CSE 510: Advanced Topics in HCI

Interface Toolkits

James Fogarty
Daniel Epstein

Tuesday/Thursday
10:30 to 12:00

CSE 403

Tools and Interfaces

Why Interface Tools?

Case Study of Model-View-Controller

Case Study of Animation

Sapir-Whorf Hypothesis

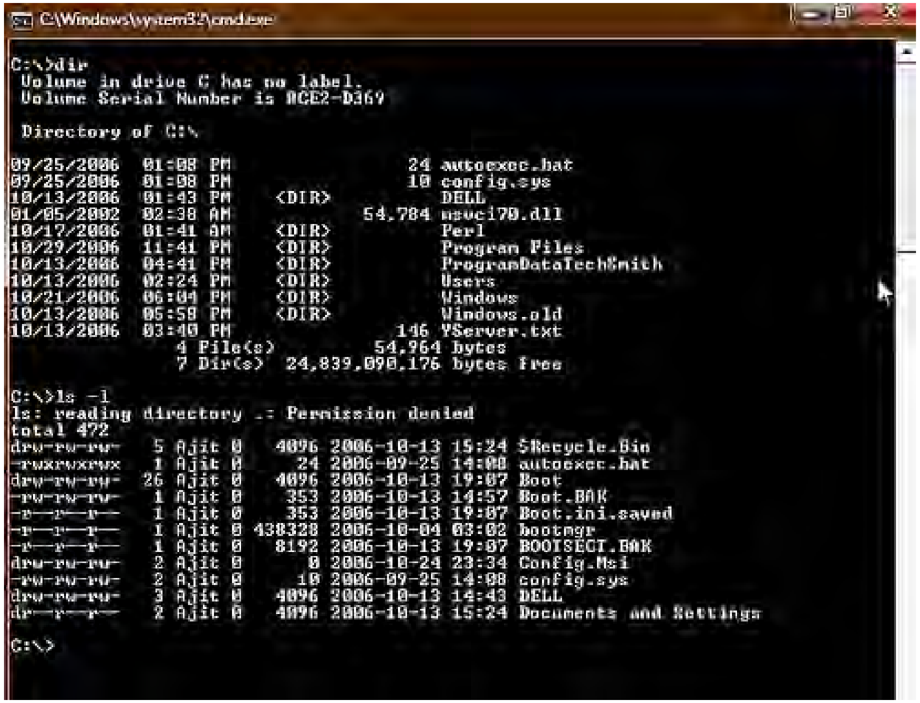
Things I Hope You Learned

Sequential Programs

Program takes control, prompts for input

Person waits
on the program

Program says when
it is ready for more
input, which the
person then provides



Sequential Programs

```
while true {
   print “Prompt for Input”
   input = read_line_of_text()
   output = do_work()
   print output
}
```

Sequential Programs

```
while true {
    print “Prompt for Input”
    input = read_line_of_text()
    output = do_work()
    print output
}
```

Person is literally modeled as a file

Event-Driven Programming

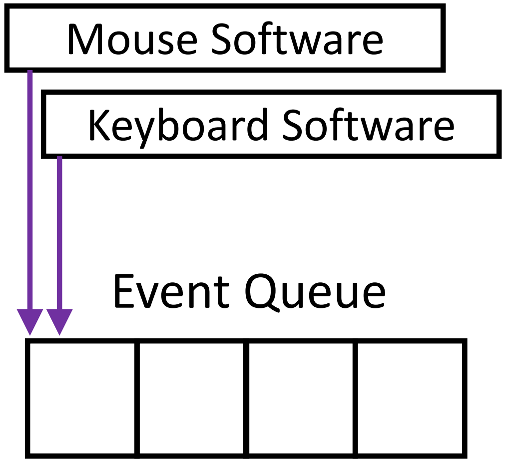
A program waits for a person to provide input

All communication done via events

- "mouse down", "item drag", "key up"

All events go to a queue

- Ensures events handled in order
- Hides specifics from applications



Basic Interactive Software Loop

```
do {
   e = read_event();               } input
   dispatch_event(e);              } processing
   if (damage_exists())
      update_display();            } output
} while (e.type != WM_QUIT);
```

Nearly all interactive software has this somewhere within it

Basic Interactive Software Loop

Have you ever written this loop?

Basic Interactive Software Loop

Have you ever written this loop?

Contrast with:

“One of the most complex aspects of Xlib programming is designing the event loop, which must take into account all of the possible events that can occur in a window.”

Nye & O'Reilly, X Toolkit Intrinsics Programming Manual, vol. 4, 1990, p. 241.

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?

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

Understanding Tools

Why is designing tools difficult?

- Need to understand the core practices and problems
- Those are often evolving with technology and design

Example: Responsiveness in event-driven interface

Event-driven interaction is asynchronous

How to maintain responsiveness in the interface
while executing some large computation?

Understanding Tools

Why is designing tools difficult?

- Need to understand the core practices and problems
- Those are often evolving with technology and design

Example: Responsiveness in event-driven interface

Cursor:

- WaitCursor vs. CWaitCursor vs. In Framework

Progress Bar:

- Data Races vs. Idle vs. Loop vs. Worker Objects

Fundamental Tools Terminology

- Threshold vs. Ceiling
  - Threshold: How hard to get started
  - Ceiling: How much can be achieved
  - These depend on what is being implemented
- Path of Least Resistance
  - Tools influence what interfaces are created
- Moving Targets
  - Changing needs require different tools

Myers et al, 2000
http://dx.doi.org/10.1145/344949.344959

Tools and Interfaces

Why Interface Tools?

Case Study of Model-View-Controller

Case Study of Animation

Sapir-Whorf Hypothesis

Things I Hope You Learned

Model-View-Controller

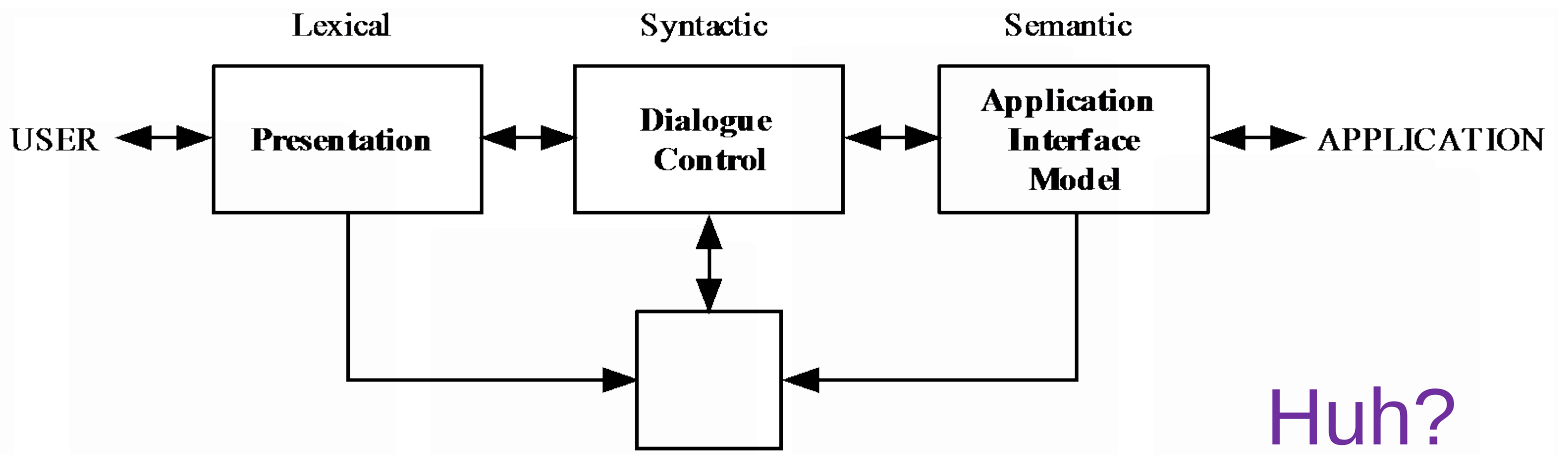
How to organize the code of an interface?

A surprisingly complicated question, with many unstated assumptions requiring significant background to understand and resolve

Seeheim Model

Buxton, 1983
http://dx.doi.org/10.1145/988584.988586

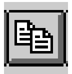
Results from 1985 workshop on user interface management systems, driven by goals of portability and modifiability, based in separating the interface from application functionality



Huh?

Seeheim Model

Lexical - Presentation

External presentation of interface — e.g., “add” vs. “append” vs. “^a” vs. 

Generates display, receive input — e.g., how to make a “menu” or “button”

Syntactic - Dialog Control

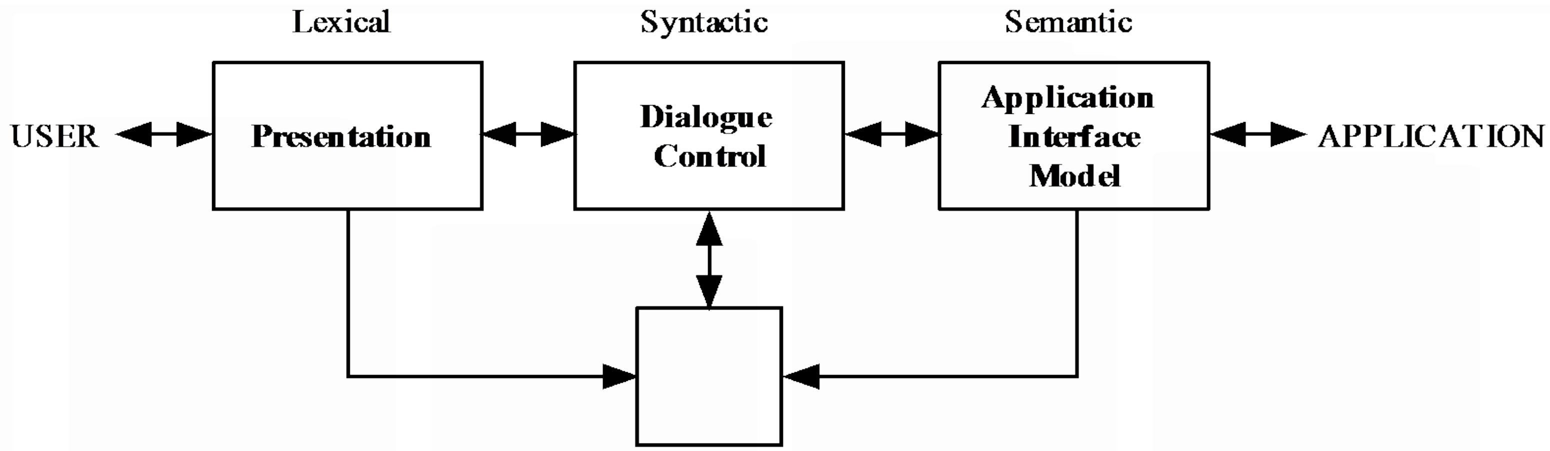
Parsing of tokens into syntax — e.g., interface modes

Maintain state

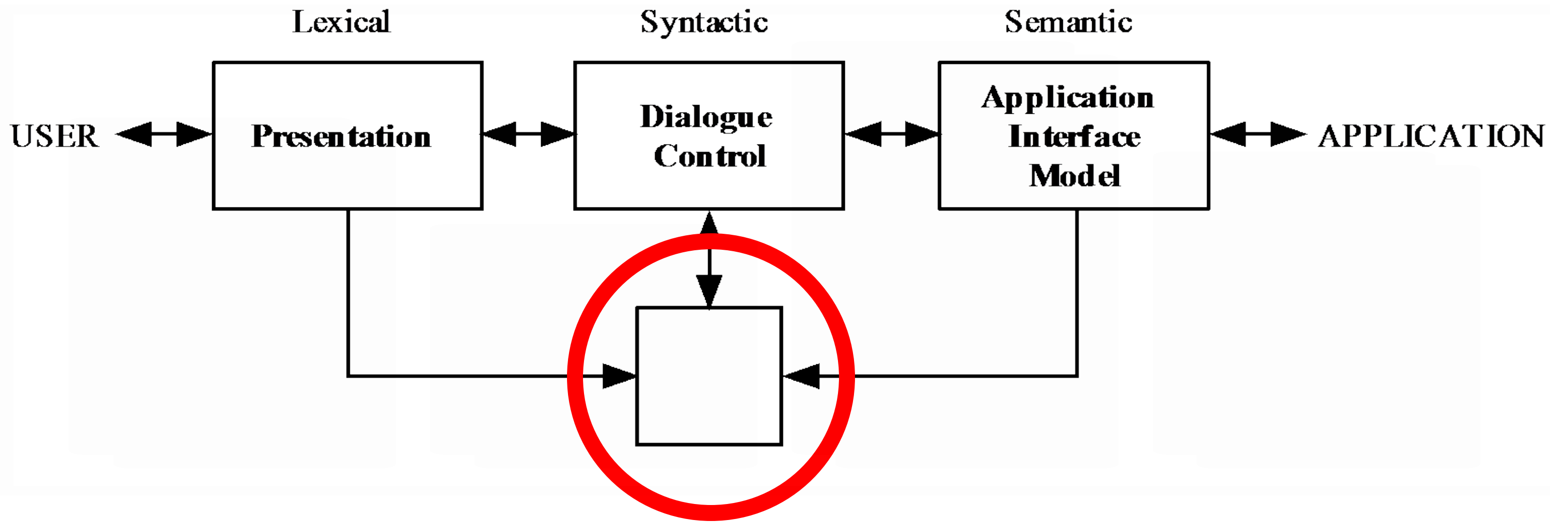
Semantic - Application Interface Model

Defines interaction between interface and rest of software — e.g., drag-and-drop target highlighting

Seeheim Model

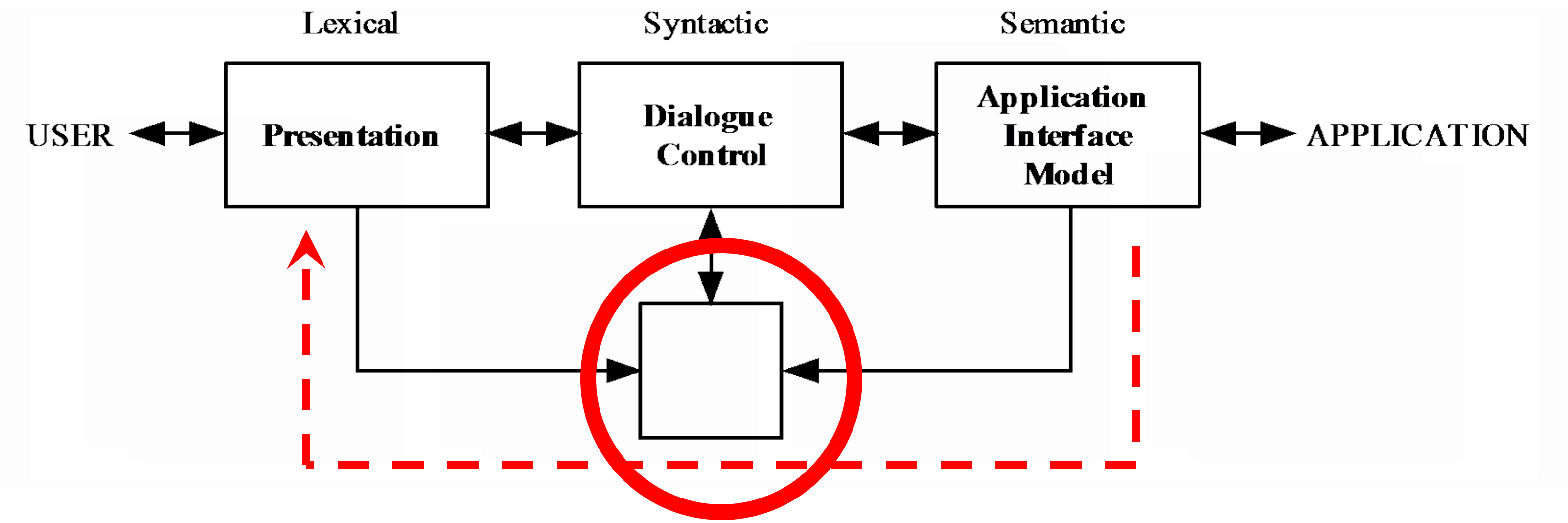


Seeheim Model



Huh?

