mouse over



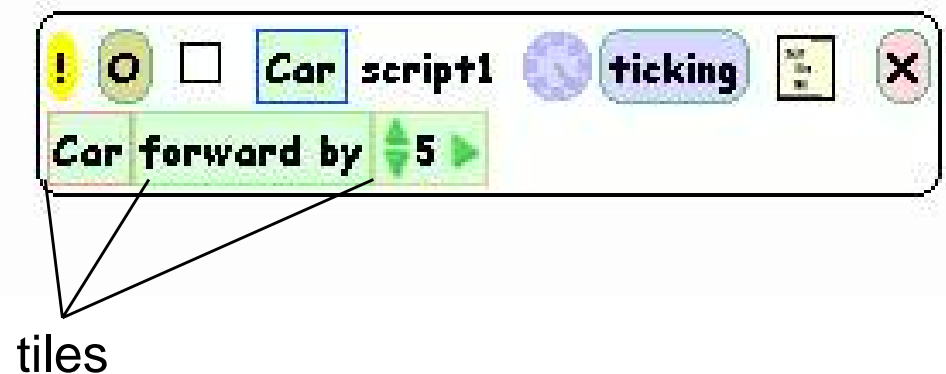
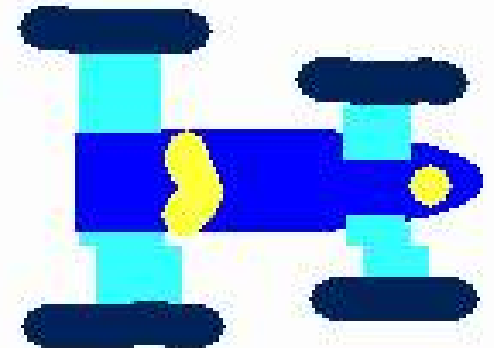
Squeak Viewers

- Step 3: Create a viewer (e.g. via the object's halo)
 - Special flap for quickly showing and hiding the viewer
 - Rename sketch in viewer e.g. to “Car”
- Shows categories of properties and commands for objects
 - Categories: Object is derived from a subclass in a complex class hierarchy
 - Viewer can show many different categories in parallel
- Commands can be immediately executed (exclamation mark button)
 - Car can be moved, turned (Note: Orientation to be set in “rotate” mode to define direction of movement)



Squeak Scripts

- *Script.*
 - simple sequence of commands
 - executed under user control or automatically through a timer (“ticking”).
- Represented by windows
 - created by drag-and-drop
 - “Tiles” represent objects and actions
- Step 4: Create a script
 - “add new script” in viewer
 - drag “empty script” onto surface
- Step 5: Add forward command
 - drag it from the Car viewer
 - adjust the parameter(s)

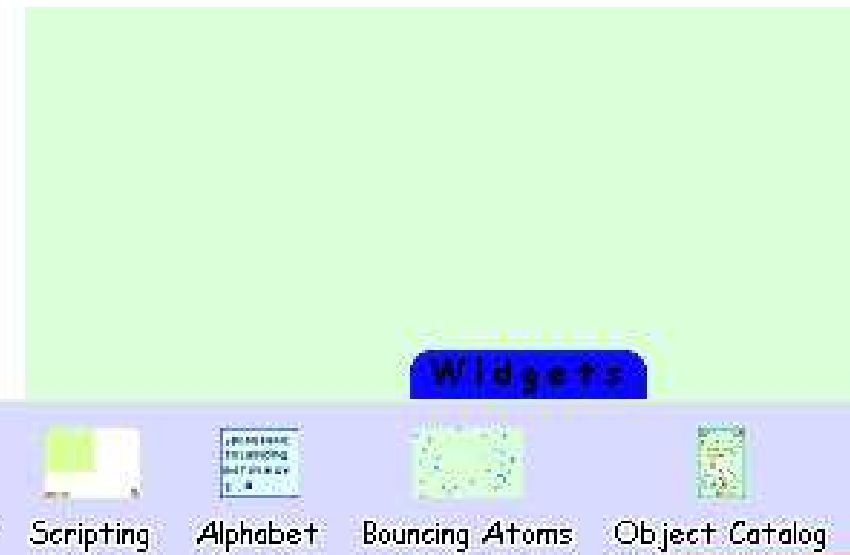


Running a Script

- Step 6: To control all scripts, use a new script control object.
 - To be found under the “Widgets” flap, like many other helpful tools
- All scripts of the project are simultaneously started and stopped through one button
 - Again just one drag operation to instantiate the object
- Example: Now car can be “driven” forward (till the border of the screen)

