More than just Software Surprises: Purposes, Processes, and Directions for Software Application Easter Eggs

MATTHEW LAKIER, School of Computer Science, University of Waterloo, Canada
DANIEL VOGEL, School of Computer Science, University of Waterloo, Canada

"Easter eggs" are features hidden inside software, and the practice of developers including them is a long-standing global phenomenon. They have seen some investigation in the context of games, but despite their prevalence in non-game software applications, their nature within this context is less clear. We perform a qualitative, investigative analysis of Easter eggs in non-game software application contexts, using primarily archival research including discussion forums, social media posts, and user-created online databases, along with select developer interviews. Our work uncovers the stories behind, motivations for creating, and intended perceptions of Easter eggs, which we present as categories of purposes with illustrative examples. This analysis also informs a categorization and discussion of processes that Easter eggs undergo concerning the social and emotional circumstances of their developers and users. Finally, we use our results to motivate future directions for applying Easter eggs in user interfaces.

CCS Concepts: • **Human-centered computing → HCI theory, concepts and models**.

Additional Key Words and Phrases: Easter eggs; playful user interfaces

ACM Reference Format:
Matthew Lakier and Daniel Vogel. 2022. More than just Software Surprises: Purposes, Processes, and Directions for Software Application Easter Eggs. *Proc. ACM Hum.-Comput. Interact.* 6, CSCW1, Article 102 (April 2022), 26 pages. https://doi.org/10.1145/3512949

1 INTRODUCTION

Linguists of emerging language for 1996 defined *Easter eggs* as an "[u]nexpected operation programmed into commercial software as a joke and activated by a secret command" [2]. In fact, they had been included in software for decades, and have become an international phenomenon. Video games frequently include them; for example, developer Warren Robinett famously added a hidden room to the Atari game Adventure as a signature for his work because Atari would not credit the game's authors [61]. Easter eggs have even been referred to as a new kind of gameplay [21]; for example, hidden cheat codes can be hunted for, potentially leading to increased replayability and sales of a game [21, 94]. Easter egg "hunters" have active social communities, and online databases of Easter eggs inside computer and other media have been created (e.g., "eeggs.com").

Easter eggs are also prevalent in the user interfaces of non-game software applications, which we will refer to simply as "software applications". Yet, Easter eggs in this context have received limited study, with the exception of a popular press book from 1997 [65] and a few online blog posts and news reports [33, 74]. Examples of software application Easter eggs include the puzzle

Authors' addresses: Matthew Lakier, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, matthew.lakier@uwaterloo.ca; Daniel Vogel, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, dvogel@uwaterloo.ca.

game hidden in the Android operating system, and the sheep sound hidden in Adobe After Effects. Although Easter eggs in software applications share commonalities with those in games, there are notable differences, such as cheat codes being unique to games, and considerations that are unique to software applications, such as Easter eggs distracting users from work.

We define software Easter eggs as hidden features in software applications that are activated either at certain times or through specific user interactions, and whose functionality is undocumented. This definition expands the mostly game-oriented definitions of Easter eggs used in previous work by refining and relaxing certain criteria, to better represent the range of hidden features in software applications. Specifically, Easter eggs do not have to be exclusive to commercial software, as open-source software also includes what developers have called Easter eggs. Easter eggs can be created by actors other than the original programmer; for example, users may inject hidden media into applications through remixing or customization, and then share this for others to discover. Not all Easter eggs must be strictly reproducible; "hidden" features can be designed so they may only be found at certain times (e.g., in association with a calendar event), and Easter egg hunting experiences can even be the result of fictitious Easter eggs that are not implemented at all. Finally, Easter eggs can provide value beyond entertainment, such as education and social awareness.

The goal of our research is to better understand these types of Easter egg phenomena in non-game software application user interfaces. This includes the stories behind their development, why developers make them and for whom, and how users have perceived and are intended to perceive them. We use the term "purpose" to encompass this set of ideas succinctly. In addition, through the process of our analysis, we identify processes related to Easter eggs, specifically those regarding their social and emotional benefits and challenges. The study of Easter eggs is important to developers, users, HCI researchers, and designers, for several reasons. First, the study of interfaces that are not only productivity-boosting, but also "playful" and provide emotional value, has seen recent interest in HCI [4]. Easter eggs are one potential way for HCI researchers and UX designers to create interfaces that provide emotional value. We can gain insights about this from studying how Easter eggs are designed to provide emotional value in the wild. Easter eggs may also have the potential to support the *hedonic quality* [39] of user interfaces, such as heightening user perceptions of originality and excitement, which can make software applications more appealing in commercial settings. Second, as we discuss in Section 6, Easter eggs can serve as design exemplars for important concepts such as microbreaks [41], emotional support using conversational agents [17], emergent help features, and remote collaboration icebreaker activities, which could be further researched as ways to improve task efficiency and workplace health and well-being with computer interfaces. Third, Easter eggs give rise to a number of social phenomena like digital communities of Easter egg hunters (e.g., "eeggs.com") and Easter egg remixers (e.g., Scratch users who have remixed the Google Chrome dinosaur game[1]). Understanding the purposes of Easter eggs and how they impact developers and users can form a starting point for future work to learn about these digital communities.

We developed a conceptual categorization of Easter egg purposes through an approach centred around archival research, using a selection of tools and principles from constructivist grounded theory [15]. The primary data source was Internet media discussing Easter eggs. To gain more insight into specific well-known Easter eggs, we conducted interviews with four developers. Through the process of constructing this categorization of purposes, we also identify and categorize a set of processes that Easter eggs undergo, concerning developers and users. We illustrate our conceptual categories using a variety of examples. We also propose design directions for applying ideas from Easter eggs to user interfaces in new ways, such as providing a structure for users to be more expressive and gain educational value from their interactions with software.

[1] https://scratch.mit.edu/projects/189057541/remixes/

2 BACKGROUND AND RELATED WORK

We briefly summarize past work on Easter eggs in games, before examining past definitions and classifications of Easter eggs in software applications. In the context of software applications, few formal publications exist.

2.1 Easter eggs in games

Easter eggs have been mainly explored in the context of video games, for which they have been conceptualized in different ways. Consalvo [21] argues that Easter eggs are a new kind of gameplay and may lead to increased sales for this reason. She breaks them down into three types: "ornamental", affecting the display of the game, "functional", adding behaviour such as game cheat codes, and Easter eggs that "attempt to make a larger statement", for example, by critiquing the status quo. Bailey [9] highlights how Easter eggs have been used to circumvent organizational restrictions, have accidentally emerged out of last-minute scrapped video game content, and have been hidden for marketing reasons. Consalvo [21] and Gazzard [32] both relate Easter eggs to the concept of cheating. An example of cheating in games is using hidden cheat codes, which give players advantages like bonus lives or weapons. Gazzard argues that exploring or exploiting hidden or unintended features of games can lead to flow-inducing player experiences, and social aspects around sharing game hacks and Easter eggs can lead to an increased sense of community among gamers.

Easter eggs can also be conceptualized through the lenses of intertextuality and paratextuality. Intertextuality can be described as a "textual strategy" involving *parodic allusion*, *creative inclusion* of other texts, or *self-reflexive reference* [70]. For example, searching "the answer to life the universe and everything" in Google Search produces "42" as a result, a parodic allusion to *The Hitchhiker's Guide to the Galaxy*. Broadening this idea from text to media more generally, some Easter eggs in games have been described intertextually, as allusions to other media [86]. Consalvo [21] argues that Easter eggs in games "set the stage for a paratextual industry" for texts such as game magazines and strategy guides. She relates this to what she terms "gaming capital": "how being a member of game culture is about more than playing games or even playing them well". In the following sections, we illustrate how Easter eggs in software applications could be understood similarly, for example, by encouraging discussion about Easter eggs on social media.

In the context of software applications, the nature of Easter eggs is necessarily different from Easter eggs in games. In particular, the idea of hidden cheat code Easter eggs in games does not have a direct analogue in software applications. Further, software application Easter eggs have received much more negative attention concerning security and developer productivity problems, because software applications are often used in mission-critical contexts. Because Easter eggs are hidden and undocumented code, they have similarities to dangerous code such as backdoors and logic bombs [67]. For example, historical versions of the PHP scripting language contained an image-based Easter egg that would render in output webpages. The images changed in each version, making it easier for hackers to identify the PHP version the server was using. This could facilitate exploitation of bugs and vulnerabilities in old versions [3]. For reasons like these, some companies like Apple and Microsoft have prohibited developers from including Easter eggs in their software [27, 69].

2.2 Definitions and characteristics in software applications

Nagy-Farkas' 1997 popular press book "Easter Eggs: Software Surprises" [65] is one of the most thorough, albeit informal, descriptions of game and software application Easter eggs. Nagy-Farkas

defines Easter eggs as features that are undocumented, written by the original programmer, reproducible, harmless, and entertaining [65]. Our expanded definition of Easter eggs, provided in the Introduction, refines and relaxes these criteria, including user-made Easter egg remixes and customizations, and non-reproducible and non-entertaining Easter eggs.

In software applications, some characteristics of Easter eggs have been discussed in specific contexts, such as with conversational agents, and in informal publications like blog posts. Work by Luger and Sellen [53] demonstrates that people will often look for Easter eggs when first interacting with conversational agents, to gauge what the agent is capable of in a playful way. Weinel et al. [90] argue that Easter eggs serve two "purposes": providing enjoyment to their creators, and adding aesthetic value to the resulting artifact (whether it be software, a video game, music, etc.). However, despite discussing software, Weinel et al.'s categorization is mainly focused on Easter eggs in musical media (e.g., images hidden in music spectrograms). Outside of academia, in a New York Times article [74], Pogue uses several examples to describe how software and hardware Easter eggs can be used for recognition, to "establish software as an art form", as a 'thank-you' notice, and to encourage prospective company employees. In a blog post, Gkogka [33] discusses how Easter eggs can lead to "delightful" user experiences that "act as special rewards", entertain users and "break the monotony", give "personality" to a product, "attract people back to [a] product", and "serve as a distraction". She also describes how Easter eggs can be "context focused", "enlightening", "artistic", and capable of "add[ing] extra functionality" or "teasing." Nagy-Farkas [65] claims software Easter eggs are created for four reasons: "recognition", "nostalgia", "humor", and "revenge". However, this classification is mostly unsubstantiated and may be less relevant today, because of new computing platforms, interface designs, and reduced concern about computer resource usage. Inspired by these classifications of Easter eggs, we develop a more complete and modern classification of Easter egg purposes.

The term "Easter egg" has also been used to describe software watermarks, historically used to discourage software intellectual property theft by hiding ownership information [20]. For example, if a certain sequence of keys is pressed, the developers of an application could be revealed as evidence of ownership. This is similar to Easter eggs where developers have discreetly signed their work, like the Atari Adventure game example.

