GULLIVER'S GAMES

Software Engineer

Take-Home Coding Challenge

We're so glad to have you consider us for your next job! This document will provide you with all the information you need regarding your application project. Let us know if you have any questions!

INTRODUCTION

Abstract

You will be presented with separate challenges. Some of them are mandatory, some of them optional. The challenges vary in difficulty and each challenge serves as a different metric for measuring your capabilities. You're not expected to complete the optional challenges, however it's recommended that you complete them if you believe they'll provide more insight into your skill set. You will be provided with the required links at the end of this document.

What's Expected Of You

For a mid-senior developer, our main criterias are: *your general programming skill*, *your ability to improve upon our systems/plugins and how much initiative you take for this purpose*, *your adaptability to different environments*. For this purpose, we've provided you with challenges that will test your skills in **code architecture**, **abstract thinking** and **improvised research.** Our secondary criteria is your *general know-how in Unity*.

Evaluation

Evaluation will be based on how you compare against other applicants, the quality of your code and your potential to improve. If you find the challenges *too hard* and you fail to complete them, you should still do as much as you can. Your mindset is just as important to us as your results!

THE CHALLENGE

Your task is to create a simple, level-based, mobile word game called *Word Puzzle*. You will develop the gameplay mechanics and an auto-solver that will determine the highest score possible for each level. You are to use Unity and C# in your project.

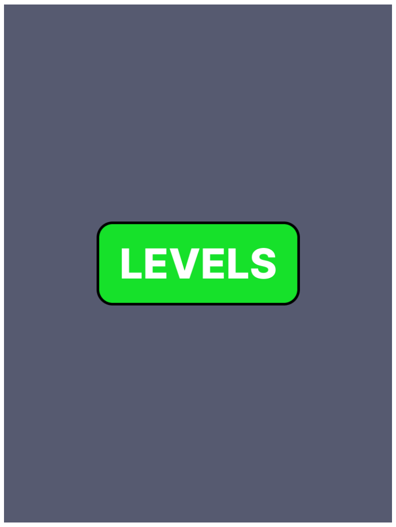
Before You Get Started

- You must use **Unity 2021.3.18f** and **C#**
- You may use third-party tween libraries for animations
- The game must support portrait mode. Landscape support is optional.
- The screenshots provided were taken on iPhone 11 in the Device Simulator, but the game must work on other devices too
- The screenshots are only given as samples, the visuals may not match as long as the functionality is implemented correctly

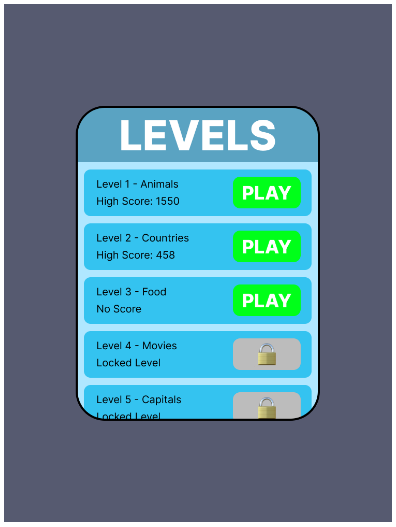
Flow

- Upon launch, the user will see a levels button
- Tapping the levels button will bring up a level selection screen
- Each item, which represents the level, should show this information
    - Level Number
    - Title
    - Highest score
    - **If the level is unlocked:** A green play button that takes the user to the level
    - **If the level is locked:** A gray lock button. A level is only unlocked if the level before is cleared
- After tapping the play button on an unlocked level, that level will be loaded
- **When a user finishes the level with a new highest score:**
    - Celebration particles and animation will be shown to the user
    - After the celebration ends, the level selection screen will appear automatically
    - The newly unlocked level will have a green play button as opposed to a gray lock button
- **When a user finishes the level without a new highest score:**
    - The level selection screen will appear automatically

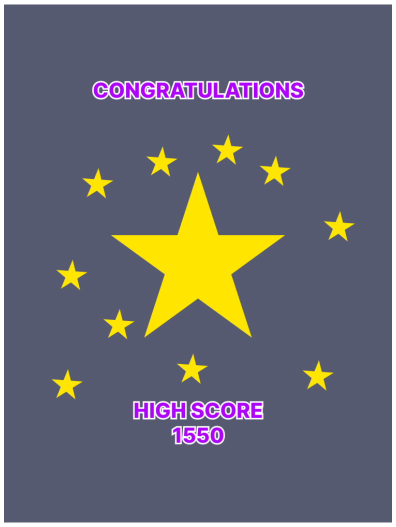
Screenshots



Main screen



Level selection screen



Celebration sequence

Gameplay

The goal of *Word Puzzle* is to get the highest score possible by forming words from tiles with letters on them. A tile can only be used if there are no other tiles above it. The level ends if you cannot form any more words from the dictionary.

