The Impact of Tutorials on Games of Varying Complexity

Erik Andersen, Eleanor O'Rourke, Yun-En Liu, Richard Snider, Jeff Lowdermilk, David Truong, Seth Cooper, and Zoran Popović
Center for Game Science
Department of Computer Science & Engineering, University of Washington
{eland,eorourke,yunliu,mrsoviet,jeff,djtruong,scooper,zoran}@cs.washington.edu

ABSTRACT
One of the key challenges of video game design is teaching new players how to play. Although game developers frequently use tutorials to teach game mechanics, little is known about how tutorials affect game learnability and player engagement. Seeking to estimate this value, we implemented eight tutorial designs in three video games of varying complexity and evaluated their effects on player engagement and retention. The results of our multivariate study of over 45,000 players show that the usefulness of tutorials depends greatly on game complexity. Although tutorials increased play time by as much as 29% in the most complex game, they did not significantly improve player engagement in the two simpler games. Our results suggest that investment in tutorials may not be justified for games with mechanics that can be discovered through experimentation.

Author Keywords
games; analytics; tutorials; multivariate testing

ACM Classification Keywords
H.5.0 Information interfaces and presentation: General

INTRODUCTION
Teaching new players how to play a game is challenging but crucial for engaging and retaining players. As a result, game designers frequently utilize tutorials to aid the learning process. Since tutorials are typically the first part of a game that new players encounter, effective tutorial design is important for retaining new players. Although modern games employ a wide varety of tutorial styles, including hints, help buttons, manuals, and interactive training challenges, the relative effectiveness of these styles is not well understood. As a result, designers must rely on intuition, personal experience, existing examples, and extensive user testing when designing tutorials. A deeper understanding of how tutorial design decisions impact player engagement and retention would help game developers create more learnable interfaces and spend valuable resources more wisely.

CHI'12, May 5–10, 2012, Austin, Texas, USA.

Learnability is widely accepted as a central component of usability, and it has been studied in the field of Human Computer Interaction for decades [11, 20, 23]. In this paper, we draw tutorial ideas from the HCI literature and present a large-scale comparative study of their effectiveness in video game tutorials. Although many factors influence tutorial design, we chose to focus on four specific tutorial characteristics: the presence of tutorials, the context-sensitivity of tutorial instructions, the freedom given to users during the tutorial, and the availability of additional help on demand. We examined the importance of these characteristics by conducting a multivariate experiment across three games that we developed: Refraction, Hello Worlds, and Foldit. These games belong to different genres and vary in complexity.

We present results gathered from over 45,000 players showing that the value of tutorials depends greatly on the complexity of the game. In Foldit, the most complex and least conventional game we studied, tutorials increased play time by as much as 29% and player progress by as much as 75%. However, we found that tutorials had a surprisingly negligible effect on player engagement in Hello Worlds and Refraction, which are less complex and more similar to other games in their respective genres. Giving tutorial instructions as closely as possible to when they were needed, rather than out of context in an up-front manual, increased play time by 16% and progress by 40% in Foldit but had no effect in the other two games. We found no evidence to support the claim that restricting player freedom in order to focus the player's attention on a target concept improves learnability. Providing help on-demand improved player engagement in Foldit, but had no effect in Hello Worlds and even negative effects in Refraction. Our results suggest that investment in tutorials may not be justified for games that can be learned through experimentation, and that the players of these games, at least, seem to learn more from experimentation than from reading text.

Tutorial presence
The first feature we considered was tutorial *presence*: whether the game provides a tutorial to its players or not. Although some successful games provide no instructions to players, such as Tetris (Pajitnov 1984), Pac-Man (Namco 1980), and Super Mario Brothers (Nintendo 1985), the use of tutorials has become very common. Therefore, we wanted to examine the impact of tutorial presence.

Hypothesis 1: *Games with tutorials will exhibit better player engagement and retention than games without tutorials.*

We examined this hypothesis by comparing versions of each game with and without tutorials.

Context-sensitivity

Another feature we considered was the *context-sensitivity* of tutorials. Existing tutorials can be divided into two categories: those that provide contextually relevant suggestions from within the application interface and those that provide documentation outside of the application context. HCI researchers have studied both types of tutorials extensively; however, the effect of context-sensitivity on tutorial success is unclear, and the existing literature does not show a clear preference for either type of information.

Historically, tutorials have been provided through primarily textual documentation that is accessed outside the application context. While paper manuals and online documentation require minimal resources to generate, they present a number of challenges for users, who struggle to keep track of instructions when switching between the tutorial and the application [1, 15, 17]. As a result, many researchers have worked on improving external software documentation by incorporating screenshots, graphics, animations, and annotations into textual content [9, 12, 13, 21].

Despite these improvements, many researchers argue that contextually relevant tutorials have greater potential for improving application learnability [4, 14, 10]. This idea is supported by situated learning, a popular learning model in education based around teaching concepts in the same context in which they will be applied [18]. Game expert James Gee also highlights the importance of teaching new mechanics just before the player needs to use them, rather than presenting them out of context [7]. Tooltips, which provide brief textual help when the user hovers over an interface component, are one of the most successful forms of contextual help [6]. On the other hand, context-sensitive help can be more frustrating than helpful, as shown by Microsoft's Office Assistant [24].

Commercial video games have used both context-sensitive and context-insensitive tutorials successfully. Up-front manuals, which present game mechanics out-of-context at the very beginning of a game, can be seen in the popular online Flash games Cursed Treasure[1] and Epic Battle Fantasy 3[2]. Other games present instructions in-context, just as the player needs to use them. This strategy can be seen in the highly-rated Flash game In the Company of Myself[3] and the popular console game Portal (Valve Corporation). To learn more about the value of context-sensitivity in game tutorials, we included both styles of presentation.

Hypothesis 2: *Tutorials that present instructions in context will be more effective than tutorials that present information out of context.*

In this study, we compare tutorials that present information out of context in an up-front manual and tutorials that present information in context just before the player needs to use a particular concept.

Freedom

We also considered tutorial *freedom*, or the degree to which tutorials force users to perform the mechanics being described. While requiring users to practice specific game mechanics could improve tutorial effectiveness, it could also frustrate players. As a result, it is not clear if and when reducing player freedom improves engagement.

Game designers have often cited the importance of allowing users to experiment while learning new concepts [7, 22]. James Gee argues that such experimentation is most effective when it occurs in a safe environment where functionality is reduced and mistakes are not punished [7]. Software learnability researchers Kelleher and Pausch argue that restricting user freedom improves tutorial performance. They found that an interactive tutorial for the Alice computer programming environment that restricts user freedom, called Stencils, helped users finish the tutorial sequence more quickly and with fewer errors than a paper-based tutorial with the same content [14]. How different types of interfaces are affected by reduced player freedom, however, is currently unknown.

