Caught in the Game: On the History and Evolution of Web Browser Gaming

Naif Mehanna, Walter Rudametkin

Caught in the Game: On the History and Evolution of Web Browser Gaming

Naif Mehanna
naif.mehanna@univ-lille.fr
Univ. Lille, CNRS, Inria
Lille, France

Walter Rudametkin
walter.rudametkin@irisa.fr
Univ. Rennes, CNRS, Inria,
Institut Universitaire de France (IUF)
Rennes, France

ABSTRACT

Web browsers have come a long way since their inception, evolving from a simple means of displaying text documents over the network to complex software stacks with advanced graphics and network capabilities. As personal computers grew in popularity, developers jumped at the opportunity to deploy cross-platform games with centralized management and a low barrier to entry. Simply going to the right address is now enough to start a game. From text-based to GPU-powered 3D games, browser gaming has evolved to become a strong alternative to traditional console and mobile-based gaming, targeting both casual and advanced gamers. Browser technology has also evolved to accommodate more demanding applications, sometimes even supplanting functions typically left to the operating system. Today, websites display rich, computationally intensive, hardware-accelerated graphics, allowing developers to build ever-more impressive applications and games.

In this paper, we present the evolution of browser gaming and the technologies that enabled it, from the release of the first text-based games in the early 1990s to current open-world and game-engine-powered browser games. We discuss the societal impact of browser gaming and how it has allowed a new target audience to access digital gaming. Finally, we review the potential future evolution of the browser gaming industry.

CCS CONCEPTS

• **Information systems** → **World Wide Web**; • **Social and professional topics** → **History of computing**.

KEYWORDS

web history, web browsers, browser games, game engines

ACM Reference Format:
Naif Mehanna and Walter Rudametkin. 2023. Caught in the Game: On the History and Evolution of Web Browser Gaming. In *Companion Proceedings of the ACM Web Conference 2023 (WWW '23 Companion), April 30-May 4, 2023, Austin, TX, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3543873.3585572

WWW'23, April 30–May 4, 2023, Austin, Texas, US

ACM ISBN 978-1-4503-9419-2/23/04…$15.00
https://doi.org/10.1145/3543873.3585572

1 INTRODUCTION

The web as we know it today is the result of a steady evolution. Accessing the internet allows users to perform a wide array of activities, including browsing interactive web pages, watching movies, or playing competitive video games directly in the browser. However, features that are considered mainstream today are the result of over 30 years of web browser evolution.

In 1990, Tim Berners-Lee created *WorldWideWeb* (later renamed to *Nexus*), which is seen as the first web browser ever created. *WorldWideWeb* enabled non-technical users to access *hypermedia* content through *hypertext*. This first version of the web did not include any graphic content, as this was not its intended purpose. Instead, the `WorldWideWeb` navigator was initially designed to edit and display simple text-based documents on the network. While the documents it supported included styling, their impact on the page was limited and the structure of their styling files was significantly different from current *Cascading Style Sheet*[1] standards.

In 1994, the first browser to natively support embedded graphics was released. *Mosaic*[20] allowed images to be displayed side-by-side with text, while being easier to install and launch than Nexus. This allowed Mosaic to reach a significantly larger user base. The browser rapidly evolved to form the basis of *Netscape Navigator*, which introduced the first version of a scripting language (initially named Mocha, then LiveScript, and finally JavaScript) in 1995 [43], enabling more interactivity in web pages. However, the initial version of JavaScript, which shipped with Netscape 2.0 [1], lacked many of the features intended by its creator, Brendan Eich, when he first prototyped the scripting language. The absence of advanced features was due to the scripting language not being considered a priority for the new version of Netscape Navigator. Furthermore, the language exhibited various bugs that made it relatively unsuitable for advanced use [55]. JavaScript's initial built-in library included a set of general-purpose objects that permitted interaction with the HTML document, setting the base for the first version of the *Document Object Model, level 0* (DOM). JavaScript 1.0 gained traction with web developers, and JavaScript 1.1 quickly followed in the next major release of Netscape Navigator (3.0) with improved performance and features.

In 1995, the release of Internet Explorer 1.0 by Microsoft and the subsequent release of Internet Explorer 2.0 three months later as freeware (unlike its competitors) started what is now known as "the first browser war" [37]. As both browsers competed for market share, new features were rapidly introduced. Netscape's release of JavaScript prompted Microsoft to announce the release of its own implementation of the scripting language in Internet Explorer

[1] https://www.w3.org/Style/CSS/Overview.en.html

3.0: JScript. JavaScript and JScript later became implementations of the ECMAScript standard, although they supported different sets of features due to the frequent feature releases of both browsers. Although JavaScript was maturing, developing dynamic content was still challenging due to compatibility issues between the two leading browsers. Content that was optimized for one browser could look different on the other, leading web developers to prioritize one browser at the expense of the other. By 1996, browsers were evolving rapidly, with customization options becoming available. This was exemplified by Internet Explorer 3.0, which introduced support for *Cascading Style Sheets* (CSS), with Netscape following suit in its fourth iteration. With the *World Wide Web Consortium (W3C)* recommending CSS, HTML being established as the standard document format, and the definition of the ECMAScript standard, web developers gained the tools needed to transition from static web pages to dynamic and reactive websites. These advancements paved the way for the emergence of browser-based games. Starting as a limited platform for game development, web browsers evolved to include complex game mechanics, such as multiplayer games that relied on real-time interactions between players, becoming ideal platforms for reaching a growing number of players. Browsers made it easy for both amateur and professional game developers to distribute their games.

In this paper, we present the history and evolution of browser-based games. In Section 2, we explore how the lack of features in early browsers only allowed games that did not require advanced interactivity. Section 3 presents the fast-paced evolution of the web and support for third-party plug-ins in browsers, such as Flash and Java, which allowed a significant leap forward for browser game development. Section 4 dives into the capabilities offered by HTML5 and its native support of 2D and 3D graphics, enabling the development of cross-platform rich games directly in the browser. Finally, in Section 5, we discuss the impact and future of browser games.

