Experimentation Methodology Report: IBC Terminal Functional Fixedness Study

Team I3
Mohit Kumar Singh
Akmal Ali
Rohan Kumar

April 24, 2025

Abstract

This report details the experimental methodology employed in the IBC Terminal research platform. The platform, developed as a text-based adventure game, investigates the cognitive phenomenon of functional fixedness. It utilizes a between-subjects design, presenting participants with either a control or an experimental (priming) variant of specific puzzle scenarios embedded within different game worlds. The study collects detailed interaction data, including command inputs, AI responses, keystroke dynamics (timing, hesitations, corrections), and puzzle-solving metrics (attempts, time to solution, success rates). Data is stored locally and uploaded to a backend MongoDB database for analysis. An AI-driven analysis component further evaluates session data to provide qualitative and quantitative insights into functional fixedness patterns. The primary goal is to assess the impact of environmental priming cues on participants' ability to overcome functional fixedness during problem-solving tasks.

1 Introduction

Functional fixedness, a concept originating from Gestalt psychology, describes a cognitive bias limiting individuals to using objects only in their traditional or most common way (Duncker, 1945). This bias can hinder creative problem-solving when a novel situation requires using an object unconventionally. The IBC Terminal research platform was developed to study this phenomenon in an engaging, interactive environment.

The platform presents text-based adventure scenarios where players must solve puzzles. Certain puzzles are specifically designed around objects prone to functional fixedness, requiring players to devise unconventional uses for these objects to progress.

Research Question: How does environmental priming, subtly suggesting unconventional object uses, affect a participant's ability to overcome functional fixedness when solving puzzles in a text-based adventure game?

Hypothesis (Inferred): Participants exposed to environmental priming cues related to unconventional object uses (Experimental Variant B) will demonstrate reduced functional fixedness compared to participants in the control condition (Control Variant A). This will manifest as faster puzzle solution times, fewer incorrect attempts, and a higher frequency of commands attempting unconventional object uses for the target puzzles.

Team Roles:

- **Game Development:** Mohit Kumar Singh
- **Data Analysis & Presentation:** Akmal Ali, Rohan Kumar
- **Experiment Conduction:** Akmal Ali, Mohit Kumar Singh, Rohan Kumar

2 Method

2.1 Participants

Participants are users interacting with the IBC Terminal web application. (Note: Specific recruitment methods, demographics, and compensation details are not detailed within the codebase and are assumed to be managed externally). Each participant session is associated with a unique device identifier ('deviceId') generated and stored locally (`src/lib/deviceId.ts`). Consent for data collection is implicitly managed via the user interface (e.g., `src/components/PrivacyPolicyConsent.tsx`).

2.2 Design

The study employs a **between-subjects experimental design**. Participants are assigned to one of two conditions for each world they play:

- **Independent Variable:** Experimental Condition (Variant)
  - **Variant A (Control):** Standard puzzle and environment descriptions.
  - **Variant B (Experimental):** Environment descriptions include subtle priming cues suggesting unconventional uses for objects, sometimes related to the puzzle's fixed-function object (`src/config/worlds.ts`).
- **Dependent Variables:** A range of quantitative and qualitative metrics is collected to assess functional fixedness and problem-solving behavior (`src/lib/dataCollection.ts`, `src/models/Interaction.ts`):
  - **Puzzle Performance:**
    - Solution Success Rate (Solved / Discovered)
    - Time to Solution (ms, from first puzzle interaction to solution)
    - Number of Solution Attempts
    - Number of Incorrect Attempts before Solution
  - **Interaction Metrics:**
    - Command Input Duration (ms)
    - Keystroke Count per Command
    - Correction Count (backspaces/deletes) per Command
    - Hesitation Count and Duration (pauses during typing)
    - Command Frequency and Patterns
  - **Functional Fixedness Indicators:**
    - Frequency of Conventional vs. Unconventional Use Commands for target objects
    - AI-Generated Functional Fixedness Analysis (Overall Level, Approach, Insight Moments) (`src/app/api/session/analyze/route.ts`)
    - Fixation Score (calculated based on command patterns)
  - **Session Metrics:**
    - Total Session Duration
    - Total Interactions
    - Completion Rate (Objectives met)

Variant performance is aggregated and compared using metrics stored in the 'worldVariants' collection (`src/models/WorldVariant.ts`).

