4 Fundamental Issues in Multimedia Programming

4.1 Multimedia Programming in Context

Looking Back: Central Issues & Alternative Approaches

The Purpose of All This?

4.2 History of Multimedia Programming

4.3 A Radically Alternative Approach: Squeak

4.4 Trends and Visions

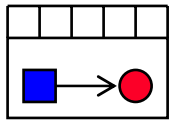
Looking Back

- Example technology Flash & ActionScript
  - Extremely powerful for creating pseudo-realistic 2D animations
  - Smooth integration of various media into one consistent user experience
    - » Information retrieved and computed from various sources
    - » Images, still and in motion
    - » Sound, including spatial sound effects
  - Much more efficient than plain programming
  - Still painful coding for complex tasks

- Managing multimedia projects
  - Much more activities than in traditional programming-only projects
    - » Production of creative idea, storyboard, graphics, sound
  - Requirements often become clear in parallel to evolution of the system
    - » Employ iterative/evolutionary process models (e.g. XP)

Key Issue: Code vs. Pictures

Graphically Oriented Authoring:
- Drawing pictures
- Using graphical metaphors (e.g. timeline)

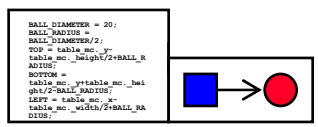


Text-Oriented Programming:
- Writing code
- Using general-purpose or specialized language

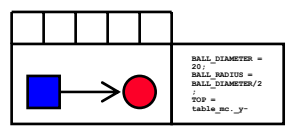
How to mix code and pictures? Three types of programs:

Type C: Code only, no pictures, no metaphors:
– Suitable only for logic-intensive programs with very simple graphics

Type CP: Large share of code and pictures, limited use of metaphors:
– Suitable for most highly interactive programs like games
– Timeline was used sparsely in game examples



Type P: Small share of code, large share of pictures, use of metaphors:
– Suitable for mostly fixed interaction like product presentation
– Timeline main tool for defining control flow

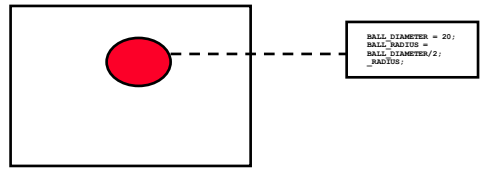


Programming Language Issues

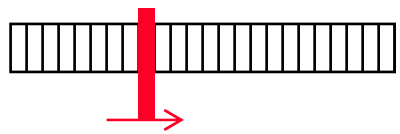
- General-purpose language (like Java) or specialized language (like ActionScript) ?
- General-purpose language:
  - Pure text languages like Java, C++ etc:
    - » Difficult to integrate with graphical authoring tool
    - » Can be complemented with multimedia frameworks (e.g. Java Media Framework)
  - Highly interactive language built for graphical development environment
    - » See below (Example Smalltalk/Squeak)
- Specialized multimedia language:
  - Typically scripting language with immediate execution (interpreted)
  - Additional training effort for developers
  - Additional development effort for tool/language designers
- Compromise taken e.g. by Macromedia with ActionScript:
  - Specialized language which is mostly identical to a standard language (ECMAScript)

Authoring Tool Paradigms

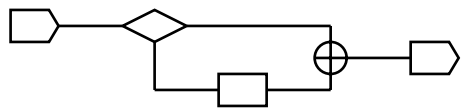
Screen-based authoring
– Examples: Apple HyperCard, Asymetrix ToolBook
– Time dimension only indirectly described



Timeline-based authoring
– Example: Macromedia Director
– Time dimension main principle of design



Flowchart-based authoring
– Example: Macromedia Authorware
– Media objects as nodes in a flow graph



4 Fundamental Issues in Multimedia Programming

4.1 Multimedia Programming in Context

- Looking Back: Central Issues
- Alternative Approaches to Multimedia Programming
- The Purpose of All This?

4.2 History of Multimedia Programming

4.3 A Radically Alternative Approach: Squeak

4.4 Trends and Visions

Literature:

N. Postman: Amusing Ourselves to Death – Public Discourse in the Age of Show Business, Viking Press, 1986

K. Rose: Amusing Ourselves to Life, *ACM Computers in Entertainment*, Vol. 2, No. 2, January 2004

Why Multimedia Programming?

- What, in the first place, was the motivation for all this?
  - Creating an immersive experience for users of computer systems
    - » Not a real answer, but a circumscription
  - Giving people fun in using computer applications?
- Why should we want to give people fun in using computers?
  - Destructive answer: To attract customers to specific products
    - » Neil Postman's criticism: Culture replaced by shallow attractions
  - Constructive answer: To improve efficiency of working
    - » Justifies only limited amount of multimedia interaction
  - Neutral answer: Fun is not the central aspect, multimedia is just the *natural way* of communicating
  - Humanistic answer: To empower those people to use computers, which do not want/can use the un-natural forms of interaction of most man-machine interfaces
    - » E.g. children!
- Where does the idea of multimedia programs come from?