2 EARLY BROWSER GAMING

The emergence of consumer-friendly web browsers in the mid-1990s allowed for the rise of game-related websites. However, early browsers had limited features and standards, and their lack of client-side scripting made interactive experiences slow, especially for users with dial-up modems. These limitations made the development of browser-based games challenging.

Text-based games, which had been around since the early days of terminal-based computers, were able to take advantage of the limited features of early browsers. Despite their simple interfaces, text-based games attracted a significant number of players and initiated the era of browser games.

As web browsers evolved rapidly, web developers worked on standardizing and making new features accessible. Client-side scripting, in the form of JavaScript and the ECMAScript standard, was introduced by Netscape 2.0 in 1995, as explained in Section 1. Over the following years, Internet Explorer introduced its implementation of client-side scripting (JScript) and the first standardized version of Cascading Style Sheets (CSS).

These new tools allowed the emergence of new types of browser games that took advantage of increased interactivity allowed by JavaScript, its Document Object Model (DOM) APIs, and standardized styling. In this section, we explore how text-based games took advantage of the early browsers' limited capabilities and how newly introduced features of web browsers enabled the emergence of dynamic games using Dynamic HTML (DHTML).

2.1 Interactive fiction

Text-based games, also known as interactive fictions, have been around since the late 1960s with the advent of mainframe computers. These games allowed players to take part in adventures by reading textual descriptions of events and entering a textual representation of their chosen actions. The genre's popularity rose with the introduction of affordable home computers and terminal monitors that were only capable of displaying text-based content.

When web browsers were first introduced, operating systems already supported advanced graphics capabilities. However, early browsers lacked the ability to display complex graphics, which made browser-based text games seem like a step back for the gaming ecosystem. This was because home computers already supported more advanced games that were comparable to console games in terms of performance. As a result, browser-based interactive fictions were less appealing to many gamers. However, web browsers offered online connectivity and platform-independent compatibility. As a result, browser-based interactive fictions started to appear. They were easily accessible through their *Uniform Resource Locator (URL)*, eliminating the need to purchase and install the game. This gave browser-based interactive fictions an edge over other gaming systems, attracting large communities of players who could share their experiences in online forums, bulletin boards, and later in social media. Figure 1 displays a screen capture of *CyberMUD*, an interactive fiction developed in 1994 using HTML. Players interact with the game through *hyperlinks*.

Another key factor that contributed to the rise of online interactive fictions is the emergence of *server-side programming*. With the release of PHP and MySQL in 1995 [51], dynamic websites became possible, allowing for more advanced user interactions. Without server-side programming, websites would have remained static and unable to react to user input, which was essential for the interactive fiction genre, and particularly for online interactive fictions.

One of the earliest successful online text-based strategy games was *Earth: 2025*, created in 1996 by Mehul Patel. This game allowed players on the internet to belong to virtual countries that they managed and defended against other players, and it gathered a significant user base during its lifespan, remaining active until 2009. In 1997, *Hattrick* and *Alien Adoption Agency* were both released and also attracted numerous players.

While online text-based games have lost popularity due to the improved abilities of browsers, the game genre still persists to this day on various websites. *TextAdventures*[2] is one such website that allows current internet users to play interactive fiction.

2.2 The DHTML era

The introduction of client-side scripting changed the shape of the web. At the end of the 1990s, the main way to access the internet was through dial-up modems, which came with several limitations, such

[2] http://textadventures.co.uk/

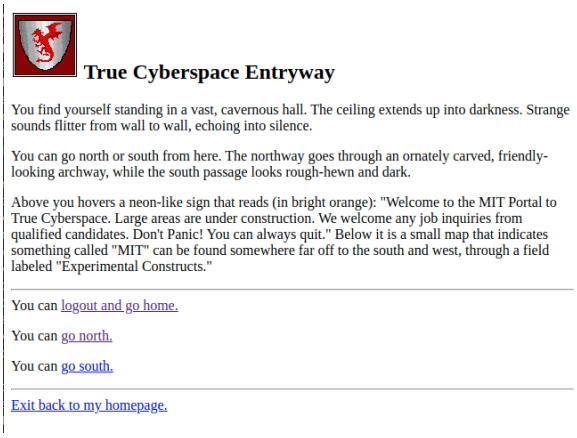

Figure 1: CyberMUD, an interactive fiction driven by hyperlinks

as low bandwidth and relatively high latency due to their 56 kbits/s speed limitation [48]. Before client-side scripting was available, validating any information on the web page took a full round trip to the server, which significantly limited the user experience. Client-side scripting allowed user interactions to happen directly in the browser. The first release of JavaScript on Netscape 2.0 was limited, only allowing text elements to be manipulated by the language, but web developers quickly saw the potential of JavaScript for web development. Although JavaScript was initially limited to text elements, it allowed the creation of a number of games that relied on *ASCII* animations running in a *textarea* element. An example of such a game can be found in the black and white ASCII implementation of *Space Invaders*, which ran directly in the Netscape browser and is still playable today [2].

The release of Netscape 3.0 and JavaScript 1.1 brought several much-needed improvements to the scripting language. One such improvement was the ability to change the source attribute of image elements, which had a significant impact on the development of browser games. Real-time animations allowed developers to create games that provided an arcade-like experience. As early as 1996, DHTML versions of the notorious *Pac-man* were developed, followed by reproductions of other arcade games like *Space Invaders* (non-ASCII reproductions) or *SmoothMaze*. However, most full DHTML-based games were developed for demonstration purposes and didn't achieve wide-scale distribution.

