International Journal of Emerging Trends in Computer Science and Information Technology
ISSN: 3050-9246 | https://doi.org/10.63282/3050-9246.IJETCSIT-V1I3P107
Eureka Vision Publication | Volume 1, Issue 3, 56-67, 2020

Original Article

Browser-Based Parametric Modeling: Bridging Web Technologies with CAD Kernels

Kiran Kumar Pappula
Independent Researcher, USA

Abstract - Parametric modeling has turned out to be a mandatory maxim in computer-aided design (CAD) that allows designers and engineers to build geometrically limited models that can be shaped or changed without problems by manipulating parameters. However, the conventional CAD systems are highly dependent on hardware-demanding software installation along with the platform-specific systems. This paper discusses the possibility of parametric modelling to work through browsers, incorporating CAD kernels with web technologies, specifically WebAssembly and JavaScript. WebAssembly makes web-capable execution fast (near-native performance), whereas JavaScript is the glue providing user interaction, presentation of a UI, and the ability to dynamically modify parameters of a design. This study explores the realities, challenges and problems associated with integrating CAD kernels like OpenCascade and Parasolid with web front-ends and the implications of dealing with performance bottlenecks, problems of persistent storage, responsiveness of user interfaces and management of data in the cloud. The idea is to assess the possibility of the existence of an alternative to conventional desktop-based modeling environments in the form of a web-based platform. One of the fundamental issues will be interactive performance, as the computational intensity is characteristic of the parametric operations. We look at WebAssembly optimizations and multi-threading (when possible), and compare the complexities of models with responsiveness trade-offs. The solutions we investigate in persistent model storage are local browser storage (IndexedDB) and remote storage through REST APIs and GraphQL. They provide details of a complete case study where a sophisticated parametric model is formulated, edited and saved entirely on the browser. Our results are compared with those of other existing traditional CAD platforms in terms of metrics such as load time, interaction delay, model regeneration rate, and user satisfaction, as measured through surveys. Our results show that parametric modeling with a browser can not only be done, but it can also be sufficient to cover a broad scope of design opportunities, especially in cases of collaborations, learning and lightweight design. Nevertheless, it has a problem in scaling up with very complex assemblies and in advanced integration of simulation. This paper lays the groundwork for future research into high-performance web-based CAD systems and their integration into broader digital engineering workflows.

Keywords - Browser-Based CAD, Parametric Modeling, WebAssembly, JavaScript, CAD Kernel, IndexedDB, WebGL, OpenCascade, Performance Optimization, Cloud CAD.

1. Introduction

The fact that high-performance web technologies found their way into the mainstream at such a rapid pace has, in essence, revolutionized the manner by which modern applications are planned, programmed, and provided. Typically, Computer-Aided Design (CAD) programs have been limited to the performance of high-end desktop systems because they rely on resource-intensive calculations and the close coupling of libraries to the operating systems. [1-4] Such conventional CAD systems usually require a significant amount of processing hardware, dedicated video cards and platform-specific installations. They are therefore not as readily available to casual users or even students or professionals who use low-end or handheld systems. Moreover, the software structure is seldom compatible; their updates and hardware limitations can be a problem, and they are not commonly portable. To address these issues, the software development community has explored web-based solutions, including technologies such as WebAssembly (WASM), WebGL, and Service Workers. These inventions enable the execution of even complex applications directly in the browser, making it possible to retain speed and functionality. There are several benefits associated with Web-based CAD solutions: they are not tied to any specific platform, do not require installation, allow for real-time updates, and can be run on almost any device with an internet connection. Such a transition not only represents a democratisation process of access to professional-grade design products but also a transition to more collaborative, lightweight, and scalable engineering processes. With surfacing browser capabilities ever closer to native environments, the exploration of in-browser CAD systems is not only insightful but also a viable avenue of research to pursue in the academic field, as well as a niche to explore in the real world.

1.1. Importance of Browser-Based Modeling

With engineering, design, and teaching moving toward digital and collaborative surfaces, browser-based modelling has emerged as a revolutionary technology. In contrast to traditional CAD tools, which require installation and a specific hardware setup, browser-based modelling tools are compiled to work strictly within web browsers, providing several benefits that transform the user's interaction with design technology.

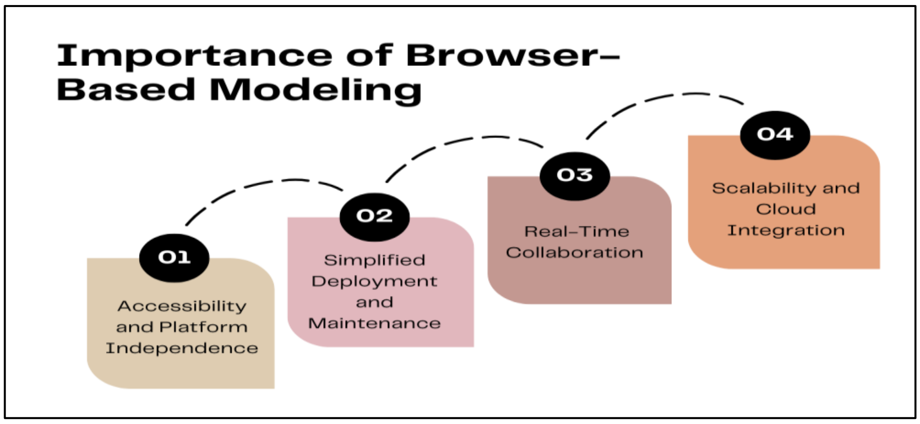
Importance of Browser-Based Modeling
01
Accessibility and Platform Independence
02
Simplified Deployment and Maintenance
03
Real-Time Collaboration
04
Scalability and Cloud Integration

Figure 1. Importance of Browser-Based Modeling

- **Accessibility and Platform Independence:** Arguably, the greatest advantage of browser-based modelling is that it is platform-independent. It features design tools that can be utilised by users on virtually any device, including desktop computers, laptops, tablets, and smartphones, with compatibility across various operating systems. It democratises CAD software, particularly for students, mobile workers, and users in developing countries with limited access to high-end workstations.
- **Simplified Deployment and Maintenance:** Since it operates in a browser, the application does not require local installation or any updates. All the users of the application are charged with the same version of the application, and problems with compatibility are minimized, resulting in performance that does not vary. This is advantageous in instructional and business settings where deploying software on multiple machines can be time-intensive and expensive.
- **Real-Time Collaboration:** Web technologies, built into browser-based platforms, inherently enable real-time collaboration. Several users can comment on models, edit and see a model at a time in different locations. This feature can smooth processes within a professional design team and help to realize interactive learning within an academic environment, supporting a more interrelated and responsive design scheme.
- **Scalability and Cloud Integration:** Browser-based modelling enables scalable storage, version control, and remote access to design files when combined with integration into cloud services. This allows users to begin a project on one device and continue it on another device, picking up where they left off. That flexibility serves contemporary and adaptive design processes, providing constant access to work irrespective of geographic location.