Mechanics

- Tapping a tile moves it from the board to the word forming area
  - A tile can only be tapped if there are no other tiles above it
  - Tiles that can't be tapped are tinted gray
  - Information about what tiles a tile blocks is provided within the level data
- The user can submit a word once they are happy with it
  - The submit button must only be active if the word is valid, i.e. exists in the dictionary and has not been submitted before
- The user can undo their moves, in case they make a mistake
  - Tapping undoes only the last move
  - Holding undoes all moves until the word submission
  - The undo button must only be active if an undo is possible
- The level ends when there are no tiles left or no valid word can be formed from the remaining tiles

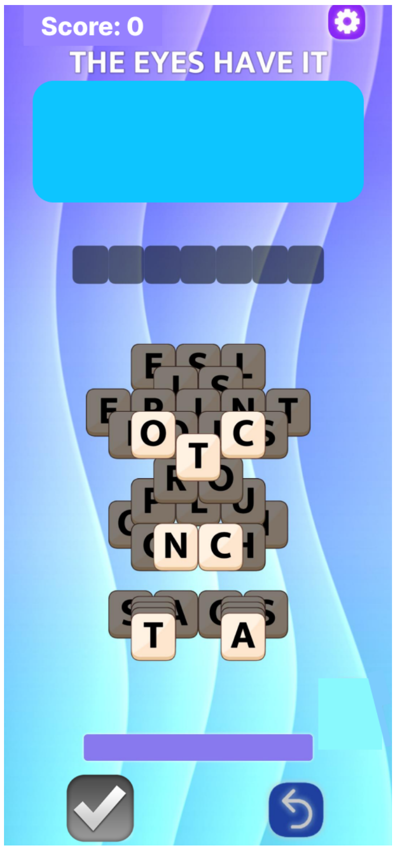
Scoring

- Each letter has a non-zero value (a la Scrabble), that is l_{ij} (i = word index, j = letter index)
  - 1 Point: E, A, O, N, R, T, L, S, U
  - 2 Points: D, G
  - 3 Points: B, C, M, P
  - 4 Points: F, H, V, W, Y
  - 5 Points: K
  - 8 Points: J, X
  - 10 Points: Q, Z
- Each submitted word of length k_i gives $10 \cdot k_i$ times the total value of the word points
- Each unused letter at the end of the level takes 100 points away

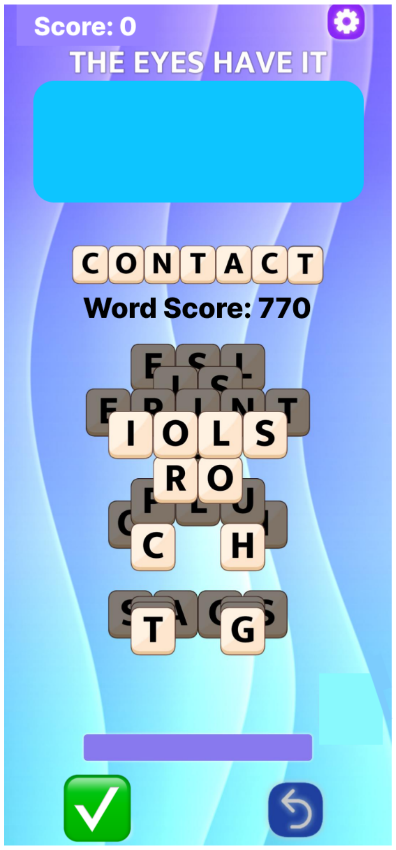
$$score = \sum_{i=1}^{n} \sum_{j=1}^{k_i} 10 \cdot k_i \cdot l_{ij} - 100 \cdot u$$

- Save the score as the new highest score if it's higher than the last highest score or if there's no high score set, and display the celebration sequence

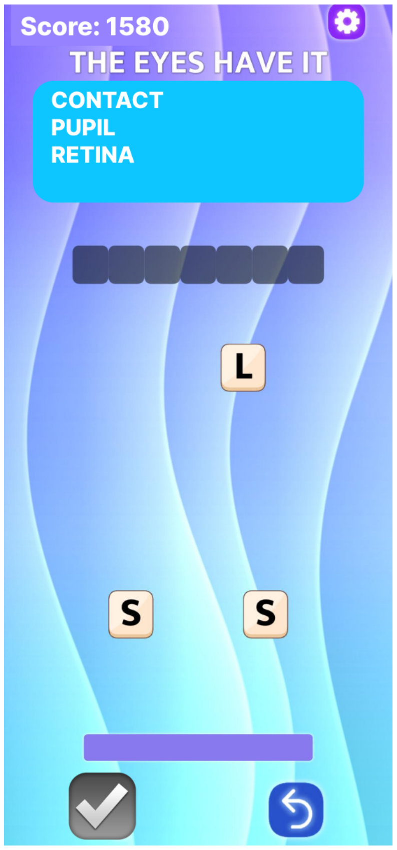
Screenshots



Level start



Before word submission



Level end

Auto-Solver

We want to be able to calculate the highest score possible for each level. However, doing this by hand proves impossible on bigger levels. Write an algorithm that will calculate the highest score possible and the moves to achieve this score on a given level. Add an auto-solver button on the gameplay screen. Pressing that button will play out the highest scoring scenario automatically.

The auto-solver algorithm must be able to work on its own without a UI. You can simply print the moves and the highest score to the console in the no UI mode. It should take a few seconds at most to calculate the optimal play. You can still submit a slower version if you can't optimize it further. A slow algorithm that gives an answer is better than no answer, after all. The levels' tile counts increase as level number goes up so later levels will take longer, but the run time should not exceed 2 seconds, if possible, at the largest level.

Levels

The level files are presented in a json format as follows:

- title (string)
- tiles (array of tile objects)
    - id (integer)
    - position (position of the tile in 3D space)
        - x (float)
        - y (float)
        - z (float)
    - character (string)
    - children (array of integers - ids of tiles that this tile blocks)

You can download the level files here

Dictionary

The dictionary is a sorted list of English words in txt format.

You can download the dictionary file here

How to Get Help

Feel free to reach out if you have any questions. You can add **bartutiryaki#8165** on Discord and ask your questions.

Time Estimate

About 12 hours depending on your overall programming experience, and also your familiarity with the Unity game engine.