There are several reasons why DHTML games never truly took off. First, slow speeds of dial-up modems resulted in long loading times, as resources needed to be loaded before any JavaScript-based game could start. As more complex games required a higher number of assets, slow internet speeds were seen as a significant limitation to DHTML games. Second, competition with plug-in-powered games highlighted many limitations of JavaScript games. They could not compete with the animations and performance offered by Java or Flash-based games (see §3). Finally, although the competition between Netscape and Internet Explorer led to many features being developed quickly, standardization efforts of the scripting language, as recommended by the W3C, were less of a priority. This made JavaScript-based game development complicated, since features might behave differently between browsers. The quick pace of browser development also introduced various breaking changes to the different APIs, making JavaScript-based games prone to breakage and incompatibility

3 THE RISE IN POPULARITY OF BROWSER GAMING

The release of Netscape 2.0 in 1996 introduced features [5], such as JavaScript, that focused on improving interactivity on the web, as discussed in Section 2. However, more notably for gaming was the release of the *Netscape Plug-in API (NPAPI)* and the resulting support for Java Applets, and later, Flash applications. The *Plug-in API* allowed the browser to interact with a wide array of new content types that would have otherwise remained unavailable to web browsers. The browser delegated content it didn't understand, such as documents, programs or multimedia, to external plug-ins if they were available. The result was a significant improvement in the extensibility of the Netscape browser, and the availability of plug-ins such as the *Java Runtime Environment*, *Adobe Acrobat PDF reader*, *Apple Quicktime*, and *Macromedia Director* that opened up new possibilities for browsers with performance comparable to native applications. Interestingly, the *NPAPI* interface was not limited to Netscape's proprietary plugins. It also allowed third-party developers to provide their own frameworks to enhance the browsing experience.

In the next subsections, we explore the evolution of Java applets and the significant advancements they introduced to browser gaming before exploring the enduring impact of the Flash plug-in.

3.1 Java-applet games

The Java programming language was created by Sun Microsystems in 1995, only one year prior to its introduction as a plug-in in Netscape Navigator 2.0. One of Java's main advantages is that it's cross-platform; code written in Java is executed by the *Java Virtual Machine (JVM)* rather than directly by the operating system. Sun Microsystems coined the slogan *"Write once, run everywhere"* to market this idea. The portability of the language made it an ideal candidate for browser integration as it could be run on any operating system given where the JVM had been ported to. Java was first introduced to the web in the *HotJava Browser* [3], developed by Sun Microsystems to promote their vision of a web built around Java applets. Netscape's support for Java applets came at a time when Java was gaining popularity and being promoted by major players in the IT industry, such as Sun Microsystems, Apple, and Microsoft.

Java applets revolutionized web development by enabling web pages to be developed as interactive applications, rather than just static pages with information. The HotJava Browser's implementation of the JVM allowed Java classes to directly interact with the page's HTML content, unlocking the full capabilities of the host machine to modify the page's content. Despite serious drawbacks, such as high memory usage and increased processor load, as well as a long history of security issues, the benefits that Java brought to web development arguably outweighed these limitations.

Java applets gained significant popularity with the launch of the Java NPAPI plug-in in 1996. Not long after, Microsoft joined the trend with the introduction of NPAPI plug-in support in Internet Explorer 3.0. By 1998, both major browsers supported the execution of Java applets [4].

Java Applets started a new era of games in the browser, as interactivity is a key component of any game. Their introduction marked a significant breakthrough as they offered animations, audio, and real-time interactivity, surpassing the limitations of JavaScript. While JavaScript was too slow to render game-grade animations and its adoption remained limited in the gaming ecosystem, Java Applets paved the way for immersive browser-based gaming experiences. Moreover, Java applets greatly reduced the burden of incompatibility between browsers, as the JVM executed Java code much more consistently across platforms.

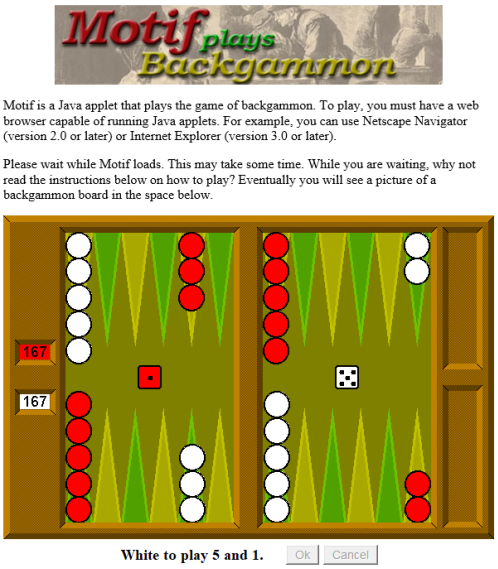

Figure 2: The Backgammon game as a Java applet

Java graphics were based on bitmap images, and games developed using Java Applets initially had simple graphics. Figure 2 shows an example of a *Backgammon* game that was available in 1998. At that time, limited internet speeds meant games with higher resolution graphics, being larger in size, took significantly longer to load. However, with the advent of broadband internet, which improved speeds, Java-based browser games naturally evolved to include better graphics and 3D assets. In addition, lower network latency allowed for improved online connectivity in web pages, leading to the release of several *Massively Multiplayer Online Role-Playing Games (MMORPGs)*. One of the most popular MMORPGs at the time was RuneScape,[3] which was written in Java and ran as an applet in the browser. The game initially included a mixture of 2D and 3D graphics before turning into a fully 3D game in later releases. Another example of a game utilizing the capabilities of Java Applets is *Minecraft*, which was released in 2009 as a 3D game with online connectivity and quickly gathered a significant user

[3]https://play.runescape.com/

Figure 3: Minecraft Classic in the browser

base. The game allows players to build their own world with almost no limits. Figure 3 shows an example of Minecraft.

