|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ДЕПАРТАМЕНТ ОБРАЗОВАНИЯ ИВАНОВСКОЙ ОБЛАСТИ  Областное государственное бюджетное профессиональное образовательное учреждение  «Ивановский промышленно-экономический колледж» | | | | | | |
|  | | | | | | |
| **КУРСОВОЙ ПРОЕКТ** | | | | | | |
|  | | | | | | |
| **Разработка мобильной игры «Переливашка»** | | | | | | |
|  | | | | | | |
| **ИВПЭК. 09.02.07. 22** | | | | | | |
|  | | | | | | |
|  | | **Специальность:** | | | **09.02.07 Информационные системы и программирование,**  **базовая подготовка** | |
|  | | | | | | |
|  | | | | | | |
|  | | | | | | |
|  | | | | | | |
|  | | | | | | |
| Руководитель курсового проекта | | |  | | | А.С. Горелов |
| Выполнил обучающийся группы 407а | | | |  | | Д.Д. Груздев |
|  | | | |  | |  |
|  | | | | | | |
|  | | | | | | |
| Курсовой проект выполнен и защищен с оценкой\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | | | | | |
|  | | | | | | |
| «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2024 г. | | | | | | |
|  |  | | | | | |
|  | Иваново 2024 | | | | |  |

Содержание

[Введение 3](#_Toc192141492)

[1 Концептуальное проектирование 4](#_Toc192141493)

[1.1 Теоретический вопрос 4](#_Toc192141494)

[1.2 Спецификация требований программного обеспечения 12](#_Toc192141495)

[2 Техно-рабочий проект 16](#_Toc192141496)

[2.1 Алгоритм решения поставленной задачи программного продукта 16](#_Toc192141497)

[2.2 Обоснование выбора средств разработки 17](#_Toc192141498)

[2.3 Разработка макета приложения 18](#_Toc192141499)

[2.4 Описание разработки приложения 22](#_Toc192141500)

[3 Рабочая документация 30](#_Toc192141501)

[Заключение 30](#_Toc192141502)

[Список использованной литературы 31](#_Toc192141503)

[Приложение А 32](#_Toc192141504)

Введение

Программирование занимает ключевую позицию в современном мире, предоставляя разнообразные инструменты и технологии для создания интерактивных приложений и игр. Одним из наиболее увлекательных направлений программирования является разработка игр, которая позволяет не только развлекать пользователей, но и развивать их навыки логического мышления, реакции и креативности.

Создание игр, таких как "переливайка", становится все более популярным среди разработчиков. Эта игра, основанная на решении головоломок, предлагает пользователям уникальный опыт, требующий стратегического мышления и планирования. Разработка игры требует глубокого понимания игровых механик, дизайна уровней и пользовательского интерфейса, чтобы обеспечить интуитивное и увлекательное взаимодействие с игроками.

Целью моего курсового проекта является разработка игры "переливайка", которая будет привлекать пользователей своей простотой и интересными задачами. Основные задачи проекта включают:

* Изучение принципов разработки игр и использование игровых движков для создания интерактивного опыта.
* Проектирование структуры игры с учетом удобства и интереса пользователя.
* Реализация игрового процесса, включая механики переливания жидкостей и решения головоломок.
* Создание уровней с различной сложностью, чтобы удерживать интерес игроков и стимулировать их к дальнейшему прогрессу. Тестирование игры для обеспечения стабильной работы и комфортного игрового опыта.

1 Концептуальное проектирование

1.1 Теоретический вопрос

Концептуальное проектирование является основополагающим этапом в разработке программного обеспечения, направленным на определение базовых аспектов архитектуры и функциональности будущего приложения. Оно позволяет разработчику структурировать цели и задачи проекта, а также уточнить основные требования, что закладывает прочную основу для всех дальнейших этапов разработки.

Концептуальное проектирование позволяет сформулировать общее представление о том, каким должно быть приложение, как оно будет работать, какие проблемы пользователей оно будет решать и какие ключевые функции будут его отличать. Этот процесс помогает избежать излишней сложности и неопределенности на этапе разработки, а также минимизировать риски, связанные с возможными изменениями в структуре проекта на более поздних этапах.

Концептуальное проектирование играет важную роль на ранних этапах создания приложения, так как оно позволяет:

1. Выявить и формализовать цели проекта: Этот этап позволяет четко определить, зачем создается игра, какую основную задачу она должна выполнять и какие потребности игроков она будет удовлетворять. В нашем случае, цель разработки – создание увлекательной и интеллектуальной игры, которая развивает логическое мышление и внимательность, а также приносит удовольствие от процесса решения задач.
2. Определить необходимую функциональность: Концептуальное проектирование помогает выявить основные функции, которые необходимы для реализации цели проекта. В случае с игрой "Переливайка" это такие функции, как создание уровней с различной сложностью, система подсказок, управление игровыми ресурсами (например, количеством доступных переливаний) и возможность отслеживания прогресса игрока. На этом этапе продумывается, каким образом игрок будет взаимодействовать с игрой, чтобы эти функции были максимально удобны и интуитивно понятны.
3. Описать архитектурные подходы: Концептуальное проектирование позволяет продумать подход к организации данных и логике игры. Это включает выбор подходящих моделей данных для хранения информации об уровнях, состоянии игры и прогрессе игрока, определение методов доступа к этим данным и проектирование базовых взаимодействий между элементами интерфейса. Например, в данной игре хранение данных о уровнях будет организовано с помощью базы данных, а доступ к игровым данным будет осуществляться через специальные классы, обеспечивающие выполнение таких операций, как добавление, изменение и получение информации об уровнях и состоянии игры.
4. Закрепить базовые принципы интерфейса: Концептуальное проектирование также позволяет обдумать ключевые элементы интерфейса, с которыми будет взаимодействовать игрок. В данном проекте это означает разработку интуитивно понятного и привлекательного интерфейса, где игроку будет удобно начинать новые уровни, получать подсказки и отслеживать свой прогресс. На этом этапе определяются основные принципы расположения элементов интерфейса, что помогает минимизировать когнитивную нагрузку на игрока и обеспечивает плавность навигации по игровому процессу.

В контексте данного проекта концептуальное проектирование служит основой для создания четкой структуры мобильного приложения для планирования задач с функцией уведомлений. Оно позволяет выделить основные функции, которые будут реализованы в приложении, чтобы оно стало полезным и удобным инструментом для организации задач. Эти функции формируются на основе потребностей пользователей, которые заинтересованы в простом способе управления личными делами и получении напоминаний.

Основные задачи для данного приложения включают:

1. Создание интуитивно понятного интерфейса для игры: Пользователю должно быть легко понять правила и начать играть. Интерфейс должен включать простые и понятные элементы управления, такие как кнопки для начала игры, сброса уровня и управления звуками. Важно продумать визуальные компоненты, которые будут отображать текущие уровни, количество доступных действий и результаты, чтобы игрок мог легко отслеживать свой прогресс.
2. Реализация системы уровней и задач: Ключевой задачей является создание разнообразных уровней и задач, которые будут постепенно увеличивать сложность. Каждое задание должно быть уникальным и требовать от игрока применения логики и стратегии для достижения цели. Это может включать в себя различные механики, такие как ограниченное количество переливаний или использование специальных объектов, которые добавляют элементы неожиданности и интереса.
3. Обеспечение сохранения прогресса и достижений: Данные о достигнутых уровнях и успехах игрока должны сохраняться, чтобы пользователь мог продолжать игру с того места, где остановился. Это включает в себя сохранение данных в базе данных или локальном хранилище, а также возможность отображения достижений игрока в виде статистики или наград. Гибкая система сохранения позволит игрокам не терять прогресс и мотивировать их к дальнейшему прохождению игры.