1.2. Problem Statement

CAD has long been at the centre of the engineering and product development process, with the main focus in this area being the use of traditional CAD software. [5,6] Nevertheless, they are usually limited by the fact that high-performance hardware and compatibility of the operating system and complicated installation processes are required. Consequently, they are not easily accessible, especially to students, individual designers, and those designing in a resource-constrained setting. Also, the vast majority of desktop CAD solutions are not natively optimized to work with real-time collaboration, cloud connection, and device mobility, which have grown to be more relevant in the contemporary distributed and agile work environment. Although certain services in the modern solutions have integrated cloud-based services, these solutions are still highly dominated by server-side-based computations and are therefore highly restricted by the consistent network connections and centralized infrastructure.

This limits their functionality in the absence of a network or a low-bandwidth application. Conversely, the web technologies are now at a much higher level, and one can write even advanced applications and run them in the browser (with the use of WebAssembly (WASM), WebGL, and JavaScript frameworks). Although this has changed, complete, functional, browser-based CAD systems are uncommon and often have a limited scope, functionality, or speed. The main hurdles that we have to overcome are the ability to process geometry efficiently in-browser, ensuring that latency is low throughout the modeling process, and effective models to handle local and remote storage, as well as the ability to enable a user experience that is similar to a traditional desktop tool. Additionally, a technical obstacle will exist in attempting to integrate CAD kernels with browser-based frontends, both due to the performance limitations this approach would entail and the technical difficulty of porting C++ codebases to WebAssembly. The main issue that is illuminated in this paper is that a good, real-time, parametric, portable, browser-based CAD solution with a nice, accessible interface does not exist, capable of providing real-time operation, persistent storage, and not requiring the setup of backend servers. A solution to this would allow a new type of lightweight, scalable, collaborative design tools that are applicable in education, quick prototyping and small-scale manufacturing and would be a remarkable change in the delivery and utilization of CAD technologies.

2. Literature Survey

2.1. Evolution of Web-Based Applications

Applications have been using the internet, and this has led to a major shift in web-based applications since the beginning of the web. [7-10] The web application has evolved from static HTML pages with little interactivity into dynamic, feature-rich applications that are on par with native grid-powered annotations of desktop software. This transformation has been brought about by enhancements in both hardware and web technologies. Other important inventions, such as JavaScript frameworks (e.g., React, Angular, Vue.js), enabled the establishment of Single Page Applications (SPAs) that provide smooth user experiences without requiring a page reload to display content. Moreover, with WebGL, one can now perform hardware-accelerated 3D graphics in the browser, and WebAssembly (Wasm) allows launching code written in C++ or Rust at near-native speeds because programs are compiled into a format that can run efficiently in the browser. Service Workers enhance offline functionality and enable background job executions, such as data synchronisation or caching.

2.2. CAD Kernels and Open Standards

Computer-Aided Design (CAD) Kernels CAD kernels are the core computational engine that does operations such as boolean modeling, surface generation and topological analysis that are carried out by any given modeling and simulation tool. There were traditionally powerful proprietary kernels, such as Parasolid or ACIS, that dominated the market, making them less accessible due to various licensing and platform restrictions. However, the advent of open-source CAD kernels, particularly OpenCascade, has made advanced modelling functions more democratically accessible. The tools that OpenCascade manipulates are intricate operations, such as B-Rep modelling, NURBS surface treatment, and error correction. Since Emscripten, a toolchain that transpiles C++ code to WebAssembly, has been developed, such kernels can now be compiled and run in a Web browser. It is a balancing act that keeps performance and functionality without requiring local installations or platform dependencies. In addition, the adoption of open specifications such as STEP, IGES, and glTF will provide interoperability between various CAD systems and promote the preservation of data over the long term.

2.3. Previous Attempts

There have been several efforts to bring CAD capabilities onto the web, which can provide useful ideas and prototypes. Cloud-based professional modeling is made possible in the case of commercial offerings such as Onshape and Autodesk Fusion 360 (web version). These systems perform computationally intensive tasks by using server-side processing to make the UI responsive in the browser. More specifically, Onshape provides a full-featured modeling environment with live collaboration, based on a cloud platform and utilizing cloud capabilities of elasticity and resource control. Likewise, open-source projects such as the WebAssembly port of FreeCAD demonstrate that one can even execute full-functional CAD programs on the client-side. However, the level of interaction and capability is low compared to their desktop counterparts. However, these solutions have limitations, particularly in terms of offline capability. The majority has an overdependence on continuous internet connections to perform computations, as well as storage, which is a limiting factor in areas with limited bandwidth or secure environments. All these constraints raise the necessity of hybrid models capable of striking a balance between the performance of server-based processing and the independence of client-based processing.

2.4. Performance Benchmarks

Performance is another major consideration in the evaluation of web-based CAD systems, particularly against native systems. Performance benchmarks between native code and WebAssembly-built native code reveal a performance delta, particularly in more complex geometric operations and various rendering tasks. Although WebAssembly significantly narrows the performance gap with traditional JavaScript, it does not meet native performance requirements in every case, particularly for multi-threaded and memory-intensive applications. It was also found that the graphics pipeline and memory management of the browser can limit the rendering of large 3D models with WebGL. However, some of these disadvantages can be alleviated with optimizations like lazy loading, offloading of computation and utilization of web workers. Practical experience, such as in the Wasm builds of FreeCAD, experimental modules in Onshape, and others, suggests that though the accessibility and deployability of browser-based systems are excellent, they are very much in need of careful optimization to satisfy industrial purposes.

