Flash and You

1

Introduction

The cover of the 2006 *Time* magazine "Person of the Year" issue featured a glossy, reflective image of a personal computer screen with the controls of a web video at the bottom of a monitor under the bold-faced word: "You." A provocative headline below reinforced the message: "Yes, you. You control the Information Age. Welcome to your world." The article proclaimed the reliance of the web on a broad population of content makers motivated by passion while "working for nothing and beating the pros at their own game" (Grossman 2006). According to *Time*, we'd entered the era of the LOLcat and "i can haz cheezburger"; the age of the YouTube dance sensation and the amateur music video parody; the year of the blogger and the Wikipedia editor. This "Information Age" brought with it web 2.0 and heightened expectations of interactivity from web-based media. Clay Shirky observed that these new activities were this generation's outlet for a cognitive surplus of incredible collective power: a surplus of mental energy once reserved for the consumption of television or gin but now turned toward production (Shirky 2011). With this revolution came new attention to amateur works and the apparent overthrow of traditional models of publishing, thanks in part to new accessible distribution channels. This new web appeared to be inherently democratic, with every voice given an equal opportunity to reach a broad audience: however, any such equality of entry is an illusion, and even the apparent ease of access owes more to the platforms shaping the web than anything inherent in the network. *Time* credited networks such as YouTube and Facebook with providing the tools of this revolution, but ignored the underlying technologies that empowered amateur creation—and the role those platforms played in determining what types of content would be the stickiest and easiest to create by newcomers.

As traditionally defined, a platform is a foundation from which ideas and ideologies are launched. In software studies, Ian Bogost and Nick Montfort define a platform as a hardware or software system that provides the "foundation of computational expression" (Montfort and Bogost 2009). Some platforms appear self-contained: a Nintendo game console, for instance, has a set configuration and manufacturer whose particular rules govern the system's potential. Personal computers vary more wildly in their capabilities thanks to the amalgamation of hardware and the range of operating system choices and software that extend their core functionality. Those capabilities might be termed "affordances"—a term used by Donald Norman to describe the "actual" and "perceived" properties of a thing, whether that thing is a piece of hardware or software or simply a pencil (Norman 1988). This term, which can be used in many ways, is helpful in examining platforms as foundations that offer certain possibilities and in doing so shape the creative works they underlie. Affordances are not just properties of a platform: they become suggestions and frameworks for the works built upon them. When examining the web (and particularly the participatory, "you"-driven web 2.0), a number of different platforms and affordances are at work that are impossible to isolate—and their various affordances have fundamentally shaped the web as a space of discourse, creativity, and interactivity.

The web itself—web 1.0, to some—relies on an agreed language foundation, HTML, for its structure. HTML, or HyperText Markup Language, was first released in 1991 by Tim Berners-Lee as part of the architecture of the World Wide Web (Berners-Lee 1994). That initial set of markup has morphed and evolved over the years as new versions and standardizations of key features made their way into the growing number of web browsers. Classic HTML was static as far as the user was concerned: its affordances were minimal, and early websites reflected strong restrictions on content types that corresponded with low bandwidth on early dial-up connections. Instead of the modern capabilities that rival desktop applications, early HTML provided a means to link files: the hyperlink offers a way either to move between documents on the web or between nodes within a page. When combined with other web technologies, HTML acts as both a skeleton and a lingua franca for more sophisticated display of content. The addition of CSS, or cascading style sheets, separates format from the structure provided by HTML. HTML originally mixed the two, including the infamous blink tag as other tags evolved haphazardly through differing support by browsers, before the expectations of web content demanded formatting and structure well beyond those foundational capabilities. Adding interactivity beyond a link-based graph structure was more complicated; it required using additional scripting languages, or different integrated web platforms, to interpret and make the content responsive.