4 Fundamental Issues in Multimedia Programming

4.1 Multimedia Programming in Context

4.2 History of Multimedia Programming

4.3 A Radically Alternative Approach: Squeak

4.4 Trends and Visions

Literature:

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

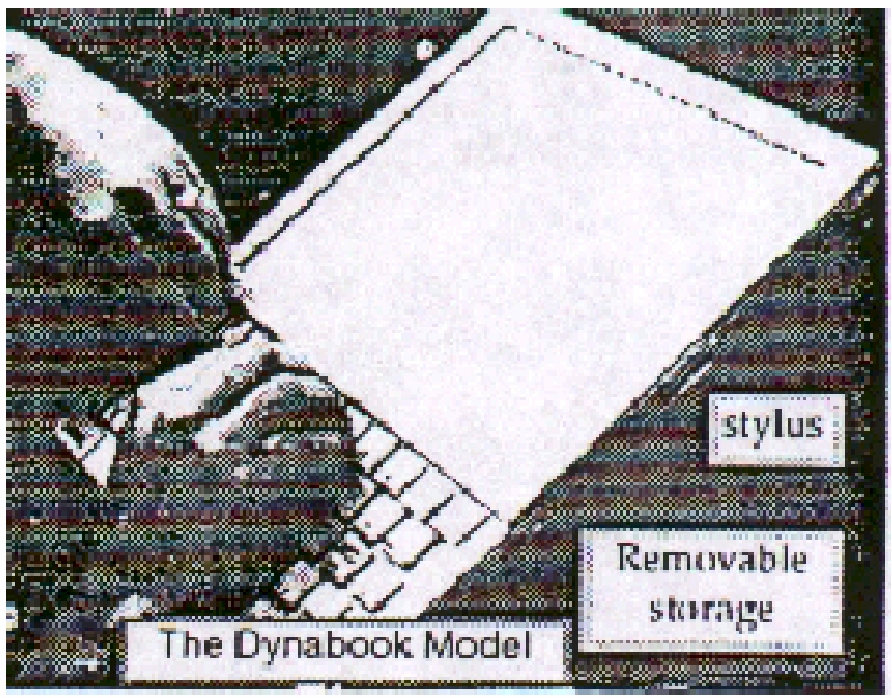
from M. Guzdial

The FLEX Machine

- “A personal computer for children of all ages”
- Includes pointing device

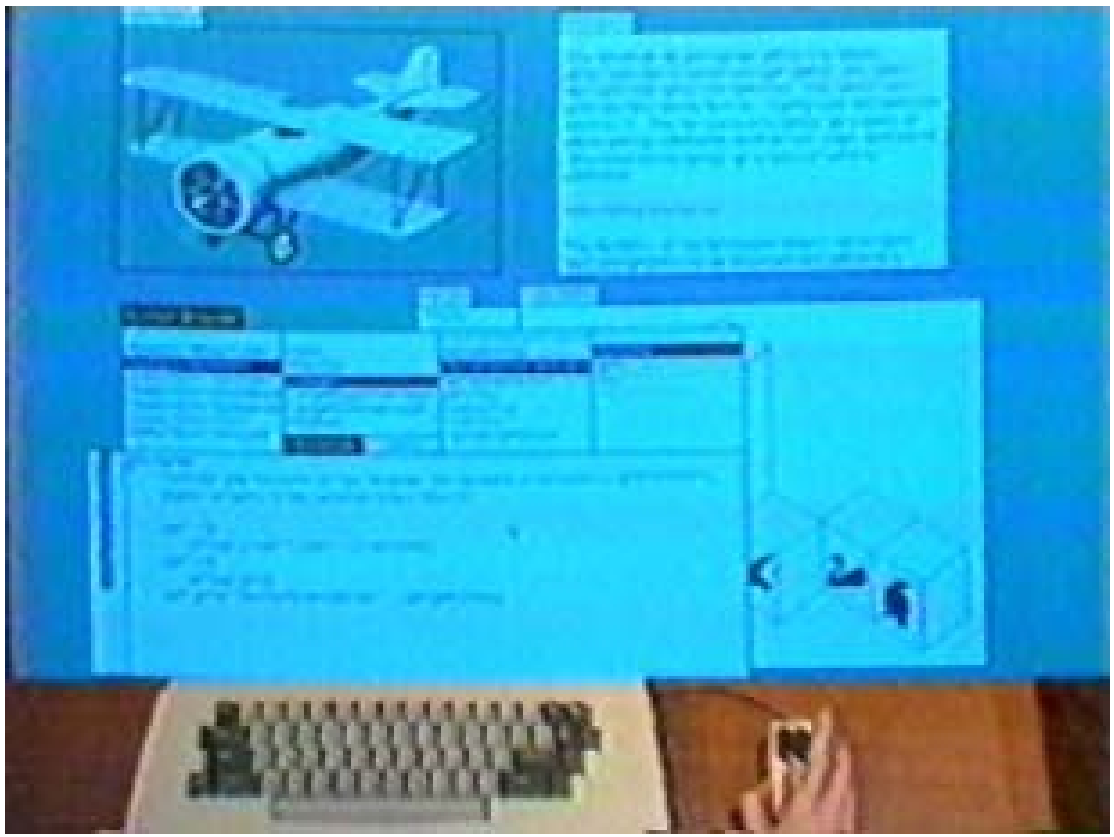
The Dynabook Vision

- 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

Smalltalk and Learning

- Original intention of Smalltalk:
  - Make the programming system so intuitive that children can use it
  - In fact a first multimedia authoring system
  - Children projects at Xerox PARC
- What happened to Smalltalk?
  - Becomes commercially targeted programming language
  - ParcPlace
  - Smalltalk-80
- Smalltalk is still the environment in which most programming inventions appear:
  - Design patterns
  - Extreme programming

(We will have a look at Smalltalk syntax and environment later on.)

