CSE 440:
Introduction to HCI

13: Patterns and Interface Implementation

May 7, 2024

Jesse J. Martinez | Avery Mack | Simona Liao

Project Status*

3a – Paper Prototype: Due Tomorrow @ 3pm

3b – Heuristic Evaluation: In-Class Activity on Thursday

3c/3d – Usability Testing:

- Check-in (3c) due Thurs 3/16 @ 8pm
- Review (3d) due Mon 3/20 @ 3pm

The Exam*

Scheduled for next Tuesday, 5/14, @ standard Lecture time

Planned format:

- 10-13 Short Answer Questions
- 0-1 Long Answer Question(s)

A few notes on Grading*

Exam is NOT curved/adjusted

How Final Grades work:

- Everyone gets a Raw grade for Core Content
  - Core Content: Project + Exam + Participation
  - Scaled linearly from 0 (0%) – 3.7 (100%)
- All grades shifted *positively* by $3.7 - Raw_{max}$
- EXP "Bonus" added to adjusted grade
  - Each EXP = +0.02; cap at +0.4 for 20EXP
- Final Adjustments
  - Any scores above 4.0 → clipped to 4.0
  - Any "borderline" grades adjusted up

Overview

Controlled A/B Experiments

Patterns

Case Study of Model-View-Controller

Case Study of Animation

Role of Interface Tools

Objectives

Be able to:

Describe the benefit of design patterns, contrast them with guidelines or templates

Describe the Model/View/Controller approach to organizing interface implementation

Describe how tools evolve with patterns, the benefits we gain from interface tools

Describe why tools eventually limit design thinking

Controlled A/B Experiments

Some questions ultimately resolve through data

Example: Amazon Shopping Cart Recommendations

- Add an item to your shopping cart
- Most sites show the cart

At Amazon, Greg Linden had idea to
show recommendations based on cart items

From Greg Linden's Blog: http://glinden.blogspot.com/2006/04/early-amazon-shopping-cart.html

Controlled A/B Experiments

Evaluation

- Pro: cross-sell more items
- Con: distract people from checking out

Highest Paid Person's Opinion:

- Stop the project

Simple experiment run:

- Wildly successful

From Greg Linden's Blog: http://glinden.blogspot.com/2006/04/early-amazon-shopping-cart.html

Controlled A/B Experiments

Many names for it

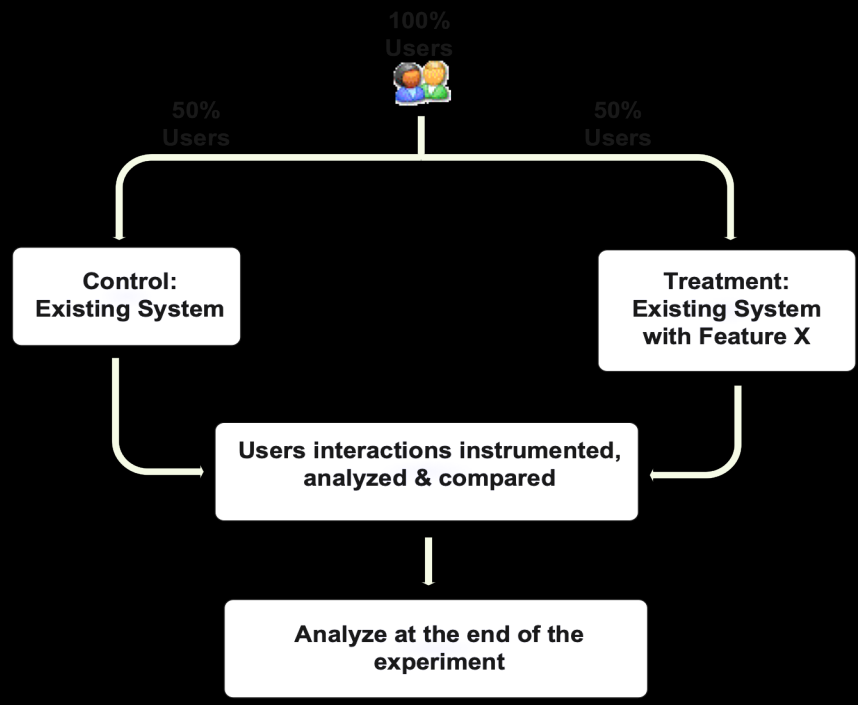
A/B tests or
Control/Treatment

Randomized
Experimental Design

Controlled experiments

Split testing

Parallel flights

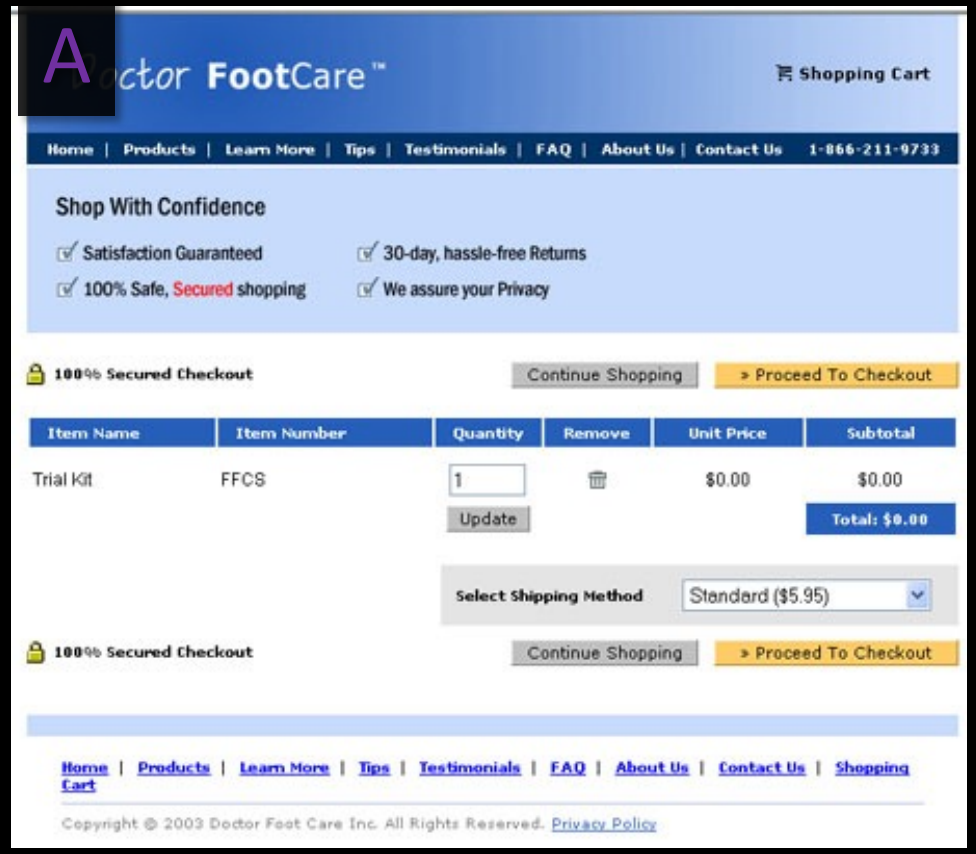


(examples from Ronny Kohavi)

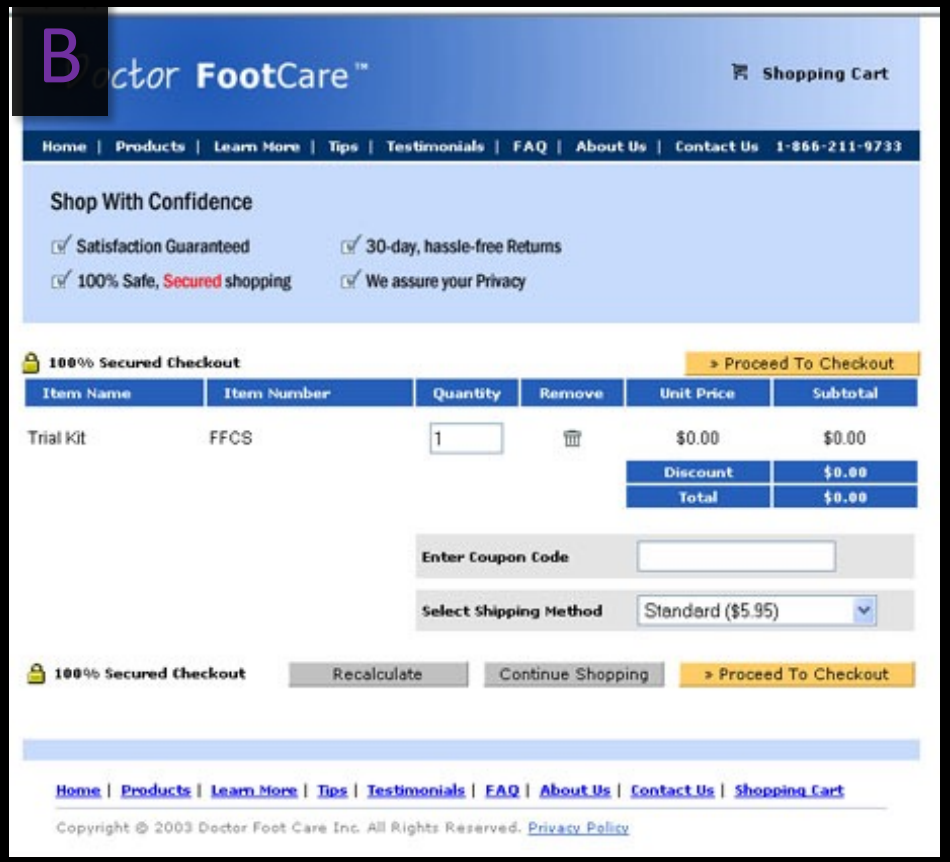
Checkout Page

Conversion rate is percentage of visits that include purchase

A



B

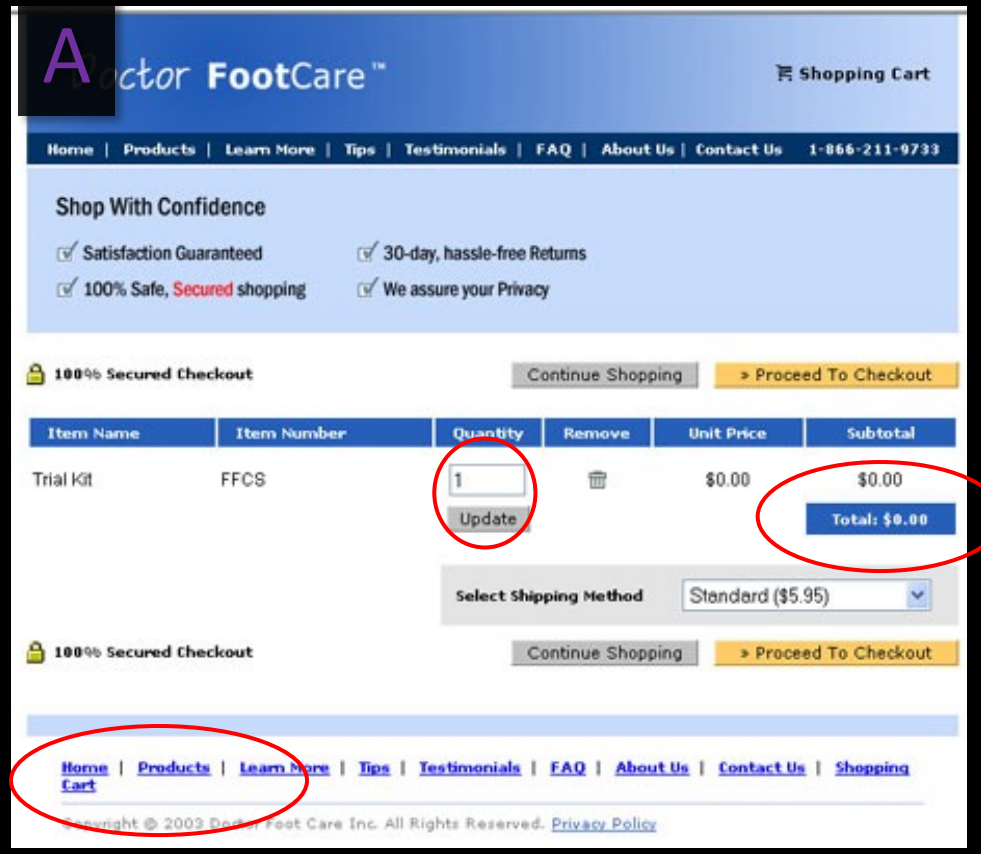


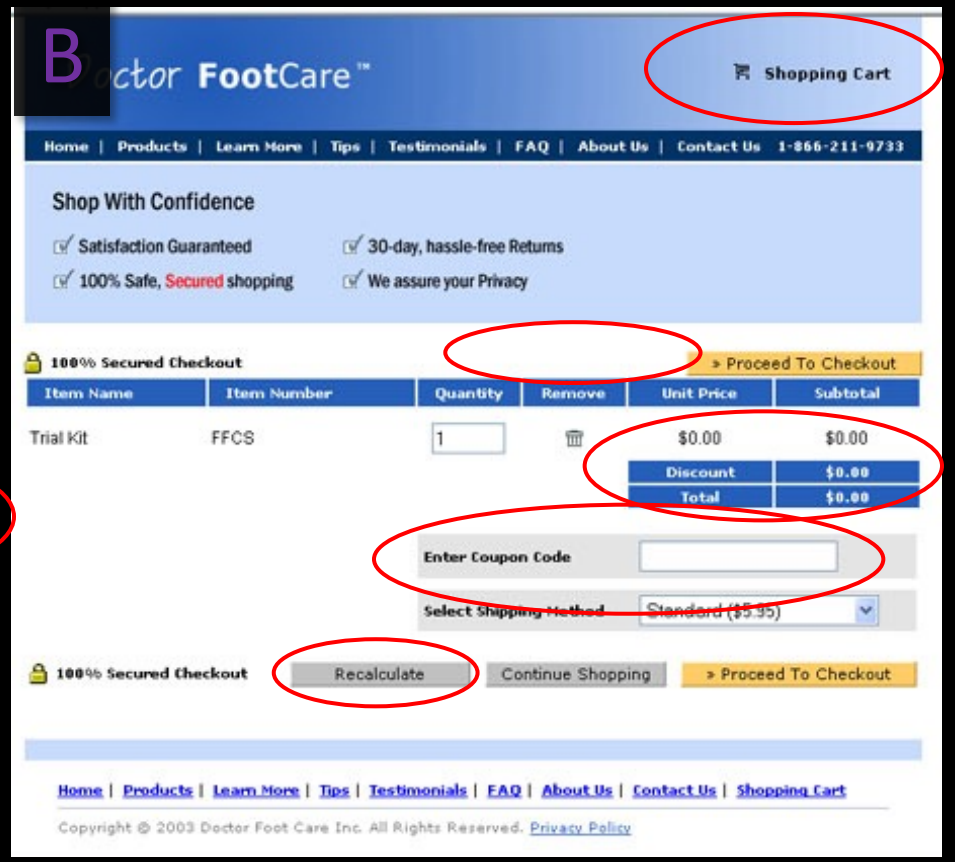
Which version has a higher conversion rate?

Example from Bryan Eisenberg's article on clickz.com

Checkout Page

Conversion rate is percentage of visits that include purchase



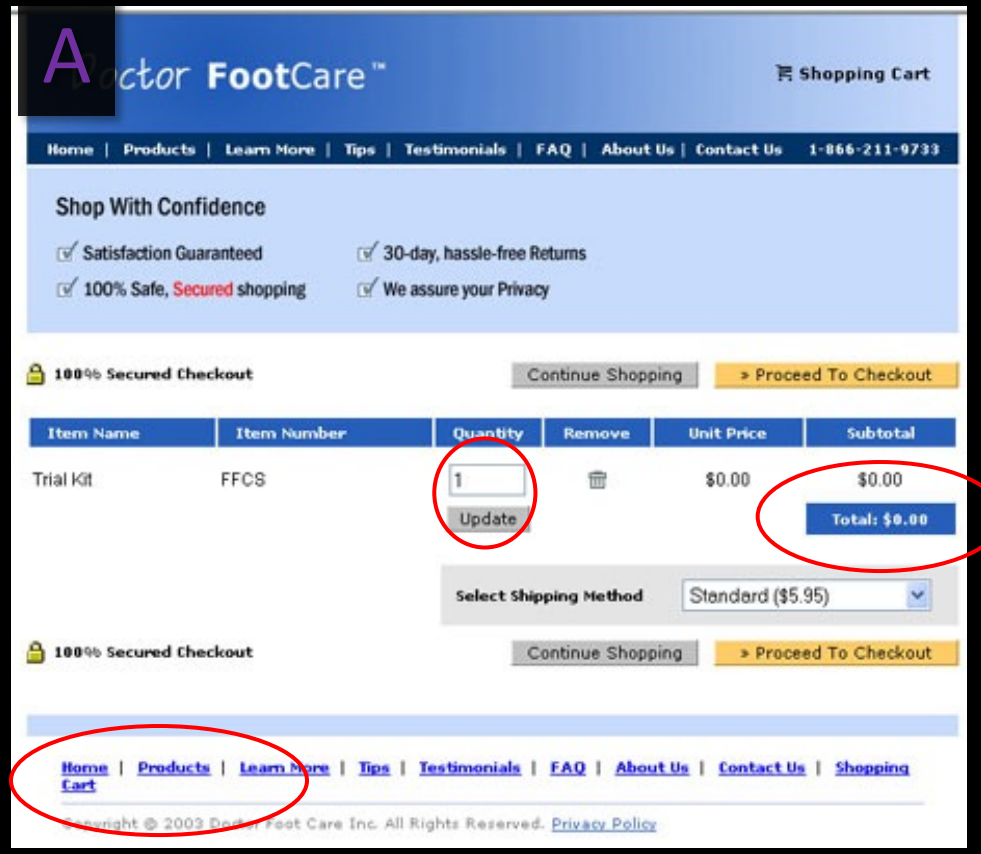


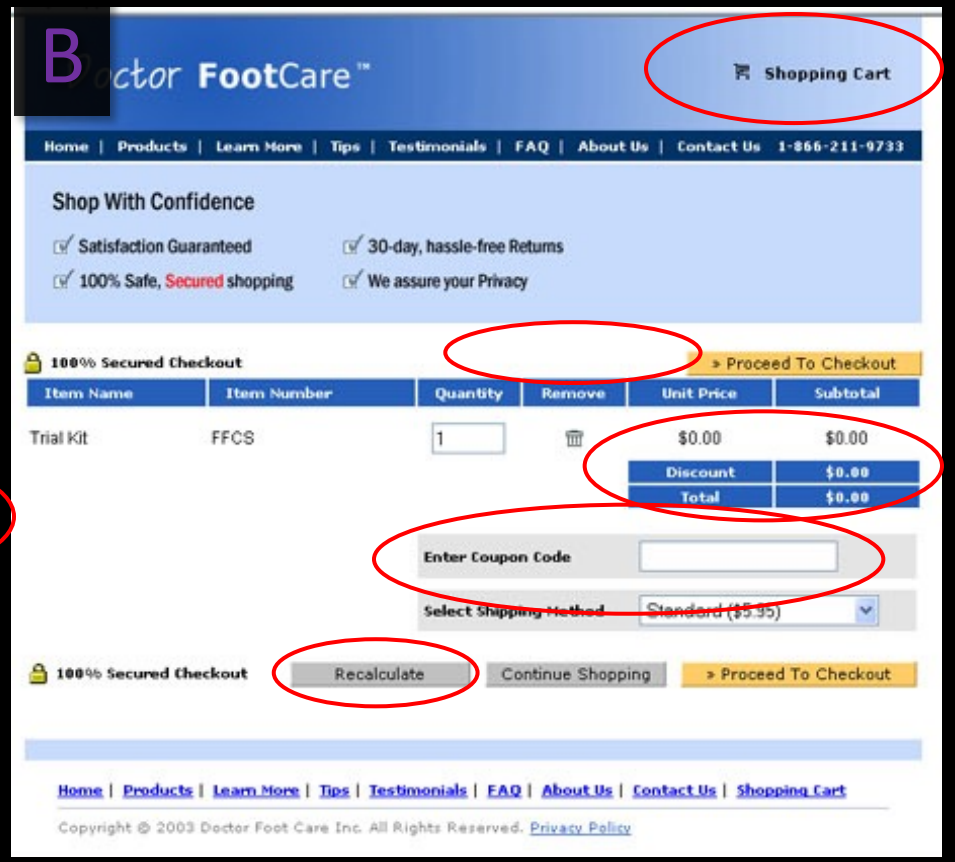
Which version has a higher conversion rate?

