Cave Save

Group 28: Romarcx Corpuz, Hector Castro, Ajani Lazo

Design Report Summary

Project Overview

Cave Save is a short-duration, replayable cave survival simulator. The game places the player inside a dynamically generated underground cave system following a collapse. Rather than relying on a traditional countdown timer, the core survival mechanic is driven by deteriorating air quality, the player must navigate to a fresh air exit before oxygen depletion and rising carbon dioxide levels cause fatal impairment. Each run is designed to last approximately five minutes, and the cave layout, airflow routes, and hazards are procedurally regenerated each time, ensuring high replayability.

Beyond entertainment, Cave Save is designed to serve as a lightweight training simulator for recognizing and responding to hazardous underground environments, such as mines or caves with poor ventilation. Its primary stakeholders include safety training programs, mine safety organizations, caving clubs, park services, and educational institutions seeking realistic simulation-based learning tools. A dedicated Training Mode allows instructors to select preset scenarios, observe learner decisions in real time, and review performance data afterward, including time in unsafe air, route choices, and key decision points.

Gameplay Mechanics and Scope

Players begin each run equipped with a headlamp and a handheld gas meter that displays oxygen (O_2) and carbon dioxide (CO_2) levels. The simulation models realistic gas behavior, CO_2 accumulates in low-lying chambers, airflow drafts indicate safer routes, and the player's movement and physical exertion accelerate gas deterioration. Sprinting, for instance, increases CO_2 production and O_2 consumption, while remaining still preserves air more slowly. Players can intervene by opening or sealing passages to reroute airflow, using fans or other tools to improve ventilation, and avoiding high-CO_2 pockets by reading environmental cues such as airflow ribbons and thermal indicators.

The game features six core scenarios, starting a run, monitoring air quality, performing ventilation interventions, reacting to collapses, navigating floods, and reaching a safe exit. Dynamic events such as tunnel collapses and rising water levels alter the cave mid-run, forcing players to adapt quickly. Success is measured by escape rate, average oxygen margin at exit, time taken, frequency of entering dangerous gas pockets, and ventilation decisions.

Requirements and Constraints

The functional requirements of Cave Save center on procedural cave generation, solvable layouts with guaranteed exit paths, real-time gas simulation, player tools such as the gas meter, dynamic hazards including collapse and flooding events, and an end-of-run performance summary. Non-functional requirements include stable 60 fps performance, support for cave maps with at least 50 chambers, crash-free operation during normal gameplay, and safe handling of invalid or unexpected inputs. Usability requirements specify that new users should be able to understand the controls within five minutes, and that the interface should remain readable and accessible across different screen sizes and setups, including colorblind-safe indicators.

System Design and Architecture

Cave Save follows a layered Model-View-Controller architecture divided into five subsystems: Core Gameplay, World Model, User Interface, Persistence, and Utilities/Configuration. The Core Gameplay subsystem includes the GameController and GameSession, which coordinate all gameplay rules, player actions, collision checking, and win/loss conditions. The World Model subsystem defines the cave environment through classes such as CaveMap, Tile, Item, Hazard, and Enemy. The User Interface subsystem handles all screen transitions, HUD updates, menus, and notification displays through a UIManager. The Persistence subsystem manages save files and user settings using a SaveManager and SerializationAdapter that store game data in JSON format. The Utility subsystem provides shared services including audio, logging, resource loading, and configuration.

Project Issues, Risks, and Retrospective

The most significant open issue is the choice of gameplay style. We have not yet committed to a turn-based, real-time, or hybrid model, and this decision has implications for control complexity, game feel, pacing, and overall system balance. A real-time system demands more complex input handling and performance management, while a turn-based approach simplifies logic but is not as interactive or immersive.