ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
ФГАОУ ВО НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Факультет компьютерных наук
Образовательная программа «Прикладная математика и информатика»

Отчет о программном проекте на тему:

Валидный сапер

Выполнил:
студент группы БПМИ218
Андрян Тигран Каренович


(подпись)

25.05.2023
(дата)

Принял руководитель проекта:
Сосновский Григорий Михайлович
Научный сотрудник
Факультета компьютерных наук НИУ ВШЭ


(подпись)

25.05.2023
(дата)

Москва 2023

Содержание

1 Аннотация

Сапер – игра-головоломка, созданная разработчиком Яна Эндрю [5], выпущенная на персональный компьютер Sinclair ZX81 в 1983 году и написанная на языке BASIC.

1.1 Принцип игры

Плоское или объёмное игровое поле разделено на смежные ячейки (квадраты, шестиугольники, кубы и т. п.), некоторые из которых «заминированы»; количество «заминированных» ячеек известно. Целью игры является открытие всех ячеек, не содержащих мины.

Игрок открывает ячейки, стараясь не открыть ячейку с миной. Открыв ячейку с миной, он проигрывает. Мины расставляются после первого хода, поэтому в новых версиях проиграть на первом же ходу невозможно. В первой версии (Windows 95-Windows XP) довольно частая ситуация, что под первой открытой ячейкой оказывалась мина. Если под открытой ячейкой мины нет, то в ней появляется число, показывающее, сколько ячеек, соседствующих с только что открытой, «заминировано» (в каждом варианте игры соседство определяется по-своему); используя эти числа, игрок пытается рассчитать расположение мин, однако иногда даже в середине и в конце игры некоторые ячейки всё же приходится открывать наугад. Если под соседними ячейками тоже нет мин, то открывается некоторая «не заминированная» область до ячеек, в которых есть цифры. «Заминированные» ячейки игрок может пометить, чтобы случайно не открыть их. Открыв все «не заминированные» ячейки, игрок выигрывает.

2 Ключевые слова

Framework(фреймворк) – программная платформа, определяющая структуру программной системы; программное обеспечение, облегчающее разработку и объединение разных компонентов большого программного проекта.

ЯП – сокращение от язык программирования

Интрейфейс – Совокупность средств, обеспечивающих взаимодействие функциональных устройств и/или программ в вычислительной системе (компьютере), а также взаимодействие их с пользователем.

UI – user interface

GitHub – крупнейший веб-сервис для хостинга IT-проектов и их совместной разработки.

BFS(Поиск в ширину) – один из методов обхода графа. Пусть задан граф $G = (V, E)$ и выделена исходная вершина s. Алгоритм поиска в ширину систематически обходит все ребра G для «открытия» всех вершин, достижимых из s, вычисляя при этом расстояние (минимальное количество рёбер) от s до каждой достижимой из s вершины. Поиск в ширину имеет такое название потому, что в процессе обхода мы идём вширь, то есть перед тем как приступить к поиску вершин на расстоянии $k+1$, выполняется обход вершин на расстоянии k.

Dart – язык программирования, созданный Google.

Flutter – комплект средств разработки и фреймворк с открытым исходным кодом для создания мобильных приложений под Android и iOS, веб-приложений, а также настольных приложений под Windows, macOS и Linux с использованием языка программирования Dart.

Widgets – центральная иерархия классов в структуре Flutter. Виджет - это неизменное описание части пользовательского интерфейса. Виджеты могут быть расширены до элементов, которые управляют базовым деревом рендеринга. Сами виджеты не имеют изменяемого состояния (все их поля должны быть заполнены).

StatefulWidget – Виджеты, которые хранят состояние, (stateful) в процессе работы приложения могут изменять свои свойства динамически. Состояние хранится в виде объекта класса State.

StatelessWidget – Виджеты, которые не имеют состояния, (stateless) в процессе работы приложения не изменяют своих свойств. Такие виджеты не имеют состояния. Они могут изменяться лишь посредством внешних событий, которые возникают на родительских виджетах-контейнерах.

