Shape Transform: Enhancing Spatial Reasoning Through an Interactive 3D Puzzle Game

Finley Neilson
fnei425@aucklanduni.ac.nz
The University of Auckland
Auckland, New Zealand

ABSTRACT

Spatial reasoning skills are critical predictors of academic and professional success within Science, Technology, Engineering, and Mathematics (STEM) fields. However, traditional diagnostic assessment methods, such as the Vandenberg Mental Rotations Test, introduce severe cognitive strain and lack long-term engagement loops. This report introduces *Shape Transform*, a web-based, interactive 3D puzzle game designed by group Blockarang to convert mental rotation tasks into a gamified, responsive spatial training environment. Developed utilizing the MongoDB, Express, React, and Node.js (MERN) tech stack, the platform serves procedurally generated 3D block puzzles that players manipulate to match target solutions. This report details the user interface design paradigm, procedural puzzle generation, client-side rendering constraints, architectural trade-offs, and a strict dual-layer testing pipeline. Finally, we reflect on agile project management outcomes, effective software engineering tooling, and future works.

CCS CONCEPTS

• **Web Project → Software Engineering**; • **Educational Game, Spatial Reasoning → Computing Education**.

KEYWORDS

Computing Education, Computer Graphics, MERN stack, Software Development Lifecycle, Spatial Skills, Procedural Generation

ACM Reference Format:
Finley Neilson. 2026. Shape Transform: Enhancing Spatial Reasoning Through an Interactive 3D Puzzle Game. In . ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Spatial skills and the ability to mentally rotate 3D objects are strong predictors of how easily people grasp technical STEM concepts [4]. From evaluating structural designs to tracing paths through complex software source code, strong visual reasoning makes handling intricate systems intuitive [3]. However, while these cognitive habits are highly useful in daily life and technical careers, they are rarely trained outside of rigid, dry academic diagnostics. Accessible, engaging tools that allow anyone to practice these spatial rotation tasks casually without heavy mental fatigue remain scarce.

Recognizing this gap, our group, *Blockarang*—composed partly of computer science students completing Honours pathways in Computer Graphics—sought to build an accessible system targeting cognitive spatial manipulation.

Our investigation focused on bringing these 3D rotation mechanics into a responsive web environment. While our group's background in computer graphics streamlined the initial design phase, implementing interactive 3D shapes within a browser runtime introduced distinct software engineering hurdles. Balancing security and architectural choices with real-time rendering performance and making this all function within an intuitive UI was a serious challenge.

Traditional instruments leveraged to assess or improve spatial faculties are historically optimized for static diagnostic metrics rather than repetitive interactive engagement. The core objective of our project was to build *Shape Transform*: a simple, clean, and highly responsive 3D web puzzle game. The game requires users to manipulate and rotate a dynamic block tree in virtual space to exactly align with a given solution rotation. By shifting spatial training from a dry testing methodology to an intuitive, gamified framework, *Shape Transform* preserves educational value while trying to significantly increasing user engagement.

2 RELATED WORK

In researching what has been done before, we looked into established psychological benchmarks as well as modern interactive titles. The standard metric for auditing mental rotation capability remains the Vandenberg Mental Rotations Test [5]. In a standard Vandenberg configuration, a participant is presented with a static base block figure on the left, flanked by four alternative target choices on the right [1, 2]. The player must accurately deduce via mental rotation which options represent true physical matches rather than mirror-symmetric distortions or structural rotations [5]. While psychometrically rigorous, the exercise is kind of boring and has limited replayability.

Consequently, we analyzed contemporary cognitive-centric digital games to observe how developer networks successfully introduce high-engagement loops. Our competitive matrix reviewed multiple market examples, evaluating them across structural cleanliness, response latency, and user interface fluidness. Our exploration highlighted a specific market gap: while several puzzle games prioritize high-fidelity UI or clean, minimal aesthetics, such as the

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

geometry-based shading tool "Boolean Method"[1] or the casual web-based "Rotate Game"[2], only a single examined title strictly integrated 3D block rotation mechanics as its primary driver: "BlockAppend"[3]. Furthermore, this identified title, along with foundational browser exercises like "3D Cube"[4], suffered from restricted feature sets, lacking deep progression structures, modern responsiveness, or centralized tracking systems.