Among this tangled web of platforms, one stands out as a driving force of the "you" revolution and web interactivity. An Internet user loading up YouTube or an online game in 2006 would likely be launching Adobe Flash Player, one of the options for adding self-contained multimedia or interactive content to a website. Adobe Flash Player is a browser extension that offered a browser-based environment with affordances well beyond the traditional web, extending the palette of online content capabilities dramatically over the years. Users of Flash Player might not even notice their reliance on Flash as, once installed, it became a fairly seamless part of the browsing experience. Flash is a multimedia platform that started as a simple animation package and grew to offer an incredible range of opportunities to author media experiences on the web. Gaming magazine *Edge* pointed out that Flash and the communities it inspired were at the real forefront of participatory culture, even if sites like YouTube were getting all the credit (Edge Staff 2007). Flash game developers cite the platform's accessibility and ease of quick prototyping for artists and the friendliness of its design that allows amateur creators to experiment. In 2005, just before *Time* announced the year of the amateur content creator, Lev Manovich declared that "Generation Flash" had emerged: a new class of media artists that "writes its own software code to create their own cultural systems, instead of using samples of commercial media" (Manovich 2005). He noted that this generation did not necessarily have to use Flash to be part of this movement, but were instead characterized by what he considered "Flash aesthetics": a style that could be translated to Shockwave, DHTML, Quicktime, and many other competing platforms for web multimedia. These other tools have some of the capabilities of Flash: animation, games, and dynamic, responsive web sites. These qualities are not exclusively the domain of Flash. But Flash's contributions, especially as a unified platform, set expectations for web content so high that the name of the platform is embedded in the aesthetic, and Manovich's generation of content-creators and artistic innovators could be rightly labeled as products of the Flash platform—a platform that itself emerged ten years before his writing, and with a very different set of affordances. This convergence of people and platform recalls Marshall McLuhan's often quoted (and often misused) concept: "the medium is the message" (McLuhan

1964). As McLuhan unpacks the concept, "This is merely to say that the personal social consequences of any medium—that is, of any extension of ourselves—result from the new scale that is introduced into our affairs by each extension of ourselves, or by any new technology." Generation Flash is a prime representation of this consequence of extension. The scale of self-amplification and creation that Flash enables likewise amplified the web.

Positioning Flash: A Brief History

Flash began its life as "SmartSketch," a software program that evolved from a tablet drawing program into an animation tool. The program's evolution reflected the search for the problem that Flash was solving: the delivery of dynamic content, not just static pages, over the web. In our interview in 2012, Flash creator Jonathan Gay recounts how the project evolved (see the appendix):

> The main constraint was our time and not understanding what the problem was. Do we put sound in the first version, what kind of features? Initially we built a drawing package that influenced a lot of things, but wasn't a product for the Internet. And then we said, well, we have drawing, let's add animation to it. There weren't a ton of choices, it was more gradual: How do we build it, what do we put in it next? What's the most important thing: what do we build next?

After retooling and refocusing its efforts for web-based distribution, FutureWave Software released FutureSplash Animator on August 19, 1996. Among its high-profile initial clients were Disney and Microsoft, both choosing FutureSplash for its ability to deliver TV-like graphics with small file sizes (Allbusiness.com 1996) for their respective web launches. After four years and a $500,000 investment by its founders, FutureWave was acquired by Macromedia only a few months after the release in December 1996, bringing it to the attention of the wider technical market. With this came the change of the product name that shortened the wordy title to Macromedia Flash 1.0.

The packaging of FutureSplash proclaimed it "a complete web site graphics tool," emphasizing the origins of SmartSketch and not yet acknowledging how the product would come to dominate its bigger brothers. The slogan suggested limitations to animated banner ads and a snazzier version of the "Under Construction" images that were everywhere on the young web as virtual homesteaders figured out how to colonize the media space. The purchase by Macromedia added additional complexity to FutureSplash's identity, as Macromedia Director 5's inclusion of an option to export to Shockwave, a format similar to Flash, made it a viable platform for creating web-based animation and interaction. Director had a long history as a multimedia editing platform, starting out as MacroMind "VideoWorks" with very limited graphic capabilities in the 1980s. MacroMind's founder and leader up to the merger with Macromedia in 1991, Marc Canter, explained the vision that drove his software design: "Because we had a direct connection to the animators, designers and musicians who were using our tools, we knew exactly what features and capabilities they needed . . . One thing we knew—that the world needed end-user 'tools' that could be used by artists, musicians and designers—to create this 'stuff'—this combination of graphics, text, music and interactivity that we knew was possible" (Canter 2003). Director's capabilities, however, were formulated for delivery over CD-ROM, whereas Flash would flourish in its newfound home across the web.

