Project Title: Word Crush: An AI-Powered Word Swapping Puzzle Game
Submitted By: Ali Wasif (22K-4511), Shahzaib Ahmed (22K-4390), Siraj Ahmed (22K-4410)
Course: Artificial Intelligence
Instructor: Ms. Ravia Ejaz
Submission Date: 14 / 03 / 2025

---

1. Project Overview

Project Topic:

Word Crush is an innovative word puzzle game inspired by Candy Crush, where players swap adjacent letter tiles to form valid words. Unlike traditional word puzzles, the game incorporates AI to dynamically analyze the board and suggest or execute optimal moves.

Objective:

The primary goal of this project is to develop an **AI-driven word formation strategy**, allowing the AI to make optimal swaps using **Minimax, Alpha-Beta Pruning, and heuristic-based move evaluation**. The game will feature **real-time word validation and cascading tile mechanics**, enhancing gameplay complexity.

---

2. Game Description

Original Game Background:

This game is inspired by **Candy Crush-style match-three games**, where players swap adjacent items to trigger cascading effects. Instead of candies, **letters are swapped to form words**, and the board refills dynamically. Players must strategically plan their swaps to maximize score and chain reactions.

Innovations Introduced:

- **Letter Swapping Mechanics:** Players swap letters instead of matching identical tiles.
- **AI-Powered Move Optimization:** AI predicts the best swaps for high-scoring words.
- **Cascading Tile Refill:** When words are formed, tiles above drop down, potentially creating chain reactions.
- **Heuristic-Based AI Decision-Making:** AI prioritizes moves based on word length, letter value, and future board states.

These modifications increase **game complexity, requiring players and AI to strategize future swaps** instead of making random matches.

---

3. AI Approach and Methodology

AI Techniques to be Used:

- **Minimax Algorithm:** AI simulates multiple moves ahead to predict the best swap.
- **Alpha-Beta Pruning:** Reduces the number of board states AI evaluates, improving efficiency.

Heuristic Design:

- **Word Length Bonus:** Prioritizes longer words for higher scores.
- **Rare Letter Weighting:** Gives extra priority to high-value letters (e.g., Q, Z, X).
- **Future Move Potential:** Evaluates whether a swap can trigger cascades.

Complexity Analysis:

- **Minimax with Alpha-Beta Pruning:** O(b^d), where **b** is the branching factor (possible swaps) and **d** is the depth of search.
- **Word Validation using Trie:** O(n) lookup time for each word formed.

---

4. Game Rules and Mechanics

Modified Rules:

- Players **swap two adjacent tiles** to create valid words.
- Words must be at least **3 letters long** to be valid.
- **Cascading Effect:** Tiles above move down after word formation.
- **Power-Ups:** Special tiles like Wildcards, Letter Bombs, and Score Multipliers.

Winning Conditions:

- Players must **reach a target score** within a limited number of moves.
- Alternatively, the game can be played in **timed mode**, where the highest score within a time limit wins.

Turn Sequence:

1. **Player selects two adjacent letters** and swaps them.
2. If the swap creates a valid word, the **word disappears and new letters drop**.
3. If an invalid word is formed after the swap, the turn is wasted, and letters remain in their original positions.
4. AI takes its turn (if in PvE mode) by **analyzing the board and making the best swap**.
5. The process repeats until **no more valid moves remain or the score condition is met**.

---

5. Implementation Plan

Programming Language: Python

Libraries and Tools:

- **Pygame** (for GUI)
- **NumPy** (for array-based board operations)
- **NLTK / Trie Structure** (for word validation)

Milestones and Timeline:

Week	Task
Week 1-2	Game design, rule finalization, and board setup.
Week 3-4	Implement tile swapping, cascading mechanics, and word validation.
Week 5-6	Develop AI strategy (Minimax, heuristics, and move selection).
Week 7	Integrate AI into the game, optimize performance.
Week 8	Final testing, debugging, and report preparation.

---

6. References

- **Trie Data Structure for Fast Lookup :-** https://www.geeksforgeeks.org/introduction-to-trie-data-structure-and-algorithm-tutorials/