Seeheim Model



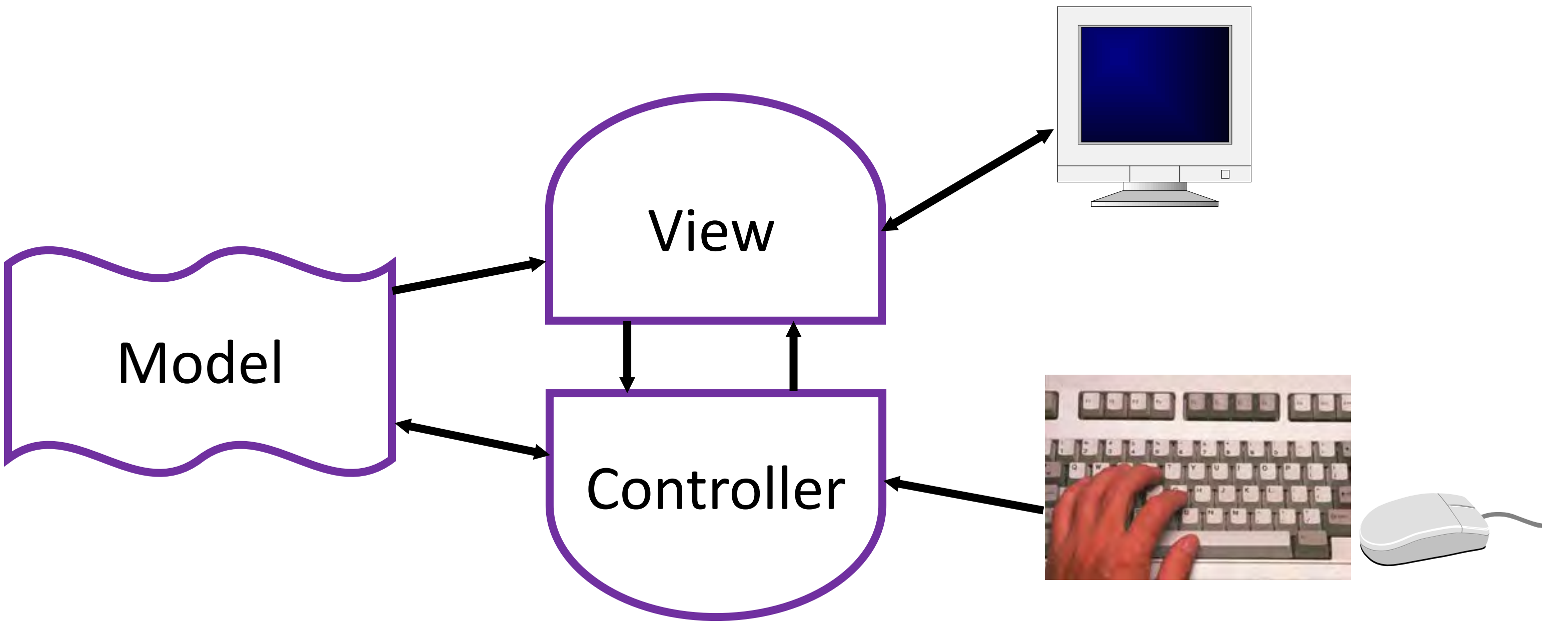
Rapid Semantic Feedback

In practice, all of the code goes in here

Model-View-Controller

Introduced by Smalltalk developers at PARC

Partitions application to be scalable, maintainable



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)

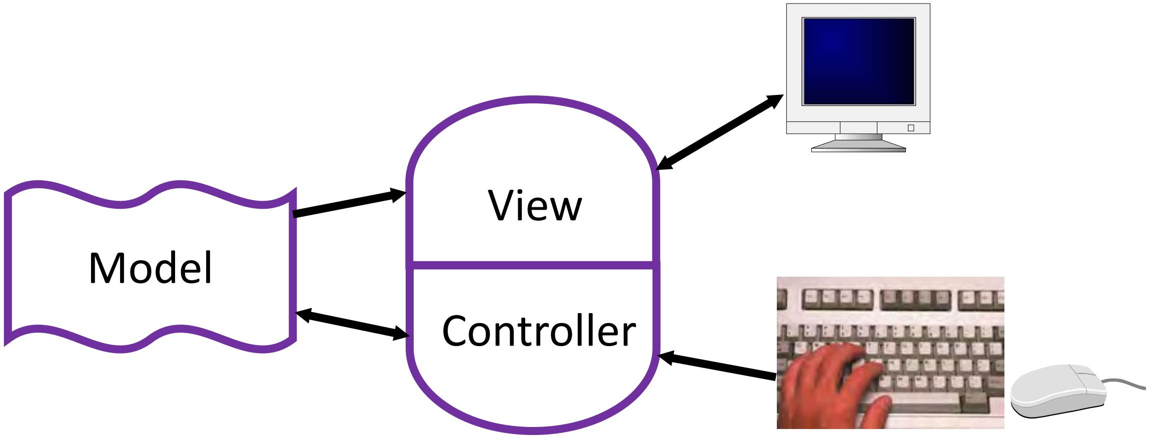
Controller contacts view to interpret what input events mean in context of a view (e.g., selection)

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

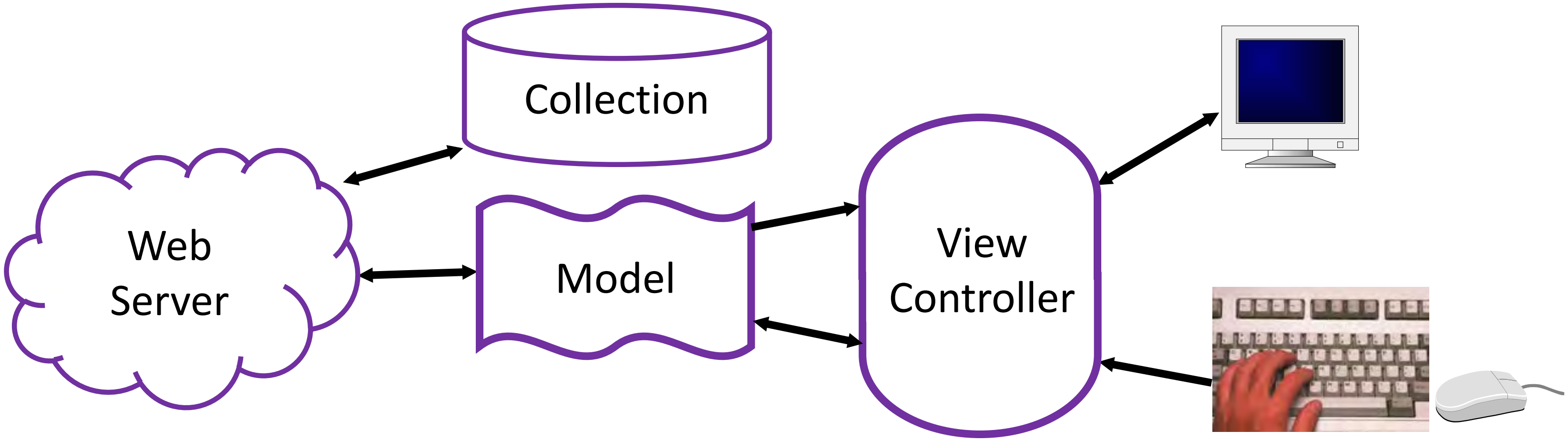
Separation of “business logic” can require care

- May help to think of model as the client model

Model-View-Collection on the Web

Core ideas manifest differently by needs

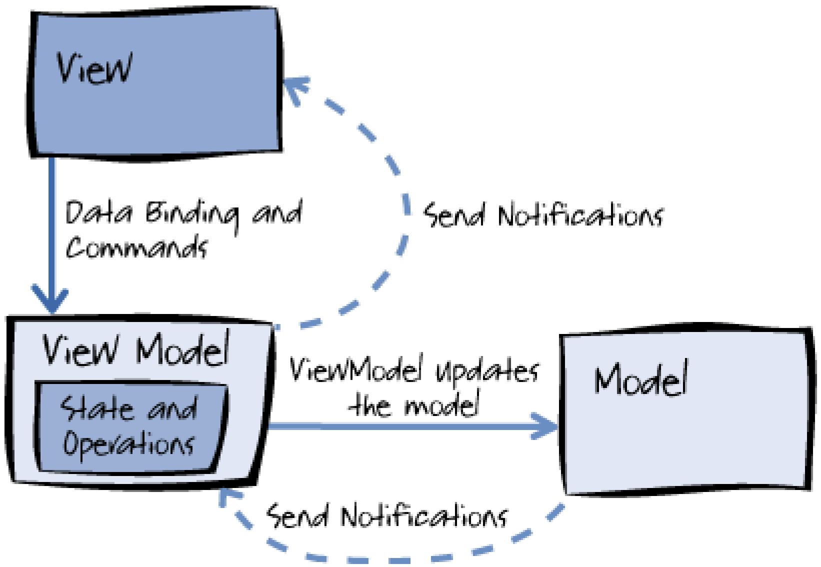
- For example, backbone.js implements client views of models, with REST API calls to web server
- Web tools often implement views as templates



Model View View-Model

Design to support data-binding
by minimizing functionality in view

Also allows greater separation of expertise



Tools and Interfaces

Why Interface Tools?

Case Study of Model-View-Controller

Case Study of Animation

Sapir-Whorf Hypothesis

Things I Hope You Learned

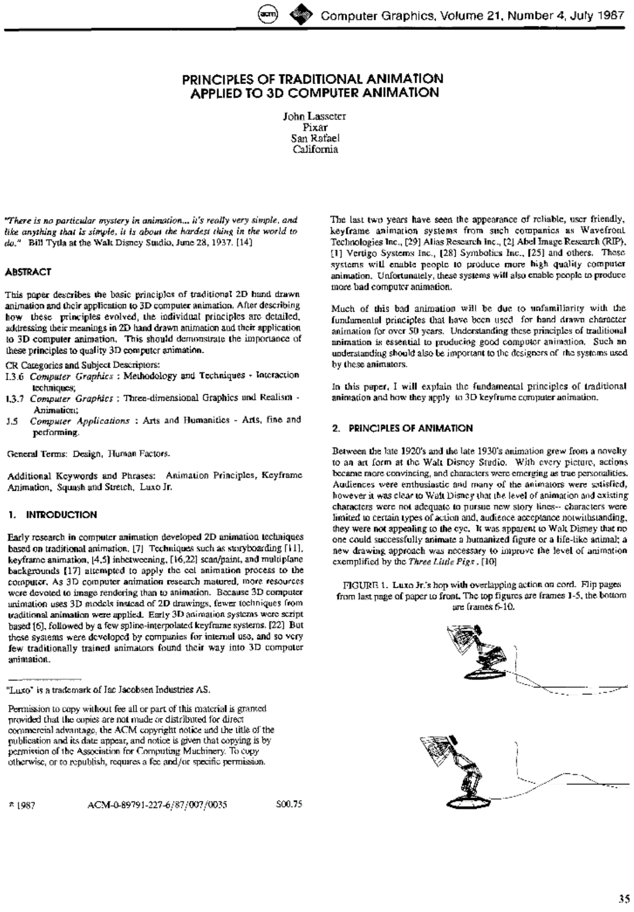
Luxo Jr.

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 these systems were developed by companies for internal use, and so very few traditionally trained animators found their way into 3D computer animation.

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] Symbolics 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.

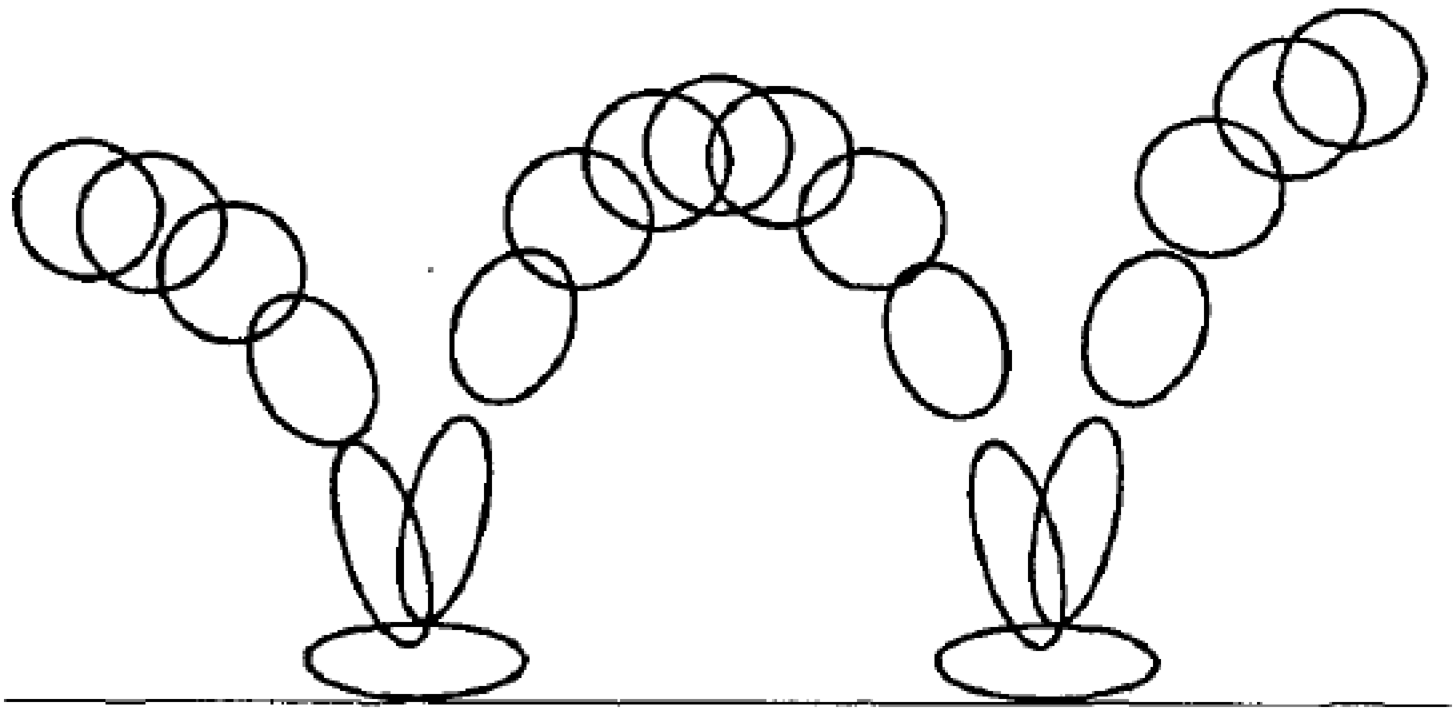
2. PRINCIPLES OF ANIMATION

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]

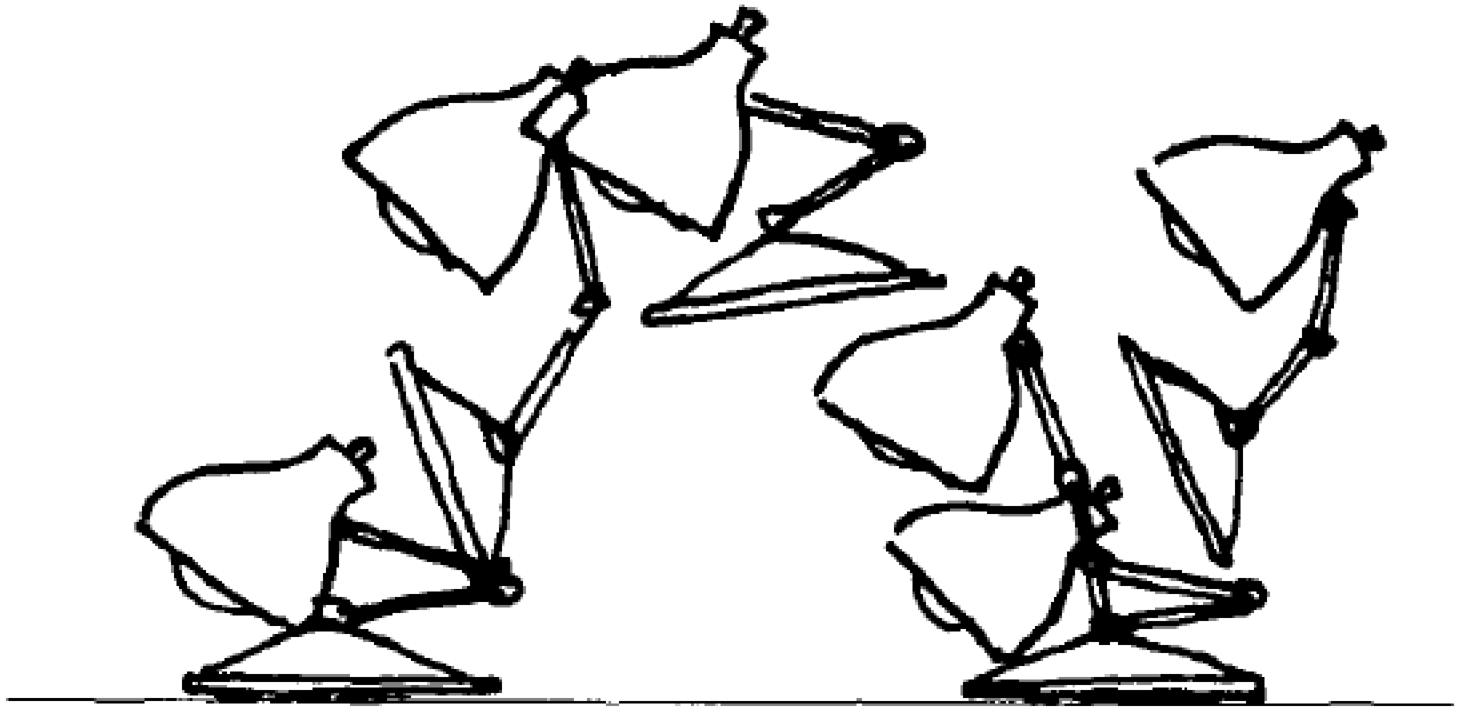
FIGURE 1. Luxo Jr.'s hop with overlapping action on cord. Flip pages from last page of paper to front. The top figures are frames 1-5, the bottom are frames 6-10.