At the time of Flash's acquisition by Macromedia, Director and Flash were competitors with more than a few similarities. Even so, Flash's early adaptations were in response to the needs of the web. Director's powerful capabilities translated into higher download times and a greater bulk in the Shockwave extension, while Flash was more limited but optimized. These optimizations built on key goals set by the team early on, giving Flash an edge in the race to play content. The creative metaphors of Flash and Director show a strong parallel, although Flash's did not extend nearly as far into the moviemaking aspects. After Macromedia was purchased by Adobe, Director stayed desktop-focused while Flash dominated web interactivity and animation. The existence of the two programs within the same ecosystem allowed them to capitalize on each other's design and affordances, to the point where Macromedia and Adobe would both have to clarify use cases for each given the substantial overlap between them. Eventually the ability to embed Flash content in Director further blurred the lines between the platforms.

The 1997 introduction of the Aftershock utility as part of Macromedia's "Universal Media" initiative addressed the technical limitations of authoring a single experience for multiple platforms. The Universal Media initiative described the core aspiration of both Flash and Java for a singular input, universal set of target formats: "With all of the available output alternatives, Director and Flash movies will now be able to be viewed by an unlimited audience" (Macromedia, Inc. 1997). This was also a preview of initiatives to making Flash accessible to non-coders: just as the first version of Flash involved no conception of objects or scripting, Aftershock allowed for the immediate embedding of a finished Flash project into an HTML document without the user understanding the internal construction of either. Aftershock offered a graphic interface for immediate export. Macromedia was clearly aiming at reaching a broader range of amateur creators, who wanted to distribute immediately over the web. This set the foundation for the future, as Canter noted in an interview in 2004: "The evolution of tools has brought us to the point where the entire business models are changing and the essence of what tools are has shifted from something a professional uses, to something everyone will need to know how to use . . . the content in our lives will get treated like content from Hollywood" (MacManus 2004). Such a tool might expedite the work of a professional user, although the limitations were many and included page layout, as seen in the interface. Instead, these interfaces targeted amateurs more familiar with file management systems than programming.

Each version of Flash extended the platform's capabilities and gradually integrated a scripting language alongside the animation framework of the timeline. The release of Flash 5 coincided with a big change in the authoring of Flash: ActionScript was now a programming language, and as such it could be attached to MovieClips, frames, and sprites. The syntax, or format and ordering of the language, also changed drastically. The language moved from the custom set of commands that are closer to the capabilities of the player to a more standardized language on which JavaScript is also based: ECMAScript. The slash syntax, which allowed authors to identify objects like a web address, remained in this version, as did the toolbox. Many programmers encountered Flash as a first programming environment after transitioning from the markup syntax of traditional web development. But Flash 5 was now recognizable even to developers bringing expectations from popular systems programming languages such as C or Java: a few key elements were adopted, including parameterized function calls and local variables that enabled higher-level organization of the code. These features enabled coders to reuse and organize their code. This was particularly relevant to games, which often share a great deal of functionality with other works in a genre. Flash 5 also included the first debugger, which allows programmers to see into the code's state as it runs. Along with these changes to the language, ActionScript was moved from a specific subeditor in the interface attached to MovieClips into its own file extension, ".as." This allowed programmers to use the same code in different projects and helps organize code based on functionality.

ActionScript's introduction, however, also marked an acceleration of complexity as well as an increase in power. First under Macromedia and increasingly under Adobe's ownership, the development of the platform moved away from the original intended purpose: a friendly authoring tool for not-quite-end users. Flash was instead split under market pressures between the needs of enterprise customers and the increasingly attractive mobile markets. The inexorable march of mobile would lead to Flash Lite, while the enterprise market birthed the Flex SDK, whose mission was to make creating relatively similar applications easier. While these tools leveraged the platform's virtual machine, the original runtime provided developers with a means to bypass the turbulent demands of web standards.

Flash was, for the better part of the 2000s, the de facto standard for dynamic online multimedia. Canter described the philosophy behind multimedia as a rejection of arbitrary categorizations in art: "We foresaw multimedia as a new art form to merge the medias together-so you could paint with the violin and make music with the paintbrush" (MacManus 2004). The platform's goals stayed consistent across owners and even through the expansion into enterprise and mobile. However, the accumulated difficulty in establishing a universal language for interactive and creative web experiences proved extraordinarily difficult. Other platforms tried to establish themselves as a universal, and some—including Java—offered similar portability but without the artist-friendly development environment that Flash featured. Flash was designed with professionals in mind, but it was co-opted by an amateur web, with users who in turn created resources and communities that steered the platform through experiments and creative works and into the current expectations for what multimedia experiences delivered on the web are capable of. As Flash's influence has spilled out beyond the browser window, we can see the consequences of the platform's styles and drawing tools everywhere from Cartoon Network to iPad games.