While Java provided web developers with improved tools to build powerful games running in the browser, applets started to lose popularity in the early 2010s. The official demise of applets was initiated in 2013 when Google Chrome, which had already become the most popular browser, began dropping support for the *NPAPI* that most plug-ins, including Java, relied on. There were several reasons for this, including a dependency on the JRE's version, numerous security issues, and the increased capabilities of JavaScript and Google Chrome's V8 engine, which allowed JavaScript to be compiled to machine code, thus improving its performance. Java applets were officially deprecated by Oracle in 2017 with the release of the Java Development Kit 9 (JDK9) [47].

3.2 Flash-based games

In 1996, *FutureWave Software* created the *Adobe Flash Player* (initially named *FutureSplash*) to allow developers to design interactive animations through a simple and accessible interface. *Flash* was first introduced in Netscape 2.0 as part of a featured list of plug-ins following the release of its *NPAPI* interface. In 1997, the plug-in was acquired by Macromedia [6], which continued its development and extensively marketed it, leading to its quick rise in popularity. Macromedia was later bought by Adobe in 2005 [8], and the plug-in was rebranded as the *Adobe Flash* plug-in.

While providing cross-environment support for its graphics and animations, Flash also came with a series of advantages over Java applets that led to its dominance in the browser-gaming ecosystem. One of the main advantages of Flash was its utilization of vector graphics, which resulted in a reduction of storage and distribution requirements for graphics when compared to bitmap images. In the era of high latency and low bandwidth, the loading times of animations were greatly reduced [49] in Flash applications, improving the user experience. Another advantage of Flash was its widely supported export format (SWF), which allowed for easy sharing of the created resources. Alongside the Flash plug-in, various tools,

such as *Adobe Animate*,[4] were released to build and develop interactive content without programming knowledge, allowing content creation to be more accessible. By 2002, Flash was installed in over 98% of browsers [40] and widely used across the web.

Interestingly, initially, Flash was mainly used for building interactive and animated content for web pages. Flash did not allow users to script actions: while it provided the ability to add buttons, the components themselves could not do significantly more than skip animation frames, limiting its use for game development. Macromedia vowed to surpass this limitation by introducing *ActionScript* in 2000. ActionScript was based on the ECMAScript standard and was therefore syntactically close to JavaScript, allowing web developers familiar with the latter to quickly start developing advanced interactive content for Flash. With the introduction of ActionScript, Flash games started appearing on various websites. The first platform to allow users to upload their Flash games was Newgrounds [21]. At its peak, Newgrounds had over 80, 000 games [39], virtually every game genre, ranging from adventure games to educational games. In a time when social media platforms were just beginning, Newgrounds provided users with a platform to express themselves. As such, various games were built as a reflection of current political events, such as the famous *Bad Dudes vs. Bin Laden* game [39], developed in reaction to the *9/11* attacks.

Flash game development became more accessible, leading to an explosion of the number of games created. Although it is impossible to quantify the exact number of Flash games developed during its 24-year lifespan, Flashpoint [16], a game preservation project, currently lists over 100, 000 Flash games in its database, making it the largest game library ever created for a single platform. For comparison, the *Playstation 2* had 3, 874 games developed for it during its lifespan. The accessibility of Flash game development also gave rise to the indie game scene, with games originally created in Flash later being ported to various consoles. For instance, *Super Meat Boy* (shown in Figure 4), whose Flash graphics style is still recognizable in current versions of the game, and *Canabalt*, which is playable on various consoles such as the *Playstation 3* and the *Playstation Portable*.

Flash's popularity peaked during the first decade of the 2000s, with many high-impact games created during this time. *Stardoll* [32], a fashion game that focused on its online community, had over 400 million users as of 2016, while *Neopets* [31] (1999), where users care for virtual animals, persists to this day in Facebook's metaverse. *FarmVille* [30], launched in 2009 on Facebook's game platform, attracted over 34 million daily users who managed their virtual farms and quickly became one of the highest-grossing games on Facebook's platform.

However, Flash's advantages were eventually outweighed by its disadvantages, including security issues introduced by the plug-in [34]. Additionally, the growing adoption of smartphones, especially Apple's iPhone, which did not support Flash due to closed-source code, high CPU usage, and extensive security issues as cited in their 2010 blog post *"Thoughts On Flash"* [11], led to its decade long decline. In July 2017, with the Flash install-base representing only a fraction of its peak, Adobe announced the scheduled end of Flash by 2020 [14]. New web standards have since been introduced,

[4]https://www.adobe.com/products/animate.html

Figure 4: A comparison between the Wii (left) and Flash (right) versions of Super Meat Boy [10].

covering much of the technical landscape that Flash was conceived for with, arguably, less drawbacks including being directly provided by the browser.

4 HTML5 AND THE ADVENT OF THE CANVAS

Over the years, the HTML markup language has evolved multiple times in an attempt to provide more advanced standardized features. The first major change to HTML was in 1997, with the release of HTML 4.0 [52], which introduced official support for CSS documents and dynamic scripting using JavaScript. Web usage continued to grow, with many users adopting the internet and increasingly relying on it to access information, purchase goods, or for entertainment. The web required more interactivity, and Adobe Flash and Java applets were created as solutions to provide dynamic content (§3). However, they relied on external plug-ins and lacked standardization. Despite its limitations, the web was quickly becoming more dynamic, and many web pages began providing interactive content. Many websites could be compared to native applications in their complexity and features. In response, an early draft for *Web Applications 1.0* was published by the *Web Hypertext Application Technology Working Group (WHATWG)* in 2005 [9]. After years of conjoint work between the *WHATWG* and the *W3C*, the Web Application 1.0 specification evolved in 2008 into what is known today as HTML 5.0, an official standard of the W3C. This was an important turning point, as web applications began to achieve features comparable to native applications. HTML5 provided many advantages. Native video and audio support allowed websites to distribute multimedia content without relying on external plug-ins, all while supporting more formats [41]. The *canvas* element was officially introduced, allowing web pages to natively integrate 2D graphics and permitting game engines based on HTML5 to emerge. Both of these new additions were enough to make the Adobe Flash plug-in obsolete.