Example from Bryan Eisenberg's article on clickz.com

Checkout Page

***Conversion rate* is percentage of visits that include purchase**



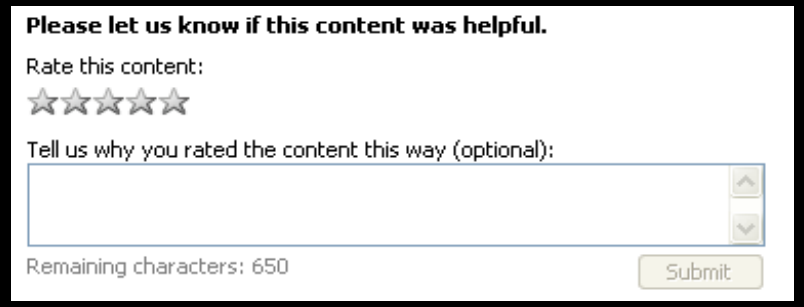


Which version has a higher conversion rate?

Coupon Code decreases by factor of 10

Office Online Feedback

A



B

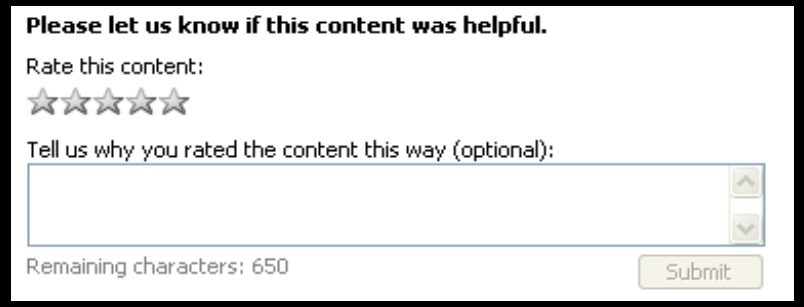
Feedback A puts everything together, whereas feedback B is two-stage: question follows rating.

Feedback A just has 5 stars, whereas B annotates the stars with “Not helpful” to “Very helpful” and makes them brighter.

Which one has a higher response rate? By how much?

Office Online Feedback

A



B

Feedback A puts everything together, whereas feedback B is two-stage: question follows rating.

Feedback A just has 5 stars, whereas B annotates the stars with “Not helpful” to “Very helpful” and makes them brighter.

Which one has a higher response rate? By how much?

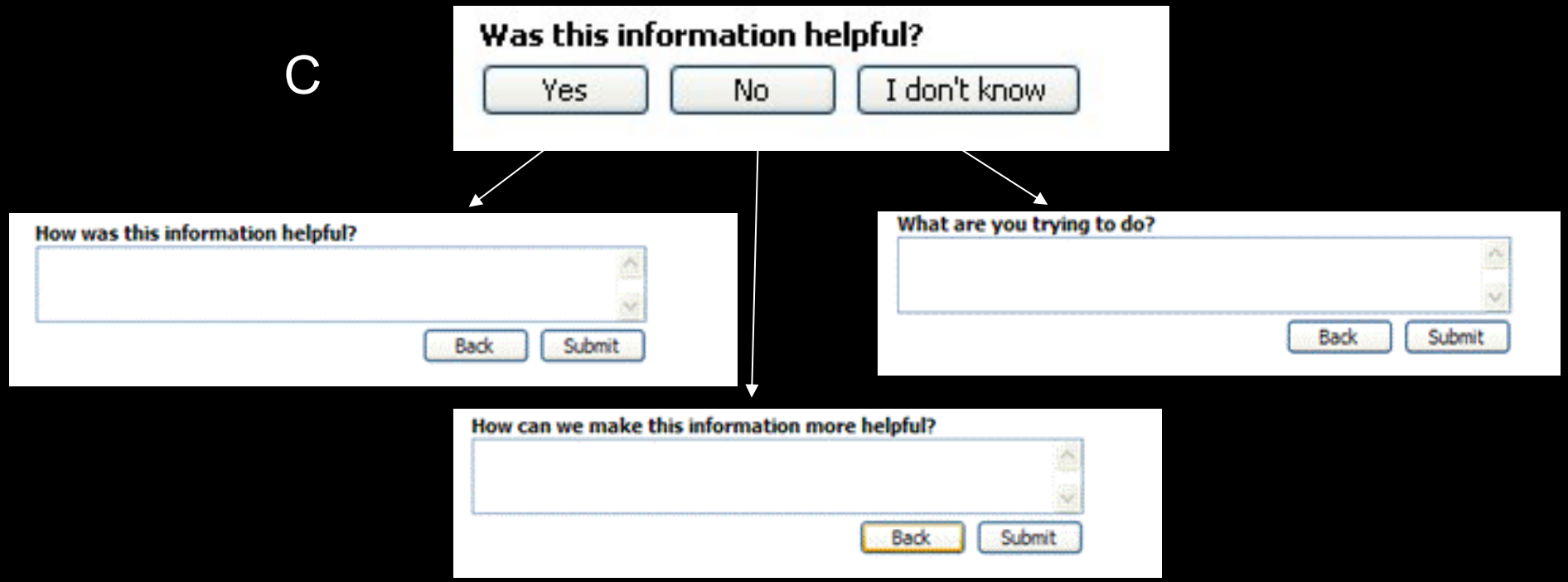
B gets more than double response rate.

Another Feedback Variant

Variant C. Like B, also two-stage.

Which one has a higher response rate, B or C?

C

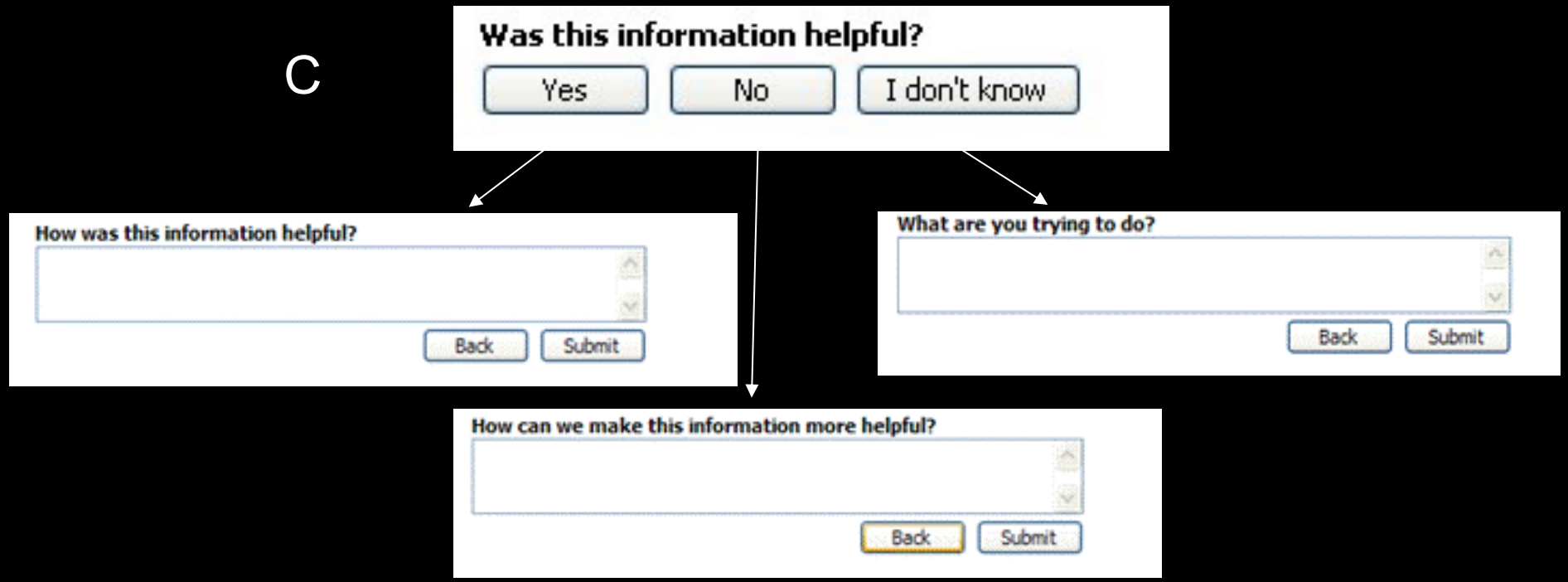


Another Feedback Variant

Variant C. Like B, also two-stage.

Which one has a higher response rate, B or C?

C

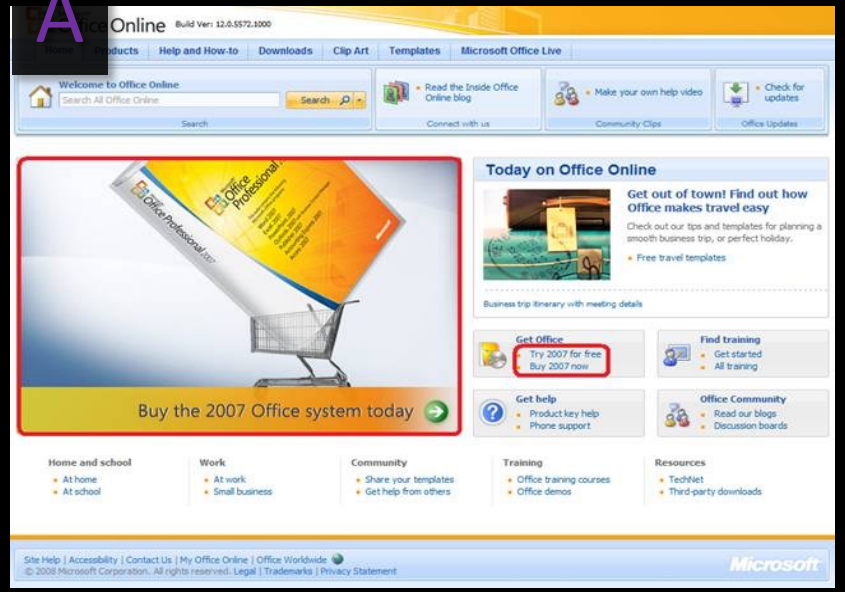


C outperforms B by a factor of 3.5

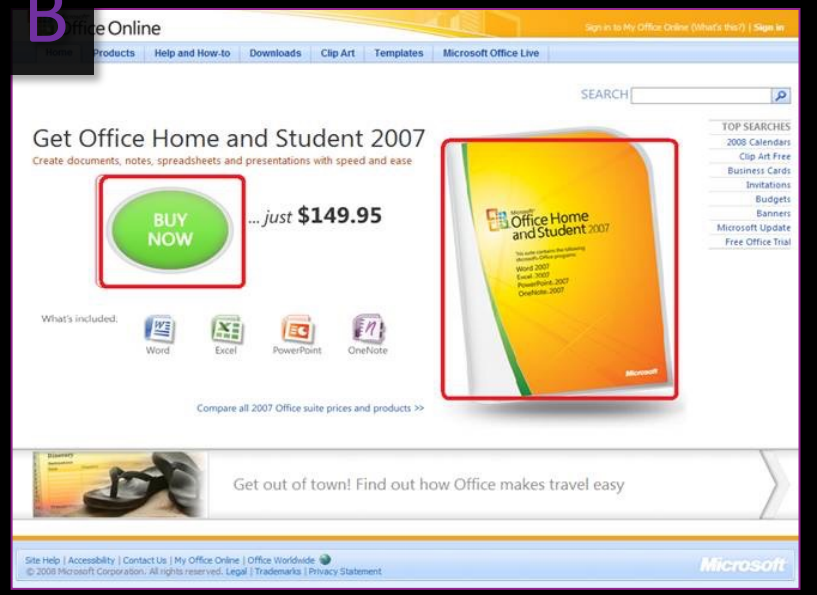
Office Online

Clicks on revenue generating links (red links)

A



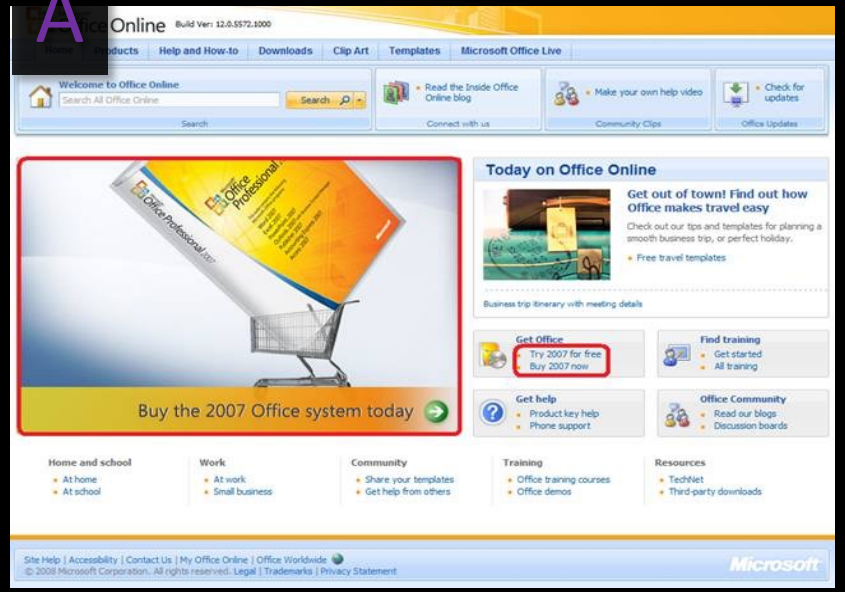
B



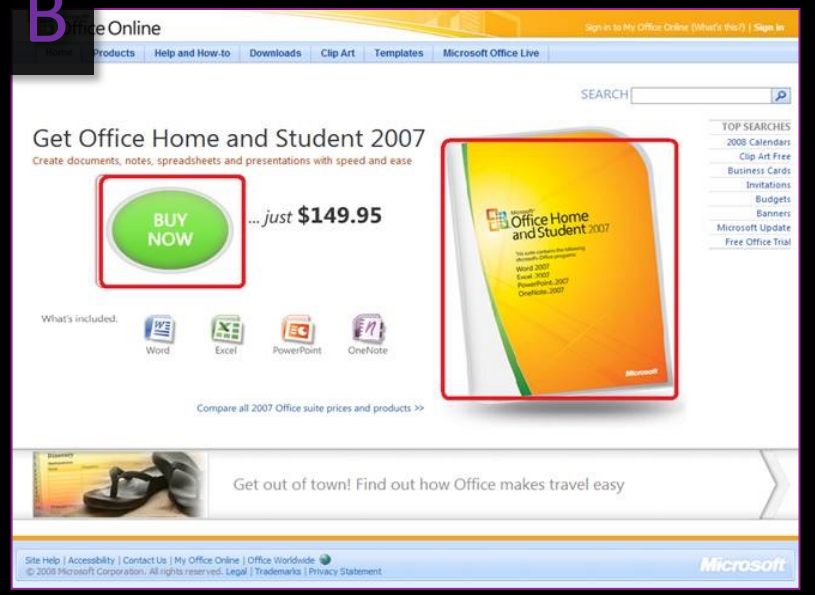
Office Online

Clicks on revenue generating links (red links)

A



B

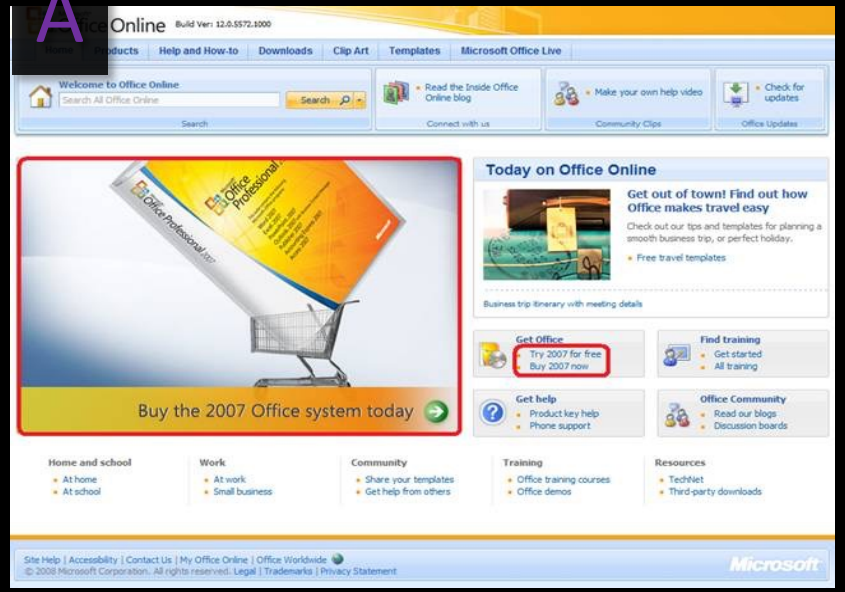


A gets many more clicks

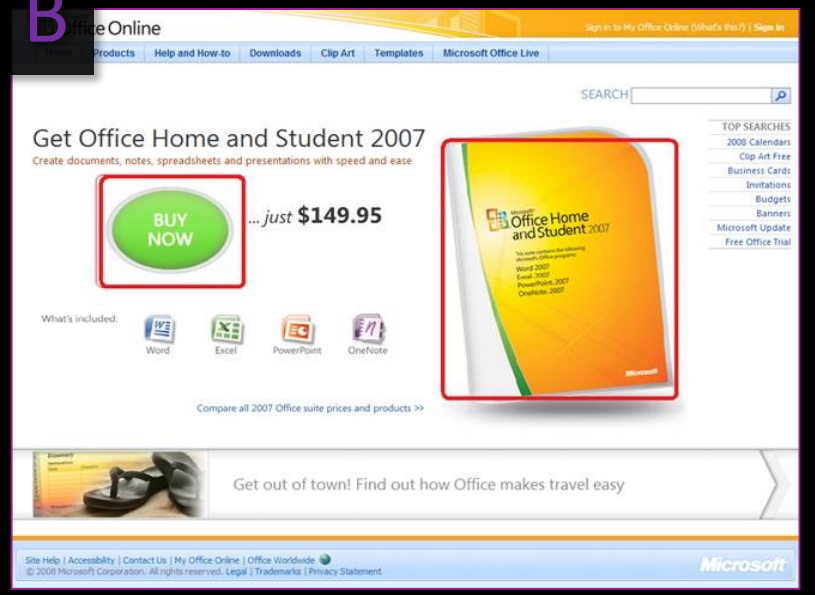
Office Online

Clicks on revenue generating links (red links)