In our examination of Flash's impact, we will be considering Flash as both a platform for developing and for distributing content. Flash refers primarily to Flash Player, which interprets the ".swf" (hereafter SWF, which originally stood for "Shockwave Flash" format but eventually was generalized to "Small Web Format") file type. Most of those files are built using the primary Flash development environment, which has changed dramatically over the years as first Macromedia and later Adobe repackaged and extended its affordances. The SWF file format packages the vector graphics, timeline animation, and compiled ActionScript created within the development environment for interpretation within the browser. "Compiling" code involves transforming source code into something a machine can read. Flash compiles into an intermediate format in a similar way to Java, where a virtual machine interprets the code. The files are designed to be compressed and delivered through the web, so they are difficult to reverse-engineer once compiled. However, Flash Player can also interpret SWF files created by tools other than official Flash development tools, and these too must be understood as part of the platform's ecosystem. The consumer of Flash content sees only the playable material in the browser, regardless of how it was compiled.

Flash as Platform

The year of "You" also happened to be Flash's tenth anniversary, and while Flash itself didn't warrant a mention in that issue of *Time*, many of the networks mentioned—including YouTube—relied exclusively on Flash. Flash's tenth anniversary was celebrated around the web by a community of creators who had seen it evolve from a limited vector animation tool to a fully featured web platform and played a role in shaping that evolution: "Flash is, and always has been, a direct incarnation of what web designers and developers have requested over the years. Macromedia didn't believe in locking a lab full of software engineers away and leaving them to their own devices to improve the product. Instead, they went directly to the people who use the product, and asked them what they wanted. This is still the case today" (Voerman 2006). Flash creators further expressed their love of the platform through their own creative works. The "Nectarine" team released a tenth anniversary animation showing Flash as an evolving superhero for the web, gaining new skills and changing his costume with each release until his 2006 high-performance incarnation. The clean lines and shapes of the animation itself reflected the style that Manovich associated with Flash aesthetics: a style that had over the course of a decade become instantly recognizable.

Those same Flash aesthetics provoke strong opinions: user-experience designer Jakob Nielsen (2000) wrote that Flash was "99% bad" thanks to its almost excessive support for dynamism that encouraged bad design, "gratuitous animation," and the abandonment of information hierarchy in favor of flashiness. Nielsen is criticizing design decisions that can be made using any platform for web development, but singling out Flash as a culprit. This is Flash's scale: the go-to technology of the active and interactive web, responsible for aesthetics both loved and hated. Likewise, those who love Flash have formed loud and active communities sharing new ideas and celebrating experiments, as Flash developer Stacey Mulcahy explained at Flash's tenth anniversary:

Flash has taught me about the importance of resourcefulness and community as a developer. The Flash community is a vibrant and open one, because the members make an effort to give back to the community as much as they have taken from it over the years. Getting help in the Flash community is often just a forum or blog posting away. People are constantly posting experiments or insights that push the boundaries of the technology. It's hard not to be inspired. (Mulcahy 2006)

Flash's popularity as an entry point for beginning programmers, along with this accessible community of knowledge, was vital to its adoption as a near-universal authoring environment for the web. In 2009, Adobe released a survey that suggested Flash was king of Internet multimedia, reaching 99 percent of Internet viewers as opposed to Java's 81 percent (Millward Brown 2009). Adobe declared Flash to be a "pervasive software platform," powered by its community of users and the "2 million professionals" then estimated to be developing content for its player (Millward Brown 2009). We will consider both the user Flash seemed to cater to in features, design, and marketing and the real users who in turn shaped Flash to their needs. Flash, unlike fixed hardware platforms, could adapt to perceived opportunities, reflecting an uncertain development context despite constant growth. The emotional charge the popular platform inspired in its creators was as powerful as the pragmatic and ideological attacks it sustained. The platform's aspirations led it to become increasingly fragmented in its focus, attempting to please users despite disjoint and often conflicting needs. Our approach reflects this multilayered platform's depth and pervasiveness; separating where possible technological limitations from the economic and social forces shaping the platform's evolution with attention to both the development and deployment of content.