Most critically, existing frameworks failed to optimize two foundational operational vectors required for prolonged consumer engagement: minimal interface noise and high-fidelity runtime responsiveness. Using these observations as our baseline design principles, our group established the functional foundation for *Shape Transform*.

3 DESIGN

The design of *Shape Transform* focuses on clean state separation—ensuring that real-time 3D rendering performance on the client side is completely uninhibited by backend database persistence transactions.

3.1 Software Architecture & Data Schema

The application leverages a decoupled web framework using an asynchronous backend coupled to a document-oriented database cluster. Storage abstractions are managed within a MongoDB Atlas engine, isolated cleanly into three primary operational collections, as illustrated in Fig. 1:

- **gamerecords**: Tracks continuous, active user session data, saving game progression states securely to ensure the server can maintain state persistence across client reloads.
- **puzzles**: Houses full puzzle definitions, structural matrices, nested 3D block-tree models, and exact spatial coordinates. At present, the persistent database contains 551 distinct, validated puzzle documents.
- **users**: Handles salted authentication records, account tracking strings, and real-time leaderboard performance stats (e.g., `bestTime`).

3.2 User Interface Paradigm

The application's interface flow was heavily prototyped in Figma prior to programmatic assembly. Interface planning started with comprehensive task-flow blueprints mapping a newcomer's transition from initial onboarding to real-time gameplay loops.

To prevent cognitive clutter, the aesthetic design adheres to a deep purple back-canvas, paired with clean white high-contrast text and luminous yellow-green accent lines to isolate target interactables. This palette creates strong contrast lines while preserving user comfort during long gaming intervals. Every interface frame incorporates fully responsive fluid-grid CSS properties, maintaining seamless geometric aspect ratios when adapting between desktop viewports and mobile layouts.

[1]https://boolean.method.ac/
[2]https://firetotemgames.itch.io/rotate-game
[3]https://www.reddit.com/r/Unity3D/comments/zydj6u/
[4]https://www.playdosgames.com/online/3d-cube/

3.3 Client versus Server Rendering Trade-offs

A major architectural choice centered on whether to render solution target puzzle states on the client browser runtime or pre-render configurations directly on the server. We conducted a comprehensive review comparing system security vulnerabilities against processing overhead performance:

- **Server-Side Rendering:** Yields higher security protocols by keeping target rotation paths hidden from client inspect windows, but incurs a 1-second resource penalty to generate and serve images dynamically.
- **Client-Side Rendering:** Offers instantaneous response metrics, smooth visual interpolation, and reduced hosting overhead, but exposes matrix parameters to localized web inspection tools.

Ultimately, the project selected real-time client rendering driven by custom web coordinate engines to support low-latency interaction loops. To ensure perfect rotation precision free from gimbal lock errors, all spatial rotations are handled using mathematical quaternions on the client side. Conveniently, the non-linear, four-dimensional nature of quaternion coordinates introduces a layer of natural data obfuscation, inherently complicating client-side memory tampering or raw state manipulation by end users.

4 IMPLEMENTATION

The gameplay environment relies on procedurally generated assets grouped into three difficulty tiers: Easy (solvable within 1–3 transform steps), Medium (solvable within 3–6 steps), and Hard (requiring 4–7 steps and integrating 45-degree rotations alongside standard 90-degree operations).

4.1 Procedural Generation and Solvability

To build reliable puzzle streams, game algorithms run a reverse-generation sequence. Starting from a structurally sound base block configuration, the generation script applies a sequence of valid spatial operations along the x, y, or z axes to calculate the final target goal orientation. This process mathematically guarantees that every puzzle served is solvable.

The algorithm discards generated instances where the initial start shape and final target rotation match perfectly. For Hard puzzles, the engine restricts generation to exactly one 45-degree transformation step, with the remaining steps confined to 90-degree turns. This approach maintains high puzzle complexity without causing extreme player frustration. At runtime, the application filters database fetches using a specific string index prefix (`gen-*`), ensuring legacy test schemas or malformed manual files are never served to active players.