Таким образом, концептуальное проектирование игры "Переливайка" позволяет создать четкое видение структуры и функциональности приложения, ориентированное на пользовательские потребности. Этот этап является основой для более детального проектирования архитектуры, интерфейса и логики игры, обеспечивая цельность и последовательность в процессе разработки.

1.1.2 Анализ предметной области и постановка задач

В основе разработки игры "Переливайка" лежит предметная область развлечений и интерактивного взаимодействия. В современном обществе существует высокий спрос на игры, которые развлекают и одновременно тренируют логическое мышление игроков. Поэтому задача приложения заключается в создании увлекательного и интуитивно понятного игрового процесса, который позволит пользователям решать головоломки, используя различные уровни сложности и уникальные механики.

1. **Предметная область** – разработка мобильной игры-головоломки "Переливашка".
2. **Название ИС** – "Paint Sort: Coloring Puzzle".
3. **Цель ИС** – создание мобильного приложения для платформы Android, которое предоставляет пользователям увлекательную и интерактивную головоломку, где необходимо переливать жидкости между сосудами для достижения цели.
4. **Анализ осуществимости ИС** – разработка игры осуществима, так как не требует сложных технологий или инновационных решений. Основные задачи включают создание игрового интерфейса, реализацию механики переливания и разработку уровней с постепенным увеличением сложности. Аналоги проекта:

|  |  |  |  |
| --- | --- | --- | --- |
| Приложение | Преимущества | Недостатки | Вывод |
| Water Sort Puzzle | -Простой и интуитивно понятный интерфейс.  - Большое количество уровней с постепенным увеличением сложности.  - Успокаивающая цветовая гамма и приятная анимация. | - Ограниченное количество жизней и необходимость ожидания для их восстановления.  - Иногда уровни могут быть слишком однообразными. | - Необходимо добавить разнообразие в уровни, чтобы игра не становилась монотонной.  - Убрать ограничения на количество попыток, чтобы игра была более доступной и менее раздражающей. |
| Sort Puzzle: Color Water Game | - Уникальные механики, такие как использование трубок и воронок.  - Регулярное обновление с новыми уровнями. | - Некоторые уровни кажутся несправедливо сложными из-за случайного распределения цветов. | - Убедиться, что уровни сбалансированы и не вызывают разочарования у игроков. |
| Color Sort: Water Puzzle Game | - Яркая и привлекательная графика.  - Возможность играть без интернета. | - Некоторые уровни требуют слишком много времени для завершения. | - Оптимизировать уровни, чтобы они не занимали слишком много времени. |

**Таким образом проект должен содержать:**

* **Современный и удобный дизайн**: интуитивно понятный интерфейс с минималистичным дизайном и приятной цветовой гаммой.
* **Игровые элементы**: механика переливания жидкостей, уровни с постепенным увеличением сложности, система подсказок и возможность отмены ходов.
* **Персонализированное обучение**: адаптивная система уровней, которая подстраивается под навыки игрока.
* **Поддержка мультимодальных сигналов**: использование звуковых и визуальных эффектов для улучшения игрового опыта.
* **Баланс между обучением и практикой**: постепенное усложнение уровней, чтобы игроки могли развивать свои навыки.

1. Размещение и пользователи ИС – клиентская часть хранится на устройствах пользователей, а также в магазинах приложений. Данные, включая прогресс пользователя, хранятся локально. Серверная часть. Пользователями ИС являются:

* Любители головоломок и казуальных игр.
* Люди, которые хотят развивать логическое мышление и концентрацию.
* Образовательные учреждения, которые могут использовать игру для развития навыков решения задач у студентов.

1. **Комплекс технических средств ИТ**:

**Компьютерная техника**:

* Мощные компьютеры для разработки и тестирования.
* Тестовые устройства (смартфоны и планшеты на Android).
* Облачные серверы (опционально для хранения резервных копий или аналитики).

**Коммуникационная техника**:

* Интернет для публикации приложения в Google Play.
* Мобильные устройства для тестирования и использования.
* **Организационная техника**:
* Проекторы для презентации проекта.
* Оргтехника для документирования и планирования.

**Оперативная полиграфия**:

* Принтеры для печати инструкций и рекламных материалов.
* ПО для дизайна (например, Adobe Photoshop, Figma) для создания графики и макетов.

1. **Системное ПО ИТ**:

**Клиентская часть**:

* ОС Android предоставляет инструменты для взаимодействия с сенсорами устройства и API для работы с графикой и звуком.

**Средства разработки**:

* Android SDK и Android Studio – основная платформа для разработки под Android.
* Язык программирования: Java или Kotlin.
* **Средства тестирования**:
* VirtualBox для создания виртуальных машин на этапе разработки и тестирования.
* Эмуляторы Android для тестирования на различных устройствах.
* **Системы обновления ПО**:
* Google Play Console предоставляет механизмы автоматического обновления приложений.

Основные особенности проекта "Переливашка":

* Интуитивно понятный интерфейс.
* Постепенное увеличение сложности уровней.
* Отсутствие ограничений на количество попыток.
* Минимальное количество рекламы.
* Возможность отмены ходов для большего удобства.
* Подсказки для сложных уровней.
* Яркая и привлекательная графика.

Основные задачи проектирования включают:

* Определение интерфейса пользователя, который будет простым и доступным для игроков всех возрастов.
* Разработка логики игры, позволяющей игрокам настраивать уровень сложности и выбирать различные режимы игры.
* Создание функционала для прохождения уровней, получения бонусов и достижения высоких результатов.
* Определение способов хранения данных о прогрессе игроков и обеспечении их сохранности.

Цель данного проекта состоит в создании качественной и увлекательной игры "Переливайка", которая будет развлекать пользователей и предоставлять им возможность развивать свои логические навыки через интересные и разнообразные задачи.

1.1.3 Анализ аналогов и существующих решений

На рынке существует множество игр, основанных на механике сортировки и переливания жидкостей, таких как "Water Sort Puzzle" и "Color Sort". Эти игры предлагают игрокам увлекательные головоломки, где необходимо переливать жидкости между бутылками, чтобы достичь определенной цели. Однако у этих игр есть свои недостатки, такие как сложные уровни, перегруженность интерфейса и отсутствие возможности настройки сложности.

В отличие от существующих игр, разрабатываемая игра "Переливайка" будет ориентирована на минимализм, удобство и простоту использования, а также на возможность точной настройки уровней и механики игры. Особое внимание будет уделено интуитивно понятному интерфейсу и адаптации игрового процесса под конкретные предпочтения пользователей.

1.2 Спецификация требований программного обеспечения

1.2.1 Функциональные требования:

1. Обучение азбуке Морзе

* Отображение правил игры и механики (как переливать жидкости между сосудами);
* Аудио- и визуальная демонстрация процесса переливания (анимации, звуковые эффекты);
* Практические задания на решение различных уровней сложности.