Previous volumes of the Platform Studies series have primarily addressed hardware: the Atari Video Computer System, the Nintendo Wii game console, and the Commodore Amiga personal computer. These examples have the advantage of being grounded in finite hardware systems, often with relatively consistent setups and affordances determined by one development group. Flash depends on hardware, but is far from integrated with any particular hardware configuration. Thus, unlike previous titles in the Platform Studies series, our study of Flash will look at how a software platform subsists between creators and their audience, shaping a user's relationship with computer hardware by enabling certain experiences on it and attempting (and often failing) to facilitate others. As Ian Bogost and

Nick Montfort, the series editors, explain it, platform studies is concerned with investigating the connections between hardware, software, and creative works. Flash, consisting mostly of virtualized hardware, is inherently more generalized in its hardware needs. As a software platform, Flash exists in a more abstract space than consoles or personal computers. Unlike hardware, where each iteration is at least in technical specifications a fairly clear progression from the previous generation, Flash's iterations are not necessarily straightforward improvements. Matthew Fuller points out that this is common in software development, which is "not subject to the rigor of the requirement for the 'better and better'" (Fuller 2006). While Fuller defines this trait in regards to software that is not aimed at consumers, the same can be said of any software: "better and better" is difficult to define without clear benchmarks. Flash has evolved repeatedly and often haphazardly through its eleven runtimes, two virtual machines, and four programming languages. Its constant expansion of scope was as much in response to developer desire as it was to rising expectations from end users and stakeholders. In our interview, Gay noted how critical developers were in shaping the platform's early direction:

> It was a gradual realization. There are different ways to define platform. Macromedia acquired Flash pretty early on; we were sitting as the little kid next to Director hoping not to get canceled. It was a gradual realization: "Wow, they have this developer community. This is an amazing thing. You have these people who have all these mental structures about how to use your tool. And the things they do inspire other people and there's this positive feedback loop. And they make money building this stuff." To me, that's the key part of the platform: when you have this community around it, and they are doing things, and they are adding value on top of it, they become invested in it. We saw it in developers first, and "Wow, it looks as if Flash is doing that, too."

Because the Flash platform has been primarily web-based, its reach extended to a wider range of users and thus played a pivotal role in defining gaming outside of dedicated platforms. Our approach will contribute insights into how such web-based platforms innovate through and despite translations onto successive hardware platforms. In particular, we'll consider the feedback loop established by its continually changing capabilities and how its identity in popular culture came about as a direct result of the developers building for the platform rather than those whose jobs were to build the platform. This book positions Flash as both a creative platform and a communal platform. It is not inherently social in the way that the Nintendo Wii is (as addressed by Steven Jones and George Thiruvathukal in their work on that platform), but it is designed as a tool for sharing (Jones and Thiruvathukal 2012). As the processing power and capabilities of personal computers have grown, Flash's own capabilities have correspondingly expanded, sometimes to its detriment, as in the case of the rapid growth of smartphone usage with their more limited hardware. Thus our attempt to capture Flash within these pages is subject to our acceptance of it as a platform constantly in flux, adapting to the changing environment created by hardware, software, and its own users.

Bogost and Montfort suggest there are five traditional layers of study common in our typical approach to media: reception, interface, form/function, code, and platform. The platform is "the abstraction beneath the code," whose construction permeates all levels of influence (Montfort and Bogost 2009). Demarcating the boundaries of the platform is difficult where software is concerned. Software itself transcends code and interface: Manovich defines software as "a layer that permeates all areas of contemporary societies" (Manovich 2013, 15). As a software platform of incredible range and influence, Flash has absolutely permeated through our culture and media, both through its stronghold on the computers it was designed for and in its aesthetic legacy and the altered expectations that resulted from it. Likewise the underlying structures and decisions made at the lowest levels have had a definitive influence on the evolution of content created using it. We'll consider both the layers of Flash as it evolved in abstraction and code serving a function, and the positioning of Flash works and user-developers—the reception and interface area of analysis typical to media studies.

When we talk about Flash, we're talking about an ecosystem of software: the many incarnations of the Flash development environment, the different versions of the Flash Player, the related specifications and the bytecode produced by the Flash compiler and interpreted by the various players. The Flash development environment bridges hardware and operating systems—primarily Mac OS and Windows PC—while the player has steadily targeted additional platforms as they came out. Simply documenting the Flash software ecosystem could fill a volume of this size, and we will not attempt to address this entire range. The full history of Flash is not well-documented, but it does live on in seemingly endless volumes of instructional books dedicated to each iteration of the environment with examples and best practices. We will be pulling from this rich history to assist in identifying the key aspects of the platform, including the instrumental Flash development environment and the application programming interfaces. The philosophy represented by the development environment and the consequences of the key abstractions and metaphors realized by the Flash API permeate the Flash platform, echoing throughout the Flash ecosystem and beyond.

