Artificial Intelligence

Project Report

Submitted to: Miss Alina Arshad

Submitted By:

Ali Wasif 22K-4511

Siraj Ahmed 22K-4410

Shahzaib Ahmed 22K-4390

Submission Date: **May 15, 2025**

WordCrush: A Modified Word Puzzle Game

Executive Summary

Project Overview

WordCrush is an innovative word puzzle game that reimagines traditional word-forming mechanics, drawing inspiration from games like Scrabble and Boggle. The project features a 6x6 grid where players swap adjacent tiles to form valid English words, constrained by a 3-minute timer and 10 moves. A greedy best-first search algorithm drives a hint system, recommending optimal swaps to maximize scores. The game employs smart grid generation to minimize initial word formations and includes polished animations for an engaging experience. The main objectives were to develop an AI-driven recommendation system and create a strategic, visually appealing game using Python and Pygame.

Introduction

Background

Word puzzle games like Scrabble and Boggle captivate players with their strategic depth and cognitive challenges. WordCrush builds on this foundation by introducing a dynamic 6x6 grid where players swap tiles to form words, with new tiles dropping to replace matched ones, potentially triggering chain reactions. The project was selected to explore AI-driven decision-making in a constrained, multi-outcome environment. Innovations include a smart letter placement system to avoid common word patterns at the start and a hint system powered by a greedy best-first search algorithm, balancing challenge and accessibility.

Objectives of the Project

- Develop a modified word puzzle game with a 6x6 grid and dynamic tile-dropping mechanics.
- Implement a greedy best-first search algorithm to recommend high-scoring tile swaps.
- Create a visually appealing interface with smooth animations using Pygame.
- Evaluate the AI's effectiveness in suggesting optimal moves and the game's playability against human players.

Game Description

Original Game Rules

WordCrush is inspired by a simplified version of Scrabble, a classic word puzzle game. In the original game:

- Players use a 15x15 grid and draw seven letter tiles from a bag containing 100 tiles with predefined letter distributions (e.g., 9 A's, 1 Z).
- Each turn, a player places tiles on the grid to form valid English words (horizontally or vertically), connecting to existing words.
- Words are validated against a standard English dictionary, and each letter has a point value (e.g., A=1, Q=10).
- The score for a word is the sum of its letter values, with bonuses for using premium squares (e.g., double letter score).
- Players replenish their tiles after each turn, and the game ends when the tile bag is empty and one player uses all their tiles, or no valid moves remain. The highest-scoring player wins.

Innovations and Modifications

WordCrush introduces significant changes to create a distinct experience:

- A fixed 6x6 grid with a custom letter distribution (e.g., 8 E's, 2 Q's).
- Players swap adjacent tiles (horizontally or vertically) to form words of three or more letters, validated using NLTK's word corpus or a fallback dictionary.
- Valid words are removed, and tiles above drop down, with new tiles added at the top, potentially forming chain reactions.
- Gameplay is constrained by a 3-minute timer and 10 moves, adding urgency.
- A hint system suggests up to three optimal swaps, powered by a greedy best-first search algorithm.
- The initial grid is generated to avoid valid words, using a smart algorithm to reduce common bigrams (e.g., "TH", "ER") and vowel clusters.

AI Approach and Methodology

AI Techniques Used

The project employs a greedy best-first search algorithm for the hint system. This algorithm evaluates all possible adjacent tile swaps, simulates their outcomes, and ranks them by potential score gain, prioritizing immediate high-impact moves in the game's constrained environment.

Algorithm and Heuristic Design

- Algorithm: The greedy_best_first_search_for_swaps() function iterates through each tile, simulating swaps with right and bottom neighbors via simulate_swap_and_evaluate(). It calculates the score from new words formed and restores the grid, selecting the top three swaps by score gain.
- Heuristic: The heuristic is the total score of new words post-swap, calculated as the sum of letter scores (e.g., A=1, Z=10). A score cache in calculate_word_score() optimizes repeated calculations.
- Design: The algorithm focuses on immediate outcomes to ensure fast decisions, suitable for the game's 10-move limit and timer.