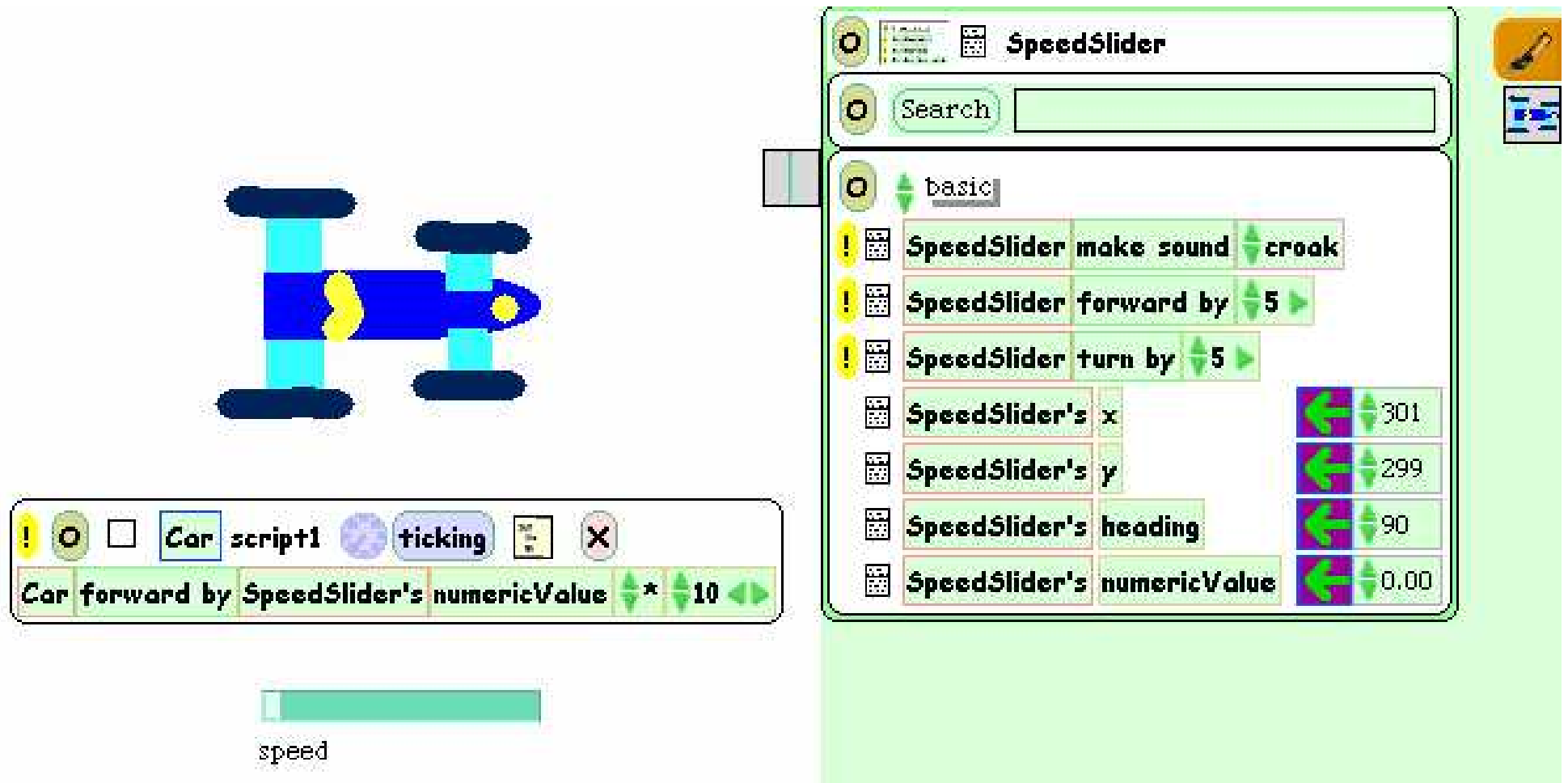


A tool that lets you see and control all the running scripts in your project



Object Interaction in Scripts

- Parameters of script commands can be computed from other objects' properties (by dragging the property onto the parameter location)
- Local adjustments can be added at the end (factor, offset etc.)

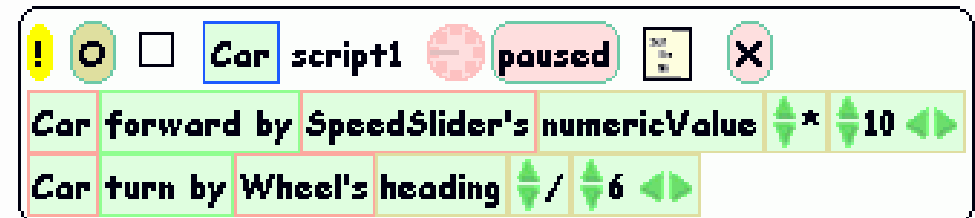
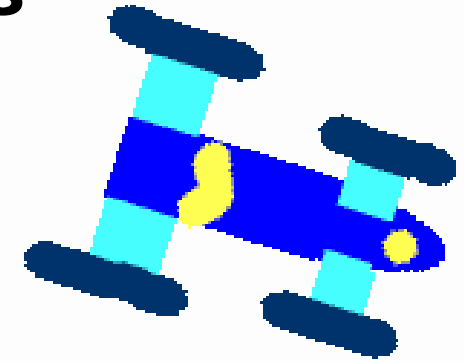