4.2 The Symmetry Resolution Problem

During early user testing, our group identified a critical visual logic error regarding symmetrical blocks. If a generated shape featured geometric symmetry, a player could manipulate the active block-tree into a position that looked identical to the target solution, yet the validation engine would return a failure status. This occurred because the internal coordinate tracking array expected an explicit

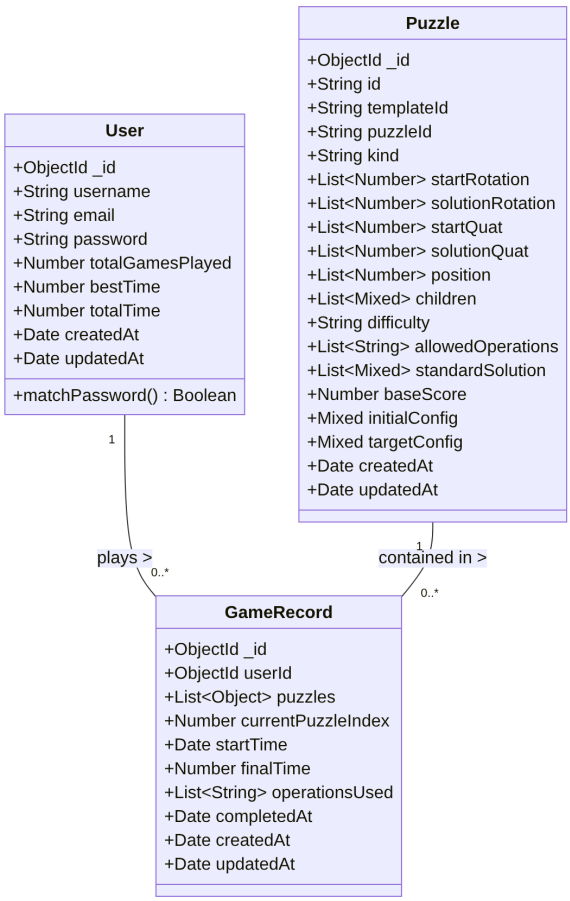

Figure 1: Data schema configuration and document collection layout for MongoDB Atlas.

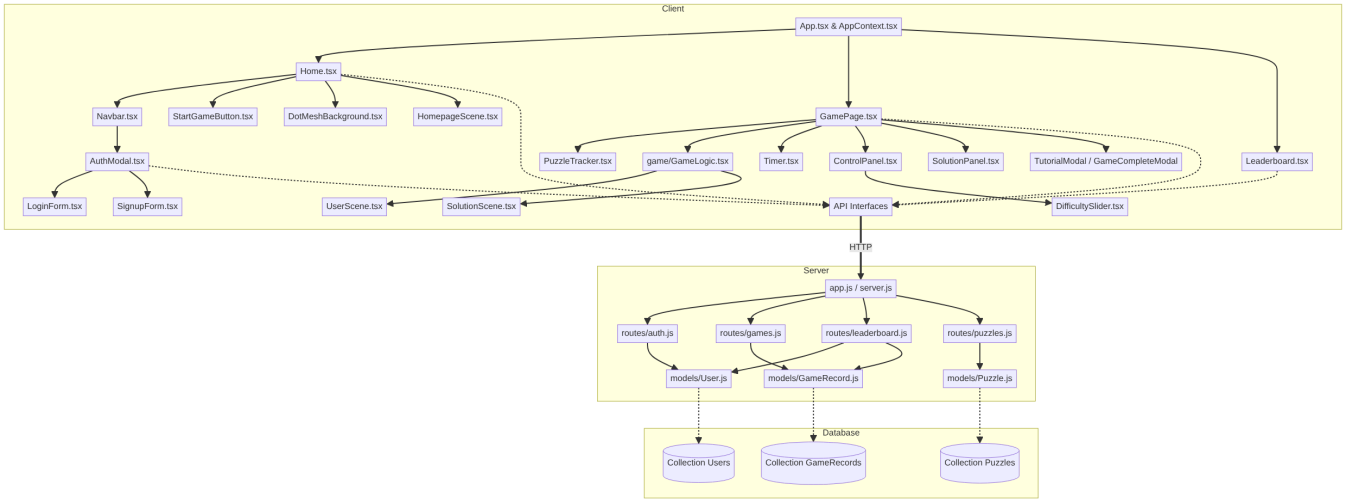

Figure 2: UML class diagram illustrating application architecture and system objects.

rotation index, failing to recognize that different quaternion values could produce identical visual states.