2.3 Gamification and Play

Easter eggs can be considered in relation to research on playfulness in user interfaces and gamification. Costello and Edmonds' pleasure framework [22], in the context of interactive art, categorizes thirteen "pleasures of play". It formed the basis of the PLEX framework [52], which is used in the design of games and playful experiences more generally. Boberg et al. [12] place Easter eggs within the PLEX category of "Discovery", in which the discovery is "associated with a degree of surprise". In gamification research, Easter eggs are considered to be a gamification element, described by Marczewski as "a fun way to reward and surprise people for just having a look around" [54]. Our investigation reveals that software Easter eggs are not necessarily playful, and can have more purposes than this more limited view of enabling discoveries or acting as rewards or surprises.

3 METHOD

The primary motivation of this investigation is to better understand, in the context of software application user interfaces, the purposes of and processes involving Easter eggs.

We focused on Easter eggs that people experience in user interfaces in the wild, rather than investigating them purely from a standpoint of developer-centric practices. Because Easter eggs are by definition hidden phenomena, this approach poses the challenge of identifying the provenance of specific Easter eggs, which is not usually possible without inside knowledge. Further, as we learned

through our investigation, the original developers of specific Easter eggs may become forgotten over time. While there are online communities of Easter egg "hunters", active engagement with these communities can only provide limited insight into Easter egg developers' intention and affect, because developers are usually not involved in these communities and users can only speculate as to Easter egg purposes.

Recognizing these constraints around possible data sources, we chose to use an approach centred around archival research, in which we performed a qualitative, investigative analysis of Internet media discussing Easter eggs. Some previous works have characterized this approach as "passive netnography" [23], a set of research practices in which data is collected through the Internet [49], itself related to a subset of methods used in digital ethnography. We also conducted interviews with four developers to gain insight into specific well-known Easter eggs. We used a selection of tools and methodological principles from constructivist grounded theory as described by Charmaz [15], which focuses on constructing conceptual categories that emerge from the data, as opposed to borrowing or applying external or pre-existing theories to analyze the data. Specifically, we conducted data collection and analysis simultaneously and iteratively, made constant comparisons across different segments of data, and drew on the data to develop conceptual categories. We focused on developing a set of Easter egg purposes and conceptual categories that have strong explanatory power and a close fit with the data, rather than aiming to develop a formal theory. This is in line with Charmaz's suggestion that grounded theory guidelines can "solve varied problems... whether or not you aim for theory development". In contrast with other qualitative approaches that separate data collection and analysis, such as thematic analysis [13], we found the need to continually return to the data to seek more types of information. Our methodology enabled us to refine our research questions throughout the process of analyzing the data [15], uncovering potential benefits and challenges associated with Easter eggs, in addition to their purposes. We describe our approach in more detail below.

Rather than provide a history of Easter eggs, we primarily focus on recent examples because they are more relevant to contemporary computing platforms, applications, and interface designs. However, throughout the investigation, some historical Easter eggs were included to highlight specific properties of different Easter egg purposes.

The results of this investigation (Sections 4–5) form conceptual categorizations, and further enable us to provide insights for design practice inspired by Easter eggs through a discussion of future directions (Section 6).

Archival Research. To scope out the space of software application Easter eggs, we performed initial sampling using the popular, community-maintained Easter egg database website "eeggs.com". The website contains categorized lists of links to pages about Easter eggs. The categories correspond to the source of the Easter egg, such as "Application Software" and "Movie". Each Easter egg page contains user-contributed instructions and possibly images for the associated Easter egg.

Though there are several different database and collection websites for Easter eggs online, we found others are more outdated, contain only a few software application Easter eggs, or do not clearly separate software Easter eggs from a larger mix of other types, such as those in movies. "eeggs.com" is itself rather outdated; only 58 of the 1656 submitted software Easter eggs were added since 2010. However, a considerable number of older Easter eggs remain in current versions of the same software. We also observed the fact that companies that have produced Easter eggs in the past are known for continuing to produce new Easter eggs (e.g., Google [74]). To ensure recent Easter eggs were represented, we scraped "eeggs.com" to collect the most common keywords in Easter egg application names, identifying top organization and application names associated with recent software (e.g., "Adobe", "Windows", rather than "Macromedia", "Lotus").

We used these as keywords in Easter egg searches, combined with search terms "Easter egg" and "hidden/secret/undocumented feature", in online discussion forums and blog posts. We also went through social media feeds associated with these organization and application names, which helped us identify non-conventional Easter eggs like Google Doodles and a fictitious "=CAKE" function in Microsoft Excel. For Easter eggs belonging to open source software, we searched through commit messages and bug reports to collect additional information. Using theoretical sampling [15], we further investigated emerging categories and discovered variations.

Interviews. During the archival research investigation, we identified a small number of specifically well-known Easter eggs that had little information available about them. For a subset of these, we were able to identify contact information for potential developers with inside knowledge. We conducted email-based interviews with software developers who created Easter eggs themselves, or were in close contact with developers who did. Our interview protocol was approved by our institution's research ethics board, and requires we preserve the anonymity of our participants.

We used a structured interview approach, defined in the same sense as Dimond et al. [28]: a fixed set of questions were provided to the developers by email and responded to via email. We used this direct, email-based approach because we were 'cold emailing' developers associated with well-known Easter eggs, and wanted to make our contact as effortless for them as possible to encourage a response. The questions asked were:

1. "What was the motivation to create this Easter egg? Is there a story behind it?"
2. "Who was this Easter egg targeted at? (For example, a particular person or population.)"
3. "From what you've seen, how have others (developers, users, etc.) responded to this Easter egg? What comments have others made to you about it?"

We received responses from four developers. We describe their companies and software in a way that preserves their anonymity, and we refer to them using the generic labels, *D1–D4*. *D1* is a developer for a popular open-source multimedia application for Android, *D2* is a senior engineer at a large, commercial software company, *D3* is a front-end developer at a different, large, commercial software company, and *D4* is the lead developer of an internationally renowned open source software application. *D1*, *D3*, and *D4* discussed one Easter egg each, whereas *D2* discussed six separate Easter eggs. We followed up with two of the developers to clarify ambiguities in their initial replies and to collect more information about gaps in properties of categories uniquely associated with their Easter eggs.

Purpose categories were constructed by coding the data in an iterative process moving between data and analysis. Initial (open) coding was line-by-line. Focused coding and theoretical sampling were used to consolidate more abstract categories and identify variations in the data. We stopped collecting Easter egg examples and associated information once we reached saturation; that is, new data was not revealing new theoretical insights or properties of Easter egg purposes [15]. During the process of focused coding, we also identified emergent social and emotional process categories, which we developed as a secondary categorization simultaneously with the purpose categories. The data analyzed comprised the developer interview text, as well as the archival research documents, specifically passages written by developers, passages from users that described objective Easter egg functionality, and any textual information contained within the Easter eggs. This process resulted in 14 purpose categories and 6 process categories. Coded passages from the data sources and a detailed description of each purpose category are included as supplemental material.

4 PURPOSES OF EASTER EGGS

We discuss the 14 Easter egg purposes that emerged from our qualitative analysis, which encompass stories behind, motivations for creating, and intended perceptions of Easter eggs. Where relevant, we also discuss the informal publications about Easter eggs in the context of specific purposes, to highlight the different types of language that can be used to describe Easter eggs. This set of purposes may be incomplete; however, the purposes we identified serve as a way of understanding specific common and interesting aspects of the functions of Easter eggs. Also note that these purposes are not mutually exclusive, as many Easter eggs are applicable to more than one category. The purposes are presented roughly from least to most prevalent in our investigation; however, measuring frequency was not our goal, and in practice, is challenging due to the data collection considerations previously discussed. Table 1 summarizes all of the Easter eggs we analyzed and their purposes.

Providing functionality in the software application. This purpose relates to Easter eggs that Consalvo would describe as "functional", as opposed to "ornamental". However, this distinction is less clear for software applications than games; the meaning of 'providing functionality' is broader, because unlike games, software applications may not have meaningful metrics for progress. Any hidden, undocumented feature that users find useful could be considered an example.

A canonical example is how, since Microsoft Windows 7, creating a folder with a specific, undocumented filename, creates a shortcut to an "All Tasks" view, with all Windows Control Panel items in a single view. Users have colloquially called this functionality "God Mode". It was an implementation detail not intended for use, but people discovered this functionality and found it useful nonetheless [71].

Teaching users transferable knowledge and skills. Some Easter eggs create educational value for users by providing knowledge or the means to access knowledge, or showing the user how to do something that might be useful in other contexts.

For example, Google Doodles, the temporary commemorative modifications to the Google homepage logo, often educate about important historical figures. "We really want to, like, kind of tickle different parts of the users' brains; you know like one day it's about dance, one day it's about science; one day it's about pop culture; like another day it's a national holiday", said Ryan Germick, a Google Doodle designer [37]. Not only can Easter eggs themselves educate users, but they can also motivate users to seek more information, for example, by clicking on the Google Doodle image to search for more sources related to a topic. To achieve this, Google Doodles are designed to "inspire curiosity" [37]. Google Search itself also has similar educational Easter eggs, such as Conway's Game of Life, which Google engineer Peter Dolan describes as: "introduc[ing] people to the Game of Life. It's often seen when you're just starting to learn programming" [59]. These Easter eggs are associated with search engines, which are often used to gain knowledge about a topic, independent of any Easter eggs. In Section 6, we describe other ways that future Easter eggs could be designed to provide educational value.

Rewarding users for their actions. This purpose is similar to Gkogka's "act as special rewards" characterization. While Gkoka refers to Easter eggs that encourage user "exploration" and brand "loyalty" [33], we focus on Easter eggs that reward users for specific beneficial actions they take within the software application.

An example is the "treat" hidden inside the WordPress terms of service. The text "If you're actually reading this, here's a treat." is hidden midway through the terms. Clicking this opens a webpage with a photo of a meal at a restaurant. This "treat" has been unintentionally copied to other websites that based their terms of service off WordPress' [74]. *D2* provided another example,

Table 1. Summary of Easter egg purposes and associated publications. 41 Easter eggs were investigated in total. Counts sum to more than this total because Easter eggs can have multiple purposes.