Many video games provide players with complete freedom during the tutorial. Braid (Number None 2008), for instance, provides the player with background suggestions for keys to press during gameplay, but does not force the player to perform the suggested action. At the same time, the use of more restrictive stenciling strategies is also popular, and can be found in PopCap's tower defense game Plants vs. Zombies (PopCap 2009), the simulation game SimCity 4 (Maxis 2003), and Facebook game Cityville (Zynga 2010). We included tutorials of both types in order to determine the value of restricting player freedom.

Hypothesis 3: *Tutorials that restrict player freedom improve engagement and retention.*

In this study, we compare tutorials that restrict player freedom using the stenciling technique proposed by Kelleher and Pausch [14] and those that give players complete freedom to ignore the tutorial content.

Availability of help

Finally, we considered the *availability of help*. Many applications provide access to documentation on-demand through help buttons, but it is unclear whether this type of information has a strong effect on interface learnability. Although it seems likely that providing access to on-demand help would positively impact learnability, it is not known how much users take advantage of on-demand information when it is available.

James Gee argues that manuals are most effective when they are accessed on-demand, after the user has become familiar with the basic interface elements and has developed specific questions [7]. Commercial software applications rely heavily on the availability of on-demand documentation, and help

[1] http://www.kongregate.com/games/IriySoft/cursed-treasure-dont-touch-my-gems

[2] http://www.kongregate.com/games/kupo707/epic-battle-fantasy-3

[3] http://www.kongregate.com/games/2DArray/the-company-of-myself

buttons can be found in many ubiquitous products such as Microsoft Word (Microsoft Inc. 2010) and iTunes (Apple Inc. 2001). On-demand tutorials are also common in games. For example, help buttons are available in the Flash game Epic Battle Fantasy 3 and the real time strategy game Starcraft 2 (Blizzard 2010). To determine the impact of on-demand help in games, we included tutorials with and without this feature.

Hypothesis 4: *Having on-demand access to help improves player retention.*

In this study, we compare games that provide access to tutorial manuals on-demand through a help button with games that do not, both with and without other types of tutorial instructions.

STUDY METHODOLOGY

Over the past decade, researchers have begun to use the Internet to collect large quantities of data on player behavior. This has given rise to a data-driven evaluation methodology that has been widely used by companies like Amazon, Microsoft, and Google to support interface design decisions [16][19][8]. Such evaluation models frequently use simple A/B testing, in which a baseline sample is compared to a variety of single-variable test samples. This methodology has been used to evaluate games as well; Andersen et al. used A/B testing to show that gameplay affects average play time more than animations and audio [2], and that secondary game objectives are most effective when they support the primary objectives [3].

In this study, however, we wanted to compare a large variety of tutorial designs defined by multiple variables. We also had a limited understanding of which combinations of tutorial characteristics would be most effective. As a result, we chose to perform a multivariate experiment considering four independent variables and a total of eight conditions, rather than using a simple A/B test.

Our games

We performed tutorial experiments on three games developed by our research group: Refraction, Hello Worlds, and Foldit. All three games contain puzzle elements, but otherwise differ greatly in their format, complexity, and player bases. We consider two of the games, Refraction and Hello Worlds, to be "casual" games. We define "casual" games as those that rely on familiar game mechanics, take only a few hours to complete, and do not require a download.

Refraction, which involves solving spacial puzzles by splitting lasers into fractional amounts, was originally designed as a math-learning game. Each level is played on a grid that contains laser sources, target spaceships, and asteroids, as shown in Figure 1. Each target spaceship requires a fractional amount of laser power, indicated by a yellow number on the ship. The player can satisfy the targets by placing pieces that change the laser direction and pieces that split the laser into two or three equal parts. All targets must be correctly satisfied at the same time to win.

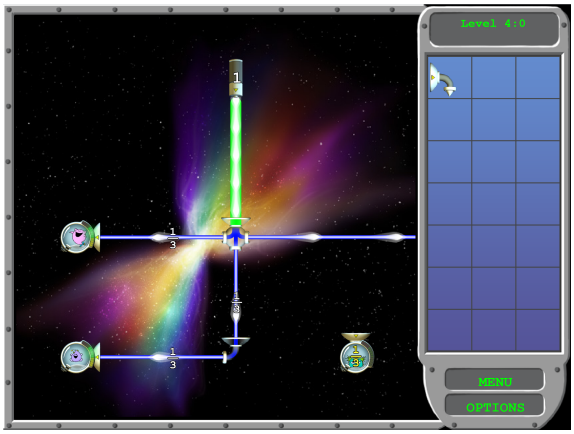


Figure 1. A level of Refraction. The goal is to use the pieces on the right to split lasers into fractional pieces and redirect them to satisfy the target spaceships. The user can pick up and put down pieces by clicking on them. The pieces are designed to be as intuitive as possible. The grid interface, pipe-flow game mechanics, and spatial reasoning puzzles are similar to many other puzzle games.

Although Refraction was originally designed to teach math concepts, the game has found success with an adult audience on the popular Flash website Kongregate, where it has been played over 480,000 times since its release in September 2010. The use of fractions as a game mechanic is relatively uncommon, but similar spacial reasoning puzzles and pipe flow mechanics appear in many other games on Kongregate and elsewhere. The artwork for the laser sources, target spaceships, asteroids, and laser manipulators was designed to utilize affordances to make the functionality of each piece as clear as possible. Refraction is freely available and can be played by anyone with a web browser.

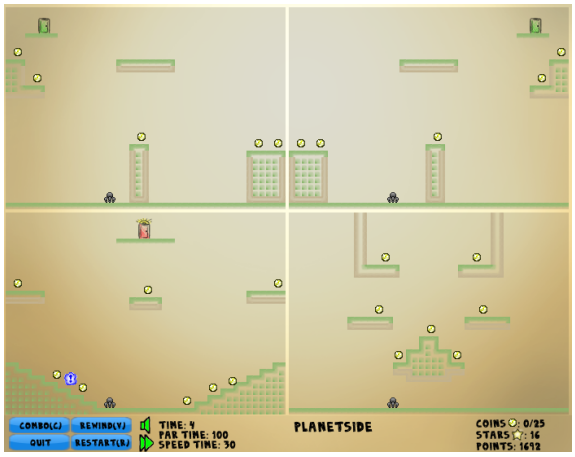


Figure 2. A level of Hello Worlds, a puzzle-platformer game. The game interface is similar to thousands of other games in the platformer genre, except that the player exists in multiple worlds at the same time. As a result, the basic game mechanics for navigating using keyboard input may be familiar to players. The player advances through each level by opening and closing doors, a common goal mechanic.