2.5. Persistent Storage Solutions

The skills in storing and handling information indefinitely on a browser platform are a virtual prerequisite for constructing operational and stable CAD applications. Browsers offer several native storage options, each differing in terms of capacity, performance, and complexity. IndexedDB is a level API that is suited to storing large volumes of structured data via asynchronous transactions and/or key-value pairs, structured data, such as 3D models and the various metadata associated with them; it is the recommended web API to use for offline web applications and supports large data sets, as well as key-value pairs. SessionStorage and localStorage have a simple API, but are limited in size and unsuitable for long-term data storage. To achieve stronger solutions, access to cloud-hosted services through APIs, such as AWS S3, Google Drive, or Firebase, enables scalable storage of data, version control, and collaboration. These cloud integrations, however, require a stable internet connection and raise concerns about security and privacy. Local and cloud storage are incontinently syncing as a significant problem, with the need to intelligently merge changes in collaborative settings being a particular challenge. Version control , e.g. Git-style in CAD data, is already under research and development, where a radical effect change can be felt in the way CAD projects are online managed.

3. Methodology

3.1. System Architecture

The system architecture of a web-based CAD application will contain three primary modules: the User Interface, the CAD Kernel (WASM Module), and the Local/Cloud Storage. [11-13] These elements collaborate to deliver a browser-based CAD experience that is interactive, high-performance and flexible.

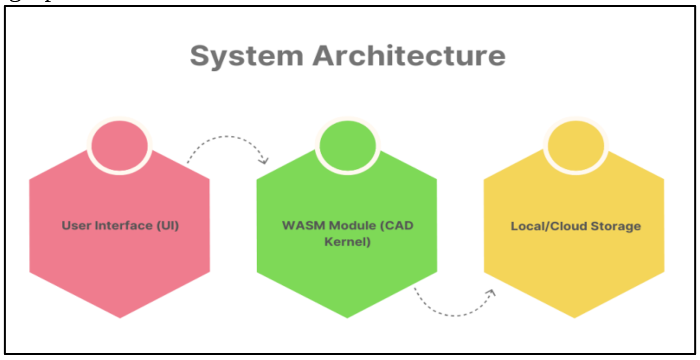

Figure 2. System Architecture

- **User Interface (UI):** The User Interface is created using the latest JavaScript frameworks, such as React or Vue.js, to create a responsive and user-friendly space where users can work with 3D models. This is an interface layer which addresses user inputs such as mouse gestures, keyboard commands and UI controls and provides visual output in WebGL in real-time 3D graphical visualization. Application workflows also exist to apply properties of an object or start modeling tasks, etc., that are managed by the UI.
- **WASM Module (CAD Kernel):** The computational heart of the system is the WASM Module, which is compiled from a native CAD kernel, such as OpenCascade, using Emscripten. It performs geometric and topological procedures, such as extrusion, filleting operations, and Boolean operations, all within the browser setting. WebAssembly enables the system to perform nearly as quickly as native code, while maintaining the same security and portability as the web. Communication between the JavaScript-based UI and the WASM module is via a well-defined JavaScript-Wasm API, which allows the UI to delegate heavy modelling work to this high-performance kernel.
- **Local/Cloud Storage:** Local/Cloud Storage is the layer that deals with the persistence and synchronization of data. On the client side, utilise technology to access offline data and quickly retrieve model data, such as IndexedDB and File System Access API. To store data collaboratively or on a long-term basis, cloud solutions such as Firebase, AWS S3, or custom RESTful APIs can be integrated to support versioning, backups, and multiple-user access. The system is designed to access both forms of storage and utilize them as both read and write, using APIs, which enables deployment flexibility, whether as an independent offline application or incorporating the system into a more comprehensive cloud-based system.

3.2. Technology Stack

The proposed system also utilises a contemporary modular technology stack that will provide good performance, cross-platform support, and development capabilities. Every layer, including the front end, middle tiers, and storage, is chosen with a view to optimizing the user experience, the effectiveness of computation, and the management of data.

- **Frontend:** The frontend is built with the bare minimum of modern JavaScript frameworks (React.js or Vue.js), providing an easy, dynamic, responsive, and component-based user interface. It supports WebGL, which enables the handling of 3D graphics directly within the browser, providing users with a real-time interaction experience for CAD models. Here, the user's inputs are processed, including mouse events, tool selections, and parameter settings, and communicated to the backend logic or the CAD kernel. It is also possible to simplify 3D visualization using UI libraries such as Three.js and make the visual design and layout responsive with CSS frameworks, e.g., Tailwind CSS or Bootstrap.
- **Backend:** According to this architecture, backend may be optional or minimal in terms of deployment mode. In the case of cloud-connected cloud, a sandboxed version alignment is realised using a lightweight backend that can utilise Python (Flask/FastAPI) or Node.js to implement user authentication, file uploads, sessions, and version control. The frontend and cloud services require structured communication with one another; RESTful or GraphQL APIs facilitate

this. Particularly in completely client-side deployments, server involvement is either reduced or nonexistent, and the logic and data are either locally cached or cached within progressive web app (PWA) functionality. Third-party APIs may also be used to integrate more advanced functionality, such as cloud storage or rendering facilities, into backend services.

- **CAD Kernel:** The heart of geometric calculation is driven by an open-source CAD kernel, such as OpenCascade, which is compiled behind the scenes to WebAssembly (WASM) using tools like Emscripten. This will bring the capabilities of native C++-based geometry libraries into the browser so that it becomes possible to perform complex modeling tasks such as boolean operations and NURBS surface manipulation, as well as B-Rep modeling. It is a sandboxed and secure WASM module that offers excellent performance, yet remains cross-platform. The interface between the kernel and the frontend is a bridge written in JavaScript, which allows access to the functionality of the former without disturbing the code or hindering work with user input and the interface.
- **Storage:** The system can store information on both the client-side and the cloud platform to ensure persistence and effective data management. IndexedDB and File System Access API provide robust local storage that can be used offline, enabling users to store and load models directly on their devices. The remote storage of data and assets that supports features like automatic backups, file synchronization, and multi-user collaboration assumes the role of services provisioning via cloud functionality such as Firebase, Amazon S3 or other custom backend APIs in cloud-enabled contexts. To be serializable and achieve compatibility, data formats such as STEP, STL, and JSON are available. The storage strategies are efficient, providing both performance and reliability in local and distributed environments.