The image displays a Scratch-like environment. On the left, a blue car object is shown with a script editor. The script contains a single block: "Car forward by SpeedSlider's numericValue * 10". Below the car is a green progress bar labeled "speed". On the right, the "SpeedSlider" object's script editor is open, showing a "basic" loop with the following blocks:

- SpeedSlider make sound croak
- SpeedSlider forward by 5
- SpeedSlider turn by 5
- SpeedSlider's x: 301
- SpeedSlider's y: 299
- SpeedSlider's heading: 90
- SpeedSlider's numericValue: 0.00

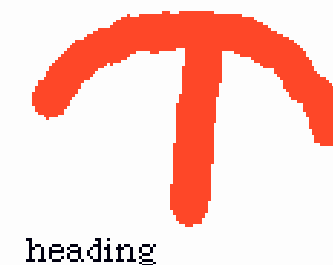
User Control through Graphical Objects

- Graphical manipulations can be used to control other objects
- Example:
 - Steering wheel graphics
 - » Drawn by hand
 - » Viewer attached
 - Rotated by user (e.g. through halo operations)
 - Heading of wheel is transferred to car
 - A “servo steering” i.e. a less sensitive transfer is recommendable



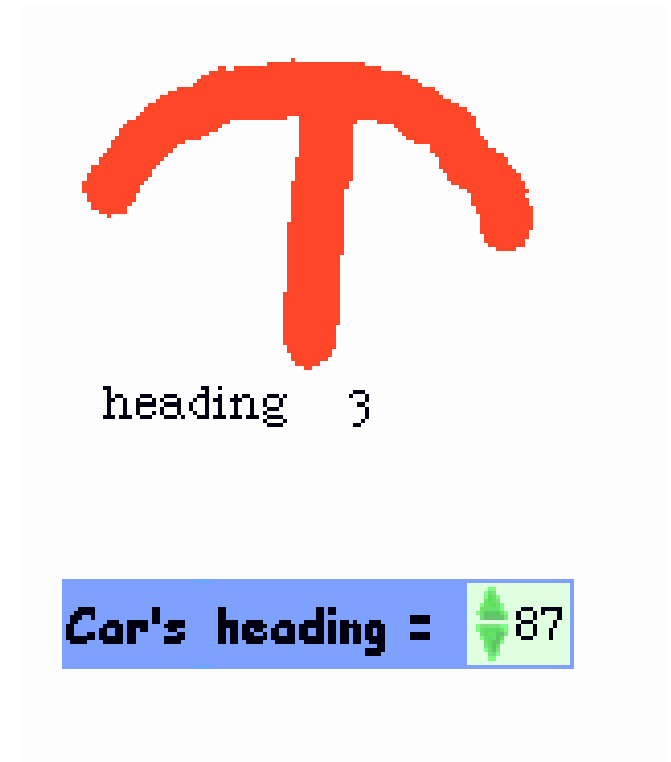
The image shows a Scratch script editor window for an object named "Car script1". The window has a "paused" status and a "help" icon. The script contains two code blocks:

```
Car forward by SpeedSlider's numericValue * 10  
Car turn by Wheel's heading / 6
```



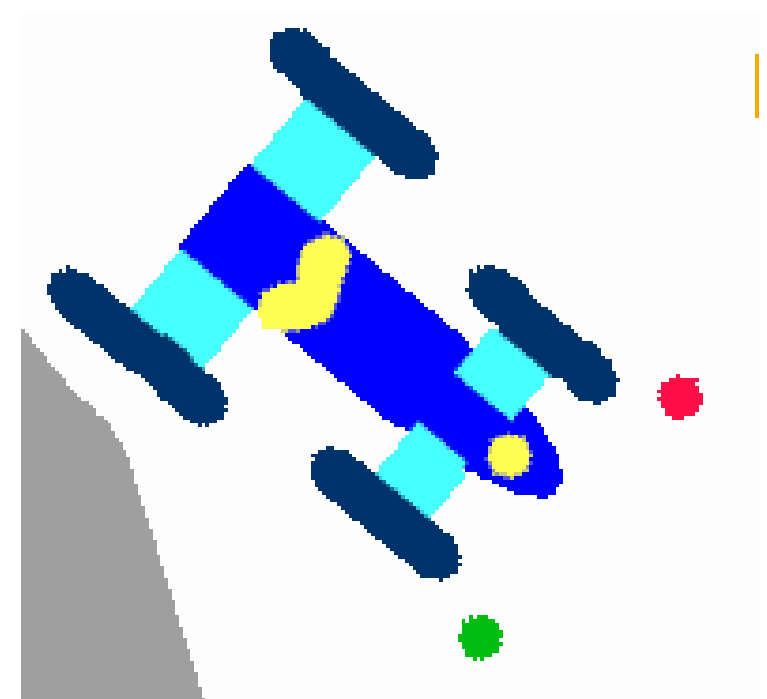
Watcher

- The values of object properties can be easily shown on the screen
 - Updated regularly and automatically
- Technically, this is an “Observer” mechanism
 - Hidden behind simple drag&drop interface
- Watcher:
 - Simple watcher (value), Detailed watcher (value plus label)
 - Can be obtained from menu left of property (in viewer)
 - Can be placed anywhere on screen