A



B



B gets more revenue

Examples Where Data Is Wrong

If something is “amazing,” find the flaw

If you have a mandatory birth date field,
and people think it’s unnecessary,
you will find lots of 11/11/11 or 01/01/01

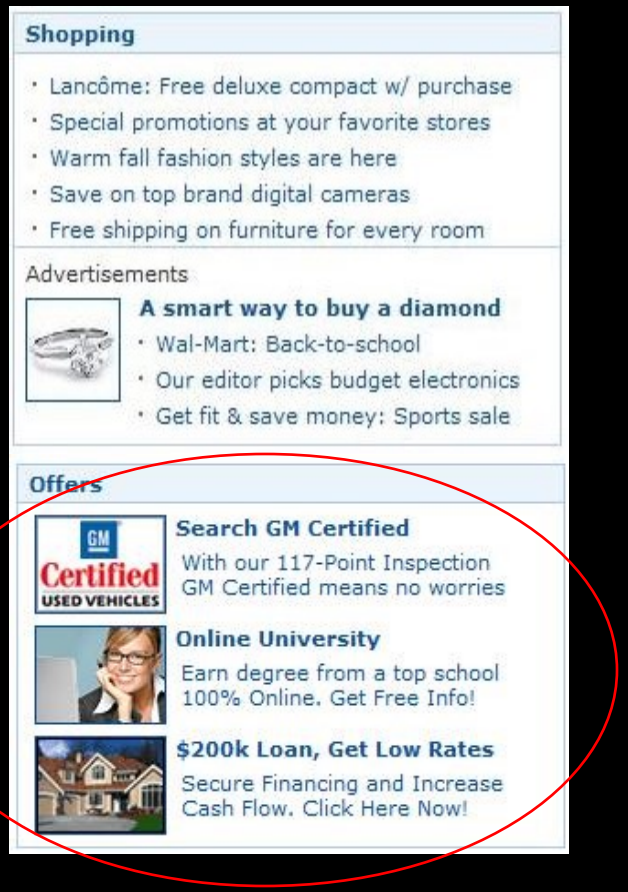
If you have an optional drop down,
do not default to the first alphabetical entry,
or you will have lots of: jobs = Astronaut

Traffic to doubled between 1-2am Nov 6 for many web sites,
relative to same hour week prior

MSN US Home Page

Proposal: New Offers module below Shopping

Control



Treatment

Experiment Results

Ran A/B test for 12 days on 5% of MSN US visitors

Clickthrough: decreased 0.49%

Page views per person-day: decreased 0.35%

Value of click from home page: X cents

Net = Expected Revenue –
Value Per Click * Direct lost clicks –
Value Per Click * Lost Due to Decreased Views

Experiment Results

Ran A/B test for 12 days on 5% of MSN US visitors

Clickthrough: decreased 0.49%

Page views per person-day: decreased 0.35%

Value of click from home page: X cents

Net = Expected Revenue –
Value Per Click * Direct lost clicks –
Value Per Click * Lost Due to Decreased Views

Net was negative (in millions of dollars), offers module did not launch

Limitations of Data Driven Testing

Drives hill-climbing, but not overall design

- A design may be better, but is it good?

Can be difficult to scale to many features

Impossible for new designs to compete

- The new design may have the potential to be much better, but is not yet "tuned" by this same process

Can lead to optimizing what is measurable rather than what is important

Today

Controlled A/B Experiments

Patterns

Case Study of Model-View-Controller

Case Study of Animation

Role of Interface Tools

Design Equals Solutions

Design is about finding solutions

Designers often reinvent

- Hard to know how things were done before
- Why things were done a certain way
- How to reuse solutions

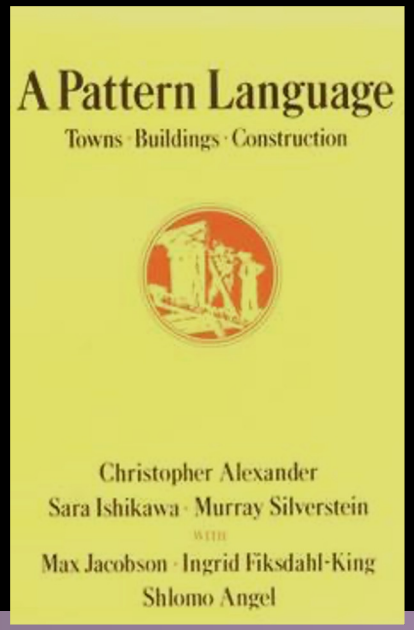
One option is patterns

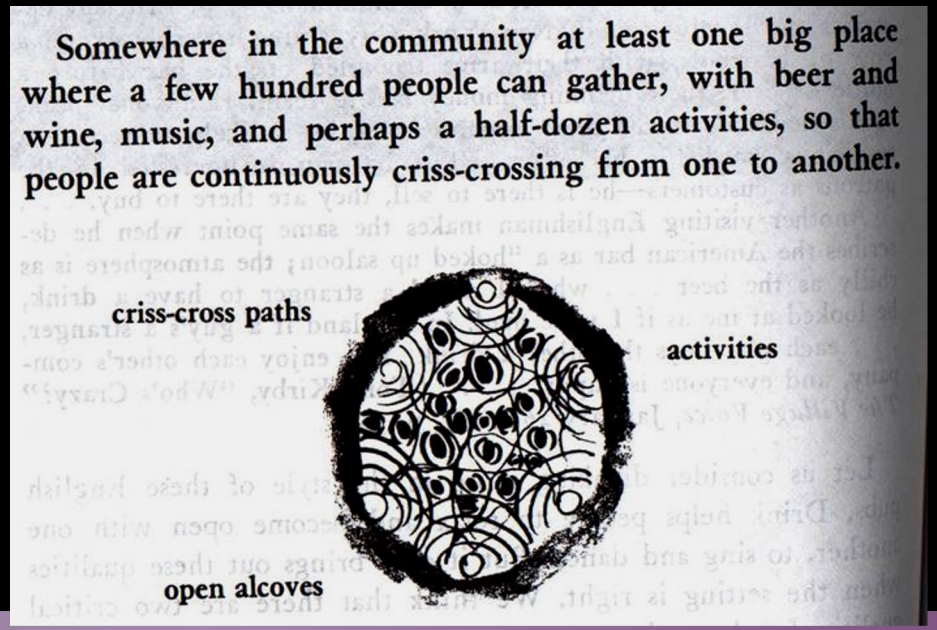
- You can look to models, you can look to prior research
- Patterns can also help you look to prior solutions

Design Patterns

Design patterns communicate common design problems and solutions

- First used in architecture
- How to create a beer hall where people socialize?





Using Design Patterns

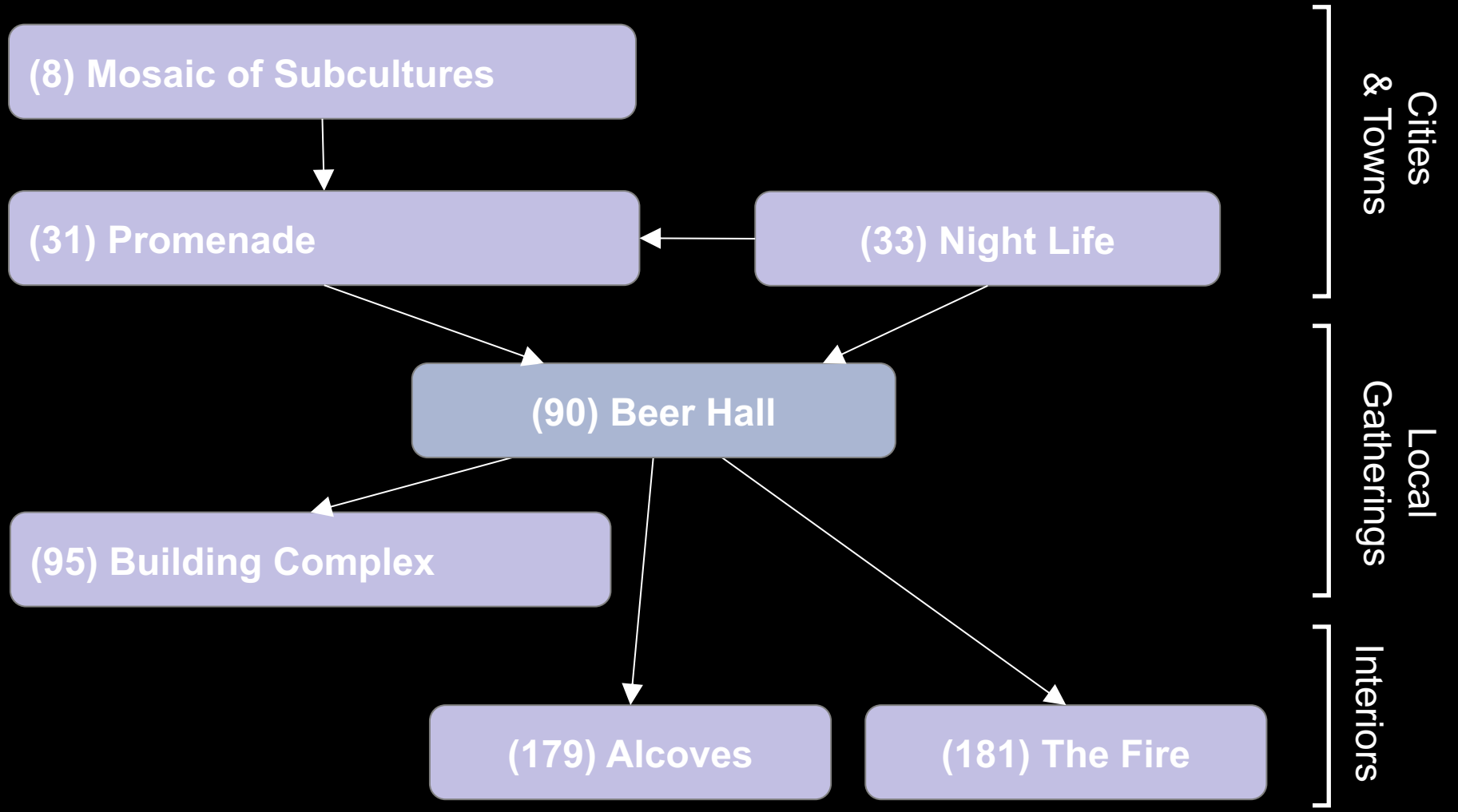
Not too general and not too specific

- use a solution “a million times over, without ever doing it the same way twice”

Design patterns are a shared language

- for “building and planning towns, neighborhoods, houses, gardens, and rooms”
  - Beer hall is part of a center for public life
  - Beer hall needs spaces for groups to be alone
  - ALCOVES

A Web of Design Patterns



Web Design Patterns

Communicate design problems & solutions

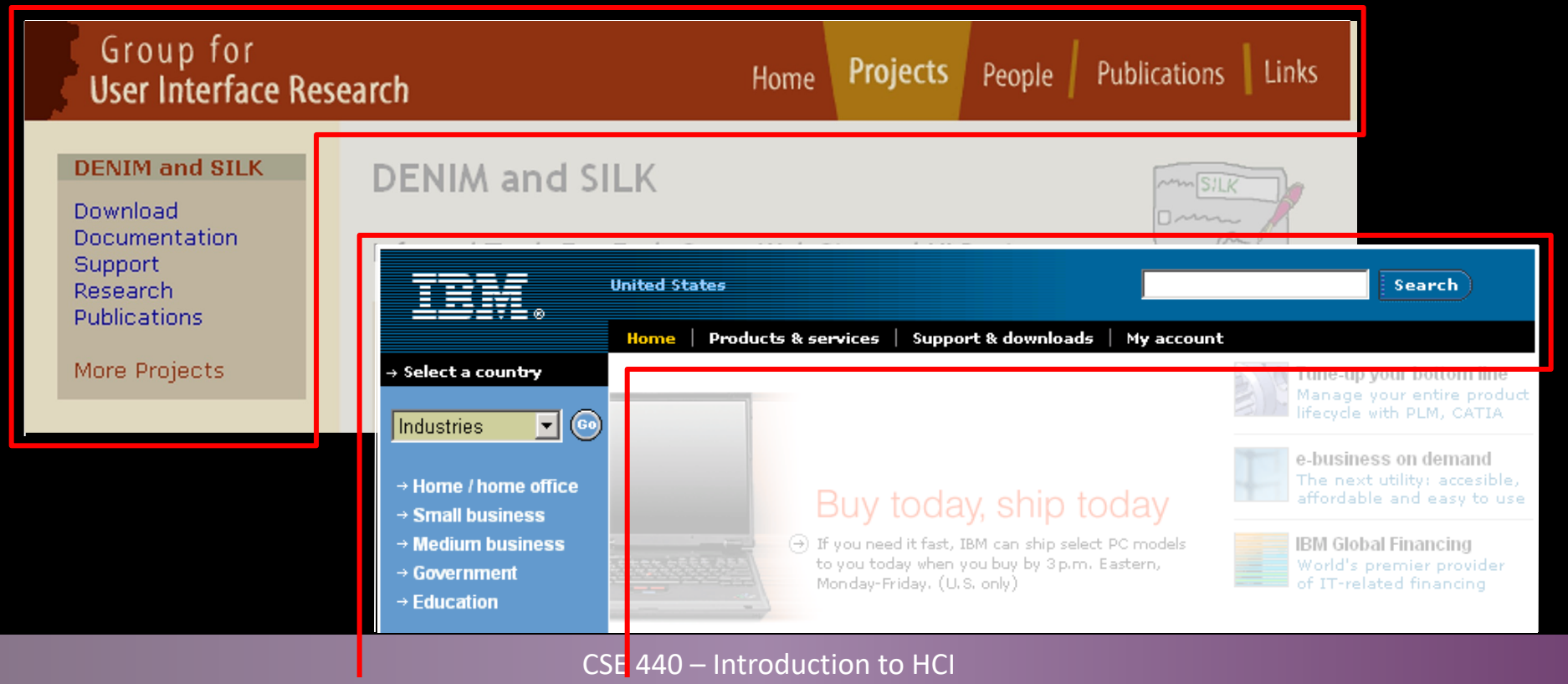
how to create navigation bars for finding relevant content

how to create a shopping cart that supports check out

how to make e-commerce sites where people return & buy

NAVIGATION BAR (K2)

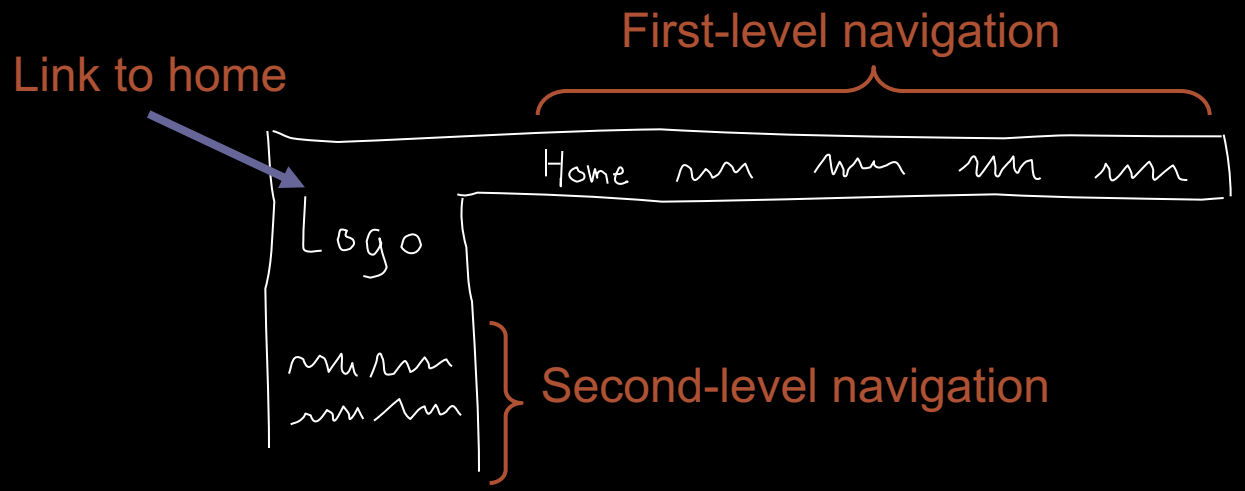
Problem: Customers need a structured, organized way of finding the most important parts of your Web site



NAVIGATION BAR (K2)

Solution diagram

Captures essence of how to solve problem



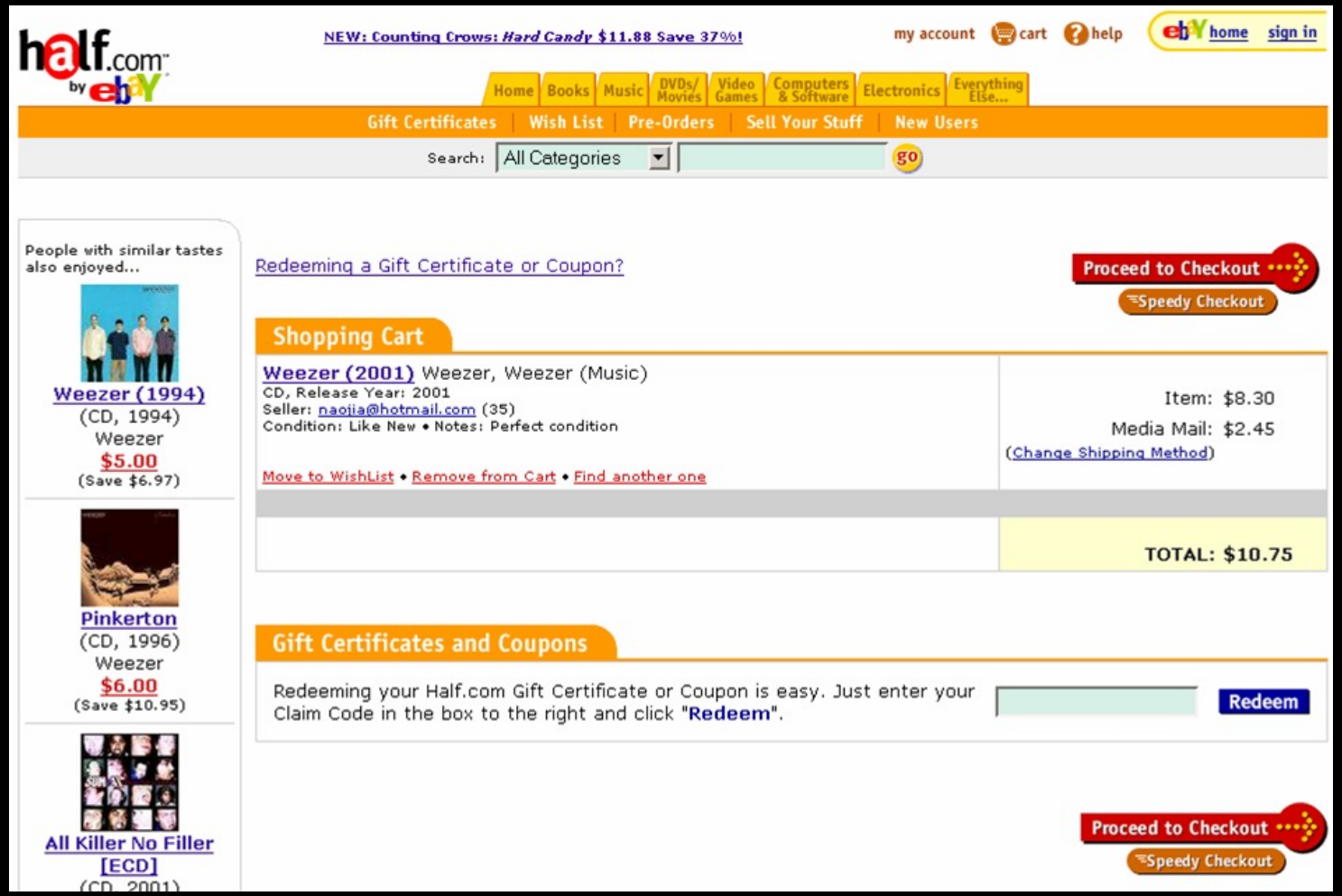
PROCESS FUNNEL (H1)

