Interactive 3D Rendering of the Human Heart on Mobile Web Using WebGL and Three.js

Sunardi[1], Herman[2], Krisna Astianingrum[3*]

[1] *Electrical Engineering Department, Universitas Ahmad Dahlan, Yogyakarta, Indonesia*
[2,3] *Master Program of Informatics, Universitas Ahmad Dahlan, Yogyakarta, Indonesia*

*corr-author: 2308048023@webmail.uad.ac.id

Abstract - The advancement of web-based 3D visualization technology has created new opportunities for interactive medical learning, particularly in anatomy education. The existing rendering techniques for the mobile web still face challenges due to limitations of cellular and mobile device capacity This study focuses on optimizing real-time rendering of an interactive 3D heart model for mobile web platforms using WebGL and Three.js. Several optimization techniques were applied, including Draco compression, polygon reduction, and the GLB file format, to achieve high rendering performance while maintaining anatomical accuracy. Performance testing was conducted on three device tiers—low-, mid-, and high-end—under different network conditions. Key metrics such as frame rate, loading time, and memory usage were systematically measured. The optimized system achieved stable rendering at 58–60 FPS with a reduced loading time from 6.2 seconds to 1.4 seconds, demonstrating strong scalability and responsiveness. From an educational perspective, this interactive 3D heart model enables medical students, trainees, and patients to dynamically explore cardiac anatomy, improving their spatial understanding of complex structures without requiring high-end VR hardware. The novelty of this work lies in its optimization pipeline tailored for mobile web, making real-time anatomical visualization lightweight and accessible. Future research will involve larger user studies to evaluate educational effectiveness.

Keywords: 3D Rendering; mobile web; WebGL; Three.js; human heart.

I. INTRODUCTION

The rapid growth of smartphone technology has transformed smartphones from communication tools into multifunctional personal assistants [1]. The implementation of a web-based framework enhances the efficiency of cloud computing migration; however, it still faces challenges in mobile access performance [2]. Mobile web applications provide easy access across various devices without installation requirements. However, 3D rendering on this platform often faces performance hurdles due to limitations in processing power, memory, and network bandwidth [3].

One of the notable studies focusing on 2D visualization in medical education is "The effect of image quality, repeated study, and assessment method on anatomy learning" [4]. This research investigated how the quality of 2D anatomical images affects learning outcomes among medical students. The study utilized static 2D illustrations to teach anatomical structures of the hand and eye, examining how different image qualities and repetition of study influenced knowledge retention and understanding. The findings indicated that higher image quality improved learning outcomes, while lower quality images negatively affected comprehension. This highlights the importance of visualization quality in medical learning environments that rely solely on 2D resources. However, this approach has several inherent limitations. First, static 2D images cannot effectively convey spatial and depth information, which are crucial for understanding complex anatomical relationships. Second, 2D visualization lacks interactivity, preventing learners from rotating, zooming, or exploring structures from multiple perspectives. Third, the effectiveness of learning is highly dependent on image quality, making it less adaptive to varied learning environments or devices. Finally, 2D visualization tends to be less engaging, which can affect learner motivation and retention over time. Advances in web-based visualization technologies offer promising solutions to these limitations. WebGL enables real-time, hardware-accelerated 3D rendering directly within web browsers, allowing complex anatomical models to be visualized interactively without installing additional software. In combination with Three.js, developers can implement features such as camera manipulation, dynamic zoom, and real-time object interaction more easily. This makes it possible to build immersive learning environments that provide depth cues, interactive perspectives, and better anatomical comprehension compared to static 2D images.

Moreover, these technologies support cross-platform access, ensuring better adaptability and scalability for broader medical education use. Optimizing 3D rendering in mobile web applications is crucial to ensure a smooth and satisfying user experience [5].

The quality of user interaction with 3D models depends not only on rendering speed but also on the fluidity of navigation, including rotation, zooming, and panning. Touch-based interaction on mobile screens requires an intuitive and responsive interface design to minimize interaction barriers and maximize learning effectiveness. The novelty of this study lies in rendering 3D anatomical objects — specifically the heart and cardiovascular apparatus — on mobile web applications using WebGL and Three.js, while optimizing model compression and employing the GLB format to ensure efficient loading and smooth real-time interaction. This approach aims to enhance accessibility, performance, and interactivity of medical 3D visualization in educational and clinical simulation contexts. The use of 3D technology has also become increasingly significant across various fields such as gaming, architecture, education, and e-commerce [6]. Users not only expect visually appealing displays but also responsive and interactive performance. In the context of mobile web applications, the challenge is the limited resources of the cellular network compared to mobile native applications [7].

This research aims to explore and develop effective optimization methods for rendering 3D models in mobile web applications [8]. One of the efforts is to minimize computational load and energy consumption, thereby extending the battery life of mobile devices and improving application responsiveness [9]. Through comprehensive literature studies and experiments, this research will identify the main challenges in 3D rendering on mobile web applications and propose innovative solutions [10]. One effective approach is implementing Level of Detail (LOD) techniques, which dynamically adjust model complexity according to the user's viewing distance. In addition, efficient shader usage and minimizing draw calls can significantly improve frame rates. These optimizations help ensure stable rendering performance on mobile devices. The results of this research are expected to contribute not only academically to the field of information technology but also provide practical guidance for mobile web application researchers in delivering optimal user experiences.

Mobile web refers to internet access applications using browser-based devices for mobile devices, aimed at wirelessly accessing data services using mobile devices such as phones, PDAs, and portable devices via cellular/wireless networks [11]. In mobile contexts, touch interaction provides users with an intuitive way to explore 3D objects directly. Gestures such as pinch-to-zoom, swipe, and tap create a more natural and immersive experience. This is especially relevant for medical education applications, where students can better understand anatomical structures through interactive visualization. Mobile web applications can offer high flexibility and accessibility as they can be accessed through various devices without requiring special installation [12]. Mobile web applications offer inclusive accessibility across devices and can reach users in regions with limited technological infrastructure.