Sensors for Environment

- Squeak objects can easily observe where they are currently located
 - Through coordinates
 - Simpler: through colours
- *Sensors*:
 - Realizable as special parts of the graphics with a unique colour
 - “color x sees color y” test: Which colour is below the sensor?
- Example:
 - Grey road, car with two sensors
 - Alert lamp shall go red when one of the sensors is not on road



Example: Alert Lamp

The image shows a Scratch script editor window titled "Car script1" which is paused. The script contains the following actions and tests:

- Car forward by SpeedSlider's numericValue * 10
- Car turn by Wheel's heading / 6
- Ellipse's color ← green color
- Test: Car's red color sees grey color
 - Yes
 - No: Ellipse's color ← red color
- Test: Car's green color sees grey color
 - Yes
 - No: Ellipse's color ← red color

Labels on the right side of the image point to the following elements in the scene:

- Test tile
- Test on left sensor
- Alert lamp
- Test on right sensor
- Assignment (dragging the green-on-purple arrow right of properties)

Example: Auto-Steering

- Interaction among objects can be designed in communication loops
- Example:
 - Car automatically moves forward
 - Sensor detects border of road
 - Car automatically steers to stay on the road
- Enables complex interactive learning experiences (setting up feedback loops)

! O □ Car script1 paused

Car forward by SpeedSlider's numericValue * 10

Car turn by Wheel's heading / 6

Ellipse's color ← green color

Test Car's red color sees gray color

Yes

No Ellipse's color ← red color
Car turn by 10

Test Car's green color sees gray color

Yes

No Ellipse's color ← red color
Car turn by -10

Wheel control better removed at this stage?

4 Overview on Approaches to Multimedia Programming

4.1 History of Multimedia Programming

4.2 Squeak and Smalltalk: An Alternative Vision

EToys: Visual Programming in Squeak

Introduction to Smalltalk

Multimedia in Squeak

4.3 Director and Lingo: Advanced Multimedia Authoring

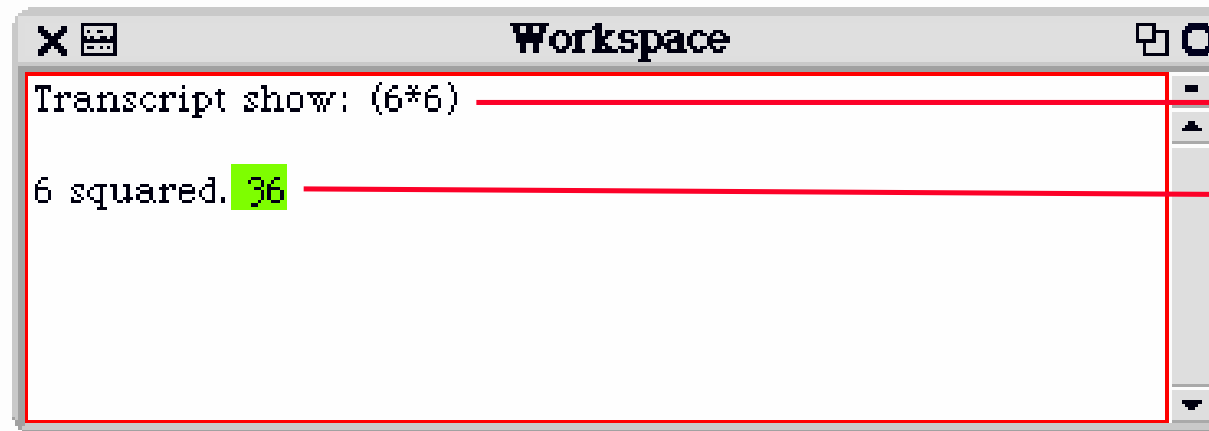
4.4 Frameworks for Multimedia Programming

Literature:

<http://www.squeak.org> (tutorials)

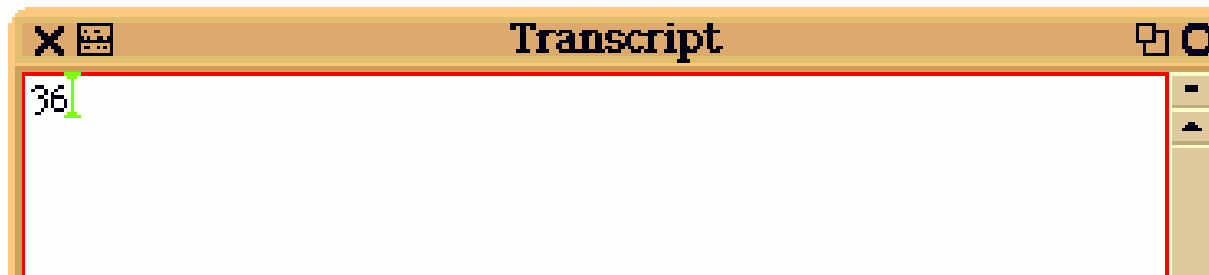
Smalltalk Programming is Open & Interactive

- Smalltalk programs are always ready for execution, even small parts of the code can be evaluated instantly
- The interpreter state is saved/loaded in an “image” file.
- The full code of the runtime system can be inspected at any time.