Hello Worlds is a puzzle game that offers a simple twist on the standard platformer genre. Platformers typically involve navigating a character through a two-dimensional world by running, jumping, and avoiding obstacles. In Hello Worlds,

the character exists in multiple worlds at the same time, and interacts with the obstacles in each world simultaneously, as shown in Figure 2. The player must find a path through the worlds to reach various doors that open and close worlds or complete the level. While the puzzle element of Hello Worlds is uncommon, the basic game mechanics for navigating using keyboard input are used in many existing platformers. Hello Worlds is available for free on the Flash game website Kongregate, and has been played over 1,300,000 times since its release in May 2010.

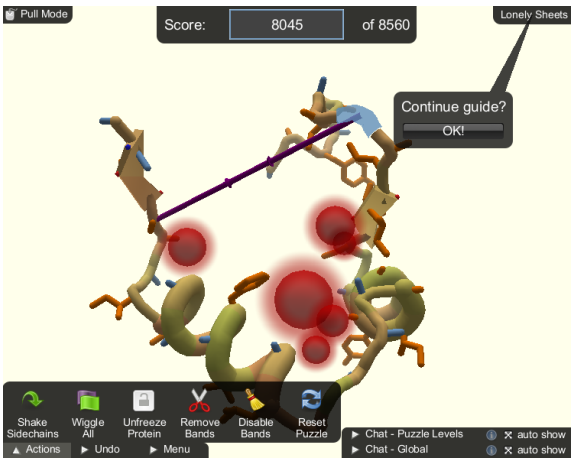


Figure 3. A puzzle in Foldit, a game in which players manipulate protein models and try to pack the shape as tightly as possible. The game is complex, has an interface with many tools and buttons, and offers a gameplay experience that is very different from other games. Therefore, players cannot rely on prior knowledge of other games very much when trying to learn how to play Foldit.

Foldit is a multiplayer, online game in which players compete and collaborate to fold protein structures efficiently [5]. Players can manipulate 3D protein structures, shown in Figure 3, by pulling on them, freezing pieces to prevent them from moving, and launching optimizations that will computationally improve the protein. Each folded protein is given a score which is used to rank solutions. Foldit has a set of offline tutorial puzzles designed to prepare players for online competitions, which we used for this study.

Foldit differs from Refraction and Hello Worlds in several key ways. First, players must download and install Foldit from a website in order to play, and must create an account to participate in online competitions. Second, the gameplay mechanics required to fold proteins are unique to this game, and as a result it is unlikely that players can benefit from past gaming experience when playing Foldit. Finally, it is much more complex than the two "casual" games and requires an understanding of some unconventional mechanics, such as placing rubber bands to hold the protein together, or running a wide variety of optimizations. Since Foldit players are working to solve complex, open scientific problems through gameplay, we do not consider this a "casual" game.

Metrics

Collecting data on player behavior through large-scale anonymous experiments has strengths and weaknesses. One key advantage is that, in contrast to laboratory user studies, our experiments are conducted "in the wild" where players do not know that they are being observed. As a result, participants are playing under their own motivation, and our findings reflect their natural behavior. However, one important limitation is that we have no interaction with our participants, and we cannot know what they are thinking or feeling. Therefore, we must infer the effects of tutorials on player engagement by measuring their behavior.

We measure player engagement in three ways. First we count the number of unique levels each player completes. Second, we calculate the total length of time that they played the game. Since players occasionally idle for long periods, we aggregated moves the player made in 30-second intervals, removing periods of two or more consecutive idle intervals from the total play time. Finally, we measured the return rate, defined as the number of times players loaded the game page in Refraction and Hello Worlds, and as the number of times the game was restarted in Foldit. While these metrics are not precisely the same as "engagement" or "learnability," we argue that they are closely related, given that players are free to leave at any time and cannot advance in a game without learning its associated rules and strategies. If the purpose of tutorials is to teach game mechanics to the player, the ability to complete levels should be the ultimate test of whether the player has learned those mechanics.

We modified each of the games to randomly assign new players to one of the eight experimental conditions. We focused solely on new players for this experiment; veteran players who were already familiar with the games were not included in our analysis. In Refraction and Hello Worlds, we tracked players using the approach described in [2], storing progress and experimental condition through the Flash cache. One drawback of this method is that if players changed computers or deleted their Flash cache, they were treated as new players. However, since the Flash cache is inconvenient to clear and this action deletes all progress in the game, we considered this risk to be small. For Foldit, we tracked players using their Foldit account and stored the condition number locally on their machine. Players who logged in on multiple machines were tracked by their account and discarded to prevent the inclusion of players who played multiple tutorial versions.

STUDY DESIGN

We performed a multivariate experiment with four variables: *presence*, *context-sensitivity*, *freedom*, and *availability of on-demand help*. We picked these variables to capture the most salient differences in tutorial implementations seen in successful games. We included a total of eight experimental conditions, shown in Table 2, each representing a combination of values for the four variables. Table 1 shows a breakdown of the concepts that each game introduces and the order in which they are introduced. The following sections explain how the four variables are implemented in each game.

Tutorial presence

We wanted to examine how the addition of tutorials affected player behavior. Therefore, the first variable in our experi-

Hello Worlds			Refraction			Foldit		
Concept	Level	Page	Concept	Level	Page	Concept	Level	Page
Character Movement (S)	1	1	Powering Ships	1	1	Clashes	1	1
Collecting Coins	1	2	Benders (S)	1	1	Pulling Sidechains	1	1
Using Doors (S)	1	1	Multiple Benders	3	-	Rotating the Camera	2	2
Red Doors	1	3	Ship Input Direction	3	4	Score	2	2
Overworld Doors	O	5	Asteroids	4	1	Shake (S)	3	3
Star Types	O	6	Coins	5	3	Pulling the Backbone (S)	4	4,5
Multiple Worlds	2	1	Splitters (S)	6	5	Undo	4	8
Combo View (S)	2	4	Fractional Ships (S)	6	4	Voids	5	6
Rewind (S)	3	4	Magnifying Glass (S)	7	6	Reset	5	8
Green Doors	6	3	Combiners (S)	33	7	Wiggle (S)	6	7
Blue Doors	10	3	Expanders (S)	37	8	Hydrogen Bonds	7	9
			Multiple Input Ships (S)	43	9	Wiggle Again (S)	7	7
			Multiple Denominators	44	9	Rubber Bands (S)	8	10
						Translating the Camera	9	2
						Rubber Bands Again (S)	9	10
						Freeze (S)	10	11
						Backbone Color	11	12
						Rebuild (S)	11	12
						Hydrophobics	12	13
						Exposeds	12	-
						Tweak Rotate (S)	13	14
						Tweak Shift (S)	14	15
						Tweak Rotate Again (S)	15	14
						Secondary Structure Mode (S)	16	16

Table 1. A breakdown of concepts taught in each game. The concepts are listed by the order in which they were taught in the context-sensitive condition. Concepts were either taught in a specific level as part of a context-sensitive tutorial or in the manual used by the context-insensitive tutorial and the on-demand help. Concepts marked with an "(S)" were stenciled during the blocking conditions. Foldit is more complex than the other two games and required many more tutorials.