Purpose	Easter Egg Examples	Num.	Associated References
Providing functionality in the software application	Windows "All Tasks" view [71]	1	"functional" Easter eggs (Consalvo [21])
Teaching users transferable knowledge and skills	Google Doodles [37], humorous Google Search results [59]	2	
Rewarding users for their actions	WordPress terms of service "treat" [74], *D2* Easter egg	2	"act as special rewards" (Gkogka [33])
Mocking or seeking revenge	"about:mozilla" page in Internet Explorer [19], *D2* Easter egg	2	"revenge" (Nagy-Farkas [65])
Showcasing developer skills or new technology	Excel 97 flight simulator [38], Matlab's "image" [31]	2	
Occupying users while waiting	Chrome dinosaur game [34, 42], uTorrent's Tetris [87]	2	"serve as a distraction" (Gkogka)
Sharing a personal message from the developer	Matlab's "reshape" [60], Amazon Risher tribute [10]	2	'thank-you' (Pogue [74])
Recruiting developers	recruitment via web browser developer tools, Google Foobar [72, 80]	2	encouraging prospective company employees (Pogue)
Giving credit to the individual developers	Excel 97 flight simulator, "about" pages/Firefox's "about:mozilla", After Effects credits [6], *D2* Easter egg	4	"recognition" (Nagy-Farkas)
Creating hype	Android version [36], After Effects credits, Matlab's "spy" [78], Matlab's "image", Excel's "=CAKE" function [58], *D3* Easter egg, *D4* Easter egg	7	
Encouraging development or remixing	Android version, GIMP's "Goat-exercise" [66, 88], *D1* Easter egg, *D2* Easter egg, *D3* Easter egg	5	
Sharing something dear to the developer	Android version, Matlab's "image", PHP version [68], Matlab's "spy", *D2* Easter egg	5	"break the monotony" (Gkogka)
Boosting moods with simple jokes or references	VLC Santa hat [45], VS Code Santa hat [26], Ant Design holiday theme [51], GIMP's "Goat-exercise", humorous Google Search results, *apt-get* "moo" [35], "about:mozilla" page in Internet Explorer, *man* Easter egg [83, 89], After Effects sheep sound [79], After Effects' "Mask Embiggen" [18], Chrome dinosaur game, AutoCAD's "CAD-PAC" (fictitious) [8], Microsoft Office Clippy background [57], PHP version, Matlab's "f***" [44], Matlab's "reshape", 4 *D2* Easter eggs	20	"humor" (Nagy-Farkas), "nostalgia" (Nagy-Farkas), "break the monotony" (Gkogka), "serve as a distraction" (Gkogka)
Acting as a creative, expressive outlet for developers	Google Doodles, humorous Google Search results, After Effects sheep sound, "about" pages/Firefox's "about:mozilla", Firefox's "about:robots" [76], *D2* Easter egg	6	"establish[ing] software as an art form" (Pogue), "personality" (Gkogka)

explaining that, historically, an image-based Easter egg in one of their software applications was "designed by prominent users / beta testers". In the case of beta testers, this meant that as a reward for using a specific version of the software application, they could showcase their design skills in an Easter egg.

Mocking or seeking revenge. This purpose is related to that of sharing a personal message, with the difference of having a specific focus of 'poking fun' at another person, group of people, organization, or piece of software. It could be understood as a superset of Nagy-Farkas' "revenge" category.

An example is an Easter egg in Internet Explorer that was a response to a similar Easter egg in Mozilla Firefox (which we also discuss later). When navigating to "about:mozilla" in old versions of Internet Explorer,[2] a blue screen is displayed. Users have suggested that this Easter egg was designed to look like the Blue Screen of Death, in reference to the instability of Netscape at the time; however, some have instead related it to Microsoft branding colours [19]. As another example, *D2* described an Easter egg in one of their applications: "Initially, [it was] just a humorous placeholder name for a new, as-yet-unnamed command in [the application], when shipped in a beta build, it annoyed a manager as unprofessional, and he ordered its removal. The programmer responded by putting in a more appropriate name for the new command, but sometimes randomly replacing other [UI controls] with [humorous text], rarely enough so that you'd wonder if you actually saw it or just imagined it – just to tweak the manager."

Showcasing developer skills or new technology. This purpose was particularly relevant historically, when developers would push the graphical capabilities of early computer systems to the limits. An example is the Microsoft Excel 97 flight simulator. In this Easter egg, selecting a specific range of cells in the spreadsheet, then entering a sequence of specific keyboard and mouse commands, would trigger a fullscreen flight simulator to be displayed. YouTube commenter "hank804" claims to have been an intern when this Easter egg was being developed, and adds: "Imagine how cool this was back then, when 3D engines were relatively new" [38]. The developers were able to show off their skill with 3D programming, as well as what contemporary computer hardware at the time was capable of.

Fig. 1. Images steganographically hidden in Matlab's 'default image'. © Image courtesy of The MathWorks, Inc.

Matlab, a numerical computing environment, contains another example of Easter eggs that showcase developer skills in the form of an image displayed when the *image* function is called with no parameters (Figure 1). The story behind this image is documented in a blog post by the developer, Steve Eddins [31]. The software company, MathWorks, had a charity auction; one of the items was the right to specify the default image for the next version of Matlab. Eddins was a newly-hired image processing specialist at the time, so he had a desire to do this. He chose to make the image steganographic (containing additional hidden data) in response to complaints that Matlab only supported double-precision values at the time. To showcase the capabilities of double-precision values and his skill as a self-proclaimed "image processing guy", he used all of the bits to encode multiple images, each of which were selected by different developers.

[2]Relocated to "res://mshtml.dll/about.moz" in recent versions.

Occupying users while waiting. Some Easter eggs are hidden in a way that they reveal themselves as a source of entertainment when the user would otherwise be waiting for a job to complete or a resource to become available in the application. Similar to Gkogka's "serve as a distraction" characterization, Easter eggs could be designed to draw users' attention away from limitations or error pages within an application. This notion of distraction was also raised by *D3*, who referred to an Easter egg that they designed as a "short distraction" for users "experienc[ing] a level of frustration".

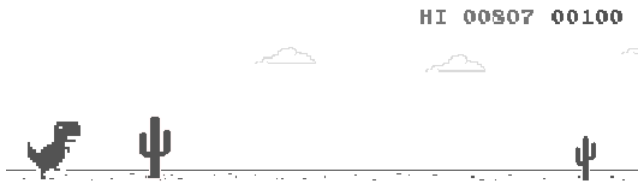

Fig. 2. Dinosaur game in Google Chrome, accessible by navigating to "about:dino". Image courtesy of Google LLC.

The dinosaur game in Google Chrome is a well-known example. When Chrome is unable to connect to the Internet, a webpage with a dinosaur is shown. Pressing the spacebar starts a minigame in which the user can make the dinosaur jump and duck to avoid obstacles. When Internet connectivity resumes, the page disappears and is replaced with the desired destination page. Another example is the game of Tetris hidden within uTorrent, a BitTorrent client. By pressing "t" on the keyboard in the application "About" window, users can play a bare-bones game of Tetris while waiting for downloads or uploads of files [87]. Interestingly, these examples share the trait that, despite neither being games themselves, the Easter eggs contained within are games.

Sharing a personal message from the developer. This purpose could be understood as a superset of Pogue's 'thank-you' category, which only refers to messages that express gratitude. We use "personal message" to refer to a written sentiment directed to a particular person or audience. One canonical example is the David Risher tribute Easter egg in Amazon, the online store. By clicking an invisible link at the bottom of the site directory page,[3] a page titled "Thank You, David Risher" is shown in tribute to Risher for his contributions to Amazon as its former senior VP for retail [10]. As another example, in older versions of Matlab, calling the "reshape" function with invalid arguments would result in the humorous and personal error message: "Cleve says you should be doing something more useful." The "Cleve" in this message is Cleve Moler, the developer of that functionality, which "temporarily fixed a bug in earlier versions where negative arguments caused a segmentation violation" [60].

Recruiting developers. This purpose relates to Pogue's category of encouraging prospective company employees. While Pogue describes an Easter egg that uses persuasive language to encourage users to apply [74], we further identify recruitment messages that act more like passive banner advertisements, or assess the user's fitness for the job through the process of interacting with the Easter egg. A common example is how recruitment messages are hidden in websites as a message printed to the developer tools console. Companies that include this type of Easter egg include Reddit, NYTimes, Etsy, and Baidu. These are sometimes decorated with ASCII art (e.g., Reddit, NYTimes), and sometimes with a small entertaining message (e.g., Baidu, in Chinese). Another example is the Google Search recruitment Easter egg, "Google Foobar" [72, 80]. It reveals itself as a secret coding

[3] https://www.amazon.com/gp/site-directory

challenge, which Google can use to both attract the attention of possible new employees and gauge their abilities. This Easter egg is far less reproducible than many other Easter eggs, and sometimes can be activated by using programming-related search keywords or is sometimes hidden inside HTML comments [72].

Giving credit to the individual developers. This purpose is similar to Nagy-Farkas' "recognition" category of personal message Easter eggs, but also considers Easter eggs that give credit regardless of the developer's "ability as a programmer" [65]. *D2* describes that Easter eggs can be used to "sign one's work" and that "[o]ther software has sometimes included the creators' names as easter eggs". While less extreme than the Atari Adventure game Easter egg, the flight simulator hidden in Microsoft Excel 97 is an example of this purpose. When the flight simulator is activated, there are scrolling credits on the terrain listing the developers of the application. Notably, these developers are not credited in the official application "About" window. Another example with an interesting story is "about:" pages (e.g., "about:blank"). These were initially added as Easter eggs in Netscape (now Mozilla Firefox) to give credit to the individual developers by showing off their names, and eventually their personal webpages [62, 97]. A modern example is the hidden credits project in Adobe After Effects, activated by holding down specific modifier keys while opening the application "About" window. This opens up a project containing a creative animated video crediting the individuals responsible for that particular version of After Effects [6].

Creating hype. Easter eggs can generate hype by attracting social media followers or encouraging users to anticipate or hunt for more. Since version 2.3 "Gingerbread", the Android operating system has contained a new Easter egg in each version, activated by repeatedly tapping the version number text in the Settings app. The Easter eggs are often interactive minigames, like a nonogram puzzle. Dan Sandler, an Android developer, described the intended user reaction as: "I wonder what they did in the next version?" [36]. A recent example is the fictitious "=CAKE" function in Excel, advertised by the Microsoft Excel team on Twitter [58]. Calling the function would purportedly turn the background of the spreadsheet into a colourful cake image (or perhaps even a real cake, as interpreted by some followers). The post was prompted by a viral trend at the time of creating cakes that look like "everyday objects", such as shoes or bags of chips [14].

Many other software companies have used social media to drive hype around Easter eggs in their software products. This may inspire brand "loyalty" [33]. Adobe has used social media to actively give hints about Easter eggs in their software, such as posts from the After Effects team [1, 93]. The Matlab team has posted teaser images to Twitter of what some of Matlab's Easter eggs look like, without giving away how to access them or what they mean [55]. Independent from the companies themselves, reports on Easter eggs by tech journalists also lead to online discussions, in which people reminisce about Easter eggs in software they have used, clarify certain aspects of Easter egg history, and discuss possible safety concerns around including Easter eggs in vehicles like cars and airplanes [27, 74]. The use of social media to present Easter eggs also relates to Consalvo's argument that Easter eggs "set the stage for a paratextual industry" [21], but in this case, the paratext is social media posts rather than magazines and strategy guides about games.

A contrasting way that Easter eggs generate hype is by satiating people's desire for discovery, similar to how "Discovery" is described as a key category of games and playful experiences [48]. Rather than promote Easter eggs publicly like Adobe or Matlab, some developers prefer to avoid exposing Easter eggs, instead relying on the thrill experienced by users when they discover one. *D3* explained that users who do not know about an Easter egg in an application "will hopefully discover it on their own and get that sense of surprise and excitement when you discover a hidden feature". *D4* takes this mindset a step further, suggesting that "It's generally known that easter eggs

are not to be exposed widely", so that they "[remain] a secret". They added: "Just like you would not tell where the egg is hidden, the search is part of the fun."

Encouraging development or remixing. This purpose refers to Easter eggs that encourage two related phenomena, development and 'remixing', by demonstrating functionality of an application or its code, or encouraging users to look at the code for themselves. What ties these together is how users apply their creativity to become part of an Easter egg themselves.