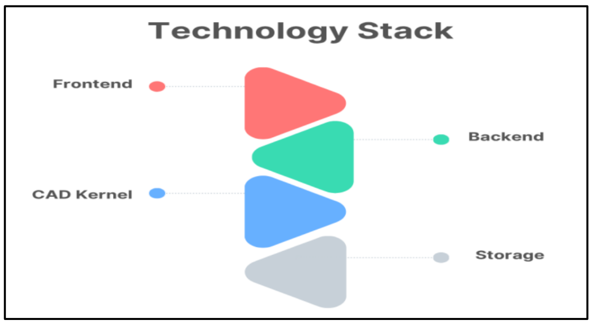

Figure 3. Technology Stack

3.3. Data Flow

Within the system, data flow is based on a logical, event-driven pipeline that can be determined by user input, which leads to real-time 3D graphic rendering. [14-17] Such an efficient process is responsive and efficient and can provide complex CAD operations within the browser entirely.

- **Parameter Input:** The data flow commences with the user performing an input action on the user interface, such as typing in dimensions, clicking on tools, or reshaping any given geometry. The inputs are usually provided through form inputs, sliders, or direct manipulation in the 3D scene (such as dragging points or edges). Each input corresponds to a modification of the model's parameters and is regarded as an event that must be passed through the model. These values are organized in a data object, most of the time in JSON, to be consistent and compatible with the following modules.
- **JavaScript Triggers Update:** After data is received through the input, the frontend JavaScript-based logic acts on the edit, and the received changes are left ready to be integrated into the CAD kernel. This layer serves as a controller, transforming the UI into well-defined functional calls. It checks reflective input and updates the internal state, and also calls the same API methods that communicate with the WebAssembly module. This step can involve extended logic, such as handling constraints, unit manipulation, or even triggering dependent calculations (e.g., resolving sketch constraints when a dimension is edited).
- **WASM Recomputes Geometry;** Parameters are updated and then supplied to the WebAssembly (WASM) module, where the compiled CAD kernel lives, e.g. OpenCascade. In this case, the kernel performs the necessary geometric calculations, including recreating a 3-dimensional solid using new measurements or carrying out a Boolean operation. This is fully browser-based processing, so it can generate a response almost in real-time without requiring any other server resources. The ensuing geometry is then transformed to a mesh or an edge/face representation, which the rendering engine can understand.
- **3D Rendered via WebGL:** The mesh or model that is returned to the rendering engine after geometry is recalculated is then transmitted back to the rendering engine, which is normally implemented in WebGL. It is a low-level graphics API that connects directly to the GPU to render the model on a canvas element in the browser. It provides real-time updates, enabling users to observe the results of their parameter changes in real time. Additionally, it includes interactive features such as zooming, rotating, and panning.

More sophisticated graphics facilities, such as shading, backlighting, shadowing, and wireframe superimposition, may be included to provide the user with a better understanding and accuracy in modeling.

3.4. Optimization Techniques

In order to maintain a responsive and efficient web-based CAD system, there are some strategies that are used in optimization. The approaches focus on both performance and resource consumption, improving user performance across different devices and network loads.

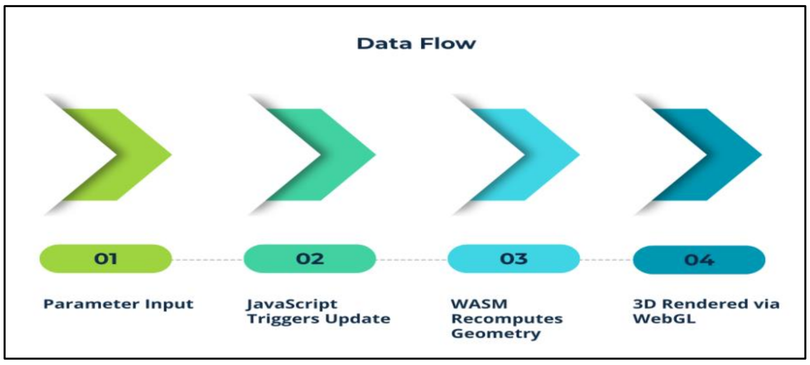

Figure 4. Data Flow

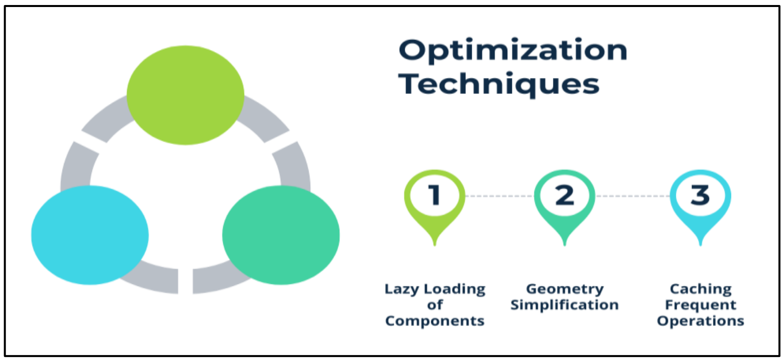

Figure 5. Optimization Techniques

- **Lazy Loading of Components:** Lazy loading is a method of loading application components and resources as needed, rather than loading them directly during startup. In a web CAD application, this can be interpreted as delaying the downloading of any heavy modules, i.e. that particular modeling tool, UI panel, or geometry library until the user himself explicitly triggers an activity that needs it. This saves a significant amount of time on startup and reduces memory usage, making the application boot faster and with an increased sense of responsiveness.
- **Geometry Simplification:** It may be difficult to render high-polygon-count CAD models with a good quality of display, resulting in application slowdown and sometimes even the inability to render some models on lower-end machines. Geometry simplification is a solution to this, and involves the removal of faces or edges in a rendered model that have little impact on visual quality. This can be applied dynamically as one navigates (e.g., zooming out or rotating the model) or preprocessed before rendering. Performance optimization is considered; techniques such as decimation, mesh merging, and level-of-detail (LOD) rendering provide high performance with no interference in use and precision of the modeling interface.
- **Caching Frequent Operations:** To minimize unnecessary recalculation of intermediate results, and to reduce response times, the system employs caching of the operations that are used often and their intermediate result. Examples of such use cases include a user switching between similar parameter values of the same geometry or switching between views in which the system can reuse computed geometry or rendering data. This caching may be performed at the JavaScript layer and at the WASM module level, and may store results in memory or in the browser's storage facilities, such as IndexedDB. Smart invalidation schemes help maintain the balance between accuracy and efficiency regarding up-to-date cache contents by only refreshing those portions that need an update.