Валидное поле – поле называется валидным, если в течение игры можно однозначно определить, где располагаются все мины.

Cell – ячейка в поле.

Группа – набор ячеек.

Солвер – алгоритм решения сапера.

СД – сокращение от Структура Данных.

MS – сокращение от Minesweeper Solver.

Widget Provider – виджет, который дает возможность изменять глобальное состояние во всем программе.

3 Введение

Сапер – популярная компьютерная игра, которая требует от игрока логического мышления и умения принимать решения на основе предоставленной информации. Цель игры заключается в раскрытии всех безопасных клеток на игровом поле, избегая взрывов мин. Сапер является классическим представителем жанра "головоломки"и широко известен благодаря включению в стандартные компьютерные операционные системы, такие как Windows. В данной курсовой работе мы будем рассматривать тему "Валидный сапер"и изучать основные аспекты создания игры сапер с соблюдением правил и логики, характерных для классической версии. Мы будем анализировать игровой процесс, включая устройство игрового поля, правила открытия клеток и обозначения потенциальных мин, а также различные уровни сложности и варианты генерации мин на поле. Основной целью данной работы является разработка и реализация валидного сапера, то есть игры, которая будет следовать всем правилам и ограничениям, не допуская неправильной расстановки мин и обеспечивая возможность решения головоломки в рамках заданной логики. Мы также будем исследовать различные подходы к созданию и оптимизации игрового процесса, включая использование алгоритмов для раскрытия безопасных клеток и определения расположения мин. Данная работа предоставит полное понимание о создании валидного сапера и позволит разработать качественную игру, которая будет вызывать интерес и представлять вызов для игроков, поощряя их логическое мышление и принятие решений на основе доступной информации.

4 Обзор Литературы

Существуют три основных подхода к реализации игры "Валидный Сапер". Первый подход основан на использовании заранее известного списка валидных полей. Однако у такой реализации имеется три значительных недостатка. Во-первых, создание нового уровня становится трудоемким процессом, поскольку необходимо вручную разрабатывать и вносить новые валидные поля. Во-вторых, такой подход требует значительных ресурсов памяти для хранения большого количества валидных полей. В-третьих из-за ограниченного количества полей, пользователю могут попадаться одни и те же расстановки, что будет вызывать недовольство.

Второй подход [6] предполагает генерацию игрового поля динамически во время игры. Однако и у этого подхода есть свои недостатки. Например, требуется разработка более сложной архитектуры проекта, способной отличать валидные поля от невалидных. Также необходимо обратить внимание на производительность программы, чтобы пользователь не замечал процесса генерации поля и игровая среда работала достаточно быстро. В приведенной реализации была обнаружена ошибка, в результате чего было отправлено обращение к автору статьи. Кроме того, возможно оптимизировать некоторые части кода, что приведет к экономии ресурсов и повышению скорости работы.

Третий подход [9] заключается в тщательном запоминании всех основных паттернов, характерных для игры в сапёра. Самые опытные и лучшие игроки способны уверенно распознавать и использовать эти паттерны наизусть, что позволяет им эффективно решать даже самые сложные уровни менее чем за минуту [1]. Однако, несмотря на это, данный подход также обладает схожими недостатками, которые были отмечены ранее в первом методе.

5 Динамическая генерация поля

5.1 Ячейка

5.1.1 Структура ячейки

Для определения состояния каждой ячейки необходимо задать её координаты, проверить наличие бомбы, установленного флага, открытости ячейки и количество бомб вокруг неё. Эти данные играют важную роль в правильной отрисовке текущего состояния игрового поля. Используя эту информацию, можно корректно отображать визуальные элементы, свя-

занные с каждой ячейкой, чтобы пользователь мог наглядно видеть и взаимодействовать с текущим состоянием игры.