“do it” (ctrl-d)

“print it” (ctrl-p)



Basic Rules of Smalltalk

- Every variable is an object.
 - There are no basic types which are not objects!
 - Even classes are objects!
- Code is always triggered by sending a message to an object.
- All methods return a value.
- There are three types of messages
 - Unary, e.g. `3 negated`.
 - Binary, e.g. `a + b`.
 - Keyword, e.g. `Transcript show: a`.
 - » `show` message with parameter `a` is sent to object `Transcript`
- All code is evaluated from left to right.
 - Unary messages first, then binary, then keyword messages
 - There are no operator precedence rules.
- Assignment evaluates right hand side and assigns the result to left hand side.

Smalltalk Blocks

- `a := [2 + 3].`
`a value.`

Result: 5

Assignment
either by
typing “:=“ or
by typing “_”

- `c := [:a :b | a + b].`
`c value: 5 value: 7.`

Result: 12
(a multiple-part message)

- `x := 3.`
`y := 5.`
`(x = y)`

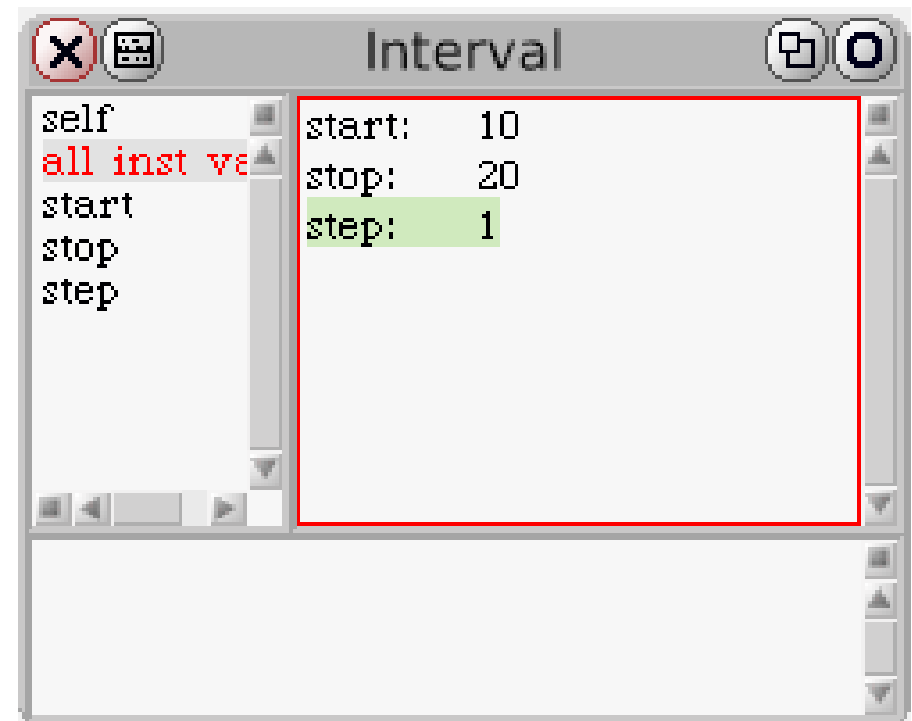
`ifTrue: [Transcript show: 'equal']`

`ifFalse: [Transcript show: 'not equal'].`

Control flow realized by message
passing mechanism

Interval Objects and Loops

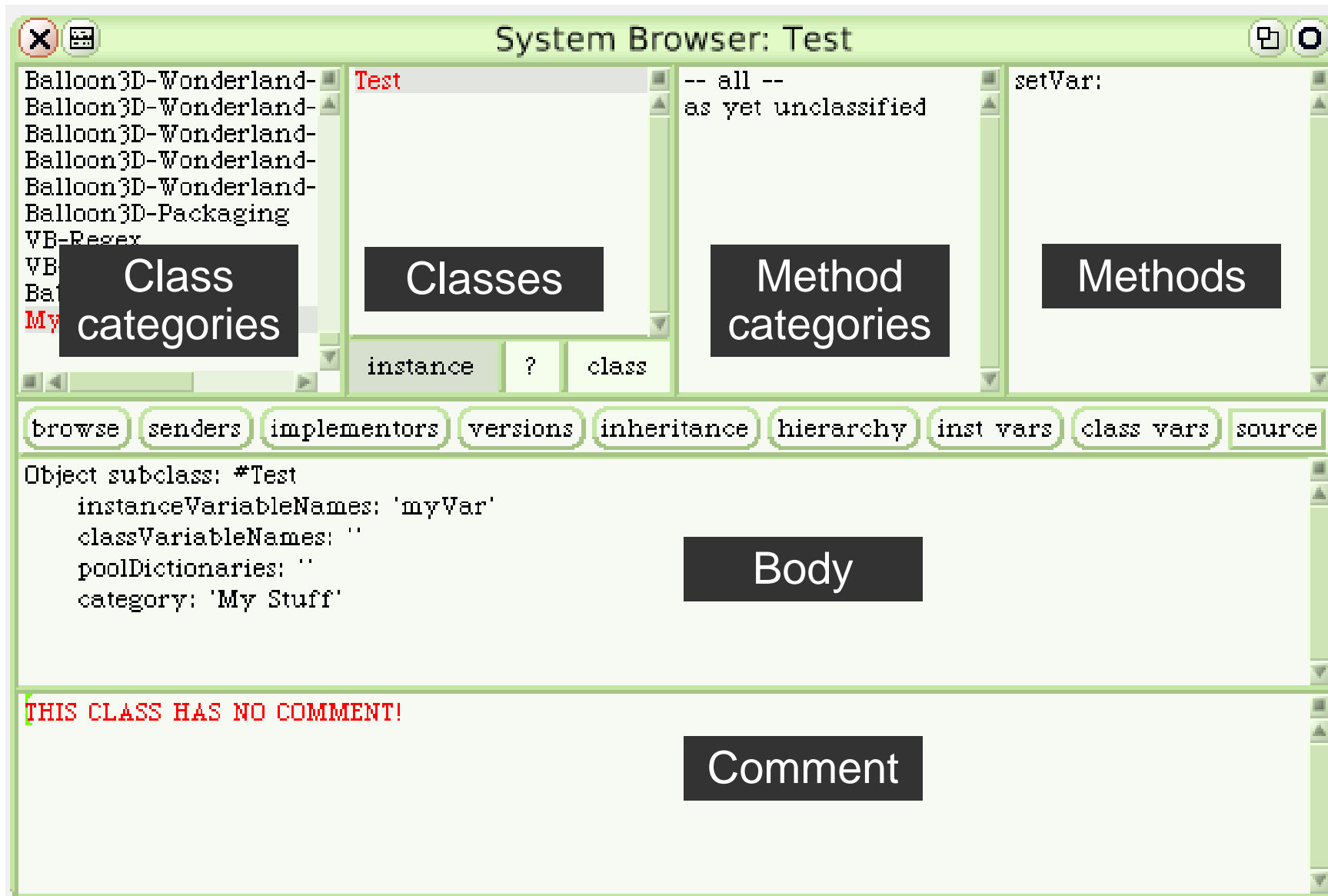
- An Interval object:
 `a := 10 to: 20.`
 `a inspect.`
- Looping through the interval:
 `a do: [:i | Transcript show: i; cr].`



Advanced Language Constructs in Squeak

- Infinite number precision
 - `1000 factorial / 999 factorial.` `1000`
 - `(1/3) + (2/3).` `1`
 - `Float infinity + 1.` `Infinity`
 - `Float infinity / Float infinity.` `NaN`
- Lazy evaluation
- High level iterators