In the following subsections, we cover the capabilities enabled by the *canvas* element for browser gaming. We will then cover the official support of *WebGL*, which allows hardware-accelerated 3D graphics to be incorporated directly into the browser, paving the

way for an entirely new dimension of browser games. Finally, we will explore cloud-based browser gaming, made possible by improved video capabilities and the adoption of fiber optic broadband, which significantly increases bandwidth and reduces latency.

4.1 The Canvas element

Although the canvas element became a standard element of HTML5, it actually originated as a non-standard element in the Webkit engine,[5] which was part of the Safari browser back in 2004 [7], four years before the official release of the HTML 5.0 specifications. Before the introduction of canvas, most browsers natively supported bitmap images, *Scalable Vector Graphics (SVG)* images, and gradients. Notably, Internet Explorer did not support SVG and instead supported the *Vector Markup Language (VML)* which never really took off.

The canvas element allowed browsers to draw graphics in real-time and could be controlled directly through JavaScript, which also has access to the page's DOM. This provided greater interactivity and animations without the need for external plug-ins. The canvas element was supported by all HTML5 compliant browsers, including mobile devices. However, as rendering frames was dependent on the underlying JavaScript code, canvas-based animations relied on JavaScript's performance. In its early stages and up until the introduction of the V8 engine by Google Chrome, JavaScript was not known for its execution speed. The V8 engine significantly improved JavaScript performance and helped establish the scripting language as a solution for fast web applications. In 2012, hardware-acceleration was added to 2D canvas-based graphics [12]. This significantly improved drawing times and helped avoid CPU bottlenecks. The drawing instructions can be offloaded to the GPU, if available and supported, drastically increasing rendering speeds. These advances made canvas-based graphics more attractive to game developers, and canvas-based game engines started to appear. These engines provided tools that developers needed to build their games, such as physics engines, making browser game development faster and simpler. This bridged some of the gap between browser-based games and native games while remaining cross-platform, centralized and not requiring users to install the games.

Phaser [23] is an example of a popular 2D game framework that was developed for HTML5. The framework uses JavaScript, making it accessible to web developers that are already familiar with the language. To date, Phaser powers a significant number of browser games that range in popularity. Figure5 shows a screenshot of *Be A Hero Again*,[6] a game developed with the Phaser framework. The game features cartoonish, colorful graphics that are rendered using the canvas element. *MelonJS*[7] is an open-source game engine that provides the tools to build a 2D canvas-based game, such as a physics engine and support for various inputs and sensors.

It's worth noting that the improvements in graphics and speed due to the HTML5 standards and the arrival of 2D game engines for the canvas element brought browsers closer in line with what was provided by the 2D graphics of the Adobe Flash plug-in years before.

[5]https://webkit.org/
[6]https://beaheroagain.com
[7]https://melonjs.org/

Figure 5: Be A Hero Again, a 2D canvas game developed with Phaser.

4.2 Hardware acceleration in the browser

Although native canvas capabilities allowed users to create interactive 2D games, PC and console-based games both offered 3D graphics. Furthermore, even recent versions of the Adobe Flash plug-in allowed developers to create GPU-powered 3D games. There was a need to introduce a standard for 3D rendering directly into the browser. In 2006, Vladimir Vukićević laid the groundwork for the first prototype of the *OpenGL 3D context* in the canvas element and later turned to the Khronos group [18] to create *WebGL*. WebGL 1.0 was officially released in 2011 and was supported by all major browsers. It added support for native 3D graphics in the browser and used the OpenGL ES 2.0 rendering API, a subset of the native OpenGL API for embedded systems. Games created with WebGL could provide a level of graphics performance that was on par with desktop games. The significant advantage of WebGL lies in the fact that it is a standardized feature of HTML5, making it possible to exploit 3D capabilities on every platform supported by compatible browsers. WebGL 2.0 was officially introduced a few years later, in 2017, supporting new APIs and based on the more recent OpenGL ES 3.0.

WebGL is an interface to interact with the OpenGL rendering API. For game development, WebGL represents a significant advancement, as it allows for the creation of 3D games with graphics that are on-par with desktop games. This simplifies the process of obtaining and installing a game in order to play it and attracted the interest of developers. However, writing web applications directly in WebGL is complicated due to its low-level nature, leading to the emergence of libraries and game engines that provide higher-level abstractions. Many major game engines now support WebGL. For example, Unity [26] and the Unreal engine [27] both offer support for 2D and 3D browser games based on WebGL. Notably, Unity previously provided support for 3D browser gaming through its Unity Web Player plug-in, but lost this capability when NPAPI ceased to be supported [13].

In addition to the established game engines that were ported to WebGL, others were built specifically for the development of

Figure 6: RoboStorm, a WebGL powered browser game.

WebGL-based games. PlayCanvas,[8] for instance, is a popular example and provides a fully capable editor (PlayCanvas Editor) for developing in-game 3D models and interactions. Being fully written in JavaScript, PlayCanvas has the advantage of not requiring compilation, as is often the case with other established game engines that primarily use the C++ language for development. Robostorm,[9] as shown in Figure 6, is an example of a 3D game that was developed with PlayCanvas. It offers online connectivity and advanced 3D graphics directly in the browser.

The popularity of WebGL is growing and new standards for browser-based 3D graphics are being developed. The *WebGPU* [29] specification, set to replace the WebGL standard, is currently in the working draft phase by the W3C. *WebGPU* is covered in greater detail in Section 5.2.

4.3 Cloud-based gaming

Cloud gaming has emerged as a competitive way to distribute games. Initially presented as a concept by *G-Cluster* at the 2000 E3, cloud-based gaming gained in popularity towards the late 2010s as high-speed internet adoption grew among gamers, making input latency acceptable [36]. Multiple platforms have proposed cloud-based gaming [35], such as Onlive and Gaikai, which was ultimately acquired by *Sony* [24] to form *PlayStation Now* and *PlayStation Plus* subscription services. Cloud gaming has made many popular AAA games available in the browser. However, not all cloud gaming companies have provided browser-based access to their services [35].