5.1.2 Отрисовка ячейки

Ячейка представляет собой виджет, который может быть изменен в любой момент времени. Однако, при отрисовке ячеек могут возникать проблемы, такие как неправильное расположение количества мин внутри ячейки или выход за границы. Для решения этих проблем была разработана функция, которая корректирует размер ячейки в зависимости от размера поля, обеспечивая правильную отрисовку.

5.2 Поле

5.2.1 Структура поля

Игровое поле представляет собой двумерную таблицу ячеек, дополненную некоторой информацией, которая помогает сделать код более читаемым и структурированным. Основными методами работы с полем являются генерация поля, открытие ячеек и проверка на корректность.

Генерация поля отвечает за создание начального состояния, включая расположение и определение количества мин вокруг каждой ячейки. Этот процесс осуществляется с использованием рандомизированного алгоритма, чтобы гарантировать разнообразие уровней.

Проверка на корректность обеспечивает правильность состояния игрового поля и его соответствие правилам игры. Этот шаг включает создание новых структур, правил и алгоритмов, которые будут обсуждены позже.

Открытие ячеек представляет собой основной механизм игрового процесса, который позволяет игроку раскрывать ячейки и взаимодействовать с содержимым поля. При открытии ячейки запускает алгоритм BFS, который позволяет быстро вскрыть все доступные ячейки.

5.2.2 Отрисовка Поля

Отметим, что в данной реализации нет необходимости отрисовывать поле, поскольку ячейки сами по себе содержат всю необходимую информацию и отвечают за свою собственную отрисовку.

5.3 Валидация

5.3.1 Постановка задачи, проблемы и возможные решения

Цель состоит в разработке комплексного алгоритма и соответствующих классов с целью осуществить проверку валидности поля или его содержимого.

В ходе разработки алгоритма возникло множество проблем, включая неправильность самого алгоритма. Одной из первых идей, которая возникла, было построение разреженного графа. В этом графе вершины связаны ребрами таким образом, что одна вершина содержит мину, а другая - нет. По сути, ячейки с минами становятся точками сочленения, что приводит нас к применению алгоритма поиска точек сочленения. После проведения ряда тестов и анализа примеров стало видно, что предложенный алгоритм не является подходящим для решения задачи.

Было принято решение использовать итерационный алгоритм, основанный на методе из статьи[6], однако он требовал некоторых доработок и поправок. Для успешной реализации этого алгоритма потребовалось создать новые классы с методами [11].

5.3.2 Документация к коду

Группа

- добавить элемент в группу - addCell
- Проверить, что 2 группы эквиваленты - isEqual
- Проверить, что одна группа является подмножеством другой - isSubset
- Взять пересечение 2 групп - intersection
- Удалить из группы подмножество - deleteSubset

Поле

- размер
- список ячеек
- количество мин в группе

Построение группы

Группа будет строиться вокруг ячейки, а именно берутся все непомеченные закрытые ячейки, которые граничат с исходной, и добавляются в группу.

Нахождение мин

Имея такую СД, не трудно найти не заминированные ячейки. Если количество мин равно нулю, то можно открывать ячейки из группы. Если количество мин равно размеру группы, то во всех ячейках есть мина.

Итерационный алгоритм

- проходимся по всем открытым ячейкам и строим группы.
- пересекаем и разбиваем на меньшие подгруппы
- определяем бомбы и пустые ячейки
- повторяем, пока есть изменения

Валидность

Если после завершения итерационного алгоритма сумма открытых ячеек и мин равняется размеру игрового поля, то расстановка объявляется валидной.

5.3.3 Валидность алгоритма

Утверждение: Пусть задана ячейка, которую пользователь хочет открыть первой. Тогда поле является решаемым из заданной позиции $\iff$ MS объявляет поле валидным.

$\Leftarrow$: Игрок может повторить действия алгоритма, которые приведут его к победе. Из чего следует, что поле валидно из заданной позиции.