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

Squash and Stretch



Squash and Stretch



Squash and Stretch

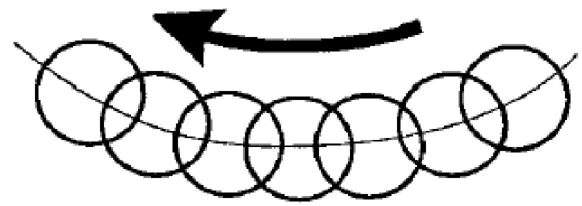


FIGURE 4a. In slow action, an object's position overlaps from frame to frame which gives the action a smooth appearance to the eye.

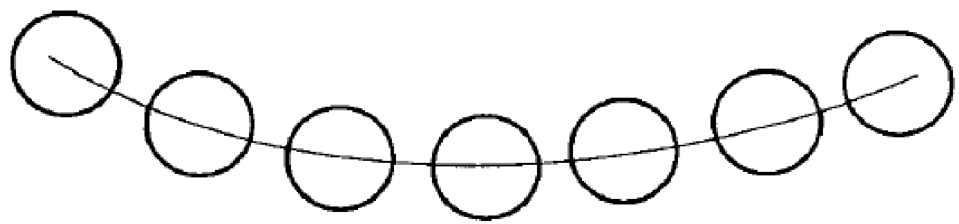


FIGURE 4b. Strobing occurs in a faster action when the object's positions do not overlap and the eye perceives seperate images.

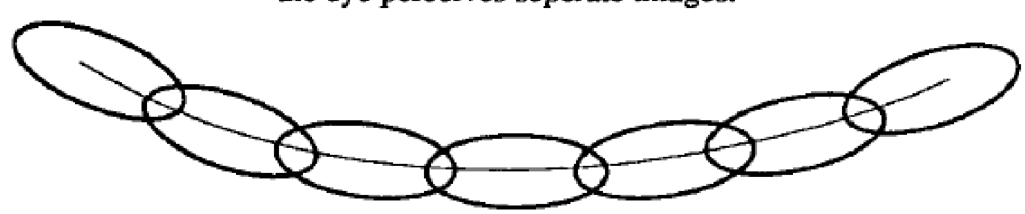


FIGURE 4c. Stretching the object so that it's positions overlap again will relieve the strobing effect.

Timing

Just two drawings of a head, the first showing it leaning toward the right shoulder and the second with it over on the left and its chin slightly raised, can be made to communicate a multitue of ideas, depending entirely on the Timing used. Each inbetween drawing added between these two "extremes" gives a new meaning to the action.

NO inbetweens........... The Character has been hit by a tremendous force. His head is nearly snapped off.

ONE inbetweens......... The Character has been hit by a brick, rolling pin, frying pan.

TWO inbetweens......... The Character has a nervous tic, a muscle spasm, an uncontrollable twitch.

THREE inbetweens..... The Character is dodging a brick, rolling pin, frying pan.

Timing

FOUR inbetweens........... The Character is giving a crisp order, "Get going!" "Move it!"

FIVE inbetweens........... The Character is more friendly, "Over here." "Come on-hurry!"

SIX inbetweens........... The Character sees a good looking girl, or the sports car he has always wanted.

SEVEN inbetweens........... The Character tries to get a better look at something.

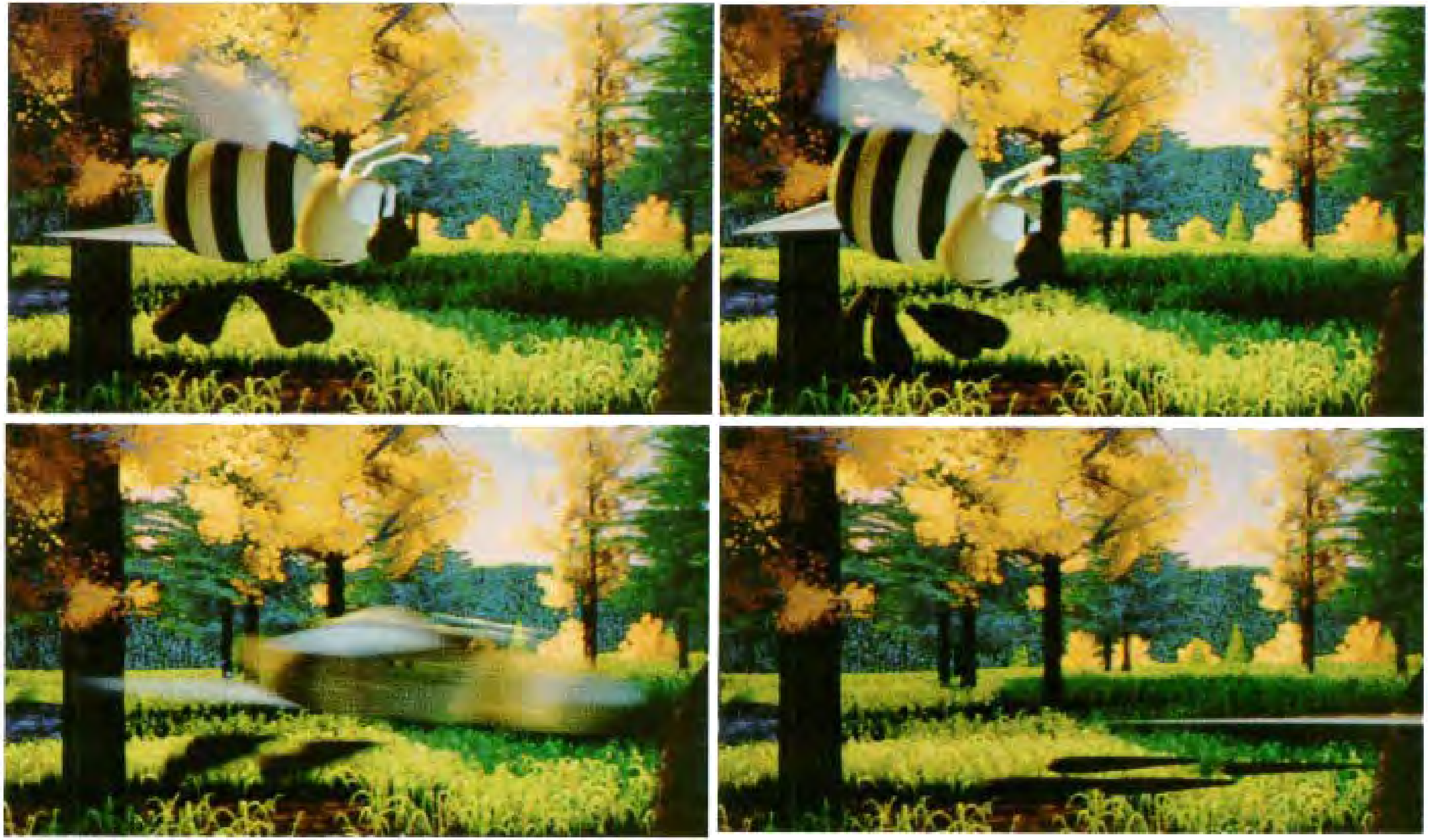
Timing

EIGHT inbetweens........... The Character searches for the peanut butter on the kitchen shelf.

NINE inbetweens...........The Character appraises, considering thoughtfully.

TEN inbetweens........... The Character stretches a sore muscle.

Anticipation



Staging

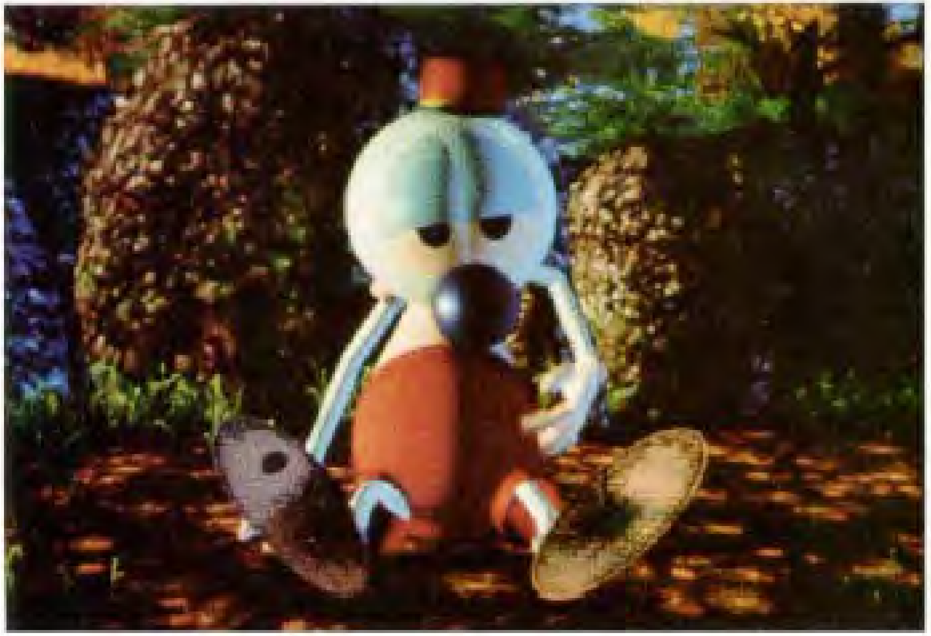
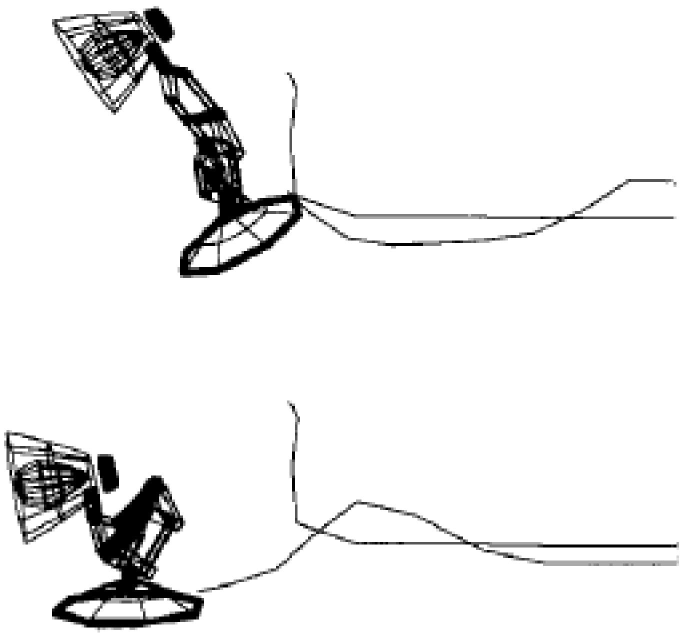


FIGURE 6. Andre's scratch was staged to the side (in "silhouette") for clarity and because that is where his itch was.

Staging

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

Follow Through, Overlap, Secondary



Pose-to-Pose, Slow In, Slow Out

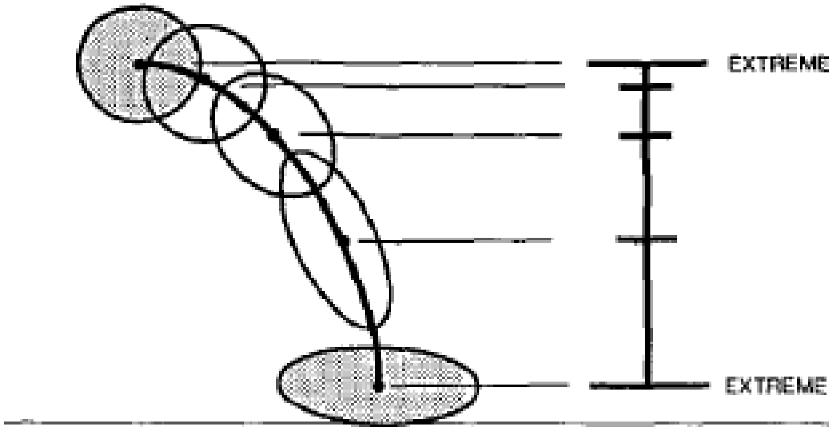
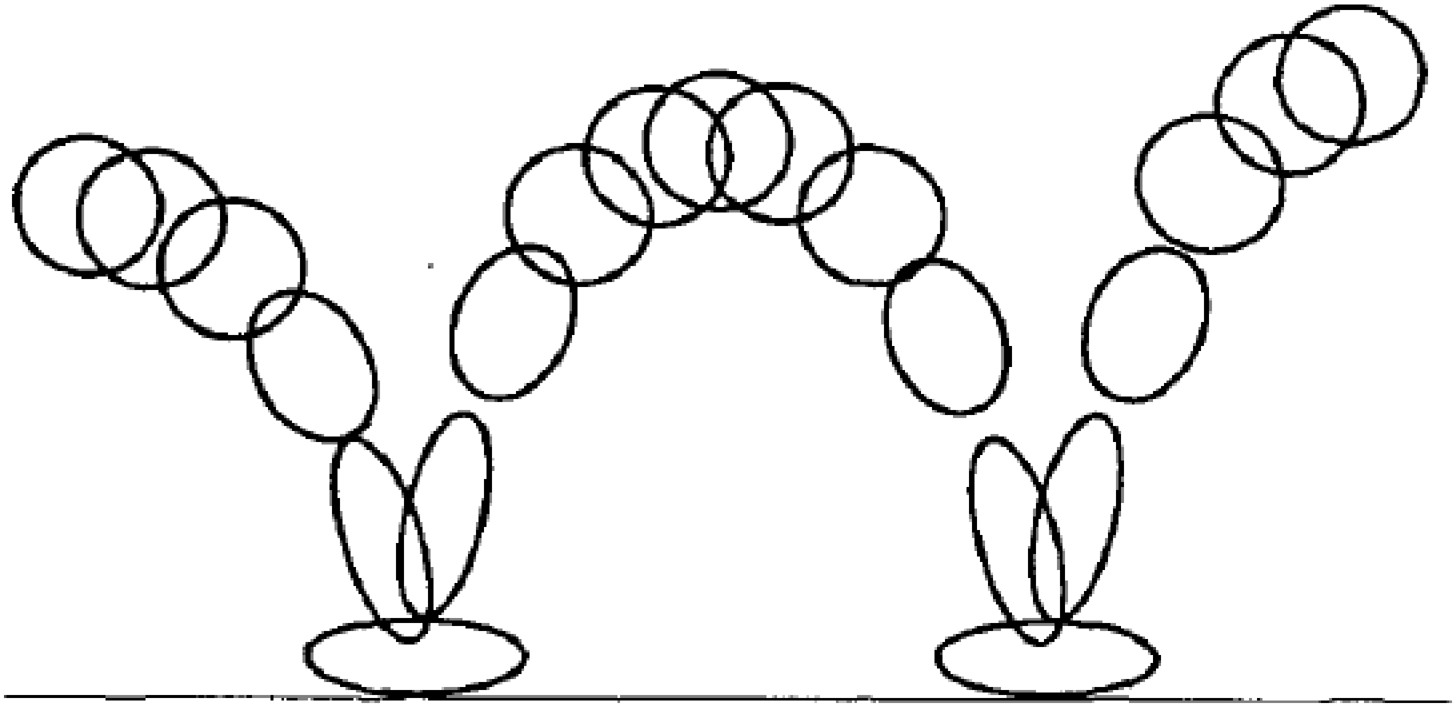


FIGURE 9. Timing chart for ball bounce.