1. Практика и тренировка

* Задачи по решению головоломок с переливанием жидкостей (разные комбинации и объемы);
* Игровые режимы для тренировки скорости и точности (викторины, челленджи);
* Возможность работы с реальными сценариями: создание собственных уровней и задач.

1. Система прогресса и геймификация

* Трекер прогресса пользователя (завершенные уровни, правильные решения);
* Награды за достижения (бэджи, уровни);
* Рейтинги и сравнение результатов с другими пользователями.

1. Функции управления и настройки

* Выбор режимов обучения (новичок, продвинутый, свободная практика).
* Настройки интерфейса (смена тем: тёмная, светлая).
* Регулировка сложности задач (для адаптации под уровень пользователя).
* Возможность отключения или настройки звука и анимаций.

1. Дополнительные функции

* Генератор уровней: создание пользовательских задач и их решение.
* Локализация: поддержка русского и английского языков (с возможностью добавления новых).

1.2.2Эксплуатационные требования:

1. Производительность

* Загрузка основного интерфейса не должна превышать 2 секунд на большинстве поддерживаемых устройств.
* Воспроизведение звуков и анимаций должно происходить без задержек.
* Игровые действия и решения должны обрабатываться в реальном времени.

1. Надежность

* Локальное сохранение прогресса на устройстве.
* Резервное копирование данных в облаке с возможностью восстановления.
* Приложение должно корректно завершать работу в случае непредвиденных ошибок, с возможностью восстановления после перезапуска.

1. Удобство

* Поддержка устройств с диагональю экрана от 4 до 12 дюймов (смартфоны, планшеты).
* Простота навигации: интуитивно понятный интерфейс, минимальное количество действий для запуска игры.
* Голосовая и визуальная обратная связь при выполнении действий пользователем.

1. Адаптивность

* Автоматическая адаптация интерфейса под размеры экрана устройства.
* Адаптивное обучение: для новичков — упрощенные задания с подсказками, для опытных пользователей — повышение уровня сложности.

1. Совместимость

* Совместимость с ОС Android (версии 8.0 и выше).
* Поддержка сенсорных экранов и мультимедийных функций устройств.
* Работа с устаревшими устройствами (минимальная оперативная память — 2 ГБ).
* Эскизный проект

|  |
| --- |
| **1.2.3 Методы решения и выбор языка программирования** **Платформа разработки**   * **Инструмент:** Android Studio — официальная среда разработки для Android-приложений, предоставляющая удобный инструментальный набор и эмуляторы для тестирования. * **Тип приложения:** Нативное приложение (под Android).   **Язык программирования**   * **Kotlin:** Современный язык разработки для Android с лаконичным синтаксисом. Полная поддержка Android API. Удобная работа с многопоточностью (например, для работы с анимациями и звуком). Высокая производительность и безопасность (Null-Safe).   1.2.4 Спецификация процессов   1. Основные функциональные блоки 2. Обучение: Интерактивные уроки по основам игры "Переливашка". Визуализация различных уровней сложности и правил игры. Режим практики: возможность играть с подсказками и без них. 3. Геймификация: Викторины и мини-игры для проверки знаний о правилах и механике игры. Награды и достижения за успешное выполнение заданий. Трекер прогресса (количество завершенных уровней, время на выполнение задач). 4. Настройки: Регулировка сложности игры (количество стаканов, объем жидкости). Переключение звуковых эффектов и анимаций. Управление профилем (сброс прогресса, изменение имени пользователя). 5. Работа с мультимедиа: Воспроизведение звуковых эффектов (наливание, переливание). Анимация стаканов и жидкости для визуализации процесса. Вибрация для тактильной обратной связи при успешном или неуспешном выполнении задания. 6. Архитектурный подход: приложение использует компонентную архитектуру с внедрением зависимостей (DI) для упрощения поддержки и тестирования: 7. Model: бизнес-логика, обработка данных .  * ViewModel: управление состоянием данных для интерфейса. * View: отображение данных и обработка взаимодействия пользователя.  1. Основные процессы  * Игровой процесс: Игрок выбирает сосуд и переливает жидкость в другой сосуд. Приложение симулирует физику переливания (например, уровень жидкости изменяется в зависимости от объёма).Игрок достигает цели уровня (например, сортировка жидкостей по цветам).Система проверяет выполнение условий уровня и начисляет очки. * Система событий: случайно генерируются игровые события (например, добавление новых сосудов, изменение цвета жидкостей, ограничение количества ходов). Приложение применяет изменения в игровом поле (например, добавляет новые элементы или усложняет условия). Игрок адаптируется к изменениям и ищет новые стратегии для решения задачи. Очки начисляются с учётом сложности ситуации (например, бонусы за прохождение уровня за минимальное количество ходов). * Сохранение прогресса: при паузе или завершении игры приложение сохраняет состояние в локальном хранилище (прогресс уровней, количество очков, достижения). При повторном запуске игры пользователь может продолжить с того же места.  1. Диаграмма структуры процессов – отображает основные потоки данных между компонентами приложения.      1. Навигационная диаграмма – отображает переходы между экранами приложения |

1. Интерфейс пользователя (UI):

* Главное меню с выбором уровней
* Игровое поле с сосудами
* Отображение объема жидкости в каждом сосуде
* Индикатор прогресса/ходов
* Кнопки управления (отмена хода, рестарт)

1. Модуль игровой логики:

* Управление состоянием сосудов
* Расчет возможных переливаний
* Проверка достижения цели уровня
* Подсчет количества ходов
* Сохранение/загрузка состояния игры

1. Модуль уровней:

* Генерация начальных состояний сосудов
* Хранение целевых состояний
* Проверка решаемости уровня
* Система прогрессии сложности

1. Модуль анимации:

* Визуальные эффекты при успешном завершении
* Анимация заполнения/опустошения сосудов

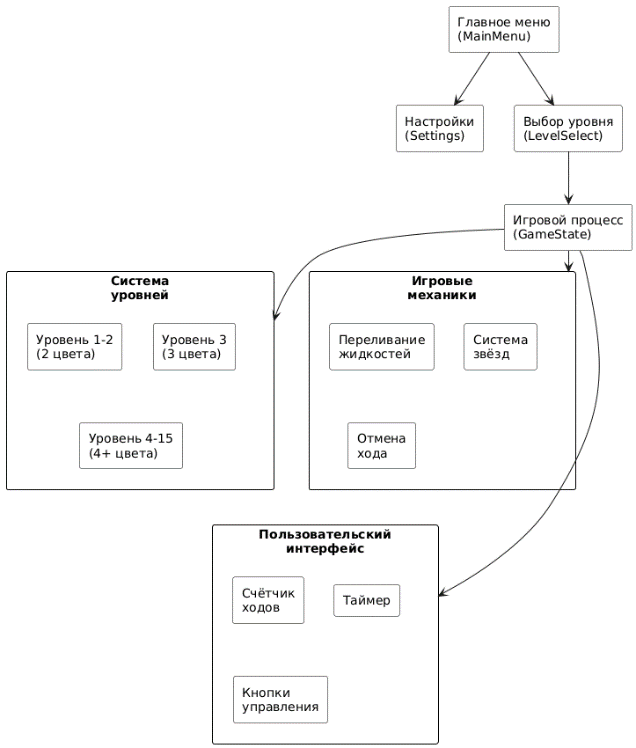
1. Модуль настроек:

* Настройка сложности
* Включение/выключение звуков
* Сброс прогресса игры
* Выбор визуальной темы

1. Система достижений:

* Отслеживание рекордов по уровням
* Подсчет общего прогресса
* Разблокировка новых уровней
* Статистика игрока

1. Структурная карта Константайна – отображает иерархическую организацию экранов и модулей приложения. Она помогает визуализировать функциональную структуру и показать, как пользователь взаимодействует с системой.