Limitations in wireless/cellular networks make it difficult to access online services, slow data transfer, and degrade the user experience, especially as many people rely on technology for their daily activities. The limitations are significant external factors affecting the performance of 3D web applications. Network delays can cause loading latency and disrupt learning experiences. To address this, techniques such as Draco compression and lazy loading can accelerate the delivery of 3D content to users' browsers without compromising visual quality. Hardware limitations on mobile devices, such as slower processors and limited memory, often pose obstacles in rendering complex 3D models. Hardware constraints on mobile devices require developers to adopt well-structured optimization strategies. Reducing polygon counts, applying baked lighting to lower computational loads, and using instancing techniques can significantly decrease resource consumption. This allows 3D models to run smoothly on minimal hardware without sacrificing critical anatomical details. Therefore, optimizing 3D rendering on this platform is essential to ensure that users receive an adequate experience without performance degradation [13].

This study systematically identifies key challenges in mobile 3D rendering and evaluates the effectiveness of applied optimization techniques to guide future development. Thus, the contributions of this research are not only academic but also practical, offering useful guidance to enhance the performance of mobile web applications in the increasingly evolving 3D technology era. In the context of medical informatics, optimizing 3D rendering on mobile platforms can provide broader access to high-quality medical visualizations, especially for medical students, clinicians in remote areas, and patient education [8]. With optimized mobile web performance, 3D cardiac models can be accessed more efficiently without requiring expensive VR hardware or

high-end computers, enabling more inclusive and scalable medical learning platforms [9,14,15]. Through a systematic approach and structured experiments, this research will identify various challenges faced in 3D rendering on mobile web applications. Additionally, this research will evaluate the effectiveness of the optimization techniques applied and provide recommendations for researchers in implementing them. Thus, the contributions of this research are not only academic but also practical, offering useful guidance to enhance the performance of mobile web applications in the increasingly evolving 3D technology era [10,16].

II. METHOD

This research encompasses several stages, starting with a literature review on 3D rendering technologies, including a technical analysis of features and capabilities [17,18,19]. The literature review examines topics related to 3D rendering, Web Graphics Library (WebGL), Three.js, and optimization on mobile devices, followed by an analysis of the requirements for interactive features and the design of the human heart model. The design of the 3D model includes adding interactive features by creating a 3D model of the human heart and determining the types of interactions (rotation, zoom, animation) for the user experience.

An initial prototype is developed using Three.js and WebGL, implementing the 3D model of heart and basic interactivity for early implementation. Next, the performance of the rendering is optimized to ensure smooth operation on mobile devices with resource limitations such as CPU, memory, and bandwidth. Performance testing and user experience evaluation are conducted on various mobile devices, along with an assessment of visual quality and interaction responsiveness. Weaknesses identified from the testing results are addressed, and comprehensive documentation regarding the processes, techniques, and outcomes of the research is created.

The prototype was developed by designing the system architecture, creating the 3D heart model, implementing rendering using WebGL and Three.js, and testing its performance and compatibility. An analysis of the features and technical capabilities of WebGL and Three.js is performed, including 3D rendering capabilities, compatibility with mobile devices, memory and resource usage, and support for various 3D file formats [20,21,22]. The entire sequence of the research is illustrated in the flowchart in Fig. 1.

Research and optimization of 3D model rendering in mobile web applications require appropriate hardware and software to ensure performance, efficiency, and platform compatibility. The choice of hardware and software must align with the needs of the 3D object creation and animation process, rendering, and accessibility via mobile web. The use of WebGL-based rendering technology is the primary choice to leverage hardware acceleration for displaying 3D graphics directly through web browsers [23]. Therefore, various hardware that supports the research process is detailed in Table I.

The hardware and software used in this research were selected to support efficient 3D rendering and mobile accessibility. Only essential specifications are presented to emphasize their role in ensuring performance consistency across devices. The design of the mobile web application structure includes the user interface (UI), navigation system, and integration with rendering technology. The 3D model of the human heart is created in Blender, and the model is subsequently rendered using WebGL and Three.js. The software used to support the research is detailed in Table II. The software is categorized into testing software, operating systems, modeling tools, and mobile web development tools. This selection ensures that the tools used genuinely support and align with the research methodologies.

The 3D heart model is displayed in a vertical position, showcasing realistic textures and materials that enhance its visual appeal. This model is derived from a GLB file, which has been specifically optimized for rendering using WebGL technology. The choice of a vertical orientation not only highlights the anatomical features of the heart but also allows for a more immersive viewing experience. In the background, a white grid serves as a reference for position and scale within the 3D space. This grid is crucial because it provides a frame of reference, helping viewers better understand the heart model's dimensions and spatial relationships. By incorporating this grid, the model gains context, making it easier for users to orient themselves and appreciate the intricacies of the 3D representation.

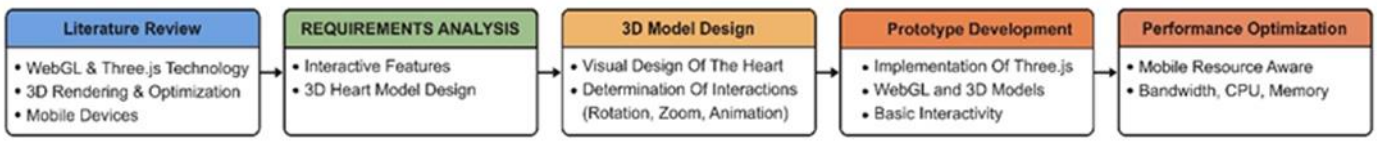

Fig. 1 Block diagram of the research

TABLE I
HARDWARE SPECIFICATION

Component		Specification
Laptop	Device name	DESKTOP-6A9TQ3C
	Processor	11th Gen Intel(R) Core(TM) i7-1185G7 @ 3.00GHz 1.80 GHz
	Device ID	Installed RAM 16,0 GB (15,4 GB usable)
	Product ID	9AE4E3E2-A611-4626-AD38-BE5BA6874BB5
	System type	00330-80000-00000-AA936
	Pen and touch	64-bit operating system, x64-based processor
		Pen and touch; No pen or touch input is available for this display
Hp Oppo Reno 8T (CPH2481)	Display	6.43-inch AMOLED, Full HD+ (2400 x 1080 pixels)
	Processor	MediaTek Helio G99, an octa-core chipset suitable
	RAM & Storage	8 GB of RAM and 256 GB of internal storage
	Operating System	ColorOS 13 based on Android 13

The Blender rendering showcases the model with high fidelity and detail, while the WebGL rendering demonstrates the capability of real-time graphics on mobile devices. This comparison emphasizes the advancements in rendering technology, allowing for a more accessible and interactive experience for users. The ability to render complex 3D models in real-time on mobile web platforms opens up new possibilities for education, medical training, and visualization. Users can interact with the model, rotating and zooming in to explore the heart's anatomy in detail. This interactive element not only enhances learning but also engages users in a way that static images cannot. Overall, the integration of realistic textures, the use of a reference grid, and the comparison of rendering techniques highlight the potential of 3D modelling and rendering in various applications, making complex subjects more approachable and understandable for a wider audience [24].