One common context in which Easter eggs have encouraged development is open source software. Easter eggs are included to teach other developers how to implement certain kinds of functionality. An example of this is the "Goat-exercise" feature in GIMP [66], a graphics processing application. The feature is a small demo for developers, showing how to make a plugin for the application using a graphics library called GEGL (Generic Graphics Library). The name is a joke, and comes from a nickname that developers gave GEGL: "Genetically Engineered Goat, Large" [88].

Another example is a hidden icon in *D1*'s Android multimedia application, which animates when tapped. *D1* told us: "At the time we started [the application], no one from the team had any previous experience with the Android OS. We were all learning on the go and we played a lot with the internal features. After a few months of development, before reaching the store for the first time... I accidentally discovered a very simple way to add animations to a visual object and started to play with it as a distraction." This approach for adding animations was something that this developer realized would be useful, so they used this Easter egg as a demonstration for their "fellow [application] teammates". They "had no intention of making an easter egg that would survive the first release of [the application] but it was kept along the years." This motivation of using Easter eggs to learn or teach about development is not unique to open source software. The "engineer wanting an excuse to learn how to [implement code using a new API]" was also mentioned by *D2* as motivation, confirming a similar source of Easter eggs in commercial software.

Because many Easter eggs are artistic in nature, it follows that remixing Easter eggs would be of interest, in line with trends in visual art and music of remixing, sampling, mash-ups, and parodies. An example is the series of Android 'version' Easter eggs, for which developers explicitly encourage remixing: Sandler emphasized that the Android Easter eggs are open source, and that users "can even, like, copy the code and make your own version" [36]. In reference to an Easter egg in a popular application developed by their company, *D3* told us that "The fact that there are many homages and remixes to the [Easter egg] has always astounded me." This indicates that there are active and creative communities of Easter egg remixers.

Sharing something dear to the developer. Developers use some Easter eggs as a sort of 'time capsule', by hiding media with personal value to them, often pictures. This purpose relates to Gkogka's "break the monotony" characterization, by giving users something "new and unexpected" to look at [33].

An example is the original Android 'version' Easter egg in Android 2.3 Gingerbread. In an interview [36], Sandler explained that Dianne Hackborn, the head of the Android framework team, was friends with an artist, whom she asked to make a painting containing the Android robot mascot: "[W]hen she got the painting, she thought it was so awesome, that she wanted to hide it in the platform, so that other people could find it". Through this Easter egg, Hackborn was able to share an artwork with personal meaning to her.

The 'default image' Easter egg in Matlab is another example (Figure 1). The images hidden inside the default image include pictures of personal importance, such as those of developers, their family members and their pets, and a developer's favourite number. The images also include some mathematically or historically interesting images and photographs. Matlab has other Easter eggs with this same purpose, such as the *spy* function. When called with no arguments, it shows an

image of a dog [78], which a MathWorks staff member confirmed to be a lead developer's pet dog [5]. PHP used to contain a similar Easter egg, whereby secret query strings would result in PHP-based websites displaying funny or personal images chosen by the developers, such as photos of the developers' pets. In each version, a new image was used [68].

Boosting moods with simple jokes or references. This purpose relates to Nagy-Farkas' "humor" category and Gkogka's "break the monotony" characterization, in that an unexpected Easter egg can break up an "ordinary task" with entertainment [33]. It also loosely relates to Gkogka's "serve as a distraction" characterization; for example, 404 "page not found" error pages frequently contain humorous content, such as cartoons (e.g., [7]) or emotional characters (e.g., [29]), which may help reduce user dissatisfaction. We combine the ideas of jokes and references because these two aspects are often found in conjunction.

Sometimes, such Easter eggs may be "motivated first by the need for temporary UI" (*D2*). For example, *D2* explains that in two separate Easter eggs, one of their software applications used humorous icons because the developers "wanted a temporary icon until the real one got designed". The humorous stand-in icons were later included as Easter eggs. Cleve's humorous "reshape" error message in Matlab is another example. It was initially added to fix a bug, but later replaced by a more serious and accurate error message.

More generally, such Easter eggs are often motivated by a developer's "impulse to do something fun and self-expressive when under stress" (*D2*). Sometimes Easter eggs are targeted at users, whereas other times, they are inside jokes among developers (the latter being a common purpose of Easter eggs in historical desktop software).

Targeted at the developers of the application, the After Effects sheep sound effect is an example of this purpose. Shift-clicking on the title pane in After Effects causes it to play a "baa" sound.[4] At a motion graphics meetup, Dan Wilk, an original developer of Adobe After Effects, passionately described the story behind the Easter egg: "So the sheep sound is... [a developer's] mother making a sheep sound. So I think what it was, it was Monaco or one of the OS9 or OS7 fonts – it actually doesn't look right anymore – one of the fonts that was in the Mac OS system had this odd glyph in one of the upper ASCII [things], [and it was this] sheep, and so as some sort of whimsical moment, [the developer] put sheep in the effects control window, and of course once that happened it got a life of its own" [79]. In other words, the Easter egg was a humorous reference to a sheep glyph in a font at the time (which itself may have been an Easter egg). Users who discover the Easter egg may find it humorous without context, but the story behind it is what makes it connect emotionally with the developers who know about it.

Some examples of Easter eggs focused on making specific references to other media are the "Mask Embiggen" Easter egg in Adobe After effects, which replaces the UI text "Expansion" with "Embiggen", in reference to a particular episode of The Simpsons [18]; and the "moo" command line option for package management application *apt-get*, which causes an ASCII-art cow to be displayed in the console, in reference to an IRC user who would use the greeting "Moo" [35]. In reference to intertextuality, these Easter eggs could be considered examples of *creative inclusion*, a form of commentary on the fragments of media they reproduce.

There have also been more controversial Easter eggs. For example, Santa-themed decorations have been included in various software applications during the winter holiday season (e.g., VLC Media Player [45], VS Code [26], Ant Design [51]), which some users have taken issue with because of the association with Christmas. As another example, old versions of Matlab had the "f***" Easter egg function (the actual swear word), which when entered, Matlab would print the unsavoury reply "Your place or mine?" to the console [44]. The developers likely felt that this was a humorous

[4]This sound is also used by the application to indicate render errors.

response [46]. In Section 5, we discuss how Santa-themed decorations have caused conflict between developers and users, and in Section 6, we suggest how Easter eggs like "f***" in Matlab could inspire the design of Easter eggs that help users emotionally.

There are also many joke and reference-focused Easter eggs targeted at the users of software applications. Until a few years ago, *man*, a documentation reading application for Unix-like systems, displayed "gimme gimme gimme" (in reference to a popular song by 1970s pop group Abba) if *man* was run "after midnight".[5] This reference was suggested on Twitter in 2011 by a user, Marnanel Thurman [83]. The developer, Colin Watson, saw this tweet and actually put the Easter egg in. After removing it in 2017, Watson said, "I'm glad that it made some people smile, which after all was the whole purpose of it." [89] This type of reaction was mirrored by *D2*, who for one of their software applications, said, "[r]eactions I've gotten from [the application] fans have mostly been amusement or delight at discovering an easter egg or recognizing its meaning." Sometimes users have also created blog or social media posts about fabricated Easter eggs, such as for April Fools' Day jokes. For example, a fictitious "CAD-PAC" Easter egg in AutoCAD was described in a blog post [8], which would purportedly start a game of *Pac-Man* in a pop-up window after entering a sequence of fourteen commands. Prank Easter eggs may entertain their creators when observing others' reactions.

Several examples of joke and reference-focused Easter eggs also contain an element of Nagy-Farkas' "nostalgia" category. For example, when describing the design of the dinosaur game in Google Chrome, the developers wanted to make the game "reminiscent of vintage video games" [34]. Another example is the "Clippy" Easter egg in the background of recent versions of Microsoft Word. If the "Office Background" theme is set to "School Supplies", then a rendering of Clippy, the infamous Microsoft Office user interface agent from historical versions, appears as part of the background [57]. This Easter egg has no dynamic behaviour – it is just a static part of the background image. It is a reminder of how times and software have changed since Clippy was included as a user interface agent in Microsoft Office.

For completeness, previously mentioned Easter eggs that also exemplify the purpose of boosting moods with jokes or references include: the WordPress "treat" Easter egg, the "Goat-exercise" feature in GIMP, the PHP version Easter egg (which specifically used humorous imagery), the Google Chrome dinosaur game (which was a joke that in the "prehistoric age", there was no Wi-Fi [34]), humorous Google Search results in response to "the answer to life the universe and everything" (also to queries such as "recursion", "anagram", "do a barrel roll", and "The Wizard of Oz" [59]), and the "about:mozilla" page in Internet Explorer (which referenced Netscape).

Acting as a creative, expressive outlet for developers. *D2* told us that "Easter eggs are motivated partly by the impulse to do something fun and self-expressive". Several informal Easter egg categorizations [33, 74] have similarly described Easter eggs as having qualities such as "establish[ing] software as an art form", giving "personality" to a product, or other functions associated with creativity and expression. Wilk described the birth of the Adobe After Effects sheep sound Easter egg as a "whimsical moment" [79]. Despite these Easter egg characterizations, our investigation found that Easter eggs are infrequently described as serving this purpose. However, we note that most of the identified Easter egg purposes are not centred around helping developers or users fulfill utilitarian productivity goals. Rather, we suggest that the *expressive* power of Easter eggs has important emotional and social value for their developers and users, defined as "supporting [someone] to build stronger social bonds; feel free, explorative and creative; or simply experience joy and affect" [4]. Easter eggs that we coded as having other purposes may and likely do share this purpose.

[5]The Abba song lyrics are: "Gimme, gimme, gimme a man after midnight".

As a specific example, "about:mozilla" was one of the "about:" pages described above that has survived to this day, providing cryptic quotes out of a fictitious "Book of Mozilla". These have been updated over time to reflect the history of Firefox in a creative way [64]. Other similar Easter eggs have been in undocumented browser "about:" pages, such as "about:robots" in Firefox, which was created through a bug report in which developers contributed creative ideas for icons and text to use [76]. As another example, while Google Doodles is time based and not considered a conventional Easter egg, it has taken this purpose of Easter eggs to the extreme, frequently presenting animations and minigames about science, culture, and holidays. Germick describes this process as: "how do you share a human moment from behind the screen?" [37]. The humorous Google Search results are similarly motivated, as Dolan describes: "Some engineer gets some idea and can't stop thinking about it, and they make it happen, and everyone says, 'Oh my god, that's amazing!' And then it goes out" [59].

In summary, there is a broad spectrum of Easter egg purposes, and Easter eggs are not merely "the result of... an individual engineer thinking something essentially random would be funny" (*D2*), but have diverse, focused purposes. Developers have different stances on who Easter eggs are for, some saying "Easter eggs aren't generally targeted at anyone in particular" (*D2*), while others saying "[Easter eggs] are for everyone" (*D3*). Our investigation reveals that it can be both.

