iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 11 **Issue:** IV **Month of publication:** April 2023

DOI: https://doi.org/10.22214/ijraset.2023.50526

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 11 Issue IV Apr 2023- Available at www.ijraset.com

Troid: A Real-Time Multiplayer FPS Game

Kaartik Nayak[1], Aman Singh[2], Omkar Narayankar[3], Prof. Shashikant Mahajan[4]
[1, 2, 3, 4]*Department of Information Technology Vidyalankar Institute of Technology* Mumbai, India

Abstract: Troid is a real-time multiplayer first-person shooter (FPS) game designed to be played on a web browser without the need for a high-end PC or dedicated graphics card. The game allows two players to join a room and engage in a fast-paced, action-packed gameplay experience. The purpose of Troid is to provide a high-quality FPS game that can be enjoyed on any device with a modern web browser and an internet connection, regardless of hardware specifications. Unlike most FPS games that require high-end hardware to run smoothly, Troid is optimized for minimal requirements. Troid's front-end is built using a reactive web framework that facilitates the creation of fast and interactive user interfaces, while the game's graphics and animation are created using a 3D graphics library.
Real-time communication between players is facilitated using a reliable real-time communication library, and an in-memory data store is utilized to store player coordinates and other in-game data. Overall, Troid exemplifies the potential of creating high-quality multiplayer games that can be played on any device with a modern web browser and an internet connection, without compromising on the gameplay experience.

I. INTRODUCTION

First-person shooter (FPS) [1] games have been a popular genre among video game enthusiasts for several decades. However, most FPS games require high-end hardware and dedicated graphics cards to run smoothly, which can be a significant barrier to entry for some players. Moreover, many popular FPS games are not available on web browsers, which further limits accessibility.

To address these issues, we present Troid, a real-time multiplayer FPS game that can be played on any device with a modern web browser and an internet connection. Troid is designed to be highly accessible, allowing players to engage in a thrilling gameplay experience without requiring a high-end PC or dedicated graphics card.

In this research paper, we describe the development and implementation of Troid, focusing on the technical aspects of the game's design, architecture, and implementation. We begin by providing an overview of the related work in the field of FPS games, including a discussion of the key challenges and limitations.

II. LITERATURE REVIEW

There are several platforms for first-person shooter (FPS) games, ranging from console-based games to desktop-based games. Some of the most notable ones are listed below:

1) .ev.io [2] is a multiplayer, web-based FPS game with various weapons and game modes, gaining popularity for its simplicity and accessibility.
2) Valorant [3] is a desktop-based FPS game with competitive gameplay experience and a dedicated fanbase in esports.
3) CS GO [4] is a desktop-based FPS game with tactical gameplay and dynamic abilities, becoming a favorite among gamers and esports enthusiasts.
4) COD [5] is a long-running series of desktop-based FPS games with single-player and multiplayer modes, offering a variety of settings and scenarios.
5) Halo [6] is a console-based FPS game with a rich single-player campaign and online multiplayer modes, known for its iconic weapons and gameplay mechanics.

These games have complex game mechanics, sophisticated graphics, and intricate server-client architectures. The advantage of these established games lies in their long history and dedicated fanbase, realistic graphics and sound effects, and robust matchmaking systems, which can make it easy for players to find opponents and compete against others online. However, Troid has the advantage of being a browser-based game that can be played on any device with an internet connection, without requiring high-end hardware or expensive gaming consoles.

This accessibility can be a major draw for casual gamers and those looking for a quick and easy gaming experience. Troid also has the potential to innovate in terms of gameplay mechanics and game modes, offering a unique experience that cannot be found in other FPS games.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 11 Issue IV Apr 2023- Available at www.ijraset.com

III. PROPOSED SYSTEM

The proposed system is a proof-of-concept prototype for a browser-based FPS game named Troid. The system comprises of several modules that work together to enable players to create and join game rooms, engage in gameplay, and view the game's end results. The prototype shows a complete overview of how the system will work. The objective of this section is to provide detailed information about our prototype.

The modules in our as follows.