3.5. Evaluation Metrics

To measure the effectiveness and efficiency of the web-based CAD system, a set of quantitative evaluation parameters has been identified. [18-20] Such measures can be used to gauge performance, responsiveness and usability in real-world conditions, such that practical expectations of the system are not compromised.

- **Load Time(ms):** Load time is the amount of time the application takes in a browser after it has been launched. This entails acquiring all the necessary sources (e.g., HTML, JavaScript, WASM modules, assets), starting the rendering engine, and displaying the initial UI. Faster load times improve the user experience, particularly in time-sensitive settings. Code splitting, lazy loading, and compressed assets delivery are the optimizations that are normally applied to reduce this indicator.
- **Regeneration Time (ms):** Regeneration time is the time required by the system to recalculate the geometry when a user changes the parameters or when the user extrudes, fillets, or uses a Boolean operation. As a crucial parameter of CAD systems, this metric has a direct influence on the user's perception of the responsiveness of a CAD system. It is the time consumed by the WASM CAD kernel to receive the input, create the new geometry, and then return it to the rendering engine.
- **Storage Latency (ms):** Storage latency - The cost of writing to/reading from a persistent store (local storage, i.e., IndexedDB or cloud-based, e.g. Firebase, AWS S3). Such measures matter to workflows that involve frequent autosaving or loading large model files. Effective serialization, compression, and asynchronous I/O activity minimize its latencies and guarantee the absence of problems related to the ease of use at save/ load operations.

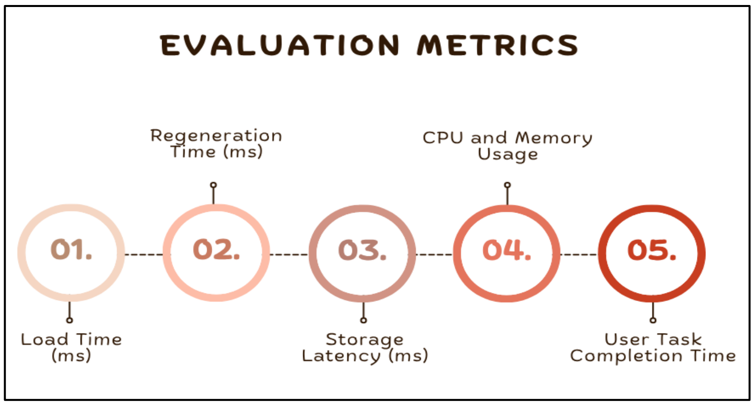

Figure 6. Evaluation Metrics

- **CPU and Memory Utilisation:** The level of CPU and memory usage denotes the efficiency of computation and memory usage in the system. These are essential insofar as they cater to a variety of hardware, especially mobile and low-end devices. A large consumption of CPU resources may cause the UI to be slow or power consumption to be high, whereas an excessive amount of memory may result in crashing or slowing down. It is always monitored and optimized, keeping track at all levels, including JavaScript, WebAssembly, and rendering, to guarantee consistent performance and scale.
- **User Task Completion Time:** A user-centric metric is a measure of how long it takes an average user to accomplish certain aspects of modeling, generating a part, making use of a turning operation or exporting a file. It provides an insight into how easily the system is used and understood. The less time a task takes, the easier the work process goes and the more professionally the UI was made; the more time, the less professionally the UI is made and/or a section of the application should be tuned to perform better.

4. Case Study / Evaluation

4.1. Experimental Setup

An experimental study was conducted to compare the proposed browser-based CAD system in terms of its capabilities and performance, assessing the effectiveness of using such a system in creating parametric models. This model has been selected because it has relevance to the above model in a real engineering context, and it is also of reasonable complexity, as it reflects both geometric constraints and multiple parts with parameter variations. Several rigid links were connected, and the rotational joints constituted the mechanical interface that permitted free activity, manipulating direction and angles. Important parameters like the length of links, angles of joints and positions of pivot points were revealed in the user interface in order to allow the user to update the model and see in real time the behaviour of the model. Users could type in numerical values or drag sliders to change parameters, triggering JavaScript events that updated the parameters of the WebAssembly-driven CAD kernel. It did the recomputation of the geometry using these inputs and rendered the resultant 3D model using WebGL in the browser. This model allowed for editing components individually, as well as making assembly-level transformations, and challenged the system to perform parametric mutual dependence requirements as well as space-based updates.

The traffic between the frontend and backend was captured to evaluate the latencies of computation and rendering time. Additionally, the permanent storage solution was tested by storing the model on the local computer using IndexedDB and then uploading it to the cloud-based storage system to assess collaboration. Records of performance have been taken, including regeneration time, memory load, and the time taken by the user to complete the task, where repeated adjustments of parameters and manipulation of models were used. The test aimed to simulate commonly occurring usages, such as design iteration, visual inspection, and parameter-driven redesign, and thus provide an accurate playing ground for the usability of the system in practical use. Generally, the experimental environment provided a comprehensive testbed through which the functional characteristics and performance nature of the web-based CAD platform could be analysed within the context of interactive engineering design practice.

4.2. Comparative Analysis

To assess the usefulness and effectiveness of the browser-based CAD system, we compared it with the free and popular desktop CAD program FreeCAD. The comparison aimed at offering crucially important performance indicator results on the normal modeling tasks, and it shed light on trade-offs in accessibility and computational power.