$\Rightarrow$: Что означает, что поле является валидным? Это означает, что в любой момент времени мы можем детерминировано определить либо ячейку с миной, либо пустую ячейку. Предположим, что мы каким-то образом поняли, что в ячейке нет мины. Это означает, что перебором всех вариантов расстановок можно определить, что находится в интересующей ячейке. А MS построен на переборе, только с определенными эвристиками, которые дают подходящую нам асимптотику. Из чего следует, что мы можем определить, что находится в искомой ячейке.

5.3.4 Почему такая реализация?

Одним из ключевых факторов, почему была выбрана данная реализация, является удобство написания тестов и отладки благодаря простоте самого кода. Это облегчает процесс разработки и обеспечивает более эффективное тестирование. Кроме того, благодаря особой структуре хранения ячеек в группе, где они отсортированы в определенном порядке,

достигается высокая производительность и быстродействие обработки поля. Это позволяет эффективно решать задачу проверки валидности поля или его содержимого.

5.3.5 Асимптотика:

Саму асимптотику довольно трудно вычислить, но если говорить а наихудшем случае, то можно сделать такие выводы: нам понадобится $8n^2$ операций, чтобы пройтись по всему полю и построить группы внутри одной итерации. После нам нужно пересечь все группы, групп не больше, чем количество самих ячеек, поэтому будем учитывать, что их n^2. Для перебора всех пар групп и применения новой информации нам понадобится $O(n^4)$ операций. Максимальное количество итераций - количество бомб, то есть наша асимптотика имеет вид $O(n^4 * bombAmount)$, где n - размер поля.

5.3.6 Статистика

Подсчитаем, сколько в среднем MS тратит времени на то, чтобы решить поле. В среднем, чтобы сгенерировать поле размера 5 на 5 и проверить его на валидность нужно 7 миллисекунд и 3 итерации алгоритма. Для полей 7 на 7 понадобилось в среднем 10 миллисекунд и 4 итерации, а для полей 12 на 12 в среднем 25 миллисекунд и 8 итераций. Это дает понять, что на самом деле MS хорошо справляется со своей задачай. К тому же, человек способен заметить задержку от 100 миллисекунд, то есть игрок не заметит, что поле появляется не сразу.

5.3.7 Потенциал оптимизаций

Основная вычислительная нагрузка алгоритма состоит в составлении и переборе всех пар групп. Однако алгоритм был специально разработан с учетом возможности применения многопоточности или многопроцессорности для ускорения вычислений при обработке больших таблиц. Также для сокращения потребления памяти следует переписать класс ячеек, ведь все поля этого класса можно поместить в один объект типа int, использую bitset. Это позволит сократить на порядок расходы на память.

6 Пользовательский Интерфейс

6.1 Ячейки

Интерфейс у ячеки примитивен, игрок может либо открыть ячейку, либо поставить на нее флаг. Оба эти события вызовут функцию build из cell.dart [12], где в зависимости от события будет нарисован флаг или же произойдет отрисовка внутренности ячейки.

6.2 Начальная страница

На начальной странице приложения пользователь может выбрать одну из трех цветовых раскрасок. Чтобы применить выбранный цветовой стиль ко всем виджетам в приложении, используется виджет Provider[10], который позволяет автоматически уведомлять все дочерние виджеты о любых изменениях. Кроме того, на главной странице доступны две кнопки - "Выбор уровня"и "Начать игру". Система не позволяет пользователю начать игру до выбора уровня сложности. За эту проверку отвечает специальное всплывающее окно Alert. Выбор уровня сложности реализован в виде кольца, которое можно бесконечно вращать. После выбора уровня сложности и нажатия кнопки "Начать игру пользователь переходит на вторую страницу, где отображаются игровые ячейки и дополнительные инструменты.