1) Create/Join Room
2) Game Start
3) Game Play
4) Game End
5) End Game Screen

Each module will be implemented with the goal of ensuring smooth and uninterrupted gameplay, with minimal lag or downtime. The lobby management module will allow the room creator to manage the players and game settings. The game initialization and start module will initiate the game and load the game environment. The core module of the system is the game play module, which allows players to move and shoot within the game environment. Once the game is over, the end game screen module displays the game results. The implementation of the proposed system involves the integration of various technologies such as Svelte.js [7] for the front-end and Node.js [8] for the back-end, with Redis [9] used as the database for storing player coordinates in-game. Figure 1 shows the system architecture as explained above.

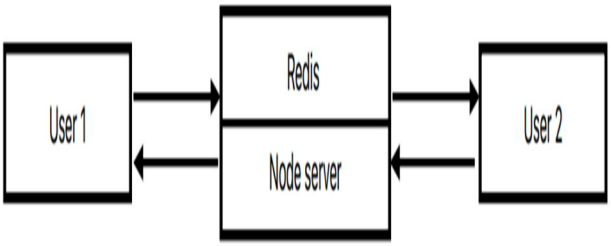


Fig. 1: Game Architecture

IV. IMPLEMENTATION

In this section, we will describe the implementation of our first-person shooter (FPS) game application. Our goal in developing this application was to create a fun and engaging gaming experience that could be played online by users across the world. To achieve this, we designed a client-server architecture that enables real-time communication between players and implemented a range of features such as player movement, shooting, and room creation/joining.

A. Designing The System Architecture

The system architecture of Troid is designed to be scalable and efficient, with the backend implemented using a client-server architecture. Troid is a client-server application where the game logic runs on the server while the game graphics run on the client-side. The client-server communication is facilitated using the Socket.IO library, which provides real-time bidirectional communication between the client and server. On the server side, Troid uses Node.js with the Fastify web framework to create a lightweight and scalable server. The server is responsible for maintaining the game state, handling room creation, and joining, and sending game updates to the clients. On the client side, Troid uses a combination of Svelte for the user interface and Three.js [10] for the game graphics. The client communicates with the server using Socket.IO to receive game updates and send player input as illustrated in Figure 2.

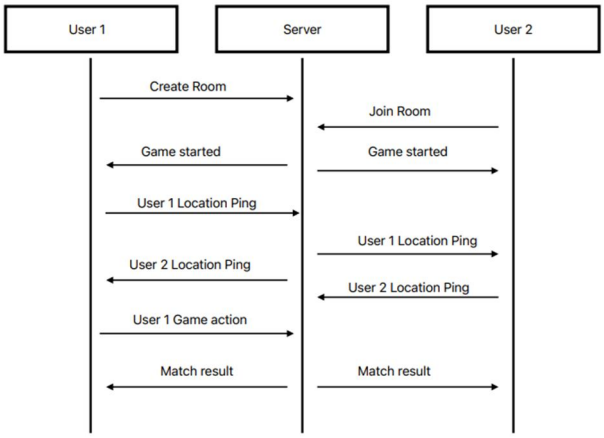


Fig. 2: User Interaction with the server

B. Creating A Backend

The backend handles the game logic, managing player data, and facilitating communication between players. The backend was developed with Fastify [11], a fast and low-overhead web framework for Node.js. Socket.IO [12] has also been used in Troid which is a real-time communication library that enables bidirectional communication between client and server.Adding a load balancer [13] in the backend can greatly enhance the stability and scalability of the game. By distributing the traffic across multiple servers, it becomes easier to handle high traffic loads and ensure that the game remains responsive and available to users. To implement a load balancer, a dedicated server is set up to act as the entry point for incoming traffic. The load balancer then distributes incoming requests to multiple servers, using algorithms such as round-robin, least connections, or IP hash to ensure a fair distribution of traffic. This setup also allows for easy addition or removal of servers as traffic demands change. Implementing a load balancer in the backend can greatly enhance the performance of the game, providing a better experience for users and allowing for future scalability as the game grows in popularity.

C. Creating Player And Map Models

In Troid, player models are designed using Blender [14], a popular 3D modelling software. These models are then exported in a format that can be easily used in the game engine. Map models, on the other hand, are designed to represent the game environment. These models are also created using Blender. The map models are designed to be both visually appealing and functional for gameplay purposes. The models created using Blender are exported as files and then loaded into the game engine.

