Prototyping User Interfaces in HyperCard

Alan Dix and Devina Ramduny

http://www.hiraeth.com/alan/tutorials/prototyping/

Prototyping User Interfaces in HyperCard

Alan Dix and Devina Ramduny

School of Computing
Staffordshire University
PO Box 334
Beaconside
Stafford ST18 0DG
email: A.J.Dix@soc.staffs.ac.uk
D.Ramduny@soc.staffs.ac.uk

Tutorial Notes

presented at

HCI’95, Huddersfield

Contents

Prototyping

- What is a prototype
- Why we need to prototype
- Who is it for
- What tools to use

What is a prototype?

A mock up or model

usually of a not-yet existing system

Similar where it matters ...

... but may differ where it doesn't:

- It may be partial
- It may lack functionality
- It may differ in appearance
- It may be built using different materials

Examples:

- 1/18 scale Jaguar XJ220
  1.56 Kilo ≈ 4 tons fullsize!

  1/40 scale Mercedes Benz
  0.11 Kilo ≈ 7 tons
- Architects model – must look right
- Windtunnel – Raleigh number

Why prototype?

- To refine the requirements
- To refine the design
- To compare designs
- To sell the design
- To demonstrate an idea
  (envisionment)
- To perform experiments

Who is it for?

- The client
- The users
- Yourself
- Your peers

Problems

The prototype is too poor

- appearance
- functionality
- robustness

The prototype is too good !

- no time
  users want it now
- no resources
  'you've almost finished'

Hard to solve both problems!

Lessons:

- understand your limitations
- educate your audience

Tools

- Full programming environment

  e.g., Visual C++, Think Pascal/C/C++
  often include visual layout tools

  ✔ can do anything

  ✘ hard to do anything

- Multimedia or presentation tool

  e.g., Macromind Director, PowerPoint
  may include some scripting facilities

  ✔ easier to use, good graphics

  ✘ very limitted order of presentation

- Rapid prototyping tool

  e.g., HyperCard, Visual Basic

  ✔ do a bit of both

  ✘ not always as well as you'd like

  ✔ evolutionary path

HyperCard

Features of:

- Hypertext authoring
- Database
- Interface designer

Basic model:

- A ‘stack’ of electronic filing cards

Each card consists of:

- Graphics: lines, shapes, clip-art,
  freehand painting
- ‘Fields’ for text
- ‘Buttons’ which link to other cards
  (or do things)

Plus HyperTalk

- full programming language
- ‘English’-like syntax ... well sort of
- object-based model   ... well sort of

... and more

‘Backgrounds’

(N.B. in HyperCard ‘background’ ≠ painting layer)

- group similar cards
- for common graphics, buttons etc.

Multimedia support

- play sampled sounds
- quicktime movies

Launch and control other programs

- single HyperTalk command

  `open` filename `with` application

  attach to a button to launch
- or use AppleScript (in v2.2 and above)

  full control of any scriptable application

Network communication

- prototype groupware ...

Many third party add-ons

- XCMDs & XFCNs – act as if built-in

e.g., database linkage, Internet

Storyboards

- Sequence of snapshots of system
- From animation and film industry
- Demonstrate a single scenario
- Usually part of a ‘talk through’
- Fixed or very limited set of options

Storyboards are good for

Quick demonstration of ideas

- during early design

Envisionment:

- new ideas not well supported by existing tools
- technology may not even currently exist

Addressing a specific feature

- bypass irrelevant features

N.B. the designer/demonstrator is in control

Low-tech prototyping

On paper:

- Hand or printed drawing on paper or card
- Very flexible – modify on the fly
- Cut outs, overlays etc.
- Reduce problems of over fidelity

On screen:

- Printed or even hand drawn OHPs
- Professional but purpose is clear

On computer:

- Use presentation tools
- Looks like the real thing

... HyperCard is almost overkill!

Using HyperCard

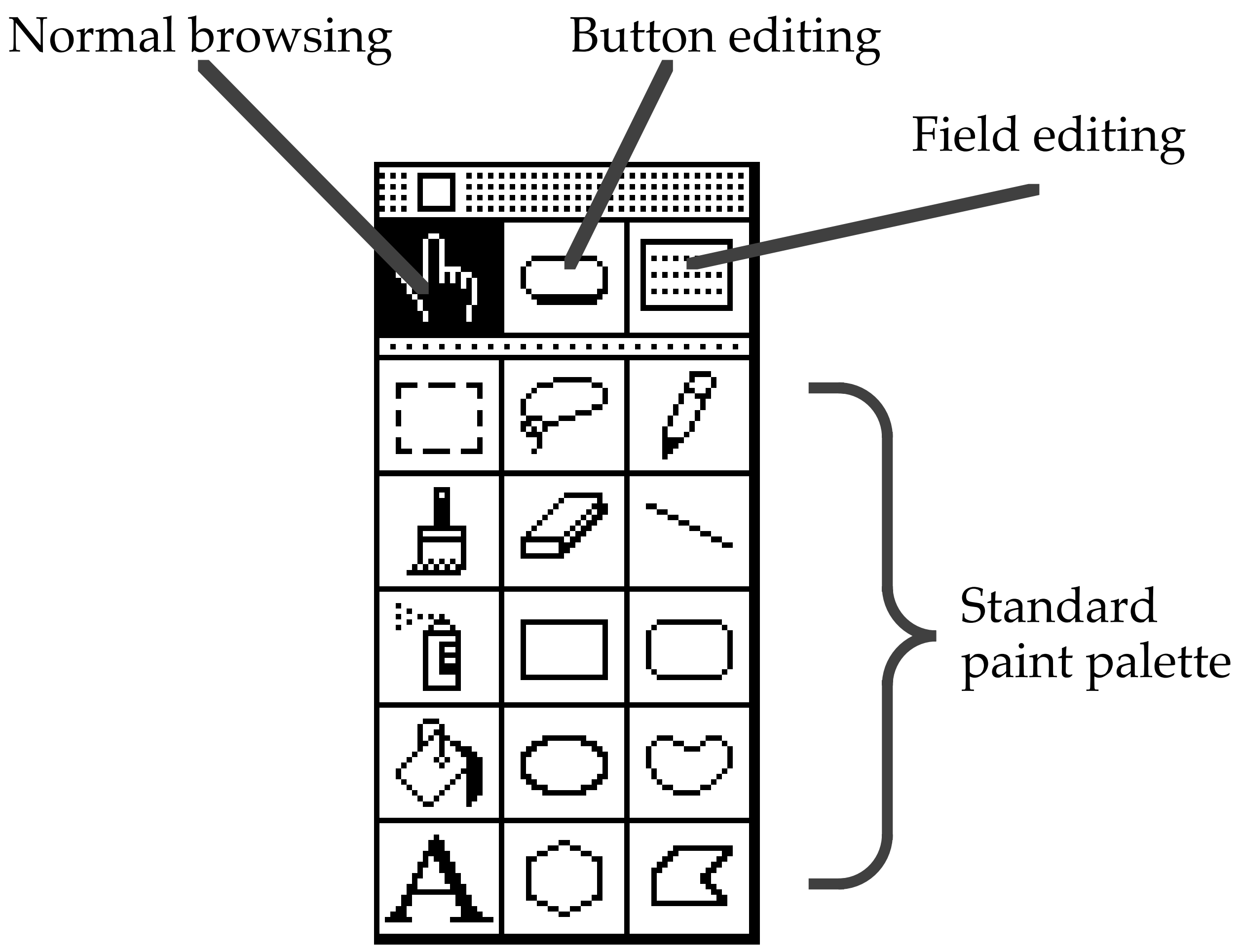
- Draw snapshots one percard
  - use painting pallete
  - and/or add fields and buttons
    (tip – text in fields is easier to edit)
  - many screens will be similar
    - ❍ copy cards or parts of cards
    - ❍ use backgrounds (see later)

- Move from card to card
  - Using next/previous buttons
  - Using cursor keys

- Ordering cards
  - New cards inserted after current card
  - Reorder by cut and paste
  - Also see move buttons in

Readymade Buttons

The tools palette



N.B. Selecting "New Button" from the "Objects" menu puts you into Button editing mode.
Similarly for fields.