Using the lenses of media studies, critical code, and software studies, we will examine Flash's contribution to the landscape of online media and its role in defining web genres, including "Flashimation," browser-based gaming, and Internet-enhanced applications. Unraveling Flash's history and the role of competing interests of performance, security, developers and users also offers insight into the fate of the next universal languages that hope to supersede its relevance. User-developer involvement historically took place across periodicals, forums, wikis, and newsgroups, which is unique to a platform existent in this period of the web. Consequentially, that is where the focus of our assessment of decision and responses between the communities can glean the most insight. Flash's evolution is particularly useful in understanding the transformations inherent in any non-fixed platform. Its context moved from hypertext to web 2.0, from game arcade to social gaming's cornerstone and from web plug-in to Internet application operating system. Flash's dual role in the critical evolution of aesthetic and procedural affordances has shaped the participatory web and online multimedia, but the platform's ambiguous positioning in the field of openness and its defeat as a web extension in the garden of the iPhone ultimately prevented the platform from maintaining its status as a dominant standard, de facto or otherwise.

Our Approach

The platform studies series is traditionally based on collaboration, in part because of the diversity of disciplinary perspectives necessary to consider the different layers at which a platform exerts its influence. We are likewise from diverse backgrounds: one of us is in media studies and digital narratives, while the other is from computer science and game studies. We are both interested in Flash as a fundamental force that shaped how the web was perceived as well as Flash's role in enabling digital narratives and new forms of gaming. Both of us grew up with the web, making our own experiments in Flash and spending hours on Flash game sites and watching early hits like *Homestar Runner* make their way into popular culture. We have experienced Flash in several ways ourselves: as a development platform and as a tool for delivering content, as a space for artistic expression and a medium for play, as an instructional medium for teaching upcoming coders to create interactive art, and as an apparently diminishing medium still informing the landscape of today's dynamic web. Flash is behind everything from web banner ads and portfolio sites to animation and gaming. Its versatility allows it to impact a range of media. But many of these resulting works render the platform almost invisible, as some users aren't even aware of their reliance on the Flash plug-in until it is blocked. This transparency makes Flash easy to overlook, a tendency that perhaps explains the absence of Flash as more than a technical footnote in the dominant history of media studies: while many tomes have been devoted to the use and application of Flash, none have examined Flash as an object of critical study.

Thus Flash has been noted by several disciplines but rarely studied, thanks in part to that same transparency that makes it an accepted and often unquestioned part of the web's framework. Media studies approaches to the web have often considered the communities and works that Flash makes possible, but not the platform itself, or its role in shaping the ecosystem that makes those communities and their productions possible. The last decade has seen a number of theorists examine the rise of "participatory culture," which Henry Jenkins describes as one in which everyone is active in creating and mediating the culture (Jenkins 1992). This correlates with the democratization of publishing across media, along with the aesthetic impact of this media landscape on more traditional media venues. Studies of participatory culture as part of media often end at the level of reception, or more rarely of interface, rather than delving into the underlying structures. We'll consider these top-level implications of Flash works, but our goal is to better understand the position of Flash as a platform for shaping the discourse of this online (r)evolution.

One of the greatest influences of Flash has been in the space of gaming, and thus Flash games and the emergence of virtual arcades is already part of the discourse of game studies. Game studies is highly interdisciplinary, though typically focused more on the playable object than on the software underneath it. This interface/reception focus is changing, and we will be drawing on the related disciplines of critical code and software studies to further illuminate Flash games and their role in continually pushing the boundaries of the platform's affordances. However, Flash projects are most often released in their compiled form, so we will be discussing both completed works and internal workings where we have been able to acquire the source: this limitation is always a problem when studying software.

Throughout the book, we will zero in on some of the technical elements of Flash that most illuminate the platform's affordances, as well as their context and role in defining the very nature of web interactivity. Flash is a proprietary product, despite the semantic war to position it one way or another. Its creators are inextricably bound to the interests of its owners, Adobe and Macromedia. This dynamic compounds an already complex ecosystem of software layers and competing technologies, but is vital to understanding the insights into the platform's legacy. User-developers are also responsible for creating many of the libraries and resources that are integrated with Flash as a platform, and they must therefore be considered alongside it as part of Flash's role in shaping scripting and procedural literacy in a broader context.