Problem:

Need a way to help people
complete highly specific stepwise tasks

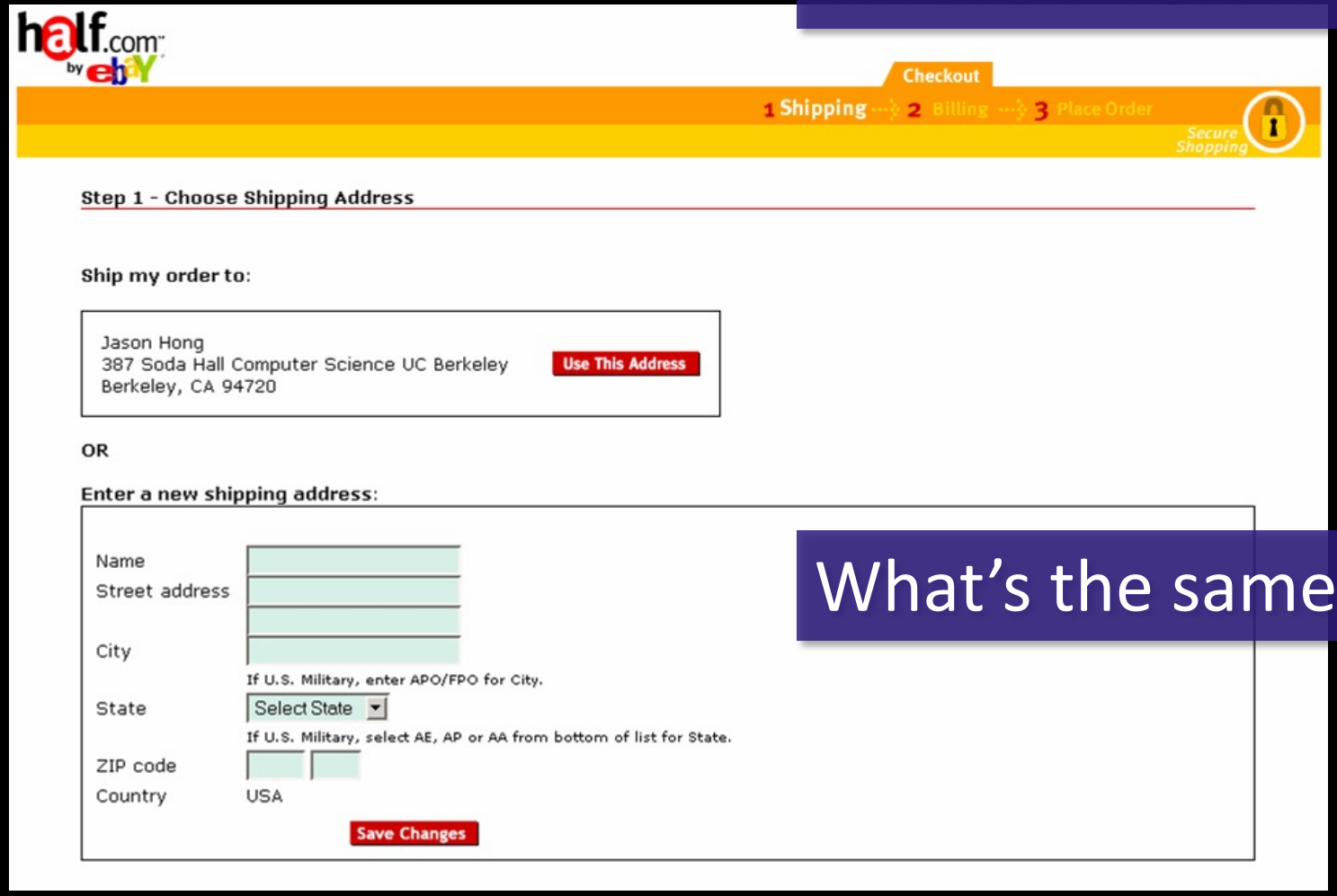
- Ex. Create a new account
- Ex. Fill out survey forms
- Ex. Check out

PROCESS FUNNEL (H1)



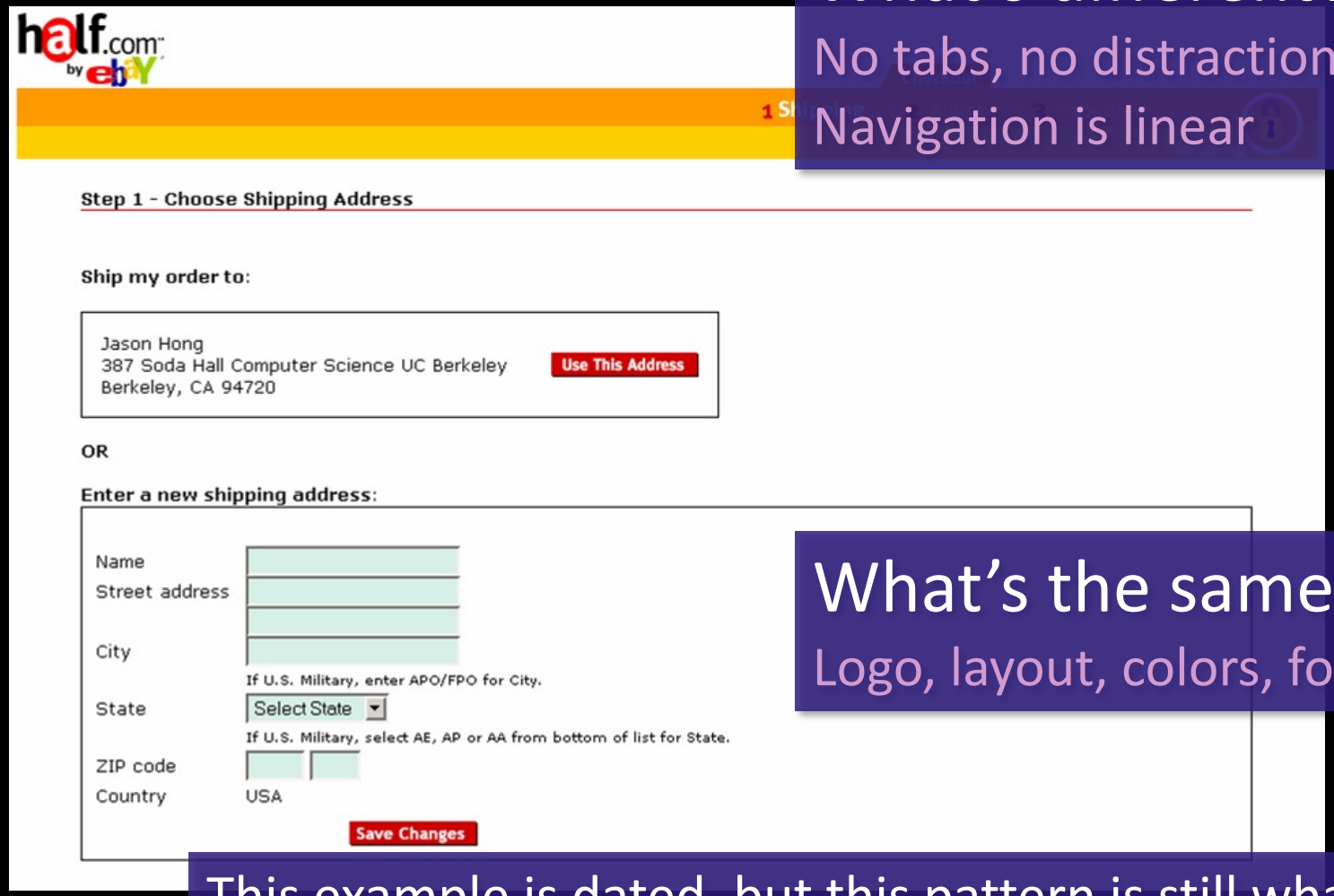
PROCESS FUNNEL (H1)

What's different?



What's the same?

PROCESS FUNNEL (H1)



What's different?

No tabs, no distractions

Navigation is linear

What's the same?

Logo, layout, colors, fonts

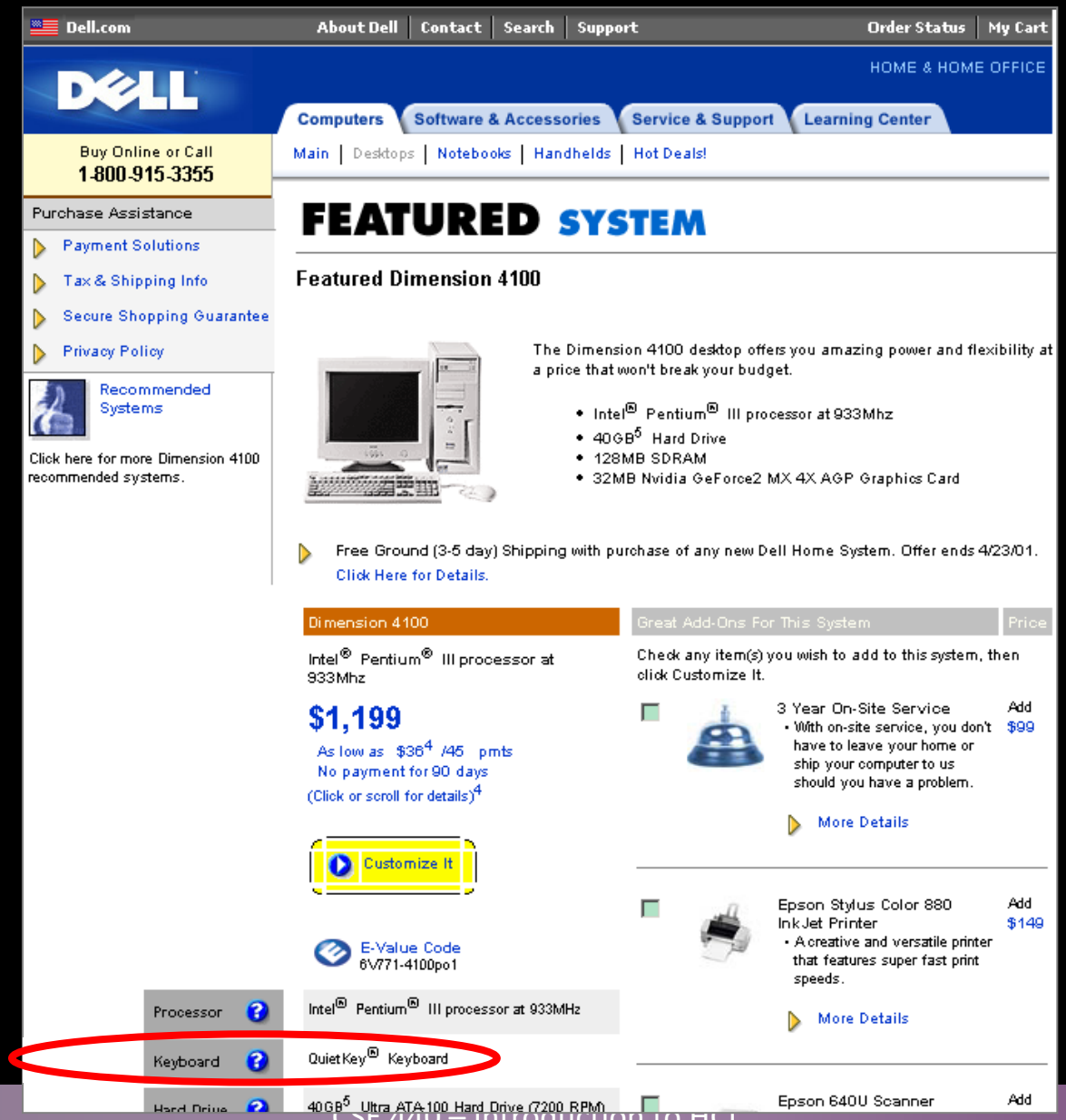
This example is dated, but this pattern is still what you encounter

PROCESS FUNNEL (H1)

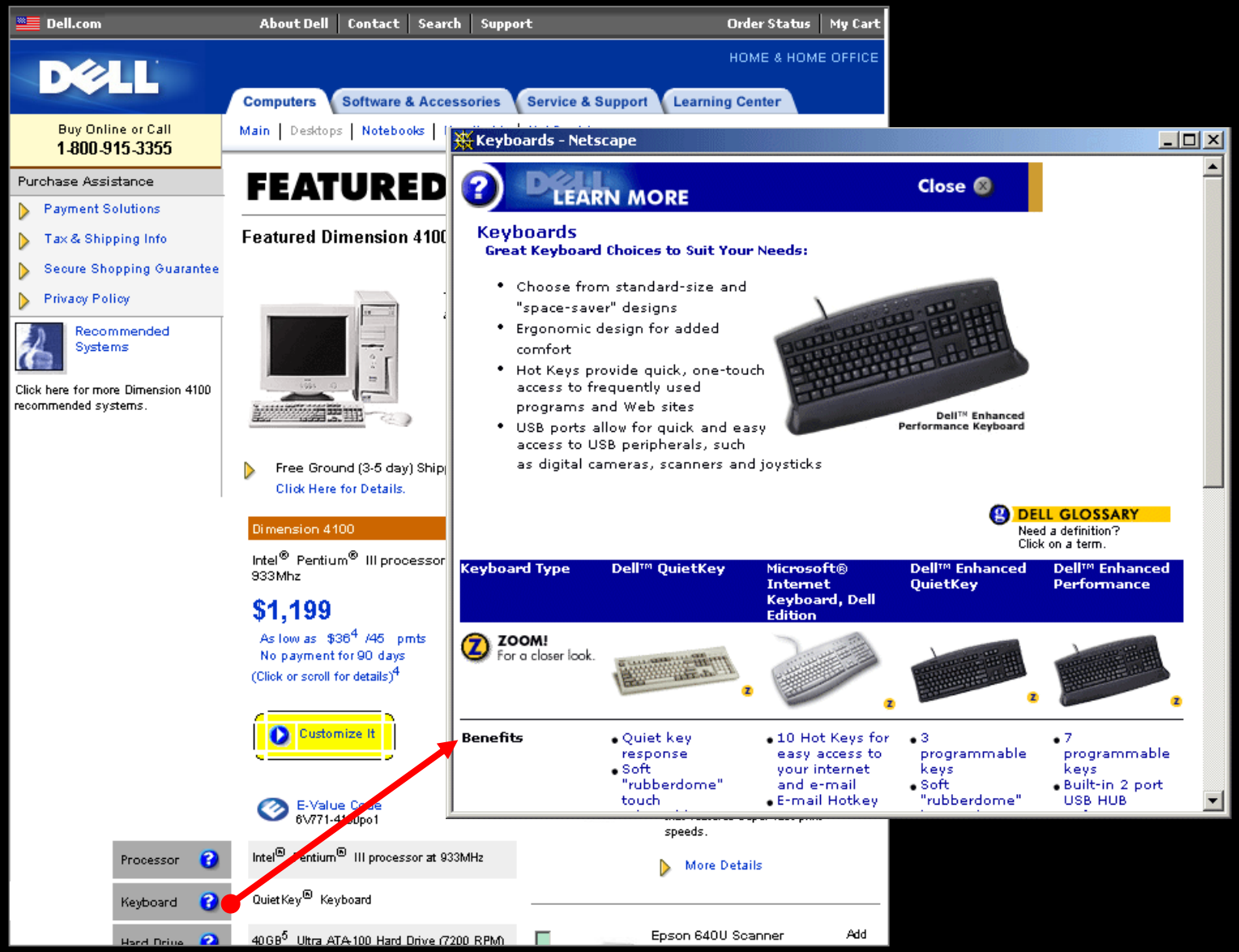
Problem:

What if people need extra help?