Select the 'browse' tool when you're done

Using backgrounds

N.B. background ≠ the graphics in the paint layer

- Each cards is in a ‘background’

- Backgrounds have content:
  - graphics, fields and buttons (and scripts!)
  - shared by all the cards with that background
  - also individual card buttons, etc.
  - card elements always appear on top

- Editing backgrounds
  - use **B** to edit the backgound (or Edit menu)
  - card elements disappear (temporarily!)
  - proceed as for individual cards
  - see the “Objects” menu for a new background

- See also HyperCard Tour

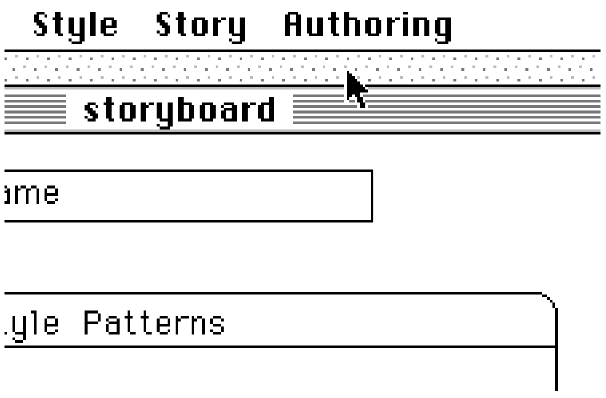
A simple storyboard shell

Special fields

- Screen name
- Sequence number

Menus for authoring

- move cards back and forth
- show/hide name, seq. nos. etc.

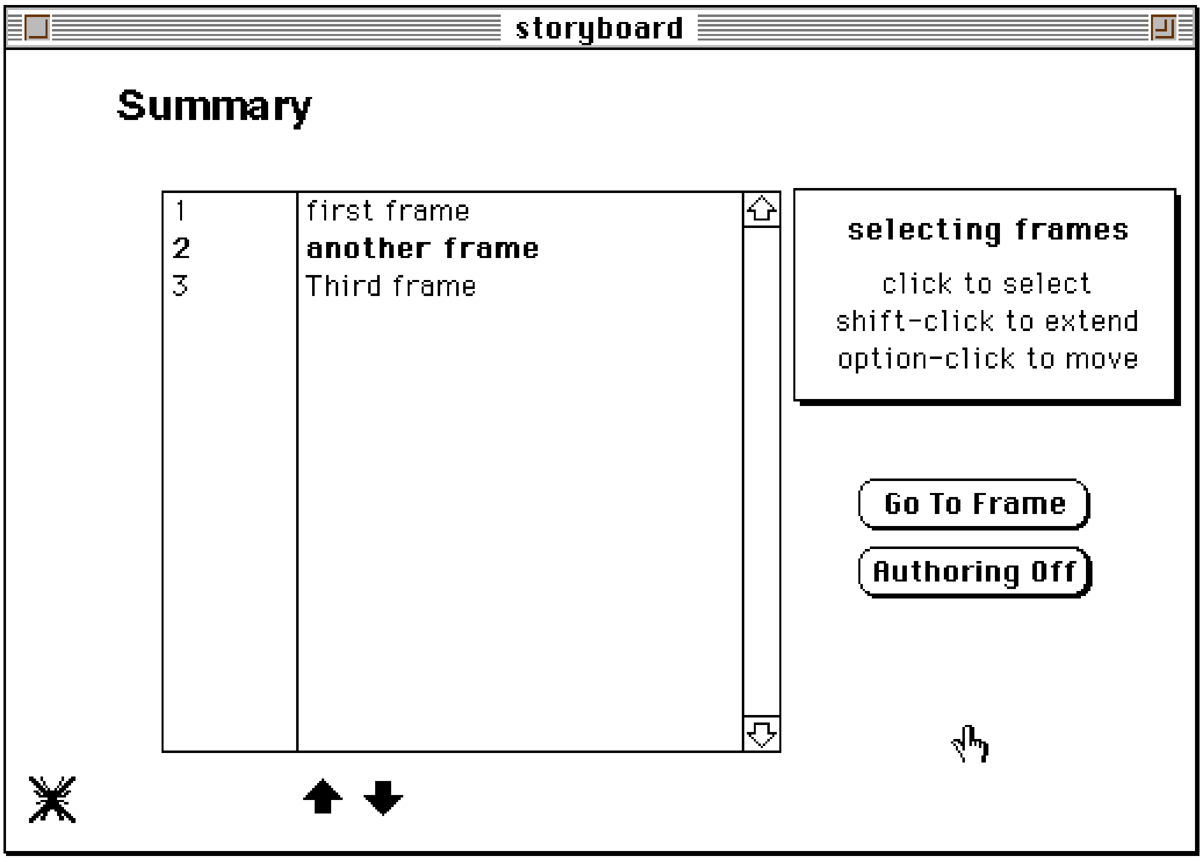


But only one background for storyboards

Storyboard shell – 2

Summary card

- Overview of presentation
- Skip to particular point
- Move cards or groups of cards



1	first frame
2	**another frame**
3	Third frame

selecting frames

click to select
shift-click to extend
option-click to move

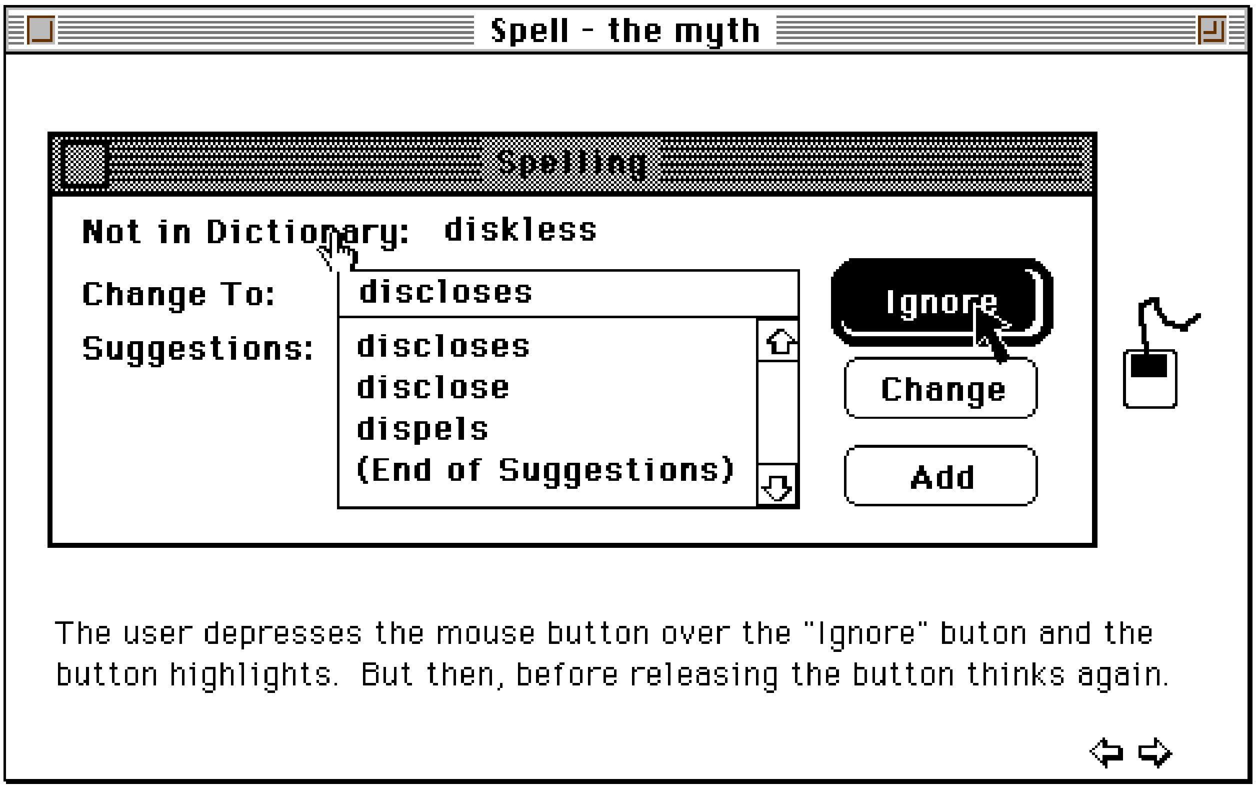
Extra flourishes

- Use links for alternative paths

  allow the audience some choices

- Animate and add explanatory text

  e.g., post as demo to prospective clients



The user depresses the mouse button over the "Ignore" buton and the button highlights. But then, before releasing the button thinks again.