Condition	Presence	Context	Freedom	Help
Insensitive+Help	Present	Insensitive	Nonblocking	On-demand
Insensitive	Present	Insensitive	Nonblocking	None
Sensitive+Help	Present	Sensitive	Nonblocking	On-demand
Sensitive	Present	Sensitive	Nonblocking	None
Blocking+Help	Present	Sensitive	Blocking	On-demand
Blocking	Present	Sensitive	Blocking	None
Help Only	None	N/A	N/A	On-demand
No Tutorials	None	N/A	N/A	None

Table 2. The eight tutorial conditions that we implemented for our multivariate experiment, based on four independent variables: presence, context-sensitivity, freedom, and availability of help. These conditions were designed to represent a wide range of existing tutorial designs and evaluate multiple hypotheses simultaneously.

ment was *presence*, resulting in two conditions: *tutorials* and *no tutorials*. For conditions in which there were *no tutorials*, the game explained nothing on its own and players were forced to either experiment or seek help outside the game. The conditions with *tutorials* are further broken into more categories in the following sections.

Context-sensitivity

To evaluate the effect of tutorial context-sensitivity on engagement and player retention, we introduced an additional variable, *context-sensitivity*. For conditions in which tutorials were present, we organized the tutorials from each game into two versions: *context-sensitive* and *context-insensitive*.

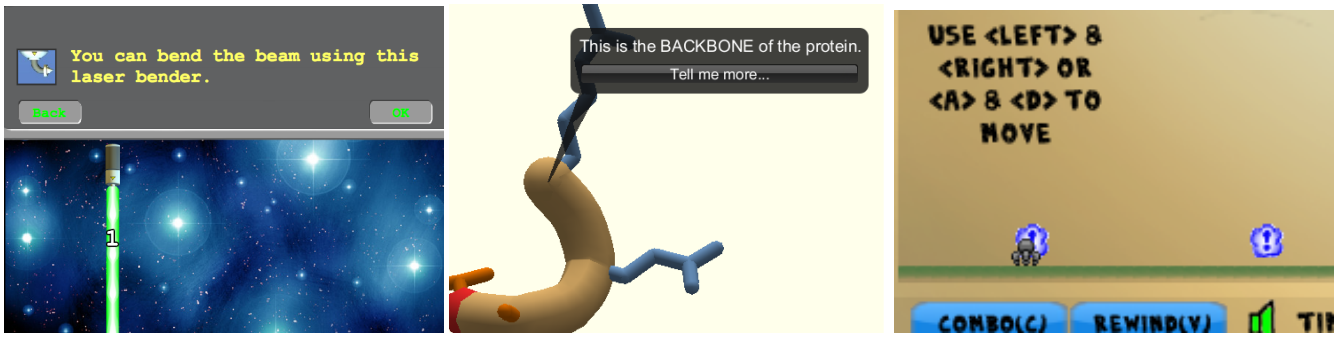
For the *context-sensitive* version, we tried to introduce each concept as closely as possible to when the player needed to use it. Therefore, for each of the concepts in Table 1, we added a short message explaining that concept to the first level in the game requiring knowledge of it. Some tutorials also contained contextual images to reinforce the text. Each message was controlled for size so that it could be easily ignored. Additionally, the player could dismiss any tutorial message.

In Refraction, as shown in Figure 4(a), the instructions appeared in a notification bar at the top of the main grid. Refraction also contained "just-in-time" messages that would activate when the player tried to do something that was not allowed. For example, when a player powered a target spaceship with less than the required amount, a message saying "too little power" appeared over the spaceship. Each of these messages appeared every time the spaceship was underpowered, but player could click "don't show again," which would cause that particular message to stop appearing.

Foldit tutorial messages appeared in pop-up boxes as shown in Figure 4(b). In Refraction and Foldit, longer context-sensitive messages were sometimes displayed across multiple pages, which the user could advance by clicking on the "next" or "Tell me more..." button.

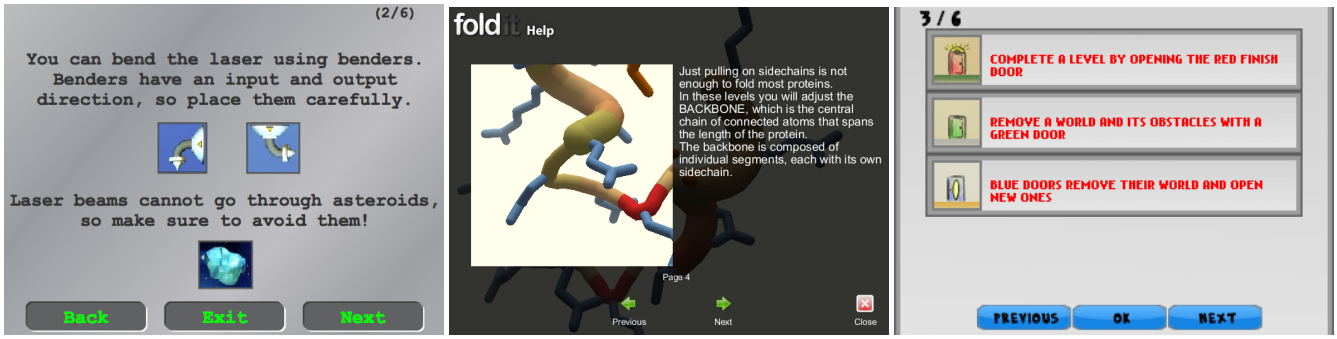
In Hello Worlds, shown in Figure 4(c), help icons were scattered throughout the game to give the player information when appropriate. These help icons were activated whenever the player stood over them, as shown in 4(c). When activated, the text relating to that help icon would appear in the space above the icon, allowing the user to easily read or ignore it. If the player's avatar then moved off of the help icon, the text would immediately disappear. If the player moved their avatar back over the help icon, the text would reappear.