Flash is, however, primarily a platform for the creation and distribution of software. There are several lenses for considering the nature of software: software studies and critical code studies provide an important starting point. Early Flash did not require the manipulation of code as we would recognize it, while the primary language—ActionScript—developed more complexity over the years. Flash is inherently visual, and its code is often intertwined with images, movie clips, and objects in its editor. We will begin by examining those early visual works and delve into the layers of Flash's code—both as a software platform and a platform for the development and release of software. From these roots, we will consider the Flash platform as a force in the production of culture across mediums, returning to the concerns of media studies and the humanities. These many layers are inseparable, though they are often studied in isolation. We hope to bridge that gap in understanding software platforms.

The Book's Plan

It is impossible to address the full scale of Flash works. Many are released anonymously or pseudonymously, and communities dedicated to Flash each host thousands of games, animations, and other works. Thus, we will be using a case study model and looking at a few significant examples chosen for their suitability to illuminate Flash's affordances and the role those affordances have played in constructing our experience of web multimedia. Chapter 2 examines Flash's contributions as a platform for developing animation. In particular, the concept of the timeline and keyframing allowed for the manipulation of MovieClips and the development of new animated shows without the requirements of a studio support team. This gave rise to an entire aesthetic, "Flashimation," with consequences not only for the web but also for the commercial production of animation. We'll look at several examples of Flash animation and their reception as well as their position as a cultural force removed from traditional "gatekeepers" of commercial animation.

As Flash evolved, the demands of web developers and an increasingly dynamic model of the web brought new affordances for experience design. In chapter 3, we'll look at the evolution of the Flash platform for scripting and interactive design through the lens of platformers. Flash has been repeatedly used to recreate and extend Mario's adventures. These tutorials and their developers were part of the force behind the development of Flash web arcades and communities dedicated to Flash as a tool for digital distribution of immediately playable games. Many of these games are part of what Jesper Juul termed the "casual revolution," as they are inherently cross-platform with none of the demands of dedicated hardware or even dedicated installations that accompany most games. The type of gaming that Flash enabled resembles the casual gaming that Jones and Thiruvathukal (2012) describe in their platform study of the Wii. But while the Wii requires hardware and invades the living room, Flash brought casual gaming to nearly any computer, as a background escape in the office. In chapter 4, we'll examine some of the games and genres that emerged through Flash's role in reconstructing the browser as a games portal powered by many user- and developer-sustained communities.

The variety of affordances built in to Flash to accommodate three different development cases—interactive websites, animation and game design—were picked up by experimental creators who transformed and combined them to create new media art. In chapter 5, we'll look at different forms and genres of new media art, including electronic literature and art games, and the strong presence of Flash as a creative tool. There are two primary ways experimenters interacted with the platform: they either took the affordances of the platform and the expectations users brought to Flash works and subverted them in the production of art, or broke out of the affordances and created new systems for their art. We'll examine examples of both models and the implementation of several works, including Stuart Moulthrop's "textual instrument" *Pax* and Daniel Benmergui's wistful art game *I wish I were the Moon*.

While many of the examples under examination in this text clearly owe their existence to the openness and sharing model of Flash's developer community, Flash historically has had an uneasy relationship with openness. In chapter 6, we'll depart from examining a single media artifact to contextualize Flash's stance on openness and standards. A software platform that pervades the web must compete not only on the basis of performance and ease of use, but also in its relevance. Flash's owners systematically opened up parts of the platform in an attempt to maintain their control over the platform as well as the platform's value. Many of these choices reflected an awareness of the value of open source software, as well as the ongoing value that the owning company obtained from the control. The two strategies met in a platform-defining battle over the ability to participate in the success of Apple's own proprietary platforms.

Flash has recently been declared dead, and as of November 2011 Adobe acknowledged it was no longer pushing forward with Flash as a tool for mobile: Adobe noted that it had made a "long term commitment to Flash Player on desktops" but ceded mobile to HTML5 (Parr 2011). However, not only is Flash not dead, but its role as the definitive cross-platform tool for web-based experiences cannot easily be replaced. We'll look at Flash alongside HTML5, and consider the future of Flash—and its legacy—in chapter 7. Flash as we have known it may fade, but its influence continues to be felt on the web. Flash did not merely power the integration of interactive multimedia on the web, it defined it.