2D Online Game Using JS Phaser

COMPSCI 399 Capstone Course Project Report

15/06/2025

Name	Role(s)
Fei Lan	Project Manager, UI Designer (Apple Catcher & Marble Track)
Jervid Cheng	Level Developer – Marble Track L3, Technical Debugging, Data Collection, Game Mechanics Refinement
Sophia Zhang	Level Developer – Marble Track core gameplay logic & L0, Technical Support, Game Mechanics Refinement
Linh Ha	Level Developer – Apple Catcher L0–2, Data Collection Support
Shuxuan Huang	Level Developer – Apple Catcher L3–4 & Marble Track L2, Data Collection Support
Yiqing Cheng	Level Developer – Marble Track L1 & Apple Catcher Navigation Page, Data Collection Support

Executive Summary

In early childhood research, collecting meaningful and detailed data about how young children learn and think is a significant challenge. Traditional methods rely heavily on in-person observations using physical tasks, which can be time-consuming, inconsistent, and prone to missing valuable data like precise timing, retry counts, and detailed player actions. To address this, game-based assessments have emerged as a powerful solution, combining engaging digital gameplay with robust data collection capabilities.

Our project contributes to the Early Learning Lab, an online platform developed by the University of Auckland's School of Psychology, by delivering Marble Track — an interactive physics-based game designed for children aged 3 to 5. Marble Track enables researchers to assess cognitive skills such as cause-and-effect understanding through engaging marble puzzles, while simultaneously recording detailed gameplay data automatically.

We designed Marble Track with modular, scalable architecture that supports smooth gameplay, reliable physics simulation, and precise data tracking across multiple levels. The system allows unique user identification, tracks attempts, movement paths, scores, and time taken, and exports structured JSON data for straightforward analysis. Through rigorous testing across devices and browsers, we ensured stable performance, intuitive interaction, and high-quality user experience tailored for young children.

Key challenges, such as adapting the scoring system from a previous project and mitigating physics simulation quirks, were met with innovative technical solutions that enhanced system reliability without sacrificing usability. Our modular design not only facilitated maintainability but also positioned the platform for future expansion.

Looking forward, we recommend completing the partially implemented Apple Catcher game to broaden cognitive skill assessments and integrating cloud-based data storage for real-time access and multi-participant studies. Additional improvements like adaptive difficulty, accessibility features, and enhanced deployment options would make the platform more inclusive and versatile. Strengthening code modularity will further ease future development and integration of new games.

In summary, Marble Track successfully delivers a sophisticated yet child-friendly tool that bridges playful engagement with rigorous data collection, offering researchers a powerful means to advance early childhood cognitive research. With planned enhancements, this platform

holds great promises for expanding the scope, accessibility, and impact of game-based assessments.

Table of Contents

Introduction

Understanding how young children develop cognitive skills such as causal reasoning is a central challenge in early childhood research. Traditional methods often rely on physical tasks and in-person observations, which can be time-consuming, inconsistently recorded, and difficult to scale. With the increasing availability of digital tools, game-based platforms now offer researchers a more efficient, engaging, and data-rich alternative for studying developmental behaviour.

This project supports that research direction by delivering Marble Track, an interactive, browser-based game designed for children aged 3 to 6. The game enables researchers to evaluate early cognitive abilities, particularly spatial reasoning and cause-effect understanding, through structured digital tasks. In addition to completing Marble Track, our team also worked on enhancing the existing Apple Catcher game to expand the range of cognitive assessments supported by the platform. These games are designed to be child-friendly and intuitive, while also capturing detailed behavioral data suitable for scientific analysis.

The project scope included the full implementation of Marble Track across four levels, the development of a unified behavioral logging and scoring system, and the partial extension of Apple Catcher to support further evaluation of attention and motor skills. We adopted an agile development process using TypeScript and the Phaser 3 framework, along with MatterJS for physics simulation. Collaboration was managed via GitHub, and feedback from the client guided iterative improvements throughout development.

The final deliverables include a stable, scalable game system that supports realistic physical interaction, consistent cross-level data recording, and one-click JSON export of structured gameplay data. The system is deployable via GitHub Pages and suitable for cross-device use. Overall, this project contributes a robust and extensible research tool for studying cognitive development in early childhood settings.

Background

Understanding cognitive development in early childhood—particularly how children build problem-solving, spatial-reasoning, and causal-reasoning skills—is critical for both educational research and practice. Traditional studies rely on in-person tasks such as puzzles or physical toys. These methods, however, are labor-intensive, limited in sample size, and difficult to standardize across settings, which hinders the generalizability and validation of findings.

The availability of digital technologies and game engines has led to a rise in the use of game-based research methods in recent years, interactive digital games calm children and bring about behaviors they would exhibit in real-life, which enhances the quality of data for research(Ullman et al., 2017). Moreover, fully digital environments can streamline data collection through automated logging, boosting precision and reducing manual errors.(Saxe & Carey, 2006; Sobel et al., 2004).

The Early Learning Lab at the University of Auckland has launched an open-access online platform that uses the JavaScript-based Phaser engine to host interactive games for child-development studies. Two games are already live and provide basic data collection, yet they leave room for richer interactions and finer-grained data.