Building a general real-time symmetry check within procedural generation loops proved computationally expensive for our timeline. Furthermore, symmetrical puzzles are typically less interesting because they reduce the unique steps required to solve them.

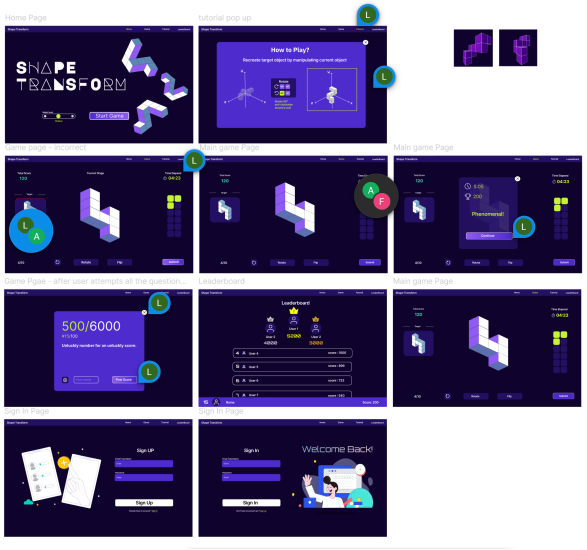

Figure 3: Interactive viewport wireframes and UI task-flow mapping designed within Figma.

Consequently, we resolved this issue by removing symmetrical configurations from our generation models entirely.

4.3 Gamification Framework and Mechanics

To sustain high engagement, *Shape Transform* features custom gamification metrics. Harder difficulties utilize a negative-offset starting timer, granting users more baseline allocation to manage complex configurations before time penalties apply. A fresh player profile automatically triggers an overlay onboarding modal upon launch, introducing interaction controls before gameplay begins. When a puzzle is solved, the system updates a persistent global leaderboard via automated REST endpoints, encouraging competition across both accuracy and time efficiency vectors.

5 TESTING

To guarantee runtime resilience, we built a dual-tier testing infrastructure consisting of isolated unit tests and broader extended integration suites.

5.1 Unit Testing Framework

Unit tests prioritize fast execution and strict isolation. Built around Node.js's native test runner alongside the `Supertest` HTTP asset library, this framework mocks all primary database models to evaluate backend route handling and payload compliance without database dependency overhead. These checks cover:

- **System Health Interfaces:** Verifies that `GET /` calls trigger an operational 200 OK status message.
- **Authentication Workflows:** Tests that signup processes reject missing fields or duplicate usernames and email registrations.

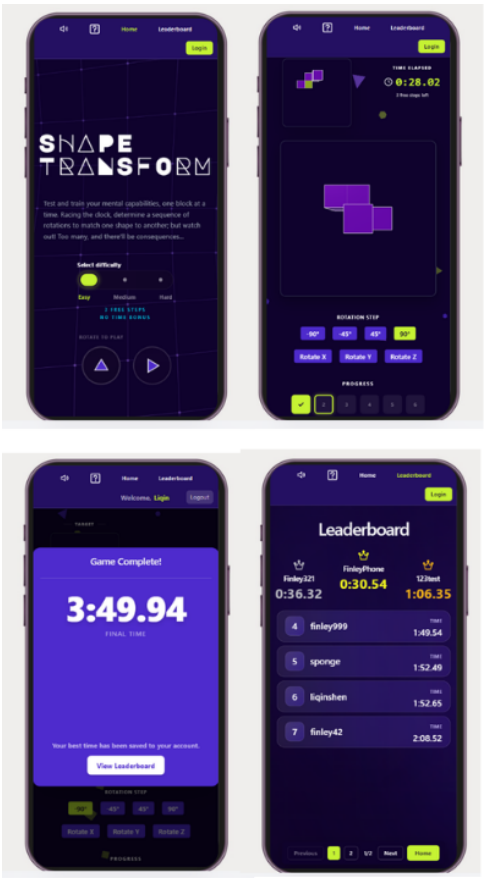

Figure 4: Responsive mobile user interface grid scaling and responsive viewport layout.