Cloud gaming overcomes issues developers face when porting to WebGL. For example, porting native OpenGL games to WebGL can be problematic because WebGL relies on the OpenGL ES subset API for embedded systems. Porting from other APIs requires extensive rewrites. Instead, cloud-based games do not run directly in the browser, they are streamed from powerful servers thanks to the video and audio elements in HTML5, side-stepping portability and hardware performance issues. Players only need high-speed connectivity and enough processing power to decode streamed content. Cloud gaming in browsers is made possible by the support of video and audio elements in HTML5. Nevertheless, despite the advantages, cloud gaming suffers from a relatively limited adoption, arguably due to:

(1) High-speed internet access remains highly disparate. Some regions have fiber internet access, while many others still rely on ADSL [22]. Cloud gaming requires low-jitter, low-latency, high-speed networks to be effective. Limited internet access reduces the reach of the system.
(2) The upcoming WebGPU specification promises better performance, and with the growing processing power of portable devices, WebGPU might overcome the limitations posed by WebGL. A better 3D API in the browser, with more powerful devices, could reduce the appeal of cloud gaming.

Currently, two of the most popular browser-compatible cloud-gaming services, *Nvidia's GeForce NOW* and *Microsoft's Xbox Game Pass*, announced a total of 20 million players each in 2022 [19, 25].

5 DISCUSSION

5.1 Impact of browser gaming

The widespread adoption of browser-based gaming early in the internet's history can be attributed to various reasons. Games for browsers were optimized for the low internet speeds of the era, resulting in an overall positive gaming experience. Flash's vector graphics, for instance, allowed games to load quickly, even on dial-up modems, making them accessible to a high number of players. As the internet grew and more people purchased computers for general use, browser games were typically able to run on them using the limited amounts of processing power available, making them widely accessible. Additionally, most browser games were offered free of charge on platforms such as *Newgrounds* or *Armor Games*, which provided vast catalogs of free-to-play games. The price of high-end gaming computers and home consoles often made them inaccessible, whereas browser gaming appealed to a wide audience. It also required little setup beyond installing a plug-in, which was often pre-installed with the browser. Browsers proved to be ideal platforms for casual gaming.

The rise of video games on the browser was not limited to single-player casual gaming. Many *MMORPG*s emerged, connecting players from different regions of the world. *RuneScape*, mentioned in Section 3, was a highly popular browser-based MMORPG that gathered millions of users, some of whom would play for hours on end. Another online MMORPG that attracted over 10 million monthly players was *Habbo Hotel*,[10] where users played in a virtual hotel.

In Section 3.2, we discussed how Flash games were a significant precursor to the indie game scene, allowing small game creators to develop popular games that were monetized and ported to multiple platforms. *Super Meat Boy*, for instance, sold millions of copies when it first debuted on the *Xbox* platform. Moreover, *Minecraft*, which originated as a browser game (§3.2), ranks among the most popular games of all time according to a 2021 ranking by *HP* [42]. Flash was accessible and straightforward, allowing even novice developers to create games and animations with relative ease. This simplicity

[8] https://playcanvas.com
[9] https://robostorm.io
[10] https://habbo.com

enabled users to express themselves through Flash creations, especially games, many of which satirized political situations when social media platforms were not yet as widespread as they are today.

Web games span a wide range of genres, including educational, and even, adult-oriented games. Educational games, in particular, have the potential to foster new learning strategies by creating interactive and educational content that can be distributed over the internet and reach a broad audience. Browser games can also benefit scientific research [38], as seen in examples such as *Stall Catchers*,[11] aimed at benefiting medical research, and Phylo,[12] designed to decipher human DNA.

Today, the gaming industry is bigger than both the movie and music industries combined [50], showing that video games are an integral part of our entertainment culture. A significant part of this success can be attributed to the rise of social games on Facebook, which gained popularity due to their simple mechanisms that attracted many first-time gamers, as well as their use of the social graph and asynchronous play [54]. These features allowed users to play at their own pace while still sharing their gaming experience with friends, further contributing to the popularity of social games and browser gaming in general.

5.2 The future of browser-based games

Browser-based games have come a long way since the release of the early text-based games, and they continue to evolve. In 2019, the *Unreal Engine 4* dropped native support for HTML5 and instead provided an extension to their engine [15] to cross-compile the C++ code to JavaScript. This move signaled a disinterest in WebGL-based games. As WebGL is based on the OpenGL API, its limitations are starting to show as the world moves towards more general-purpose uses of GPUs, such as crypto-mining or training neural networks. Thus, there is currently a need to develop a new up-to-date standard that better exploits modern GPU APIs. One such standard is *Vulkan* [28].

The upcoming *WebGPU* standard is based on three current GPU APIs being developped: *Vulkan* by the Khronos Group, *Metal* by Apple [33], and *Direct3D 12* by Microsoft [17], which provide significant advantages over OpenGL, on which WebGL is based. In their work, Lujan *et al.* [44] show that the Vulkan API significantly outperforms OpenGL in various scenarios. Therefore, WebGPU is expected to open a new era for 3D games in the browser, leveraging the advantages of the three GPU APIs, each adapted to their respective platforms.

The *WebXR Device API* [46] is a working draft developed by the W3C. WebXR standardizes access to *virtual reality* (VR) and *augmented reality* (AR) devices. Interest in VR has been particularly high since easily accessible VR equipment and games reached the market. Multiple use cases can be found for VR and AR, such as GPS navigation, educational content, immersive gaming, among many others that can leverage the technology to provide better content.