Back to the Future: Squeak

- 1995: Alan Kay, Dan Ingalls, Ted Kaehler at Apple
  - Still want "A development environment in which to build educational software that could be used—and even programmed—by non-technical people and by children"
  - 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

4 Fundamental Issues in Multimedia Programming

4.1 Multimedia Programming in Context

4.2 History of Multimedia Programming

4.3 A Radically Alternative Approach: Squeak

EToys

4.4 Trends and Visions

Literature:

www.squeakland.org

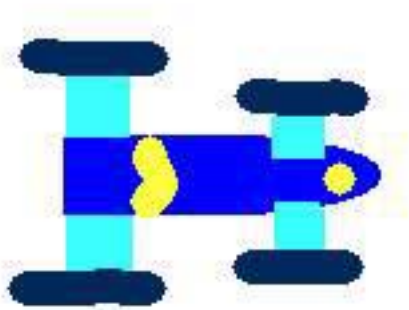
Electronic Toys: EToys

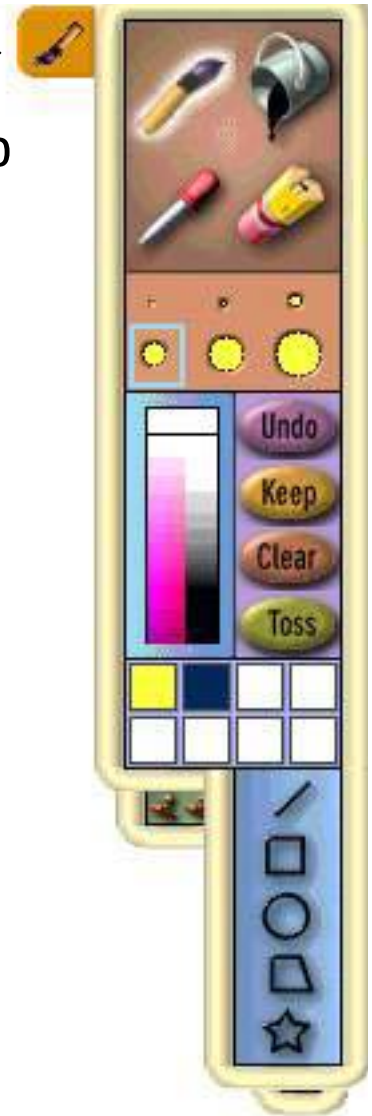
Alan Kay, EToys and Simstories in Squeak

- “EToys are computer environments that help people learn ideas by building and playing around with them. They help an “omniuser” - usually a child - create a satisfying and enjoyable computer model of the idea and give hints for how the idea can be expanded. SimStories are longer versions of Etoys that - like essays - string several ideas together to help the learner produce a deeper and more concerted project. A good motto for Etoys and SimStories is: *We make not just to have but to know.* Another motto that applies here is: *Doing with images makes symbols.* That is, the progression of learning moved from kinaesthetic interactions with dynamic images to a symbolic expression of the idea.”

Etoys: Example “Car Race” (1)

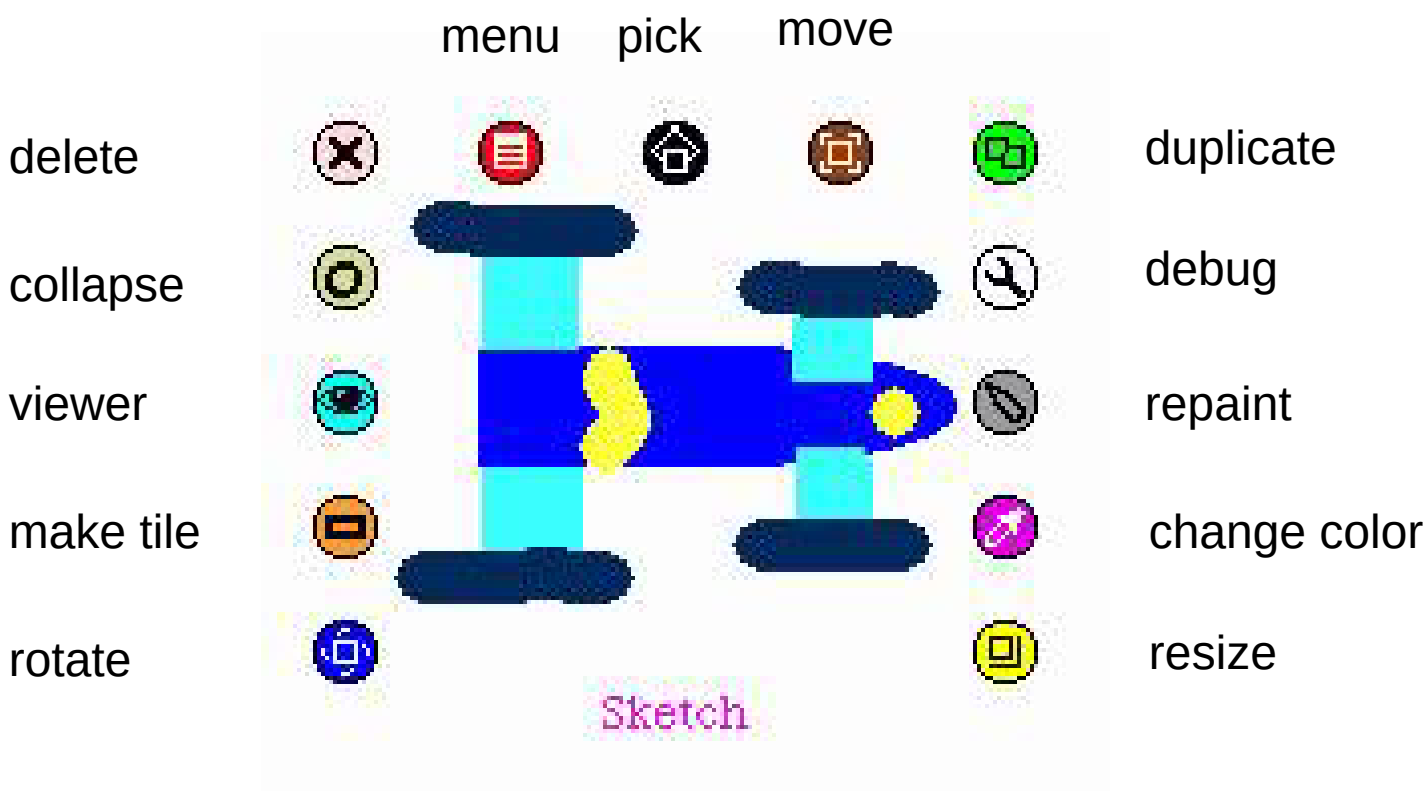
- Step 0:
  Create a new empty project
- Step 1: Draw the things with which we want to play
  - Very simplistic bitmap-oriented painting tool
- Step 2: “Keep” the drawing
  - We get a Squeak object
    - » Free form, not square
  - Can be moved around