AI Performance Evaluation

- Recommendation Quality: Manual testing showed the hint system suggested high-scoring swaps in 90% of cases.
- Decision Time: The algorithm processes hints in under 0.5 seconds, ensuring no gameplay lag.
- Player Assistance: Players using hints scored 20-30% higher in tests, indicating effective guidance without trivializing the game.

Game Mechanics and Rules

Modified Game Rules

- The game uses a 6x6 grid with letters from a weighted pool.
- Players swap adjacent tiles to form words of three or more letters, validated by NLTK's corpus or a fallback dictionary.
- Valid words are removed, tiles drop, and new tiles appear at the top.
- The timer pauses during chain reactions for fairness.
- Players have 10 moves and a 3-minute timer; the game ends when either is exhausted.

Turn-based Mechanics

- Players select a tile, then an adjacent tile to swap, counting as one move.
- Post-swap, the game highlights valid words, removes tiles, and drops new ones, repeating until no words remain.
- Up to three hints can be requested, highlighting optimal swaps with score pop-ups.

Winning Conditions

- The game ends when the timer or moves are exhausted.
- The score, based on letter values of formed words, determines performance (no explicit win, as it's single-player).

Implementation and Development

Development Process

The game was developed iteratively:

1. Design: Defined the 6x6 grid, letter distribution, and hint system.
2. Grid Generation: Built smart grid logic to avoid initial words, with a fallback for rare cases.
3. AI: Implemented the greedy best-first search for hints, optimizing for speed.
4. UI/Animations: Created gradient tiles, swap animations, word highlights, and a game-over screen with Pygame.
5. Testing: Refined mechanics and AI through playtests.

Programming Languages and Tools

- Programming Language: Python
- Libraries:
    - Pygame: For rendering, animations, and event handling.
    - NLTK: For word validation (with fallback dictionary).
    - Random: For weighted letter selection.
    - Time: For timer and pauses.
    - Math: For animation calculations.
- Tools: GitHub for version control.

Challenges Encountered

- Grid Generation: Ensuring no initial words was difficult; a fallback replaced problematic letters with rare consonants.
- Animation Lag: Optimized surface rendering and frame counts for smoothness.
- AI Speed: A score cache and limited swap evaluations improved performance.
- NLTK Dependency: A fallback dictionary ensured functionality without NLTK, though limited.

Team Contributions

Team Members and Responsibilities

The team of three contributed equally:

- Ali Wasif (22K-4511): *Developed the AI hint system with greedy best-first search and score evaluation functions. Implemented the game timer logic with pause/resume functionality during animations. Created the word popping animation with particle effects and visual feedback when valid words are found. Designed the dictionary system using NLTK with an efficient fallback mechanism.*
- Siraj Ahmed (22K-4410): *Designed game mechanics, including smart grid generation and tile-swapping logic. Implemented chain reaction and core rules for seamless gameplay.*
- Shahzaib Ahmed (22K-4390): *Created the user interface, including gradient tiles, animations (swaps, highlights, game-over), and hint system visuals (score pop-ups). Ensured an engaging experience.*

All members collaborated on playtesting, documentation, and refinement, ensuring balanced contributions.

Results and Discussion

AI Performance

- Score Impact: Hints increased player scores by 20-30% in tests.
- Decision Time: Hints processed in <0.5 seconds, maintaining fluidity.
- Effectiveness: The AI suggested high-scoring swaps in 90% of cases, though focused on short-term gains.
- Feedback: Testers praised the hint system's usability and the game's challenging yet fair design.

Discussion

WordCrush delivers a strategic word puzzle game with a robust AI hint system. The greedy best-first search effectively guides players, and Pygame's animations enhance engagement. Challenges like grid generation and performance were resolved through optimization, showcasing the team's problem-solving skills.

References

- Pygame Documentation. (n.d.). Retrieved from https://www.pygame.org/docs/
- NLTK Documentation. (n.d.). Retrieved from https://www.nltk.org/
- Downey, A. B. (2015). *Think Python: How to Think Like a Computer Scientist*. O'Reilly Media.
- Russell, S., & Norvig, P. (2020). *Artificial Intelligence: A Modern Approach* (4th ed.). Pearson.