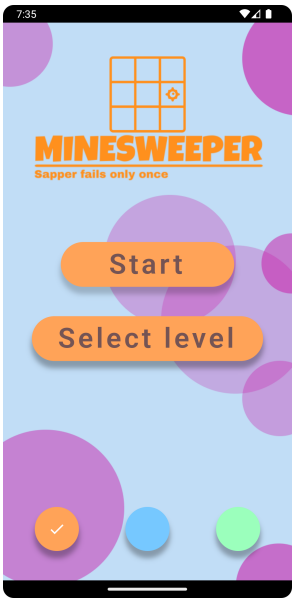

Рис. 6.1: Скриншот начального экрана

6.3 Игровое меню

В игровом меню доступны счетчик флагов, само поле, а также подсказки, которые помогут пользователю легче справиться с задачей.

6.3.1 Флаги

Счетчик флагов также реализован с использованием provider, поскольку он инициализируется и изменяется в различных виджетах.

6.3.2 Подсказки

Если пользователь в какой-то момент не может определить пустую ячейку, то у него есть возможность воспользоваться встроенными подсказками, которые становятся доступными после просмотра рекламного контента. В приложении предусмотрено три типа подсказок. Первый тип подсказки показывает местоположение мины и автоматически открывает ячейку с красным крестом. Второй тип подсказки позволяет пользователю оценить вероятность наличия мин вокруг выбранной им ячейки. Эта подсказка особенно полезна на самых сложных уровнях, когда поле специально создано с невалидными конфигурациями для повышения сложности игры. Третий тип подсказки позволяет пользователю открыть область на поле. При использовании этой подсказки запускается специальный алгоритм, который определяет область, в которой безопасно открывать ячейки. Таким образом, встроенные подсказки помогают пользователю в решении сложных ситуаций и повышают интерес игры.

Рис. 6.2: Скриншот игрового меню

7 Результаты

В итоге разработки приложения удалось достичь привлекательного и эстетичного дизайна. Однако, несчастливым обстоятельством стало отсутствие рекламы, так как не удалось разобраться в ее настройке. При визуальном осмотре приложения на фотографии, представленной на рисунке 6.3.2, видно, что на нижней части экрана присутствует рекламный баннер, однако он остается пустым. При анализе журналов событий можно установить, что данная проблема возникает из-за ошибки при установлении соединения с сервером, необходимым для отображения рекламного контента. В качестве временного локального решения было предложено использование VPN для подключения, однако невозможно интегрировать VPN непосредственно в само приложение, что приводит к некорректной работе рекламы.

Список литературы

[1] Уровень эксперт за 30 секунд. URL: https://shazoo.ru/2020/09/19/99730/12-letnij-igrok-v-sapyor-postavil-rekord-za-30-sekund-zakryl-uroven-ekspert.

[2] Dart Documentation. URL: https://dart.dev/.

[3] Flutter Documentation. URL: https://flutter.dev.

[4] Dart Packages. URL: https://pub.dev/.

[5] Wikipedia. URL: https://ru.wikipedia.org/wiki/%D0%A1%D0%B0%D0%BF%D1%91%D1%80_(%D0%B8%D0%B3%D1%80%D0%B0).

[6] Александр Власов. *Алгоритмы логики бота для игры «Сапёр»*. URL: https://habr.com/ru/articles/211188/ (дата обр. 02.04.2023).

[7] Ответы на все вопросы. URL: https://stackoverflow.com/.

[8] GitHub репозиторий с кодом. URL: https://github.com/tigran-edu/course_work.

[9] Паттерны. URL: https://givi.olnd.ru/msweeper/index.html.

[10] Андрян Тигран. *Провайдер*. URL: https://github.com/tigran-edu/course_work/blob/main/lib/stuffs/providers/theme_provider.dart (дата обр. 24.05.2023).

[11] Андрян Тигран. *Реализация валидации*. URL: https://github.com/tigran-edu/course_work/tree/main/lib/research (дата обр. 24.05.2023).

[12] Андрян Тигран. *Реализация ячейки*. URL: https://github.com/tigran-edu/course_work/blob/main/lib/grid/cell.dart#L34 (дата обр. 24.05.2023).