 - `a := #(1 2 3).`
 - `a collect: [:x | x*2].` `#(2 4 6)`
 - `a reject: [:x | x odd].` `#(2)`

Browser Window



BankAccount Example

- Constructed interactively
 - Create new class template
 - Fill in instance variable (balance)
 - Fill in methods
 - » initialize
 - » deposit
 - » withdraw
- At any point in time, creation of objects and inspection is possible
- (Credits for the example: John Maloney)

Defining Classes: BankAccount

```
Object subclass: #BankAccount
  instanceVariableNames: 'balance'

balance
  ^ balance.
initialize
  balance := 0.
deposit: amount
  balance := balance + amount.
withdraw: amount
  (amount > balance)
    ifTrue: [^ self inform: 'No more money!'].
  balance := balance - amount.
```


BankAccount with History

- Extend class with history variable
 - Initialize with empty ordered collection

```
history := OrderedCollection new.
```

- Update history

```
balance: newBalance
```

```
    balance := newBalance.
```

```
    history addLast: newBalance.
```

```
deposit: amount
```

```
    self balance: (balance + amount).
```

```
withdraw: amount
```

```
    (amount > balance)
```

```
        ifTrue: [^self inform: 'No more money!'].
```

```
    self balance: (balance - amount).
```

Graphical Object (Morph) for BankAccount

```
historyMorph
```

```
  "displays account history as barchart"
```

```
  | bars m |
```

```
  bars := history collect:
```

```
    [:v | Morph new extent: 30@v].
```

```
  m := AlignmentMorph newRow
```

```
    hResizing: #shrinkWrap;
```

```
    vResizing: #shrinkWrap;
```