Dialogue sequence

What is dialogue?

Minister:	Do you *man's name*, take this woman ...
Man:	I do
Minister:	Do you *woman's name*, take this man ...
Woman:	I do
Man:	With this ring, I thee wed ... (*places ring on woman's finger*)
Woman:	With this ring, I thee wed ... (*places ring on man's finger*)
Minister:	I now pronounce you husband and wife

- the <u>pattern</u> of interaction
  ... but not the meaning (semantics)

  - ❍ some variations don't matter
    (e.g. "woman's name")
  - ❍ some do ! (e.g. "I do")
  - ❍ alternative orders ...

Alternative paths

Storyboard

- 1 fixed sequence

  e.g. “Myth” stack 2 scenarios not an alternative

Dialogue

- all states/modes
- all alternatives
- all paths

Using HyperCard

- each card $\equiv$ 1 state/mode
- buttons to move between states

N.B. cards are ‘cheap’ $\Rightarrow$ can be lots & lots

Building a HyperCard dialogue

① Specification

- sketch out main modes/states

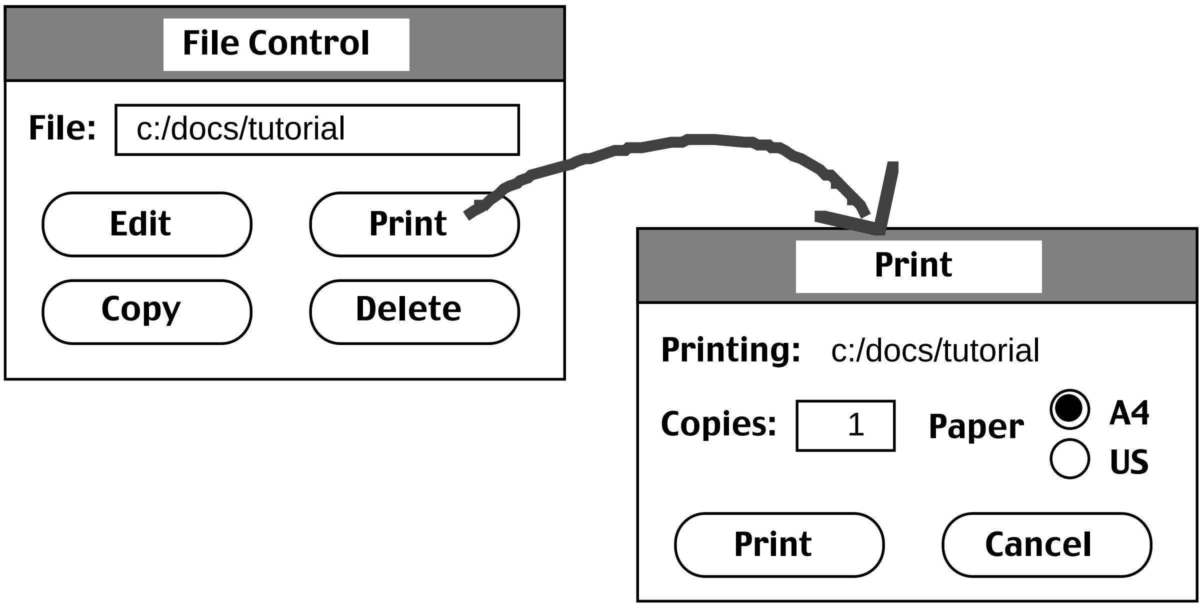
  (see Dix et al. 1993, ch. 8, for techniques)

② Create card for each state

- copy cards
- use backgrounds
- use graphics and fields ...

  ... and especially buttons!

③ Link screens using buttons



Creating a link

① select button tool

(from tools palette)

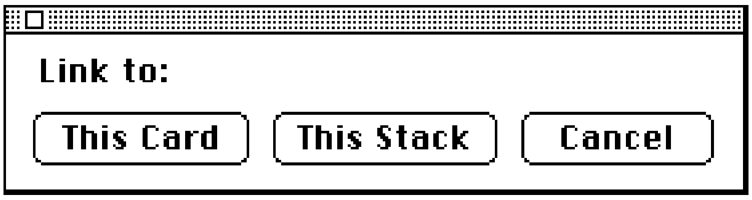
② select button to link

③ use "**Button Info ...**" from "**Objects**" menu

(or double click button)

√ click "**LinkTo...**" option

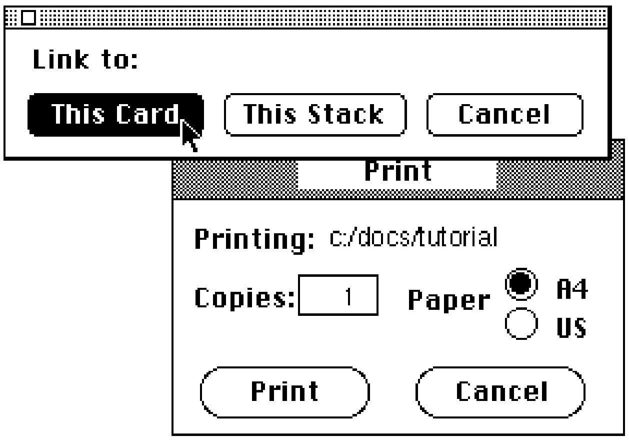
<< palette appears >>



⑤ move to the destination card

(use cursor keys or card buttons)

≈ click "**This Card**" on palette



Realism

N.B. this is a prototype

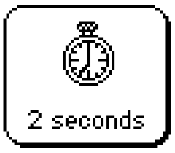
What’s on a card?

- card = screen
- card = window
- card = part of screen/window (close up)
- part of card = any of the above
  leave room for annotations, title etc.
- extra cards for ‘in between’ states

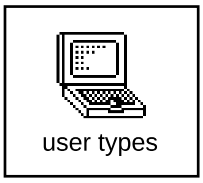
HyperCarders do it with buttons
... but what if you can’t manage it

- invisible buttons for active areas
- extra button area for non-button actions

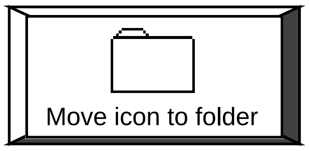
time



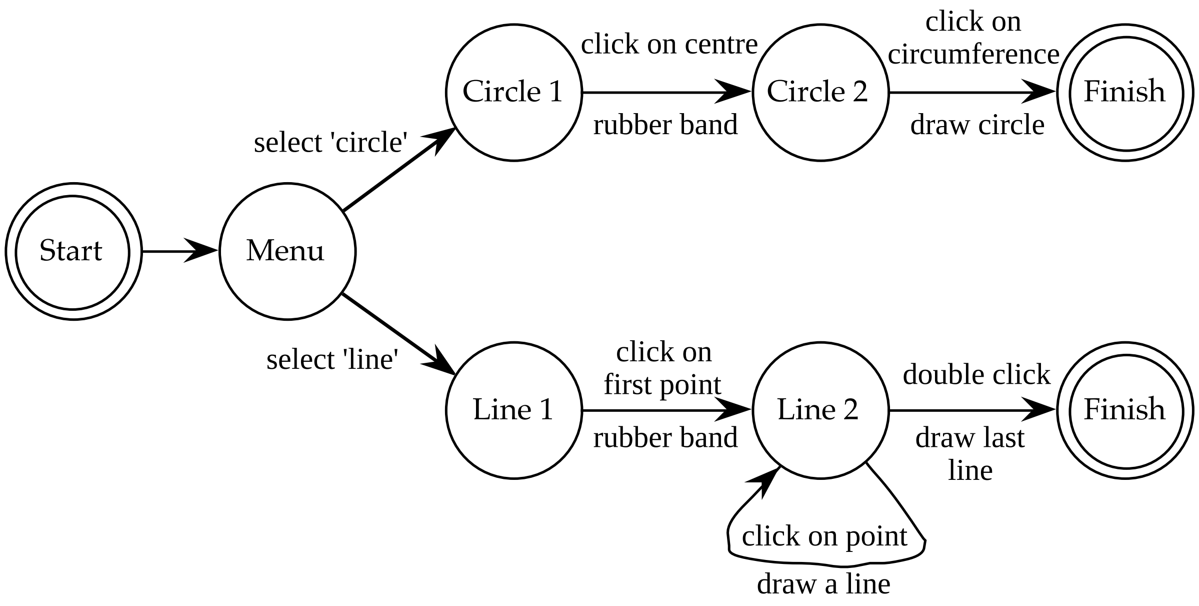
text ...