PROCESS FUNNEL (H1)

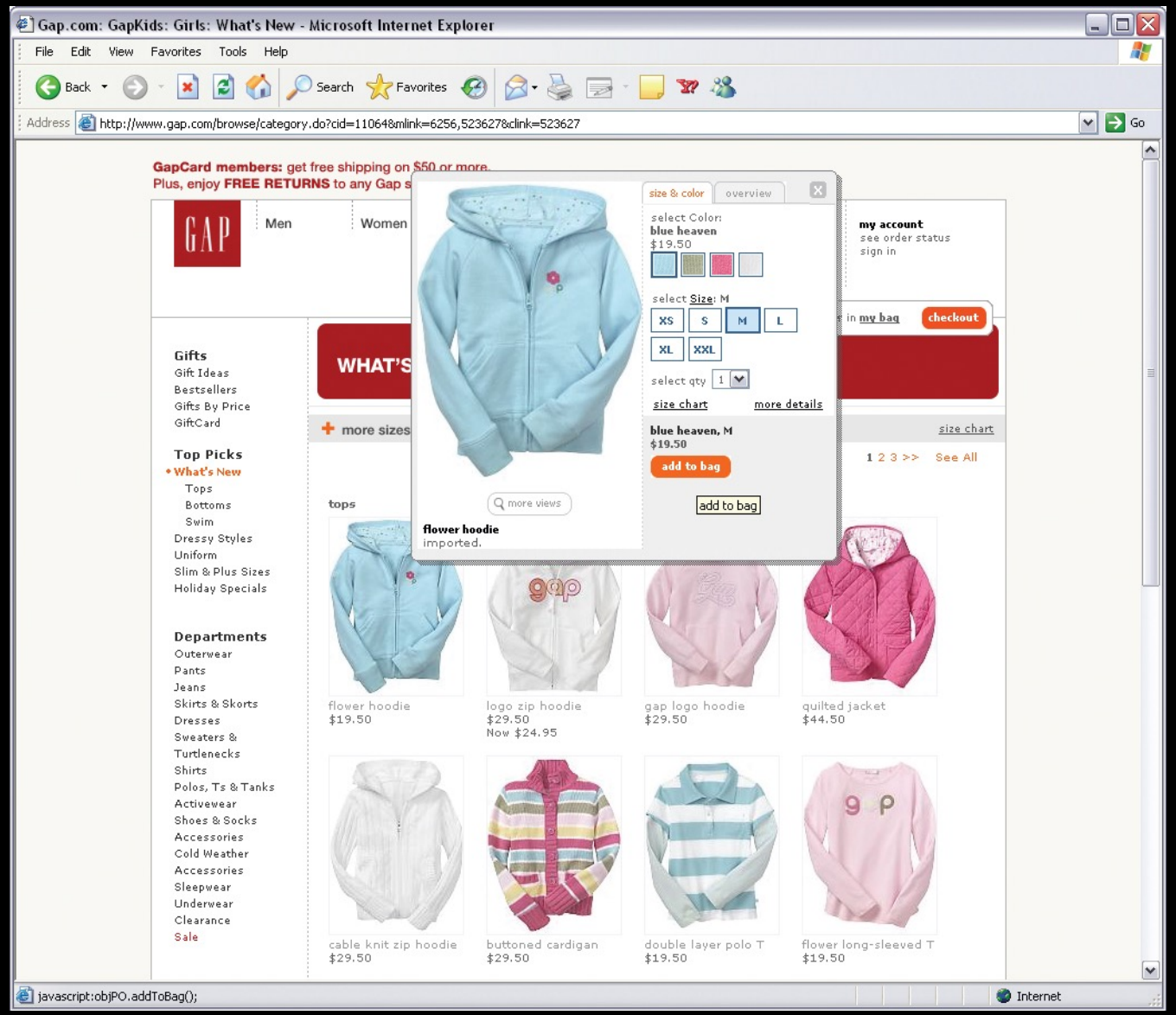


CONTEXT-SENSITIVE HELP (H8)



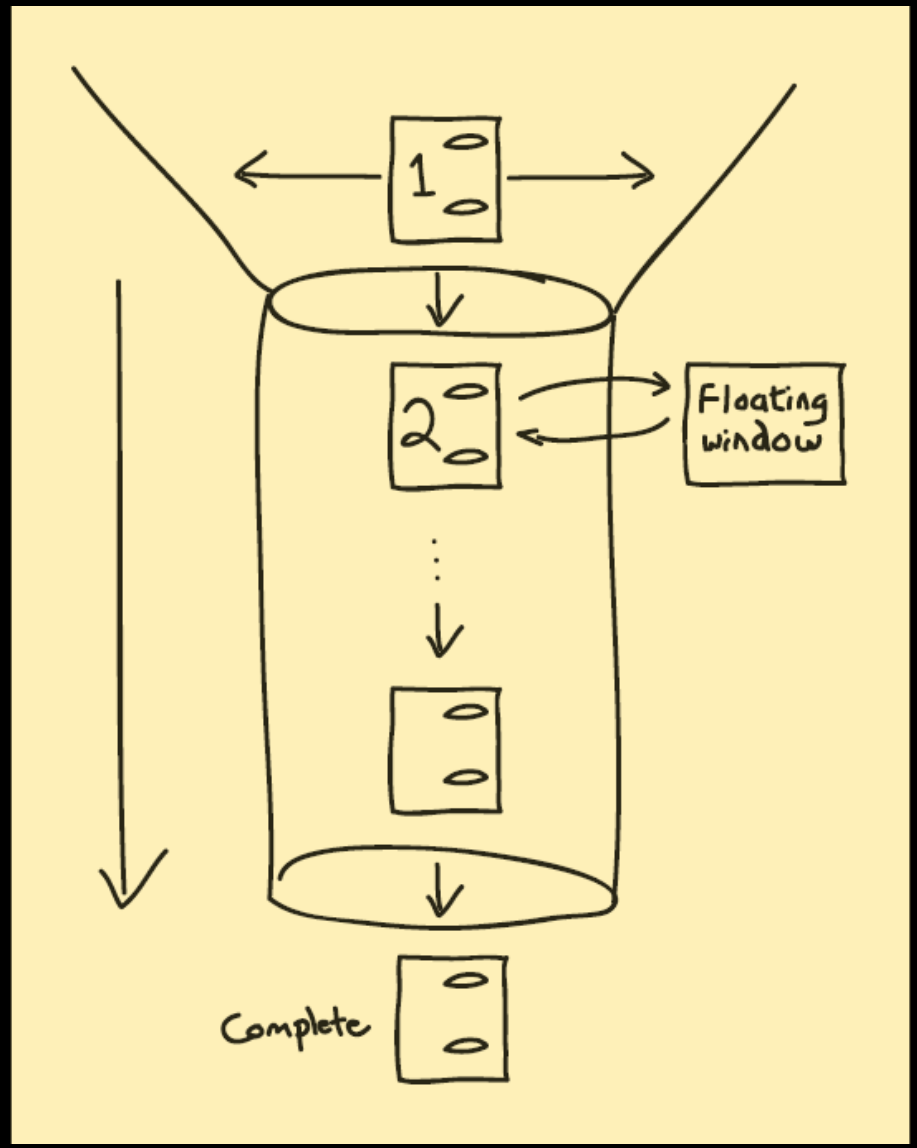
FLOATING WINDOWS (H6)

FLOATING WINDOWS (H6)

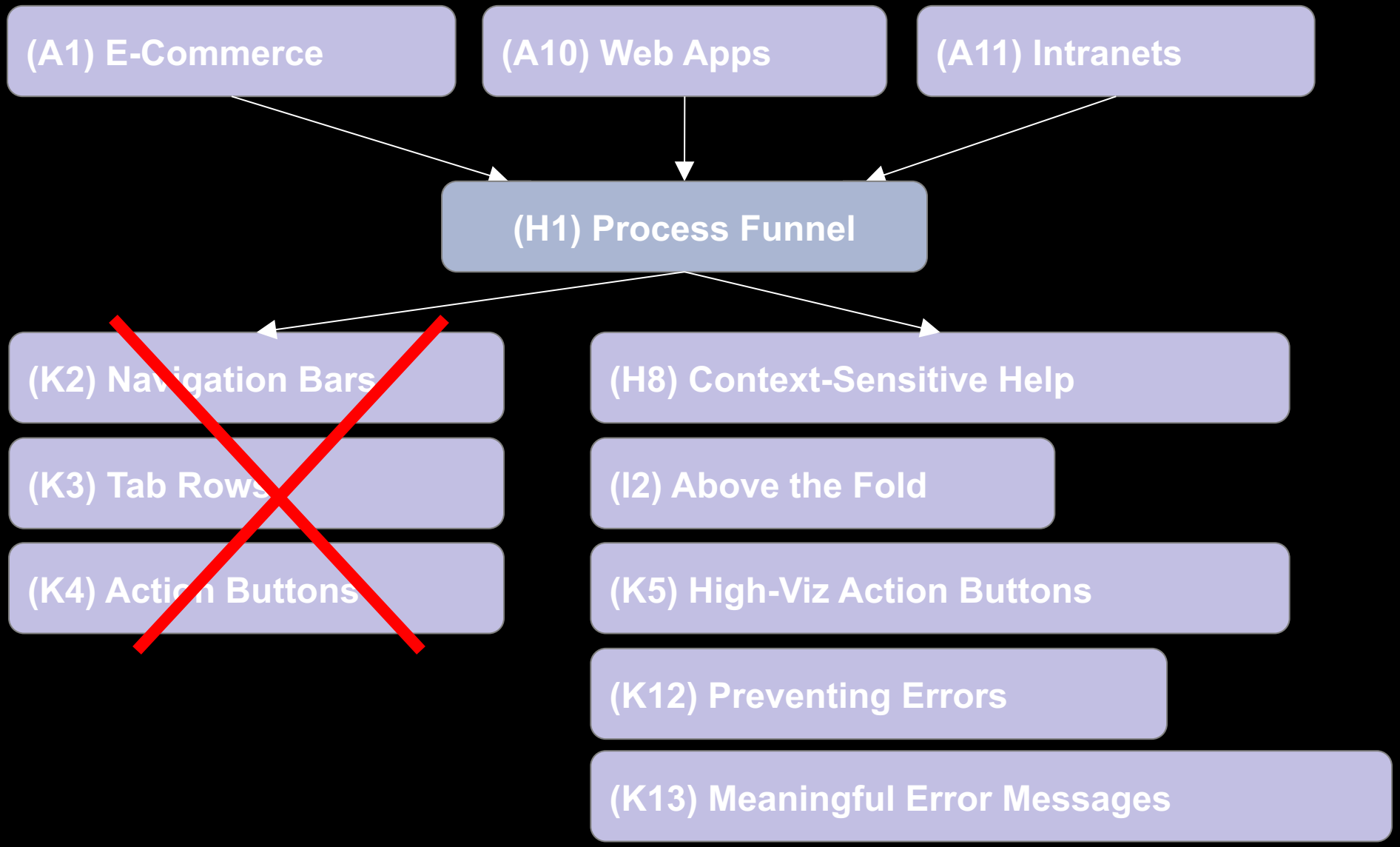


PROCESS FUNNEL (H1)

Solution Diagram



Related Patterns



Patterns Support Creativity

Patterns come from successful examples

- sites that are so successful that lots of people are now familiar
- designs that work well across many sites (e.g., shopping carts)

Not too general and not too specific

- you need to specialize to your needs

Patterns let you focus on the hard,
unique problems of your design situation

Principles, Guidelines, Templates

Patterns help design without over-constraining

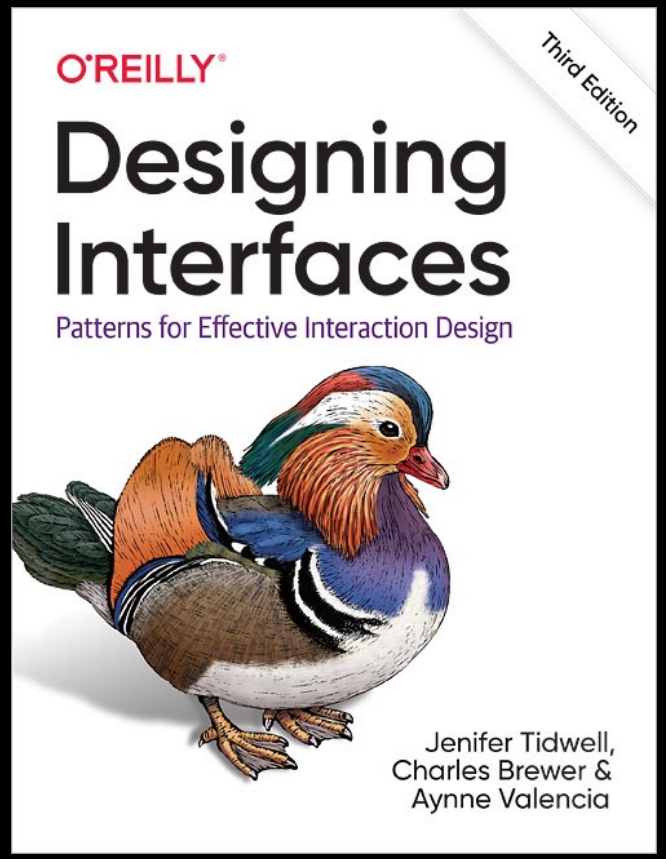
- unlike principles, patterns are not too general
- unlike guidelines, patterns discuss tradeoffs,
  show good examples, and tie to other patterns
- unlike style guides, patterns not too specific,
  can then be specialized to a design
- unlike templates,
  patterns illustrate flows and relationships among different elements,
  guide your understanding, not just instantiate a single instance

Additional Reading

Classic collection
of web-focused patterns

Additional Reading

Both have some online resources



UI Patterns
User Interface Design Pattern Library

Patterns

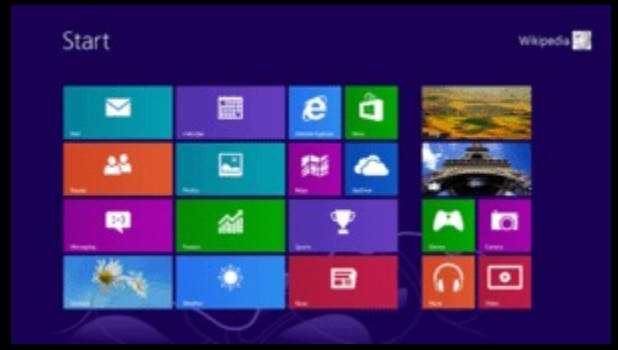
When you see advice, consider its depth

- Result of an individual study / rant
- Pre-pattern based on some meta-analysis
- Established pattern

Be aware of misapplying patterns

- And be aware of anti-patterns

Touch and Microsoft Windows



Microsoft Windows xp Tablet PC Edition

2004

2012

Consistency vs. Specialization

Beware of simply copying a design language

- Consistency is your friend
  until is it not your friend

Not limited to platform-level decisions

- One “look” for your app
- Or targeted at each device

Deceptive Design Patterns

A Deceptive Design Pattern is an interface that has been carefully crafted to trick people into doing things, such as buying insurance with their purchase or signing up for recurring bills.

(aka Dark Pattern, but shifting away from that language)

Disguised Ads

Ads that are disguised as other kinds of content or navigation,
in order to get people to click on them

Friend Spam

A site or game asks for your credentials,
then goes on to publish content or send out bulk messages

Today

Controlled A/B Experiments

Patterns

Case Study of Model-View-Controller

Case Study of Animation

Role of Interface Tools

Model-View-Controller

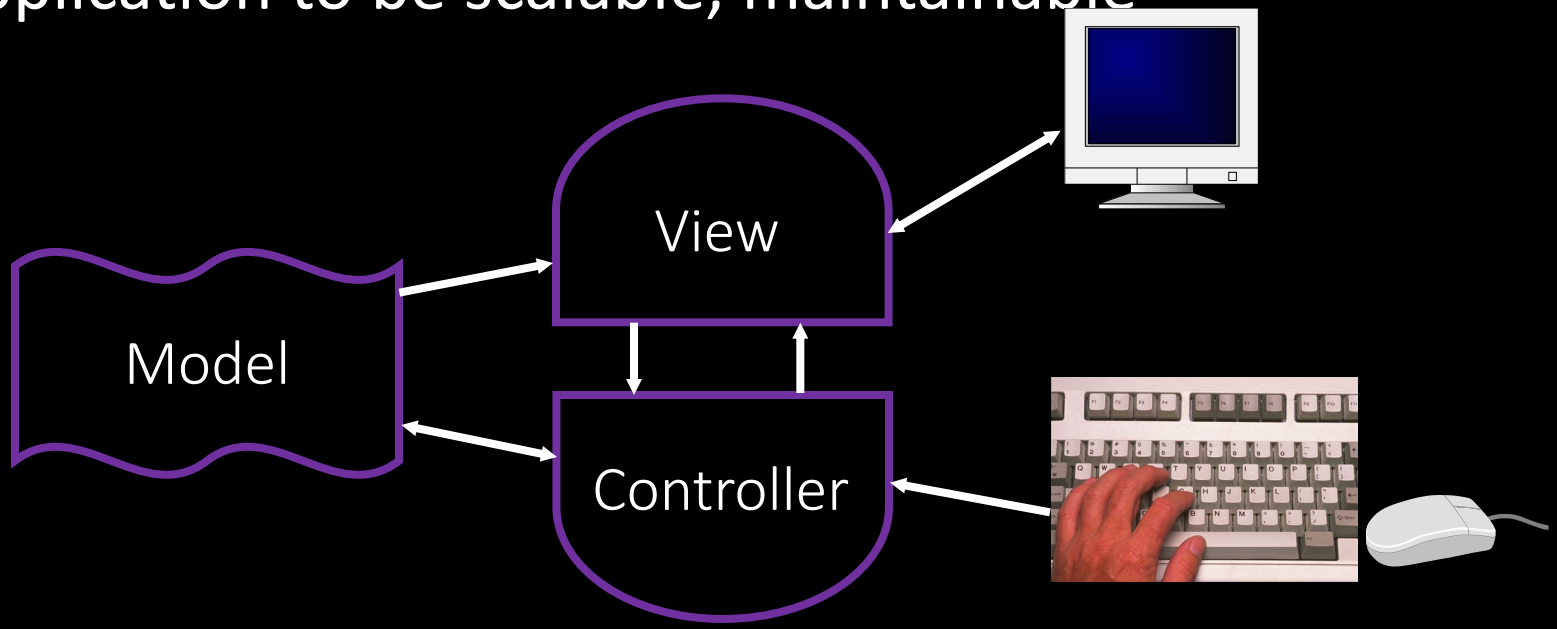
How to organize the code of an interface?

This is a surprisingly complicated question,
with unstated assumptions requiring significant background to understand and resolve

Model-View-Controller

Introduced by Smalltalk developers at PARC

Partitions application to be scalable, maintainable



A pattern!

View / Controller Relationship

In theory:

Pattern of behavior in response to input events

(i.e., concerns of the controller)

are independent of visual geometry

(i.e., concerns of the view)

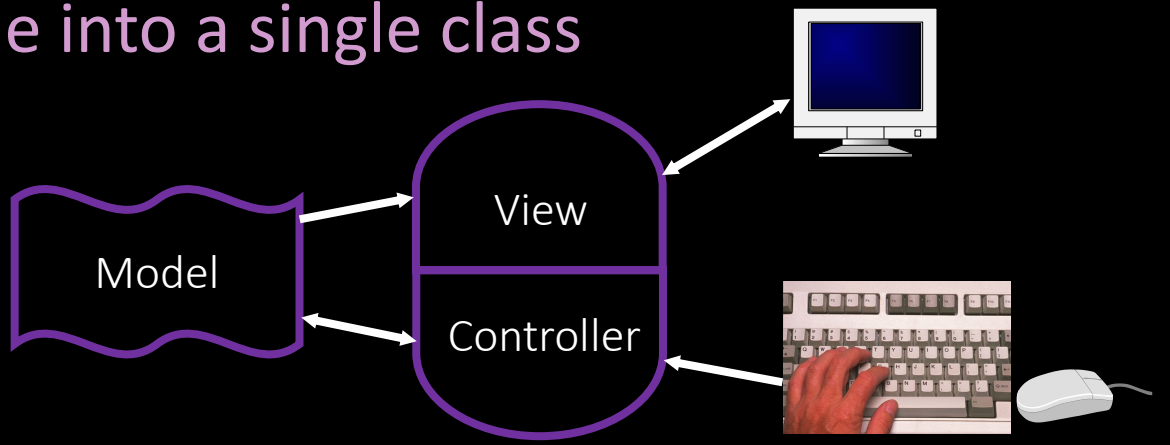
Controller contacts view to interpret what input events mean in context of a view (e.g., selection)

View / Controller Relationship

In practice:

View and controller often tightly intertwined,
almost always occur in matched pairs

Many architectures combine into a single class



Model-View-Controller

MVC separates concerns and scales better than global variables or putting everything together

Separation eases maintenance

- Can add new fields to model,
  new views can leverage, old views will still work
- Can replace model without changing views