This heart model is loaded from an external file named heartGLB.glb. The rendering process utilizes DracoLoader to integrate the model into a 3D canvas within the browser. This type of rendering allows users to interact with the 3D model, enabling them to rotate, zoom in, or pan the model using a mouse or other controls [24]. This display demonstrates the successful application of WebGL in presenting the 3D heart model with real-time rendering capabilities in the browser.

The 3D heart model used in this study was adapted from open-access medical datasets to ensure transparency and reproducibility. Specifically, the model was obtained from NIH 3D Print Exchange, which provides scientifically validated anatomical models derived from medical imaging data. The dataset is licensed under Creative Commons Attribution 4.0 (CC-BY 4.0), allowing modification and redistribution with proper attribution. This ensures that the model source is both credible and accessible to other researchers for further use. After obtaining the raw 3D model, several optimization steps were performed to adapt it for mobile web rendering. The model was processed using Blender, including mesh simplification, topology cleanup, and polygon reduction to achieve better performance without compromising anatomical accuracy. Texturing and basic shading were also applied to enhance visual clarity and support real-time interaction in web-based environments. No patient-identifiable information was included in the dataset, as the heart model represents generic anatomical structures derived from de-identified medical imaging. This guarantees compliance with ethical and licensing standards while supporting reproducibility of the research. By using an openly licensed dataset combined with optimization techniques, this study ensures that the 3D model can be easily accessed, modified, and validated by other researchers in the field of medical informatics and education [25,26].

TABLE II
SOFTWARE

Category	Software Name	Purpose
Operating System	Windows 10	Platform for development and testing
3D Modeling Software	Blender	Creating the 3D model of the human heart
Graphic library	Three.js	Implementing 3D rendering in the web app
Graphic API	WebGL	Enabling hardware-accelerated graphics
Testing Tool	Browser	Testing application functionality
tool used to load 3D geometry	Draco loader	has been compressed supports various
source code editor developed by Microsoft	Visual Studio Code (VS Code)	programming languages

III. RESULT AND DISCUSSION

WebGL is a JavaScript API that provides direct access to the Graphics Processing Unit (GPU) in web browsers for rendering 2D and 3D graphics in a hardware-accelerated manner without the need for additional plugins. WebGL serves as the foundation for graphics rendering in modern browsers, enabling the creation of complex 3D visualizations that run in real-time across various devices, either mobile or desktop.

The simplification of WebGL usage is achieved by providing abstractions that are easier to understand and use, such as mesh objects, cameras, lighting, and animations, which require Three.js, a high-level JavaScript library built on top of WebGL. WebGL is low-level and requires manual handling of shaders, buffers, and complex rendering loops. With Three.js, researchers can create and manage 3D scenes with just a few lines of code, enhancing productivity while maintaining optimal rendering performance.

3D models typically contain large vertex data and attributes, which can result in significant download times and memory usage when used on the web. This necessitates the use of specialized libraries for compressing and decompressing 3D geometry data, such as Draco, developed by Google. Draco significantly reduces the file size of 3D models through efficient compression without sacrificing visual quality. This accelerates the delivery and loading of 3D assets, especially over slow networks like those found on mobile devices.

WebGL serves as the rendering foundation, while Three.js simplifies the development of applications that utilize WebGL to display 3D graphics. Draco is used to compress 3D data (e.g., models in the glTF format supported by Three.js) for faster downloading and processing. In practice, this research employs Three.js to create scenes and display 3D models rendered by WebGL. Before the model is loaded, Draco can be applied to compress the 3D data. When the web application is run, Three.js uses the Draco decoder to decompress the model directly in the browser, then passes the decompressed data to WebGL for rendering. The process of using Three.js and Draco to display 3D models in a web application begins with the researcher displaying a 3D model of the human heart. This model is created using Blender modelling software and then exported to the format required by WebGL, which is glTF.

The human heart object originates from medical scanning technologies such as MRI, CT scans, and similar technologies. There is a connection between the Physical World and the Digital World in the context of medical health. This illustrates how data from the physical world is processed in the digital realm to provide beneficial services to users, including patients, doctors, and healthcare professionals. The physical components that are monitored and analysed make the human heart the focus of this research. Health conditions can be assessed based on the state of the heart. Efforts are made to minimize the loading time of rendered objects to enhance system responsiveness, allowing 3D objects to appear smoother and more responsive [20]. 3D modelling has become a crucial component in digital production today, spanning various fields such as 3D assets, interior design, architecture, film, gaming, animation, and more. Rendering optimization can be demonstrated, for instance, through results that feature overhead lighting, as shown in Fig. 2.

The research object is a human heart that has been prepared in Blender through an export process to obtain an object with the GLTF Binary (.glb) extension. GLB is the binary format of GLTF that allows Blender to export 3D models in a single file, which includes geometry, textures, animations, materials, and other important information, as shown in Fig. 2. The scaling needs are adjusted according to the type of object selected to synchronize the measurement units to match the expected size in WebGL. The adjusted scale will also affect the level of texture smoothness of the 3D model. Once the scale has been adjusted, it will be exported to obtain the GLB extension. The GLB format is binary, making it more compact than the text-based GLTF format; therefore, it is smaller and more efficient for distribution and online transfer. The GLB extension allows for the use of 3D models in web environments without requiring extensive additional setup, enabling WebGL to render more quickly. The export process is illustrated in Fig. 3.

This step produces a 3D object with a GLB extension on Fig. 3. GLB is designed to provide a balance between file size and quality, making it ideal for use on the web or mobile devices that require fast performance without sacrificing the visual quality of the 3D model. Compressed textures and optimized geometry allow GLB files to be lightweight while still retaining the necessary detail. The GLB format is highly compatible with various modern 3D applications and game engines, one of which is Three.js, a JavaScript library for WebGL. This facilitates collaboration and the exchange of 3D models across different platforms without the need for repeated format conversions.

