Описание

Задание представляет из себя проект на юнити, который находится в zip архиве. В проекте есть три сцены. Для каждой из сцен необходимо дописать недостающий функционал так, чтобы при их запуске не падало никаких ошибок, и сцены работали согласно описанным к ним заданиям.

Задание “Филлворды”

Сцена отображает уровни из ресурсов игры. Кнопки влево/вправо циклически переключают уровни, доступные в папке. Количество уровней для переключения задается из конфигурации (настройка на игровой сцене). Игра представляет из себя квадратную (ширина равна высоте, например 3x3, 4x4 и т.д.) сетку символов, на которой игрок сможет находить слова.

Экран филлворды

Задачи:

- реализовать загрузку и парсинг уровней (функция **ProviderFillwordLevel.LoadModel**).
- уровни не гарантировано верные. В файлах с уровнями могут быть ошибки - предусмотреть их и обработать. Если не удалось корректно распарсить уровень, то вернуть null.
- реализовать функционал, что если не удалось загрузить уровень, то пробуем загрузить следующий. Если не удалось загрузить ни один из уровней - кидаем ошибку (для данного задания можно кинуть базовую ошибку throw new Exception()).

Формат уровней

Уровни представлены файлом словаря со всеми доступными в игре словами и файлами самих уровней. Формат данных в уровне подразумевает, что сетка будет квадратной всегда.

Если исходя из данных нельзя заполнить квадратную сетку уровня - то уровень невалидный.

Невалидный уровень если:

- кол-во символов в уровне не возможно уложить в квадратную сетку так, чтобы не было ни пустых и лишних символов
- в данных уровня в есть символы которые ссылаются на одну и ту же клетку
- уложив все символы по своим клеткам остаются пустые клетки
- слова из словаря по индексу не совпадает по длине с индексами из уровня
- есть индексы которых не может быть на данном уровне (например кол-во символов 9 на уровне что значит что сетка 3x3, но есть индекс 15 - очевидно такого индекса не может быть)

Словарь представляет из себя файл, в котором каждая строка содержит слово. Уровни описаны в файле pack_0.txt. Каждый уровень - это строка файла.

0 0;1;3;2 - пример описания уровня с одним словом.
29 8;7;6 30 1;2;5 31 0;3;4 - пример уровня с тремя словами.

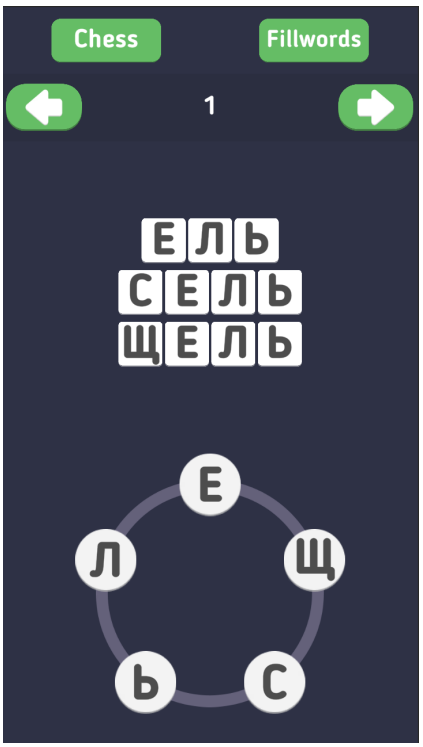
Во втором примере 29, 30, 31 - номера слов в словаре. После каждого слова содержится описание позиций символов на игровой сетке, разделенных точкой с запятой. Номера слов и список позиций разделяются пробелами.

0	1
2	3

Пример индексов на игровой сетке

Задание “Поиск слов”

Сцена визуализирует уровни, которые лежат в ресурсах проекта. Кнопки влево/вправо циклически переключают уровни, доступные в папке.



Экран поиск слов

Задачи:

- реализовать загрузку и парсинг уровней (функция **ProviderWordLevel.LoadLevelData**);
- реализовать алгоритм получения букв для игровых кнопок (функция **FactoryLevelModel.BuildListChars**).

При составлении слова каждая буква на панели ввода будет использована максимум один раз. На панели должно быть столько символов, сколько нужно для сборки всех загаданных слов. Лишних символов быть не должно.

Примеры:

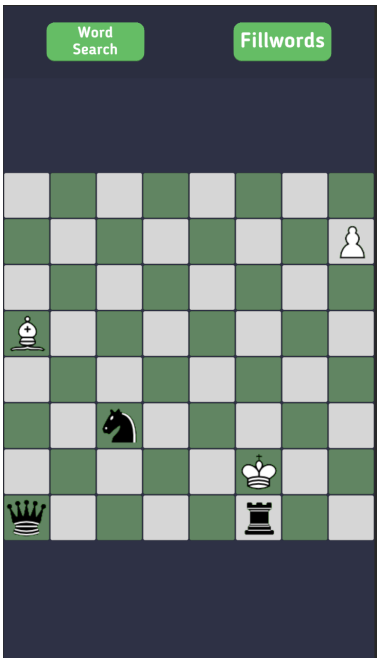
Заданные слова	**Набор символов**
мама, папа	м м а а п п
кот рот лот	к т р о л

Путь к уровням в архиве: **App/Resources/WordSearch/Levels/**

Задание “шахматы”

Сцена отображает поле-сетку и шахматные фигуры на ней. Фигура выбирается кликом на нее. Далее кликом по любой свободной клетке фигура анимировано перемещается в эту клетку, если для этого типа фигуры существует путь в эту клетку.