Current mobile devices, complemented by a supporting device, are able to provide convincing virtual-reality experiences, prompting game developers to create more content for these systems. Additionally, Meta's announcement of its *Metaverse* in 2021 [45] further supports the need for a VR and AR standard [53]. There are already a number of promising games that currently make use of the WebXR experimental API, such as *Space Disaster*[13] or *Moon Rider*.[14] However, it remains to be seen if, or when, these APIs will become popular. The technology and the possibilities are very exciting, but they could still go the way of 3D television, or maybe remorph into something new.

6 CONCLUSION

In this paper, we reviewed the past, present, and future trends of browser gaming, highlighting the technologies at hand. The history of browser-based web games provides a unique perspective on the evolution of the web, and as web browsers evolve and adopt new features, browser game development has grown. Web games have pushed the boundaries of what browsers can do, leading to the development of new features and, quite importantly, their standardization across platforms.

From the early days of text-based games on the Netscape browser, to modern, 3D MMORPGs, browser games have been developed to account for the limitations and requirements of the era. Initially, text-based games did not even include graphics due to the lack of support from browsers. Later, the emergence of DHTML games took advantage of the new dynamic capabilities of browsers to build interactive games. The introduction of the Java plug-in and applets enabled developers to create complex games, comparable to their native counterparts, while the Flash plug-in empowered an enormous user base to create novel games with advanced animations. With the advent of HTML5, the browser now natively supports 2D and 3D graphics, allowing for console-grade graphics with WebGL.

Overall, the past and present of browser games have been exciting, and the future looks even more promising as new technologies and standards continue to be developed. Moreover, the future introduction of WebGPU is set to further reduce, or even eliminate, the gap between browser-based games, native PC games, and game consoles. Additionally, the *WebXR Device API* standardizes access to virtual and augmented reality devices, which could open up new possibilities for immersive gaming experiences. With widespread access, browser games have become a popular and ubiquitous form of entertainment and an important part of our culture. The advancements made in browser game development have not only impacted the gaming industry but have also contributed to the evolution of the web as a whole.

ACKNOWLEDGMENTS

We thank Felipe Pepe, Romain Fouquet and Lunar for their invaluable feedback during the writing of this paper.

This work has been financially supported by the *Agence Nationale de la Recherche* through the ANR-19-CE39-00201 FP-Locker[15] and ANR-21-CE39-0019 FACADE[16] projects, and was made possible by *Software Heritage*, [17] the great library of source code.

[11] https://stallcatchers.com/
[12] https://phylo.cs.mcgill.ca/
[13] https://www.blend4web.com/apps/space_disaster/space_disaster.html
[14] https://moonrider.xyz/
[15] https://anr.fr/Projet-ANR-19-CE39-0002
[16] https://anr.fr/Project-ANR-21-CE39-0019
[17] https://www.softwareheritage.org/

REFERENCES

[1] 1995. Netscape Navigator 2.0b1. https://web.archive.org/web/20020203083536/http://www25.netscape.com:80/eng/mozilla/2.0/relnotes/windows-2.0b1.html.

[2] 1996. ASCII version of Space Invaders - Reproduced by N. Landsteiner. https://www.masswerk.at/termlib/sample_invaders.html.

[3] 1996. HotJava 1.0 alpha2 (Web Archive). https://web.archive.org/web/19961225173659/http://sunsite.unc.edu:80/pub/sun-info/hotjava/.

[4] 1996. Internet Explorer 3.0 news release. https://news.microsoft.com/1996/08/13/microsoft-launches-microsoft-internet-explorer-3-0-with-exclusive-free-content-offers-from-top-web-sites/. Accessed: 2022-10-30.

[5] 1996. Netscape Navigator 2.0 (Web Archive). https://web.archive.org/web/19970709120829/http://home.netscape.com/comprod/products/navigator/version_2.0/index.html.

[6] 1997. Macromedia Rides the FutureWave. https://www.wired.com/1997/01/macromedia-rides-the-futurewave/. Accessed: 2022-11-06.

[7] 2004. Extending HTML. https://ln.hixie.ch/?start=1089635050. Accessed: 2022-11-07.

[8] 2005. Adobe Buys Macromedia for $3.4 Billion. https://www.nytimes.com/2005/04/19/technology/adobe-buys-macromedia-for-34-billion.html. Accessed: 2022-11-06.

[9] 2005. Web Applications 1.0 - Early working draft. https://whatwg.org/specs/web-apps/2005-09-01/. Accessed: 2022-11-06.

[10] 2009. SuperMeatBoy Development Blog. http://supermeatboy.blogspot.com/2009/04/i-sleep-to-sounds-of-squrils-making.html. Accessed: 2022-11-07.

[11] 2010. Thoughts on Flash (Web Archive). https://web.archive.org/web/20100630153444/https://www.apple.com/hotnews/thoughts-on-flash/. Accessed: 2022-11-03.

[12] 2012. Taking advantage of GPU acceleration in the 2D canvas. https://developer.chrome.com/blog/taking-advantage-of-gpu-acceleration-in-the-2d-canvas/.

[13] 2015. Mozilla - NPAPI Plugins in Firefox. https://blog.mozilla.org/futurereleases/2015/10/08/npapi-plugins-in-firefox/.

[14] 2017. Flash & the Future of Interactive Content. https://blog.adobe.com/en/publish/2017/07/25/adobe-flash-update. Accessed: 2022-11-07.

[15] 2019. Unreal Engine - HTML5 Game Development. https://docs.unrealengine.com/4.27/en-US/SharingAndReleasing/HTML5/. Accessed: 2022-11-06.

[16] 2022. Bluemaxima's Flashpoint. https://bluemaxima.org/flashpoint/. Accessed: 2022-11-05.

[17] 2022. Direct3D 12 Api. https://docs.microsoft.com/en-us/windows/win32/direct3d12/direct3d-12-graphics.

[18] 2022. Khronos Group. https://www.khronos.org/.