For the *context-insensitive* version, we grouped the concepts of each game into multi-page manuals by topic and order of introduction. Since each game requires a different number of concepts, the length of the context-insensitive manuals varied across the games: 9 pages for Refraction, 16 for Foldit, and 6 for Hello Worlds. The manuals for Refraction, Foldit, and Hello Worlds are shown in Figures 5(a), 5(b), and 5(c), respectively. The manual consisted of a brief textual description of each of the game's concepts accompanied by



(a) (b) (c)

Figure 4. Screenshots of the context-sensitive tutorials. For Refraction, shown in Figure 4(a), a window appeared at the top of the screen containing text, pictures, an OK button to continue, and a back button to return to a previous message. Foldit's tutorial, shown in Figure 4(b), included a small box containing tutorial text, which related to the protein at hand, and a "Tell me more..." button to go onto the next message. For Hello Worlds, shown in Figure 4(c), blue help icons could be found throughout the game. Tutorial text appeared above an icon whenever the player's avatar overlapped with it.



(a) (b) (c)

Figure 5. Screenshots of the context-insensitive tutorials. All three games featured a large window containing text, pictures, page numbers, "Next" and "Previous" buttons, and a "Close" button to help the player navigate and understand the concepts of the game. The layout of this window varied from game to game. Figure 5(a) shows the layout for Refraction, Figure 5(b) shows the layout for Foldit, and Figure 5(c) shows the layout for Hello Worlds.

relevant pictures associated with that concept. These manuals appeared before the player could interact with any levels, forcing them to interact with the manual in some way before playing the game. The player could navigate between pages with the "next" and "back" buttons and gauge their progress using page numbers present on each page. The player could also close the manual at any time by clicking the "close" button.

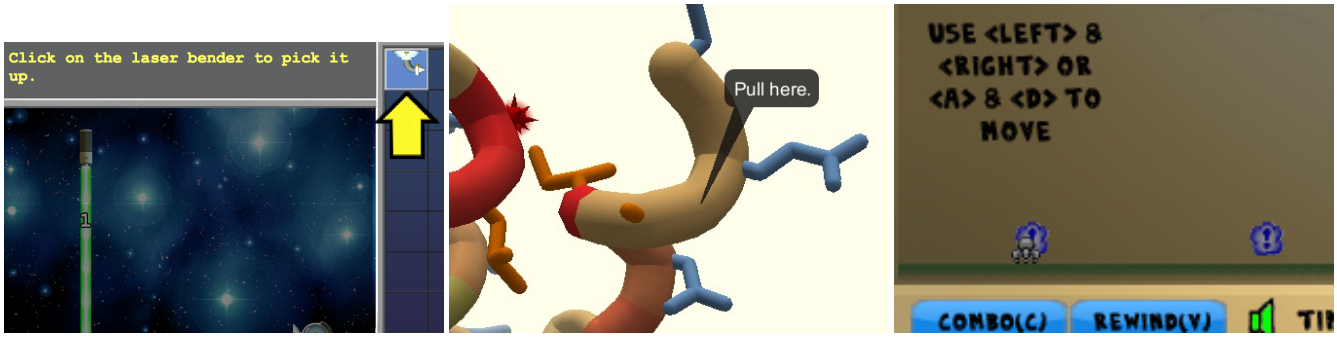
Due to their length, we broke the Refraction and Foldit tutorials into logical "chapters." As the player advanced, new chapters became available. At each new stage, the *context-insensitive* tutorial reappeared showing the first page in the new chapter, and the player could navigate to previous chapters if desired. Refraction had three chapters, displayed at the beginnings of worlds one, five and six respectively. Foldit had four chapters, displayed every four levels.

Freedom

To evaluate the effect of tutorial freedom, we divided the context-sensitive condition described above into two conditions: *blocking* and *nonblocking*. To create the *blocking* condition, we identified several concepts from each game that involved using a new tool, a new interface object, or a new keyboard key. We then added a "stencil" for each of these concepts that forced the player to use the new tool, blocking forward progress in the game until the player performed the desired action. After completing the stencil, freedom to interact with the full user interface was restored. We implemented stencils slightly differently in each game due to their different interaction styles.

In Refraction, when the player encountered a stencil, the game drew the player's attention to the target interface object by making the entire window gray except for that object, displaying the instructions on a notification bar at the top of the screen, adding an animated yellow arrow that pointed to the object, and ignoring all user input directed at anything other than the object. This implementation is shown in Figure 6(a). If a level had both context-sensitive tutorial messages and a stencil, then the game forced the player to click somewhere on the screen to advance each tutorial message before forcing the player to complete the stencil instructions.

In Foldit, the game displayed a message telling the player what to do, as shown in Figure 6(b). The screen was not grayed out, and the user could still manipulate the camera and interact with game's user interface controls. However, the game prevented the player from manipulating the protein in any other way until they performed the desired action. In



(a) (b) (c)

Figure 6. Screenshots of the blocking tutorials. A blocking tutorial prevented the player from continuing until they completed a specified action. For Refraction, shown in Figure 6(a), the tutorial grayed out the interface except for a window containing tutorial text, an object for the player to interact with, and an arrow pointing at that object. Foldit, shown in Figure 6(b), used a small text box to tell the player what action they needed to perform. Hello Worlds, shown in Figure 6(c), grayed out the screen except for character, the tutorial text, and the interface buttons whenever the blocking tutorial was active.

some levels the player needed to advance through preceding context-sensitive tutorial messages to reach the stencil.

In Hello Worlds, as in Refraction, a gray window was drawn over the entire game except for the player's avatar, the message text, and the user interface buttons. Since the user primarily interacts with Hello Worlds through the keyboard, the displayed message associated with each stencil told the user what key(s) to press to continue, as shown in Figure 6(c). All other keys were ignored until the user pressed one of the valid keys. The user was allowed to interact with the interface buttons, such as "quit," but could not make any progress on the level until completing the instructions in the stenciled message.

Availability of help

To evaluate whether players would look for help when they needed it, and to measure the effect of tutorial availability on player engagement, we added another variable, *availability of help*, with two possibilities: *on-demand help*, and *no help*. We created a version of the game with on-demand help for each of the previously described conditions.

There are many ways to give tutorial information on-demand. For this experiment, we added a help button to the main toolbar of each game. Since Foldit's main toolbar can be closed, another floating help button was added near the score display. Clicking on this help button opened the same manual used in the context-insensitive tutorial condition described previously. The manual included only the concepts which the player could have encountered up to that point. If later levels introduced new concepts, the help manual did not include those concepts until the user reached that level. In order to make the help provided as context-sensitive as possible, the manual automatically opened to the page that most closely related to the level the user is currently playing. If desired, the player could view all of the pages of the help screen with the "next" and "back" buttons. The player could close the help screen and return to the current level by clicking on a "close" button. When activated, the help screen covered the main interaction area for each game and restricted gameplay until the player closed the tutorial.