Table 1. Comparative Analysis

Operation	Browser-Based	Desktop (FreeCAD)
Load Time (ms)	320	120
Param Change (ms)	85	45
Save Time (ms)	45	30
Memory (MB)	120	200

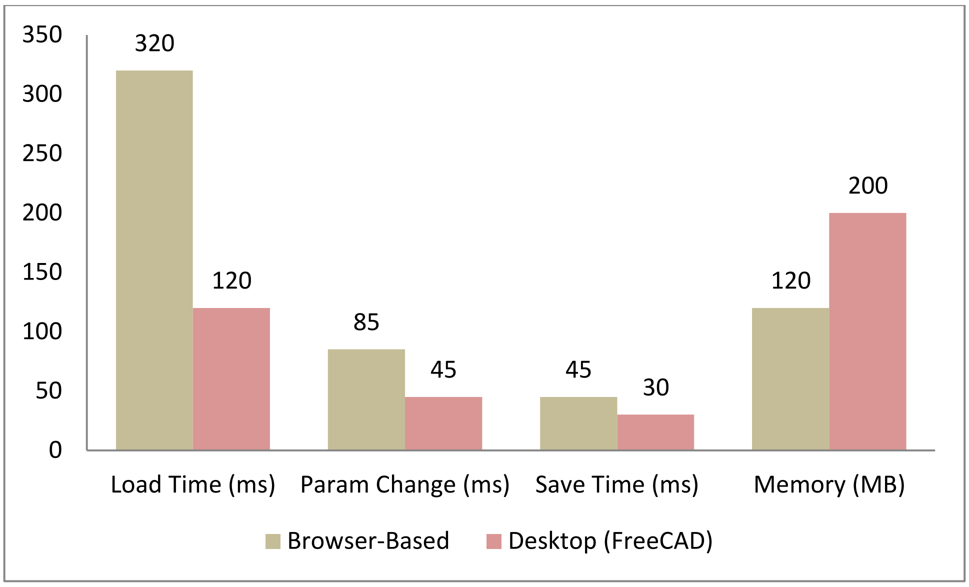

Figure 7. Graph representing Comparative Analysis

- **Load Time:** The load time refers to the time it takes for an application to start up and load its initial user interface and model data. The web-based system took longer to load, with a value of 320 milliseconds, whereas FreeCAD had a shorter load time, equal to 120 milliseconds. The extra overhead primarily results from WebAssembly modules and the time required for JavaScript resource fetching and loading. Instead, local execution and precompiled binaries make the desktop application startup faster.
- **Parameter Change Time:** This is a measure of the duration required to update the model geometry when a user-modified design parameter is changed. An average of 85 milliseconds was required in the browser-based system to make a parameter change, whereas FreeCAD required 45 milliseconds on average. The desktop environment with WebAssembly remains a step ahead in terms of computation speed and resource allocation, particularly in the display of geometry regeneration, which is handled by the native C++ component.
- **Save Time:** Save time indicates the amount of time it takes to write model data to persistent storage. When the browser-based system saved its data in IndexedDB, the time it took was 45 milliseconds. In comparison, FreeCAD was able to save 30 milliseconds to the local file system. Despite the slight disparity, direct access to the file system is available on desktop systems, and browser storage utilises asynchronous data operations with some abstraction present.

- **Memory Usage:** Interestingly, the browser-based system required 120 MB of memory when functioning, which is smaller in comparison to FreeCAD's 200 MB. One of the reasons is that the web is a modular environment, and a large number of features are only loaded when needed (lazy loading). FreeCAD is a full-featured desktop application that loads more libraries and background tasks at startup, which further increases memory usage.

4.3. User Survey

To evaluate the usability and general user experience within the browser-based CAD system, a user survey was conducted with a group of individuals who had varying levels of familiarity with 3D modelling tools, including engineering students, CAD professionals, and hobbyists who enjoyed working with such applications. The four key evaluation criteria considered in the survey were ease of rupture, visual appeal, responsiveness, and portability. Concerning usability, a majority of users commented that the interface was usable to a point, especially acknowledging the simplified layout and the logical organization of tools. The presence of features such as drag-and-drop, fields for entering parameters, and interactive tooltips helped achieve an easy onboarding process, even for novice users. In the visual clarity category, the participants commended the quality of the 3D rendering done using WebGL, stating that the overall shading, sharpness of the lines, and camera controls gave them a professional presentation effect. Nonetheless, some users recommended enhanced edge highlighting as well as real-time responses upon choosing model features. Another significant aspect was responsiveness, and users commented that the system was responsive to any changes in input, particularly when creating parameter-based models.

Although the browser-based system lagged behind desktop models slightly in terms of geometry regeneration, the lag was easily perceived as insignificant by users in real-life applications. Finally, portability was mentioned as one of the main benefits. Users can connect to the system simultaneously on various devices, such as laptops, tablets, and smartphones, without installing any software. This ensured that the system could be easily edited with speed, facilitating collaboration over long distances or in education. The survey, in general, revealed that the impressions about the system were overwhelmingly positive, with a majority of users pointing to the system's flexibility and modernity. Future ideas to be implemented included the incorporation of undo/redo history, multi-part assemblies, and improved touch-screen support. The survey findings confirm that not only is the system technically functional, but it is also accepted by the intended users of the products in the market.

5. Results and Discussion

5.1. Performance Gains

An important performance enhancement in the implementation of browser-based CAD systems came with the inclusion of WebAssembly (WASM), which offloads computationally demanding sections of the application that were formerly covered by pure JavaScript. WebAssembly is a low-level binary instruction set compiled to first-order code to execute with close performance to a native metal program inside the browser, which provides significant performance benefits in applications where geometric operations, model refresh, or parsing are involved. The WebAssembly benchmarking performed similarly to JavaScript in the by-reference benchmarking that performs the same operations, but 2x to 5x faster on the operations using the Boolean model optimization, parameteric updates, and mesh generation. Much of this performance gain is attributed to the good memory model, type safety and compilation of WASM out of optimized C++ (here, the OpenCascade CAD-kernel). JavaScript is powerful and simpler to code, but it is interpreted or just-in-time compiled, which creates an efficiency bottleneck, especially in loops, recursive geometry algorithms, and floating-point arithmetic prevalent in the CAD workflow.