After the 3D model is exported, the next step is to compress it using Draco. Draco is a mesh compression library designed to reduce the size of 3D data without sacrificing visual quality. This process involves using the Draco compression tool to compress the 3D model file, resulting in a compressed file that can be used in web applications. Computational limitations become a major aspect in the 3D rendering process because rendering requires intensive data processing, including geometry processing, lighting, and texturing. When hardware is unable to handle this complexity, rendering times increase, and the quality of the final output can be affected. The 3D rendering process involves processing models with thousands to millions of polygons. Computational limitations can lead to difficulties in processing.

Lighting techniques such as global illumination and ray tracing require deep calculations to simulate the interaction of light with objects. Limitations in computational power can result in reduced quality of lighting and shadow effects produced. The use of high-quality textures and complex materials also requires significant resources. Computational limitations can extend rendering times, impacting workflow efficiency. With computational limitations, artists may be forced to sacrifice visual quality for speed. This can diminish the aesthetic appeal of the final output, especially in highly competitive industries such as film and gaming. In large projects involving many elements, computational limitations can restrict the ability to add additional details or effects, which are crucial for creating an immersive visual experience. Researchers then set up a web application involving the Three.js library and the Draco decoder library in a project that subsequently configures the scene, camera, and WebGL renderer.

Among these, JavaScript plays a crucial role as the primary scripting language for creating interactive web content. It enables developers to implement complex functionalities and dynamic behaviors within web pages. HTML (HyperText Markup Language) serves as the backbone of web content, providing the structure and layout for web applications. It allows developers to define elements such as headings, paragraphs, images, and links, which are essential for presenting information to users. WebGL is a JavaScript API that enables rendering 2D and 3D graphics within any compatible web browser without the need for plugins. It leverages the capabilities of the GPU to provide high-performance graphics rendering, making it ideal for applications that require real-time rendering of complex 3D models. Together, these technologies form a powerful toolkit for developers, allowing them to create rich, interactive experiences on the web. The integration of these libraries and technologies facilitates the development of applications that can efficiently handle 3D graphics, enhancing user engagement and interactivity.

Fig. 4 showcases the integration of the Three.js and Draco libraries, which are pivotal in the development of 3D web applications. Three.js is a popular JavaScript library that simplifies the process of creating and displaying animated 3D graphics in a web browser using WebGL. It provides a wide range of features, including support for various geometries, materials, lights, and cameras, making it easier for developers to build complex 3D scenes. The library abstracts many of the complexities of WebGL, allowing developers to focus on creating engaging visual experiences without needing to delve deeply into the underlying graphics API. Draco, on the other hand, is a mesh compression library designed to reduce the size of 3D models without compromising their visual quality. By compressing the geometry and texture data of 3D models, Draco enables faster loading times and reduced bandwidth usage, which is particularly beneficial for web applications that need to deliver 3D content efficiently. The integration of Draco with Three.js allows developers to easily load and render compressed 3D models, enhancing performance and user experience.

The mobile web application that renders the 3D model has been tested in various network types in different locations. The test results are shown in Table III.

The higher bandwidth networks such as WiFi and 5G showed superior performance, with average page loading times under 1.5 seconds and stable rendering of the 3D anatomical model. In particular, the 5G network achieved the fastest response time of 0.29 seconds, followed by 4G and WiFi connections, indicating excellent compatibility between the web-based application and high-speed networks. Conversely, 3G networks displayed slight delays but still maintained stable access, whereas 2G connections consistently failed to load the application, highlighting their inadequacy for rendering interactive 3D content. These findings emphasize that network speed and stability significantly influence rendering performance and user experience, reinforcing the importance of optimizing web-based visualization applications for higher-speed network environments to support real-time interaction and educational use.

This study also evaluates users' perception of the mobile web application. Table IV shows the users' perception and suggestions for the proposed system.