5 PROCESSES IMPACTING DEVELOPERS AND USERS

To demonstrate potential beneficial processes that Easter eggs engender, and challenges associated with their development and use, we discuss six processes Easter eggs undergo that relate to the social and emotional circumstances of their developers and users. Rather than forming a comprehensive theory, this section provides a descriptive account of some important processes that Easter eggs engender, which emerged during focused coding of Easter egg purposes. We believe these processes can provide insight to developers, users, researchers, and designers about the context around Easter eggs, and inspire further work about Easter eggs.

5.1 Potential Benefits of Easter Eggs

We highlight positive processes that Easter eggs engender, such as coming to be expected, becoming part of a tradition, and encouraging playful exchanges online.

5.1.1 Coming to be standardized or expected. In contrast with Easter eggs that are removed after some time, sometimes Easter egg behaviour remains long-term, either because it has evolved into core functionality, or users come to expect it of the software application.

One example is the "about:" pages initially included as an Easter egg in Netscape [62]. Because of how browser standards evolved, "about:" pages became *standardized* [63] and implemented across competing software applications (browsers). While still also used for Easter egg purposes (e.g., "about:mozilla"), they implement core functionality in browsers, such as configuration pages and the "blank" page.

Easter eggs can also maintain longevity by coming to be expected by users. In other words, some Easter eggs 'upgrade' from being an Easter egg to being a core application feature. The Google Chrome Dino game is an example of this. Initially, it was only available without an internet connection, or at "chrome://network-error/-106", an internal page. However, the Easter egg later became a "feature" of Chrome, when the developer made the functionality more accessible by hosting it at "about:dino" [42] (which, incidentally, is an "about:" page).

5.1.2 Getting "a life of its own". Borrowing the phrase that Dan Wilk used to describe the After Effects sheep sound, an Easter egg can "[get] a life of its own" [79], continuing to exist beyond the control of the original developer, sometimes going as far as becoming part of a tradition.

Recurring Easter eggs can lead to traditions with social value to developers and users. For example, the developer's anticipated user reaction of "I wonder what they did in the next version?" [36] to the Android Easter egg suggests that Android users expect the Easter egg to be continued in future releases of the operating system. The tradition of having these Easter eggs has become a part of the Android brand image, and has led to online communities discussing and sharing these Easter eggs (e.g., [77]). This idea is also applicable to application developers. *D2* told us that "Some of the [product] easter eggs are part of a tradition. [A specific Easter egg] has occasionally reappeared in various places... ever since" an early version of the software. *D1* said that "New developers working on [the application] are warned that any work... should retain this particular easter egg, it became part of the identity of the platform and it was replicated on at least one other platform". These Easter eggs have contributed to the identity of applications, even if only in the minds of the developers.

When Easter eggs are removed, they can gain nostalgic value. We note that this is different from our conceptualization of Easter eggs that contain embedded nostalgic media. An example is the Microsoft Excel 97 flight simulator. Online discussion often revolves around users reminiscing about how impressive the 3D technology supporting this Easter egg was, and telling stories about using it in school: "[G]ood times. I remember once in computers class in primary school I played this sim and the teacher yelled at me"; "Lol I remember playing this in high school on our computers in class. The teacher thought i installed software and started to write me up to send me to the principal" [38].

As alluded to earlier, even when Easter eggs are remembered, their original developers are often forgotten. Despite the fact that *D1*'s Easter egg became "part of the identity of the platform", they noted that "Besides a few developers that were present back in the days, I think most, if not all, of the current maintainers don't even know I was the original author." *D2* shared a similar sentiment: "The easter eggs tend to linger in memory longer than their creators unless there's some controversy surrounding it... The ratio of the number of people who know there used to be [two specific Easter eggs], relative to the number of people who know who wrote those easter eggs..., has to be many thousands to 1. People don't mostly write easter eggs for the fame or reputation."

5.1.3 Creating playful exchanges even when fictitious. Easter eggs do not have to be real or implemented in an application to stimulate online playful discussions about them. An example of this is the Microsoft Excel "=CAKE" function, which only ever appeared as a fabricated screenshot in the Microsoft Excel team's Twitter feed, yet resulted in a large amount of playful discussion including sarcastic exclamations, jokes about using the Easter egg, and jokes about using the software more generally. One user implemented and shared a real function with the name "=CAKE" using Visual Basic, with functionality inspired by the original tweet [95]. From a user perspective, often as pranks or April Fools' Day jokes, people have concocted elaborate instructions for Easter eggs that do not actually exist (e.g., the CAD-PAC Easter egg described earlier), which have fooled users in discussion forums [47].

5.2 Challenges with Easter Eggs

Our Easter egg purpose categories highlight that Easter eggs are mostly made in good faith. However, because Easter eggs are undocumented and often untested, we highlight how Easter eggs, even those made with good intentions, have raised challenges affecting their development and use, or caused tensions between developers and users.

5.2.1 Becoming a maintenance burden. The need for quality assurance testing has been a motivating factor for companies to exclude Easter eggs from software applications. In response to a blog post about Easter eggs in Matlab, Moler recounted the fate of the "reshape" Easter egg: "As our code base has increased, including such goodies in the MATLAB core has become problematic because of the strain it puts on rigorous automated testing. It is still possible to include them in a few 'leaf' M functions, like 'spy', that other functions do not depend upon" [60]. This comment suggests that testing Easter eggs that are part of core features puts an additional burden on the company. Despite the fact that only a few Easter eggs remain in Matlab, its users still enjoy them and discuss them online. Some of the other Easter eggs we described are also effective at minimizing testing burden. In reference to the application that *D2* develops, they identified that "Some teams at [the company] permit them as long as they're not time-based or random (because that represents too much of a testing / quality risk)". In the case of Microsoft Word's Clippy background Easter egg, the Easter egg is built into the image resources of the application, and would not need separate functional testing. There is also the option to construct an entirely fictitious Easter egg, like the "=CAKE" function promoted by the Microsoft Excel team.

After a certain period of time, many people who were going to discover any given Easter egg will have already discovered it, and Easter eggs centred around jokes or references become less novel over time. The *man* Easter egg gained a level of notoriety six years after it was created, when a developer posted a question on the *Unix & Linux* StackExchange noting that the Easter egg caused some of their automated tests to fail [89]. In response, Colin Watson, the developer, removed the Easter egg, later explaining that "It was getting a bit stale, though, and humour does require novelty", highlighting that "six years seems like a pretty good run for that sort of thing" and "it probably isn't going to get significantly better exposure than it already unexpectedly has" [89]. Removing "stale" Easter eggs eliminates the requirement to maintain this code and the risk of unexpectedly breaking user workflows.

5.2.2 Becoming a distraction. Both the acts of creating Easter eggs and using Easter eggs have been described as a "distraction". This process contrasts with Gkogka's categorization of Easter eggs that "serve as a distraction" [33], which refers to developers including them as a way to distract users from "negative or unpleasant" situations caused by the application, like 404 error pages. Instead, our identified process focuses on how Easter eggs can distract developers or users from their work.

In the case of developers, *D1* described that they created their Easter egg when "play[ing] with [animations] as a distraction". Others have described creating Easter eggs as a break from ordinary coding work. *D4* described that they create Easter eggs because of "Sometimes just being bored with useful work and wanting to add somet[h]ing useless that might make people smile". Netscape developer Jamie Zawinski blogged about adding his personal "about" page Easter egg [96], explaining that programs should be "fun to write". He adds: "If dropping in an easter egg allows a hacker to blow off some steam and consequently stick around the office for a few hours longer, and put in a 20 hour day instead of merely a 16 hour day, then those are resources well spent." *D2* suggested a similar sentiment, indicating that Easter eggs are often created "after too many late hours coding".

From a user perspective, Easter eggs are usually considered secondary to the functionality of the software application that contains them. However, Easter eggs can be 'too' entertaining, to the extent that they risk distracting worker or student audiences that need to use the application for important purposes. An example of this was the dinosaur game in Google Chrome. The game is hugely popular, and was becoming such a problematic distraction in schools and businesses that Google had to add a way for system administrators to disable it [34].

5.2.3 Causing controversy and polarization. We have emphasized the value of the expression afforded by Easter eggs. However, if software expresses values that are too controversial or polarizing, this can result in social and economic harm. A value or message carried by an Easter egg may be independent from, or even in conflict with, the purpose of the software; the meaning of an Easter egg may be taken differently by users of different backgrounds.

There have been instances of time-based Easter eggs inserted by developers that have offended users. An example of this is the Santa hat in VLC Media Player. Some users complained that the hat is a religious symbol, the developers are being selfish by including it, and it should be removed from the application. The reply from the developers stated that there was an option to disable this feature in the preferences, and that it is open source software – users can make their own fork, or use a different application altogether [45]. A very similar situation arose more recently, with a Santa hat icon added in VS Code, an open source text editor application developed by Microsoft. A user opened a GitHub issue stating that they were offended by the Santa hat and wanted it removed. The developers promptly removed the icon, despite a backlash from other users referring to the original poster as a "troll" [26, 75]. In both cases, whether the Easter egg was left in or removed, it led to divisiveness within the community. However, in the case of VS Code, this also prompted developers to add a feature for customizing the icon in question, enabling users to add celebratory icons of their choosing.

A more extreme example, involving the popular web UI framework Ant Design, was blogged about by software developer Shun Liang [51]. The developers of the framework had decided to stealthily insert a Christmas Easter egg that, for all websites using the framework, changed the page titles to "Ho ho ho" and rendered snow on top of the UI buttons. Unsurprisingly, this became problematic, leading to several very heated GitHub issues, partially fuelled by local Chinese government crackdowns on Christmas celebrations. One user thought that the codebase was hacked [85], two users claimed to have a pay reduction [50], and one user claimed to have been fired [11]. The feature was removed shortly thereafter, with an apology from the developer [73].

A noteworthy property of the Ant Design example is that the Easter egg would *propagate*; in other words, it would activate in unintended situations, in other software dependent on the software containing the Easter egg. This property also applies to the PHP version Easter egg, which contained humorous images of animals and people making funny faces, and propagated to any website using PHP. Independent of security considerations, if the Easter egg were somehow accidentally exposed in a business-critical context, this could cast doubt on the professionalism of the website owners. Time-based activation can exacerbate the risk of Easter eggs that propagate, because they can be triggered without any change to the workflows dependent on the software containing the Easter egg. *D2* told us: "Using [less common] signals or especially program state and date makes [Easter eggs] more time-consuming to develop and riskier to deploy. Date-related easter eggs, for instance, were specifically forbidden at [the company] after [an incident involving an Easter egg]. The same considerations would apply to having other triggers, like camera data, network traffic, etc. trigger an easter egg." This risk is also evident with the *man* Easter egg, which, as mentioned above, caused a developer's automated testing to fail only at certain times.

These examples converge to an important idea, captured by Larry Osterman in his blog post about why Microsoft products do not have Easter eggs [69]: "It's about trust. It's about being professional. Yeah, it's cool seeing your name up there in lights. It's cool when developers get a chance to let loose and show their creativity. But it's not cool when doing it costs us the trust of our customers." Osterman claims that Easter eggs are "irresponsible" because everything on one's computer should be "planned". If there is one hidden feature, how can users and businesses *trust* that there will not be more? What is "professional" and "responsible" will vary depending on the

developers, the users, and their relationship. For example, Adobe products have Easter eggs, yet this software is widely used by professional multimedia creators. Whether developers make Easter eggs purely for their own satisfaction or target them at users, there is a complex social, political, trust- and value-driven dynamic.