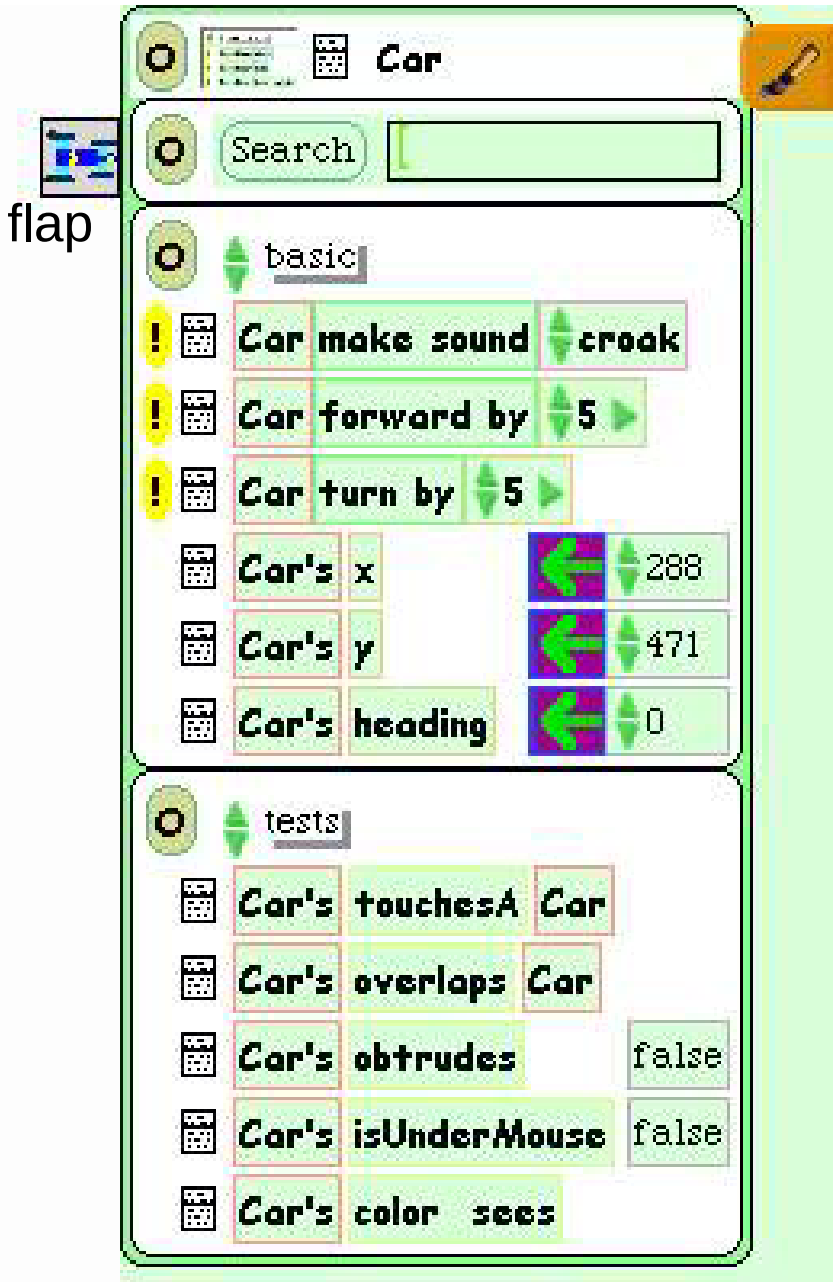
“Halo” of a Squeak Object

- The “halo” is a circular graphic menu which can be invoked on any object by a mouse click (depending on playfield configuration often just by mouse over, always with Crtl-click).