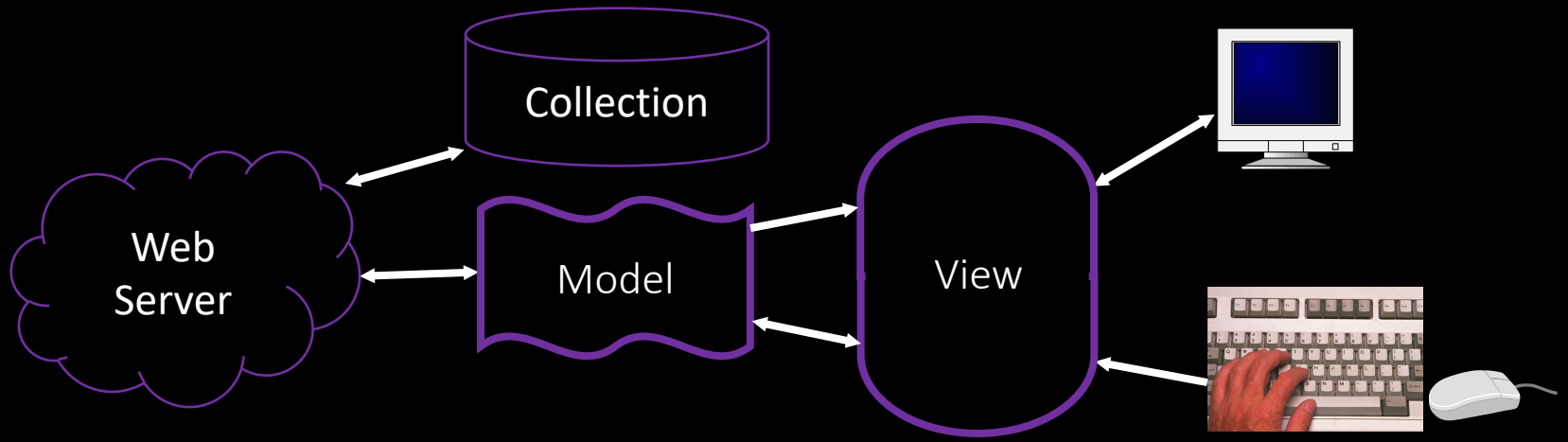
Separation of “business logic” can require care

- May help to think of model as the client model

Model-View-Collection on the Web

Core ideas manifest differently according to needs

- For example, backbone.js implements client views of models, with REST API calls to web server
- Web tools often implement views as templates

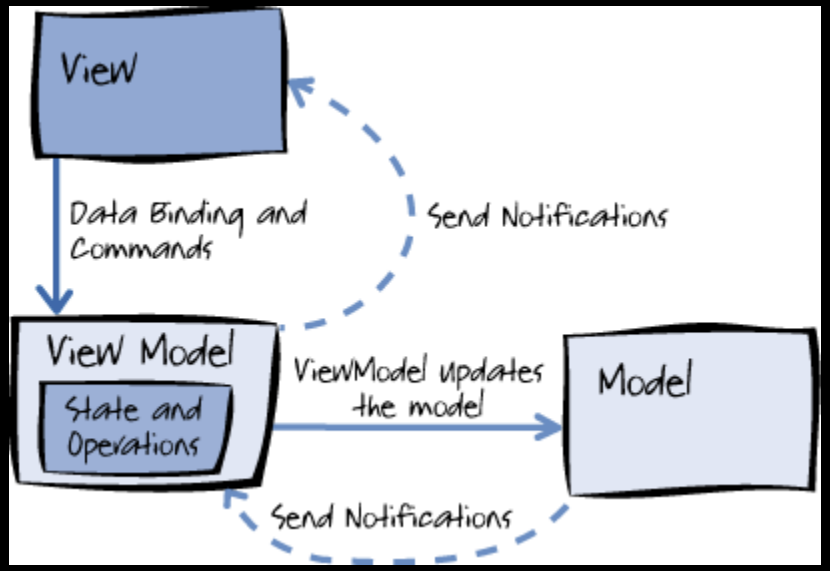


An instantiation of the pattern, adapted to the needs of the web

Model View View-Model

Design to support data-binding
by minimizing functionality in view

Also allows greater separation of expertise



An instantiation of the pattern, adapted to the needs of design tools

Today

Controlled A/B Experiments

Patterns

Case Study of Model-View-Controller

Case Study of Animation

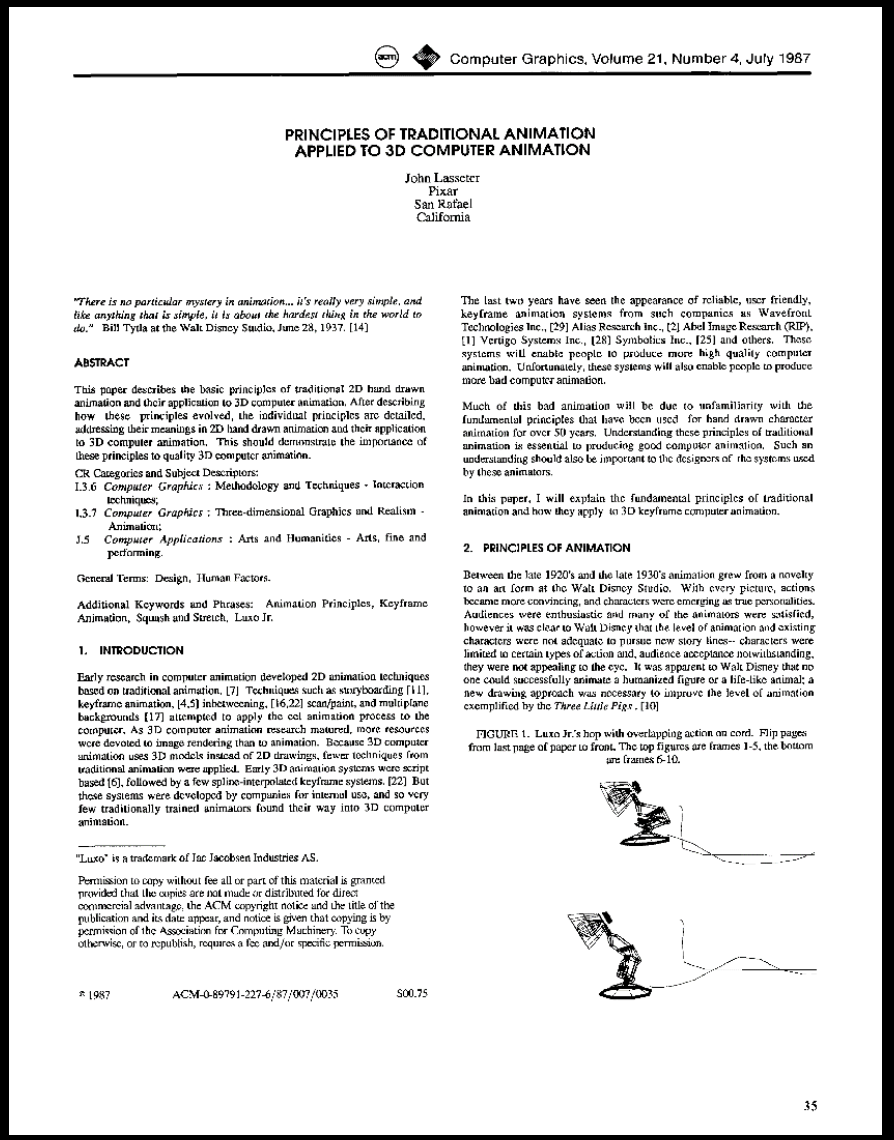
Role of Interface Tools

Animation Case Study

Principles of Traditional Animation Applied to 3D Computer Animation

Lasseter, 1987

http://dx.doi.org/10.1145/37402.37407



PRINCIPLES OF TRADITIONAL ANIMATION APPLIED TO 3D COMPUTER ANIMATION

John Lasseter
Pixar
San Rafael
California

"There is no particular mystery in animation... it's really very simple, and like anything that is simple, it is about the hardest thing in the world to do." Bill Tytla at the Walt Disney Studio, June 28, 1937. [14]

ABSTRACT

This paper describes the basic principles of traditional 2D hand drawn animation and their application to 3D computer animation. After describing how these principles evolved, the individual principles are detailed, addressing their meanings in 2D hand drawn animation and their application to 3D computer animation. This should demonstrate the importance of these principles to quality 3D computer animation.

CR Categories and Subject Descriptors:
I.3.6 *Computer Graphics* : Methodology and Techniques - Interaction techniques;
I.3.7 *Computer Graphics* : Three-dimensional Graphics and Realism - Animation;
J.5 *Computer Applications* : Arts and Humanities - Arts, fine and performing.

General Terms: Design, Human Factors.

Additional Keywords and Phrases: Animation Principles, Keyframe Animation, Squash and Stretch, Luxo Jr.

1. INTRODUCTION

Early research in computer animation developed 2D animation techniques based on traditional animation. [7] Techniques such as storyboarding [11], keyframe animation, [4,5] inbetweening, [16,22] scan/paint, and multiplane backgrounds [17] attempted to apply the cel animation process to the computer. As 3D computer animation research matured, more resources were devoted to image rendering than to animation. Because 3D computer animation uses 3D models instead of 2D drawings, fewer techniques from traditional animation were applied. Early 3D animation systems were script based [6], followed by a few spline-interpolated keyframe systems. [22] But those systems were developed by companies for internal use, and so very few traditionally trained animators found their way into 3D computer animation.

"Luxo" is a trademark of Jac Jacobsen Industries AS.

ACM-0-89791-227-6/87/007/0035 $00.75

The last two years have seen the appearance of reliable, user friendly, keyframe animation systems from such companies as Wavefront Technologies Inc., [29] Alias Research Inc., [2] Abel Image Research (RIP), [1] Vertigo Systems Inc., [28] Symbolica Inc., [25] and others. These systems will enable people to produce more high quality computer animation. Unfortunately, these systems will also enable people to produce more bad computer animation.

Much of this bad animation will be due to unfamiliarity with the fundamental principles that have been used for hand drawn character animation for over 50 years. Understanding these principles of traditional animation is essential to producing good computer animation. Such an understanding should also be important to the designers of the systems used by these animators.

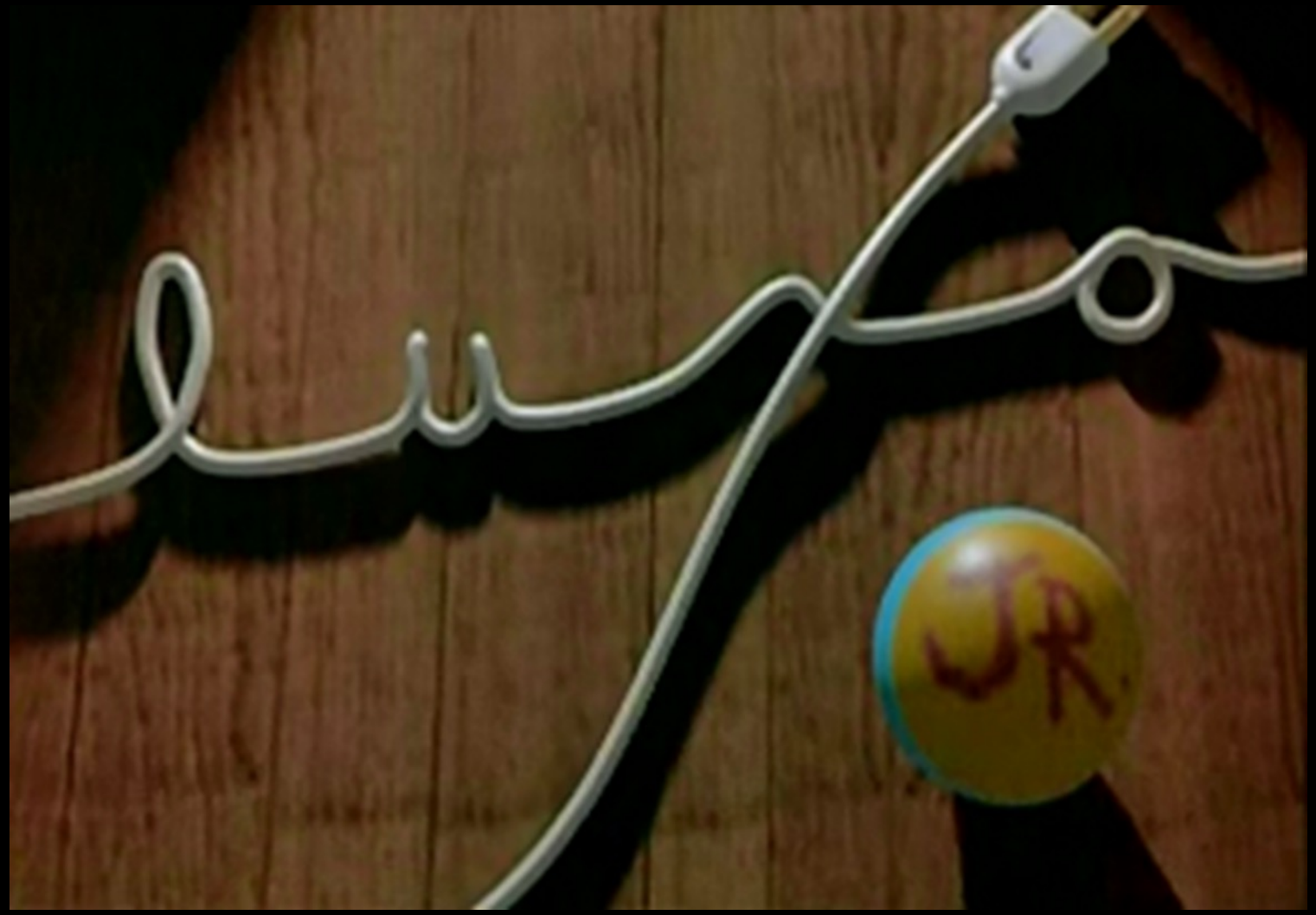
In this paper, I will explain the fundamental principles of traditional animation and how they apply to 3D keyframe computer animation.

2. PRINCIPLES OF ANIMATION

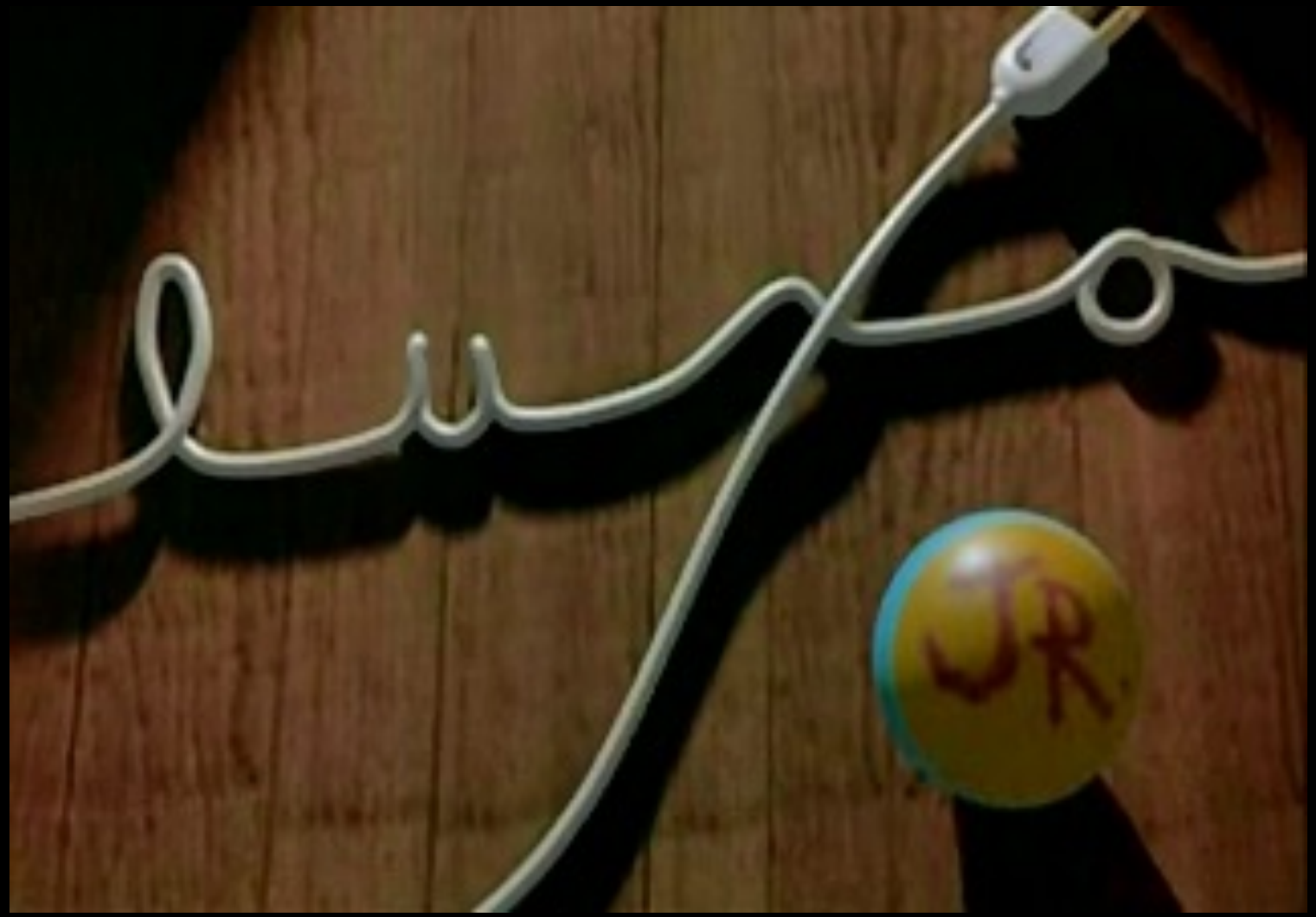
Between the late 1920's and the late 1930's animation grew from a novelty to an art form at the Walt Disney Studio. With every picture, actions became more convincing, and characters were emerging as true personalities. Audiences were enthusiastic and many of the animators were satisfied, however it was clear to Walt Disney that the level of animation and existing characters were not adequate to pursue new story lines-- characters were limited to certain types of action and, audience acceptance notwithstanding, they were not appealing to the eye. It was apparent to Walt Disney that no one could successfully animate a humanized figure or a life-like animal; a new drawing approach was necessary to improve the level of animation exemplified by the *Three Little Pigs* . [10]

FIGURE 1. Luxo Jr.'s hop with overlapping action on cord. Flip pages from last page of paper to front. The top figures are frames 1-5, the bottom are frames 6-10.