Objects with mass must accelerate and decelerate

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

Arcs

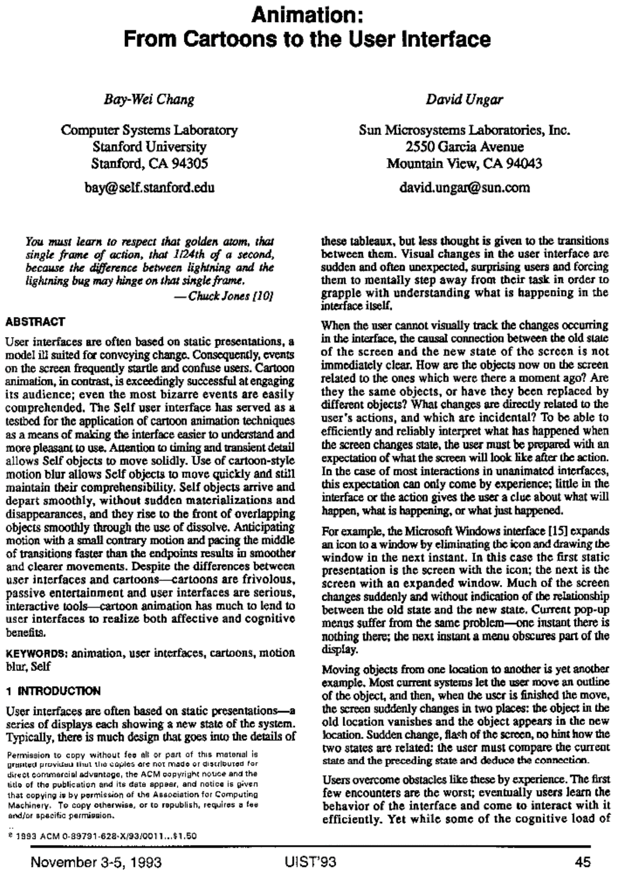


Animation Case Study

Animation: From Cartoons to the User Interface

Chang and Ungar, 1993

http://dx.doi.org/10.1145/168642.168647



Animation:
From Cartoons to the User Interface

Bay-Wei Chang

Computer Systems Laboratory
Stanford University
Stanford, CA 94305

bay@self.stanford.edu

David Ungar

Sun Microsystems Laboratories, Inc.
2550 Garcia Avenue
Mountain View, CA 94043

david.ungar@sun.com

You must learn to respect that golden atom, that single frame of action, that 1/24th of a second, because the difference between lightning and the lightning bug may hinge on that single frame.

—Chuck Jones [10]

ABSTRACT

User interfaces are often based on static presentations, a model ill suited for conveying change. Consequently, events on the screen frequently startle and confuse users. Cartoon animation, in contrast, is exceedingly successful at engaging its audience; even the most bizarre events are easily comprehended. The Self user interface has served as a testbed for the application of cartoon animation techniques as a means of making the interface easier to understand and more pleasant to use. Attention to timing and transient detail allows Self objects to move solidly. Use of cartoon-style motion blur allows Self objects to move quickly and still maintain their comprehensibility. Self objects arrive and depart smoothly, without sudden materializations and disappearances, and they rise to the front of overlapping objects smoothly through the use of dissolve. Anticipating motion with a small contrary motion and pacing the middle of transitions faster than the endpoints results in smoother and clearer movements. Despite the differences between user interfaces and cartoons—cartoons are frivolous, passive entertainment and user interfaces are serious, interactive tools—cartoon animation has much to lend to user interfaces to realize both affective and cognitive benefits.

KEYWORDS: animation, user interfaces, cartoons, motion blur, Self

1 INTRODUCTION

User interfaces are often based on static presentations—a series of displays each showing a new state of the system. Typically, there is much design that goes into the details of these tableaux, but less thought is given to the transitions between them. Visual changes in the user interface are sudden and often unexpected, surprising users and forcing them to mentally step away from their task in order to grapple with understanding what is happening in the interface itself.

When the user cannot visually track the changes occurring in the interface, the causal connection between the old state of the screen and the new state of the screen is not immediately clear. How are the objects now on the screen related to the ones which were there a moment ago? Are they the same objects, or have they been replaced by different objects? What changes are directly related to the user's actions, and which are incidental? To be able to efficiently and reliably interpret what has happened when the screen changes state, the user must be prepared with an expectation of what the screen will look like after the action. In the case of most interactions in unanimated interfaces, this expectation can only come by experience; little in the interface or the action gives the user a clue about what will happen, what is happening, or what just happened.

For example, the Microsoft Windows interface [15] expands an icon to a window by eliminating the icon and drawing the window in the next instant. In this case the first static presentation is the screen with the icon; the next is the screen with an expanded window. Much of the screen changes suddenly and without indication of the relationship between the old state and the new state. Current pop-up menus suffer from the same problem—one instant there is nothing there; the next instant a menu obscures part of the display.

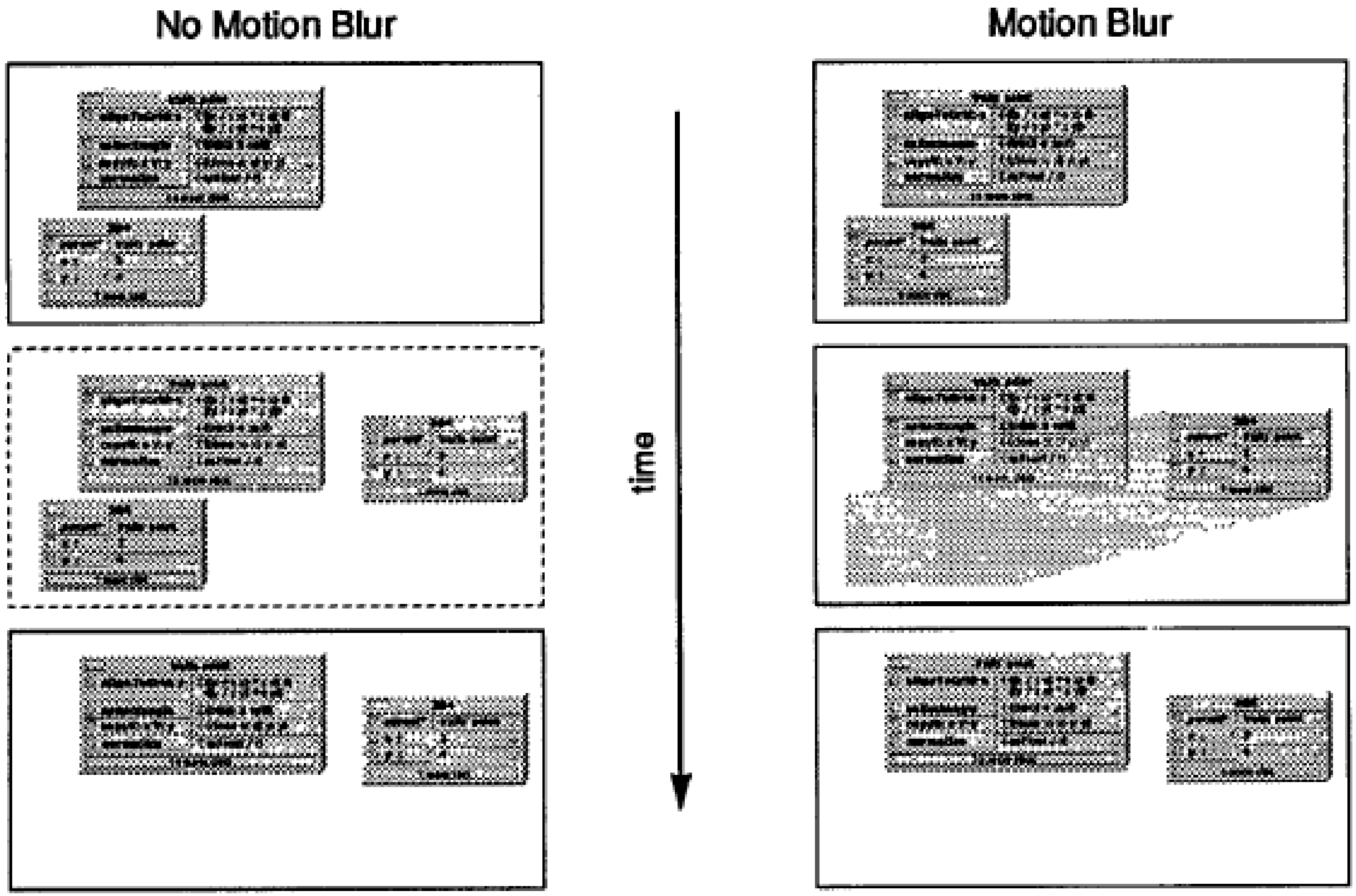
Moving objects from one location to another is yet another example. Most current systems let the user move an outline of the object, and then, when the user is finished the move, the screen suddenly changes in two places: the object in the old location vanishes and the object appears in the new location. Sudden change, flash of the screen, no hint how the two states are related: the user must compare the current state and the preceding state and deduce the connection.

Users overcome obstacles like these by experience. The first few encounters are the worst; eventually users learn the behavior of the interface and come to interact with it efficiently. Yet while some of the cognitive load of

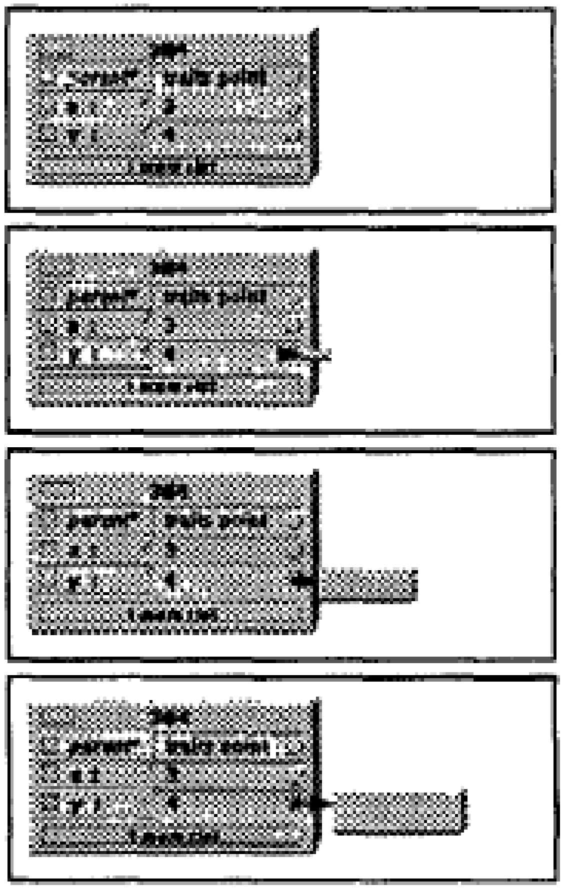
States Three Principles

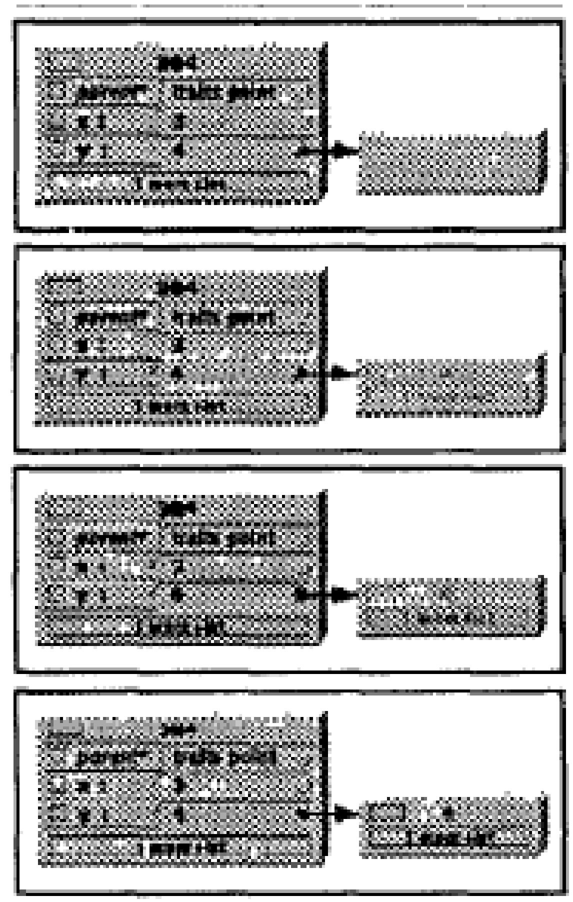
- Solidity
  - Desktop objects should appear to be solid objects
- Exaggeration
  - Exaggerate physical actions to enhance perception
- Reinforcement
  - Use effects to drive home feeling of reality

Solidity: Motion Blur

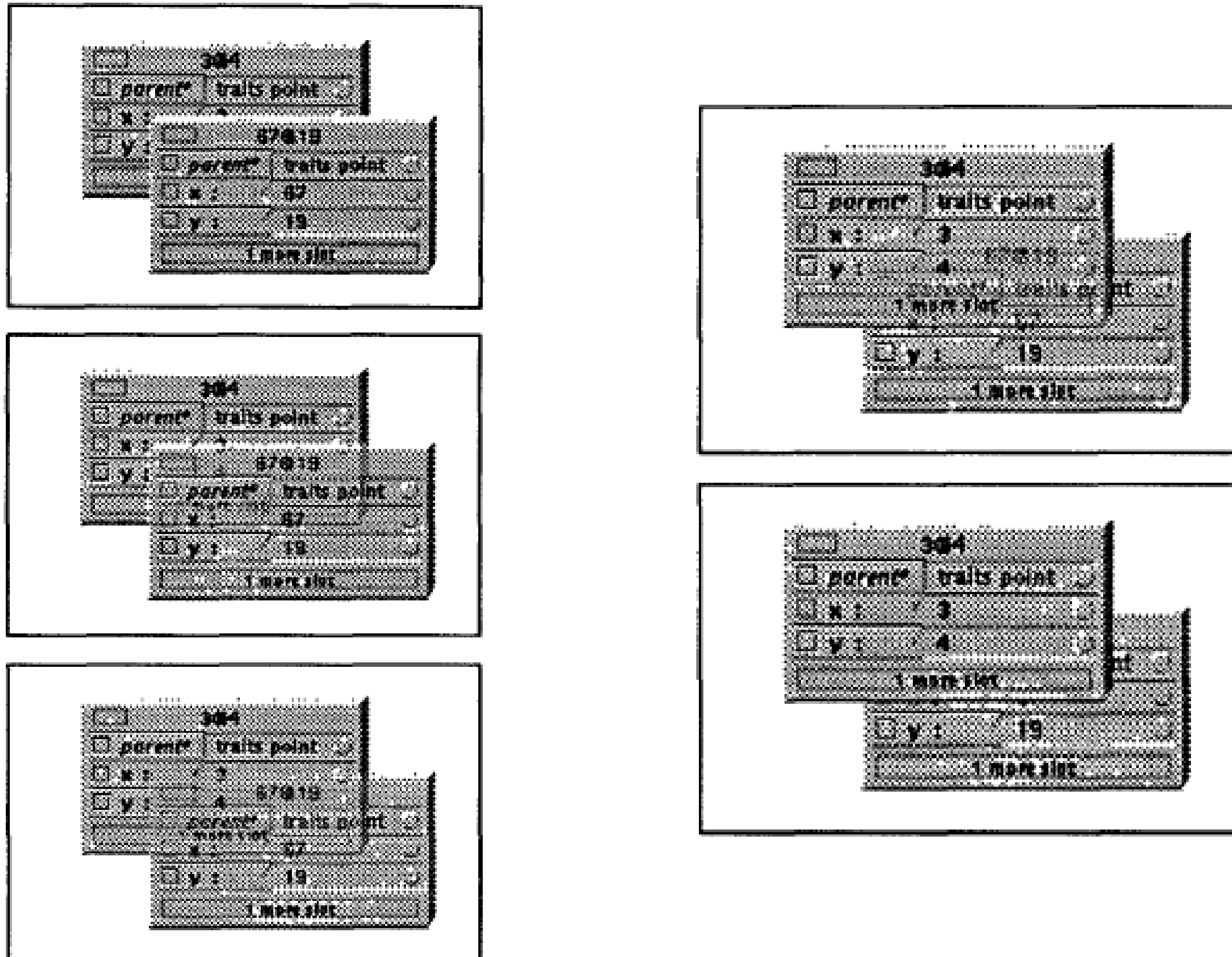


Solidity: Arrival and Departure





Solidity: Arrival and Departure



Exaggeration: Anticipation

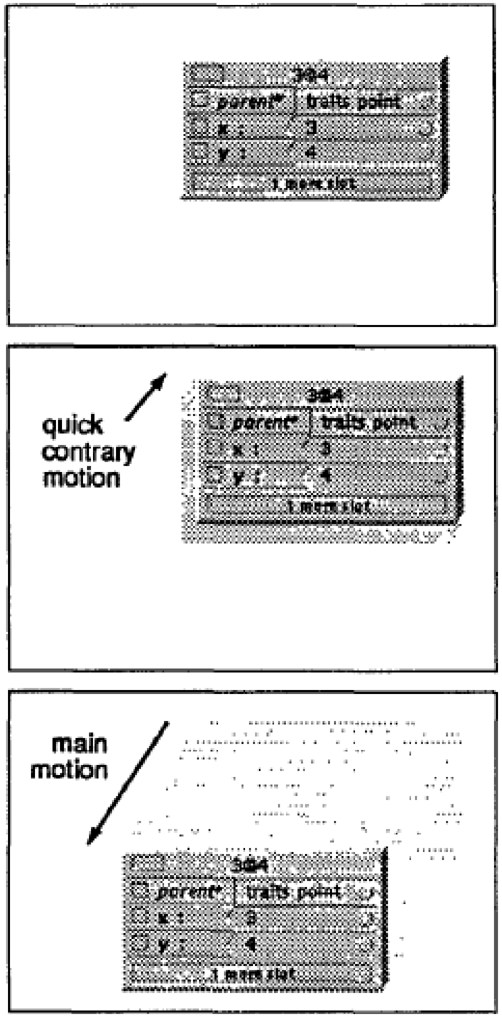


Figure 7. Objects anticipate major actions with a quick contrary motion that draws the user eye to the object in preparation for the main motion to come.