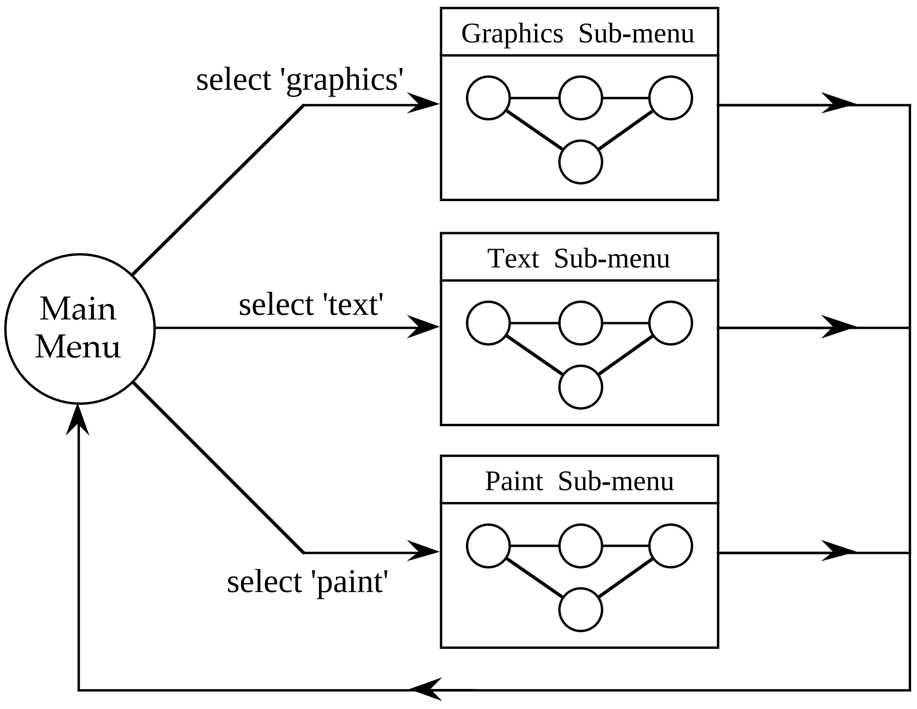


and direct manipulation



Example – drawing tool

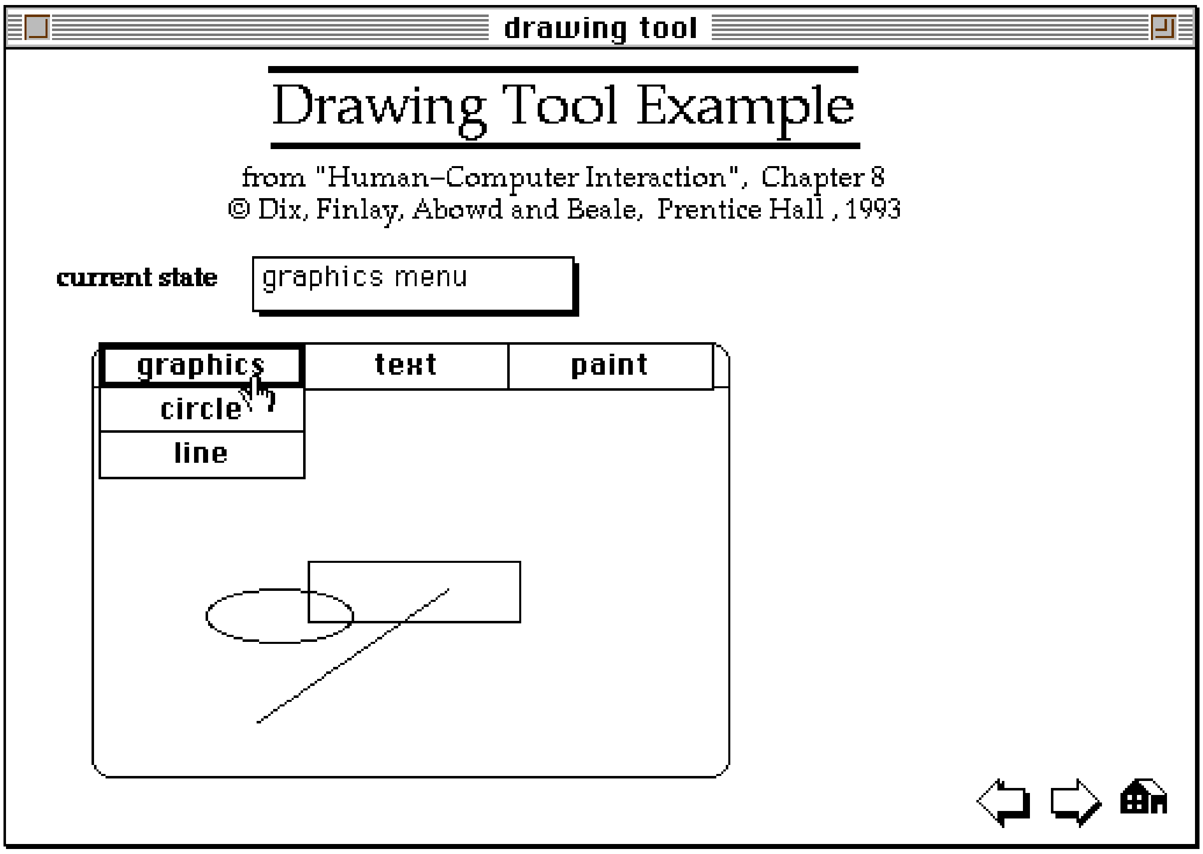




(from *Human–Computer Interaction*, Dix, Finlay, Abowd and Beale, 1993)

Drawing tool – 2

Create a screen for each state ...


drawing tool
Drawing Tool Example
from "Human–Computer Interaction", Chapter 8
© Dix, Finlay, Abowd and Beale, Prentice Hall , 1993
current state
graphics menu
graphics
text
paint
circle
line