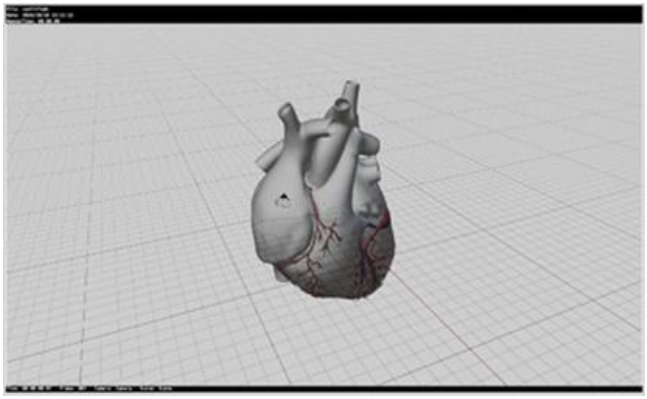

Fig. 2 Human heart object

Fig. 3 Exporting the object to obtain the GLB Extension

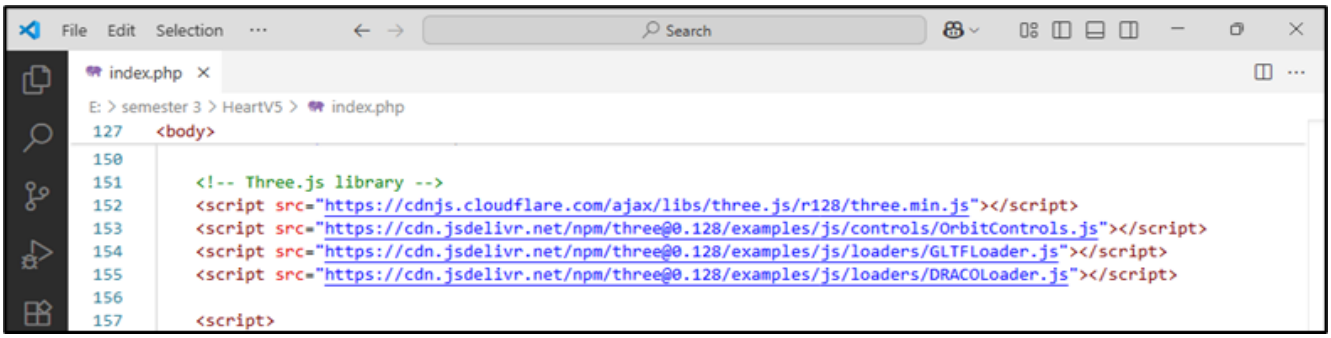

Fig. 4 The Three.js and Draco script libraries

TABLE III
TEST RESULT

No	Test Location Coordinates	Network Type	Mbps Download	Mbps Upload	Loading Time (seconds)	Image Description
1	7°37′47.1″ S, 111°08′01.4″ E	WiFi IndiHome	20.3	5.43	1.29	Page loaded successfully and remained stable.
2	7°37′47.1″ S, 111°08′01.4″ E	WiFi ICONET	20.9	8.74	1.31	Page opened quickly and interactively.
3	7°37′47.1″ S, 111°08′01.4″ E	4G Mobile Network	34.2	11.6	1.14	Page loaded fast, 3D model fully rendered.
4	7°37′47.1″ S, 111°08′01.4″ E	3G Mobile Network	8.6	2.1	1.33	Slight delay but page loaded stably.
5	7°37′47.1″ S, 111°08′01.4″ E	2G Mobile Network	0	0	-	Page could not be loaded (unstable connection).
6	7.7927° S, 110.3828° E	BSI UAD Campus 3 Access Point	8.97	15.3	1.21	Page loaded with full interactive display.
7	7.7927° S, 110.3828° E	4G Mobile Network	24.5	9.8	1.25	Page loaded well and was responsive.
8	7.7927° S, 110.3828° E	3G Mobile Network	7.4	1.9	1.37	Page loaded slowly but successfully.
9	7°48′05″ S, 110°21′53″ E	5G Mobile Network	46.8	18.5	0.29	Page loaded very fast with smooth interaction.
10	7°48′05″ S, 110°21′53″ E	4G Mobile Network	32.0	13.0	0.32	Page loaded quickly and remained stable.

TABLE IV
USER'S PERCEPTION AND SUGGESTION

No	Positive Feedback	Improvement Suggestion
1	Navigation is smooth and responsive.	Add more anatomical detail on small structures.
2	Fast response and clear visualization.	Improve initial loading speed on slower networks.
3	Good performance overall.	Zoom interaction can be made more precise.
4	Easy to use and explore.	Add label toggle feature for learning clarity.
5	Visualization clear despite network lag.	Optimize for low-bandwidth conditions.
6	High-quality 3D rendering.	Provide dark mode or adjustable contrast.
7	Excellent user experience.	Add a brief instruction or tutorial for first-time users.
8	Interaction is responsive and clear.	Better zoom control and smoother panning.
9	Near-instant loading on 5G, excellent clarity.	Increase model detail for clinical learning.
10	Stable and good performance.	Improve performance on 3G networks.

The qualitative feedback from respondents further supports the positive quantitative findings. Most participants highlighted smooth navigation, fast loading, and clear visualization as key strengths of the application. Constructive feedback primarily focused on enhancing anatomical detail, improving zoom and panning precision, and optimizing performance under low-bandwidth conditions, particularly on 3G networks. Several users also suggested the addition of instructional features, label toggling, and customizable display options to support more flexible learning experiences. This combination of positive feedback and targeted improvement suggestions provides valuable insight for iterative development, ensuring that future versions of the system can offer not only technical performance excellence but also richer educational functionality and accessibility. Besides users' perception, this study also evaluates the user acceptance of the application. Table V shows the results of the user acceptance test.

The user acceptance test indicates a 5 high level of user experience across all tested dimensions, with the majority of respondents rating ease of use, visual clarity, interaction responsiveness, and overall satisfaction at "good" to "very good" levels (mean scores ranging from 4 to 5). This indicates that the user interface design and rendering performance are effectively optimized to support intuitive navigation and real-time interaction. The high ratings for visual clarity confirm that the use of 3D rendering enhances anatomical comprehension, while strong responsiveness scores reflect stable performance across different network conditions. Minor issues such as zoom control and network-related latency on slower connections were noted but did not significantly reduce overall satisfaction. These findings emphasize the platform's strong usability foundation and its readiness for broader implementation in medical education contexts.

After the web application is ready, the researchers load a 3D model of the human heart into the application using DracoLoader, which is responsible for loading the compressed files. They then configure the loader to use the included Draco decoder. When the compressed model is loaded, DracoLoader decompresses the model directly in the browser. This process involves retrieving the compressed data and using the Draco decoder to convert it back into a mesh format that can be utilized by Three.js, resulting in a mesh object that is ready for rendering.

Once the model is decompressed, the final step is to render the model using WebGL. This process includes adding the mesh object to the Three.js scene, setting up lighting, materials, and other visual effects, and calling the render function to display the scene on the screen. The researchers then added interactivity to enhance the user experience by incorporating device sensor controls, allowing users to count steps while accessing the web application.

They also conducted testing to ensure that the 3D model was displayed correctly across various devices and browsers. If necessary, optimizations could be made to improve performance and loading times. With this approach, loading times are reduced, bandwidth usage is minimized, and rendering can proceed smoothly on both mobile and desktop devices. Thus, the three components complement each other: WebGL serves as the rendering engine, Three.js acts as the user-friendly framework, and Draco functions as the optimizer for 3D data size to achieve maximum performance on the web.

TABLE V
USER ACCEPTANCE RESULT

No	Respondent	Ease of Use (1–5)	Visual Clarity (1–5)	Interaction Responsiveness (1–5)	Overall Satisfaction (1–5)
1	R1	5	5	5	5
2	R2	4	5	4	4
3	R3	4	4	4	4
4	R4	4	5	4	4
5	R5	3	4	3	3
6	R6	4	5	4	4
7	R7	5	5	5	5
8	R8	4	4	4	4
9	R9	5	5	5	5
10	R10	4	4	4	4

The 3D rendering capabilities of WebGL and Three.js reach 60 FPS on high-spec mobile devices. The 3D model of the human heart can be rendered with a file size of 3 MB and a rendering time of under 1.4 seconds. The mobile web application can run on various browsers, including Chrome, Firefox, and Safari, as well as on Android and iOS mobile devices. Memory usage and system resource allocation occur optimally.