Reinforcement: Slow In Slow Out

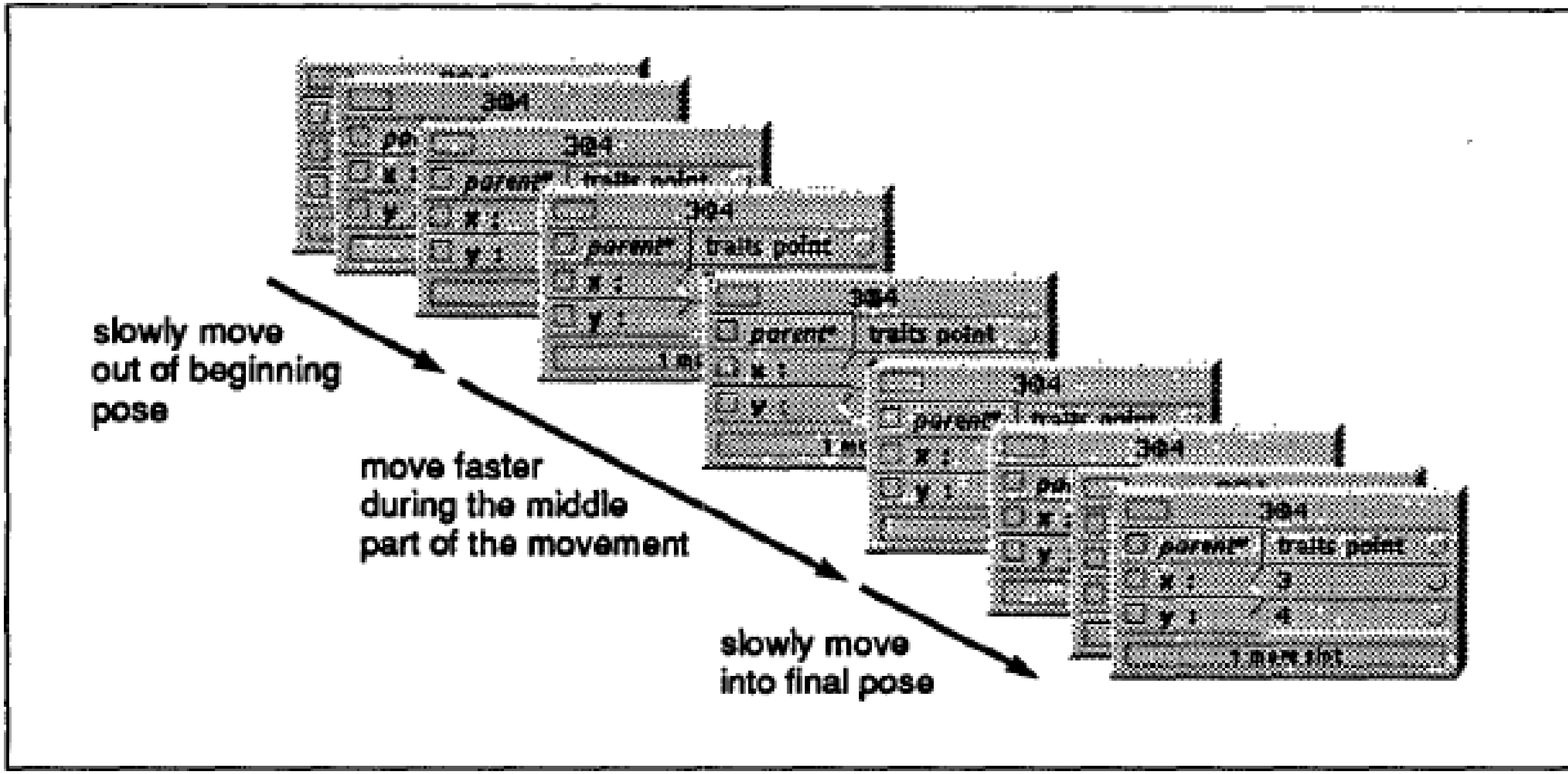


Figure 8. Objects ease out of their beginning poses and ease into their final poses. Although these motions are slower than that during the main portion of the movement, they are still quite fast.

Reinforcement: Arcs

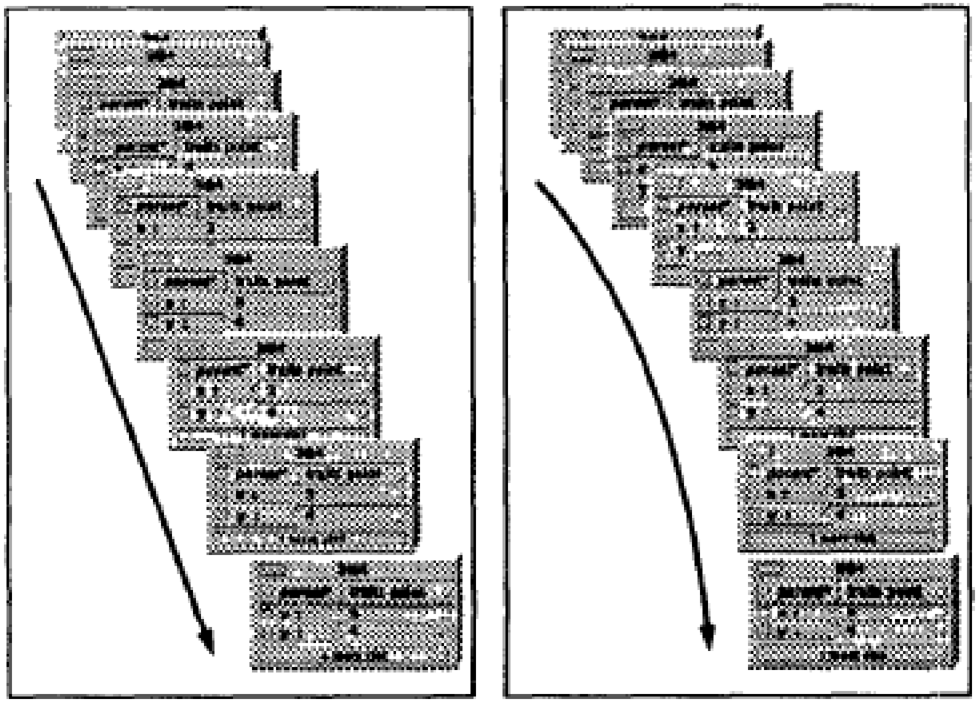


Figure 9. When objects travel under their own power (non-interactively), they move in arcs rather than straight lines.

Reinforcement: Follow Through

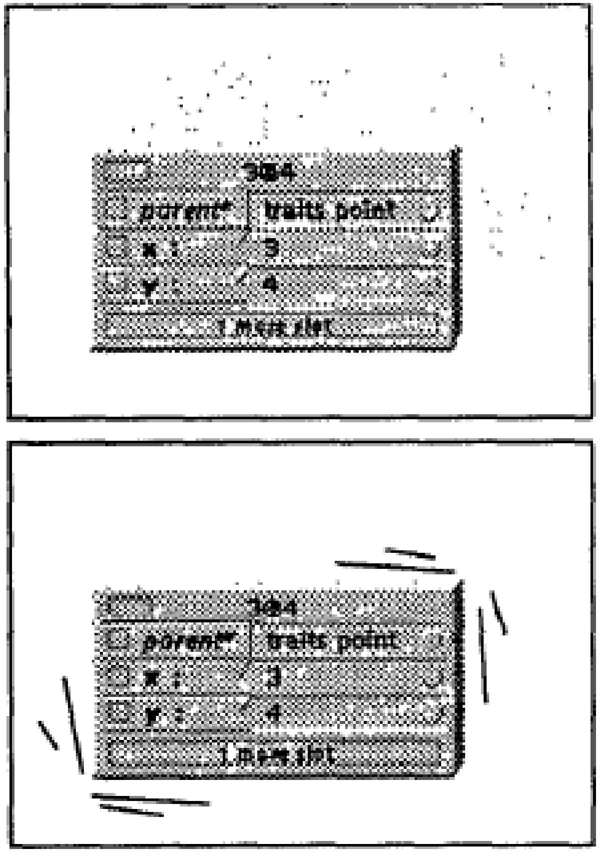


Figure 10. When objects come to a stop after moving on their own, they exhibit follow through in the form of wiggling back and forth quickly. This is just suggested by the "wiggle lines" in the figure—in actuality, the object moves back and forth, with motion blur.