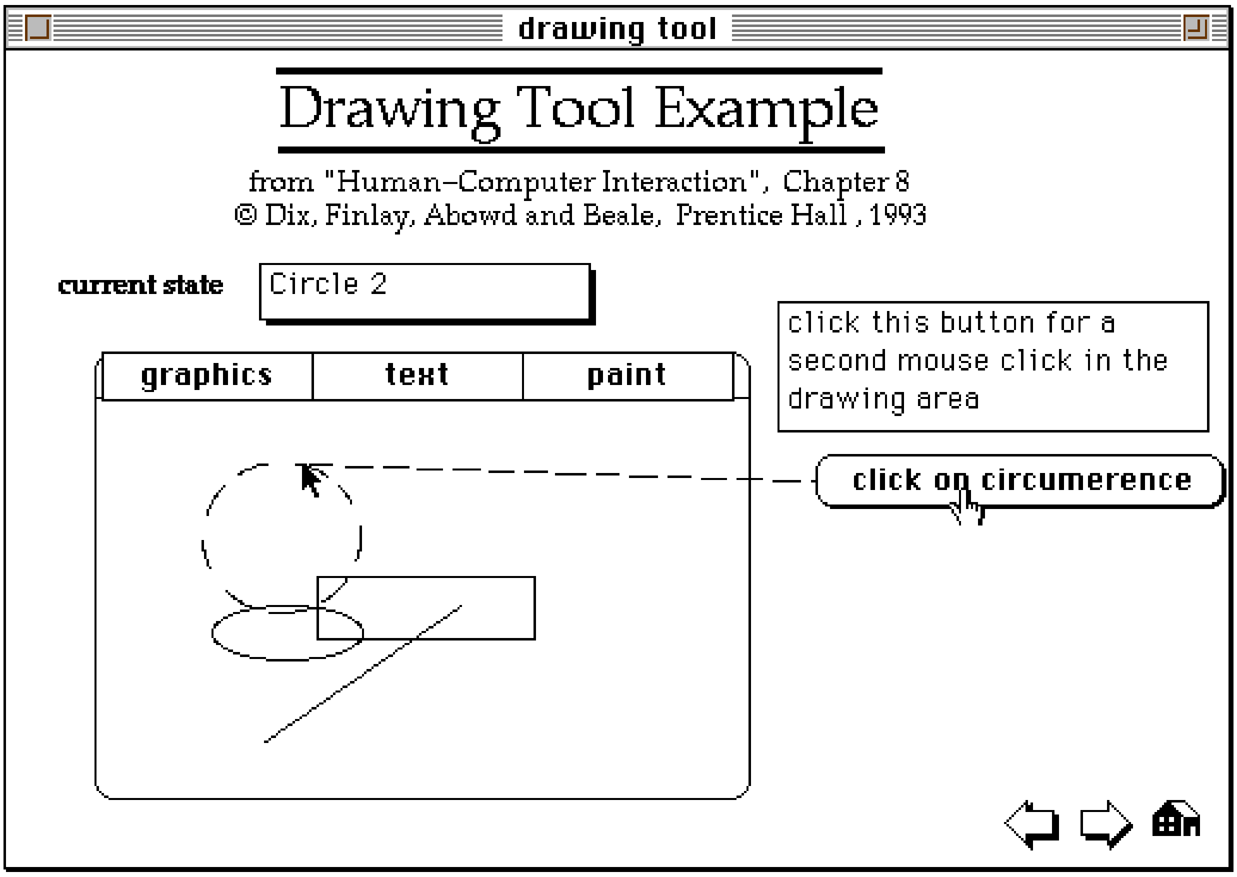
... and add buttons

N.B. button on 'menu bar'

Drawing tool – 3

What if you can’t do it with buttons?

❋ when in doubt simulate

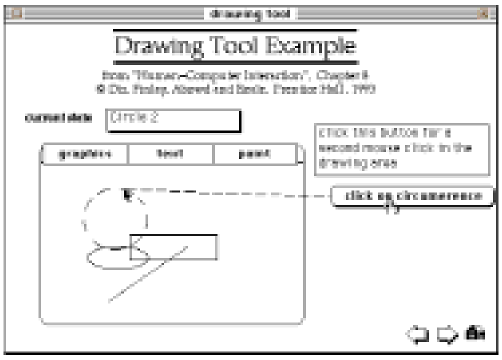

drawing tool
Drawing Tool Example
from "Human–Computer Interaction", Chapter 8
© Dix, Finlay, Abowd and Beale, Prentice Hall, 1993
current state
Circle 2
graphics
text
paint
click this button for a second mouse click in the drawing area
click on circumerence

But no semantics

Fixed screens

⇒ no permanent effect

e.g., circle is drawn in drawing tool p'type,
but disappears on return to main menu



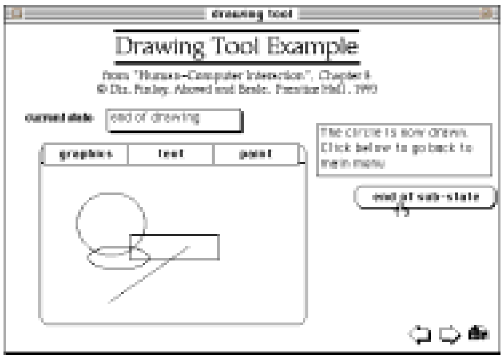
drawing tool

Drawing Tool Example

from "Human–Computer Interaction", Chapter 8
© Dix, Finlay, Abowd and Beale. Prentice Hall, 1993

current state: Circle 2

click this button for a second mouse click in the drawing area



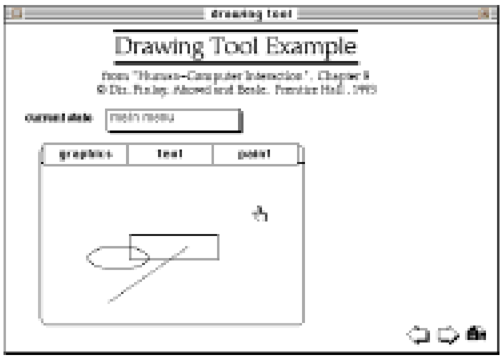
drawing tool

Drawing Tool Example

from "Human–Computer Interaction", Chapter 8
© Dix, Finlay, Abowd and Beale. Prentice Hall, 1993

current state: end of drawing

The circle is now drawn. Click below to go back to main menu



drawing tool

Drawing Tool Example

from "Human–Computer Interaction", Chapter 8
© Dix, Finlay, Abowd and Beale. Prentice Hall, 1993

current state: main menu

Get More from Dialogue

- Can do checks on dialogue

    completeness:

        have you considered every action?

    reachability:

        are there any dead ends?

    reversability:

        how difficult is it to 'undo' an action?
        N.B. 'undo' only at dialogue level

✘ But, need explicit description

✔ Use dialogue notation

- use it to start with
- extract it from existing stacks

Use it to start with

① Specify the dialogue

- ❋ in a machine readable form

② Execute the specification

- to <u>see</u> what it's like
- either abstract state names
  or real screen imagess

③ Analyse the specification

- exhaustive testing
- global properties

Examples:

- HyperDoc (Thimbleby)
  – state transitions
- Action Simulator (Monk)
  – production rules

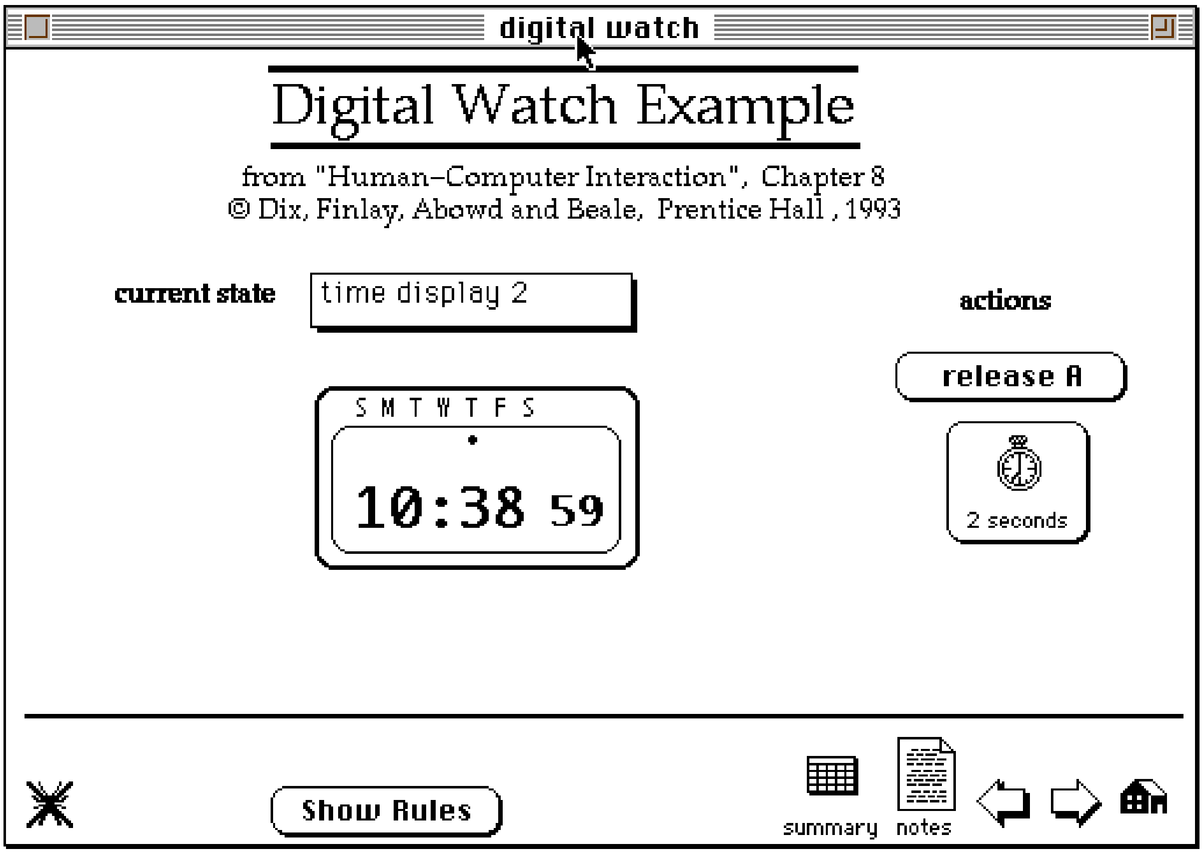
A Dialogue Simulator

Demonstrator only

– example of prototyping!