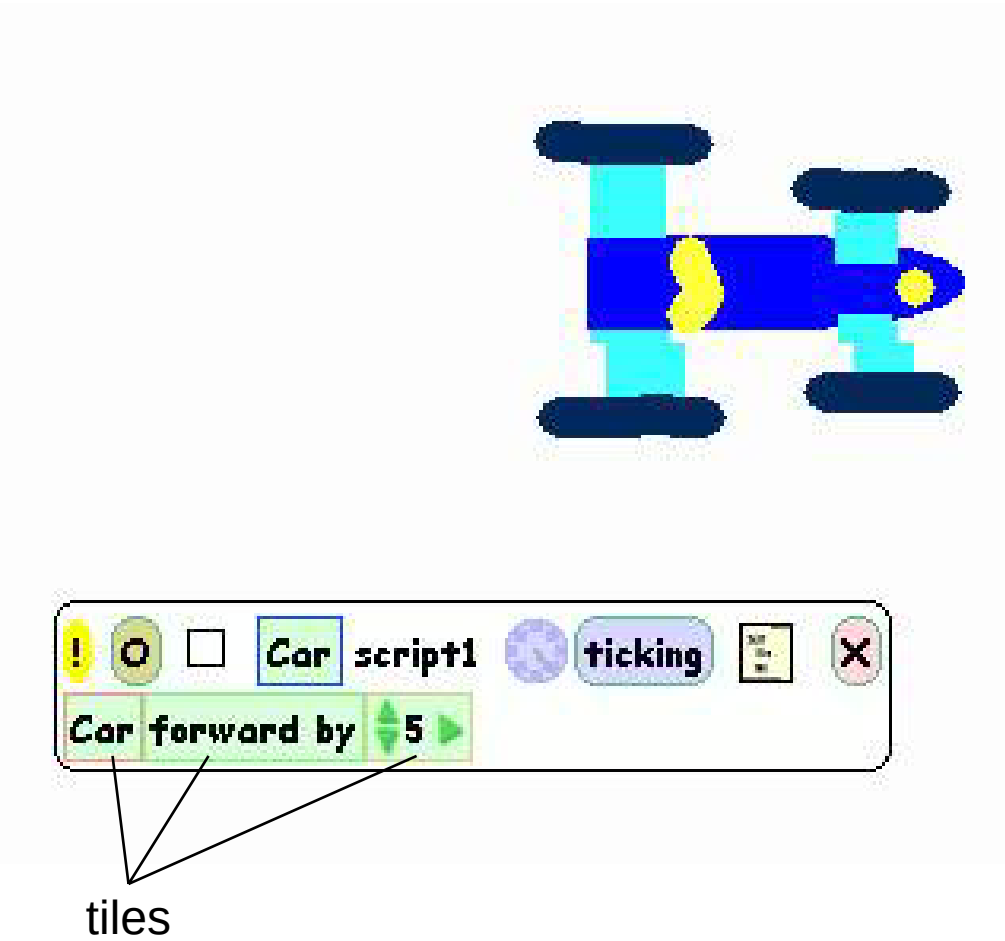
Squeak Viewers

- Step 3: Create a viewer (e.g. via the object's halo)
- A viewer shows categories of properties and commands for objects
  - Many different categories
  - Origin: Object is derived from a subclass in a complex class hierarchy
- Commands can be immediately executed (exclamation mark button)
- A viewer always creates it own flap for quickly showing and hiding the viewer
- A viewer can show many different categories in parallel



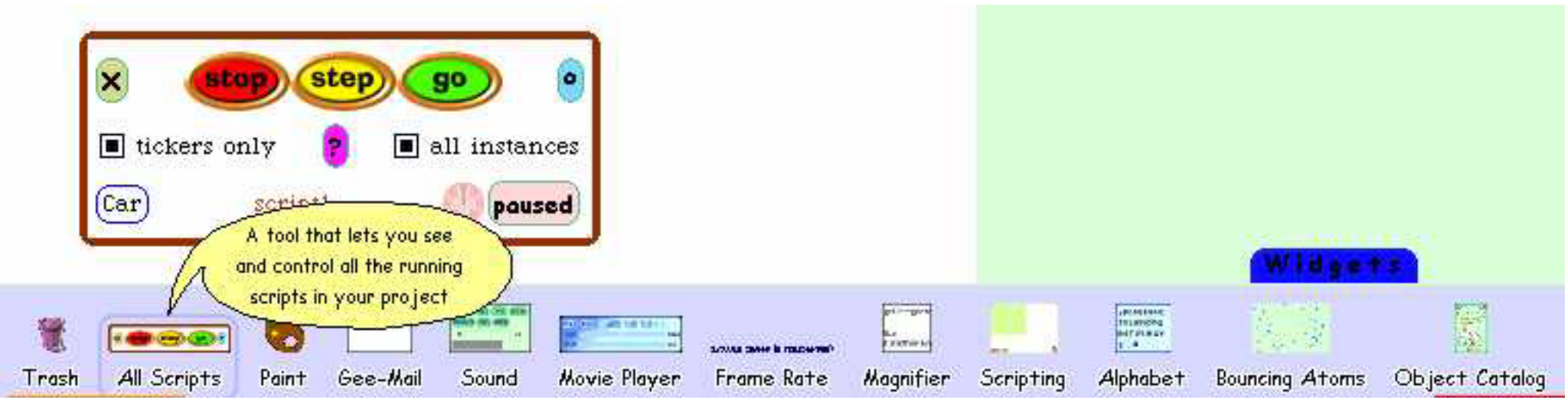
Squeak Scripts

- A *script* is a simple sequence of commands which can be executed under user control or automatically through a timer (“ticking”).
- Scripts are represented by graphical windows, segmented into tiles, and are created in a full drag-and-drop style
- Step 4: Create a script by dragging the “empty script” onto surface
- Step 5: Add the forward command by dragging it from the Car viewer
- Adjust the parameter(s)