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

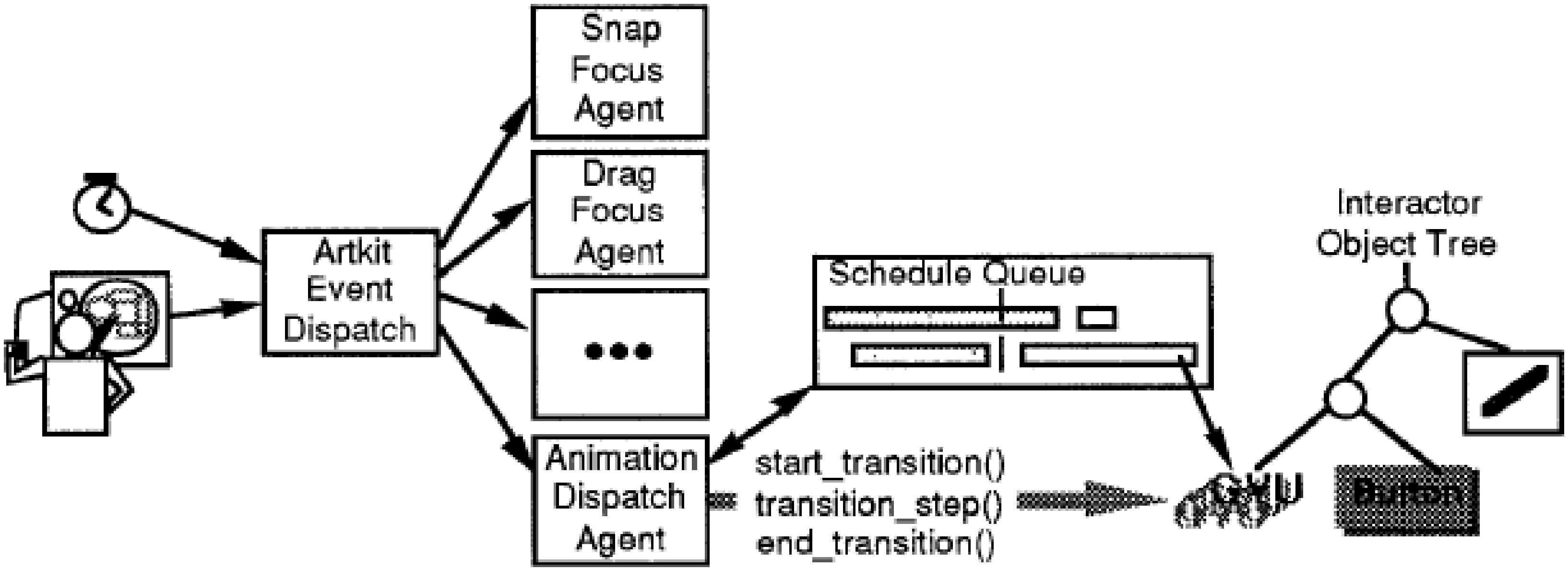


Figure 5. Animation Event Translation and Dispatch

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

Events and Animation

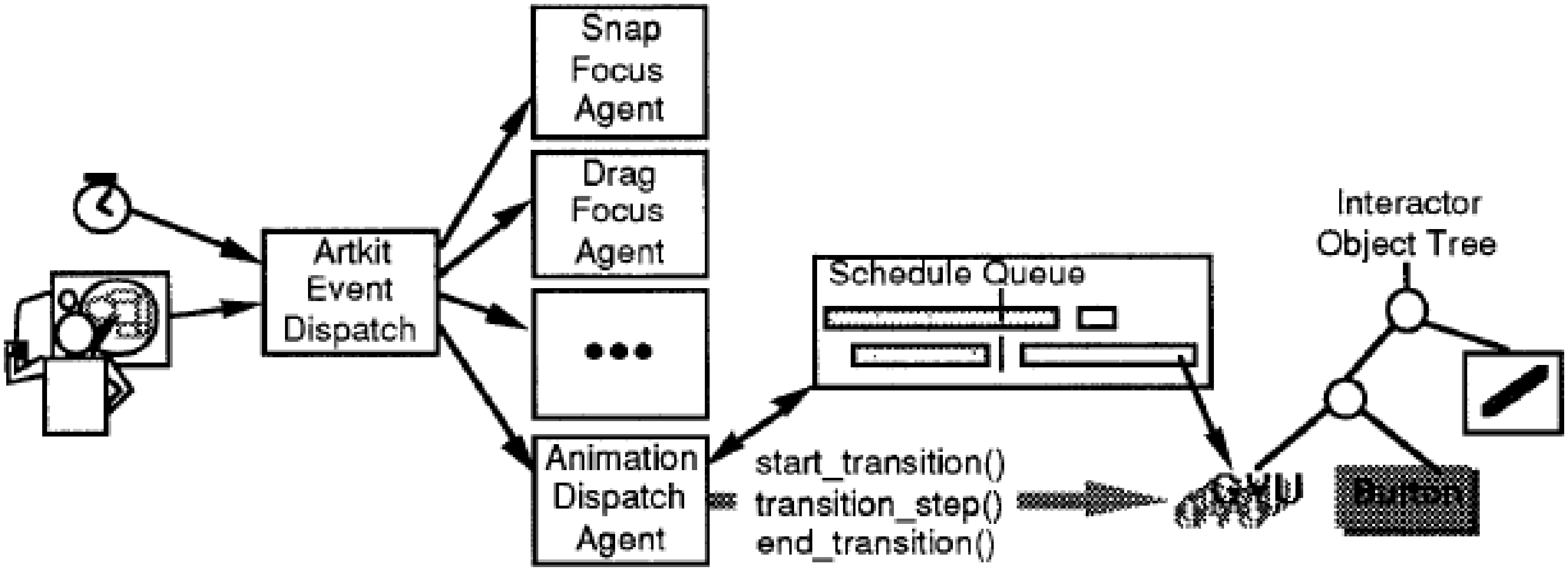


Figure 5. Animation Event Translation and Dispatch

Transition Object

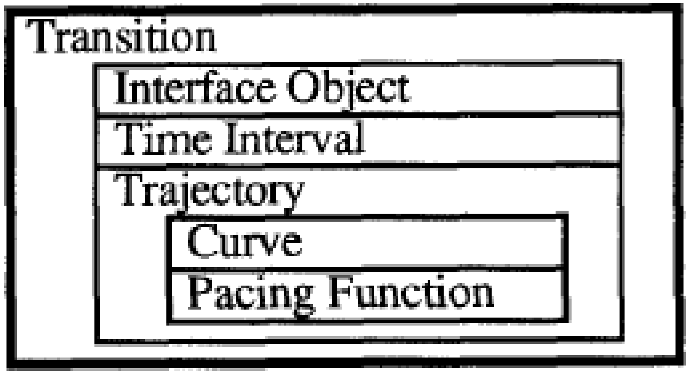


Figure 3. Parts of a Transition Object

Pacing Function

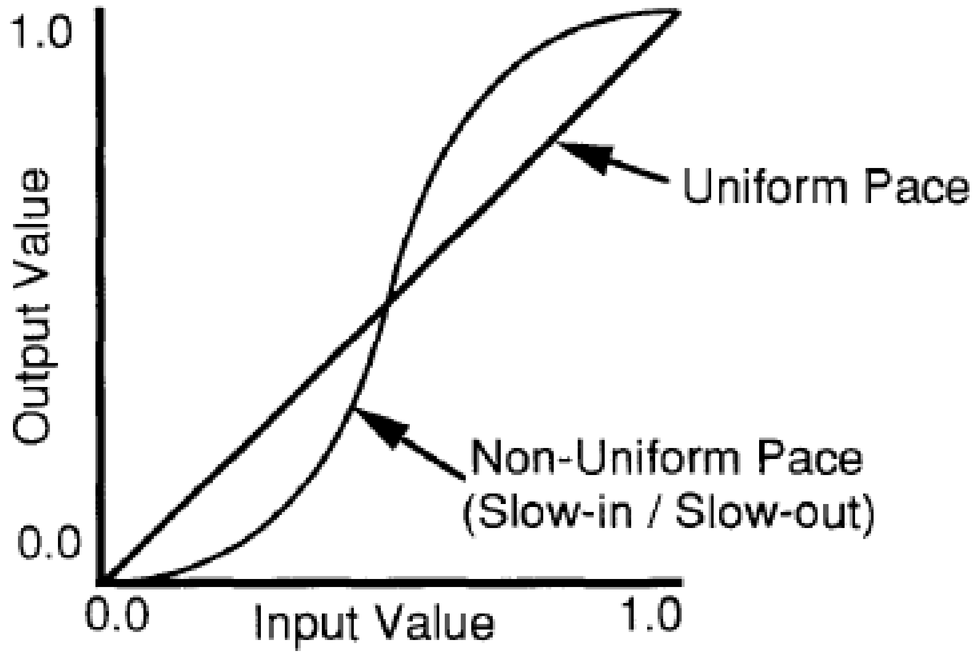


Figure 4. Two Example Pacing Functions

Computing a Frame

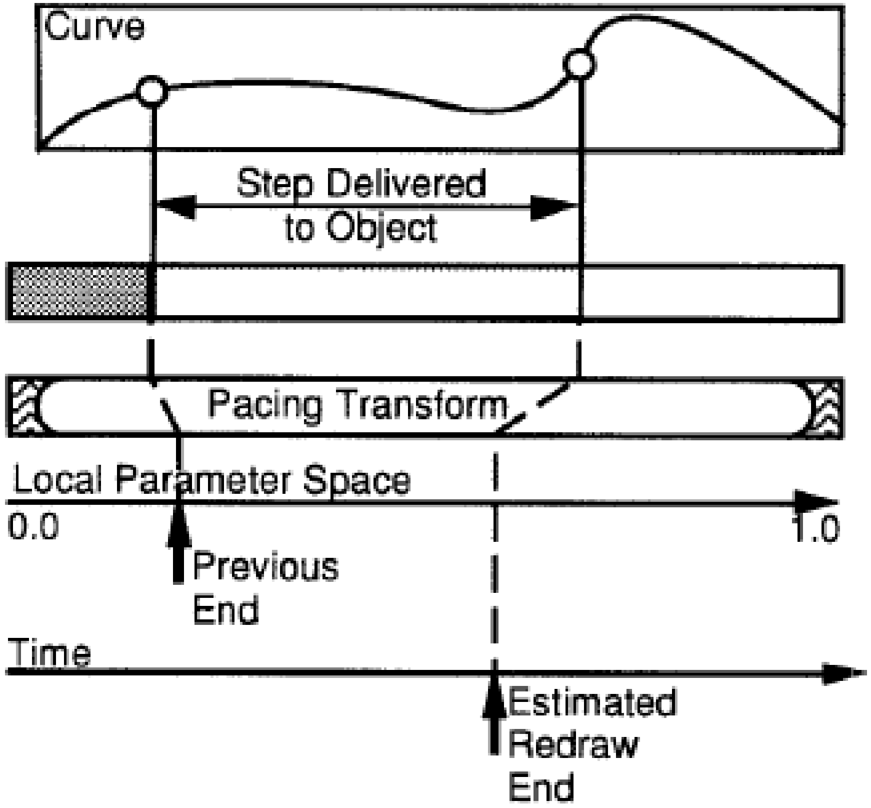


Figure 8. Translation from Time to Space

Animation Case Study

Based on increased understanding of how animation should be done in the interface, increasingly mature tools develop

Now built into major commercial toolkits (e.g., Microsoft’s WPF, JavaFX, jQuery)

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

Animation Case Study

The Kinetic Typography Engine: An Extensible System for Animating Expressive Text

Lee et al, 2002

http://dx.doi.org/10.1145/571985.571997

The Kinetic Typography Engine: An Extensible System for Animating Expressive Text

Johnny C. Lee*, Jodi Forlizzi*†, Scott E. Hudson*
*Human Computer Interaction Institute and †School of Design
Carnegie Mellon University,
Pittsburgh, PA 15213 USA
{ johnny, forlizzi, scott.hudson }@cs.cmu.edu

ABSTRACT
Kinetic typography – text that uses movement or other temporal change – has recently emerged as a new form of communication. As we hope to illustrate in this paper, kinetic typography can be seen as bringing some of the expressive power of film – such as its ability to convey emotion, portray compelling characters, and visually direct attention – to the strong communicative properties of text. Although kinetic typography offers substantial promise for expressive communications, it has not been widely exploited outside a few limited application areas (most notably in TV advertising). One of the reasons for this has been the lack of tools directly supporting it, and the accompanying difficulty in creating dynamic text. This paper presents a first step in remedying this situation – an extensible and robust system for animating text in a wide variety of forms. By supporting an appropriate set of carefully factored abstractions, this *engine* provides a relatively small set of components that can be plugged together to create a wide range of different expressions. It provides new techniques for automating effects used in traditional cartoon animation, and provides specific support for typographic manipulations.

KEYWORDS: kinetic typography, dynamic text, time-based presentation, automating animation effects

INTRODUCTION

The written word is one of humanity's most powerful and significant inventions. For over 4000 years, its basic communicative purpose has not changed. However, the method in which written communication is authored and presented has never stopped evolving. From cuneiform markings on clay tablets, to pen and parchment, to the Gutenberg press, to computers and the internet, technology has always provided text with new mediums to express itself. The explosion of available computing power has added a new possibility: *kinetic typography* – text that moves or otherwise changes over time.

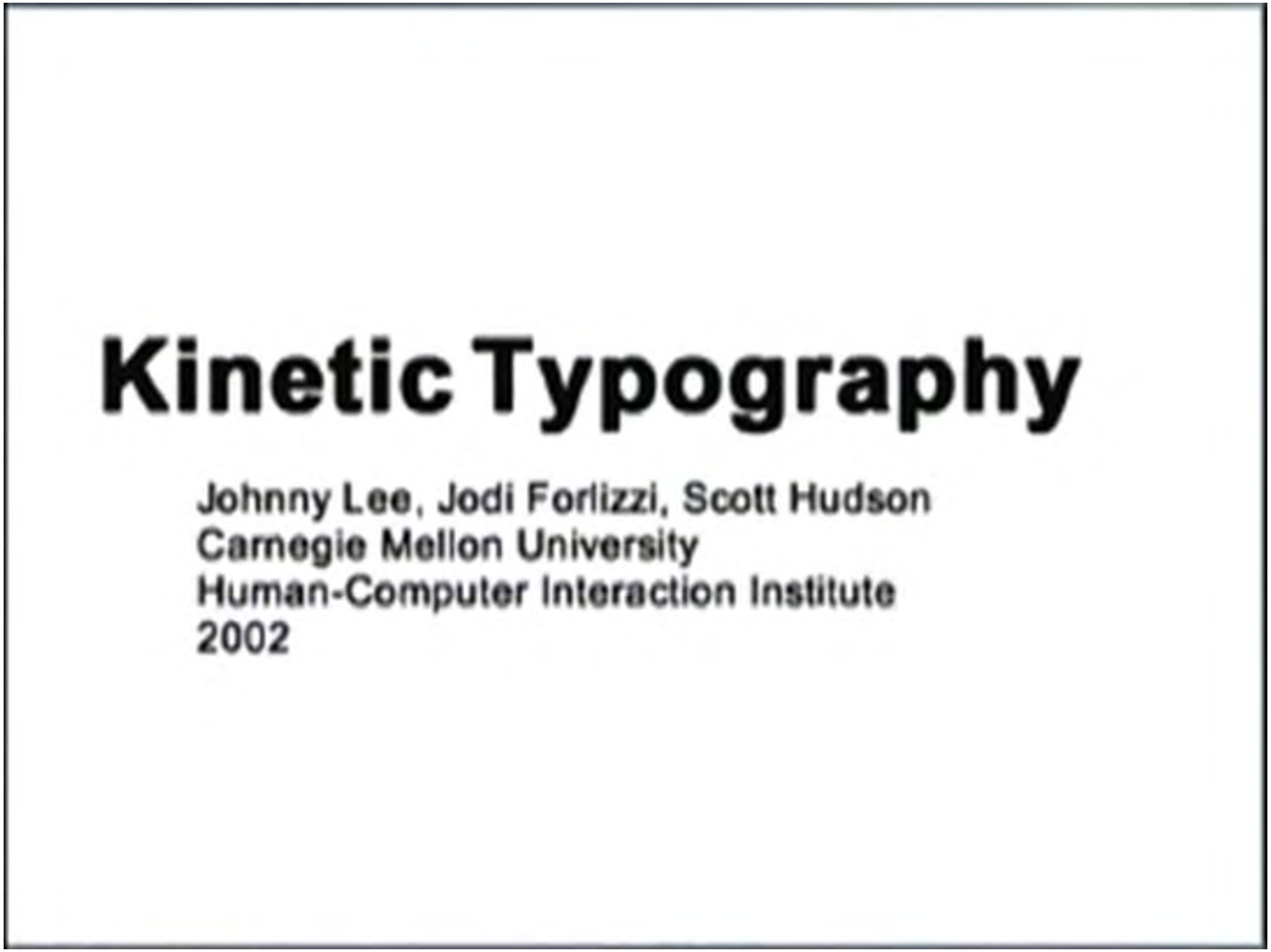
Kinetic typography can be seen as a vehicle for adding some of the properties of film to that of text. For example, kinetic typography can be effective in conveying a speaker's tone of voice, qualities of character, and affective (emotional) qualities of text [Ford97]. It may also allow for a different kind of engagement with the viewer than static text, and in some cases, may explicitly direct or manipulate the attention of the viewer.

In fact, the first known use of kinetic typography appeared in film – specifically, Saul Bass' opening credit sequence for Hitchcock's *North by Northwest* [Bass59] and later *Psycho* [Bass60]. This work stemmed in part from a desire to have the opening credits set the stage for the film by establishing a mood, rather than simply conveying the information of the credits. Use of kinetic typography is now commonplace for this purpose, and is also very heavily used in TV advertising where its ability to convey emotive content and direct the user's attention is generally a good match to the goals of advertising. We believe that if it can be made accessible via good tools, the power of kinetic typography can also be applied to benefit other areas of digital communications.

A second origin for time-based presentation of text comes independently from psychological studies of perception and reading. For example, [Mill87] studies perceptual effects of a number of text presentations, such as scrolling text. One of the most fruitful of these is a method known as *Rapid Serial Visual Presentation* (RSVP), where text is displayed one word at a time in a fixed position [Pott84]. Studies have shown that, because scanning eye movements are unnecessary when using RSVP, it can result in rapid reading without a need for special training. In addition, RSVP techniques provide advantages for designers because they allow words to be treated independently without regard to effects on adjacent text elements. Finally, RSVP can be seen as a means for trading time for space, potentially allowing large bodies of text to be shown at readable sizes on small displays.

Figures 1-3 illustrate some of the things that kinetic typography can do. (Please refer to the video proceedings for dynamic renditions of these figures.) Figure 1 shows two different renditions of the same words expressing a different emotional tone. As described by Ishizaki [Ishi97]:

Kinetic Typography Engine



Kinetic Typography Engine

Goals of Kinetic Type

- Emotional content
- Creation of characters
- Direction of attention

Based on existing work

Animation Composition

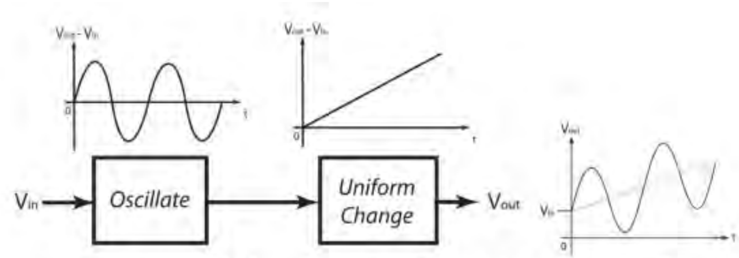


Figure 6. Waveform addition by chaining"

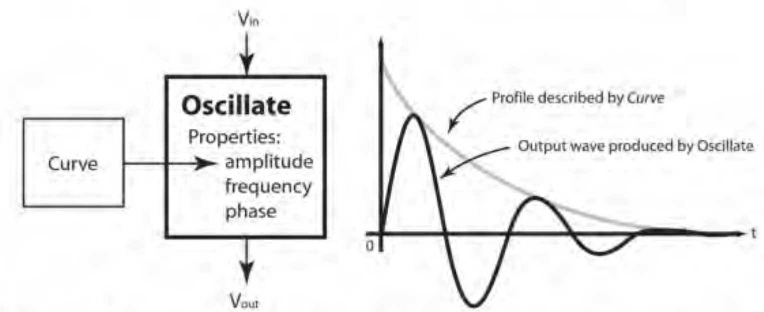


Figure 7. Waveform scaling by functional composition with amplitude

Animation Case Study

Prefuse: A Toolkit for Interactive Information Visualization

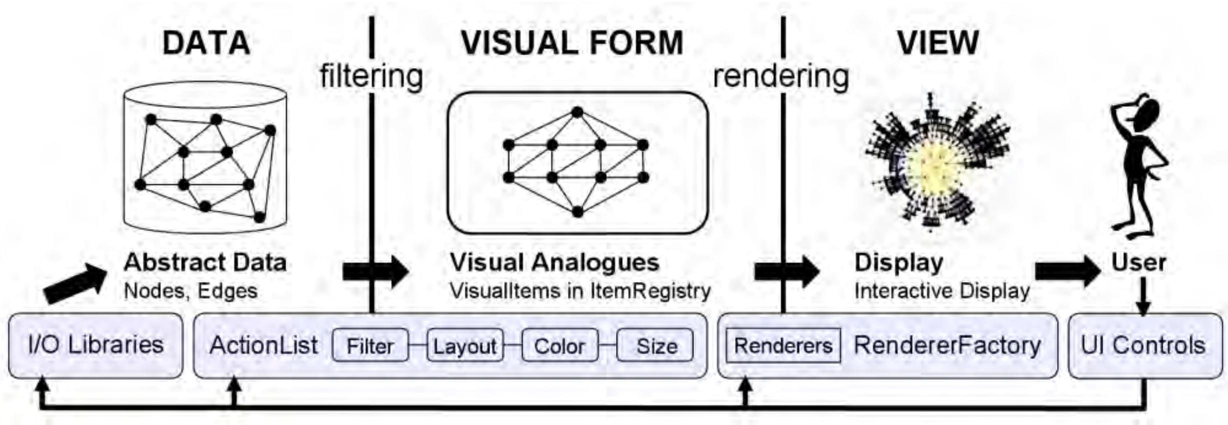
Heer et al, 2005

http://dx.doi.org/10.1145/1054972.1055031

D3: Data-Driven Documents

Bostock et al, 2011

http://dx.doi.org/10.1109/TVCG.2011.185



Tools and Interfaces

Why Interface Tools?