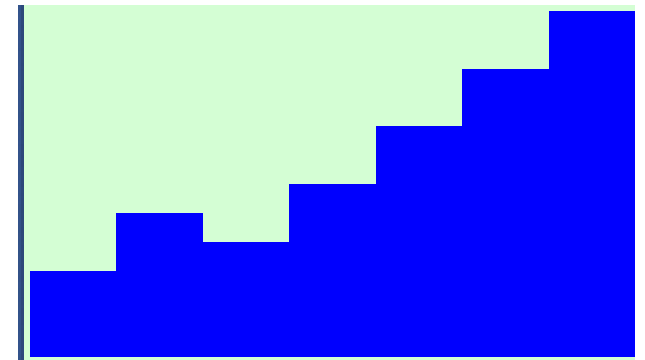
```
    cellPositioning: #bottomRight.
```

```
  m addAllMorpha: bars.
```

```
  ^m.
```

Make visible by:

```
acc historyMorph openInWorld.
```



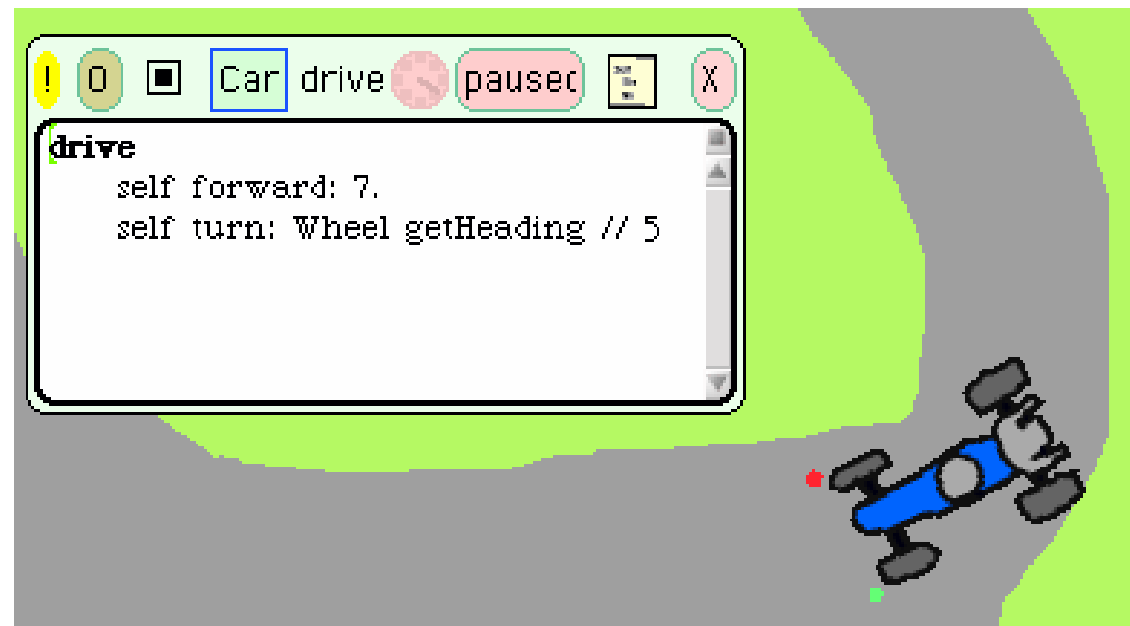
Event Handling in Morphs

```
Morph subclass: #TestMorph
  category: 'My Stuff'
handlesMouseDown: evt
  ^ true
mouseDown: evt
  self position: self position + (10 @ 0).

TestMorph new openInWorld.
```

EToys and Smalltalk

- Squeak contains a full Smalltalk development system
- EToy scripts can be switched between iconic or textual representation
- EToy scripts are found in the browser hierarchy
- EToy scripts are just shortcuts in writing Smalltalk



4 Overview on Approaches to Multimedia Programming

4.1 History of Multimedia Programming

4.2 Squeak and Smalltalk: An Alternative Vision

Squeak

EToys: Visual Programming in Squeak

Introduction to Smalltalk

Multimedia in Squeak

4.3 Director and Lingo: Advanced Multimedia Authoring

4.4 Frameworks for Multimedia Programming

Literature:

<http://www.squeak.org>

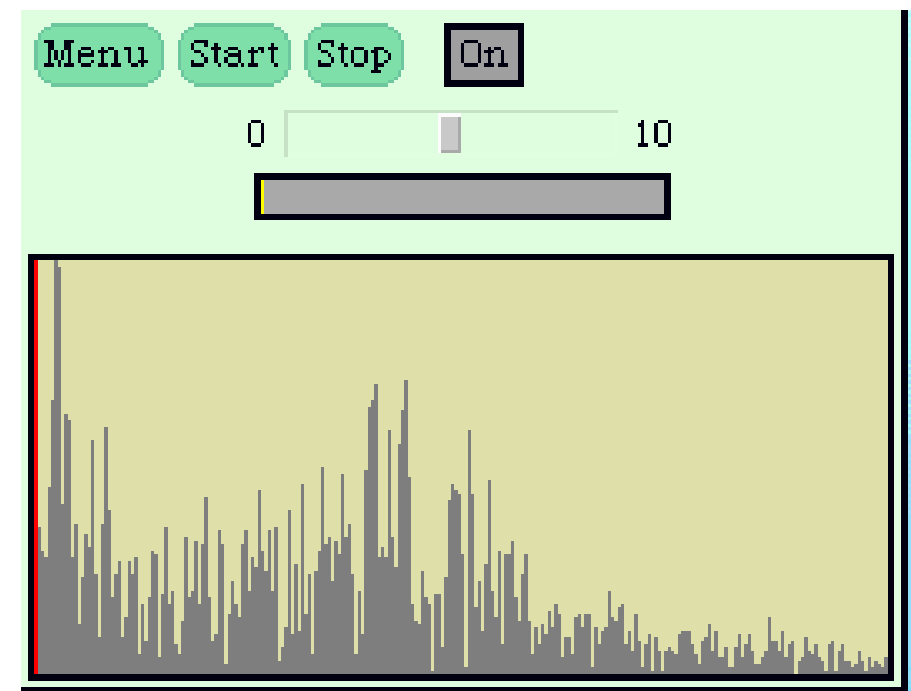
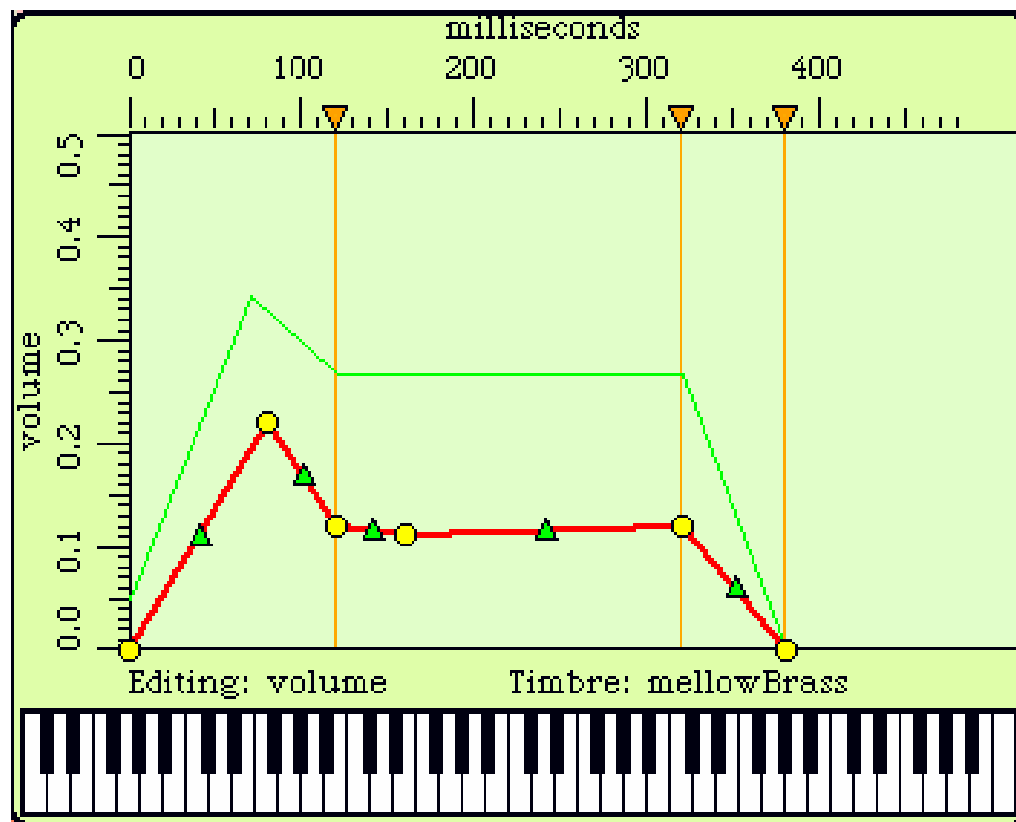
Wonderland: 3D Worlds in Squeak

- 3D objects can be moved around in intuitively simple manner
 - Prefabricated models
 - Simple self-drawn sketches (“Pooh drawings”)
- 3D objects are EToys.
- 3D objects can be manipulated with Smalltalk programs.



Squeak as a Multimedia Experimentation Platform

- Example: Sound in Squeak



Example: Playing Musical Notes in Smalltalk

```
instr := AbstractSound soundNamed: 'oboe1'.
note1 := instr soundForPitch: #c4 dur: 0.5 loudness: 0.4.
note2 := instr soundForPitch: #ef4 dur: 0.5 loudness: 0.4.
note3 := instr soundForPitch: #g4 dur: 0.5 loudness: 0.4.
(note1, note2, note3) play.
(note1 + note2 + note3) play.
```

```
song := AbstractSound noteSequenceOn: instr from: #(
    (c4 0.35 400)
    (c4 0.15 400)
    (d4 0.5 400)
    (c4 0.5 400)
    (f4 0.5 400)
    (e4 1.0 400)).
song play.
```