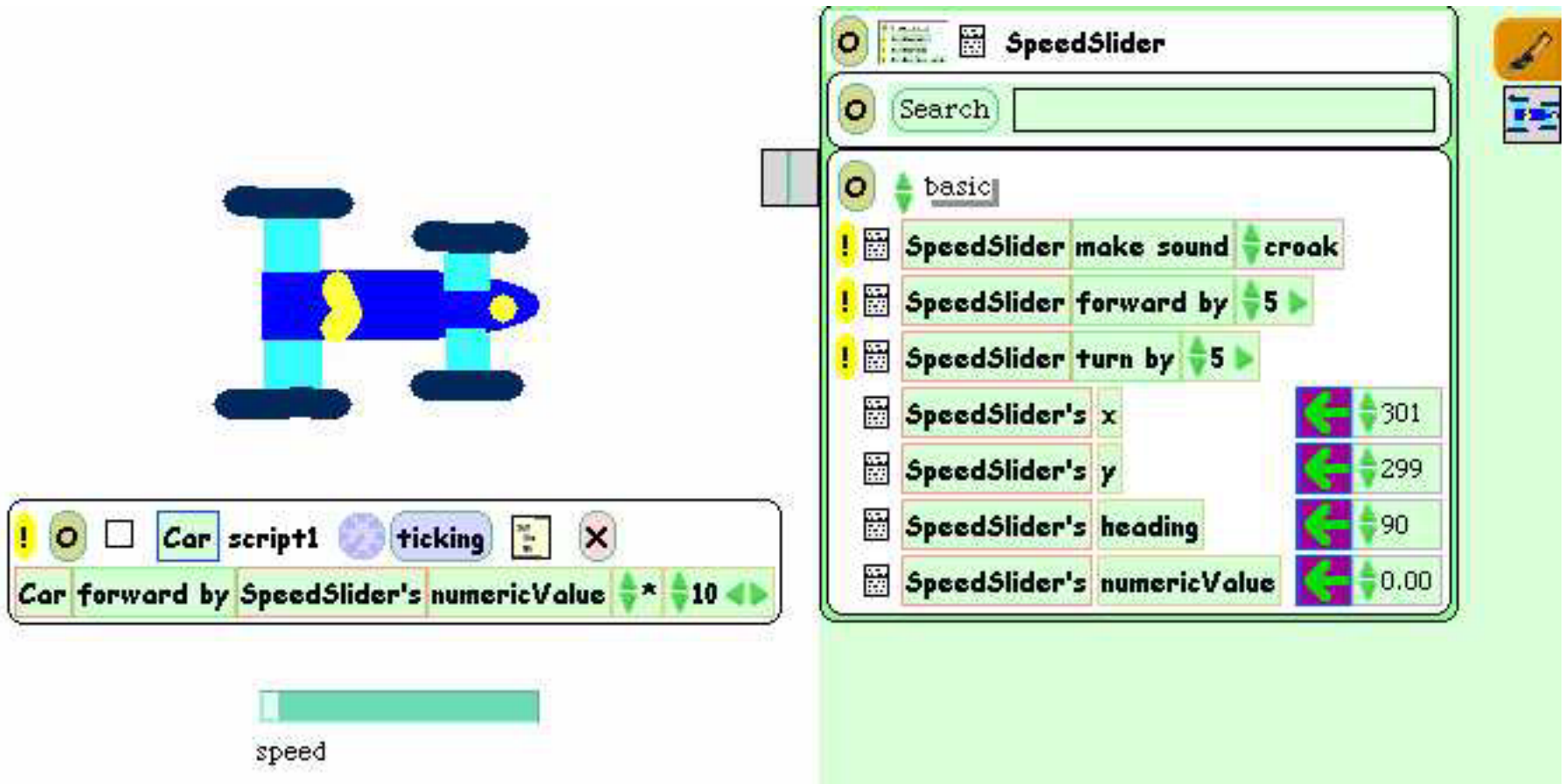
Running a Script

- 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)



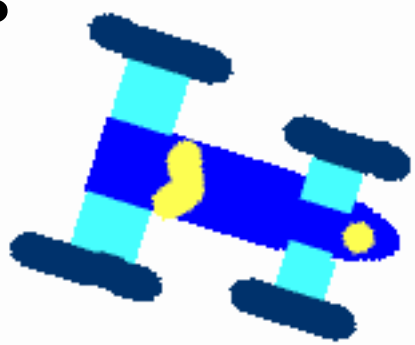
Object Interaction in Scripts

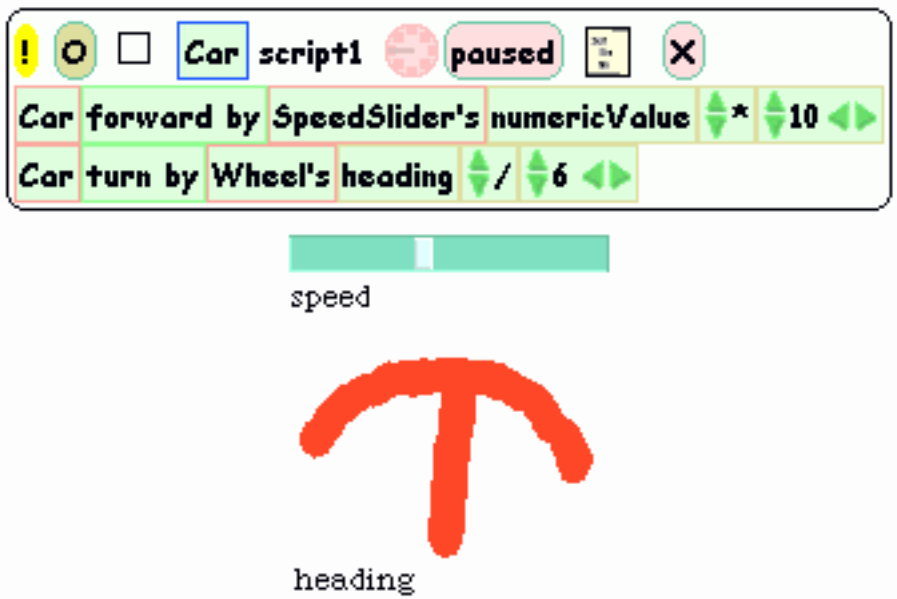
- Parameters of script commands can be computed from other objects' properties (by dragging the property onto the parameter location)
- Local adjustments can be added at the end (factor, offset etc.)