1. Структурная карта Джексона – предназначена для отображения иерархической структуры процессов приложения, включая порядок выполнения операций. Этот подход акцентирует внимание на последовательности действий и взаимосвязях между компонентами системы.



1.3 Требование к программному средству

|  |  |  |
| --- | --- | --- |
| № | Раздел | Содержание |
| 1 | Общие сведения | 1. Наименование системы: Мобильная игра-головоломка "Переливашка" (рабочее название: "Переливашка"). 2. Основание для разработки: Приложение создается на основании инициативы разработчика в целях удовлетворения потребностей пользователей, заинтересованных в увлекательных и развивающих головоломках. Игра направлена на развитие логического мышления, концентрации и навыков решения задач, а также предоставляет возможность расслабления и отдыха. 3. Наименование и условное обозначение темы разработки:  * Наименование: Разработка мобильной игры-головоломки "Переливашка". * Условное обозначение: "Переливашка v1.0".  1. Разработчик и заказчик:  * Разработчик: Груздев Д.Д. * Заказчик: Горелов А.С.  1. Плановые сроки начала и завершения работ:  * Начало разработки: 10.15.2024. * Завершение разработки: 10.31.2024. |
| 2 | Назначение и цели | 1. Назначение системы: Игра "Переливашка" предназначена для пользователей, которые хотят развивать логическое мышление, концентрацию и навыки решения задач через увлекательную головоломку. Приложение предоставляет возможность переливать жидкости между сосудами, решая задачи различной сложности, а также включает элементы геймификации для повышения мотивации и вовлеченности игроков. 2. Цели разработки:  * Создать современное, удобное и доступное мобильное приложение для любителей головоломок. * Обеспечить кроссплатформенность (доступность на Android и iOS). * Включить элементы геймификации, такие как система достижений, рейтинги и награды, для повышения мотивации пользователей. * Внедрить функциональность адаптивного обучения, чтобы уровни подстраивались под навыки игрока. * Реализовать систему отслеживания прогресса, чтобы пользователи могли видеть свои успехи и улучшения. |
| 3 | Требования к системе | 1. Требования к функциональности:   Игровая механика:   * Переливание жидкостей между сосудами для достижения цели (например, сортировка по цветам). * Постепенное увеличение сложности уровней с добавлением новых элементов (например, трубок, воронок, ограниченного количества ходов). * Система геймификации: * Уровни, достижения и награды за выполнение заданий. * Челленджи (например, пройти уровень за минимальное количество ходов). * Трекер прогресса: * Фиксация достижений пользователя (пройденные уровни, количество ходов, время прохождения). * Подсказки и помощь: * Возможность использовать подсказки для сложных уровней. * Отмена ходов для исправления ошибок. * Мультимодальные сигналы: * Использование звуковых и визуальных эффектов для улучшения игрового опыта.  1. Требования к техническому исполнению:  * Платформы: Android (версии от 8.0 и выше) * Языки программирования: Kotlin/Java для Android.  1. Производительность:  * Загрузка интерфейса: не более 2 секунд. * Обработка пользовательского ввода: мгновенная реакция на действия игрока (например, переливание жидкостей).  1. Надежность:  * Сохранение данных: прогресс пользователя (пройденные уровни, достижения) сохраняется локально на устройстве. * Восстановление данных: возможность восстановления прогресса при переустановке приложения.  1. Условия эксплуатации:   Мобильные устройства:   * Минимальная оперативная память: 2 ГБ. * Поддержка устройств с разными размерами экранов (от 4,7 дюймов и выше).  1. Интерфейс:   Удобный и современный пользовательский интерфейс:   * Адаптация под разные размеры экранов. * Поддержка тёмной и светлой тем. * Интуитивно понятное управление: * Простое взаимодействие с сосудами и жидкостями (тапы, свайпы).   Визуальные эффекты:   * Яркая и привлекательная графика. |
| 4 | Состав работ | 1. Разработка проектной документации. 2. Создание прототипа интерфейса. 3. Реализация клиентской части приложения. 4. Тестирование приложения (ручное и автоматизированное). 5. Публикация в Google Play Store и Apple App Store. 6. Поддержка и обновление приложения. |
| 5 | Порядок приемки | 1. Этапы контроля:  * Проверка прототипа интерфейса. * Тестирование отдельных модулей (функциональное тестирование). * Интеграционное тестирование. * Финальное тестирование перед релизом.  1. Критерии приемки:  * Корректная работа всех функций, указанных в ТЗ. * Полное соответствие дизайна утверждённому макету. * Успешная публикация в магазинах приложений. |
| 6 | Требования к документации | 1. Руководство пользователя (на русском и английском языках). 2. Документация для разработчиков. |
| 7 | Дополнительные требования | 1. Поддержка локализации (русский, английский языки, с возможностью добавления других). 2. Возможность интеграции с внешними сервисами (например, авторизация через Google). 3. Планы по развитию приложения (внедрение новых функций на следующих этапах). |

2 Техно-рабочий проект

2.1 Алгоритм решения поставленной задачи программного продукта

Разработка игры "Переливашка" требует четкого алгоритма, который позволит реализовать все функциональные требования и обеспечить корректную работу приложения. В данном разделе описывается последовательность действий, необходимых для создания игры, а также представлена диаграмма прецедентов, которая визуализирует взаимодействие пользователя с системой.

#### 2.1.1 Основные этапы разработки алгоритма

1. **Инициализация игры:**

* Загрузка начальных параметров игры: размер поля, количество цветов, начальное состояние игрового поля.
* Генерация игрового поля с использованием случайных цветов.
* Отображение игрового интерфейса: поле, кнопки управления, счетчик ходов.

1. **Обработка пользовательских действий**:

* Ожидание выбора пользователем цвета для заливки.
* Проверка допустимости выбранного цвета (цвет должен отличаться от текущего цвета игрового поля).
* Обновление игрового поля: заливка всех смежных клеток выбранным цветом.
* Увеличение счетчика ходов.

1. **Проверка условий завершения игры**:

* Проверка, все ли клетки игрового поля заполнены одним цветом.
* Если условие выполнено, игра завершается, и пользователю выводится сообщение о победе.
* Если лимит ходов исчерпан, игра завершается с сообщением о поражении.

1. **Реализация дополнительных функций**:

* Возможность перезапуска игры без перезагрузки приложения.
* Сохранение и загрузка прогресса игры.
* Настройка сложности (изменение размера поля или количества цветов).

1. **Тестирование и отладка**:

* Проверка корректности работы всех функций игры.
* Устранение ошибок и оптимизация производительности.