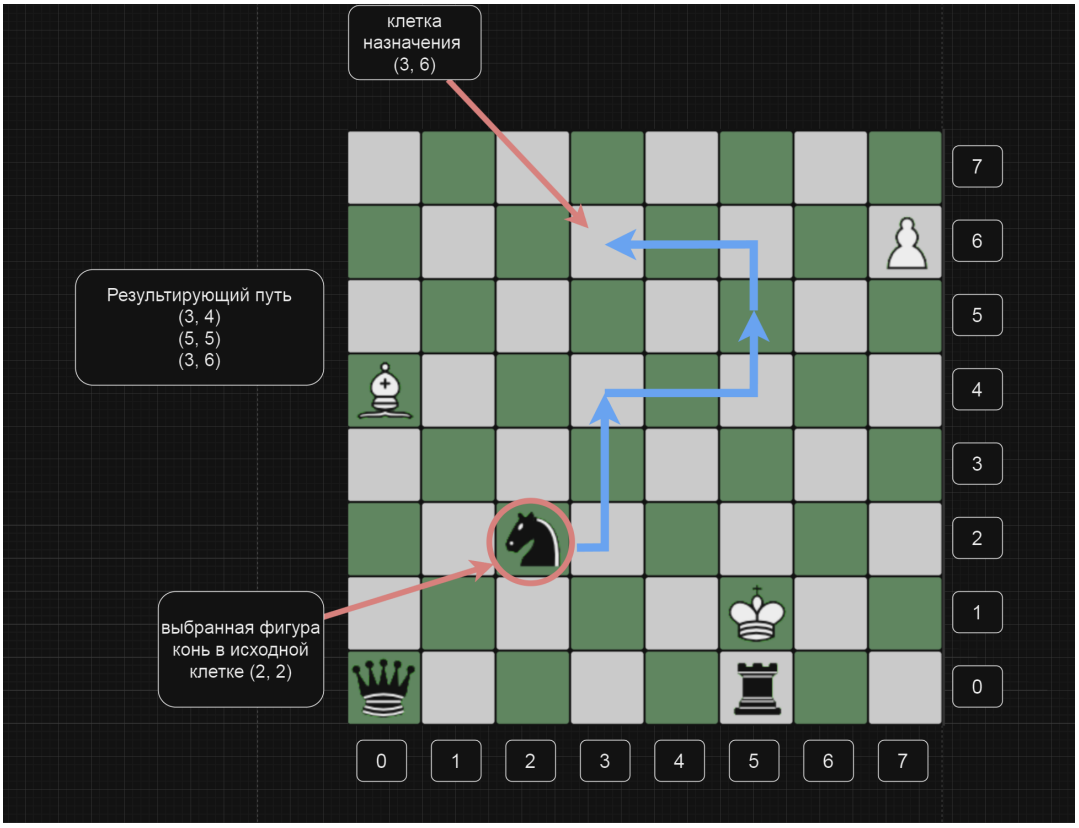
Фигуры должны перемещаться согласно правилам шахмат (смотреть википедию). Путь фигуры до клетки может формироваться из нескольких последовательных ходов.



Экран шахматы

Задача

- реализовать алгоритм поиска пути для всех фигур (**ChessGridNavigator.FindPath);**
- предполагаем, что белые всегда снизу - так белая пешка ходит всегда только вверх, а черная - только вниз;
- путь должен быть минимальным относительно количества ходов фигуры;
- для анимации вычисляется путь, состоящий из клеток, по которым фигура может последовательно прийти в точку назначения. Если пути нет, возвращается **null**;
- фигура не может наступать на другие фигуры ни во время пути, ни в конечной точке.



Пример

Выделена фигура коня, и клик по клетке (3, 6). Находим минимальный возможный путь к искомой клетке. Стартовую клетку в результат не включаем.

Общие требования

- версия юнити 2021.3.21f1;
- рабочий проект залить на публичный гит;
- приложение должно работать без ошибок;
- приложение должно собираться и работать на телефоне (под андроид);
- для парсинга JSON можно использовать сторонние библиотеки, при выполнении всех остальных пунктов (приложение собирается и работает на устройстве)
- **НЕ МЕНЯТЬ** уже написанный код. Можно только дописывать свой код: методы, классы и тд. Сигнатуры методов уже написанного кода **НЕ МЕНЯТЬ**;
- уделить внимание корректности своих решений и их проверке.