Offloading complex geometry computations to a separate thread (using Web Workers) also occurred due to the switch to WebAssembly, allowing even heavy modelling operations not to stall the UI. As a specific example, it took approximately 85 milliseconds in WASM for one dimension of a component to change and regenerate a model; however, this may take even longer than 200 milliseconds with a JavaScript-only implementation. This decrease in latency was observable to users when controlling parameters in real-time and was a significant addition to easier parameter manipulation. We also have better control of memory management in WASM, which is more predictable and does not have the garbage collection pauses that applications written in JavaScript usually have. The net effect of implementing WebAssembly was that the raw computation speed was improved, and the system's scale, responsiveness, and consistency were also enhanced, making the CAD platform based on the browser a viable alternative to desktop applications in various use cases.

5.2. Usability

Preference was found to be very much for a browser-based CAD system as a result of tests conducted on usability and user feedback on their preferences. In contrast to the old desktop solution, which required minutes of installation and was dependent on the system, whose updates needed to be made periodically, the browser-based solution provided immediate access to the same URL on any modern web browser. This client-side method, which is lightweight and does not require installation, dramatically reduces the barrier to entry, particularly among students, hobbyists, and professionals using shared or low-performance devices. The ability to use it on various platforms, including Windows, macOS, Linux, and even tablets, without compatibility problems was also noted by the participants as a convenience of using the tool. This portability meant it was perfect for ad-hoc modeling, education and distant collaboration. UI design also significantly contributed to the favourable responses in usability. Users liked the simple, sparse design, emphasizing command only on the needed tools of modeling and eliminating clutter common in desktop, fully featured applications. The interface conformed to more contemporary web design, featuring collapsible side panels, tooltips, and real-time previews of the model as it was being edited. The parameters were entered via interactive controls, such as sliders, and feedback was provided in real-time via WebGL, allowing the user to see exactly what happens when they alter the parameters.

This is why even a person new to CAD found it to behave intuitively. The user feedback regarding responsiveness to the browser system's movement, including rotating models, altering dimensions, or progress, is also saved. As a result, little to no interruption or slowdown was reported during testing. Some users mentioned that the system is so straightforward that people feel more in control of the entire process, as it does not allow too many complex settings. On the whole, the usability results are more than suggestive of the hypothesis that simplicity and ease of accessibility do not need to be sacrificed at the cost of functionality. The browser interface was favoured over the old-style desk solutions, which were mostly usable by many applications, mainly for light to medium CAD modelling tasks.

5.3. Limitations

Although the browser-based CAD system offers easy access and ease of use, there are several pertinent limitations that need to be taken into account. Among the major ones, there are its short offline capabilities. Although technologies such as Service Workers and IndexedDB enable at least part of the operations to be performed offline, e.g., opening recent models or making simple edits, a substantial part of the functionality, including cloud sync, collaboration, and extraction of remote information, remains non-functional without internet access. This reduces the applicability of the system for users operating in remote areas or those in places with a limited or patchy connectivity infrastructure, where desktop applications still show a distinct advantage. The second limitation is the memory and processing ability of the browser, whereby when dealing with large assemblies or very detailed models, the browser's limitations would be felt. In contrast to desktop applications, which are able to utilize all the system resources, browsers have a limit to the amount of memory and execution time so as to maintain the stability of the system.

Consequently, an attempt to load or even manipulate a complex assembly that consists of thousands of components or high-resolution meshes results in a significant drop in performance. Trying to go beyond browser memory limits can cause the browser to crash or become unresponsive, limiting the system's use to less intensive applications within the realm of industrial-scale CAD. Additionally, the system does not currently support higher-quality simulation and analysis tools, which are often found in more comprehensive desktop applications like FreeCAD or Fusion 360. Capabilities such as finite element analysis (FEA), kinematic simulations, and thermal or structural stress-related tests are not offered at all or, at best, through an elementary plug-in or vendor API. These are critical tools for engineers involved in product testing and real-world testing. Without them, the system cannot perform entire engineering tasks. The system is well-equipped in parametric design and visualisation; the listed weaknesses open the possibility of targeted development in terms of offline capabilities, resource management, and the use of advanced simulation functionality as a means to bridge the gap with desktop CAD systems.

5.4. Broader Applicability

The browser-based CAD system proves to have substantially greater suitability in various fields, including education, collaborative design, and manufacturing. The platform can be used to educate subjects about the basics of CAD without the burden or equipment needs of the desktop tools, making it an excellent educational tool in the field of education. The lightweight, device-agnostic architecture means that students do not need high-performance machines or software installations, since they can access the modeling tools using laptops, Chromebooks, or even tablets. The bendiness aids distance learning and reduces the entry-level barrier to newbies, hence it is very significant in school classes (K12), vocational and university introductory design classes. Under collaborative design, the system provides real-time co-editing when its cloud backend is incorporated. Several users can simultaneously view and modify the same model, allowing them to collaborate with team members who are not in the same location. Design communication and the speed of the iteration cycle are enhanced through features such as live updates of parameter values, shared view state, and comment threading.

These features are particularly useful when working with a distributed engineering team, freelance designers in contact with clients, or on a group project in academia. The fact that the sessions are browser-based reduces the likelihood that a participant cannot join due to software conflicts or lengthy setups, which adds further to productivity. To connect parametric models to the stakeholders of a manufacturing application, the system can be implemented as a product configuration portal, where stakeholders operate with parametric models using a browser. In particular, this helps when customising products at scale, as end-users or sales representatives can simply change the size, features, or materials of a product by working through an easy-to-understand interface without the need to gain CAD skills. The system can display immediate changes to 3D models, making it suitable for pre-production visualisation, quoting, and approval procedures. Altogether, it is an environment that is open, adaptive, and has real-time options; it can be considered a useful instrument, not only in the case of design, but also in terms of the broader potential of the industry exposures, which can be a boon in the case of lightweight, web-accessible CAD technology.