2.2 Обоснование выбора средств разработки

Для разработки приложения для планирования задач с функцией уведомления я решил использовать Android Studio и язык программирования Kotlin. Это современный и широко используемый стек для создания приложений под платформу Android, обладающий рядом преимуществ:

1. Официальная поддержка Google:

Android Studio является официальной средой разработки для Android-приложений, что гарантирует оптимизацию под платформу, регулярные обновления, поддержку современных библиотек и инструментов. Это обеспечивает актуальность проекта и доступ к современным возможностям.

1. Стабильность и производительность:

Android Studwio предлагает мощные инструменты для разработки, такие как встроенный эмулятор, инспектор памяти и профайлер, которые упрощают тестирование и отладку. Среда предоставляет стабильную производительность и помогает оптимизировать приложение для плавной работы на различных устройствах.

1. Kotlin – современный и безопасный язык:

Kotlin стал официальным языком для Android-разработки и предлагает удобный, лаконичный синтаксис, повышая скорость написания кода и снижая вероятность ошибок. Благодаря поддержке null-безопасности и расширенной совместимости с Java, Kotlin обеспечивает высокую стабильность кода и облегчает поддержку.

1. Широкое сообщество и обширная документация:

Вокруг Android Studio и Kotlin сложилось большое сообщество разработчиков, что облегчает доступ к руководствам, примерам кода и форумам. Это помогает решать проблемы и позволяет находить решения для нестандартных задач.

1. Интеграция с инструментами разработки интерфейса:

Android Studio поддерживает инструменты визуального дизайна интерфейсов, такие как Layout Editor, что помогает создавать и настраивать интерфейс с минимальным количеством ручного кода. Эта функция особенно полезна для обеспечения удобства использования приложения.

1. Расширяемость и интеграция библиотек:

Android Studio поддерживает широкий спектр библиотек и фреймворков для разработки мобильных приложений, таких как Jetpack Compose для создания пользовательских интерфейсов и WorkManager для управления задачами. Это делает разработку приложения гибкой и позволяет добавлять дополнительный функционал без значительных затрат времени.

Исходя из этих аргументов, выбор Android Studio и Kotlin в качестве инструментов разработки для проекта является обоснованным и позволяет эффективно реализовать поставленные цели, обеспечивая гибкость, надежность и высокую производительность приложения.

2.3 Разработка макета приложения

Логотип проекта:



Для проектирования макета и интерфейса приложения используется Word.

Всего в приложении 3 экрана:

* Главное меню
* Экран выбора уровня
* Экран игры

Главное меню (Рисунок 1) содержит в себе:

1. название игры,
2. кнопка начало игры,
3. кнопка настроек,
4. кнопка выхода.

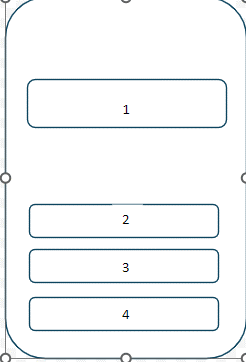


Рисунок 1 - главный экран

Экран выбора уровня (Рисунок 2) содержит:

1. кнопки выбора уровня,
2. имя экрана "Select Level"

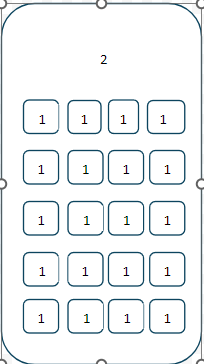


Рисунок 2 - экран выбора уровня

Экран игры (Рисунок 3) содержит в себе:

1. сколько раз выполнено движений,
2. название игры,
3. бутылки,
4. кнопка перезапустить,
5. выход.

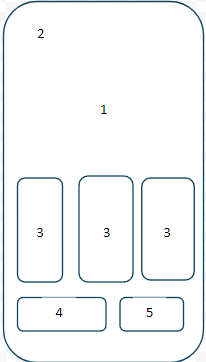


Рисунок 3 - экран игры2.4 Описание разработки приложения

Для создания приложения необходимо реализовать несколько вещей:

1. Главное меню.
2. Экран уровней.
3. Механика игры.

Начнем с того что, создаем новый интерфейс screens.kt. В этом классе мы создаем функцию MainMenu которая отвечает за создание главного меню. Она принимает два параметра: функции обратного вызова onStartGame и onSettings, которые будут вызваны при нажатии соответствующих кнопок и ещё используется LaunchedEffect для установки видимости меню, что позволяет анимировать его появление. Меню состоит из Box, который заполняет весь экран и имеет градиентный фон. Внутри Box находится AnimatedVisibility, который управляет анимацией появления и исчезновения меню. Мы используем градиентный фон который использует вертикальный градиент для создания привлекательного фона и текст заголовка с помощь которого заголовок игры отображается в центре экрана.

Кнопки создаются с помощью функции MenuButton, которая принимает текст и действие при нажатии. Кнопки создаются с помощью функции MenuButton, которая принимает текст и действие при нажатии. Кнопка сжимается при нажатии, что создает эффект взаимодейств ия. Используется animateFloatAsState для анимации изменения масштаба кнопки. Кнопка имеет закругленные углы, тень и зеленый цвет фона, что делает её визуально привлекательной. Кнопка "Выход" завершает работу приложения, вызывая метод finish() у текущей активности. С помощью LaunchedEffect реализована задержка, которая сбрасывает состояние нажатия кнопки после 100 миллисекунд.

Функция SettingsScreen отвечает за создание экрана настроек. Она принимает функцию обратного вызова onBack, которая будет вызвана при нажатии кнопки "Назад". Используется LaunchedEffect для установки видимости экрана настроек, что позволяет анимировать его появление. Экран настроек состоит из Box, который заполняет весь экран и имеет градиентный фон. Внутри Box находится AnimatedVisibility, который управляет анимацией появления и исчезновения экрана. Используется вертикальный градиент для создания привлекательного фона экрана настроек. Заголовок "Settings" отображается в центре экрана, что делает его заметным для пользователя. С помощью кнопки назад позволяет пользователю вернуться на главное меню, вызывая функцию onBack.

Функция LevelSelectScreen отвечает за создание экрана выбора уровня. Она принимает два параметра: onLevelSelected, который вызывается при выборе уровня, и onBack, который возвращает игрока на предыдущий экран.  Используется LaunchedEffect для установки видимости экрана выбора уровня, что позволяет анимировать его появление. Экран выбора уровня состоит из Box, который заполняет весь экран и имеет градиентный фон. Внутри Box находится AnimatedVisibility, который управляет анимацией появления и исчезновения экрана. Уровни отображаются в виде сетки с использованием LazyVerticalGrid, что позволяет удобно размещать карточки уровней. Функция AnimatedLevelCard создает карточку для каждого уровня. Она принимает уровень и функцию обратного вызова onLevelSelected. Каждая карточка уровня появляется с задержкой, основанной на индексе уровня, что создает эффект последовательного появления. Карточка уровня создается с использованием компонента Card, который имеет стильный дизайн и интерактивность. Каждая карточка уровня реагирует на нажатие, вызывая функцию onLevelSelected, что позволяет игроку выбрать уровень.