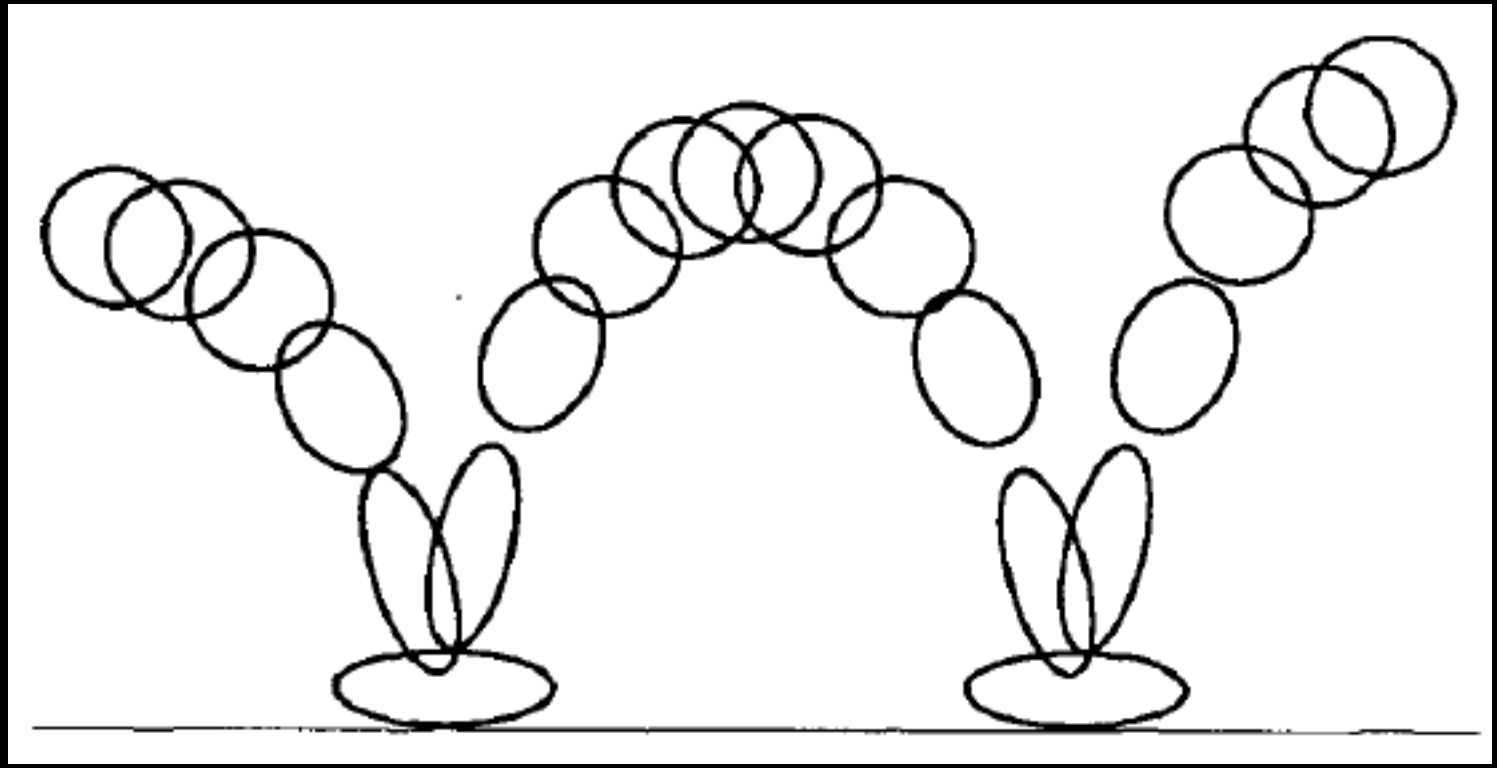
Luxo Jr.



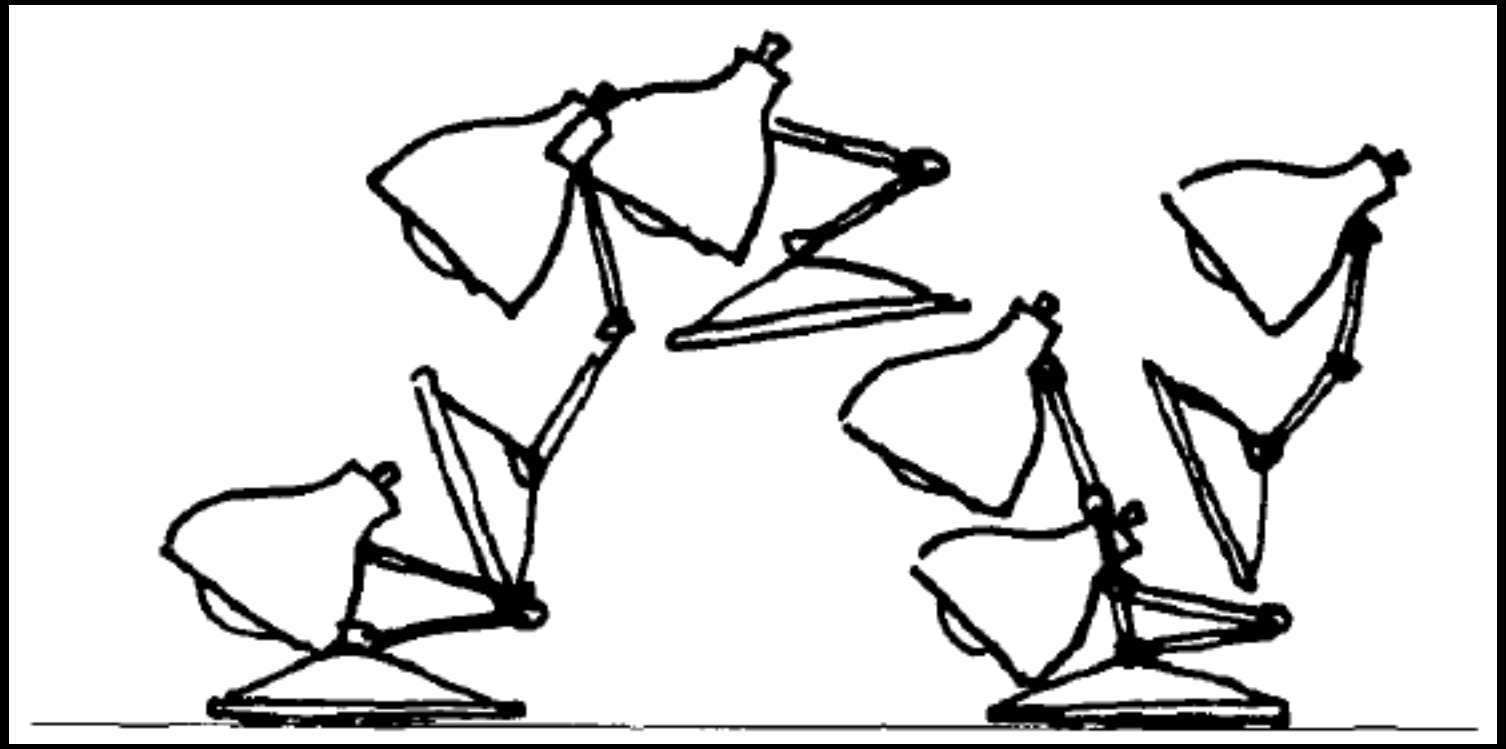
Luxo Jr.



Squash and Stretch



Squash and Stretch



Squash and Stretch

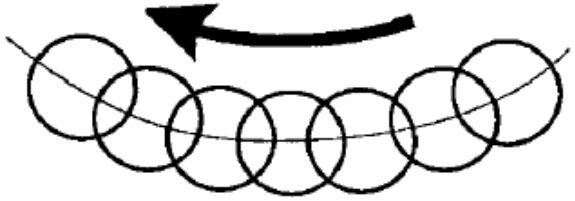


FIGURE 4a. In slow action, an object's position overlaps from frame to frame which gives the action a smooth appearance to the eye.

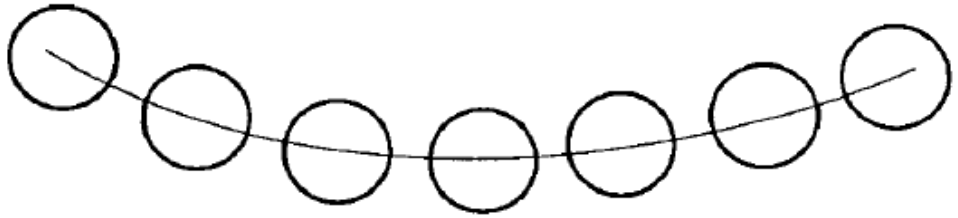


FIGURE 4b. Strobing occurs in a faster action when the object's positions do not overlap and the eye perceives seperate images.

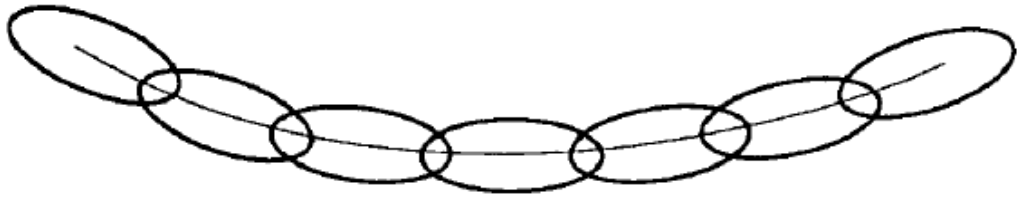


FIGURE 4c. Stretching the object so that it's positions overlap again will relieve the strobing effect.

Staging

FIGURES 7-8. In *Luxo Jr.*, all action was staged to the side for clarity.

Pose-to-Pose, Slow In, Slow Out

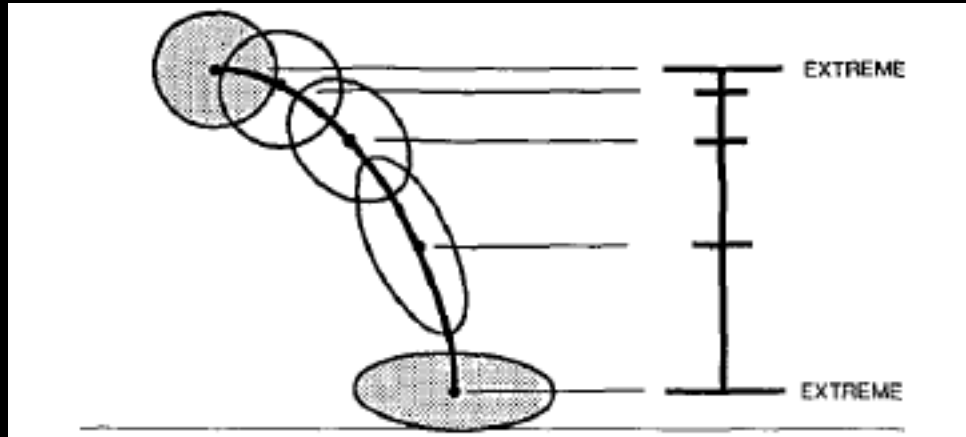


FIGURE 9. Timing chart for ball bounce.

Objects with mass must accelerate and decelerate

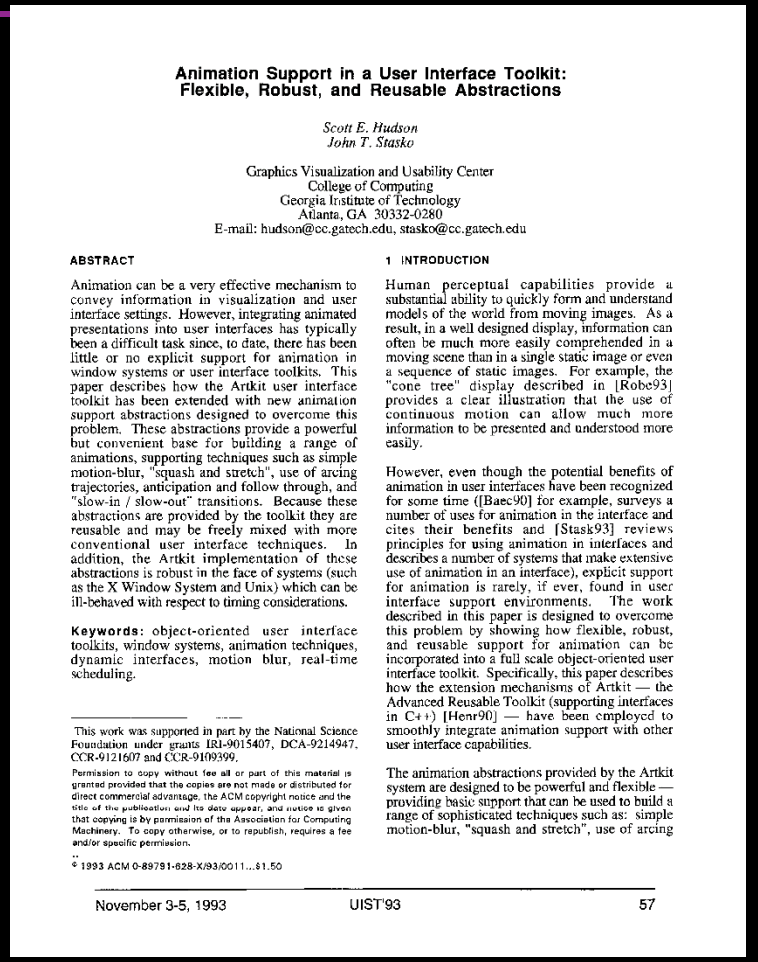
Interesting frames are typically at ends,
tweaks perception to emphasize these poses

Animation Case Study

Animation Support in a User Interface Toolkit: Flexible, Robust, and Reusable Abstractions

Hudson and Stasko, 1993

http://dx.doi.org/10.1145/168642.168648



Animation Support in a User Interface Toolkit: Flexible, Robust, and Reusable Abstractions

Scott E. Hudson
John T. Stasko

Graphics Visualization and Usability Center
College of Computing
Georgia Institute of Technology
Atlanta, GA 30332-0280
E-mail: hudson@cc.gatech.edu, stasko@cc.gatech.edu

ABSTRACT

Animation can be a very effective mechanism to convey information in visualization and user interface settings. However, integrating animated presentations into user interfaces has typically been a difficult task since, to date, there has been little or no explicit support for animation in window systems or user interface toolkits. This paper describes how the Artkit user interface toolkit has been extended with new animation support abstractions designed to overcome this problem. These abstractions provide a powerful but convenient base for building a range of animations, supporting techniques such as simple motion-blur, "squash and stretch", use of arcing trajectories, anticipation and follow through, and "slow-in / slow-out" transitions. Because these abstractions are provided by the toolkit they are reusable and may be freely mixed with more conventional user interface techniques. In addition, the Artkit implementation of these abstractions is robust in the face of systems (such as the X Window System and Unix) which can be ill-behaved with respect to timing considerations.

Keywords: object-oriented user interface toolkits, window systems, animation techniques, dynamic interfaces, motion blur, real-time scheduling.

This work was supported in part by the National Science Foundation under grants IRI-9015407, DCA-9214947, CCR-9121607 and CCR-9109399.

1 INTRODUCTION

Human perceptual capabilities provide a substantial ability to quickly form and understand models of the world from moving images. As a result, in a well designed display, information can often be much more easily comprehended in a moving scene than in a single static image or even a sequence of static images. For example, the "cone tree" display described in [Robe93] provides a clear illustration that the use of continuous motion can allow much more information to be presented and understood more easily.

However, even though the potential benefits of animation in user interfaces have been recognized for some time ([Baec90] for example, surveys a number of uses for animation in the interface and cites their benefits and [Stask93] reviews principles for using animation in interfaces and describes a number of systems that make extensive use of animation in an interface), explicit support for animation is rarely, if ever, found in user interface support environments. The work described in this paper is designed to overcome this problem by showing how flexible, robust, and reusable support for animation can be incorporated into a full scale object-oriented user interface toolkit. Specifically, this paper describes how the extension mechanisms of Artkit — the Advanced Reusable Toolkit (supporting interfaces in C++) [Henr90] — have been employed to smoothly integrate animation support with other user interface capabilities.

The animation abstractions provided by the Artkit system are designed to be powerful and flexible — providing basic support that can be used to build a range of sophisticated techniques such as: simple motion-blur, "squash and stretch", use of arcing

Events and Animation

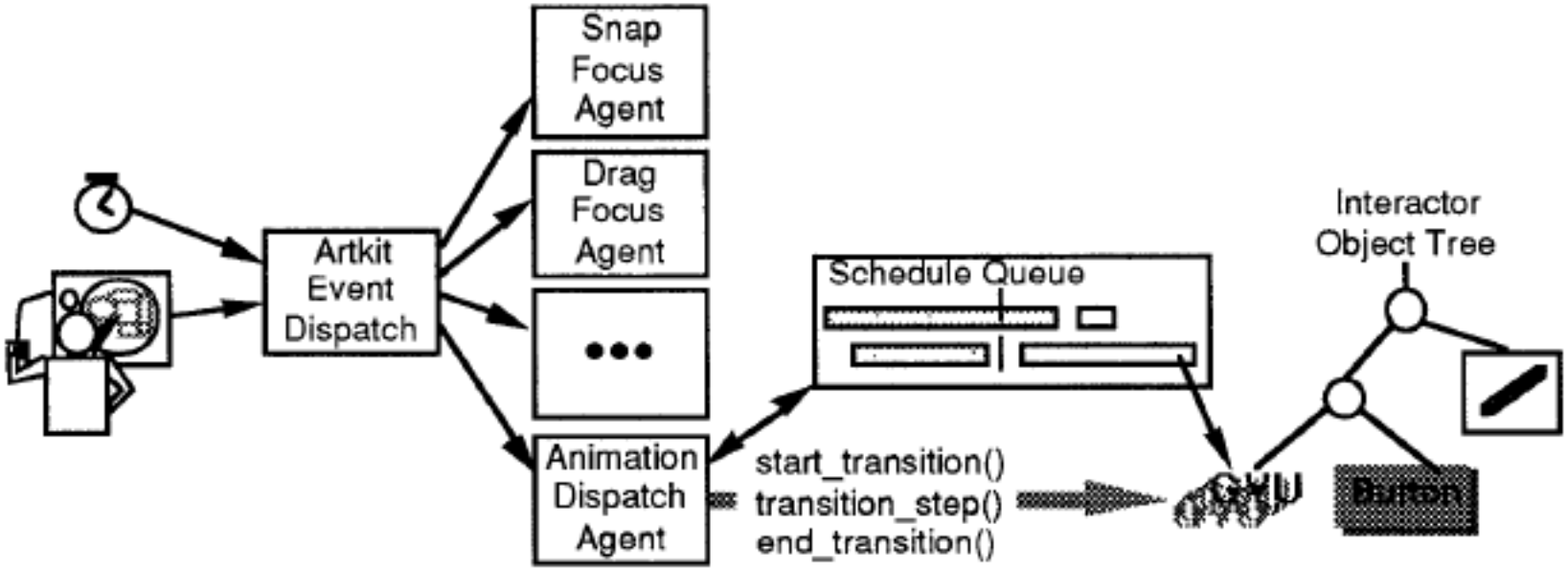


Figure 5. Animation Event Translation and Dispatch

Not Just an Implementation

Provides tool abstractions for
implementing previously presented styles of animation

Overcomes a fundamental clash of approaches

- Event loop receives input, processes, repaints
- Animations expect careful control of frames,
but the event loop has variable timing