D. Integrating The Backend With The Frontend UI

The frontend of Troid is responsible for handling the user input, displaying the game state, and updating the game visuals. It communicates with the backend server using Web Sockets to exchange game data and events in real-time. Three.js is a JavaScript library that supplies a set of tools and APIs for creating 3D graphics on the web. It allows developers to create and manipulate 3D objects, add textures and materials to them, and apply lighting and shading effects. Three.js is used in Troid to render the game environment and the player models.

E. Implementing Room Creation And Joining Logic

When a user wants to create a new room, they fill out a form with their desired room name and their own name. This information is sent to the backend server, which creates a new room with the given name and adds the user as the first player in the room. Figure 3 demonstrates the room logic. The room is then added to a list of available rooms that other players can join. When a player wants to join a room, they enter the room name and their own name into a form. This information is sent to the backend server, which checks if the requested room exists and has a free slot for another player. If the room is available, the server adds the player to the room and sends a notification to both players that a new player has joined. To prevent more than two players from joining a room, the server checks the number of players in the room before allowing a new player to join. If there are already two players in the room, the server rejects the request to join the room and notifies the user.

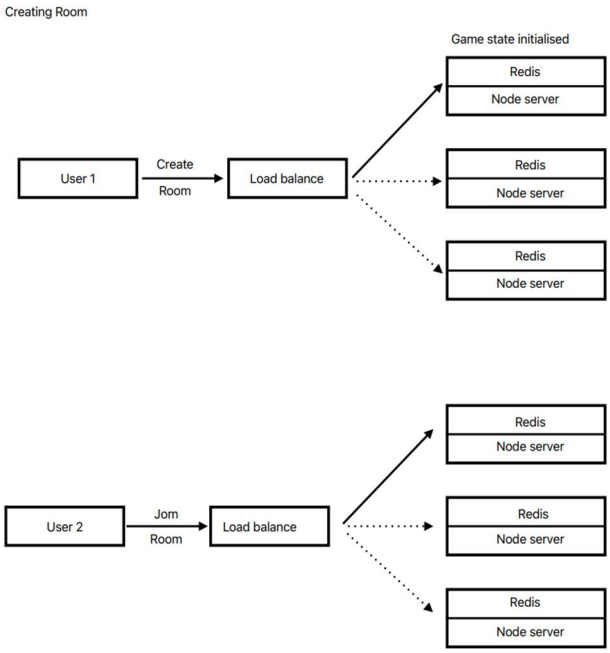


Fig. 3: User Join/Create Room

F. Implementing Player Movement And Shooting

Once both players have joined the room, a 5-second timer is set to synchronize both users. After the timer runs out, the game starts with each player's character placed at opposite ends of the map. During the game, each player can move their character and shoot at the other player. The server updates the positions of the players and the bullets in real-time and sends this information to the clients. The clients then render the game's state on their screens. When one player's health reaches zero, the game ends. The server sends a message to both clients with the game end information, and the clients then display the end game screen with the winner's name and the option to play again or leave the room. Overall, the game start and end logic are critical to ensure that both clients are in sync and receive the same information about the game's state.

G. Testing And Deployment Of The System

Once the implementation is complete, the next step is to test the system thoroughly to ensure that it meets the requirements and specifications. During the testing phase, we will perform various types of testing. Manual testing will also be performed to simulate user interactions and ensure that the game is intuitive and enjoyable to play. This includes testing the user interface, game mechanics, and player controls. Additionally, stress testing will be performed to ensure that the game can handle many concurrent players without crashing or becoming unresponsive. After testing, the application is ready for deployment. The deployment process involves uploading the application to a hosting platform, configuring the necessary settings, and ensuring that the application is available to the end-users.

V. RESULTS

The proposed system, Troid, was successfully developed as a proof-of-concept prototype for a two-player online game. The game allows players to create and join rooms, move their players around the game map, shoot at each other, and ultimately end the game with a clear winner. The implementation involved using various technologies such as Blender for designing player and map models, Svelte and Three.js for the front-end, and Fastify and Socket.IO for the backend. To ensure the system was functional and efficient, thorough testing was conducted at every stage of development.