Функция PaintSortGame отвечает за создание игрового экрана. Она принимает состояние игры и функцию для возврата в меню.  Используется LaunchedEffect для установки видимости игрового экрана, что позволяет анимировать его появление. Игровой экран состоит из Box, который заполняет весь экран и имеет градиентный фон. Внутри Box находится AnimatedVisibility, который управляет анимацией появления и исчезновения экрана.Игровой экран отображает количество ходов, лимит ходов и оставшееся время, что позволяет игроку отслеживать прогресс. Используется LazyVerticalGrid для отображения бутылок, что позволяет удобно организовать элементы игры. Кнопки "Reset", "Undo" и "Menu" предоставляют игроку возможность управлять игрой и возвращаться в меню.

После этого создаем класс Components.kt в нем я разробатываю ключевые компоненты интерфейса для игры. Функция TimeProgressBar отвечает за отображение прогресс-бара, который показывает оставшееся время в игре. Функция принимает два параметра: timeLeft (оставшееся время) и totalTime (общее время).

Используется LinearProgressIndicator для визуализации прогресса. Прогресс рассчитывается как отношение оставшегося времени к общему времени.Функция BottlesGrid создает сетку бутылок, отображая текущее состояние игры. Используется LazyVerticalGrid для отображения бутылок в виде сетки, что позволяет удобно организовать элементы интерфейса. Каждая бутылка отображается с помощью функции Bottle, которая принимает состояние бутылки и индекс. Функция Bottle отвечает за отображение отдельной бутылки, включая анимацию наливания. Используется LaunchedEffect для запуска анимации наливания, если бутылка находится в состоянии наливания. Внутри функции Bottle создается визуальное представление бутылки с использованием Canvas. Используется Canvas для рисования бутылки, ее содержимого и анимации наливания. Бутылка имеет градиентный цвет и эффект блеска, что делает ее более реалистичной. Функция LevelCompletedAlert создает диалоговое окно, которое уведомляет игрока о завершении уровня. : В диалоговом окне используется анимация Lottie для создания визуально привлекательного эффекта, который подчеркивает завершение уровня. Отображается сообщение о завершении уровня, количество использованных ходов и кнопка для перехода к следующему уровню.

Создаем класс GameState.kt. В этом классе я разробатываю основная механику игры.Класс Bottle представляет собой бутылку, содержащую цветные жидкости. В классе есть параметтры capacity - максимальная вместимость бутылки и contents - список цветов, находящихся в бутылке. Перечисление Level определяет различные уровни игры, каждый из которых имеет свои параметры:

* bottleCount: количество бутылок на уровне.
* colorCount: количество различных цветов.
* fullBottles: количество полных бутылок.
* emptyBottles: количество пустых бутылок.
* moveLimit: ограничение по количеству ходов (опционально).
* timeLimit: ограничение по времени (опционально).
* specialColors: список специальных цветов (опционально).
* stars: количество звезд, присуждаемых за завершение уровня.

Класс GameState управляет состоянием игры, включая бутылки, выбранные бутылки, количество ходов и другие параметры:

* bottles: список бутылок в игре.
* selectedBottle: индекс выбранной бутылки.
* moves: количество сделанных ходов.
* undoStack: стек для отмены действий.
* moveLimit и timeLeft: ограничения по ходам и времени.
* isGameOver и isLevelCompleted: флаги для проверки состояния игры.

Метод initializeLevel создает бутылки и заполняет их цветами в зависимости от параметров уровня. Метод generateColors создает список цветов, включая специальные цвета. Метод generateContents заполняет бутылки цветами, перемешивая их для случайности. Метод selectBottle обрабатывает выбор бутылки и наливание между бутылками. Метод pourBetweenBottles выполняет наливание из одной бутылки в другую, проверяя условия, такие как наличие содержимого и вместимость. Метод checkLevelComplete проверяет, завершен ли уровень.  Уровень считается завершенным, если все бутылки пустые или заполнены одним цветом. Метод updateTimeLeft обновляет оставшееся время и проверяет, не истекло ли оно. Если время истекло, вызывается метод checkGameOver. Метод navigateToNextLevel позволяет игроку перейти к следующему уровню после завершения текущего. ровни циклически переключаются, что позволяет игроку продолжать игру, пока не достигнет последнего уровня.

2.5 Тестирование приложения

Как только пользователь запустит приложение, его встретит главное меню. Главное меню показано на рисунке 3.

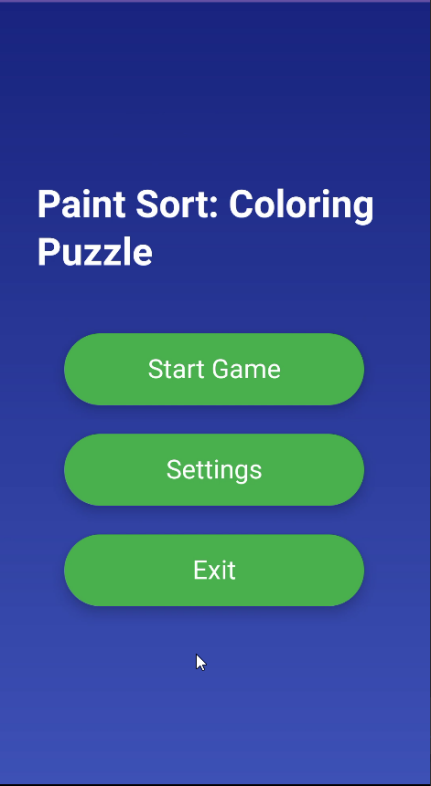


Рисунок 4 – Главное меню

После того как пользователь нажимает кнопку старт игры появляется выборка уровня. Выбор уровня показано на рисунке 5.

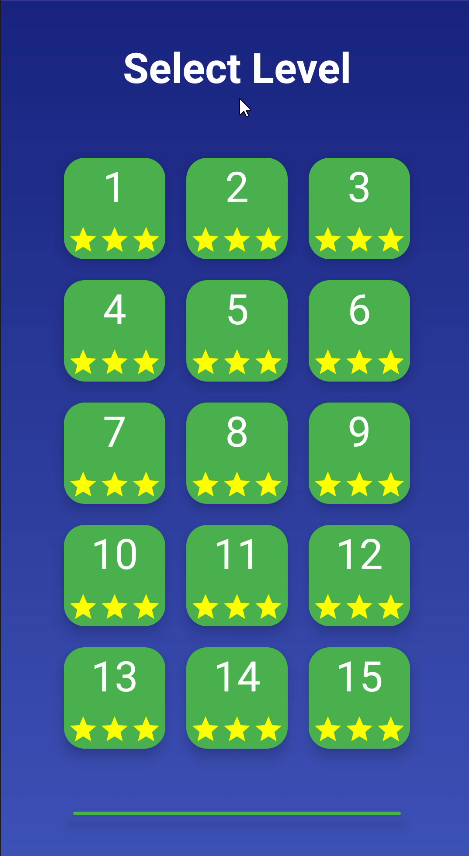


Рисунок 5 - экран выбора уровня

После того как пользователь выбрал уровень появляется игра в котором пользователю нужно решить головоломку. Экран головоломки показано на рисунке 6.