6 FUTURE DESIGN DIRECTIONS FOR SOFTWARE APPLICATION EASTER EGGS

Based on gaps and opportunities we identified through constructing the Easter egg purposes and processes above, and inspired by work in the contexts of playful interaction and CSCW, we propose a series of directions for how ideas associated with Easter eggs could be applied in software application interfaces in new ways. Before looking at specific directions, we emphasize two broad perspectives from which we approach the subject.

First, many of the previously identified Easter egg purposes, such as sharing a personal message from the developer, mocking or seeking revenge, and boosting moods with simple jokes or references, could be associated with expression, or what Pogue calls "software as an art form". For the same reasons that drawing, making music, and building crafts bring emotional value to people's lives, the personal expression enabled through the act of creating and sharing Easter eggs can too. In a blog post, Netscape developer Zawinski emphasized that "dropping in an easter egg allows [for]... a better program because the people who wrote it *cared* about it", also describing creating Easter eggs as "creative" and as "art" [96]. Enabling "self-expressive[ness]" (*D2*) through creative outlets in the software development or usage process is a desirable goal. From a developer standpoint, computers impose interesting hardware and software constraints, which can boost creativity through "out-of-the-box" thinking [37]. From a user standpoint, thinking about Easter eggs is one way we can work towards the goal of enabling applications to provide emotional value [4], also serving as inspiration for other user interface approaches.

Second, independent of social value fostered by the expressiveness of Easter eggs, their diversity and hidden nature provide a medium for building social communities. Our Easter egg examples demonstrate that there are many communities formed around Easter eggs, such as Easter egg "hunters", who search for and share Easter eggs; developer communities, for which Easter eggs enable tradition; and "maker" communities, who share ideas with developers and make Easter egg-like media of their own. Throughout our discussion below, we address some of these communities in more detail.

User-made Easter eggs. The rise of social media and propagation of popular culture via the Internet has led to a new range of possibilities for interactions between developers and users of software applications. In the past, users of software have been involved in Easter egg development by discussing ideas with developers over social media. Examples of this include the *man* Easter egg, which the developer implemented in response to a user's tweet; the "=CAKE" Excel Easter egg, which enabled a playful communication channel between the software team and the users; and the "moo" Easter egg in *apt-get*, which inspired joking behaviour with users, requesting different visuals for the displayed cow [82]. In other words, Easter eggs can form a topic of discussion that fosters dialogue providing emotional value to, and mutual understanding between developers and users through playful behaviour, and guiding more tailored interfaces based on user feedback.

Following this idea further, our expanded definition of Easter eggs considers that they may be created by users themselves. If we consider user customizations to software applications as a form of Easter egg, then we can make a connection to other related ideas such as "modding" and remixing. Cheliotis and Yew [16] outline that remix culture enables personal expression, supports social relationships, and helps develop communities. In the context of Easter eggs, software customizations that are shared can themselves contain Easter eggs, allowing for an exchange of Easter eggs between

application users. This is comparable to the ability for players to share their own levels in the game Super Mario Maker, or for users to share their own keyboard command scripts made with applications like AutoHotkey. The ability to share, remix, and discuss extensions to applications fosters an additional social level and forms a community around an application's users. Users may also make customizations as a form of *appropriation* [30], through which people "adopt and adapt technologies, fitting them into their working practices". For example, work by Hecht et al. [40] shows that the "Location" field in Twitter user profiles has been used for purposes other than sharing location. Because the location field in many online social platforms can be customized, it enables personal expression in a way that is 'hidden' from normal view. Slack, the communication tool, lets users provide a "status", which can be instead utilized to hide Easter egg-like messages in the tooltips of their names in the sidebar. Creating software applications with appropriation in mind could open opportunities for users to create their own forms of Easter eggs.

Easter eggs to encourage taking breaks. We identified that some Easter eggs entertain users when they are waiting, like the Chrome dinosaur game, or try to cheer users up when they are frustrated, like some 404 error pages. These purposes of Easter eggs can also be understood as ways to enable users to take a mental break from the task at hand. Because interactions with Easter eggs are usually brief, Easter eggs may be able to inspire the design of microbreaks built into software, which help reduce fatigue when using computer interfaces [41] and may or may not be entertaining. Easter eggs could be integrated directly within the process of completing a task, for example, by showing a short entertaining movie clip when clicking on an unexpected UI element. Alternatively, they could be integrated within dedicated microbreak sessions as a form of hide-and-seek game mechanic, similar to how Dai et al. [24] include other forms of game-like microbreaks during a crowdwork task to improve crowdworker retention.

Collaborative and collective Easter eggs. Our qualitative analysis revealed that Easter eggs are mostly an individual phenomenon, in that individual developers are usually responsible for crafting Easter eggs, and Easter eggs are usually activated and experienced by an individual user. For example, *D2* described the creation of Easter eggs as "generally the result of... an individual engineer". Given that social play activities can strengthen social bonds [4], we see an opportunity for Easter eggs to enter into this space, and be designed to be activated and experienced by multiple users in collaborative contexts. Wikström et al. developed the "SynchroMouse" game [92], in which players must move their mouse cursors in unison. Taking inspiration from a game like this, a remote collaborative application could unlock an entertaining Easter egg when remote participants synchronize their actions to simultaneously perform a secret gesture. Hunting for Easter eggs could serve as form of ice breaker activity built into remote collaboration software, or accidental discovery of Easter eggs activated collaboratively could serve to improve social bonds between remote collaborators.

Educational value from Easter eggs. We identified that some Easter eggs are designed to educate users. We propose some additional ways Easter eggs could inspire the design of educational user interfaces.

For example, Easter eggs can improve people's engagement with software by indirectly teaching them something about its implementation or usage, perhaps by improving the discoverability of less-well-known features. The well-known webcomic *Mr. Lovenstein* released a comic in which an application was not responding, and the user responded by repeatedly clicking the mouse. The program resumed, displaying "317 clicks has corrected the error" [91]. While this is a facetious example, Easter eggs could be designed with this idea in mind. For example, if a user does not understand how to use an application, and repeatedly clicks the mouse button, the application

could trigger an Easter egg that brings up a tutorial. In other words, Easter eggs can serve as alternative pathways to achieve application functionality, so long as the preferred way to achieve that functionality is made clear.

As another example, Easter eggs may be able to improve comprehension of website or software terms of service license agreements. In the case of the WordPress "treat" hidden in their terms of service, the Easter egg is so atypical, that rather than convincing people to read the terms of service, it acts only as a reward for those who do. However, if terms of service had more frequent Easter eggs, it may convince users to read them, and improve comprehension. This is similar to how adding visual elements, such as pull-quotes and icons, to license agreements can improve comprehension [43].

Combining the ideas of educational Easter eggs and Easter eggs that reward users for their actions, the above two examples suggest that Easter eggs could also gamify software applications in an educational way. However, some users may respond to this type of design more positively than others [84].

Another take on educational Easter eggs is to consider how they can provide eudaimonic value, in other words, longer-term value associated with well-being and "striving towards one's personal best" [56]. One way this could be achieved is by helping users work through their emotions. Work by Chin et al. [17] demonstrates that if a user verbally abuses (e.g., insults) a conversational agent (which is likely not a documented or intended use case), then the agent's choice of response can influence the person's anger and guilt levels. The "f***" (swear word) function in old versions of Matlab, discussed earlier, is an example of how an Easter egg has been designed to respond to verbal abuse-like input. Though this particular example produced an unsavoury response, Klein et al. [46] argued that this Matlab Easter egg could be considered a form of emotional support, because it provides feedback similar to active listening. They also describe an anecdote of how the Easter egg turned a user's "utter frustration" into "surprise and delight". Following this idea, the principle of trying to cheer up users when they are frustrated with a user interface may be a promising direction for future Easter eggs. Undocumented behaviour like this could be designed by therapists, to help calm users of software who are prone to anger outbursts, or for software used in high-stress situations.

From an open source software development perspective, having new contributors create Easter eggs could help them familiarize with a new codebase, gain interest in contributing to a project, and address some of the social barriers that Steinmacher et al. [81] identify, like finding a task to start with or lack of domain expertise or technical background. This is similar to how *D1* created an Easter egg to familiarize with developing for a new platform. Such an approach could also contribute to establishing traditions around Easter egg development as discussed earlier. In line with the above ideas about user-made Easter eggs and remixing, a more specific possibility would be to have newcomers remix an existing Easter egg in the software, considering that Dasgupta et al. [25] show how remixing supports learning and is positively associated with computational thinking.

7 CONCLUSION

To better understand the stories behind, motivations for creating, and intended perceptions of software application Easter eggs, we performed a qualitative analysis of non-game software application Easter eggs, including archival research and select developer interviews. We identified 14 different purposes that Easter eggs serve. Some were shared across many Easter eggs, such as *boosting moods with simple jokes or references* and *acting as a creative, expressive outlet for developers*. However, we also identified Easter eggs with other important goals such as providing functionality, recruiting developers, and teaching something to users. Through the process of constructing the

purposes, we categorized a set of processes associated with Easter eggs, highlighting ways that Easter eggs can impact developers and users, for example, by gaining nostalgic value or becoming a maintenance burden. We also presented future design directions for applying ideas from Easter eggs in new ways, emphasizing how users could be involved in Easter egg creation, or how Easter eggs could form microbreaks or sources of educational value for users.

Future work could investigate Easter eggs from the perspectives of other audiences. For example, it would be interesting to understand the goals, methods, and impressions of online Easter egg "hunter" communities. As another example, interviews with Easter egg developers could provide insight into how developers feel that practices around Easter eggs impact their development workflow and workplace culture, independent of end user experience considerations. At a more fundamental level, controlled studies and interviews with users could be conducted to understand how users encounter and respond to Easter eggs. For example, a study could be conducted comparing software with and without Easter eggs to measure effects on emotion and productivity. It would also be interesting to compare how well developers' intentions match up with users' perceptions in practice. Systems or toolkits for integrating Easter eggs into popular user interface frameworks could also be designed and evaluated. This may encourage the creation of more Easter Eggs, while also standardizing their development to make quality assurance testing easier and maintaining control over security and ethical policies.

Through this work, we have illustrated how Easter eggs are more than just "software surprises", and we hope that this work inspires developers, users, HCI researchers, and designers to consider the value that Easter eggs can bring to user interfaces.

ACKNOWLEDGMENTS

This work was made possible by NSERC Discovery Grant 2018-05187. We would like to extend a special thank you to the developers we interviewed, who took time out of their busy schedules to share valuable information with us. We would also like to thank the individuals and companies who helped us obtain copyright permissions to use screenshots of their Easter eggs in this paper.

REFERENCES

[1] Adobe After Effects. 2013. Adobe After Effects on Twitter. Retrieved January 11, 2021 from https://twitter.com/adobeae/status/293741987776040960

[2] John Algeo and Adele Algeo. 1996. Among the New Words. In *American Speech*, Vol. 71. Duke University Press, 184–197. Issue 2.