6. Conclusion and Future Work

This paper introduces and tests a Web-based CAD system utilising the latest web technologies, particularly WebAssembly, for high-performance geometric calculations. Through integrating a native CAD kernel (OpenCascade) into WebAssembly with the help of Emscripten, this system was able to port significant modeling features from desktop-based models into web browsers with minimal loss to performance and all the benefits of web apps accessibility and portability. These significant processes, such as parameter-based updates, boolean geometry modification, and mesh rendering, were performed with minimal latency, demonstrating that in-browser CAD is viable for real-life design. It also introduced the concept of dual storage architecture, where data can be persistently stored in local storage through the use of IndexedDB and cloud-based APIs. This structure enabled the modelling of offline data and facilitated smooth synchronisation with cloud services through data backup and sharing. Using the performance standards and user surveys, the system's functionality can be considered technically sound, along with being acceptable in terms of usability, portability, and responsiveness.

These contributions, taken together, confirm the viability of web-native CAD systems as a realistic alternative to desktop-based applications, especially in educational settings, collaboration, and lightweight industrial applications. Although the system shows great basic potential, its use is still limited by several future directions that offer to increase its functionality and extend its applicability. Another point is the possibility of integrating simulation tools, such as finite element analysis (FEA), kinematic simulation, or thermal analysis, which would greatly enhance the system's use as an engineering design validator. Such features may be configurable through WebAssembly compilations of open-source simulation libraries or cloud-computation pieces. The next impactful enhancement is working on multi-user editing, which will make the shared workspace possible, allowing co-workers to edit models in real-time, as is the case with new productivity software like co-editing documents. Such workflows would require implementing version control and conflict resolution constructs. Lastly, the idea of integrating AI recommendation systems directly into designs themselves, through which they can help and guide users by proposing the best parameters, identifying design deficiencies, or possibly automating other modelling tasks, is gaining popularity. Using machine learning models trained on CAD data, this feature would have enormous potential to significantly accelerate the design process and reduce the learning curve for novice users. All of this bodes well for a bright future ahead for the second generation of browser-based CAD solutions.

References

[1] Haas, A., Rossberg, A., Schuff, D. L., Titzer, B. L., Holman, M., Gohman, D., ... & Bastien, J. F. (2017, June). Bringing the web up to speed with WebAssembly. In Proceedings of the 38th ACM SIGPLAN conference on programming language design and implementation (pp. 185-200).

[2] Nicolaescu, P., Derntl, M., & Klamma, R. (2013, October). Browser-based collaborative modeling in near real-time. In 9th IEEE International Conference on Collaborative Computing: Networking, Applications and Worksharing (pp. 335-344). IEEE.

[3] Groß, T., Pfitzmann, B., & Sadeghi, A. R. (2005, September). Browser model for security analysis of browser-based protocols. In European Symposium on Research in Computer Security (pp. 489-508). Berlin, Heidelberg: Springer Berlin Heidelberg.

[4] Virtanen, J. P., Hyyppä, H., Kurkela, M., Vaaja, M. T., Puustinen, T., Jaalama, K., ... & Hyyppä, J. (2018). Browser-based 3D for the built environment. Nordic Journal of Surveying and Real Estate Research, 13(1), 54-76.

[5] Gonzalez, A., & Reid, L. G. (2005, May). Platform-independent accessibility api: Accessible document object model. In Proceedings of the 2005 International Cross-Disciplinary Workshop on Web Accessibility (W4A) (pp. 63-71).

[6] Baba, Y., & Nobeoka, K. (1998). Towards knowledge-based product development: the 3-D CAD model of knowledge creation. Research policy, 26(6), 643-659.

[7] Jazayeri, M. (2007, May). Some trends in web application development. In Future of Software Engineering (FOSE'07) (pp. 199-213). IEEE.

[8] Miller, M. (2008). Cloud computing: Web-based applications that change the way you work and collaborate online. Que Publishing.

[9] Di Lucca, G. A., & Fasolino, A. R. (2006). Testing Web-based applications: The state of the art and future trends. Information and Software Technology, 48(12), 1172-1186.

[10] Bates, C. (2002). Web Programming: Building Internet Applications. John Wiley & Sons.

[11] Encarnacao, J. L., Lindner, R., & Schlechtendahl, E. G. (2012). Computer-aided design: fundamentals and system architectures. Springer Science & Business Media.

[12] Sprumont, F., & Xirouchakis, P. (2002). Towards a knowledge-based model for the computer-aided design process. Concurrent Engineering, 10(2), 129-142.

[13] Lan, H. (2009). Web-based rapid prototyping and manufacturing systems: A review. Computers in industry, 60(9), 643-656.

[14] Pedro Company, Contero, M., Otey, J., Camba, J. D., Agost, M. J., & Pérez-López, D. (2017). Web-based system for adaptable rubrics: case study on CAD assessment. Journal of Educational Technology & Society, 20(3), 24-41.

[15] Ou, S. C., Sung, W. T., Hsiao, S. J., & Fan, K. C. (2002). An interactive web-based training tool for CAD in a virtual environment. Computer Applications in Engineering Education, 10(4), 182-193.

[16] Danesi, F., Denis, L., Gardan, Y., Lanuel, Y., & Perrin, E. (2001, July). Towards a Web-based CAD system. In Proceedings Fifth International Conference on Information Visualisation (pp. 269-274). IEEE.
[17] Reddy, E. J., Sridhar, C. N. V., & Rangadu, V. P. (2018). Development of a web-based knowledge-based system for CAD modeling and manufacturing. Materials Today: Proceedings, 5(13), 27241-27247.
[18] Álvares, A. J., Ferreira, J. C. E., & Lorenzo, R. M. (2008). An integrated web-based CAD/CAPP/CAM system for the remote design and manufacture of feature-based cylindrical parts. Journal of Intelligent Manufacturing, 19(6), 643-659.
[19] Rosenman, M., & Wang, F. (2001). A component agent-based open CAD system for collaborative design. Automation in Construction, 10(4), 383-397.
[20] Li, J., & Su, D. (2008). Support modules and system structure for a web-enabled collaborative environment in design and manufacturing. International Journal of Production Research, 46(9), 2397-2412.
[21] Vassallo, K., Garg, L., Prakash, V., & Ramesh, K. (2019). Contemporary technologies and methods for cross-platform application development. Journal of Computational and Theoretical Nanoscience, 16(9), 3854-3859.