To extend the platform's research capabilities, our team created Marbles Track, an interactive physics game for 3- to 5-year-olds. Marbles Track assesses spatial and causal reasoning while logging each child's decision patterns, reaction times, and interaction traces. Its clear structure and game mechanics address long-standing limitations—such as small sample sizes and data-standardization hurdles—offering researchers a broader, more convenient tool.

Project value

- **Structured, research-ready data**
  Data quality and reproducibility are increased by local-storage logging and one-click JSON export, which capture important behaviors (attempt count, completion time, drag paths) without requiring a back-end.
- **Natural engagement and authentic behavior**
  Realistic physics and progressive levels keep kids interested without the need for additional teaching, improving user satisfaction and data validity.
- **High-fidelity physical interaction**
  By integrating MatterJS into Phaser, the game simulates believable motion (falling, bouncing, rolling, friction, collisions), giving children a more realistic causal experience and enabling researchers to probe their intuitive physics.

- **Extensible open-source architecture**

  Marbles Track is an open-source, modular component of the Early Learning Lab platform. Its components can be extended or reused by researchers to create new experiments. With shared modules across games, the Phaser + TypeScript stack enables both standalone development and smooth deployment via GitHub Pages.

Methods & tools

- **Tech stack:** Phaser 3 (front-end game framework), MatterJS (physics engine), TypeScript (type-safe maintainable code).
- **Version control & collaboration:** Git and GitHub.
- **Development environment:** Visual Studio Code with WSL for cross-platform work; IntelliJ IDEA; GitHub workflows.
- **Data management:** Browser localStorage, JSON export—no additional server required.
- **Deployment:** Jekyll + GitHub Pages – to host the landing page and deploy our game.

Project Specification

User Requirements

The Marble Track game is designed primarily for children participating in educational experiments. Therefore, it must be intuitive, interactive, and capable of recording player behavior for later analysis by researchers. The key user requirements are as follows:

(1) Interactive Gameplay: The game must allow players to interact with marbles and tracks through intuitive drag-and-drop mechanics.

(2) Progressive Levels: The game should contain multiple levels, each introducing new gameplay mechanics to maintain engagement and support learning objectives.

(3) Player Identification: Users should be able to input a unique player ID before starting the game to associate data with individual players.

(4) Accurate Data Collection: The game must automatically record all relevant interactions, including attempts, scores, durations, and movement paths.

(5) Data Export and Reset: Researchers must be able to download player data in JSON format and reset the data storage when necessary.

Use Cases

Below are several use cases that describe typical user interactions:

(1) Use Case 1: Starting the Game

The user visits the homepage and selects the Marble Track game.

They are redirected to the Main Menu, where they input a player ID and press the Start button to begin the experiment.

(2) Use Case 2: Completing a Level

The user enters a level and interacts with marbles or tracks.

They press the “Drop” button to release a marble and attempt to guide it to the goal.

If the result is unsatisfactory, they may press “Reset” to try again.

After completion, they proceed to the next level using the “Next” button.

(3) Use Case 3: Researcher Downloads Data

After the gameplay session, a researcher returns to the Main Menu.

They click the “Download Data” button to export all player interaction records as a JSON file.

If needed, they click “Delete Data” to clear local storage for the next participant.

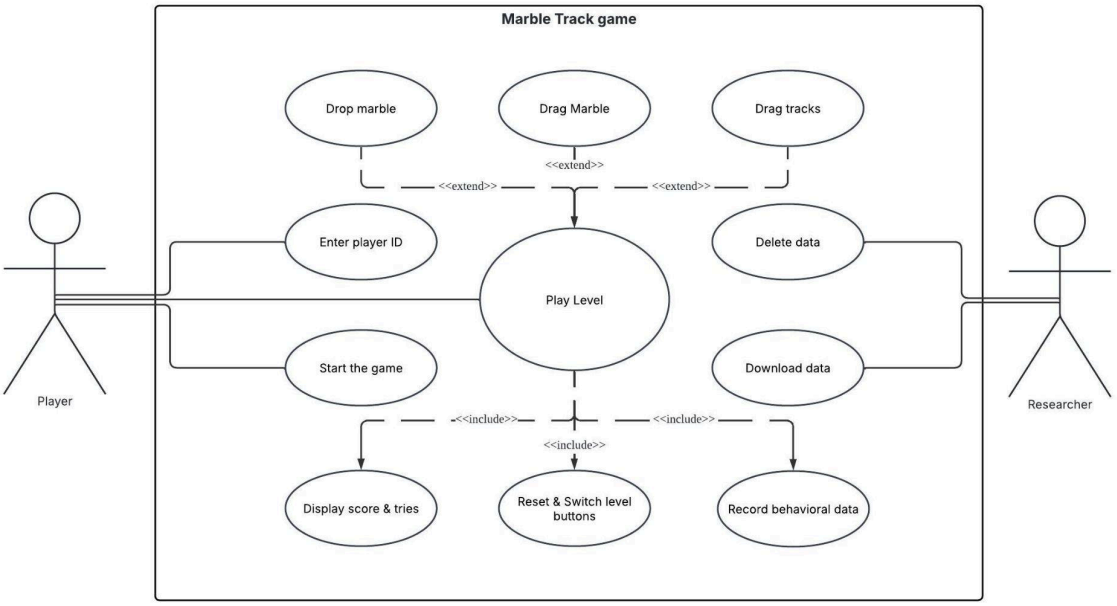


Figure 1. UML Use Case Diagram depicting user interactions and system functionalities

These use cases are further illustrated in the included UML Use Case diagram. They demonstrate how the system supports both the end-user (child player) and the experiment facilitator (researcher), ensuring a smooth and trackable gameplay experience.

Project Design

Our system is a browser-based educational physics game designed using the Phaser framework and the MatterJS physics engine. The game simulates marble movement through various obstacles, allowing players to interact by dragging marbles or tracks. It is structured for experimental use, supporting behavioral data collection and analysis.

Design Choices and Justifications

We chose Phaser for its structured Scene system and support for modular game design.

Instead of Phaser's Arcade Physics engine, we chose the more advanced MatterJS physics engine. While Arcade Physics offers basic support for object motion, it lacks features necessary for simulating realistic physical interactions. In prior implementations such as Apple Catcher, manual coding was required to simulate the complex movement of objects. In contrast, MatterJS provides built-in support for more complex dynamics including rolling, bouncing, and accurate collision detection. This transition enabled more realistic marble behavior and streamlined the implementation by reducing the need for hardcoded physics logic, thereby improving both code readability and maintainability.

Scene-Based Architecture

The game is divided into multiple Scenes: MainMenu, Level0–Level3, and GameOver. All level scenes inherit from a custom MarbleTrackScene class, which encapsulates common logic like UI layout, drag/drop handling, and score recording. This inheritance model minimizes code duplication and improves maintainability.

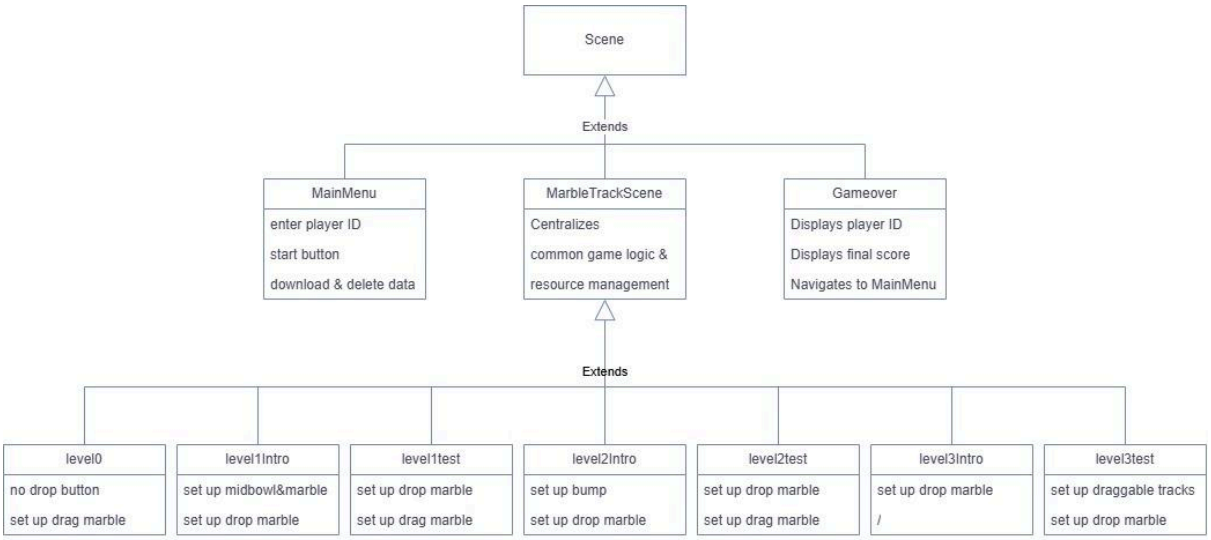


Figure 2. UML Class Diagram showing the inheritance structure of game scenes

User Interface and Data Management

We included UI elements like Drop, Reset, Next/Previous buttons, and real-time display of score and tries. Each Scene also records player behavior in localStorage and supports JSON export. This supports experimental tracking without needing a backend.

Data Flow

Scene Entry → Creates new attempt record with timestamp.
Player Action → Triggers tries, score, duration and movement path logs.
Data Storage → Each attempt with behavioral data is saved in local storage.
Download → Data from local storage can be exported as JSON format via MainMenu.

UI design

Design goals for 3- to 5-year-olds

- Vivid colors and clear layouts to spark interest.
- Graphical controls with minimal text, enabling pre-literate children to play smoothly.
- Fast, explicit feedback to maintain game pace and motivation.

We redesigned the user interface, removing the rudimentary color blocks, blank backgrounds, and absent audio found in previous games like Apple Catcher, while building upon the Early Learning Lab framework.

UI Core improvements & innovations

Visual makeover: Replaced flat color panels with high-resolution vector illustrations (game scenes, marble textures, wooden tracks, button icons). Unified the look of buttons, score bars, and headers to create a coherent brand. All screens now feature lush botanical backgrounds; rounded, large-face fonts with high contrast boost focus.

Button upgrades: Key buttons (Drop, Reset, Next) use high-contrast, large, rounded rectangles suited to gross-motor skills. Music and home buttons sit in opposing corners as icons, minimizing reading requirements.

Audio system: A unique victory tune and energetic background loops are added by the AudioManager module. Children are engaged by sound design without having to make a lot of decisions.

Level-clear celebration: In order to create ceremony and positive reinforcement without interfering with flow, the Game Over view uses Phaser Tween + timed events to trigger congratulatory music and fireworks.

Data-aware layout: After each level, a results page compiles metrics for researchers and parents. Score, attempts, and elapsed time stay on-screen.

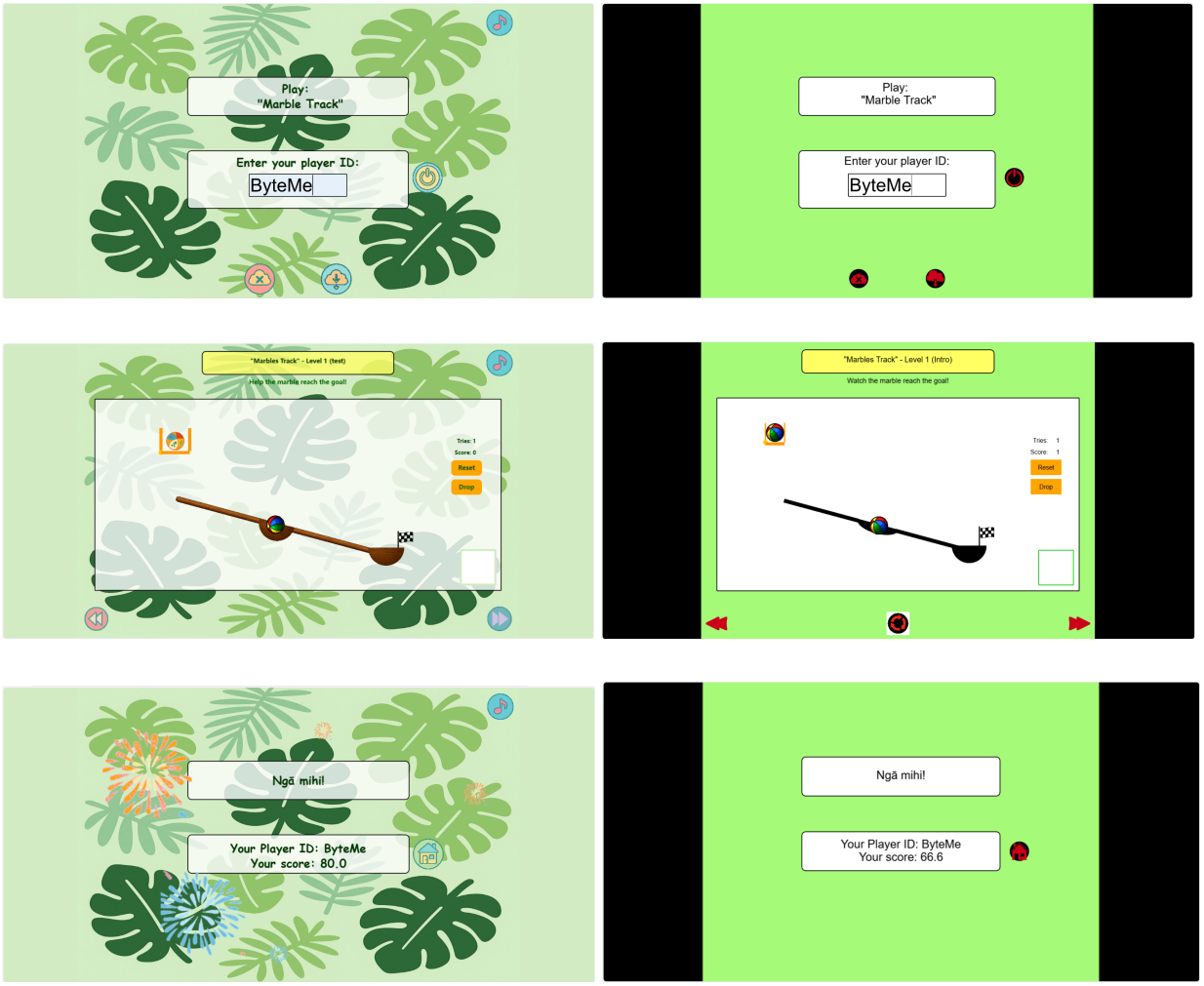


Figure 3. Real-time gameplay screenshots: final (left) vs. original (right) design.

UI Modularity and technical architecture

Implementation: Multiple levels extend a custom MarbleTrackScene based on Phaser.Scene.

Audio: Encapsulated in AudioManager.ts, which manages global mute/unmute and resource loading.

Styling: In-game banners are stored in banners.ts, while all UI constants (colors, text, and layout) are kept in constants.ts.

Project Implementation

Implementation details

The system was implemented using the Phaser framework in combination with TypeScript. Phaser's Scene system was employed to structure the game into modular, manageable components. Each scene in the game is implemented as a class that extends Phaser's built-in Scene class, providing a consistent structure for asset loading, user interaction, and game logic execution.

Phaser scenes follow a standard lifecycle comprising the following key methods:

(1) constructor() – Initializes core values and invokes the parent constructor with scene configuration.
(2) init() (optional) – Used to initialize variables and constants before asset loading begins.
(3) preload() – Handles the loading of all required assets such as images, sounds, and fonts.
(4) create() – Constructs the game world once assets are ready. This includes UI setup, object creation, and interaction configuration. All the functions needed to set up the level are called here.
(5) update() (optional) – Called once per frame; used in this project primarily to manage real-time interactions such as drag-and-drop behavior.

The overall architecture consists of three categories of scenes: Main Menu, Game Levels, and Game Over. All scenes extend from Phaser's Scene base class. Game levels share an additional layer of abstraction via a custom superclass named MarbleTrackScene. This class encapsulates shared logic and behavior across all levels, including UI elements (e.g., Drop, Reset, Next, and Previous buttons) and utility functions such as:

(1) createTrack() – Generates track objects based on configurable parameters including length, angle, and position.
(2) handleDrag() – Implements the logic for managing user interactions with draggable elements such as marbles or tracks.
(3) handleCollision() – Detects collisions between marbles and goal objects, and determines whether the correct target marble has reached the goal. Upon successful detection, it triggers the corresponding score assignment.
(4) recordScoreForPlayer() – Retrieves data generated during the current level session —such as number of tries, score, duration and movement path—and writes the structured results into localStorage for later analysis.

This approach ensures that common logic is centralized, reducing redundancy and simplifying future maintenance. Each individual level extends MarbleTrackScene and overrides or extends methods as needed to implement level-specific behaviors. For example, some levels support marble dragging, others focus on drop-only mechanics, and Level 3 introduces draggable tracks. Despite these differences, shared components and data recording mechanisms remain consistent across all levels.

This structured use of object-oriented inheritance supports modularity and scalability while maintaining a coherent user experience throughout the game.

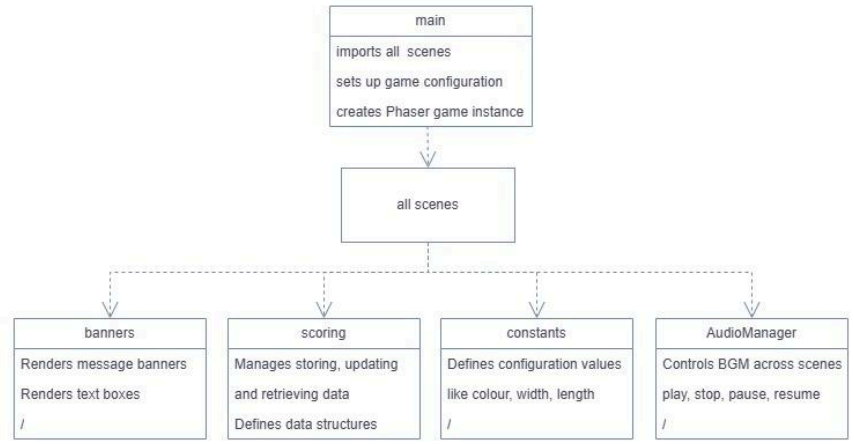


Figure 4. UML Component Diagram illustrating the dependency relationships among modules

In addition to the scene inheritance hierarchy, the implementation also relies on several independent modules that provide shared functionalities. These modules are designed as singletons and imported where needed, establishing dependency relationships with scenes rather than forming a parent-child structure.

(1) AudioManager: This module manages the playback of background music across all game scenes. It provides functions to play, pause, resume, and stop music globally, ensuring consistency in the auditory experience throughout the game.

(2) constants: This module defines key configuration parameters such as object sizes, color schemes, and data storage keys. Centralizing these values helps maintain uniformity across the application and simplifies updates or modifications.

(3) banners: This module offers utility functions for rendering decorative banners and message boxes within the scenes. It encapsulates visual formatting and positioning logic for reusable textual overlays.

(4) scoring: This module maintains the data structures and logic required to store detailed scoring records. It supports multiple players and tracks multiple attempts per level. Data is persisted using the browser's localStorage, allowing continuity between sessions and detailed behavioral analysis.

(5) main.ts: This is the central entry point of the system. It imports all game scenes and supporting modules, defines the game configuration (e.g., canvas size, physics engine settings, scale mode), and initializes the Phaser engine. It also registers all scenes and controls their loading sequence. Once configured, it instantiates the Phaser.Game object, thereby starting the application.

Collectively, these modules support modularity and separation of concerns in the overall system architecture. They are essential in organizing the game's functionality in a maintainable and scalable way.

Reflect on the risks & unexpected challenges

1	**Identified Risk**: Workload imbalance or missed deadlines	
A	**Mitigation Strategy**: Tasks clearly divided and updated weekly.	**Reflection**: After each client meeting, we held internal team meetings to review completed tasks, identify remaining work, and confirm task assignments.
B	**Mitigation Strategy**: Reallocate tasks if a team member encounters difficulties.	**Reflection**: This strategy proved effective, as we successfully reallocated tasks on multiple occasions, which helped maintain steady development progress.

	Faced Challenge	Strategy Reflection
1	**Integrating the scoring system from Apple Catcher into our Marble Track:** Due to significant differences in game structure and implementation, directly adapting the scoring system from Apple Catcher proved difficult. Furthermore, after the first prototype, the client introduced additional requirements that differed from the original, including a new data structure and new metrics such as duration and movement path.	Instead of modifying and reusing the existing Apple Catcher code, we first studied and understood its underlying principles. Based on this understanding, we designed a new scoring data structure and push mechanism tailored to the architecture of our game. This approach proved effective and allowed us to meet both legacy compatibility and new functional requirements.
2	**Balancing physics simulation with performance during drag interactions:** Phaser's physics engine does a great job with collisions and forces, but using it directly to control dragging often causes lag or shaky movement.	Our solution was to pause the physics engine when the player starts dragging, manually update the object's position while dragging, and then resume physics when they let go. At that point, we give the object an initial velocity based on the final drag direction. This avoids constant interference in the physics loop and keeps the drag experience smooth.
3	**Unstable and unpredictable collision behavior in Level 1 caused by the randomness of MatterJS physics simulation:** Specifically, the dropped marble sometimes failed to collide with a stationary marble trapped in a bowl, or got stuck upon impact, breaking the intended level logic.	Despite extensive adjustments to physical parameters (e.g., friction, restitution, mass), the problem remained inconsistent and difficult to reproduce reliably. Instead of relying solely on physics simulation, we introduced an invisible physics object to temporarily block the first marble and ensure proper collision with the second. This workaround stabilized the interaction while preserving intended gameplay. The challenge highlighted the importance of rethinking the problem rather than over-tuning simulation parameters.

Results & Evaluation

The main goal of this project was to develop an interactive physics-based game for children's cognitive research and to record structured data during gameplay. Based on the final outcome, our team achieved almost all key goals. We also made several updates based on client feedback to improve the system's stability, make it more usable, and suitable for research.

Project Goal Achievement

Our team successfully built and deployed the Marble Track game. The system includes a main menu, four experimental levels (Level0 to Level3), a game-over screen, and a data export function. The game allows users to drag and drop marbles and tracks in an intuitive way. It includes different types of gameplay, such as simple drops, interactions between marbles, and complex path planning.

Before starting, users can enter a unique ID. The system then automatically records the number of attempts, movement paths, score, and time taken. All this data can be exported in JSON format with one click. These features fully meet the client's requirements for data collection.

Testing and Quality Assurance

We had a test plan for each stage of development. Testing was done in the following ways:

- **Function testing**: We checked that each button (Drop, Reset, Next, Download Data, etc.) worked as expected.
- **Interaction testing**: We tested the drag-and-drop functions on different devices and browsers, and checked whether any errors happened during use.
- **Data testing**: We made sure that each level's data was saved correctly in localStorage, and that the exported JSON file had the right format.

During testing, we found and fixed many problems, such as drag delay, unstable physics, and data issues. The solutions were explained earlier in this report, so we will not repeat them here. The final version runs smoothly on major browsers and also works well on iPads and similar devices.

We performed testing across three main development stages, based on the Gantt chart timeline:

Stage 1 – Basic gameplay and UI (Week 6–7):

We tested Level0, the main menu, and early user interface functions. We checked marble dropping, score display, and basic localStorage recording. 10 manual test cases were run. For example, one test case examined how repeated clicks on the Reset button without performing

any in-game interaction affected the number of tries, and confirmed that such actions should not increment the count. 2 bugs were found with the reset button and score not clearing, both fixed immediately.

Stage 2 – Level logic and Data collection (Week 8–9):

After all levels were implemented, we tested drag mechanics, marble collisions, and structured data logging, particularly in complex multi-action scenarios. 14 test cases were conducted, with emphasis on edge cases involving non-standard user actions that deviated from the intended gameplay path.

For example, in Levels 1 and 2, where both drop and drag actions are available, we tested different action sequences to verify whether tries were correctly recorded and whether the Drop button functioned properly. Regarding data collection, one of the test cases evaluated whether the movement path would be correctly recorded when the duration of dragging interaction is shorter than the sampling interval.

Many of the test cases were discovered unintentionally through exploratory interactions, which revealed certain design flaws and bugs in the original system. These issues, once identified, allowed us to continuously refine and improve the implementation. In this stage, 5 interaction bugs (including physics lag during drag) were fixed.

Stage 3 – Game Over screen and cross-device testing (Week 10–11):
We tested result display, JSON export structure, and device compatibility on iPad and desktop browsers (Chrome, Edge). 4 test cases were designed, all passed. JSON output was verified against a defined schema, and no format issues were found.

In total, **28 manual test cases** were conducted across all three phases. All major bugs were resolved before the final delivery. Testing covered all core features and was confirmed on both desktop and tablet environments.

Cost Evaluation

This system was built using an HTML5 stack and Phaser. It runs in the browser and does not need a server, so the cost of maintenance is very low. Data is saved locally, and there is no need for a database or cloud service. The game can be hosted for free using GitHub Pages. In the future, new features or games can be added by reusing the current structure and adding new scenes. This makes it easy to further extend the platform.

Strengths and Weaknesses

Strengths:

- **Good user experience**: The interface is clear and easy to use. Young children can play without help.
- **Clear data structure**: The system records all needed data. The format is simple and easy to understand, which helps future feature development.
- **Simple and extendable code**: The code is organized in a way that makes it easy to add new games or change the current ones.
- **Improved visuals and sound**: Compared to older games, this version looks and sounds better. The UI is clearer, and the game logic is easier to follow. This helps keep children engaged and gives better data.

Weaknesses:

- **Limits of the physics engine**: Even after many changes, MatterJS still has problems in rare cases. Some issues cannot be fully fixed because of how the engine works.
- **Limited features**: Right now, the system only supports local data collection for one player. It does not support online syncing or multi-device use.

Overall Evaluation

In conclusion, the Marble Track project has reached its goals in terms of functionality, user experience, and use in research. The modular design also makes it ready for future updates. While some parts can still be improved, the current system already offers a useful and reliable tool for studying cognitive development in young children.

Future Work

While the Marble Track game was successfully developed to meet the core requirements of the project brief, there remain areas for further development and enhancement. These include one key feature that was partially implemented but not completed, as well as several ideas proposed by the team to improve the platform's research potential, usability, and accessibility in the future.

The only feature that was formally requested by the client but not completed was the Apple Catcher game. After we had completed development on Marble Track, the client asked us to begin work on this second game, which was designed to assess other cognitive skills such as attention, visual scanning, and quick decision-making. We managed to make progress by setting up the game's basic structure, modifying the user interface, and implementing a couple of initial levels. However, due to time constraints, we were unable to fully complete the Apple Catcher game within the duration of the project. Completing this game in future work would significantly expand the scope of cognitive tasks available to researchers and offer a broader range of behavioral insights.

In addition to this, our team identified several opportunities for future enhancement that were not part of the original requirements but would greatly improve the system's overall functionality and value. One idea is the integration of a remote data storage system. Currently, gameplay data is stored locally on the user's device. While this is sufficient for small-scale studies, implementing cloud-based data storage would allow researchers to access data in real-time, track usage across different participants, and conduct longitudinal studies with greater ease.

Another enhancement would be the inclusion of accessibility features. Although the current interface is designed to be intuitive for young users, it could be made more inclusive by supporting visual, auditory, or motor accessibility needs. For instance, high-contrast color modes, or simplified interaction methods that would make the game more usable for children with diverse learning or developmental profiles.

The addition of customizable or adaptive difficulty settings is another area with strong potential. Presently, the difficulty level of each Marble Track puzzle is fixed. Introducing a dynamic difficulty adjustment system based on a child's age, previous performance, or researcher input could allow for more personalized engagement while still ensuring reliable data collection.

Looking ahead, supporting broader deployment of the platform beyond laboratory settings is also a promising direction. This would require adding mechanisms for obtaining parental consent and collecting caregiver or educator feedback. Such features would be essential for

ethical deployment in home or early education environments and would allow researchers to gather insights from more diverse populations.

Finally, improving the internal code structure to support reusability would be a valuable enhancement for long-term maintainability. As the platform grows to include more games and researchers, making the codebase more modular and extensible would significantly reduce the learning curve for new developers. By abstracting shared components, standardizing game logic templates, and documenting core systems, future teams would be able to create and integrate additional games more efficiently without needing to deeply analyze the existing code.

Conclusion

The primary aim of this project was to develop an interactive, physics-based game designed to support early childhood cognitive research by providing a fun and engaging way for young children to demonstrate cause-and-effect understanding, while simultaneously capturing detailed and structured gameplay data for researchers.

Our key findings show that the Marble Track game meets these objectives successfully. We delivered a fully functional game with multiple levels that allow for different types of gameplay interactions, from simple marble drops to complex puzzle solving. The game features realistic physics simulations powered by MatterJS within the Phaser 3 framework, offering high-fidelity interactions that enable children to explore natural cause-and-effect scenarios in an engaging way. The user interfaces were designed with simplicity and accessibility in mind, allowing young children to play independently, which fosters authentic behavior and improves data validity. The game effectively tracks user inputs, movement paths, attempts, scores, and completion times, and exports this data in a format readily usable for analysis. Additionally, through iterative testing and thoughtful technical solutions, we overcame challenges related to physics simulation stability and user interaction responsiveness, resulting in a smooth and accessible user experience across devices.

The major outcome of this project is a versatile, modular game platform that not only meets the client's data collection requirements but also establishes a strong foundation for future research tools. A significant achievement is the development of a reusable, extensible data collection and scoring system shared across both games. This system logs key behavioral data, including attempt counts, completion times, and interaction paths, locally and supports one-click JSON export without requiring back-end infrastructure. By automating detailed data capture within an engaging game environment, Marble Track provides researchers with unprecedented insight into early cognitive processes. Additionally, the open-source, modular architecture allows for future expansion and customization, supporting scalable remote cognitive testing and experimental flexibility.

Overall, the project delivers a robust platform that advances traditional early learning research methods by combining engaging gameplay with precise and structured data capture. Its successful integration of child-friendly design, realistic physics, and comprehensive logging tools marks a meaningful step forward for the Early Learning Lab platform, providing researchers with powerful new capabilities to study young children's cognitive development remotely and at scale.

References

Ullman, T. D., Spelke, E., Battaglia, P., & Tenenbaum, J. B. (2017). Mind games: Game engines as an architecture for intuitive physics. Trends in Cognitive Sciences, 21(9), 649–665.

Saxe, R., & Carey, S. (2006). The perception of causality in infancy. Acta Psychologica, 123(1-2), 144–165.

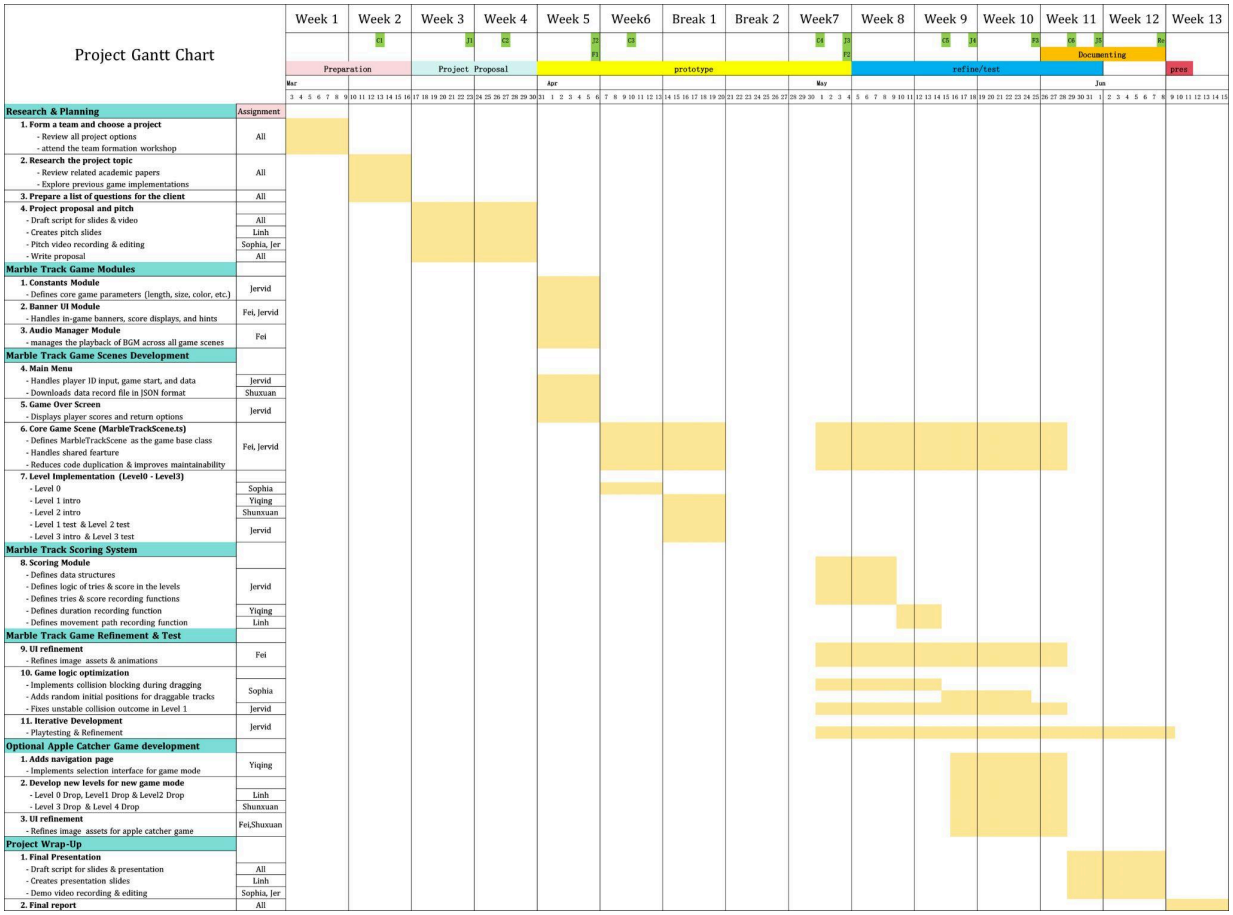
Sobel, D. M., Tenenbaum, J. B., & Gopnik, A. (2004). Children's causal inferences from indirect evidence: Backwards blocking and Bayesian reasoning in preschoolers. Cognitive Science, 28(3), 303–333.

Declaration of Authorship

Section	Author(s)	Signature(s)
Executive Summary	Linh Ha	Linh Ha
Table of Contents	Shuxuan Huang	Shuxuan Huang
Introduction	Shuxuan Huang	Shuxuan Huang
Background	Fei Lan	Fei Lan
Specification & Design	Jervid Cheng/Fei Lan	Jervid Cheng/Fei Lan
Implementation	Jervid Cheng	Jervid Cheng
Results & Evaluation	Yiqing Cheng	Yiqing Cheng
Future Work	Sophia Zhang	Sophia Zhang
Conclusion	Linh Ha	Linh Ha
Appendices	Jervid Cheng/Yiqing Cheng	Jervid Cheng/Yiqing Cheng

Appendices

A. Gantt Chart



B. Link to final build

https://uoa-compsci399-2025-s1.github.io/capstone-project-2025-s1-team-21/

C. Link to video (demo)

https://drive.google.com/file/d/14FbGy_pvA-B83W9AbcC-rixacvAA6p3i/view?usp=sharing