User Control through Graphical Objects

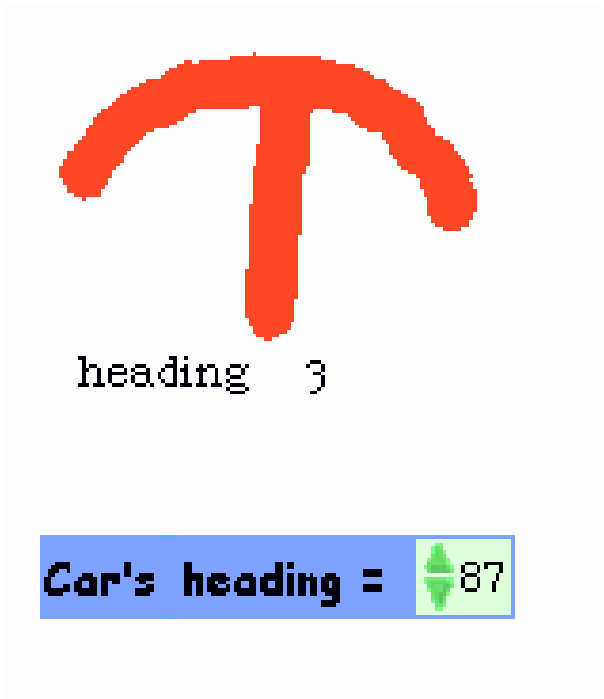
- Graphical manipulations can be used to control other objects
- Example:
  - Steering wheel graphics
    - » Drawn by hand
    - » Viewer attached
  - Rotated by user (e.g. through halo operations)
  - Heading of wheel is transferred to car
  - A “servo steering” i.e. a less sensitive transfer is recommendable