[19] 2022. Microsoft First Quarter Earnings Conference Call. https://www.microsoft.com/en-us/Investor/events/FY-2023/earnings-fy-2023-q1.aspx. Accessed: 2022-11-06.

[20] 2022. Mosaic browser. https://www.ncsa.illinois.edu/research/project-highlights/ncsa-mosaic/. Accessed: 2022-10-31.

[21] 2022. Newgrounds. https://www.newgrounds.com/. Accessed: 2022-11-05.

[22] 2022. OECD - Broadband statistics. https://www.oecd.org/sti/broadband/broadband-statistics/.

[23] 2022. Phaser Game Engine. https://phaser.io/.

[24] 2022. Sony Corporation. https://sony.com.

[25] 2022. Transcript - Nvidia's earnings call. https://www.fool.com/earnings/call-transcripts/2022/08/24/nvidia-nvda-q2-2023-earnings-call-transcript/. Accessed: 2022-11-06.

[26] 2022. Unity. https://unity.com.

[27] 2022. Unreal Engine. https://www.unrealengine.com.

[28] 2022. Vulkan API. https://www.vulkan.org/.

[29] 2022. W3C - WebGPU working draft. https://www.w3.org/TR/webgpu/.

[30] 2023. Farmville. https://www.farmville3.com/.

[31] 2023. Neopets. https://www.neopets.com/.

[32] 2023. Stardoll. https://www.stardoll.com/.

[33] Apple. 2022. Metal API. https://developer.apple.com/metal/.

[34] Damjan Buhov, Julian Rauchberger, and Sebastian Schrittwieser. 2018. FLASH: Is the 20th Century Hero Really Gone? Large-Scale Evaluation on Flash Usage & Its Security and Privacy Implications. *J. Wirel. Mob. Networks Ubiquitous Comput. Dependable Appl.* 9, 4 (2018), 26–40.

[35] Wei Cai, Ryan Shea, Chun-Ying Huang, Kuan-Ta Chen, Jiangchuan Liu, Victor CM Leung, and Cheng-Hsin Hsu. 2016. A survey on cloud gaming: Future of computer games. *IEEE Access* 4 (2016), 7605–7620.

[36] Kuan-Ta Chen, Yu-Chun Chang, Po-Han Tseng, Chun-Ying Huang, and Chin-Laung Lei. 2011. Measuring the latency of cloud gaming systems. In *Proceedings of the 19th ACM international conference on Multimedia*. 1269–1272.

[37] Yingyot Chiaravutthi. 2006. Firms' strategies and network externalities: Empirical evidence from the browser war. *The Journal of High Technology Management Research* 17, 1 (2006), 27–42.

[38] Vickie Curtis. 2014. Online citizen science games: opportunities for the biological sciences. *Applied & translational genomics* 3, 4 (2014), 90–94.

[39] Mikhail Fiadotau. 2020. Growing old on Newgrounds: The hopes and quandaries of Flash game preservation. *First Monday* (2020).

[40] Andrew Guldman. 2002. Building Rich Internet Applications with Macromedia Flash MX and ColdFusion MX. *Macromedia white paper* (2002).

[41] Rob Hawkes. 2011. *Foundation HTML5 Canvas: For Games and Entertainment*. Apress.

[42] HP. 2021. Top 50 best-selling video games of all time. https://www.hp.com/us-en/shop/tech-takes/top-50-best-selling-video-games-all-time. Accessed: 2022-11-05.

[43] Jeremy Keith. 2005. A Brief History of JavaScript. *DOM Scripting: Web Design with JavaScript and the Document Object Model* (2005), 3–10.

[44] Michael Lujan, Michael Baum, Dayuan Chen, and Ziliang Zong. 2019. Evaluating the Performance and Energy Efficiency of OpenGL and Vulkan on a Graphics Rendering Server. In *2019 International Conference on Computing, Networking and Communications (ICNC)*. IEEE, 777–781.

[45] Meta. 2021. Our vision for the metaverse. https://tech.fb.com/ar-vr/2021/10/connect-2021-our-vision-for-the-metaverse/. Accessed: 2022-11-06.

[46] Mozilla. 2022. WebXR Device API. https://developer.mozilla.org/en-US/docs/Web/API/WebXR_Device_API. Accessed: 2022-11-06.

[47] Oracle. 2017. JDK 9 Release Notes. https://www.oracle.com/java/technologies/javase/9-deprecated-features.html. Accessed: 2022-11-04.

[48] Michael D Patrick. 2019. A Brief History of Digital Communications. In *Social Media for Medical Professionals*. Springer, 23–47.

[49] Anastasia Salter and John Murray. 2014. *Flash: Building the interactive web*. MIT Press.

[50] Statista. 2020. Gaming: The Most Lucrative Entertainment Industry By Far. https://www.thc-pod.com/episode/the-gaming-industry-is-now-bigger-than-movies-and-music-combined. Accessed: 2022-11-08.

[51] Simon Stobart and Mike Vassileiou. 2004. Introduction to PHP. In *PHP and MySQL Manual*. Springer, 7–11.

[52] W3C. 1996. HTML 4.0 press release. https://www.w3.org/Press/HTML4-REC-fact.html. Accessed: 2022-11-06.

[53] Yuntao Wang, Zhou Su, Ning Zhang, Rui Xing, Dongxiao Liu, Tom H Luan, and Xuemin Shen. 2022. A survey on metaverse: Fundamentals, security, and privacy. *IEEE Communications Surveys & Tutorials* (2022).

[54] Jennifer R Whitson and Claire Dormann. 2011. Social gaming for change: Facebook unleashed. *First Monday* (2011).

[55] Allen Wirfs-Brock and Brendan Eich. 2020. JavaScript: The First 20 Years. *Proc. ACM Program. Lang.* 4, HOPL, Article 77 (jun 2020), 189 pages. https://doi.org/10.1145/3386327