DATA ANALYSIS AND RESULTS

We collected data from Refraction and Foldit for approximately two weeks, accumulating 13,158 Refraction players and 9,743 Foldit players. During this period, Kongregate added a new badge to Refraction, pushing the game to the top of the "new badges list" and attracting many new players. Kongregate also featured Hello Worlds on the front page of its website as a "badge of the day," allowing us to collect data from 22,417 Hello Worlds players over a two-day period.

Table 3 shows the results of our experiment. Our measurements of levels completed and time played were not normally distributed, so we used on a non-parametric test, the Wilcoxon/Kruskal-Wallis 2-sample test, to analyze levels completed and time played for large-scale effects. The Z-value reported for this test is a standard normal random variable, a scaled version of the distribution of relative rankings of the lower-ranked condition. For return rate, we used Pearson χ^2 analyses to compare the percentages. The variance in our measures was very high, so a great deal of data was necessary to show statistical significance at the $p = 0.05$ level.

Tutorials were only justified in the most complex game

To evaluate whether tutorials improved player engagement, we compared the version of the game with no tutorials to the versions with context-sensitive tutorials and context-insensitive tutorials. The "presence" section of Table 3 shows the results for these comparisons.

We expected to find that including tutorials, either in-context or out-of-context, would lead to higher engagement than providing no instruction at all. In Foldit, this was the case. Foldit players with context-sensitive tutorials played 75% more levels and 29% longer than those with no tutorials. Players with context-insensitive tutorials played 25% more levels and 12% longer than with no tutorials. We found no significant effects for return rate.

However, tutorials were not as effective in the other two games. We found no significant effects for either comparison in Refraction. In Hello Worlds, we found no significant

Experimental variable	Game	Condition	Player Count	Time Played		Levels Completed		Return Rate	
Presence	Foldit	Context sensitive	1242	***660s***	$p < 0.001$	***7***	$p < 0.001$	19.16%	$p = 0.408$
		No Tutorials	1210	***510s***	$Z = -5.070$	***4***	$Z = -10.982$	20.50%	$\chi^2 = 0.686$
		Context insensitive	1147	***570s***	$p = 0.010$	***5***	$p < 0.001$	19.62%	$p = 0.594$
		No Tutorials	1210	***510s***	$Z = -2.592$	***4***	$Z = 4.353$	20.50%	$\chi^2 = 0.284$
	Refraction	Context sensitive	1634	990s	$p = 0.437$	15	$p = 0.294$	28.46%	$p = 0.925$
		No Tutorials	1678	1050s	$Z = -0.778$	16	$Z = -1.049$	28.61%	$\chi^2 = 0.009$
		Context insensitive	1687	1020s	$p = 0.483$	15	$p = 0.272$	28.45%	$p = 0.922$
		No Tutorials	1678	1050s	$Z = 0.702$	16	$Z = 1.099$	28.61%	$\chi^2 = 0.010$
	Hello Worlds	Context sensitive	2817	750s	$p = 0.697$	10	$p = 0.537$	***17.96%***	$p < 0.001$
		No Tutorials	2815	720s	$Z = -0.389$	10	$Z = -0.618$	***21.60%***	$\chi^2 = 11.733$
		Context insensitive	2754	690s	$p = 0.572$	10	$p = 0.770$	20.59%	$p = 0.356$
		No Tutorials	2815	720s	$Z = -0.565$	10	$Z = 0.292$	21.60%	$\chi^2 = 0.854$
Context sensitivity	Foldit	Context sensitive	1242	***660s***	$p = 0.014$	***7***	$p < 0.001$	19.16%	$p = 0.779$
		Context insensitive	1147	***570s***	$Z = -2.470$	***5***	$Z = -7.727$	19.62%	$\chi^2 = 0.079$
	Refraction	Context sensitive	1634	990s	$p = 0.901$	15	$p = 0.994$	28.46%	$p = 0.998$
		Context insensitive	1687	1020s	$Z = -0.124$	15	$Z = 0.008$	28.45%	$\chi^2 = 0.000$
	Hello Worlds	Context sensitive	2817	750s	$p = 0.348$	10	$p = 0.736$	***17.96%***	$p = 0.013$
		Context insensitive	2754	690s	$Z = -0.938$	10	$Z = -0.337$	***20.59%***	$\chi^2 = 6.175$
Freedom	Foldit	Blocking	1210	630s	$p = 0.454$	***7***	$p = 0.007$	18.93%	$p = 0.881$
		Non-blocking	1242	660s	$Z = -0.749$	***7***	$Z = 2.697$	19.16%	$\chi^2 = 0.022$
	Refraction	Blocking	1634	930s	$p = 0.142$	14	$p = 0.139$	26.37%	$p = 0.187$
		Non-blocking	1551	990s	$Z = -1.467$	15	$Z = -1.477$	28.46%	$\chi^2 = 1.743$
	Hello Worlds	Blocking	2729	723s	$p = 0.740$	9	$p = 0.376$	18.94%	$p = 0.346$
		Non-blocking	2817	750s	$Z = -0.332$	10	$Z = -0.886$	17.96%	$\chi^2 = 0.889$
Availability of help	Foldit	Help (aggreg.)	4939	600s	$p = 0.536$	5	$p = 0.208$	18.34%	$p = 0.130$
		No help (aggreg.)	4809	600s	$Z = -0.619$	5	$Z = -1.258$	19.55%	$\chi^2 = 2.297$
		Help Only	1238	***570s***	$p = 0.036$	***4***	$p = 0.001$	18.98%	$p = 0.347$
		No Tutorials	1210	***510s***	$Z = -2.101$	***4***	$Z = -3.197$	20.50%	$\chi^2 = 0.885$
	Refraction	Help (aggreg.)	6608	960s	$p = 0.806$	15	$p = 0.515$	28.50%	$p = 0.528$
		No help (aggreg.)	6550	990s	$Z = 0.245$	15	$Z = 0.651$	28.00%	$\chi^2 = 0.399$
		Help Only	1678	***900s***	$p = 0.031$	***14***	$p = 0.013$	27.43%	$p = 0.451$
		No Tutorials	1655	***1050s***	$Z = -2.161$	***16***	$Z = -2.496$	28.61%	$\chi^2 = 0.569$
	Hello Worlds	Help (aggreg.)	11302	720s	$p = 0.626$	10	$p = 0.972$	***18.56%***	$p = 0.021$
		No help (aggreg.)	11115	720s	$Z = 0.487$	10	$Z = 0.035$	***19.78%***	$\chi^2 = 5.314$
		Help Only	2864	754s	$p = 0.434$	11	$p = 0.190$	***18.44%***	$p = 0.003$
		No Tutorials	2815	720s	$Z = -0.782$	10	$Z = -1.310$	***21.60%***	$\chi^2 = 8.875$

Table 3. Summary of data gathered during the experiment, organized around our four experimental variables of *presence*, *context-sensitivity*, *freedom*, and *availability of help on-demand*. Statistically significant results are shown in blue italics. Our results show that tutorials only had positive effects on player behavior in Foldit, that context-sensitive tutorials outperformed context-insensitive tutorials in Foldit, that restricting player freedom did not improve engagement in any game, and that providing access to help on-demand was beneficial in Foldit, ineffective in Hello Worlds, and actually harmful in Refraction.

effects for levels completed and time played, and we found that 3.5% *fewer* players returned with context-sensitive tutorials than with no tutorials.