The resultant developed system successfully demonstrated the feasibility of an online multiplayer game with essential features, and the results of the testing proved its efficiency and scalability. Future improvements could include adding more players to a single game room, introducing new game maps and models, and implementing more complex game mechanics. Overall, the developed system can serve as a starting point for creating more complex and engaging online multiplayer games.

VI. FUTURE SCOPE

The FPS game application presented in this research paper has potential for several future enhancements that could make it an even more engaging and enjoyable game for players. Some of these enhancements include:

1) *Improved AI For Enemies*: The current AI for the enemies in the game is simple and could be improved to make the gameplay more challenging and engaging. By implementing more complex and realistic AI, the enemies could react to player actions and create a more dynamic and challenging gaming experience.
2) *Multiplayer Support:* The game currently supports real-time communication between the client and server, which could be extended to support multiplayer gameplay. This would enable more than two players to play and compete against each other in real-time, adding a new level of excitement to the game.
3) *Virtual Reality Support:* The game could be adapted to work with virtual reality devices, which would provide a more immersive gaming experience. By integrating with virtual reality technology, the player would be able to feel like they are truly inside the game, creating a more intense and immersive experience.
4) *Mobile Support:* The game could be adapted to work on mobile devices, which would make it more accessible to a wider audience. By making the game available on mobile platforms, players would be able to play the game anytime, anywhere, and on any device.
5) *Integrating Sound:* The addition of head-related transfer function (HRTF) [15] support would enable players to have a more realistic and immersive sound experience. By simulating the way sound behaves in the real world, players would be able to hear the direction and distance of sounds in the game, making it easier to locate enemies and other game elements.

VII. CONCLUSION

Web-based game development provides numerous advantages over traditional game development, such as accessibility, easy maintenance, and scalability. It eliminates the need for users to install large game files or purchase expensive gaming consoles, enabling anyone with a web browser to access the game. The ease of maintenance and updates also makes it simpler for developers to fix bugs, add features, and enhance the game's overall performance. This project has highlighted the importance of proper planning and implementation of the game development process. The use of industry-standard tools such as Blender, Node.js, and Socket.io has enabled the team to develop the game efficiently and effectively. The implementation of load balancers in the backend architecture has also helped improve the game's stability and scalability. Adding the game’s functionalities, including the player movement, shooting, room creation, joining, and game start and end logic, demonstrates the feasibility of creating an immersive gaming experience in a web-based environment.