- **Game Verification Routers:** Validates that missing game sessions throw explicit 404 responses, and ensures leaderboard scoring arrays return records in descending sorted order.
- **Puzzle Logic Audits:** Evaluates target alignment state detectors to ensure accurate pass/fail checks during user rotation inputs.

5.2 Extended Integration Testing

Extended suites test full end-to-end system operations. Rather than mocking dependencies, these suites instantiate a clean, live database environment using `mongodb-memory-server`. This ensures our authentication layers, custom data models, and API endpoints interact seamlessly with a real database. For example, our integration tests verify that initiating a puzzle successfully writes an operational `gamerecord` document tied to that specific user ID.

5.3 CI/CD Automation

All testing suites run through automated GitHub Actions pipelines on every remote commit. The automation runtime pulls down target branch data, configures a Node.js 20 base image, executes clean

dependency maps using `npm ci`, establishes temporary JSON Web Token (JWT) environment keys, and triggers our validation suites.

To preserve pipeline resources, we configured explicit workflow concurrency filters. If a developer pushes a new commit while an older build runner is active, the system automatically cancels the legacy run. To optimize development velocity, fast unit tests run across all branch pushes, while resource-intensive integration tests are reserved for pull requests merging into the `main` branch.

6 METHODOLOGY

Project management throughout our development lifecycle followed an agile framework structured around distributed responsibilities and consistent sync cycles.

6.1 Team Coordination & Roles

At project kickoff, core roles were allocated based on individual technical strengths, while specific tasks were distributed flexibly week-to-week via an issue board to maintain workflow velocity. The development team hosted synchronized weekly status meetings to evaluate code quality, review milestone achievements, and re-align technical goals. These meetings occurred consistently throughout the project timeline, except during the initial planning phase and academic recess breaks.

6.2 Communication Channels

Team communication was divided across three primary digital tools to prevent coordination silos. Discord served as our central asynchronous chat network, utilized for daily status check-ins, multimedia asset sharing, and feature debugging outside of scheduled hours. Structural group administration, project roadmap tracking, and milestone visualizations were centralized using GitHub Projects to maintain transparent task progression. Lastly, shared document repositories were maintained via Google Docs to facilitate collaborative drafting and system design outlining.

7 TOOLING

The technical stack selected for *Shape Transform* relies on a standardized MongoDB, Express, React, and Node.js (MERN) architecture. While the group evaluated alternative backend frameworks, we chose the MERN stack due to its reliability and low complexity for team members new to full-stack application development. Within the client-side layer, the system utilizes Three.js via React Three Fiber to abstract WebGL operations, enabling hardware-accelerated 3D rendering and real-time mesh manipulations directly inside the component lifecycle.

Version control tracking was managed using Git, integrated with the Jujutsu (`jj`) version control system command line to simplify branch tracking and history operations. The complete web ecosystem is deployed live on Render.com. While effective for automated hosting, the platform introduces a minor operational drawback: the free tier introduces a server cold-start spin-up delay when pulling containers out of idle states after periods of inactivity.

Generative Artificial Intelligence tools were utilized throughout the development lifecycle, though use cases varied significantly across team members. AI was exclusively applied to speed up routine tasks, write boilerplate code, and troubleshoot backend integration routes. Crucially, AI tools were barred from creative tasks, including game flow ideation, structural gameplay mechanics, and core aesthetic design choices, ensuring the intellectual foundation remained fully original.

8 DISCUSSION

Evaluating our final product against our initial development requirements shows that all critical system features were successfully delivered.

8.1 Achieved Deliverables

Our development velocity allowed the team to deliver all core "Must-Have" requirements, establishing a solid foundation for the fundamental gameplay loop. The onboarding tutorial system is fully operational, featuring an instruction modal that prevents initial friction by introducing new users to game controls and rules automatically upon launch. The core interaction engine updates smoothly, allowing real-time spatial rotation and manipulation of 3D block structures inspired by cube composition tasks. This visual experience is delivered through a clean, responsive, and minimalist user interface designed to maximize usability.