2.3 Materials

- **IBC Terminal Platform:** A web application built with Next.js, providing a terminal interface for interacting with the text-based adventure game (`src/components/Terminal.tsx`).
- **Adventure Worlds:** Multiple distinct game worlds defined in `src/config/worlds.ts`. Each world has unique narratives, objectives, and environments. The specific worlds are detailed below. Note: Worlds 5 and 6, marked as "(SHORT)", were designed as shorter experiences to accommodate potential time limitations during experiment conduction, allowing participants to complete a session more quickly while still engaging with functional fixedness puzzles.

2.3.1 Adventure World Details

- **World 0: Neo-Tokyo 2099**
  - *Description:* A cyberpunk dystopia with unreliable automated systems, requiring creative solutions to navigate.
  - *Objectives:* Infiltrate TechCorp, locate a prototype neural interface, escape the city.
  - *Starting Inventory:* Personal Datapad, ID Chip, Credstick (empty).
  - *Puzzles:*
    - The Sliding Door (Fixed Object: Datapad)
    - The Circuit Breaker (Fixed Object: ID Chip)
    - The Biometric Lock (Fixed Object: Adhesive bandage)
- **World 1: Forgotten Castle**
  - *Description:* A medieval fantasy realm with ancient magic, secret passages, and fading mystical barriers.
  - *Objectives:* Find the lost crown, discover the cause of abandonment, escape before barriers seal.
  - *Starting Inventory:* Unlit torch, Small dagger, Empty waterskin.
  - *Puzzles:*
    - The Ancient Lock (Fixed Object: Dagger)
    - The Broken Bridge (Fixed Object: Shield)
    - The Sealed Chamber (Fixed Object: Torch)
- **World 2: Chronos Station**
  - *Description:* An abandoned space station with malfunctioning systems and localized temporal anomalies.
  - *Objectives:* Restore power, discover the crew's fate, secure an escape pod before orbit decays.
  - *Starting Inventory:* Emergency Beacon (inactive), Multitool, Half-charged Oxygen Tank.
  - *Puzzles:*
    - The Radiation Barrier (Fixed Object: Food tray)
    - The Sealed Airlock (Fixed Object: Access Card)
    - The Quantum Lock (Fixed Object: Holographic Display)
- **World 3: Subterranean Nexus**
  - *Description:* An underground network of caves housing the remains of a technologically advanced lost civilization.
  - *Objectives:* Activate the transport network, recover historical records, find a way back to the surface.
  - *Starting Inventory:* Rechargeable Lantern (low battery), Coil of rope, Energy bar.
  - *Puzzles:*
    - The Crystal Mechanism (Fixed Object: Water bottle)
    - The Sonic Chasm (Fixed Object: Metal container)
    - The Pressure Mechanism (Fixed Object: Notebook)
- **World 4: Ethereal Planes**
  - *Description:* A dreamlike dimension where reality is malleable and responds to thoughts and emotions.
  - *Objectives:* Stabilize presence, locate the Nexus of Consciousness, return to own dimension.
  - *Starting Inventory:* Reality Anchor (partially functional), Memory Crystal, Emotion Vial (empty).
  - *Puzzles:*
    - The Barrier of Doubt (Fixed Object: Mirror)
    - The Memory Seal (Fixed Object: Emotion Vial)
    - The Shifting Chasm (Fixed Object: Reality Anchor)
- **World 5: The Late Night Office (SHORT)**
  - *Description:* Accidentally locked inside an office building after hours with partial power shutdown.