Case Study of Model-View-Controller

Case Study of Animation

Sapir-Whorf Hypothesis

Things I Hope You Learned

Sapir-Whorf Hypothesis

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

Tools not only make it easy to build certain types of software, they push you to think in terms of the types of software they can support

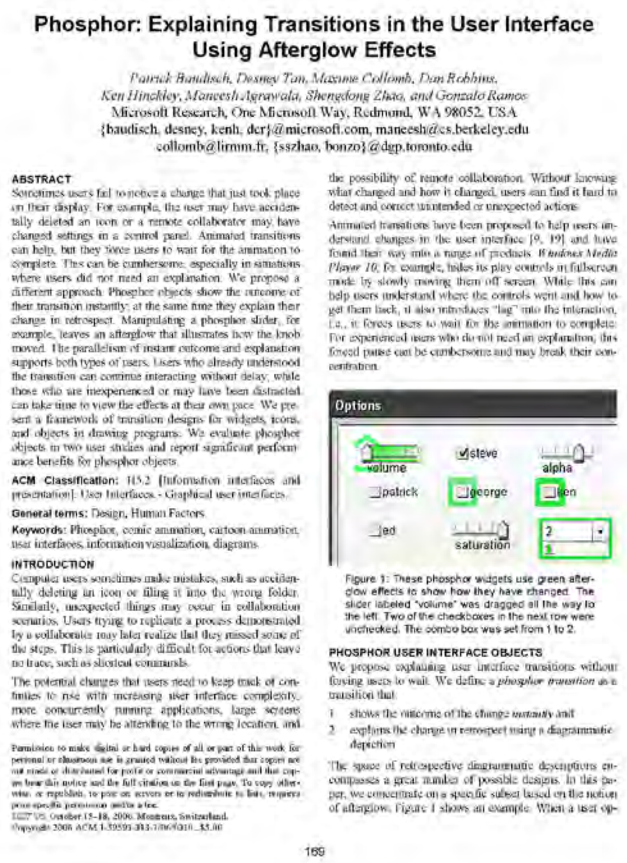
You must be aware of this when choosing tools, designing applications, and creating new tools

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.

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
follow interface transitions

The right speed is crucial

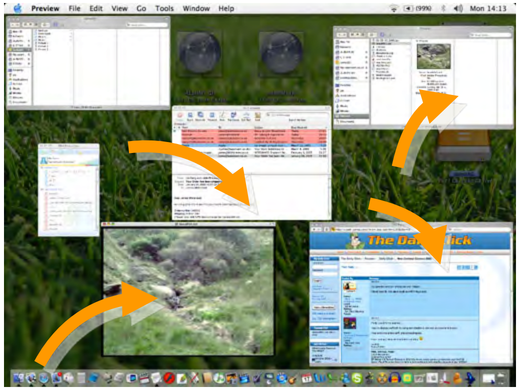
- Too fast increases error rate
- Too slow increases task time

The right speed depends on familiarity, distraction, and other such factors

- It cannot be determined

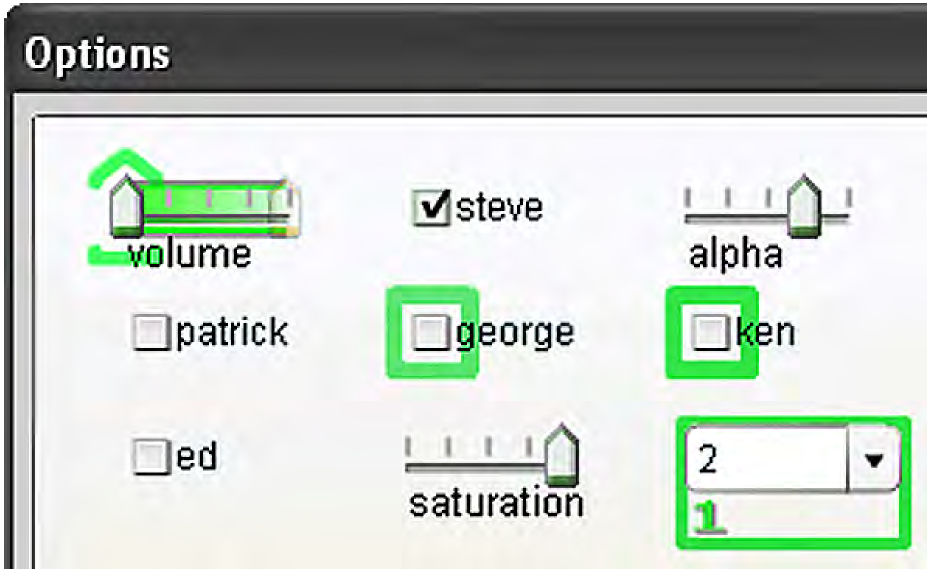
Windows Media Player

Apple Expose

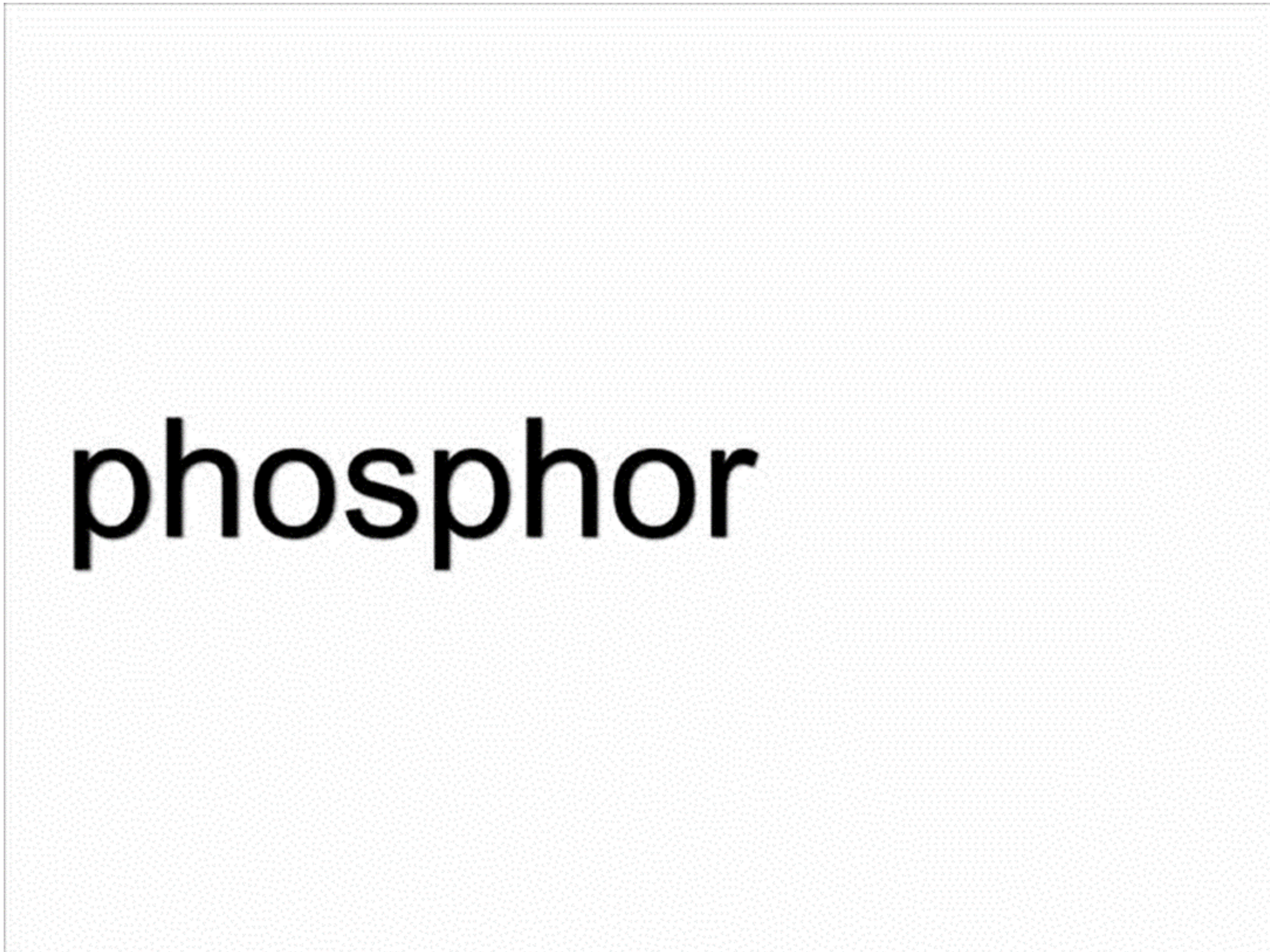


Phosphor

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

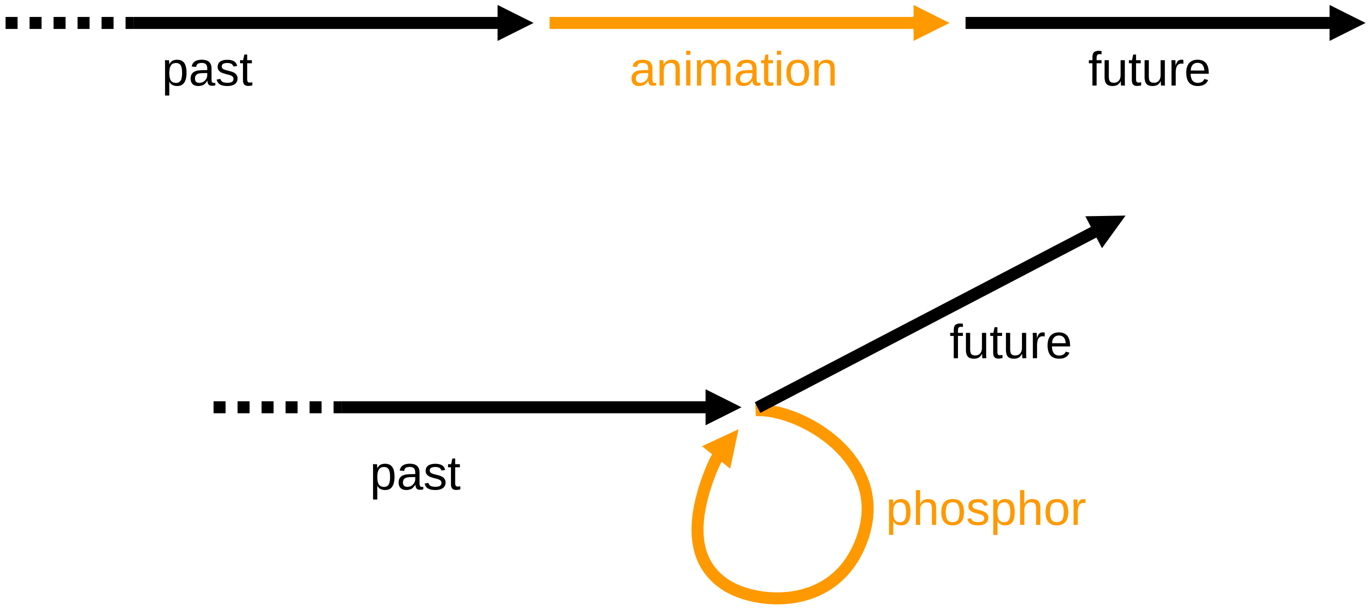


Phosphor



Challenging Assumptions of Tools

Phosphor breaks from the assumptions that current tools have evolved for transitions



Tools and Interfaces

Why Interface Tools?

Case Study of Model-View-Controller

Case Study of Animation

Sapir-Whorf Hypothesis

Things I Hope You Learned

Sapir-Whorf Hypothesis

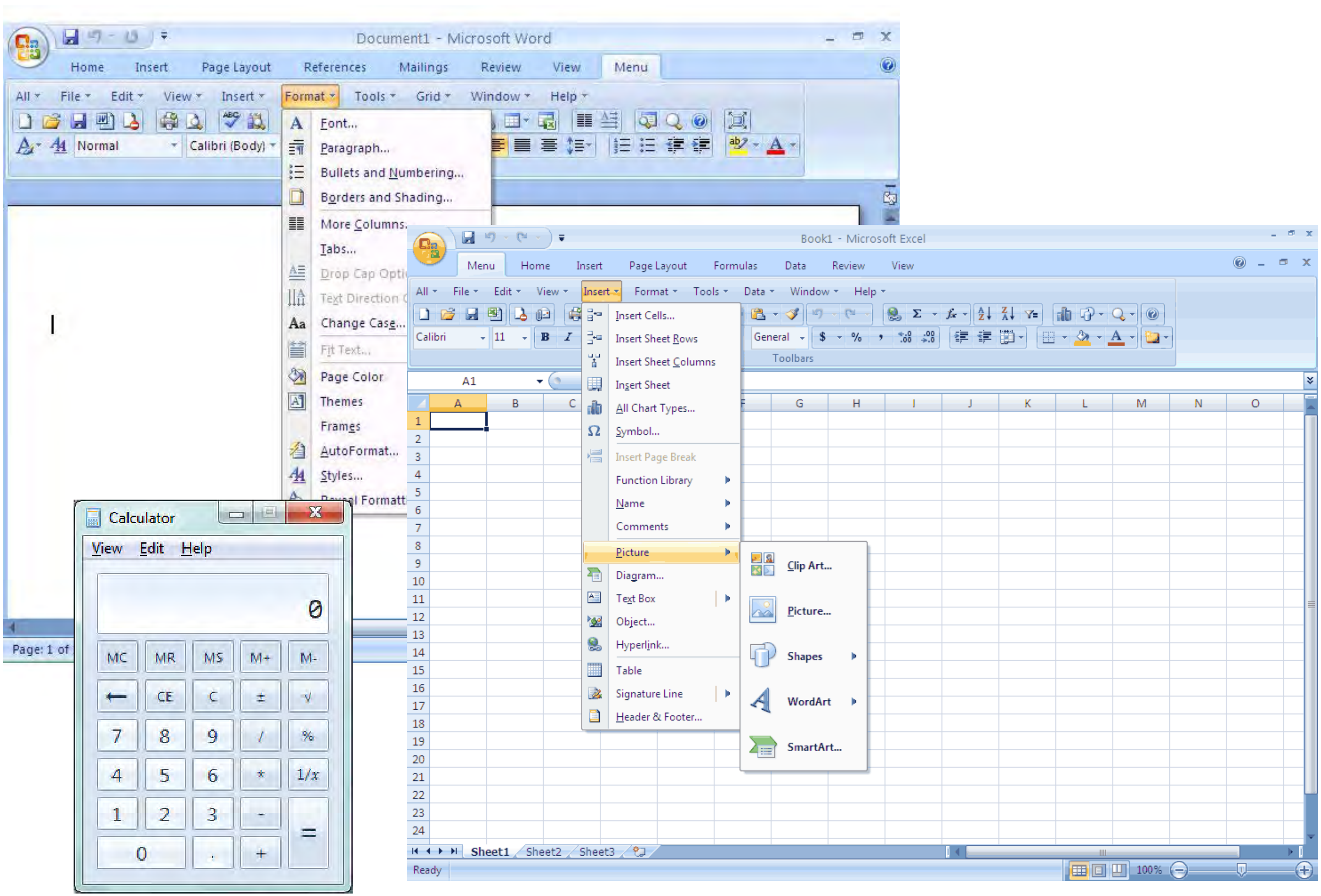
Roughly, some thoughts in one language cannot be stated or understood in another language

Our tools define the language of interaction

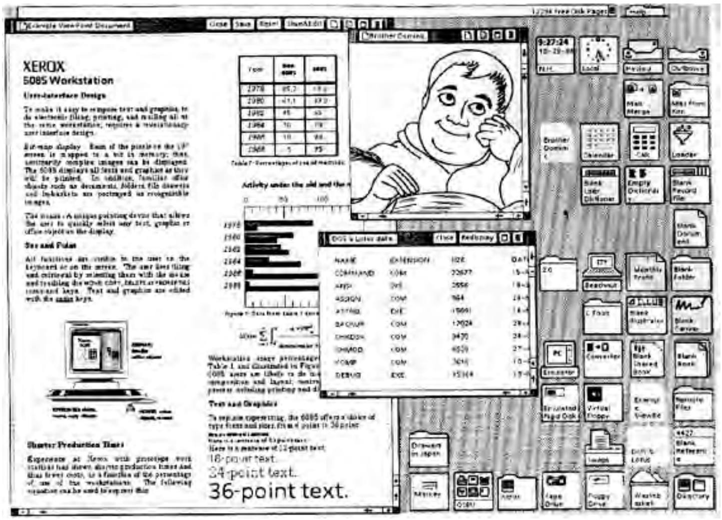
- Beyond the simple matter of code
- Frame how we think about possibilities

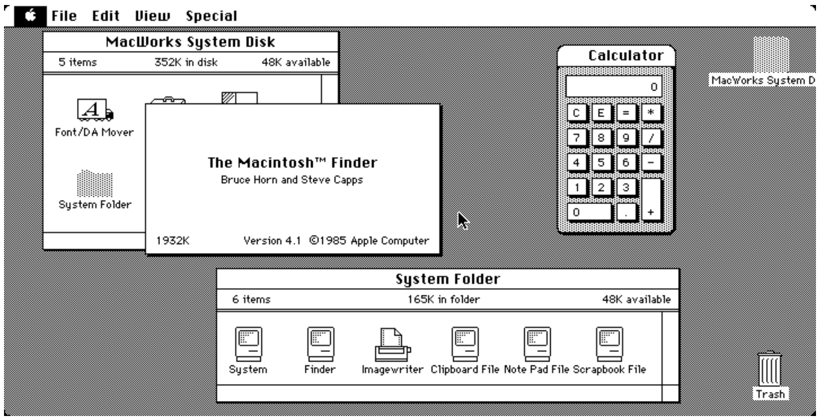
Myers, Hudson, Pausch. Past, Present, and Future of User Interface Software Tools. *TOCHI 2000.*