Optimal memory and system resource usage ensure that device performance is not burdened. Compatibility is achieved as the application can run on various devices and browsers. Memory limitations also become a key aspect in the 3D rendering process, as rendering involves processing and storing large amounts of data, including geometry, textures, and lighting information. 3D models often consist of millions of polygons and vertices, with each element requiring memory space for storage. Memory limitations can restrict the size and complexity of models that can be rendered, thereby reducing detail and realism. High-quality textures, often used to provide visual detail on object surfaces, require significant memory space. If memory is limited, artists may have to use lower-resolution textures, which can diminish visual quality. Advanced lighting techniques, such as global illumination and ray tracing, require storage for information about light sources and light interactions with objects. Memory limitations can hinder the ability to store this data efficiently, which can affect the quality of the resulting lighting.

During the rendering process, data is often stored in caches or buffers to speed up access. Memory limitations can reduce the size of these caches, which can slow down the rendering process as data must be fetched from main memory more frequently. 3D rendering is often performed in parallel, where many elements are processed simultaneously. Memory limitations can restrict the number of threads or processes that can run concurrently, which can reduce rendering efficiency and speed. Memory limitations can impact the final quality of the render. If there is not enough memory to store all the necessary data, the rendered output may appear less detailed or unrealistic.

Fig. 5 explain scale of the 3D model of the human heart being in binary format, GLB is more compact than the text-based GLTF format, resulting in a smaller size and greater efficiency for distribution and online transfer. The GLB extension allows the use of 3D models in web environments without requiring extensive additional setup, enabling WebGL to render more quickly. One built-in feature that is ideal for use in WebGL-based application development is Visual Studio Code (VS Code), a lightweight yet powerful source code editor that supports various programming languages and technologies, including JavaScript, HTML, and WebGL. Complete features such as syntax support, debugging, and built-in terminal make VS Code a very helpful tool for WebGL application research. The rendering process and manipulation of 3D models become easier in the browser using libraries like Three.js, while keeping the code neat and efficient. The JavaScript code written in VS Code (for example, in the scripts.js file) serves to create and control the 3D heart elements within the HTML5 canvas. This code interacts directly with the WebGL API or libraries like Three.js. The 3D heart file, now in GLB format and stored in the assets folder, is loaded using assetLoader. Using VS Code greatly simplifies the editing and importing of this 3D heart model. For this research, WebGL requires an additional library, namely Three.js, which is managed using npm

from the VS Code terminal. The command `npm install three` is used to install Three.js for this research.

After writing and running the code, Live Server will be used to view the 3D heart model rendered in the browser. If there are issues such as model loading errors (e.g., "Error loading the model"), debugging can be performed to fix them. VS Code is equipped with built-in debugging tools that allow monitoring and inspecting the code, enabling the viewing of variables, checking for errors, and tracking animations or rendering in the WebGL project. The use of a 3D grid configuration and mobile web server allows rendering to be executed and tested in real-time, providing users with a detailed visual representation of the loaded model. This stage demonstrates a successful WebGL application for displaying the 3D heart model with real-time rendering in the browser, as shown in Fig. 6.

The 3D heart model is displayed in a vertical position with realistic textures and materials. This heart model is the result of loading from a GLB file used in optimizing rendering with WebGL. The white grid in the background serves as a marker for position and scale in 3D space, helping provide visual context for the orientation and position of the 3D model within the 3D environment. It also demonstrates potential use in medical education by enabling interactive anatomical visualization. The figure shows a comparison of the 3D heart model in Blender with a more realistic visual appearance resulting from real-time rendering on the mobile web.

The heart model displayed on the mobile web, as shown in Fig. 6, is a 3D object rendered using WebGL through the Three.js library. This heart model is loaded from the external file `jantungGLB.glb` in the assets, as seen in the previous Fig. 6. By using DracoLoader, the model is rendered into the 3D canvas in the browser. This result allows users to interact with the 3D model, such as rotating, zooming, or panning the model with a mouse or other controls. This view demonstrates that the WebGL application successfully displays the 3D heart model with real-time rendering in the browser. With the configuration of a 3D grid and a local server, the 3D heart model can be run and tested in real-time, providing users with a detailed visual representation of the loaded model. The use of the GLB/GLTF format for the heart model is a lightweight yet highly efficient file format for WebGL. GLTF is a standard for storing 3D models that includes geometry, textures, animations, and materials in a single small compressed package. The use of GLTF allows for faster model loading times and is also compatible across various devices and browsers.

Network bandwidth limitations also affect the 3D rendering process. Bandwidth limitations may slow data transfer, disrupting real-time rendering and degrading user experience. Bandwidth limitations can cause lag or a decrease in visual quality, which can detract from the user experience. Touch-based user experiences in interactive 3D rendering refer to user interactions with 3D models through devices that support touch input, such as tablets, smartphones, or touch screens. This experience is crucial in various applications, including design, education, gaming, and presentations. User experience in interacting with the application is shown in Fig. 7.

Users can interact with 3D models using natural touch gestures, such as tapping, swiping, and pinching. This makes the experience more intuitive and easier to understand, even for users without a technical background. Touching to rotate, zoom in, and zoom out of the 3D model with finger movements allows users to view the object from different angles and gain a better understanding of its shape and details. Touch-based experiences can enhance accessibility for users with varying abilities. A well-designed interface can facilitate interaction with 3D content for users with physical limitations.

The development and optimization of the 3D cardiac model demonstrate that the visualization successfully represents the main anatomical structures of the heart, including the four chambers (right and left atria and ventricles), the primary valves (tricuspid, mitral, pulmonary, and aortic), as well as major vessels such as the aorta, vena cava, pulmonary arteries, and pulmonary veins. These anatomical details serve as a strong foundation to ensure that the model provides not only technological vlue but also medical relevance to support cardiovascular anatomy learning effectively. As part of the usability testing, a direct observation session was conducted at Tugu Jogja to evaluate system accessibility under different network conditions. One tester carried a 5G-enabled smartphone on-site and followed a structured procedure to access the provided URL. Device specifications and GPS coordinates were recorded, followed by accessing the system using three different network types (3G, 4G, and 5G). A total of four screenshots were collected to document the process and system behavior.