Dialogue construction:

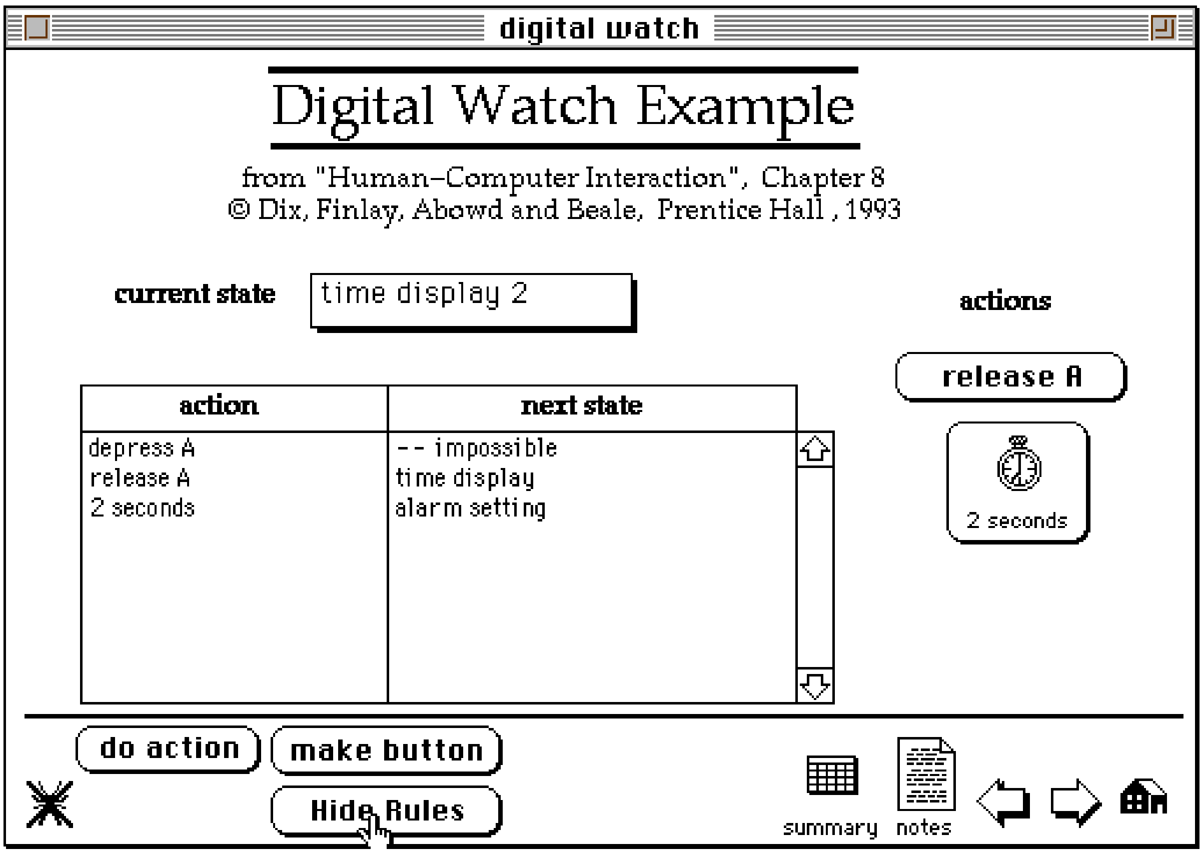
- draw screens by hand
- all screens share a single background
- buttons added using tool



Action–effect rules

Each state has a rule table

- ❍ lists each possible action from the state
- ❍ and says what the next state will be



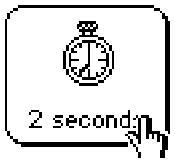
from "Human–Computer Interaction", Chapter 8
© Dix, Finlay, Abowd and Beale, Prentice Hall , 1993

current state: time display 2

action	next state
depress A	-- impossible
release A	time display
2 seconds	alarm setting

Executing the rules

- When a button is pressed



- NOT linked directly

- Instead table used ...

  ① action looked up in table
  ② corresponding next state found
  ③ if "— impossible" – error message
  √ if "— do nothing" – stay put
  ⑤ otherwise go to the next state

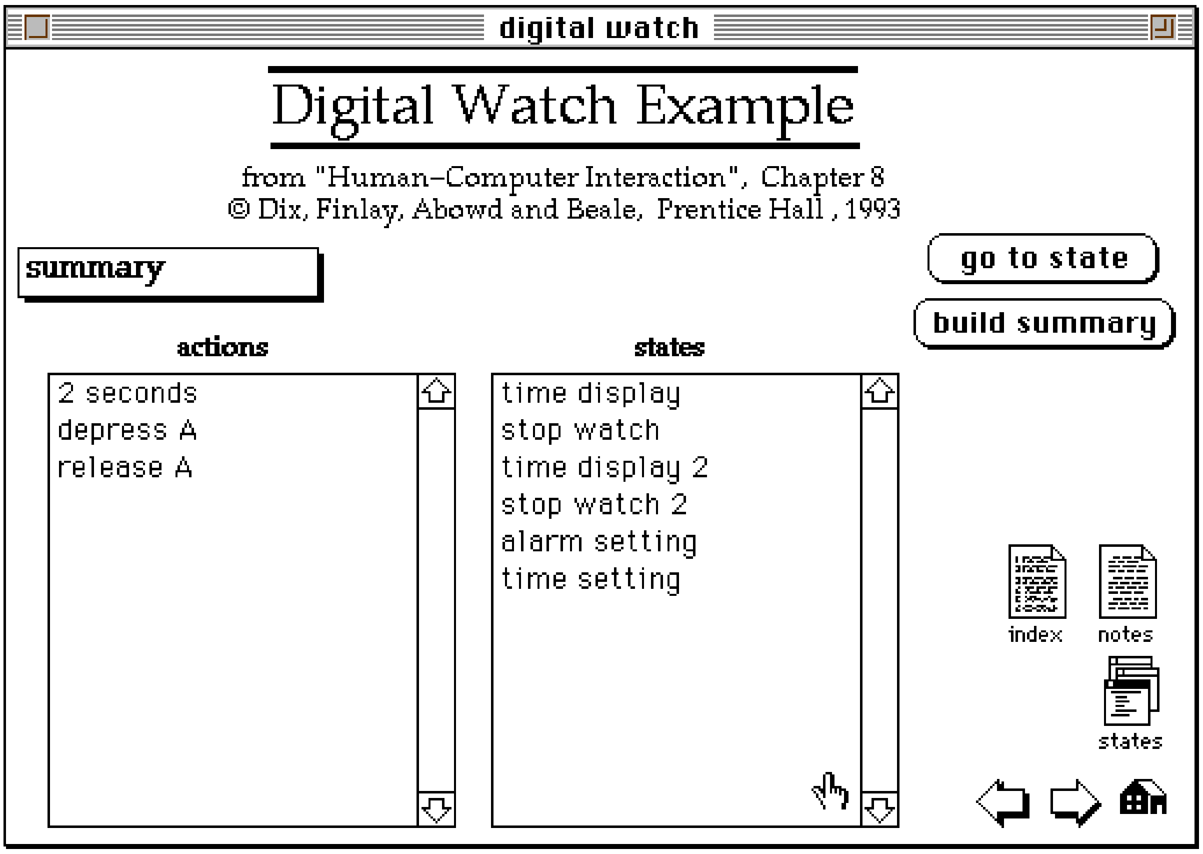
Analysing the rules

Summary card lists:

- all states
- all actions used in any state

Can check:

- that all 'next states' exist
- reachability from any state
- that all actions have rules in each state



Digital Watch Example

from "Human–Computer Interaction", Chapter 8
© Dix, Finlay, Abowd and Beale, Prentice Hall, 1993

summary

actions	states
2 seconds	time display
depress A	stop watch
release A	time display 2
	stop watch 2
	alarm setting
	time setting

Extracting dialogue

Analyse an existing program

- ✔ Don’t need to use a special notation
- ✘ Hard – dialogue and semantics mixed

Example code (from ‘Inside Macintosh’)

```
REPEAT
    gotEvent := WaitNextEvent( everyEvent, myEvent, 15, NIL )
    If NOT DoHandleDialogueEvent(myEvent) THEN
        BEGIN
            CASE myEvent.what OF
                mouseDown:          DoMouseDown(myEvent);
                updateEvt:
                    DoUpdate(WindowPtr(myEvent.message));
                keyDown, autoKey:   DoKeyDown(myEvent);
                        etc
                nullEvent:          DoIdle(myEvent);
                OTHERWISE:          ;
            END;  {CASE}
        END
    ELSE
        DoIdle(myEvent);
UNTIL gDone;
```

Work that out !