Beyond the baseline requirements, the final implementation successfully absorbed several key "Should-Have" and "Could-Have" targets into production. To foster competition and replayability, we engineered a secure user authentication system (login/signup) backed by a real-time, centralized leaderboard tracking system that monitors user performance metrics. Time-based gameplay is fully supported via a visible countdown timer that adds structural urgency to the puzzles.

Crucially, the team successfully implemented an automated procedural puzzle generation engine originally classified as a "Nice-to-Have" stretch goal. This engine was leveraged to populate our persistent document store with 551 distinct, validated puzzle configurations spanning a multi-tiered difficulty schema. The entire application is rounded out with fluid UI transition animations to enhance overall game feel, and cross-platform responsive optimization making the entire experience fully usable on mobile viewports.

8.2 Project Limitations

Despite achieving our core goals, project time constraints forced us to deprioritize a few secondary features:

- **Contextual Hint Engine:** Designed to assist players stuck on complex configurations, this feature was cut to ensure core rotation mechanics remained stable.
- **Granular Leaderboards:** The database effectively tracks user scores dynamically, but the presentation layer does not yet separate rankings by individual difficulty tiers.
- **Sandbox Mode:** A free-form construction sandbox was excluded to focus development efforts on validating and balancing the main progression loop.

These omissions do not impact the core gameplay loop. Our development priorities ensured a stable, polished user experience over an overloaded but buggy feature set.

9 CONCLUSION

The successful deployment of *Shape Transform* demonstrates that spatial training can be effectively shifted from stressful diagnostic tests into an engaging, replayable digital game. This project highlights the importance of keeping early design goals realistic. Our initial scope included complex transformations such as shearing matrices, logical boolean block intersections, colored multi-face zones, and dice-based mechanics. While compelling on paper, early development reviews showed these features risked overcomplicating the project and missing deadlines. Focusing on a refined, polished core rotation loop allowed us to deliver a complete, high-quality application.

A critical lesson learned during this engineering lifecycle was the necessity of proactive, rigorous team communication regarding backend architecture. Because architectural patterns were not thoroughly synchronized in early collaborative meetings, a "shanty town" anti-pattern gradually emerged across the codebase as individual features were rapidly implemented in isolation. This lack of structural alignment necessitated extensive, late-stage refactoring and culminated in severe multi-file merge conflicts that disrupted development velocity. Moving forward, establishing strict architectural boundaries and collective code ownership protocols during early sprints is vital to prevent technical debt from accumulating under compressed timelines.

Future iterations of *Shape Transform* will introduce animated tutorial guides to help users visualize 3D space shifts before playing. We also plan to integrate a "Puzzle of the Day" system to drive daily user retention through shared competitive tracks. To support a more comprehensive competitive framework, the leaderboard architecture will be refactored to separate and track user rankings dynamically across individual difficulty tiers. Finally, expanding the generation engine to support varied geometric forms—such as prisms, pyramids, and non-cuboid polyhedra—alongside diverse spatial transformations will broaden the game's long-term educational impact and improve engagement.

REFERENCES

[1] Amanda F Caissie, François Vigneau, and Douglas A Bors. 2009. What does the Mental Rotation Test Measure? An Analysis of Item Difficulty and Item Characteristics. *The Open Psychology Journal* 2, 1 (2009), 94–102. https://doi.org/10.2174/1874350100902010094

[2] Alfredo Campos. 2012. Measure of the Ability to Rotate Mental Images. https://doi.org/10.1037/t14408-000

[3] Madeline Endres, Madison Fansher, Priti Shah, and Westley Weimer. 2021. To Read or to Rotate? Comparing the Effects of Technical Reading Training and Spatial Skills Training on Novice Programming Ability. In *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. ACM, Athens Greece, 754–766. https://doi.org/10.1145/3468264.3468583

[4] Sheryl A. Sorby. 2009. Educational Research in Developing 3-D Spatial Skills for Engineering Students. *International Journal of Science Education* 31, 3 (Feb. 2009), 459–480. https://doi.org/10.1080/09500690802595839

[5] Steven G Vandenberg and Allan R Kuse. 1978. Mental rotations, a group test of three-dimensional spatial visualization. *Perceptual and Motor Skills* 47, 2 (1978), 599–604. https://doi.org/10.2466/pms.1978.47.2.599