Transition and Pacing

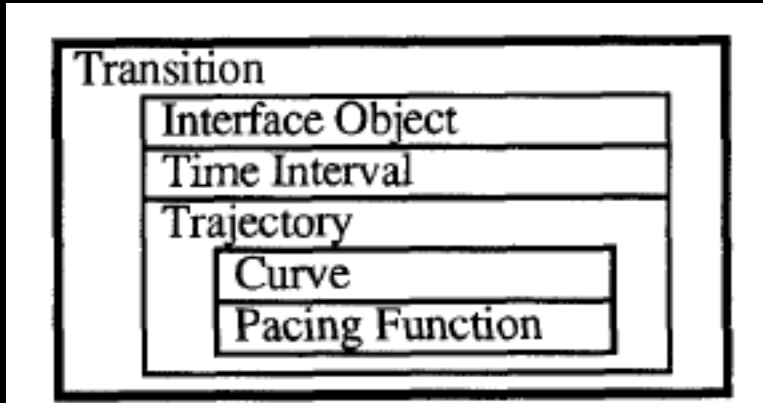


Figure 3. Parts of a Transition Object

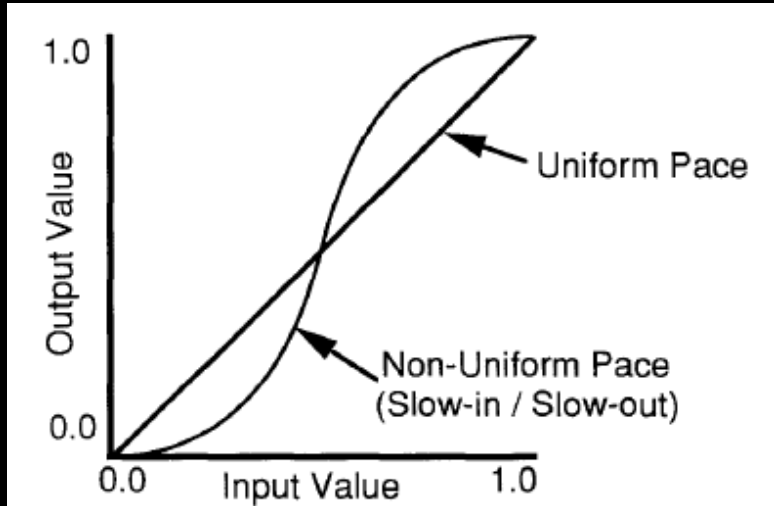


Figure 4. Two Example Pacing Functions

Computing a Frame

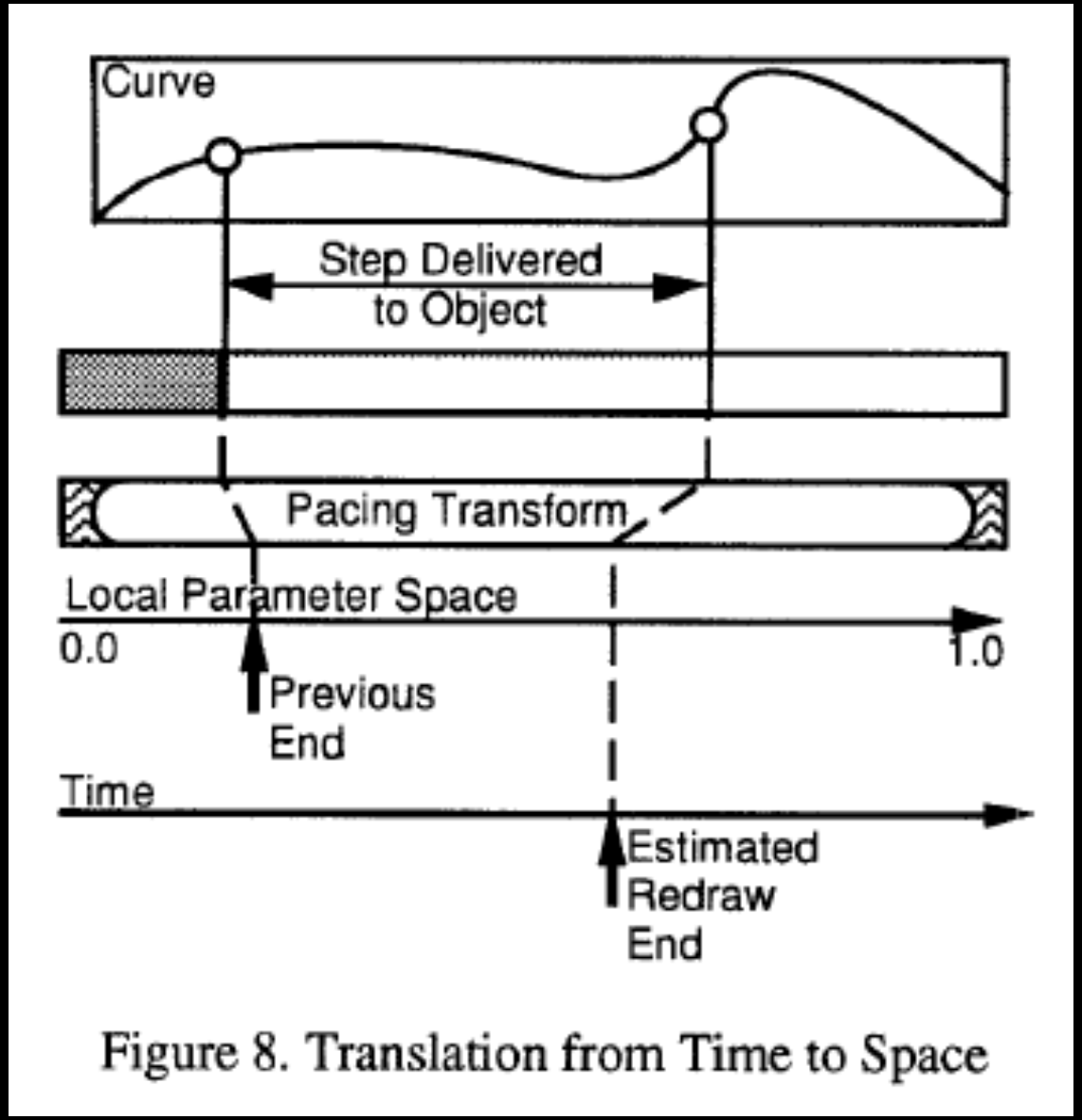


Figure 8. Translation from Time to Space

Animation Case Study

Now built into major commercial toolkits
(e.g., as jQuery's "easing functions")

Based on increased understanding
of how to animate interface transitions,
increasingly mature patterns and tools develop

Once mature, begins to be used as a
building block in even more complex behaviors

Today

Controlled A/B Experiments

Patterns

Case Study of Model-View-Controller

Case Study of Animation

Role of Interface Tools

Understanding Tools

We use tools because they

- Identify common or important practices
- Package those practices in a framework
- Make it easy to follow those practices
- Make it easier to focus on our application

What are the benefits of this?

- Being faster allows more iterative design
- Implementation is generally better in the tool
- Consistency across applications using same tool

Sapir-Whorf Hypothesis

Sapir-Whorf Hypothesis

Roughly, some thoughts in one language cannot be stated or understood in another language

- Language is not simply a way of voicing ideas, but is the very thing which shapes those ideas

Our tools define the language of interaction

- Beyond the simple matter of code
- Frame how we think about possibilities

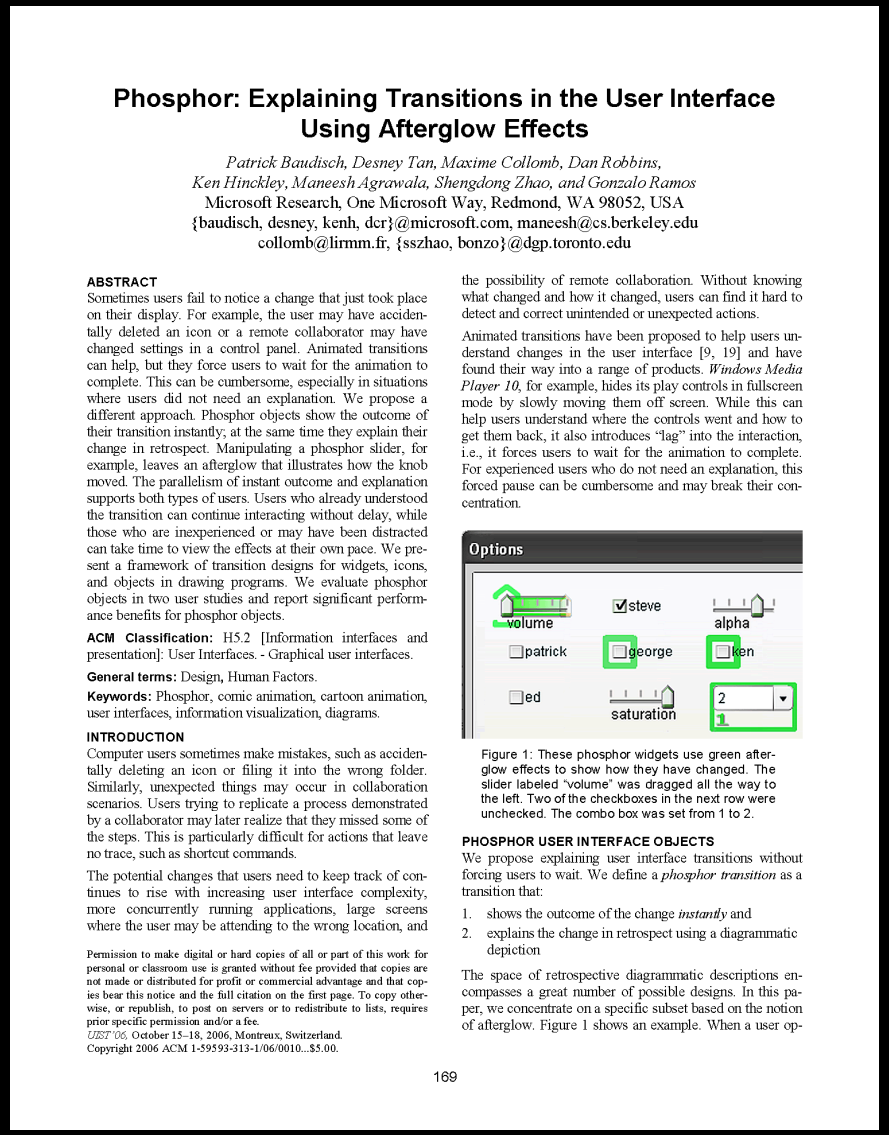
You must be aware of this when choosing tools, designing applications, and creating new tools

Animation Case Study

Phosphor:
Explaining Transitions
in the User Interface
Using Afterglow Effects

Baudisch et al, 2006

http://dx.doi.org/10.1145/1166253.1166280



Phosphor: Explaining Transitions in the User Interface Using Afterglow Effects

Patrick Baudisch, Desney Tan, Maxime Collomb, Dan Robbins, Ken Hinckley, Maneesh Agrawala, Shengdong Zhao, and Gonzalo Ramos
Microsoft Research, One Microsoft Way, Redmond, WA 98052, USA
{baudisch, desney, kenh, dcr}@microsoft.com, maneesh@cs.berkeley.edu
collomb@lirmm.fr, {sszhao, bonzo}@dgp.toronto.edu

ABSTRACT
Sometimes users fail to notice a change that just took place on their display. For example, the user may have accidentally deleted an icon or a remote collaborator may have changed settings in a control panel. Animated transitions can help, but they force users to wait for the animation to complete. This can be cumbersome, especially in situations where users did not need an explanation. We propose a different approach. Phosphor objects show the outcome of their transition instantly; at the same time they explain their change in retrospect. Manipulating a phosphor slider, for example, leaves an afterglow that illustrates how the knob moved. The parallelism of instant outcome and explanation supports both types of users. Users who already understood the transition can continue interacting without delay, while those who are inexperienced or may have been distracted can take time to view the effects at their own pace. We present a framework of transition designs for widgets, icons, and objects in drawing programs. We evaluate phosphor objects in two user studies and report significant performance benefits for phosphor objects.

ACM Classification: H5.2 [Information interfaces and presentation]: User Interfaces. - Graphical user interfaces.

General terms: Design, Human Factors.

Keywords: Phosphor, comic animation, cartoon animation, user interfaces, information visualization, diagrams.

INTRODUCTION
Computer users sometimes make mistakes, such as accidentally deleting an icon or filing it into the wrong folder. Similarly, unexpected things may occur in collaboration scenarios. Users trying to replicate a process demonstrated by a collaborator may later realize that they missed some of the steps. This is particularly difficult for actions that leave no trace, such as shortcut commands.

The potential changes that users need to keep track of continues to rise with increasing user interface complexity, more concurrently running applications, large screens where the user may be attending to the wrong location, and the possibility of remote collaboration. Without knowing what changed and how it changed, users can find it hard to detect and correct unintended or unexpected actions.

Animated transitions have been proposed to help users understand changes in the user interface [9, 19] and have found their way into a range of products. *Windows Media Player 10*, for example, hides its play controls in fullscreen mode by slowly moving them off screen. While this can help users understand where the controls went and how to get them back, it also introduces "lag" into the interaction, i.e., it forces users to wait for the animation to complete. For experienced users who do not need an explanation, this forced pause can be cumbersome and may break their concentration.

UIST'06, October 15–18, 2006, Montreux, Switzerland.

Figure 1: These phosphor widgets use green afterglow effects to show how they have changed. The slider labeled "volume" was dragged all the way to the left. Two of the checkboxes in the next row were unchecked. The combo box was set from 1 to 2.

PHOSPHOR USER INTERFACE OBJECTS
We propose explaining user interface transitions without forcing users to wait. We define a *phosphor transition* as a transition that:

1. shows the outcome of the change *instantly* and
2. explains the change in retrospect using a diagrammatic depiction

The space of retrospective diagrammatic descriptions encompasses a great number of possible designs. In this paper, we concentrate on a specific subset based on the notion of afterglow. Figure 1 shows an example. When a user op-

Phosphor

Animation can help people follow interface transitions

Windows Media Player

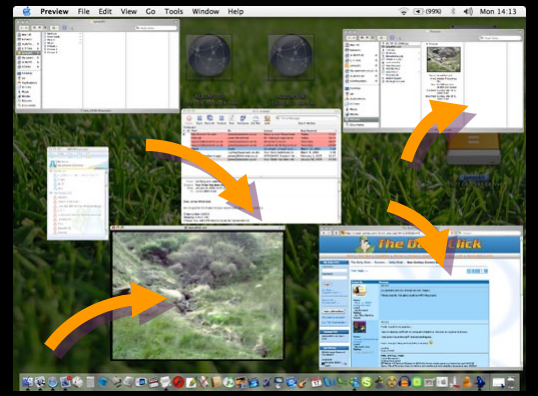
But the right speed is crucial

- Too fast increases error rate
- Too slow increases task time

The right speed depends on familiarity, distraction, etc.

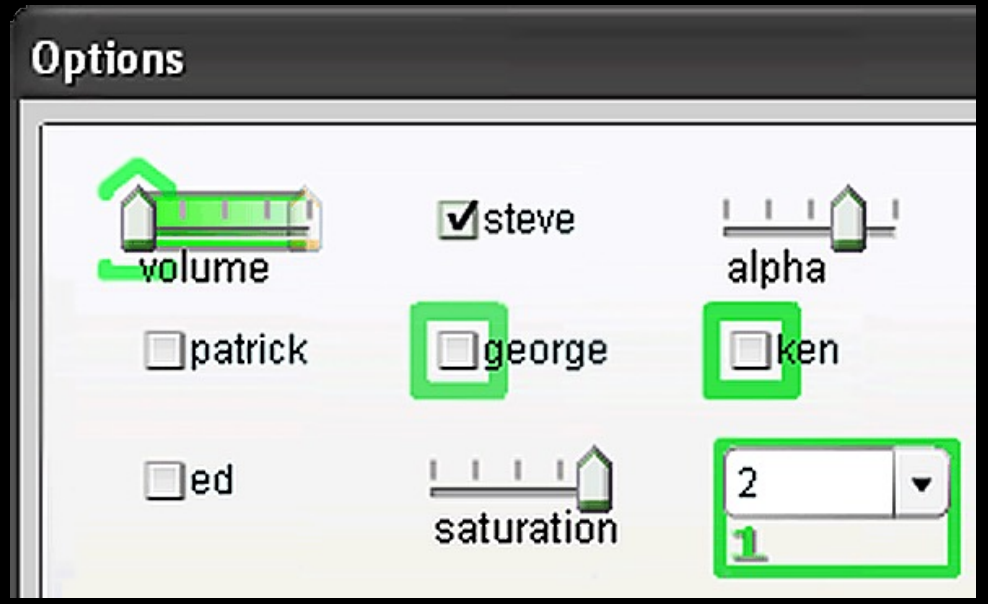
- It cannot be determined

Apple Expose



Phosphor

Phosphor shows the outcome immediately, then explains change in retrospect using a diagrammatic depiction

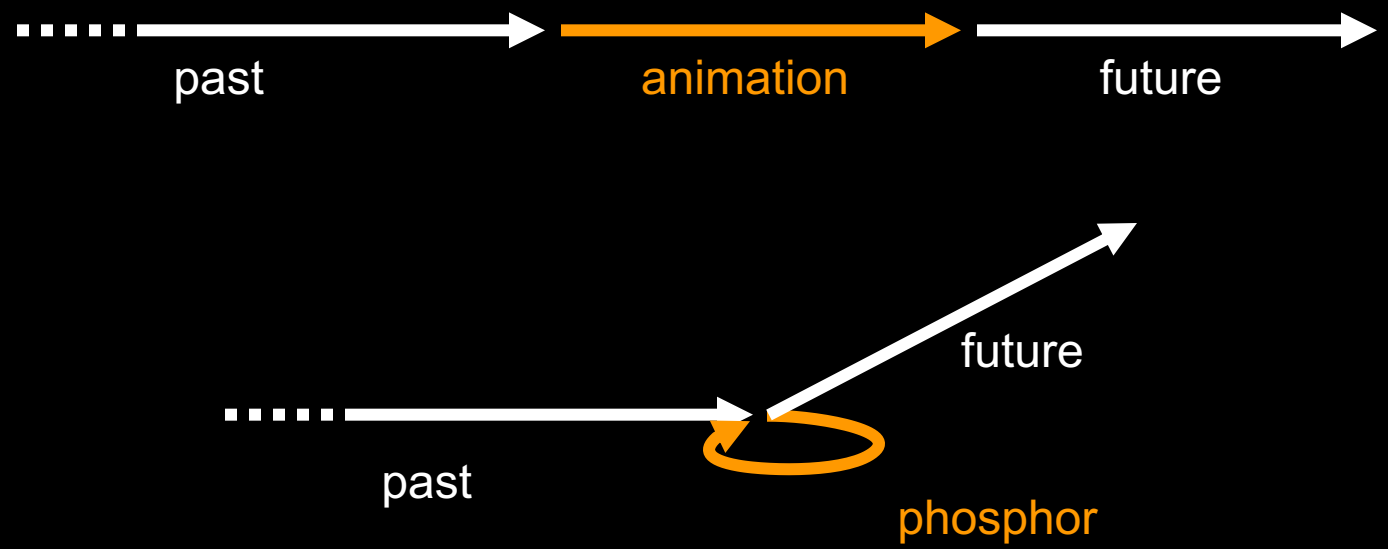


Phosphor

Phosphor

Challenging the Pattern in Tools

Phosphor breaks from the pattern that has evolved into current transition tools



Understanding Tools

Codification eventually constrains design

- Inevitable consequence of codification versus evolving understanding of emerging technologies

Codification goes deeper than the code

- Frames how we think about our applications

Myers, Hudson, Pausch. Past, Present, and Future of User Interface Software Tools. *TOCHI 2000.*

Tools and Interfaces

Tools embody expertise and assumptions

Tools evolve based on emerging understanding of how to address categories of problems

Be conscious of your tool decisions

- Try to think about designs before tying to a tool
- Choose good and appropriate tools
- Understand what you are getting in a tool
- Push yourself to think outside the tool

Rebuilding the Language of Design

We regularly rebuild the entire system

- Command Line, Text Screens
- Multiple Generations of Desktop
- Multiple Generations of Web
- Mobile Apps

We will do it again

- Several near-term challenges require it
- e.g., Touch, Cloud, Distributed Interfaces

Backward compatibility helps, but is not required

CSE 440:
Introduction to HCI

13: Patterns and Interface Implementation

May 7, 2024

Jesse J. Martinez | Avery Mack | Simona Liao