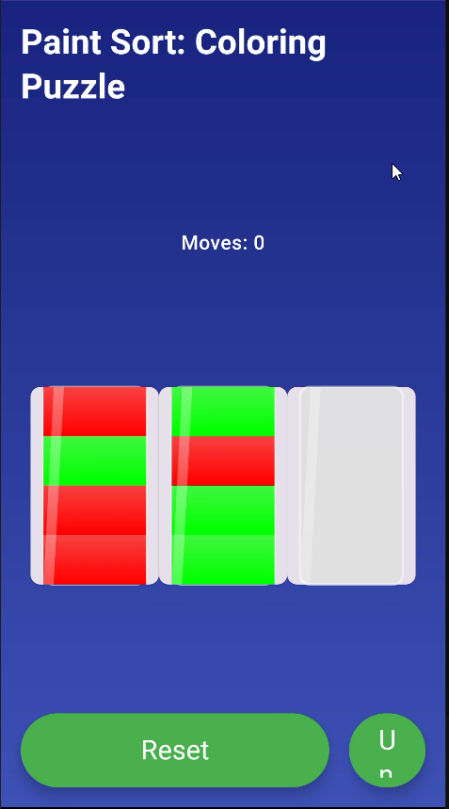


Рисунок 6 - экран головоломки

После решении головоломки появляется надпись что он выполнил уровень. Надпись выполнение уровня показано на рисунке 7.

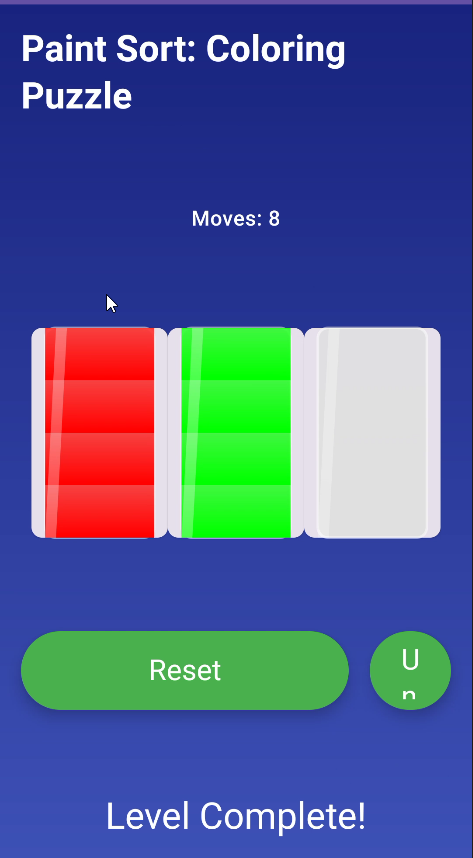


рисунок 7 - надпись выполнение уровня

3 Рабочая документация

3.1 Руководство пользователя

Диаграмма вариантов использования (Рисунок - 1) для приложения "Paint Sort: Coloring Puzzle" представляет собой визуальное описание взаимодействия пользователя с системой. Она включает ключевые функции приложения и их взаимосвязи. Основным актором в данной диаграмме является Пользователь, который взаимодействует с игрой для выполнения различных задач.

Основные функции приложения представлены следующими вариантами использования:

1. **Начать игру** – Пользователь может начать новую игру, что является отправной точкой для взаимодействия с системой.
2. **Играть уровень** – Пользователь выбирает и проходит уровни, что является основным игровым процессом.
3. **Открыть настройки** – Пользователь может настроить параметры игры, такие как звук, графика и другие предпочтения.
4. **Выйти из игры** – Пользователь может завершить игровой сеанс.

Дополнительные функции включают:

* **Просмотр статистики** – Пользователь может просматривать свои достижения и прогресс в игре.
* **Управлять бутылками** – Основной игровой процесс, где пользователь управляет бутылками для решения головоломки.
* **Проверка завершения** – Пользователь может проверить, выполнены ли условия для завершения уровня.

Связи между вариантами использования включают:

* **Include** – Некоторые функции, такие как "Просмотр таймера", включены в основной игровой процесс для отслеживания времени.
* **Сбросить уровень** – Пользователь может сбросить текущий уровень и начать заново.
* **Выбрать уровень** – Пользователь может выбрать конкретный уровень для прохождения.

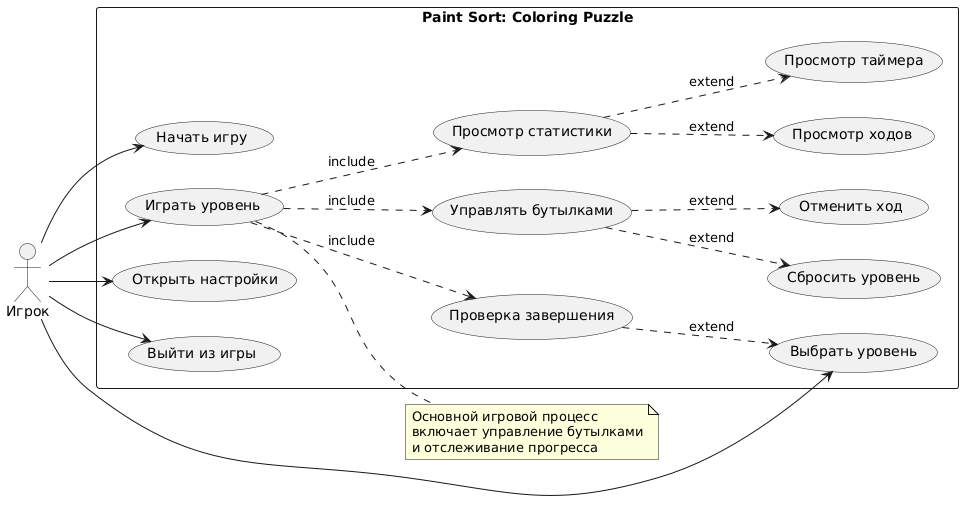


Рисунок - Диаграмма использования

Диаграмма последовательностей (Рисунок - 2) для игрового процесса описывает взаимодействие между пользователем и различными компонентами системы в процессе игры. Основная цель диаграммы — показать последовательность действий, которые выполняются при взаимодействии пользователя с игрой, а также обмен сообщениями между компонентами системы.

В диаграмме присутствуют следующие акторы и объекты:

|  |  |  |
| --- | --- | --- |
| Элемент | Тип | Назначение |
| Пользователь | Актор | представляющий конечного пользователя, который взаимодействует с игрой. |
| **Главное меню** | Объект | представляющий основной интерфейс игры, с которого начинается взаимодействие. |
| **Система уровней** | Объект | отвечающий за загрузку и управление уровнями игры. |
| **Игровой движок** | Объект | который загружает и управляет игровыми уровнями. |
| **Система очков** | Объект | отвечающий за подсчет и отображение очков в процессе игры. |
| **Система сохранений** | Объект | который управляет сохранением и загрузкой прогресса игры. |

Диаграмма отражает следующий процесс:

1. **Запуск игры:**

* Пользователь запускает игру, инициируя взаимодействие с системой.
* Главное меню загружается и отображает интерфейс.
* Этот шаг необходим для начала игрового процесса.

1. **Выбор уровня:**

* Пользователь выбирает опцию "Играть" в главном меню.
* Главное меню передает запрос в систему уровней для загрузки доступных уровней.
* Система уровней загружает данные о прогрессе и отображает выбор уровней.
* Пользователь выбирает уровень, и система уровней передает запрос в игровой движок для загрузки выбранного уровня.