In contrast to our expectations, our results show that tutorials are only useful for improving player engagement in some games. It may be the case that tutorial value depends on game complexity. Refraction and Hello Worlds are typical of their genres and have intuitive interfaces. Foldit, on the other hand, is unconventional, complex, and requires deep strategy and spatial insight. Another possible explanation is that Foldit players are more patient. Since the barrier to play a downloadable game like Foldit is higher than for online casual games like Refraction and Hello Worlds, players who are willing to overcome this barrier may be willing to devote more time reading the tutorials. Further work is necessary to know the exact reasons for this effect. Regardless, our results show that tutorials may only be worthwhile for complex games, such as Foldit, and may not be worth the investment of time and resources in games with mechanics that are more easily discovered through experimentation, such as Refraction and Hello Worlds.

Context-sensitivity can improve engagement

We next evaluated the importance of presenting tutorial information in context. We expected that presenting information as closely as possible to when the player needed it would be better than presenting information out of context. The results for the comparison of context-sensitive and context-insensitive tutorials are shown in the "context-sensitivity" section of Table 3.

Context-sensitivity improved player engagement in Foldit. Foldit players played 40% more levels and 16% longer with context-sensitive tutorials than context-insensitive tutorials. However, we did not find positive effects for context-sensitivity in the other two games. We found no significant effects for Refraction. In Hello Worlds, we found that the return rate was about 2% less with context-sensitive tutorials than with context-insensitive tutorials.

Therefore, the importance of context-sensitivity in a particular game depends on whether tutorials positively impact player engagement in that game. For complex games where tutorials are beneficial, such as Foldit, presenting the information in context seems beneficial. However, for games in

which tutorial presence does not have an impact, it does not matter whether or not the information is presented in context.

Tutorial freedom did not affect player behavior

To evaluate whether restricting player freedom would improve tutorial effectiveness, we compared the versions with context-sensitive blocking tutorials to those with context-sensitive nonblocking tutorials. The results of this comparison are shown in the "freedom" section of Table 3.

We expected that blocking tutorials would focus player attention and improve learning. In Foldit, we found a significant effect for levels completed, but the median number of levels completed remained constant at 7 levels. We found no significant effects for Refraction and Hello Worlds.

Therefore, we found no evidence supporting the practice of restricting player freedom in order to focus player attention on target interface objects. This result may further reinforce our conclusion that players learn how to use the interface primarily through experimentation, and that forcing the player to perform a series of actions may not be effective. It may also be the case that players do not like having their freedom restricted and that this cancels out any positive effects on learning. Further examination is necessary to understand whether restricting player freedom improves player engagement and learning in some cases.

On-demand help harmed and helped player retention

We next evaluated whether player engagement would be improved by providing additional help that players could access when needed. We expected that providing additional help in this way would be beneficial. The results for the following comparisons are shown in the "availability of help" section of Table 3.

To look for large-scale effects of providing on-demand help in general, we first aggregated the four conditions with a help button and the four conditions without a help button and compared them against each other. We found only one significant effect across all three games, which was a 1.2% increase in return rate for Hello Worlds.

We then assumed that the effect of providing help on-demand would be strongest when no other instructions were given. Therefore, we compared the version with a help button and no other tutorials with the version with no help button and no tutorials. In Foldit, providing a help button in this case increased engagement. Players played about 12% longer with a help button than without. We found a significant effect on the number of levels completed but this effect did not change the median. However, we found negative effects in the other two games. In Hello Worlds, players returned about 3% less frequently with on-demand help than without. In Refraction, we found that players with on-demand help completed 12% *fewer* levels than those without access to help, and played for 15% *less* time.

The negative impact of the help button, particularly in Refraction, is difficult to explain. Only 31% of players that had access to the help button in Refraction ever used it. One possible explanation is that the knowledge that there is help available discourages players from putting as much effort into the game. It is also possible that players will only click "help" if they are already frustrated, and will quit even sooner if they are unable to find the help they need. We can only speculate at this point, however, since multivariate testing does not tell us what players are thinking. Future studies are needed to reproduce and understand this effect. Nevertheless, it is clear that players can respond unpredictably to tutorial implementations, and that tutorials can harm player retention unexpectedly. This points to the importance of testing tutorial implementations to avoid unexpected negative effects.

CONCLUSION

Our examination of tutorials in three games of varying complexity showed that tutorial effectiveness depends on the complexity of the game. Tutorials had the greatest value in the most unconventional and complex game, Foldit, increasing play time by as much as 29%. However, tutorials had *surprisingly little* impact on player behavior in Refraction and Hello Worlds, which are less complex, more similar to other games, and easier to learn through experimentation. Our results suggest that tutorials may not be effective for such games, and that designers should consider the complexity and discoverability of game mechanics when deciding whether to invest resources in tutorials. It is unlikely that a single approach will work for tutorial design in all games.

Since players seem to learn more from exploring than from reading text, we believe that it is important to design early levels in a way that maximizes a player's ability to experiment and discover game mechanics. A key question that arises is how to facilitate this experimentation while ensuring that the player learns how to play and does not become frustrated. We found little evidence to suggest that restricting player freedom to focus attention on a particular interface object or game mechanic is beneficial. Although it may be tempting to provide help on-demand, we found that adding a help button was only effective in Foldit, and actually *reduced* player progress by 12% and play time by 15% in Refraction. Future work is needed to understand how to break down a complex game into smaller "chunks" that can be learned through exploration, how to detect when a player is confused or frustrated, and how to intervene, if necessary, in a way that causes learning without negatively impacting engagement.

One of the drawbacks of our methodology is that we do not know demographic information about our players. It is likely that each game attracted different kinds of players and that self-selection contributed to the differences in player behavior that we observed. However, we believe that it is most useful to tailor tutorials for the players who do self-select to play each game because they are the players who are likely to play the most.