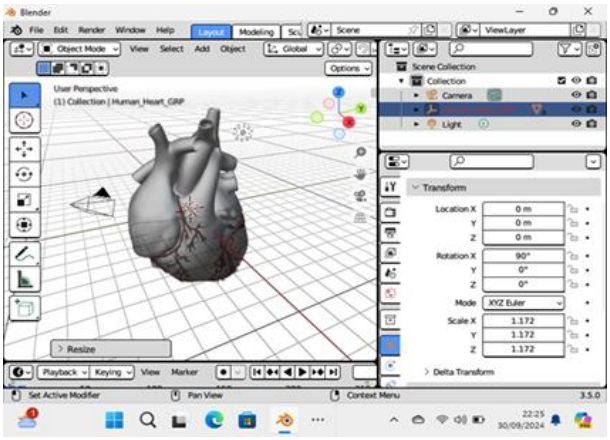

Fig. 5 Scale of the 3D model of the human heart

Fig. 6 3D model rendering

The observation showed that the system was accessible on all three networks, but the loading time varied significantly. On 3G, the page took noticeably longer to load, while on 4G the response was moderate. The fastest and most stable access was recorded on 5G, with minimal delay and smooth loading. Tester feedback indicated positive usability, with stable interface performance. However, lower network speeds reduced user comfort. The feedback suggests that while the system is generally usable and stable, optimizing for slower networks could improve user experience further. Based on these findings, the 3D heart model shows promising potential as an accessible and interactive web-based anatomy learning tool. Further development—particularly through improving anatomical precision and expanding the evaluation to larger user groups—will strengthen both its medical and educational validity. This approach is expected to bridge the integration of interactive 3D technology with modern medical education practices, making them more inclusive and widely accessible.

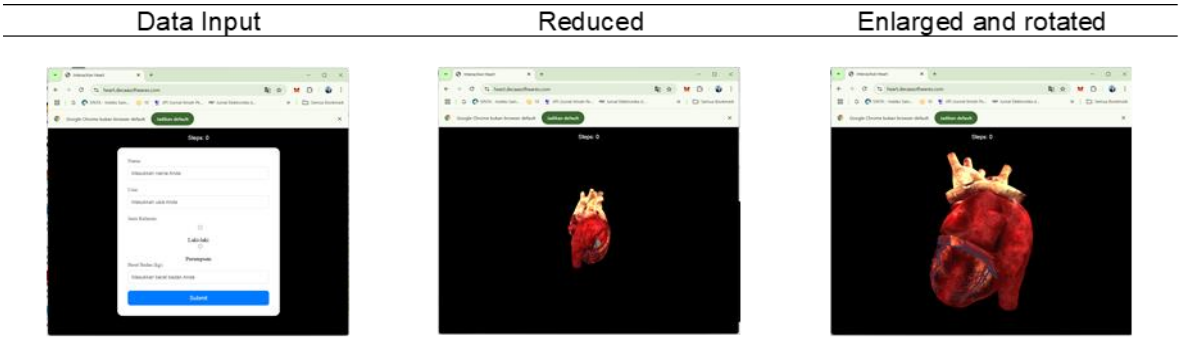

Fig. 7 Various user interactions in real-time

IV. CONCLUSION

This study successfully developed and evaluated a small-scale application as a Proof of Concept (PoC) Prototype for 3D cardiac visualization in medical education. The prototype was tested on five medical students to assess accessibility, usability, and anatomical clarity through direct observation. The preliminary results demonstrated that the application was accessible across different network conditions and provided a positive user experience in terms of interface intuitiveness and visual presentation. These findings indicate the potential applicability of the prototype as a complementary learning tool that can enhance students' understanding of complex cardiac anatomical structures through interactive 3D visualization. Future work should focus on enhancing the system's technical performance and scalability. This includes implementing a more efficient dynamic compression algorithm, investigating specific rendering requirements for human organ visualization, and expanding user testing to involve a larger and more diverse group of participants. By addressing these aspects, the application can be further developed into a robust educational platform that complements traditional instructional resources and supports more effective medical learning experiences. This study demonstrates the feasibility of a mobile web-based 3D cardiac model using WebGL and Three.js to enhance performance and interactivity. Nevertheless, the current work is constrained by the use of a small-scale testing environment and a limited range of hardware devices, without broader user involvement or comprehensive educational validation. Addressing these limitations will be the focus of future work, which includes optimizing the application for multiple device types to ensure stable performance across platforms, as well as conducting extensive user studies and educational validation to measure its impact on learning outcomes. Furthermore, future research will emphasize the refinement of interactive features and rendering algorithms to support more complex anatomical structures and clinical learning scenarios, strengthening its potential integration into medical education.

REFERENCES

[1] N. M. Farhan and B. Setiaji, "Indonesian Journal of Computer Science," *Indonesian Journal of Computer Science*, vol. 12, no. 2, pp. 284–301, 2023.

[2] N. Setiyawati and V. A. O. P. Warisman, "Integrasi Framework Kivy dan Webix pada Pembangunan Framework Mobile Web Easy Development System," *JUITA: Jurnal Informatika*, vol. 8, no. 2, p. 253, Nov. 2020, doi: 10.30595/juita.v8i2.7464.

[3] F. Adinda, A. K. Hidayah, D. Sunardi, and M. Muntahanah, "Comparison of the Use of Blender and Sketchup Applications in 3d Animation (Case Study: PT Rico Putra Selatan," *Jurnal Komputer, Informasi dan Teknologi*, vol. 2, no. 2, pp. 569–570, 2022, doi: 10.53697/jkomitek.v2i2.1011.

[4] B. Fenesi, C. Mackinnon, L. Cheng, J. A. Kim, and B. C. Wainman, "The effect of image quality, repeated study, and assessment method on anatomy learning," *Anat Sci Educ*, vol. 10, no. 3, pp. 249–261, Jun. 2017, doi: 10.1002/ase.1657.

[5] J. Munkberg. "Extracting Triangular 3D Models, Materials, and Lighting From Images," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2022-June, pp. 8270–8280, 2022, doi: 10.1109/CVPR52688.2022.00810.