[3] Fredrik Nordberg Almroth. 2012. Do you dare to show your PHP easter egg? Retrieved January 11, 2021 from https://labs.detectify.com/2012/10/29/do-you-dare-to-show-your-php-easter-egg/

[4] Ferran Altarriba Bertran, Elena Márquez Segura, and Katherine Isbister. 2020. Technology for Situated and Emergent Play: A Bridging Concept and Design Agenda. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI '20)*. Association for Computing Machinery, New York, NY, USA, 1–14. https://doi.org/10.1145/3313831.3376859

[5] Yair Altman. 2012. Spy Easter egg take 2. Retrieved June 11, 2021 from https://undocumentedmatlab.com/articles/spy-easter-egg-take-2#comment-80767

[6] Dain Anderson. 1999. Adobe After Effects Easter Egg - See the Creators. Retrieved January 11, 2021 from https://eeggs.com/items/915.html

[7] Andrews McMeel Syndication. 2021. 404 Error – Dilbert by Scott Adams. Retrieved June 11, 2021 from https://dilbert.com/404

[8] "Antony". 2017. Incredible AutoCAD 2018 Easter Egg. Retrieved January 11, 2021 from https://hedproject.wordpress.com/2017/04/01/incredible-autocad-2018-easter-egg/

[9] Wm. Ruffin Bailey. 2008. Hacks, Mods, Easter Eggs, and Fossils: Intentionality and Digitalism in the Video Game. In *Playing the past : history and nostalgia in video games*. Vanderbilt University Press, 69–90.

[10] Jeff Bezos. 2002. Thank You, David Risher. Retrieved January 11, 2021 from https://www.amazon.com/gp/feature.html?ie=UTF8&docId=447307

[11] Davide Bianchi. 2018. Christmas easter egg. Retrieved January 11, 2021 from https://github.com/ant-design/ant-design/issues/13098

[12] Marion Boberg, Evangelos Karapanos, Jussi Holopainen, and Andrés Lucero. 2015. PLEXQ: Towards a Playful Experiences Questionnaire. In *Proceedings of the 2015 Annual Symposium on Computer-Human Interaction in Play* (London, United Kingdom) *(CHI PLAY '15)*. Association for Computing Machinery, New York, NY, USA, 381–391. https://doi.org/10.1145/2793107.2793124

[13] Virginia Braun and Victoria Clarke. 2006. Using thematic analysis in psychology. *Qualitative Research in Psychology* 3, 2 (2006), 77–101. https://doi.org/10.1191/1478088706qp063oa arXiv:https://www.tandfonline.com/doi/pdf/10.1191/1478088706qp063oa

[14] Caters Clips. 2019. Realistic Cakes Looks Like Everyday Objects. Retrieved January 11, 2021 from https://www.youtube.com/watch?v=ZOUnLrIFiu0

[15] Kathy Charmaz. 2014. *Constructing Grounded Theory, 2nd Edition*. SAGE Publications Ltd.

[16] Giorgos Cheliotis and Jude Yew. 2009. An Analysis of the Social Structure of Remix Culture. In *Proceedings of the Fourth International Conference on Communities and Technologies* (University Park, PA, USA) *(C&T '09)*. Association for Computing Machinery, New York, NY, USA, 165–174. https://doi.org/10.1145/1556460.1556485

[17] Hyojin Chin, Lebogang Wame Molefi, and Mun Yong Yi. 2020. Empathy Is All You Need: How a Conversational Agent Should Respond to Verbal Abuse. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI '20)*. Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3313831.3376461

[18] Mark Christiansen. 2008. *Adobe After Effects CS4 Visual Effects and Compositing Studio Techniques*. Adobe Press.

[19] "CoFfEeMaN". 1999. Internet Explorer Easter Egg - Blue Screen (v4.0). Retrieved January 11, 2021 from https://eeggs.com/items/731.html

[20] Christian Collberg and Clark Thomborson. 1999. Software Watermarking: Models and Dynamic Embeddings. In *Proceedings of the 26th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages* (San Antonio, Texas, USA) *(POPL '99)*. Association for Computing Machinery, New York, NY, USA, 311–324. https://doi.org/10.1145/292540.292569

[21] Mia Consalvo. 2009. *Cheating: Gaining Advantage in Videogames*. MIT Press.

[22] Brigid Costello and Ernest Edmonds. 2007. A Study in Play, Pleasure and Interaction Design. In *Proceedings of the 2007 Conference on Designing Pleasurable Products and Interfaces* (Helsinki, Finland) *(DPPI '07)*. Association for Computing Machinery, New York, NY, USA, 76–91. https://doi.org/10.1145/1314161.1314168

[23] Leesa Costello, Marie-Louise McDermott, and Ruth Wallace. 2017. Netnography: Range of Practices, Misperceptions, and Missed Opportunities. *International Journal of Qualitative Methods* 16, 1 (2017), 12 pages. https://doi.org/10.1177/1609406917700647 arXiv:https://doi.org/10.1177/1609406917700647

[24] Peng Dai, Jeffrey M. Rzeszotarski, Praveen Paritosh, and Ed H. Chi. 2015. And Now for Something Completely Different: Improving Crowdsourcing Workflows with Micro-Diversions. In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing* (Vancouver, BC, Canada) *(CSCW '15)*. Association for Computing Machinery, New York, NY, USA, 628–638. https://doi.org/10.1145/2675133.2675260

[25] Sayamindu Dasgupta, William Hale, Andrés Monroy-Hernández, and Benjamin Mako Hill. 2016. Remixing as a Pathway to Computational Thinking. In *Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing* (San Francisco, California, USA) *(CSCW '16)*. Association for Computing Machinery, New York, NY, USA, 1438–1449. https://doi.org/10.1145/2818048.2819984

[26] Chris Dias. 2019. Updated 12/23: The repository is open again. Retrieved January 11, 2021 from https://github.com/microsoft/vscode/issues/87440

[27] Jesus Diaz. 2012. The Easter Eggs Are Back in OS X—And This One Is Insanely Great. Retrieved January 11, 2021 from https://gizmodo.com/the-easter-eggs-are-back-in-os-x-and-this-one-is-insane-5929286

[28] Jill P. Dimond, Casey Fiesler, Betsy DiSalvo, Jon Pelc, and Amy S. Bruckman. 2012. Qualitative Data Collection Technologies: A Comparison of Instant Messaging, Email, and Phone. In *Proceedings of the 17th ACM International Conference on Supporting Group Work* (Sanibel Island, Florida, USA) *(GROUP '12)*. Association for Computing Machinery, New York, NY, USA, 277–280. https://doi.org/10.1145/2389176.2389218

[29] Disney / Pixar. 2021. Pixar Animation Studios. Retrieved June 11, 2021 from https://www.pixar.com/404

[30] Paul Dourish. 2003. The Appropriation of Interactive Technologies: Some Lessons from Placeless Documents. *Computer Supported Cooperative Work (CSCW)* 12, 4 (Dec. 2003), 465–490. https://doi.org/10.1023/A:1026149119426

[31] Steve Eddins. 2006. The Story Behind the MATLAB Default Image. Retrieved January 11, 2021 from https://blogs.mathworks.com/steve/2006/10/17/the-story-behind-the-matlab-default-image/

[32] Alison Gazzard. 2012. Re-coding the Algorithm: Purposeful and Appropriated Play. In *Media in the Ubiquitous Era: Ambient, Social and Gaming Media*. 200–214. https://doi.org/10.4018/978-1-60960-774-6.ch011

[33] Eleana Gkogka. 2017. Easter eggs, little delights in UI/UX design. Retrieved January 11, 2021 from https://medium.com/@eleana_gkogka/easter-eggs-little-delights-in-ux-design-fa26911cd8a3

[34] Google Keyword Team. 2018. As the Chrome dino runs, we caught up with the Googlers who built it. Retrieved January 11, 2021 from https://www.blog.google/products/chrome/chrome-dino/

[35] Jason Gunthorpe. 2013. Very well internet, you win. Let me tell you a tale about cow powers. Super o... Retrieved January 11, 2021 from https://web.archive.org/web/20190322061230/https://plus.google.com/113373031907493914258/posts/jGBx4hA26nv

[36] Natalie Hammel and Lorraine Yurshansky. 2015. The Secret Games Hiding Inside Your Phone. Retrieved January 11, 2021 from https://www.youtube.com/watch?v=sOdiWXFF9Ms

[37] Natalie Hammel and Lorraine Yurshansky. 2016. Google Doodle Team Q&A - Your Questions Answered! Retrieved January 11, 2021 from https://www.youtube.com/watch?v=IglWB66vn7g

[38] "hank804". 2014. Excel 97 Easter Egg - Turn Your Spreadsheet Into a Space Sim. Retrieved January 11, 2021 from https://www.youtube.com/watch?v=c6nY0QkG9nQ&lc=Ugjqsci940peDXgCoAEC

[39] Mare Hassenzahl, Axel Platz, Michael Burmester, and Katrin Lehner. 2000. Hedonic and Ergonomic Quality Aspects Determine a Software's Appeal. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (The Hague, The Netherlands) *(CHI '00)*. Association for Computing Machinery, New York, NY, USA, 201–208. https://doi.org/10.1145/332040.332432

[40] Brent Hecht, Lichan Hong, Bongwon Suh, and Ed H. Chi. 2011. Tweets from Justin Bieber's Heart: The Dynamics of the Location Field in User Profiles. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Vancouver, BC, Canada) *(CHI '11)*. Association for Computing Machinery, New York, NY, USA, 237–246. https://doi.org/10.1145/1978942.1978976

[41] Robert A. Hennfng, Steven L. Sauter, Gavriel Salvendy, and Edward F. Krieg Jr. 1989. Microbreak length, performance, and stress in a data entry task. *Ergonomics* 32, 7 (1989), 855–864.

[42] Edward Jung. 2016. Add chrome://dino short URL to link to the internet disconnected network error interstitial. Retrieved January 11, 2021 from https://chromium.googlesource.com/chromium/src.git/+/8d968d89bd70b82af787d1a04dd915a006e59dbf

[43] Matthew Kay and Michael Terry. 2010. Textured Agreements: Re-Envisioning Electronic Consent. In *Proceedings of the Sixth Symposium on Usable Privacy and Security* (Redmond, Washington, USA) *(SOUPS '10)*. Association for Computing Machinery, New York, NY, USA, Article 13, 13 pages. https://doi.org/10.1145/1837110.1837127

[44] Fritz Keinert. 2011. Matlab Overview. Retrieved January 11, 2021 from http://orion.math.iastate.edu/keinert/math561/matlab_demos/matlab.html

[45] Jean-Baptiste Kempf. 2011. VLC Santa Hat? Retrieved January 11, 2021 from https://forum.videolan.org/viewtopic.php?t=96539

[46] Jonathan Klein, Youngme Moon, and Rosalind W. Picard. 2002. This computer responds to user frustration: Theory, design, and results. *Interacting with Computers* 14, 2 (02 2002), 119–140. https://doi.org/10.1016/S0953-5438(01)00053-4 arXiv:https://academic.oup.com/iwc/article-pdf/14/2/119/2284802/iwc14-0119.pdf

[47] "KLYPHY". 2019. Any Easter Eggs? Retrieved June 11, 2021 from https://forums.autodesk.com/t5/autocad-forum/any-easter-eggs/m-p/8615397/highlight/true#M969264

[48] Hannu Korhonen, Markus Montola, and Juha Arrasvuori. 2009. Understanding playful user experiences through digital games. In *Proceedings of the 2009 Conference on Designing Pleasurable Products and Interfaces (DPPI '09)*. Compiègne, France, 274–285.