Since we only tested tutorials in Refraction, Hello Worlds, and Foldit, we cannot know for sure how these results generalize to other games and genres. In particular, our games are not representative of commercial games, and as a result we can draw no conclusions about the effectiveness of tutorials

in games that players must purchase. Further research with a wider variety of games is required to determine whether our results will apply to commercial games and games of different genres.

Furthermore, future work is necessary to understand how tutorial *presentation* affects player behavior. There are many ways to design tutorials, and we only experimented with basic designs. Although video game tutorials frequently include audio, animations, and videos, we only examined tutorials containing pictures and text. Another question is whether designers should integrate tutorials into the main game progression or provide them in a separate "tutorial mode" that can be avoided entirely, as suggested by [1] and utilized in highly successful games such as Deus Ex (Ion Storm Inc. 2000). Future experiments with a greater variety of tutorial styles will improve our understanding of how to design effective tutorials.

Our results point to the importance of analytics and large-scale analysis of player behavior because these effects would be difficult to discover without large amounts of data. We believe that this methodology will continue to provide valuable insights that improve our understanding of good game design. Such experiments are important because player behavior is often counterintuitive; each of our four hypotheses turned out to be either incorrect or incomplete. Furthermore, we believe that the ability of games to attract large numbers of participants, allowing researchers to perform tests with many experimental variables, will make them an important mechanism for future HCI research.

ACKNOWLEDGMENTS

We would like to acknowledge the members of the Refraction, Hello Worlds, and Foldit teams for developing the games and making this work possible. We also thank Anthony Pecorella and August Brown of Kongregate for promoting our games and helping us gather data. This work was supported by the University of Washington Center for Game Science, DARPA grant FA8750-11-2-0102, the Bill and Melinda Gates Foundation, NSF grant IIS0811902, two NSF Graduate Fellowships, Adobe, and Intel.

REFERENCES

1. E. Adams. The designer's notebook: Eight ways to make a bad tutorial. *Gamasutra*, 2011.
2. E. Andersen, Y.-E. Liu, R. Snider, R. Szeto, S. Cooper, and Z. Popović. On the harmfulness of secondary game objectives. In *FDG '11: Proceedings of the Sixth International Conference on the Foundations of Digital Games*, New York, NY, USA, 2011. ACM.
3. E. Andersen, Y.-E. Liu, R. Snider, R. Szeto, and Z. Popović. Placing a value on aesthetics in online casual games. In *CHI '11: Proceedings of the SIGCHI conference on Human factors in computing systems*, New York, NY, USA, 2011. ACM.
4. L. Bergman, V. Castelli, T. Lau, and D. Oblinger. Docwizards: A system for authoring follow-me documentation wizards. In *UIST '05 Proceedings of the 18th annual ACM symposium on User interface software and technology*, New York, NY, USA, 2005. ACM.
5. S. Cooper, F. Khatib, A. Treuille, J. Barbero, J. Lee, M. Beenen, A. Leaver-Fay, D. Baker, Z. Popović, and F. Players. Predicting protein structures with a multiplayer online game. *Nature*, 466(7307):756–760, August 2010.
6. D. K. Farkas. The role of balloon help. *ACM SIGDOC Asterisk Journal of Computer Documentation*, 17, 1993.
7. J. P. Gee. Learning by design: Games as learning machines. *Interactive Educational Multimedia*, 8:15–23, 2004.
8. Google Website Optimizer. http://www.google.com/websiteoptimizer/.
9. F. Grabler, M. Agrawala, W. Li, M. Dontcheva, and T. Igarashi. Generating photo manipulation tutorials by demonstration. In *ACM SIGGRAPH 2009*, New York, NY, USA, 2009. ACM.
10. T. Grossman and G. Fitzmaurice. Toolclips: An investigation of contextual video assistance for functionality understanding. In *CHI '10 Proceedings of the 28th international conference on Human factors in computing systems*, New York, NY, USA, 2010. ACM.
11. T. Grossman, G. Fitzmaurice, and R. Attar. A survey of software learnability: Metrics, methodologies and guidelines. In *CHI '09: Proceedings of the 27th international conference on Human factors in computing systems*, New York, NY, USA, 2009. ACM.
12. S. M. Harrison. A comparison of still, animated, or nonillustrated on-line help with written or spoken instructions in a graphical user interface. In *CHI '95 Proceedings of the SIGCHI conference on Human factors in computing systems*, New York, NY, USA, 1995. ACM.
13. J. Huang and M. B. Twidale. Graphstract: Minimal graphical help for computers. In *UIST '07 Proceedings of the 20th annual ACM symposium on User interface software and technology*, New York, NY, USA, 2007. ACM.
14. C. Kelleher and R. Pausch. Stencils-based tutorials: Design and evaluation. In *CHI '05 Proceedings of the SIGCHI conference on Human factors in computing systems*, New York, NY, USA, 2005. ACM.
15. K. Knabe. Apple guide: A case study in user-aided design of online help. In *CHI '95 Conference companion on Human factors in computing systems*, New York, NY, USA, 1995. ACM.
16. R. Kohavi, R. M. Henne, and D. Sommerfield. Practical guide to controlled experiments on the web: listen to your customers not to the hippo. In *KDD '07: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 959–967, New York, NY, USA, 2007. ACM.
17. T. Lau, L. Bergman, V. Castelli, and D. Oblinger. Sheepdog: Learning procedures for technical support. In *IUI '04 Proceedings of the 9th international conference on Intelligent user interfaces*, New York, NY, USA, 2004. ACM.
18. J. Lave and E. Wenger. *Situated learning: Legitimate peripheral participation*. Cambridge University Press, Cambridge, England, 1991.
19. G. Linden. Early Amazon: Shopping cart recommendations, 2006. http://glinden.blogspot.com/2006/04/early-amazon-shopping-cart.html.
20. J. Nielsen. *Usability Engineering*. Morgan Kaufmann, San Francisco, CA, USA, 1993.
21. S. Palmiter and J. Elkerton. An evaluations of demonstrations for learning computer-based tasks. In *CHI '91 Proceedings of the SIGCHI conference on Human factors in computing systems: Reaching through technology*, New York, NY, USA, 1991. ACM.
22. S. G. Ray. Tutorials: Learning to play. *Gamasutra*, 2010.
23. B. Shneiderman. *Designing the User Interface: Strategies for Effective Human-Computer Interaction*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1986.
24. J. Xiao, R. Catrambone, and J. Stasko. Be quiet? evaluating proactive and reactive user interface assistants. Technical report, Georgia Institute of Technology, Atlanta, GA, USA, 2003.