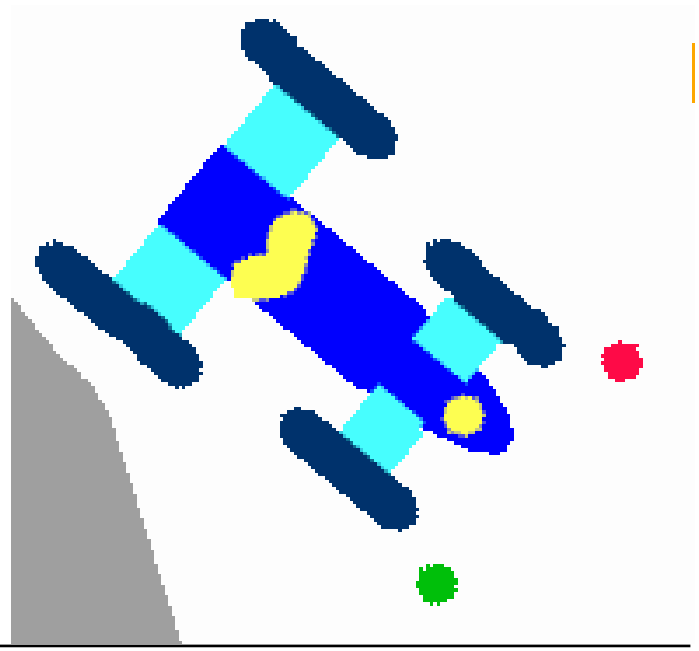
Watcher

- 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

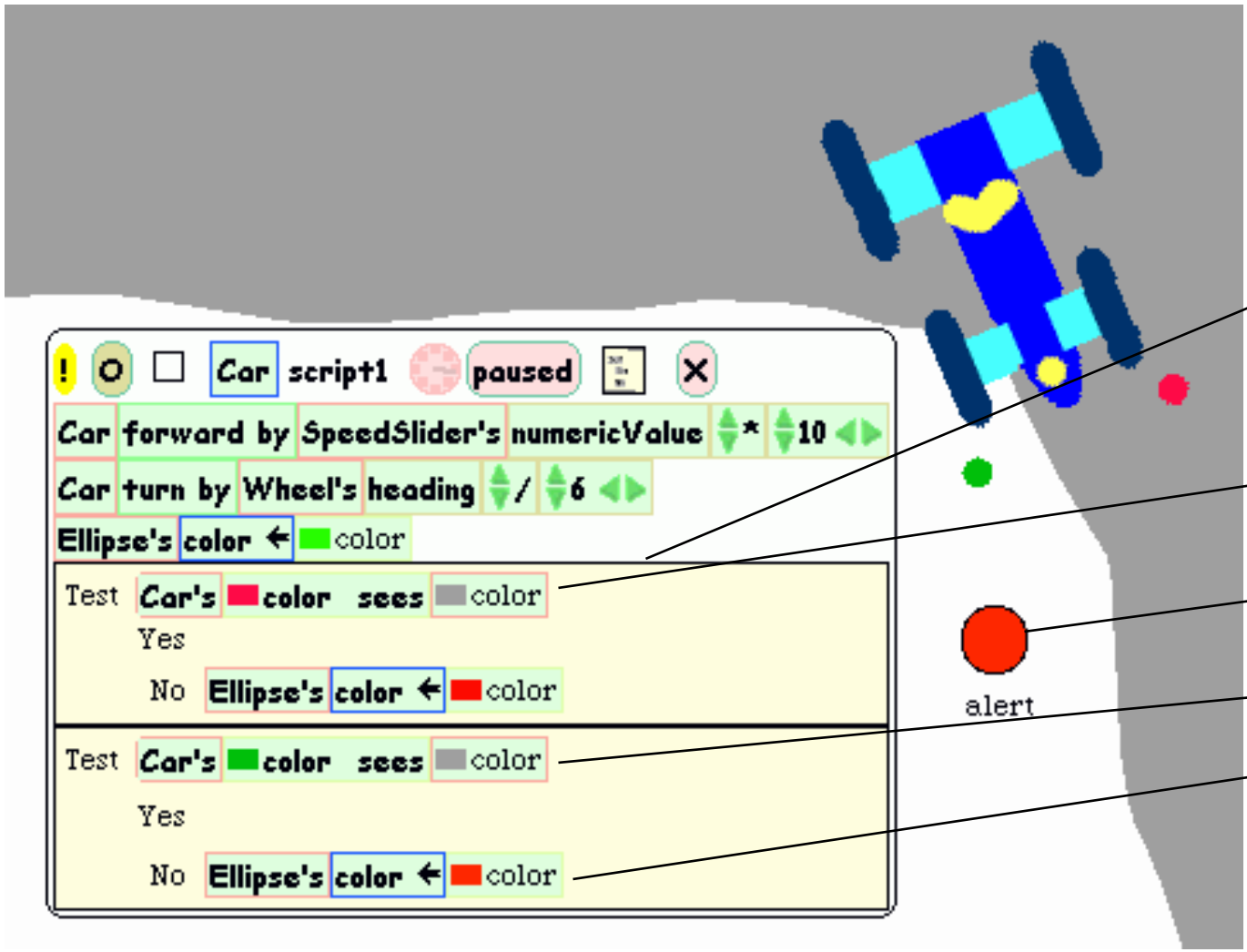


Sensors for Environment

- 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



Test tile

Test on left sensor

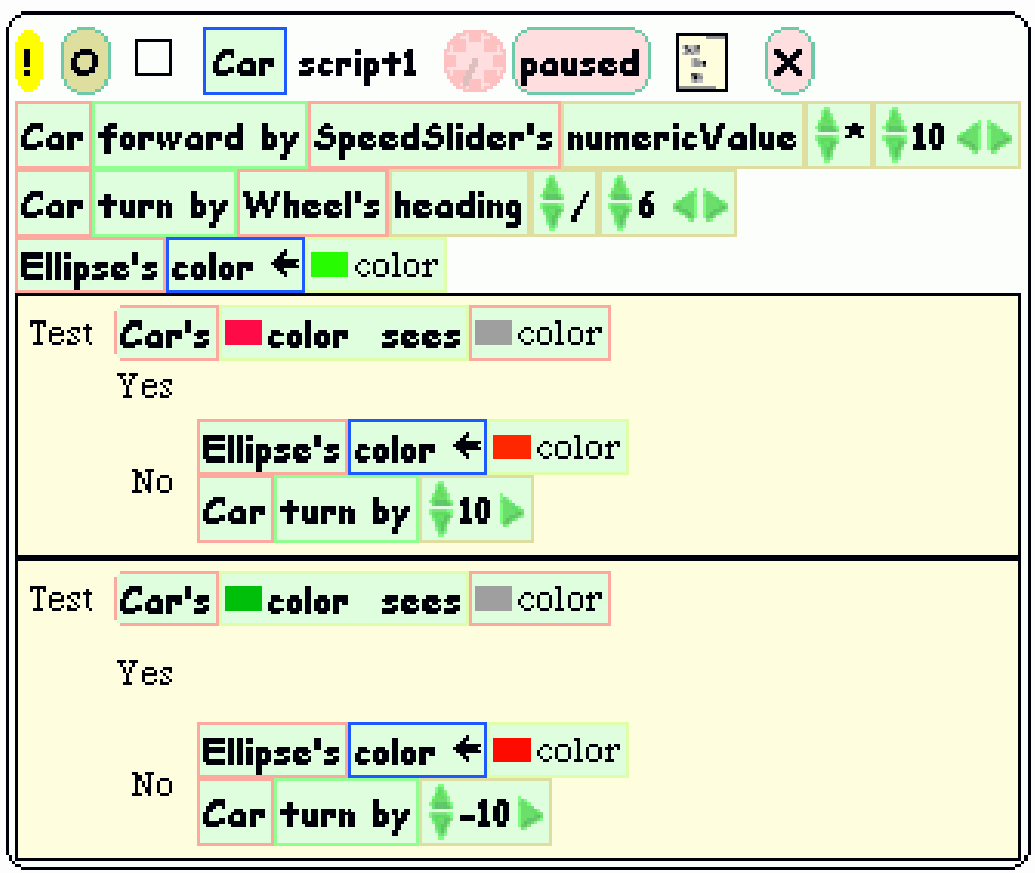
Alert lamp

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)



Wheel control better removed at this stage?