Analysing HyperCard

- Can be as bad ...

```
get the long date
convert it to dateItems
lock screen
go card it
```

(script from Appointments)

- But ...

  ... <u>if</u> stack <u>only</u> uses buttons linked with “Link to ...”
  relatively easy to extract dialogue

- Used in “Collector” stack
  - ❍ on-going collaborative project
    with Gregory Abowd and others, Georgia Tech.
  - ❍ part of suite of dialogue analysis tools
    - ① extract dialogue from HyperCard stack
    - ② turn into structured state update rules
    - ③ use in proof checker
      (reachability etc.)
  - ❍ simple version of ① included as taster ...

Adding functionality

Multimedia systems

- nearly there already !

    simply add quicktime + audio

- invisible buttons

    prototype for WWW clickable maps

Information/database system

- example stacks:

    Appointments Addresses

- lots of text entry fields!

Groupware

- use AppleEvents or MacTCP
    another tutorial !

✚ HyperTalk

- full programming features
- you can do anything you want ...

    ... or perhaps anything you can

Information systems

Simple add/delete/edit

- put fields in background
- add/delete using **New Card**/**Delete Card**
- edit using normal H'Card field editing

N.B. no explicit save

Extras:

- use "find" to do searches
- built in report printing features
- computed values:
  - use "closeField" message to update
- build summary cards

```
on makeList
   put number of cards of background “data” into nos
   put empty into card field “list”
   repeat with n = 1 to nos
      put background field “name” of card n of ¬
            background “data” after card field “list”
      put return after card field “list”
   end repeat
end makeList
```

HyperTalk

Full scripting / programming language

Data types

- everything is text

  access by character, word, item, line, etc.

```
put "hello" into word 2 of the last line of card
field "greeting"
```

- stored in
  - ❍ global variables – VERY global
  - ❍ fields on cards – can be hidden

Scripts

- English-like syntax

  (actually rather like COBOL!)

  be careful of "it"

- Poor structure, but good editor
- Sort of object-oriented

Objects?

Fixed set:

- stacks, backgrounds, cards, buttons and fields

Message passing order

- ‘top most’ object gets it
    - ➨ button or field
        - ➥ card
            - ➥ background
                - ➥ stack
                    - ➥ . . .
    - ❍ N.B. containment order, not sub-class
- but additional rules when
    - ❍ several stacks active
    - ❍ in the middle of “go to card” or “send to”

Whose fields?

message context ≠ data context

Direct manipulation

Support for all parts of WIMP interface
... but some better than others!

Speed is a problem

- lots of clever tricks
  (raid other peoples)
- work with HyperCard
  use its facilities, don't invent your own!

Remember cards are cheap

- use 'slide show' methods
- the user sees a single screen ...
  ... but really several cards
- use 'visual' effects for transitions
  (HyperCard Tour is a great example)

Windows

Three kinds of windows

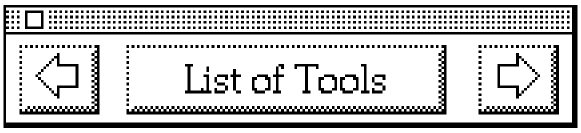
- Built in modal dialogue boxes:
  - ❍ answer – for choices
  - ❍ ask – for text entry

- Opening several stacks at once

```
go to stack “Month Calendar” in new window
```

  - ❍ write multi-window applications ...

- Palettes

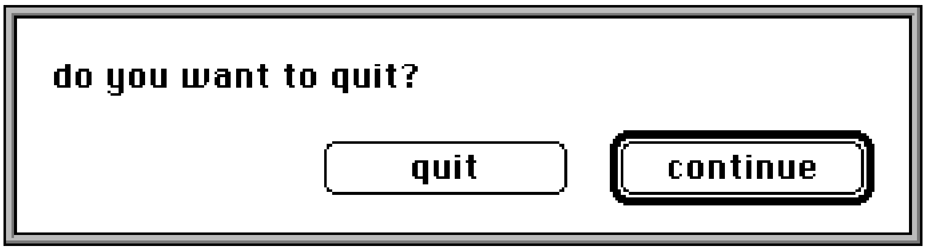


  - ❍ background graphic with active areas
    (like a WWW clickable map)
  - ❍ use PowerTools stack to make them
  - ❍ useful ones there too

- also see “picture” XCMD in PowerTools stack

Built in dialogue boxes:

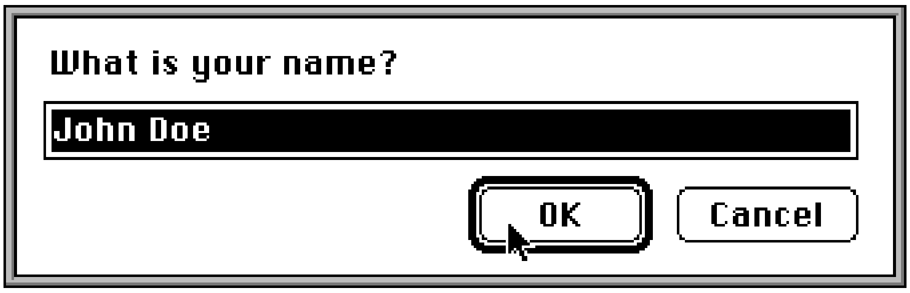
The answer dialogue box



- selects between two alternatives
- optional "with" clause to customise buttons
- choice returned in "it"

```
answer "do you want to quit?" ¬
       with "quit", "continue"
if it is "quit"    . . . etc.
```

The ask dialogue box



- prompts the user for text
- optional "with" clause for default text
- text returned in "it"   (empty if cancelled)

```
ask "What is your name?" with "John Doe"
if it is empty    . . . etc.
```

Icons

Icons on buttons

- use "**Icon…**" on "**Button Info**" to set icon
- Icon editor to make your own
    - ❍ use "**Edit…**" from icon selector
      or "**Icon..**" from "**Edit**" menu

Icons elsewhere

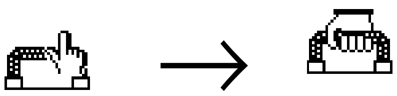
- palettes – icons part of graphic
- similar trick on cards with transparent button
- can also set cursor shape

Special effects using scripts

- "hide" and "show" buttons
- change button icon and cursor shape
  e.g., the 'amazing moving suitcase' card

mouseDown handler

```
        set the icon of me to "held handle"
        set the cursor to none
```



mouseUp handler

```
        set the icon of me to "handle"
        set the cursor to hand
```

Menus

HyperTalk lets you:

- intercept standard menu messages

```
on doMenu theItem, theMenu
  . . . etc.
```

- invoke menu actions

```
doMenu "New Card", "Edit"
```

- have pop-up menus (v2.2)

  see "**Popup**" style and "**Contents..**" in Button Info

- add your own menus

```
create menu "Story"
put "Next,Previous,Go Summary,-,Show Title," ¬
      into menu "Story" with menuMessages ¬
"nextFrame,prevFrame,goSummary,,showTitle"
set the commandChar of menuitem ¬
      "Go Summary" of menu "Story" to "S"
```

Pointers

Use the mouse for

- custom selections
- dragging and moving
- your own DM effects ...