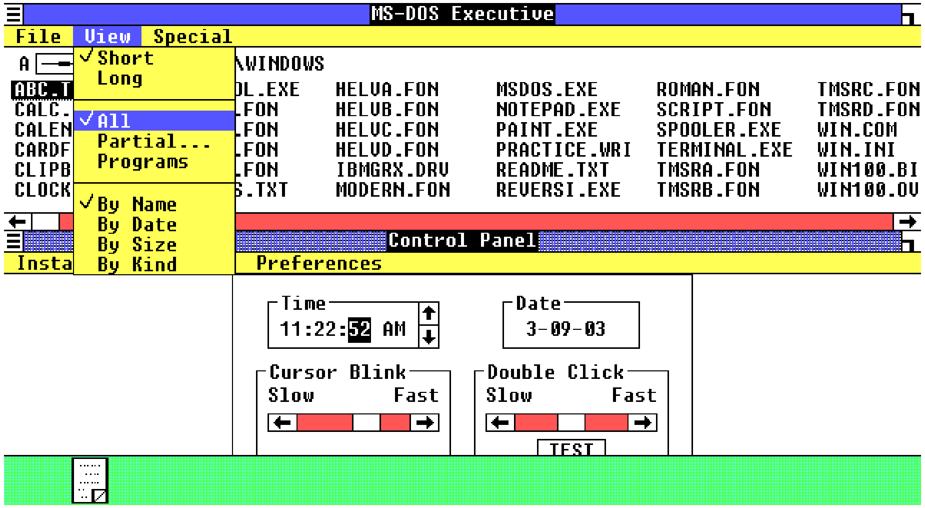
An Interaction Language



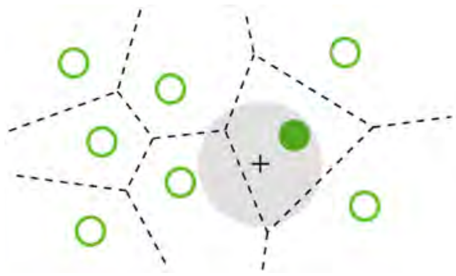
The Same Interaction Language



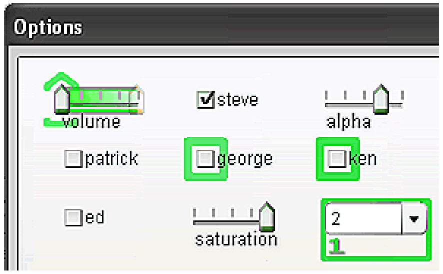




Some Proposed Interactions



Bubble Cursor

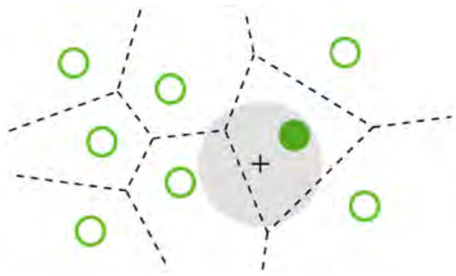


Phosphor

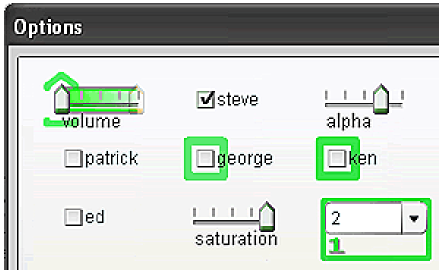
Sliding Widgets

Grossman, Balakrishnan. The Bubble Cursor ... *CHI 2005.*
Baudisch, Tan, Collomb, Robbins, Hinckley, Agrawala, Zhao, Ramos. Phosphor ... *UIST 2006.*
Moscovich. Contact Area Interaction with Sliding Widgets. *UIST 2009.*

Some Proposed Interactions



Bubble Cursor



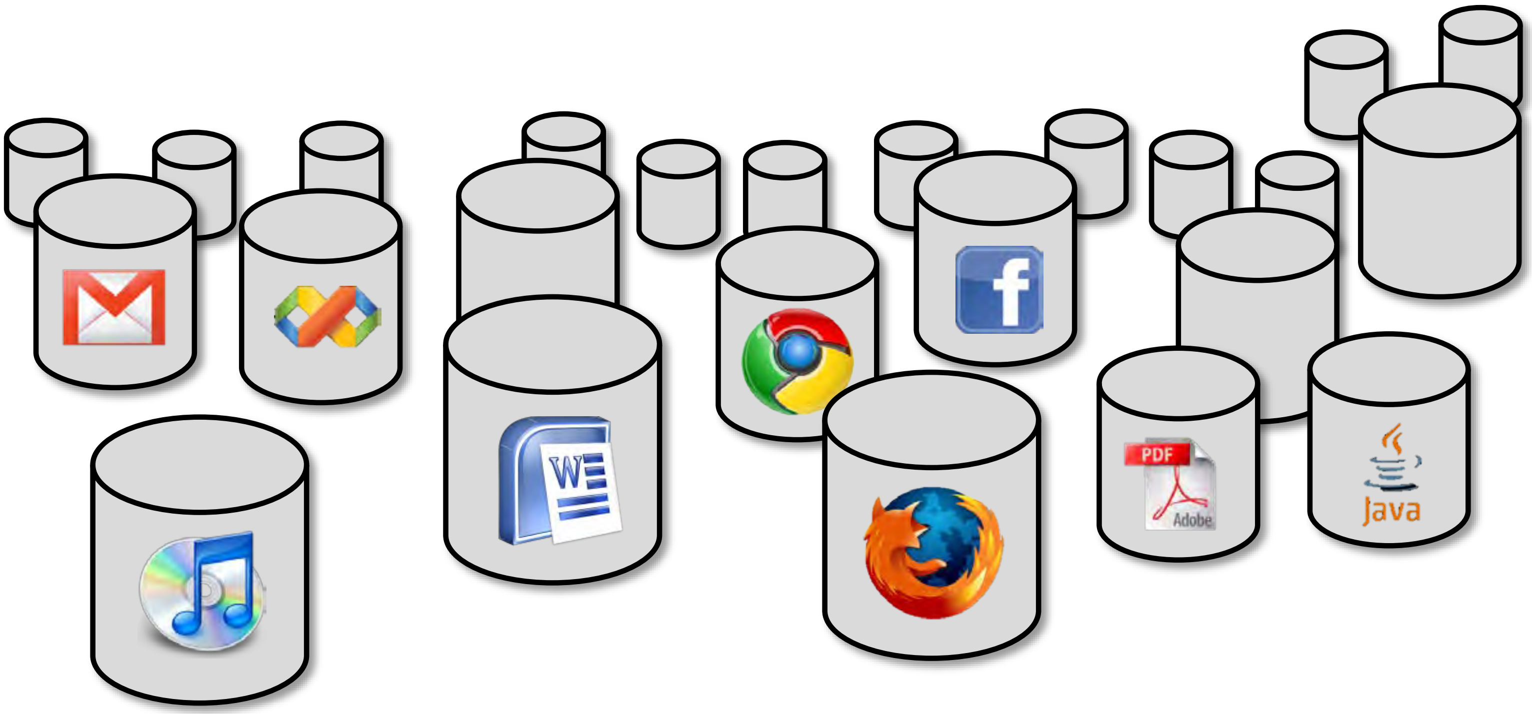
Phosphor

Sliding Widgets

None of these can be implemented in the established language of interaction

Grossman, Balakrishnan. The Bubble Cursor ... *CHI 2005.*
Baudisch, Tan, Collomb, Robbins, Hinckley, Agrawala, Zhao, Ramos. Phosphor ... *UIST 2006.*
Moscovich. Contact Area Interaction with Sliding Widgets. *UIST 2009.*

Interface Fragmentation

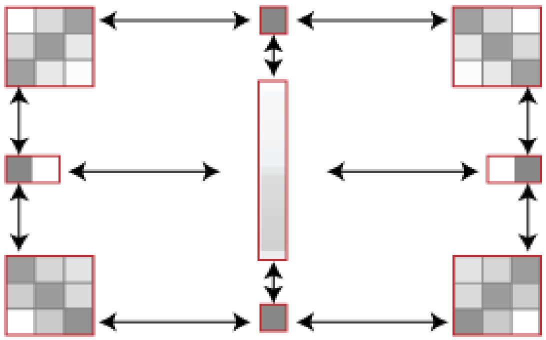


It is insufficient to innovate in any one interface

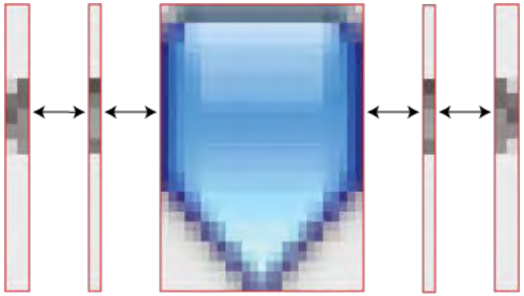
People use interfaces developed in many tools

Prefab

Pixel-based runtime modification of existing interfaces without their source or cooperation

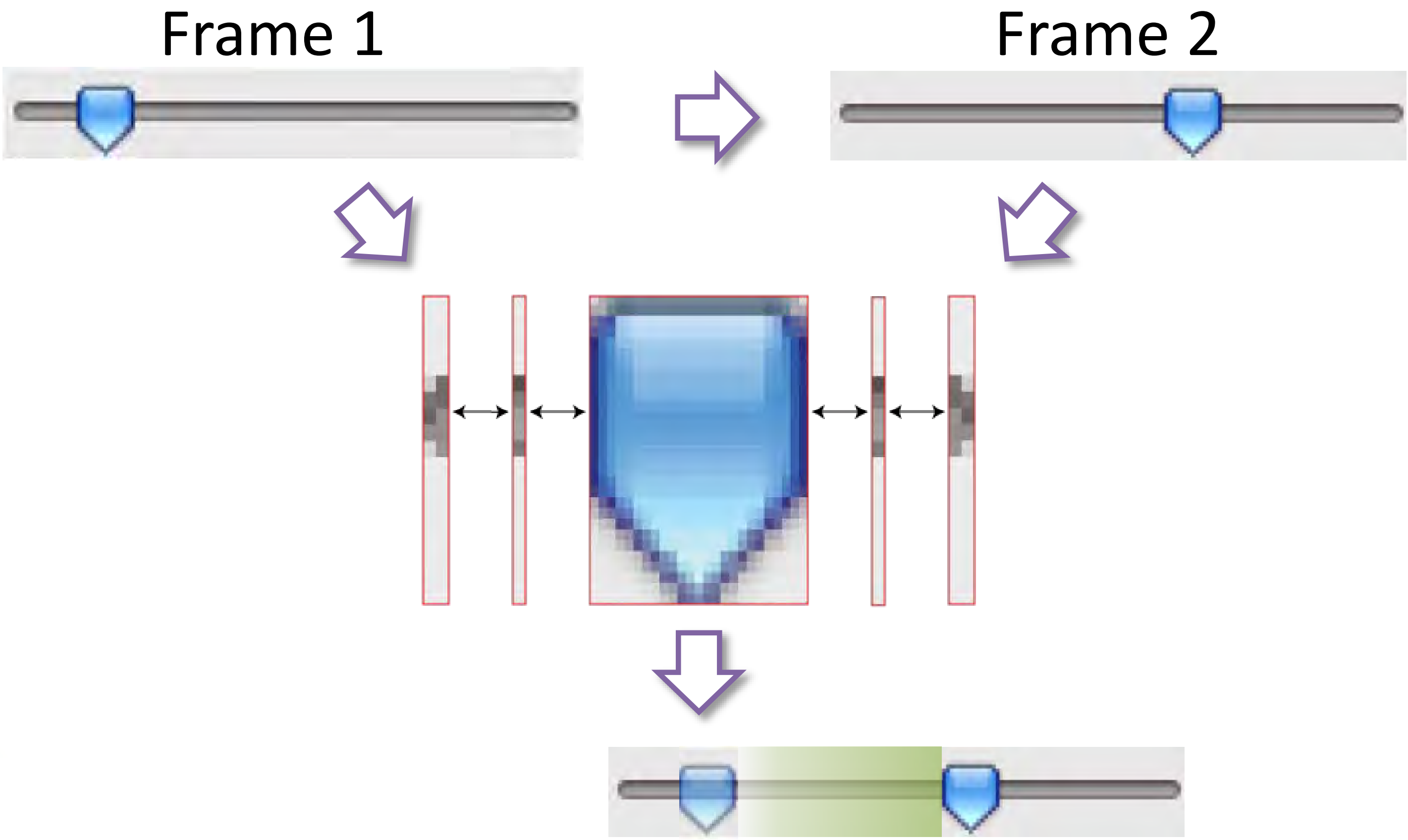


Unlocks interaction by allowing researchers to implement new ideas atop existing applications



Dixon, Fogarty. Prefab: Implementing Advanced Behaviors Using Pixel-Based Reverse Engineering of Interface Structure. *CHI 2010*.
Dixon, Leventhal, Fogarty. Content and Hierarchy in Pixel-Based Methods for Reverse Engineering Interface Structure. *CHI 2011*.
Dixon, Fogarty, Wobbrock. A General-Purpose Target-Aware Pointing Enhancement ... *CHI 2012*.
Dixon, Laput, Fogarty. Pixel-Based Methods for Widget State and Style in a Runtime Implementation of Sliding Widgets. *CHI 2014*.
Dixon, Nied, Fogarty. Prefab Layers and Annotations: Extensible Pixel-Based Interpretation of Graphical Interfaces. *UIST 2014*.

Phosphor Enhancement



Dixon, Fogarty. Prefab: Implementing Advanced Behaviors Using Pixel-Based Reverse Engineering of Interface Structure. *CHI 2010*.

Phosphor Enhancement

Dixon, Fogarty. Prefab: Implementing Advanced Behaviors Using Pixel-Based Reverse Engineering of Interface Structure. *CHI 2010*.

Phosphor Enhancement

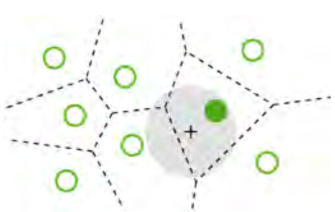
Dixon, Fogarty. Prefab: Implementing Advanced Behaviors Using Pixel-Based Reverse Engineering of Interface Structure. *CHI 2010*.

Sapir-Whorf Hypothesis

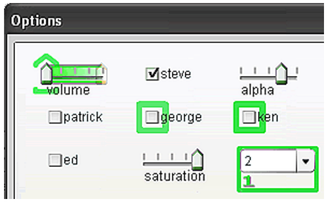
Roughly, some thoughts in one language cannot be stated or understood in another language

Our tools define the language of interaction

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



Bubble Cursor



Phosphor

Sliding Widgets

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

Rebuilding the Language

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

Olsen. Evaluating User Interface Systems Research. *UIST 2007.*

Informing the Next Language

Research explores the next generation of language, while being limited by the current

We therefore conflate:

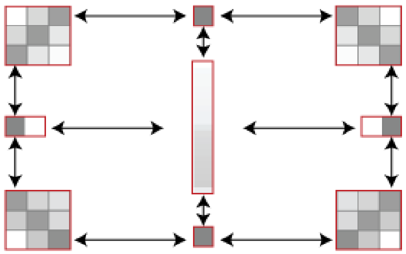
- Ideas
- Proof of Concept
- Engineering
- Implementation
- Broken Metaphors
- Unspeakably Dirty Hacks

Informing the Next Language

Research explores the next generation of language, while being limited by the current

We therefore conflate:

- Ideas
- Proof of Concept
- Engineering
- Implementation
- Broken Metaphors
- Unspeakably Dirty Hacks



Prefab is not just about ‘do everything with pixels’, but about exploring new possibilities in the current ecosystem of interface tools

Tools and Interfaces

Why Interface Tools?

Case Study of Model-View-Controller

Case Study of Animation

Sapir-Whorf Hypothesis

Things I Hope You Learned

Things I Hope You Learned

Tools embody expertise and assumptions

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

Fundamental tool terminology

- Threshold
- Ceiling
- Path of Least Resistance
- Moving Target

Things I Hope You Learned

Tools frame our design processes

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

We can and will move past our current tools

CSE 510: Advanced Topics in HCI

Interface Toolkits

James Fogarty
Daniel Epstein

Tuesday/Thursday
10:30 to 12:00

CSE 403