[49] Robert V. Kozinets. 2015. *Netnography: Redefined.* SAGE Publications Ltd.

[50] Yi Li. 2018. Why add Christmas eggs to the buttons without the developer's permission? Retrieved January 11, 2021 from https://github.com/ant-design/ant-design/issues/13819

[51] Shun Liang. 2018. The Ant Design Christmas Egg that Went Wrong. Retrieved January 11, 2021 from https://blog.shunliang.io/frontend/2018/12/25/the-ant-design-xmas-egg-that-went-wrong.html

[52] Andrés Lucero, Jussi Holopainen, Elina Ollila, Riku Suomela, and Evangelos Karapanos. 2013. The Playful Experiences (PLEX) Framework as a Guide for Expert Evaluation. In *Proceedings of the 6th International Conference on Designing Pleasurable Products and Interfaces* (Newcastle upon Tyne, United Kingdom) *(DPPI '13)*. Association for Computing Machinery, New York, NY, USA, 221–230. https://doi.org/10.1145/2513506.2513530

[53] Ewa Luger and Abigail Sellen. 2016. "Like Having a Really Bad PA": The Gulf between User Expectation and Experience of Conversational Agents. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems* (San Jose, California, USA) *(CHI '16)*. Association for Computing Machinery, New York, NY, USA, 5286–5297. https://doi.org/10.1145/2858036.2858288

[54] Andrzej Marczewski. 2019. 52 Gamification Mechanics and Elements. Retrieved January 11, 2021 from https://www.gamified.uk/user-types/gamification-mechanics-elements/

[55] MathWorks. 2018. MathWorks on Twitter. Retrieved January 11, 2021 from https://twitter.com/MathWorks/status/979766154003730433/
[56] Elisa D. Mekler and Kasper Hornbæk. 2016. Momentary Pleasure or Lasting Meaning? Distinguishing Eudaimonic and Hedonic User Experiences. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems* (San Jose, California, USA) *(CHI '16)*. Association for Computing Machinery, New York, NY, USA, 4509–4520. https://doi.org/10.1145/2858036.2858225
[57] "Michelle". 2014. Bring back Clippy! Retrieved January 11, 2021 from https://word.uservoice.com/forums/271331-word-for-the-web/suggestions/6633892-bring-back-clippy
[58] Microsoft Excel. 2020. Microsoft Excel on Twitter. Retrieved January 11, 2021 from https://twitter.com/msexcel/status/1282798387981672449
[59] Jon Mitchell. 2012. How A Google Engineer Built A Universe In An Easter Egg. Retrieved January 11, 2021 from https://readwrite.com/2012/10/05/how-a-google-engineer-built-a-universe-in-an-easter-egg/
[60] Cleve Moler. 2010. Has MATLAB lost its sense of humour? Retrieved January 11, 2021 from http://walkingrandomly.com/?p=2949#comment-35782
[61] Nick Montfort and Ian Bogost. 2009. *Racing the Beam: The Atari Video Computer System.* MIT Press.
[62] Lou Montulli. 2013. The short history of the "about:" URL. Retrieved January 11, 2021 from http://www.montulli.org/lou/about_urls
[63] S. Moonesamy. 2012. RFC 6694 – The "about" URI Scheme. Retrieved January 11, 2021 from https://tools.ietf.org/html/rfc6694
[64] Mozilla Corporation. 2019. The Book of Mozilla (HTML Source Code). Retrieved January 11, 2021 from https://www.mozilla.org/en-US/book/
[65] David Nagy-Farkas. 1997. *Easter Eggs: Software Surprises.* Abacus.
[66] Michael Natterer. 2012. Goat Invasion in GIMP. Retrieved January 11, 2021 from https://gimpfoo.de/2012/04/17/goat-invasion-in-gimp/
[67] Anne Oikarinen. 2020. Easter eggs and salami attacks – what has your code eaten? Retrieved January 11, 2021 from https://www.nixu.com/blog/easter-eggs-and-salami-attacks-what-has-your-code-eaten
[68] Philip Olson. 2008. A brief history of PHP logos. Retrieved January 11, 2021 from https://web.archive.org/web/20190126211859/http://blog.roshambo.org/a-brief-history-of-php-logos/
[69] Larry Osterman. 2019. Why no Easter Eggs? Retrieved January 11, 2021 from https://docs.microsoft.com/en-ca/archive/blogs/larryosterman/why-no-easter-eggs
[70] Brian Ott and Cameron Walter. 2000. Intertextuality: Interpretive practice and textual strategy. *Critical Studies in Media Communication* 17, 4 (2000), 429–446. https://doi.org/10.1080/15295030009388412 arXiv:https://doi.org/10.1080/15295030009388412
[71] Brandon Paddock. 2010. The so-called "God Mode". Retrieved January 11, 2021 from http://brandonlive.com/2010/01/04/the-so-called-god-mode/
[72] Raghav Patnecha. 2018. How I accidentally opened Google Foobar in almost all possible ways. Retrieved January 11, 2021 from https://www.linkedin.com/pulse/how-i-accidentally-opened-google-foobar-almost-all-ways-patnecha
[73] You Pian. 2018. About the Christmas egg and workaround. Retrieved January 11, 2021 from https://github.com/ant-design/ant-design/issues/13849
[74] David Pogue. 2019. The Secret History of 'Easter Eggs'. Retrieved January 11, 2021 from https://www.nytimes.com/2019/08/08/technology/easter-eggs-tesla-google.html
[75] Christian Schiffer. 2019. Santa Hat on vscode insiders and pushing of religion is very offensive to me. Retrieved January 11, 2021 from https://github.com/microsoft/vscode/issues/87268
[76] Justin Scott. 2008. about:robots. Retrieved January 11, 2021 from https://bugzilla.mozilla.org/show_bug.cgi?id=417302
[77] "shakermaker". 2012. Android 2.3 Easter Egg - Secret Picture. Retrieved June 11, 2021 from https://eeggs.com/items/59225.html
[78] Paulo Silva. 2011. What matlab easter eggs do you know? Retrieved January 11, 2021 from https://www.mathworks.com/matlabcentral/answers/2001-what-matlab-easter-eggs-do-you-know#comment_59598
[79] Matt Silverman. 2010. SFMOGRAPH - After Effects Team. Retrieved January 11, 2021 from https://vimeo.com/8803073
[80] Daniel Simmons. 2018. The Foobar challenge: Google's hidden test for developers. Retrieved January 11, 2021 from https://www.freecodecamp.org/news/the-foobar-challenge-googles-hidden-test-for-developers-ed8027c1184/
[81] Igor Steinmacher, Tayana Conte, Marco Aurélio Gerosa, and David Redmiles. 2015. Social Barriers Faced by Newcomers Placing Their First Contribution in Open Source Software Projects. In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing* (Vancouver, BC, Canada) *(CSCW '15)*. Association for Computing Machinery, New York, NY, USA, 1379–1392. https://doi.org/10.1145/2675133.2675215
[82] "Thosten". 2004. apt-get moo: nicer cow wanted. Retrieved January 11, 2021 from https://bugs.debian.org/cgi-bin/bugreport.cgi?bug=271810

[83] Marnanel Thurman. 2011. Marnanel Thurman on Twitter. Retrieved January 11, 2021 from https://twitter.com/marnanel/status/132280557190119424
[84] Gustavo F. Tondello, Rina R. Wehbe, Lisa Diamond, Marc Busch, Andrzej Marczewski, and Lennart E. Nacke. 2016. The Gamification User Types Hexad Scale. In *Proceedings of the 2016 Annual Symposium on Computer-Human Interaction in Play* (Austin, Texas, USA) *(CHI PLAY '16)*. Association for Computing Machinery, New York, NY, USA, 229–243. https://doi.org/10.1145/2967934.2968082
[85] Joel Tong. 2018. Please remove Christmas Easter Egg for Ant Design framework. Retrieved January 11, 2021 from https://github.com/ant-design/ant-design/issues/13818
[86] Enrique Uribe-Jongbloed, Tobias M. Scholz, and Hernán David Espinosa-Medina. 2015. The joy of the Easter egg and the pain of numb hands: The augmentation and limitation of reality through video games. *Palabra Clave* 18, 4 (2015), 1167–1195. https://doi.org/10.5294/pacla.2015.18.4.9
[87] uTorrent. 2013. On Playing Tetris in uTorrent. Retrieved January 11, 2021 from https://www.facebook.com/utorrent/photos/a.219761958059646/510906845611821
[88] "Wallace". 2012. Goat Excerise. Retrieved January 11, 2021 from http://gimpchat.com/viewtopic.php?f=8&t=16635#p229253
[89] Colin Watson. 2017. Why does man print "gimme gimme gimme" at 00:30? Retrieved January 11, 2021 from https://unix.stackexchange.com/questions/405783/why-does-man-print-gimme-gimme-gimme-at-0030/406169#406169
[90] Jonathan Weinel, Darryl Griffiths, and Stuart Cunningham. 2014. Easter eggs: Hidden tracks and messages in musical mediums. In *International Computer Music Conference Proceedings, 2014*. International Computer Music Association.
[91] J. L. Westover. 2013. Placebo Effective. Retrieved January 11, 2021 from https://www.mrlovenstein.com/comic/364
[92] Valtteri Wikström, Mari Falcon, Silja Martikainen, and Katri Saarikivi. 2019. SynchroMouse: A Game of Improvised Joint Action. In *Conference Companion Publication of the 2019 on Computer Supported Cooperative Work and Social Computing* (Austin, TX, USA) *(CSCW '19)*. Association for Computing Machinery, New York, NY, USA, 418–422. https://doi.org/10.1145/3311957.3359482
[93] Daniel Wilk. 2013. Daniel Wilk on Twitter. Retrieved January 11, 2021 from https://twitter.com/dwilk_AE/status/291395522701438977
[94] Mark J.P. Wolf. 2007. *The Video Game Explosion: A History from PONG to PlayStation and Beyond*. Greenwood.
[95] yuuboku. 2020. yuuboku on Twitter. Retrieved June 11, 2021 from https://twitter.com/yuuboku/status/1283026328497741832
[96] Jamie Zawinski. 1998. easter eggs. Retrieved June 11, 2021 from https://www.jwz.org/doc/easter-eggs.html
[97] Jamie Zawinski. 2011. The secret history of the about:jwz url. Retrieved January 11, 2021 from https://www.jwz.org/doc/about-jwz.html

Received January 2021; revised July 2021; accepted November 2021