[6] A. Rosidi and A. Afriyudi, "Aplikasi Pencatatan Keuangan Pribadi Berbasis Web Mobile," *Jurnal Teknologi Informatika dan Komputer*, vol. 9, no. 1, pp. 100–113, 2023, doi: 10.37012/jtik.v9i1.1447.

[7] U. U. Sufandi and D. Trihapningsari, "Pengembangan Aplikasi Mobile SITTA Universitas Terbuka Berbasis Android," *INSERT: Information System …*, vol. 3, no. 1, pp. 28–41, 2022.

[8] F. Riana, S. Hidayat, A. Ikhsan, R. Makbul, F. Satrya, and F. Kusumah, "Aplikasi Augmented Reality Pengenalan Tanaman Obat Keluarga (TOGA) Berbasis Android," *Krea-TIF: Jurnal Teknik Informatika*, vol. 10, no. 2, pp. 68–78, 2022, doi: 10.32832/krea-tif.v10i2.8510.

[9] A. P. Nanda, "Implementasi Web Mobile Sebagai Media Informasi," *SEAT: Journal Of Software Engineering and Technology*, vol. 2, no. 2, pp. 1–7, 2022.

[10] S. N. Kamal and A. A. Ibrahim, "3D Model Visualization Function for Responsive Web Design," *Iraqi Journal for Computer Science and Mathematics*, vol. 4, no. 4, pp. 76–91, 2023, doi: 10.52866/ijcsm.2023.04.04.007.

[11] F. Ristanto, T. W. Astuti, D. Handoko, A. Syarifuddin, A. P. Nanda, and F. A. Phang, "Mobile Web Implementation As Media Information System in Margodadi Village," *Asia Information System Journal*, vol. 1, no. 2, pp. 74–79, 2023, doi: 10.24042/aisj.v1i2.15775.

[12] D. Sukmana and S. Sugiarti, "Prototype Penerapan Hasil Kombinasi Kriptografi Diffie-Hellman, Message-Digest 5 Dan Rivest Chiper 4 Pada Layanan Pesan Singkat Smartphone Android," *JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)*, vol. 7, no. 3, pp. 926–934, 2022, doi: 10.29100/jipi.v7i3.2881.

[13] N. Rahanra, D. Erlianti, Rissa Megavitry, D. J. A. Butarbutar, and Z. HB, "Design and Development of

Android-Based Traditional Indonesian Clothing Image Guessing Game," *Jurnal Teknik Informatika (Jutif)*, vol. 4, no. 1, pp. 199–203, 2023, doi: 10.52436/1.jutif.2023.4.1.377.

[14] C. Moro, Z. Štromberga, A. Raikos, and A. Stirling, "The effectiveness of virtual and augmented reality in health sciences and medical anatomy," *Anat Sci Educ*, vol. 10, no. 6, pp. 549–559, Nov. 2017, doi: 10.1002/ase.1696.

[15] Y. Kim, "Regional thickness of facial skin and superficial fat: Application to the minimally invasive procedures," *Clinical Anatomy*, vol. 32, no. 8, pp. 1008–1018, Nov. 2019, doi: 10.1002/ca.23331.

[16] H. Petersson, D. Sinkvist, C. Wang, and Ö. Smedby, "Web-based interactive 3D visualization as a tool for improved anatomy learning," *Anat Sci Educ*, vol. 2, no. 2, pp. 61–68, Mar. 2009, doi: 10.1002/ase.76.

[17] P. Dama Ramadhan, A. Triayudi, and R. Tamara Aldisa, "Animasi Sinematik Dinosaurus Secara 3D Menggunakan Blender dengan Metode Pose to Pose," *KLIK: Kajian Ilmiah Informatika dan Komputer*, vol. 3, no. 6, pp. 1100–1107, 2023, doi: 10.30865/klik.v3i6.881.

[18] T. P. Utomo, M. I. Prasetya, and M. A. Murtadho, "Aplikasi Pengingat Sholat Dan Pencarian Masjid Berbasis Android," *Jurnal Sistem Informasi dan Informatika (Simika)*, vol. 5, no. 1, pp. 10–18, 2022, doi: 10.47080/simika.v5i1.1409.

[19] M. Muthmainnah and D. B. Tabriawan, "Prototipe Alat Ukur Detak Jantung Menggunakan Sensor MAX30102 Berbasis Internet of Things (IoT) ESP8266 dan Blynk," *JISKA (Jurnal Informatika Sunan Kalijaga)*, vol. 7, no. 3, pp. 163–176, 2022, doi: 10.14421/jiska.2022.7.3.163-176.

[20] S. Steeger, D. Atzberger, W. Scheibel, and J. Döllner, *Instanced Rendering of Parameterized 3D Glyphs with Adaptive Level-of-Detail using three.js*, vol. 1, no. 1. Association for Computing Machinery, 2024. doi: 10.1145/3665318.3677171.

[21] A. Aristoteles, A. Jasmine, Y. T. Utami, and F. R. Lumbanraja, "Design of Virtual Map Building Using Unity 3D with MDLC Method," *International Journal of Electronics and Communications Systems*, vol. 3, no. 1, p. 21, 2023, doi: 10.24042/ijecs.v3i1.17196.

[22] L. Franke and D. Haehn, "Modern scientific visualizations on the web," *Informatics*, vol. 7, no. 4, 2020, doi: 10.3390/INFORMATICS7040037.

[23] C. Wuu, "Multiface: A Dataset for Neural Face Rendering," no. 1, 2022.

[24] Z. Yan, W. F. Low, Y. Chen, and G. H. Lee, "Multi-Scale 3D Gaussian Splatting for Anti-Aliased Rendering," pp. 20923–20931, 2023, doi: 10.1109/CVPR52733.2024.01977.

[25] H. Mustafidah, A. Imantoyo, and S. Suwarsito, "Pengembangan Aplikasi Uji-t Satu Sampel Berbasis Web," *JUITA: Jurnal Informatika*, vol. 8, no. 2, p. 245, Nov. 2020, doi: 10.30595/juita.v8i2.8786.

[26] N. Setiyawati and V. A. O. P. Warisman, "Integrasi Framework Kivy dan Webix pada Pembangunan Framework Mobile Web Easy Development System," *JUITA: Jurnal Informatika*, vol. 8, no. 2, p. 253, Nov. 2020, doi: 10.30595/juita.v8i2.7464.