HyperCard sends messages

- when the mouse button is pressed or released
  mouseDown, mouseUp
- when the mouse is dragged
  mouseStillDown
- when it enters or leaves a button or field
  mouseEnter, mouseLeave
- when it moves over a button or field
  mouseWithin

Messages go to:

- buttons
- locked fields (all messages)
- all fields (mouseEnter, mouseLeave, mouseWithin)
- card, background or stack
  (if no button or field handler)

Mouse messages

Order of messages very important

(see examples in 'moving buttons' stack)

Typical order:

```
mouseEnter, mouseWithin, ...,
     mouseDown, mouseStillDown, ..., mouseUp,
          mouseWithin, ..., mouseLeave
```

But if the user drags off the button:

```
mouseEnter, mouseWithin, ...,
     mouseDown, mouseStillDown, ...,
                    (mouse dragged off button here)
          mouseStillDown, ..., mouseLeave
```

Often see script where:

- mouseDown handler initialises something
- mouseStillDown adjusts it
- mouseUp tidies up

✘ Wrong – no mouseLeave handler

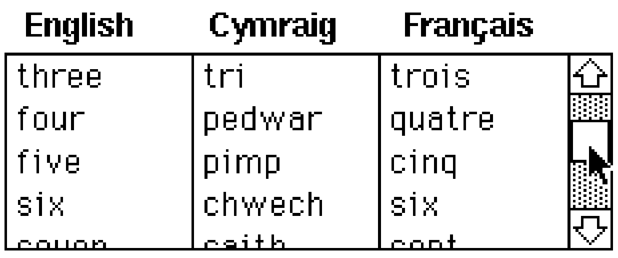
- things left in unfinished state

Mice and fields

Multiple scrolling fields

(see 'Power Tools' and several tutorial stacks)

- uses mouseWithin handler

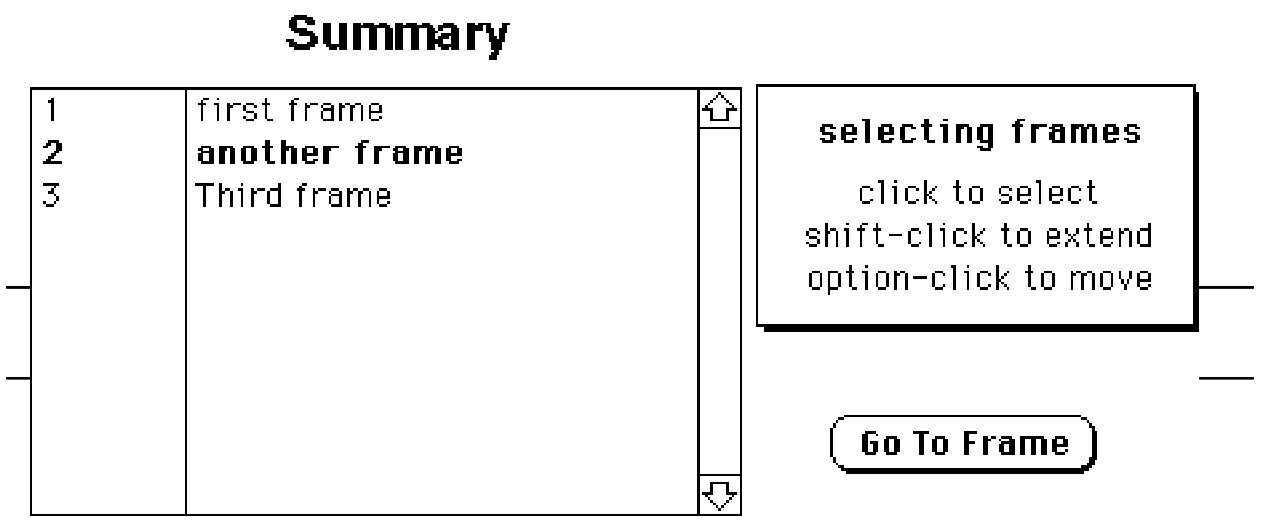


English	Cymraig	Français
three	tri	trois
four	pedwar	quatre
five	pimp	cinq
six	chwech	six

Selecting locked text

(see summary card of Storyboard shell)

- uses mouseDown handler and the clickLine



Summary

1	first frame
2	**another frame**
3	Third frame

selecting frames

click to select
shift-click to extend
option-click to move

Mice and buttons

Highlighting

if 'Auto Hilite' is not sufficient

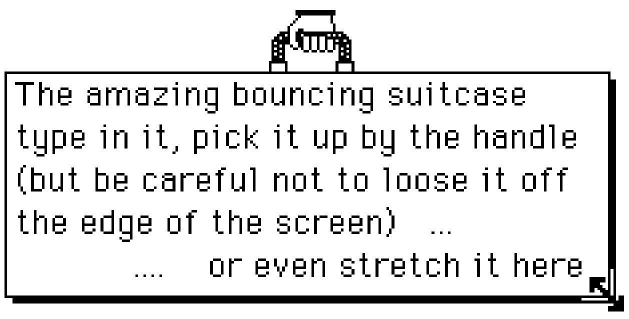
- to set icon or highlight button:
  - ❍ when under mouse
    - use mouseEnter and mouseLeave
  - ❍ when depressed
    - use mouseDown, Up and Leave
- to animate button:
  - ❍ when under mouse – mouseWithin
  - ❍ when depressed – mouseStillDown

Dragging (see "moving buttons" stack)

① on mouseDown
   – change to dragging icon

② on mouseStillDown
   – move loc of button to the mouseLoc

③ on mouseUp (and mouseLeave!)
   – restore icon and do anything else



The amazing bouncing suitcase
type in it, pick it up by the handle
(but be careful not to loose it off
the edge of the screen) ...
.... or even stretch it here

Comparing Tools

storyboard ...... ①

dialogue .... ②

information system .... ③

direct manipulation .... √

	①	②	③	√
Presentation tools	✔✔✔	? (script)	✘	✘
WWW	? (slow)	✔	✔	Hot Java!
HyperCard	✔	✔✔✔	✔✔	✔
Database 4GL	✘	?	✔✔✔	?
Visual Basic	?	✔ ?	✔✔✔	✔
C++/Pascal	✘	?	✔✔	✔✔

Further information

HyperCard

One of the best ways to find out about HyperCard is to look at other people's stacks. However, this is not as easy as some claim. If you are used to scripting other applications: spreadsheets, databases etc., or programming, you will probably find HyperTalk quite easy to get into. Happily, HyperCard comes with good tutorial and reference material. The stacks designed for this tutorial also demonstrate some features, and you'll find more ideas in the stacks supplied with HyperCard. A good start-off to scripting is to take an existing stack and modify it a little.

- G. Coulouris and H. Thimbleby, *HyperProgramming – Building Interactive Programs with HyperCard* , Addison–Wesley, Wokingham, UK, 1993.

This assumes no initial knowledge of HyperCard, but delves quite deeply into scripting with HyperTalk. It teaches HyperCard 'tricks', but does so in a disciplined manner which will allow you to build applications which are not 'write-only'! The accompanying disk has many examples including examples of hypertext, animation and direct manipulation.

Dialogue and Interface Design

See any leading HCI text book ... for example (!)

- A. Dix, J. Finlay, G. Abowd and R. Beale, *Human–Computer Interaction*, Prentice Hall International, UK, 1994.

Chapter 8 compares different dialogue notations and discusses different things you can use dialogue descriptions for (including prototyping!).

Harold Thimbleby's Hyperdoc is described in several places including:

- Thimbleby, H.W. (1993). Combining systems and manuals. In *HCI '93: People and Computers VIII,* J.L. Alty, D. Diaper and S. Guest (Eds.) pp. 479-488. Cambridge University Press.
- Thimbleby, H.W. (1995). Hyperdoc: an interactive systems tool. In *HCI '95, People and Computers X,* M. Kirby, J. Finlay and A. Dix (Eds.). Cambridge University Press.

These describes the Hyperdoc tool, which supports simulation, dialogue analysis and automatic documentation. This is ongoing work, so grab Harold at the conference ...

If you're prototyping ready for a full Macintosh application, you should read the *Macintosh User Interface Guidelines* (Addison-Wesley). Alternatively, if you are prototyping on the Mac, but the target is another platform consult the appropriate style guide such as the *Motif Style Guide* (OSF & Prentice Hall), or IBM's Common User Access (CUA) specification.