    - *Objectives:* Bypass the locked main exit door, escape the building.
    - *Starting Inventory:* Paperclip, Plastic Ruler (30cm), Empty Coffee Mug.
    - *Puzzles:*
        - The Dropped Key (Fixed Object: Paperclip)
        - The High Emergency Button (Fixed Object: Plastic Ruler)
  - **World 6: Server Room Glitch (SHORT)**
    - *Description:* Trapped in a university server room after a power fluctuation triggers a security lockdown.
    - *Objectives:* Find and activate the manual door release, escape before backup power fails.
    - *Starting Inventory:* Ethernet Cable (3m, unplugged), Blank CD-R, USB Stick (FAT32 formatted, empty).
    - *Puzzles:*
        - The Override Jumper (Fixed Object: Ethernet Cable)
        - The Misaligned Sensor (Fixed Object: Blank CD-R)

- **Puzzles:** Embedded within each world are specific puzzles. Key puzzles are designed to elicit functional fixedness:
  - Each puzzle identifies a 'fixedFunctionObject' (e.g., Datapad, Dagger, Food Tray).
  - The 'solution' requires using this object in an unconventional manner (e.g., Datapad as a wedge, Dagger handle as a key).
  - Each puzzle has distinct 'controlVariant' and 'experimentalVariant' descriptions presented to the user based on their assigned condition. The experimental variant often includes subtle environmental cues related to the unconventional use.
  - A 'narrativeJustification' explains the puzzle's importance within the game's story.
- **AI Narrator:** A backend AI model (Google Gemini 2.5 Flash, via `src/lib/gemini.ts`) generates narrative descriptions, responds to user commands, and evaluates puzzle solutions based on predefined world configurations and system prompts (`src/config/systemPrompts.ts`).
- **Data Storage:** MongoDB database stores session data, interaction logs, and aggregated variant statistics (`src/lib/mongodb.ts`, `src/models/`).

2.4 Technology Stack

The IBC Terminal platform is built using a modern web technology stack:

- **Frontend Framework:** Next.js (v15.3.1) with React (v19) and TypeScript.
- **Styling:** Tailwind CSS (v4).
- **Backend Logic:** Next.js API Routes (Serverless Functions) running on Node.js.
- **Database:** MongoDB (via 'mongodb' driver v6.16.0) for storing session data, interactions, and aggregate statistics.
- **AI Model:** Google Gemini 2.5 Flash (specifically 'gemini-2.5-flash-preview-04-17' via '@google/genai' v0.9.0) for generating narrative responses, world introductions, and session analysis.
- **Unique Identification:** 'uuid' library (v11.1.0) for generating persistent device identifiers.
- **Analytics:** Vercel Analytics ('@vercel/analytics' v1.5.0) for frontend usage tracking (optional).

2.5 Procedure

1. **Session Initialization:** Upon starting, the user's client generates or retrieves a 'deviceId'. The client requests session initialization from the backend (`/api/session/initialize`), providing the 'deviceId', desired 'worldId', and assigned 'variant' ('A' or 'B'). The backend creates or retrieves a session record in MongoDB (`src/models/Session.ts`).

2. **World Introduction:** The backend provides an initial world introduction message, specific to the assigned 'worldId' and 'variant'. For new sessions, this is generated by the AI; for returning sessions, it's retrieved from history. Markdown is stripped from the intro (`src/app/api/session/initialize/route.ts`).

3. **User Interaction:** The participant interacts with the game by typing commands into the terminal interface (`src/components/Terminal.tsx`, `src/components/CommandInput.tsx`).

4. **Keystroke Logging:** As the participant types, detailed keystroke data (key, timestamp), corrections (backspace/delete), and hesitations (pauses > threshold) are captured locally by the client (`src/lib/dataCollection.ts`).

5. **Command Submission:** When a command is submitted, the client records the final command, input duration, command length, and associated metrics. This interaction data, along with 'deviceId', 'worldId', 'variant', and timestamp, is temporarily stored in the browser's localStorage (`trackUserInteraction` in `dataCollection.ts`).

6. **AI Processing:** The command is sent to the backend (`/api/command`). The backend retrieves the session history, appends the user command, and sends the history along with system prompts to the AI model to generate a response (`src/lib/gemini.ts`). The AI response also determines if a puzzle was attempted or solved.

7. **Response and Update:** The AI's response is sent back to the client and displayed (`src/components/ResponseDisplay.tsx`). The client updates the corresponding interaction record in localStorage with the AI response text and any puzzle context (e.g., 'isAttemptedSolution', 'isSolutionSuccess') returned by the backend (`updateInteraction` in `dataCollection.ts`). Puzzle state changes (discovered, solved, attempts) are also updated in the session record on the backend (`src/models/Session.ts`).

8. **Data Upload:** Upon session completion (or potentially periodically/on browser close), the client triggers an upload of all interactions stored in localStorage, along with a generated session summary, to the backend analytics endpoint (`/api/analytics/track`) (`uploadStoredInteractions` in `dataCollection.ts`).

9. **Backend Storage:** The backend receives the batch data, records each interaction in the 'interactions' collection, and records the session summary (including puzzle attempts, timings, and AI analysis results) in the `session_summaries` collection (`src/app/api/analytics/track/route.ts`, `src/models/Interaction.ts`).

10. **Session Analysis (Optional/Batched):** The client can trigger (or the backend can perform) a detailed session analysis (`/api/session/analyze`). This involves sending the session's interactions and puzzle data to the AI to generate a structured analysis focusing on functional fixedness, calculating derived metrics, and enhancing puzzle attempt data (`src/app/api/session/analyze/route.ts`).

11. **Aggregate Statistics Update:** After a session is recorded, aggregate statistics for the specific world and variant (e.g., total users, average duration, puzzle solve rates, average attempts) are updated in the 'worldVariants' collection (`src/models/WorldVariant.ts`).

2.6 Gameplay Flow

The core gameplay experience is designed as an interactive text-based adventure mediated by an AI narrator. The flow proceeds as follows:

1. **Loading and Consent:** The user accesses the web application (`src/app/page.tsx`). If it's their first visit, they are presented with a privacy policy and consent form (`src/components/PrivacyPolicyConsent.tsx`). Consent is stored in a cookie.

2. **Device ID and Allocation:** Once consent is given (or if already present), a unique ‘deviceId‘ is generated or retrieved from local storage. Based on this ‘deviceId‘, the system deterministically assigns the participant to a specific experimental ‘variant‘ (’A’ or ’B’) using the ‘allocateWorld‘ function (`src/lib/deviceId.ts`). This ensures a balanced distribution across conditions.

3. **World Selection:** The user is presented with a list of available worlds and their descriptions. They select a world by typing its corresponding number (`src/components/Terminal.tsx`).

4. **Session Initialization:** The client sends the ‘deviceId‘, selected ‘worldId‘, and assigned ‘variant‘ to the backend API (`/api/session/initialize`). The backend retrieves or creates a session record in MongoDB.

5. **World Introduction:** For a new session, the backend requests an introductory narrative from the Gemini AI, tailored to the specific world and variant (`src/lib/gemini.ts`, `getWorldIntroduction`). This introduction sets the scene, describes the initial environment, and subtly hints at the player’s goals. For returning sessions, the first AI message from the stored history is used. The introduction text is streamed to the user interface with a typing animation and sound effect (`src/components/ResponseDisplay.tsx`, `src/components/Terminal.tsx`).

6. **Interactive Loop:**

   - **Command Input:** The user types commands (e.g., "look around", "examine datapad", "use dagger as lever") into the input field (`src/components/CommandInput.tsx`). While typing, keystrokes, timing, corrections (backspaces), and hesitations (pauses > 1 second) are recorded locally.
   - **Command Submission:** Upon pressing Enter, the command and its associated metrics (input duration, keystroke details, etc.) are captured. The command is displayed in the terminal output.
   - **Backend Processing:** The command, along with ‘deviceId‘, ‘worldId‘, ‘variant‘, and session history, is sent to the backend API (`/api/command`).
   - **Puzzle Context Analysis:** The backend analyzes the command to determine if it relates to a known puzzle, specifically checking if it involves the ‘fixedFunctionObject‘ and if it represents a conventional use or an unconventional solution attempt (`analyzePuzzleContext` in `/api/command/route.ts`).
   - **AI Response Generation:** The command and updated history are sent to the Gemini AI, along with system prompts defining the AI’s role as narrator, game rules, and world context (`src/lib/gemini.ts`, `getGeminiResponse`). The AI generates a response describing the outcome of the command within the game world.
   - **Response Display:** The AI’s response is sent back to the client. A thinking indicator (blinking cursor and sound) is shown while waiting. Once received, the response is displayed with a typing animation and sound effect (`src/components/ResponseDisplay.tsx`).
   - **State Update:** The backend updates the session history in MongoDB. If the command was a puzzle attempt, the puzzle state (attempts, solved status, timestamps) is updated (`src/models/Session.ts`). The interaction record, including the AI response and final puzzle context (e.g., ‘isSolutionSuccess‘), is updated in both local storage (`updateInteraction` in `src/lib/dataCollection.ts`) and the backend database.

7. **Puzzle Solving:** Puzzles typically require the user to identify the ‘fixedFunctionObject‘ and devise an unconventional ‘solution‘ command (e.g., "use datapad as wedge"). The AI is prompted not to give direct hints but may provide subtle environmental cues (especially in Variant B). Success is determined by matching the command pattern and analyzing the AI’s response for success indicators (`analyzeResponseForSolution` in `/api/command/route.ts`). The game continues until all puzzles defined for the world are solved.

8. **Session Completion:** The user can end the session by typing commands like "exit" or "quit". This triggers the upload of all locally stored interaction data and the generated session summary to the backend (`uploadStoredInteractions` in `src/lib/dataCollection.ts`). A confirmation message is displayed.

2.7 Data Analysis

- **Quantitative Analysis:** Statistical comparisons (e.g., t-tests, ANOVA) will be performed between Variant A and Variant B groups on the dependent variables stored in MongoDB. Key comparisons include time to solution, number of attempts, hesitation metrics, and frequency of unconventional use commands for the target puzzles. Aggregate data from ‘worldVariants‘ provides high-level comparisons.
- **Qualitative Analysis:** The AI-generated ‘functionalFixednessAnalysis‘ for each session provides qualitative insights into user strategies, points of difficulty, and potential moments of insight. Conversation transcripts (‘conversationData‘) can be reviewed for specific patterns.
- **Keystroke Dynamics:** Analysis of input duration, corrections, and hesitations can provide finer-grained insights into cognitive load and uncertainty during puzzle-solving, particularly comparing moments of conventional vs. unconventional thinking.

3 Expected Results and Discussion

Based on functional fixedness literature, we expect participants in Variant B (priming) to outperform those in Variant A (control) on the functional fixedness puzzles. Specifically, Variant B participants are expected to solve these puzzles faster, with fewer attempts, and exhibit more commands exploring or attempting the unconventional use of the target objects. Hesitation patterns might differ, potentially showing less hesitation before attempting the unconventional solution in Variant B if the priming was effective. The AI analysis is expected to rate Variant B participants as having lower overall functional fixedness and potentially identify the environmental cues as contributing factors to insight. Comparing metrics across different puzzles will help identify which objects or scenarios induce stronger functional fixedness effects.

4 References

Duncker, K. (1945). On problem-solving. *Psychological Monographs, 58*(5), i–113.