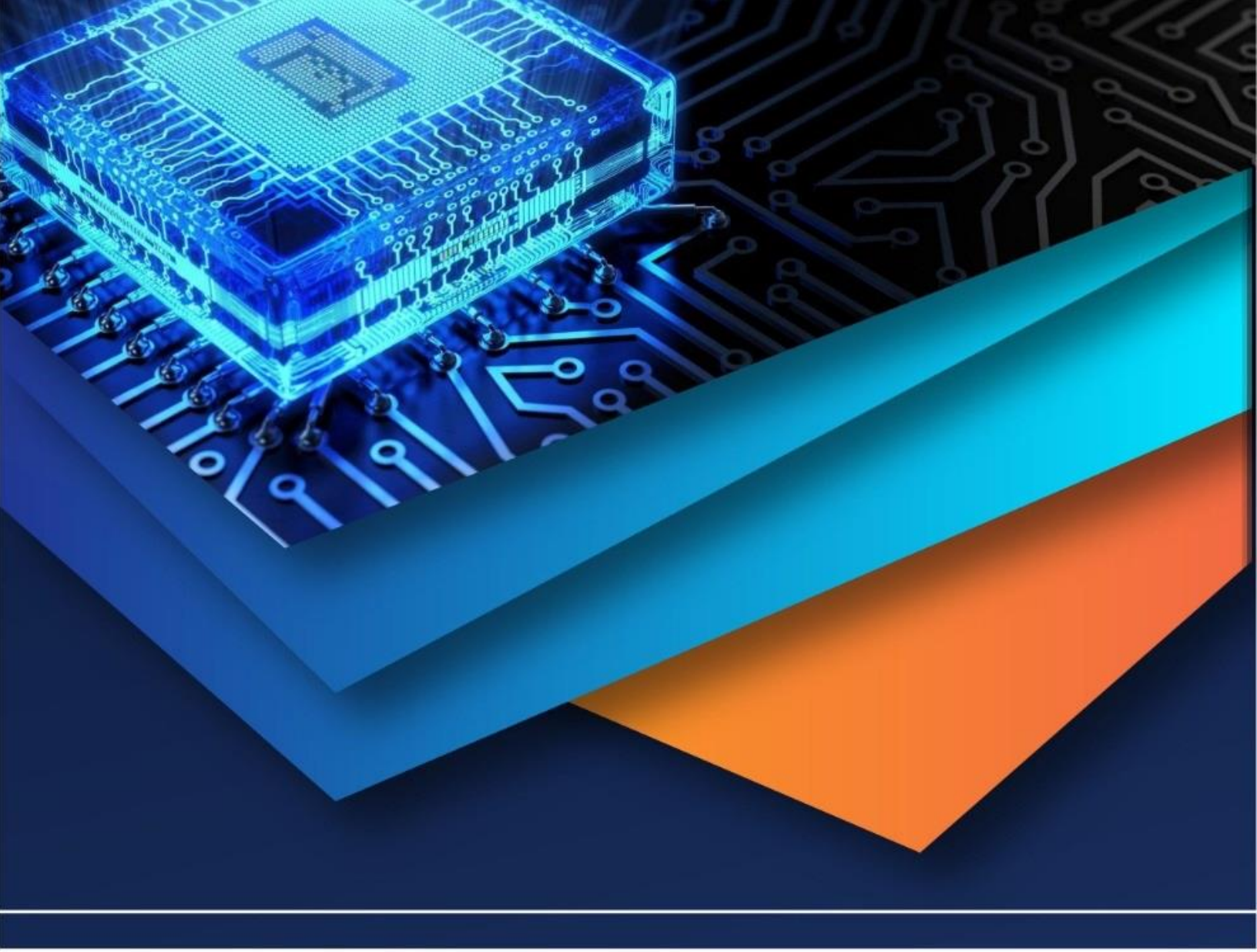
The future enhancements discussed, such as improved AI for enemies, multiplayer support, virtual reality support, mobile app development, and sound HTRF integration, present exciting opportunities for further development of the game and the underlying technology.

Overall, this project serves as a testament to the potential of web-based game development and its ability to deliver engaging and immersive gaming experiences. As web technologies advance, more advanced and sophisticated games are expected to be developed in the future, further blurring the line between web-based and traditional gaming platforms.

REFERENCES

[1] "First-person shooter," [Online]. Available: https://en.wikipedia.org/wiki/First-person_shooter.
[2] "Play ev.io - the play to earn blockchain FPS," [Online]. Available: https://ev.io.
[3] "VALORANT: a 5v5 character-based tactical FPS," [Online]. Available: https://playvalorant.com/.
[4] "Counter-Strike: Global Offensive," [Online]. Available: https://store.steampowered.com/app/730/C ounterStrike_Global_Of fensi ve/.
[5] "Call of Duty® | Best-Selling Video Game Franchise," [Online]. Available: https://www.callofduty.com.
[6] "Halo," [Online]. Available: https://www.halowaypoint.com/.
[7] "Svelte • Cybernetically enhanced web apps," [Online]. Available: https://svelte.dev.
[8] "Node.js," [Online]. Available: https://nodejs.org/en.
[9] "Redis," [Online]. Available: https://redis.io/.
[10] "Three.js – JavaScript 3D Library," [Online]. Available: https://threejs.org/.
[11] "Fastify, Fast and low overhead web framework, for Node.js," [Online]. Available: https://www.fastify.io/.
[12] "Socket.IO," [Online]. Available: https://socket.io/.
[13] "Load balancing (computing)," [Online]. Available: https://en.wikipedia.org/wiki/Load_balancing_(computing) .

[14] "blender.org - Home of the Blender project," [Online]. Available: https://www.blender.org/.
[15] "Head-related transfer function," [Online]. Available: https://en.wikipedia.org/wiki/Head-related_transfer_function.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY