Abstract:

This document details the design and development of the AMAZE prototype, a minimalist puzzle game. The main focus of this prototype is the implementation of a procedural level generation **system**, eliminating the need for manual design while ensuring engaging and diverse level creation. This scalable system enables endless replayability and adaptability for puzzle games.

Introduction:

Game Title: AMAZE

Genre: Puzzle

Platform: Mobile (iOS and Android)

Target Audience: Casual gamers of all ages who enjoy relaxing, brain-teasing challenges.

Objective: To provide a seamless and engaging puzzle game experience where players solve procedurally generated levels by painting all tiles on a grid.

Core Gameplay Mechanics:

Core Mechanics:

- Ball Movement:
    - The ball moves in straight lines based on swipe input.
    - Movement stops at the grid's edge or when hitting a wall.
- Grid Interaction:
    - Tiles change color when the ball passes over them.
    - All tiles must be painted to complete the level.
    - Obstacles and walls add complexity to movement paths.

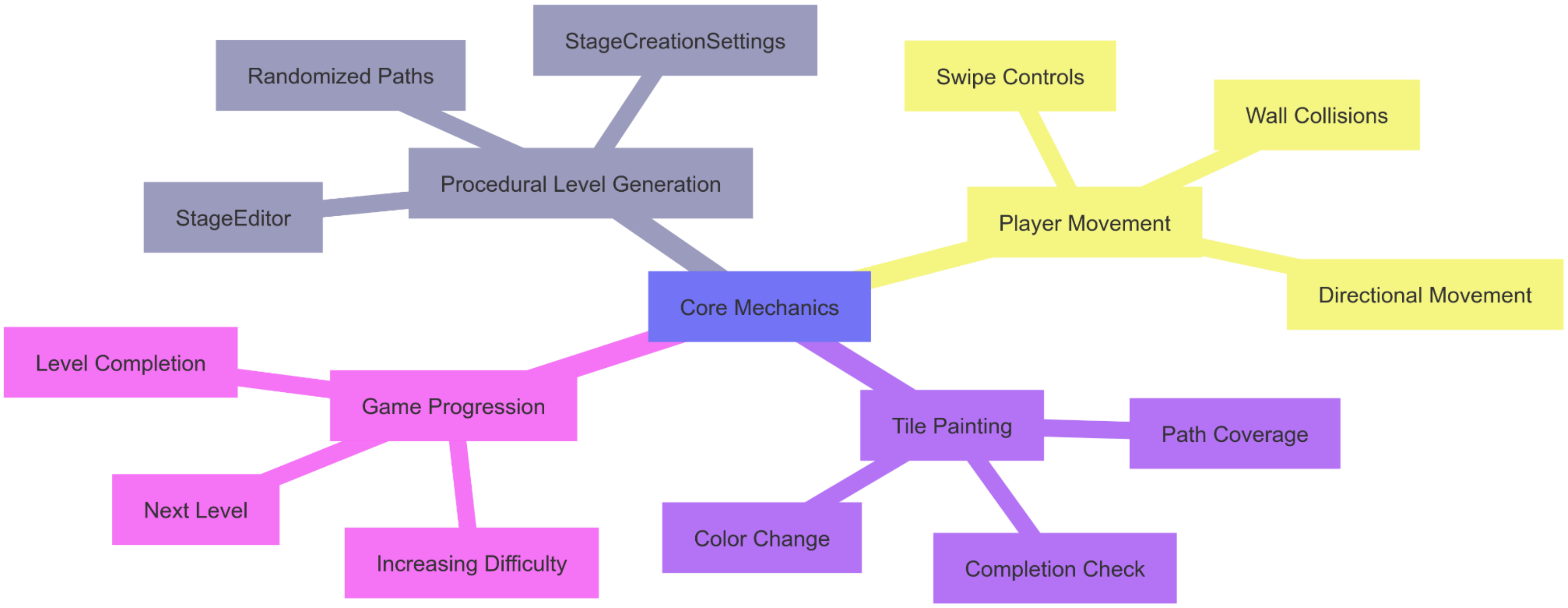
Player Controls:

- **Touch-based input**: Swipe gestures (up, down, left, right) to move the ball.
- **Keyboard support (for PC testing)**: W, A, S, D keys for movement.

Win and Lose Conditions:

- **Win Condition**: The level is completed when all tiles are painted.
- **Lose Condition**: No fail state; players can retry levels indefinitely.

Mind Map:



Procedural Level Generation System:

System Overview:

The level generation system is designed to create unique levels programmatically, minimizing human intervention. It consists of:

- `StageCreationSettings`:
    - Defines parameters such as grid size, number of moves, and path complexity.
    - Supports four difficulty levels: Easy, Moderate, Hard, and Very Hard.
- `StageEditor`:
    - A custom Unity Editor tool that generates levels based on "StageCreationSettings".
    - Saves levels as text files that can be loaded dynamically during gameplay.
- `GameplayBehaviour`:
    - Reads level files and spawns walls, tiles, and the player dynamically.
    - Handles player movement, collision detection, and win condition checks.

Advantages:

- **Scalability**: Allows creation of infinite levels without manual intervention.
- **Replayability**: Ensures variety through randomized layouts within predefined constraints.

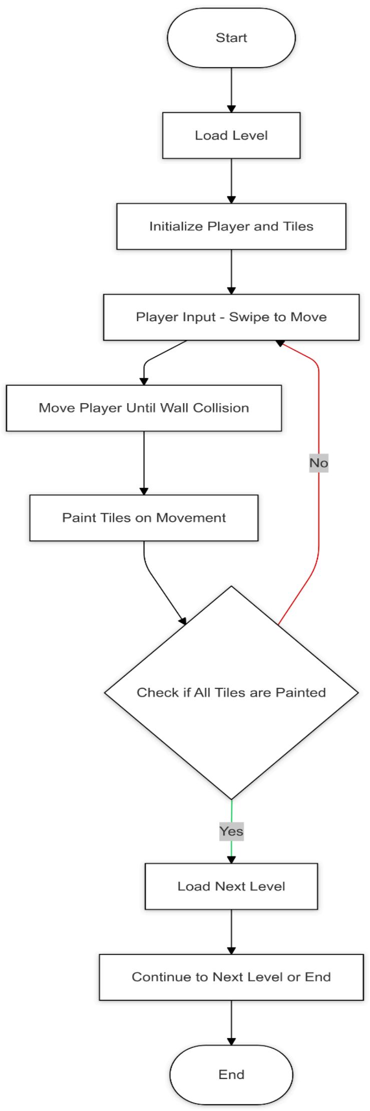
- **Customization**: Designers can tweak parameters to adjust difficulty and introduce new challenges.

<u>Level Complexity:</u>

Levels are generated with increasing difficulty:

- **Easy**: Small grids with minimal obstacles.
- **Moderate**: Medium grids with additional obstacles.
- **Hard**: Large grids with more complex paths.
- **Very Hard**: Large grids with multiple constraints and complex routes.

<u>Gameplay Loop:</u>


Start
Load Level
Initialize Player and Tiles
Player Input - Swipe to Move
Move Player Until Wall Collision
Paint Tiles on Movement
Check if All Tiles are Painted
No
Yes
Load Next Level
Continue to Next Level or End
End

Gameplay Integration:

Level Loading and Initialization:

- “StageEditor” generates and saves levels as .txt files.
- “GameplayBehaviour” reads these files and dynamically instantiates walls, tiles, and the player.
- The camera adjusts based on grid size.

Player Movement and Interactions:

- Players swipe to move the ball in straight lines.
- Movement stops when the ball reaches a wall or obstacle.
- Tiles change color upon contact.
- If all tiles are painted, the level transitions to the next stage.

Stage Completion and Transition:

- The game checks for unpainted tiles.
- If all tiles are painted, a celebratory effect is displayed.
- The system loads the next level seamlessly.

Art Style and Theme:

Visuals:

- Minimalist design with bright, vibrant colors for tiles and the ball.
- Clean and uncluttered UI to emphasize gameplay.

Animations:

- Smooth ball movement and subtle tile-painting effects.
- Visual feedback upon hitting walls.

Technology Stack:

Game Engine:

- Unity

Programming Language:

- C#

Tools Used:

- Unity Editor for level design and procedural generation.
- Photoshop or similar tools for UI and art assets.
- Custom StageEditor tool for automated level creation.

Monetization Strategy:

Free-to-Play Model:

- Advertisements: Shown between levels or as a reward for hints.
- In-App Purchases: Options to remove ads or purchase additional hints.

Future Enhancements:

- Advanced AI-driven procedural generation: Creating more dynamic and engaging levels based on player behavior.
- Timer-based competitive modes: Encouraging speedruns and high-score challenges.
- Global leaderboards: Players can compete for the best completion times.
- Undo move feature: Allowing players to backtrack their last move.
- Level editor for user-generated content: Allowing players to create and share custom levels.

Conclusion:

The AMAZE prototype demonstrates the potential of procedural level generation for puzzle games. By utilizing StageEditor and StageCreationSettings, the system can generate an endless variety of levels while maintaining structured difficulty progression. This ensures long-term engagement and replayability for players. The next phase of development will focus on refining level complexity and integrating competitive features.

References:

- Unity Documentation
- Procedural Content Generation in Games
- Market Analysis Reports for Puzzle Games
- Comparable Games:
    - Roll the Ball (BitMango)
    - Color Fill 3D (Good Job Games)
    - Roller Splat! (VOODOO)
    - Line Color 3D (Coda)
    - AMAZE! (Crazy Labs)