1. **Загрузка уровня:**

* Игровой движок загружает выбранный уровень и отображает игровой экран.
* Пользователь начинает игровой процесс.

1. **Игровой процесс:**

* В процессе игры система очков отслеживает и минимизирует счетчик очков.
* Пользователь взаимодействует с игрой, выполняя задачи уровня.

1. **Сохранение прогресса:**

* По завершении уровня или в процессе игры система сохранений сохраняет прогресс игрока.
* Прогресс может включать данные о пройденных уровнях, набранных очках и других достижениях.

1. **Завершение игры:**

* Пользователь может вернуться в главное меню для выбора нового уровня или завершить игровой сеанс.

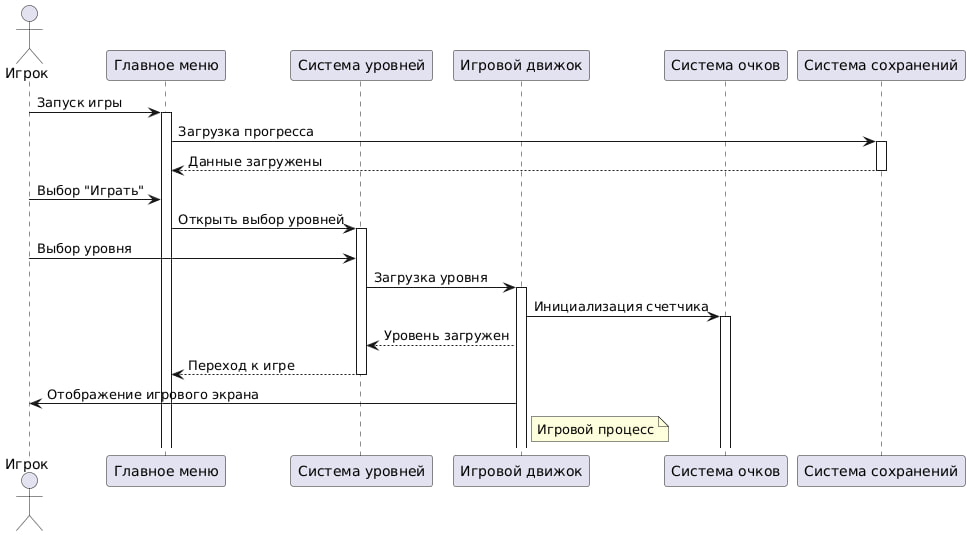


Рисунок - Диаграмма последовательностей

Диаграмма кооперации (Рисунок - 3) для игры представляет собой визуализацию взаимодействия между различными объектами системы в рамках игрового процесса. Она демонстрирует, как объекты взаимодействуют друг с другом для выполнения определенных задач, таких как запуск игры, управление уровнями, начисление очков и сохранение прогресса.

Диаграмма включает следующие объекты:

1. **Главное меню** – Точка входа в игру, где пользователь может выбрать режим игры, открыть настройки или запустить игру.
2. **Система уровня** – Управляет загрузкой уровней, сохранением прогресса и переходом между уровнями.
3. **Уровень** – Объект, который управляет текущим уровнем игры, включая начисление очков и обновление состояния.
4. **Достижения** – Отслеживает и управляет достижениями игрока, такими как завершение уровней и получение наград.
5. **Сохраняет** – Управляет сохранением и загрузкой игрового прогресса, а также очисткой данных.

Последовательность взаимодействий отражает процесс выполнения игрового сценария:

**Этап 1: Запуск игры**

* **Шаг 1:** Пользователь инициирует взаимодействие с системой, выбирая в главном меню пункт "Запустить Игру".
* **Главное меню** является центральным элементом навигации, с которого пользователь получает доступ к различным функциям игры.

**Этап 2: Загрузка уровня**

* **Шаг 2:** Главное меню передает управление объекту **Система уровня**, который загружает выбранный уровень.
* **Шаг 3:** Система уровня передает управление объекту **Уровень**, который начинает игровой процесс.

**Этап 3: Игровой процесс**

* **Шаг 4:** Объект **Уровень** управляет текущим состоянием игры, включая начисление очков и обновление резервов.
* **Шаг 5:** Пользователь взаимодействует с игрой, выполняя задачи уровня.

**Этап 4: Начисление очков и достижений**

* **Шаг 6:** Объект **Уровень** передает информацию о начисленных очках объекту **Система очков**.
* **Шаг 7:** Объект **Достижения** обновляет статус достижений игрока на основе выполненного уровня.

**Этап 5: Сохранение прогресса**

* **Шаг 8:** Объект **Сохраняет** сохраняет текущий прогресс игры, включая очки и достижения.
* **Шаг 9:** Пользователь может завершить игру, и система сохраняет все данные для последующего использования.

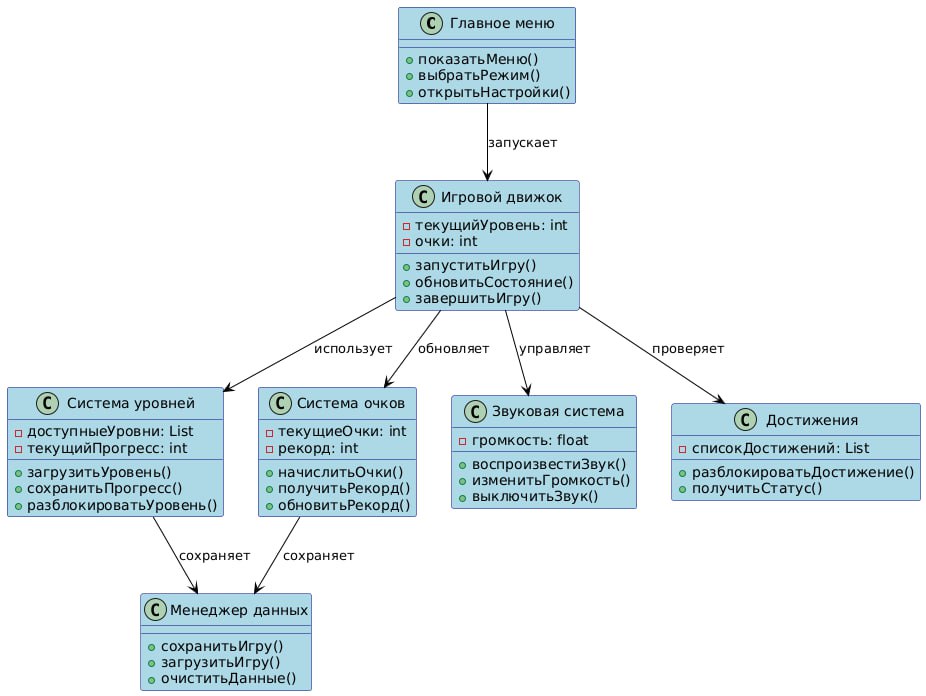


Рисунок - Диаграмма кооперации

Диаграмма развертывания (Рисунок - 4) для игры "Переливашка" представляет собой блок-схему процесса установки и начала использования мобильного приложения. Она демонстрирует последовательность действий пользователя — от поиска приложения в магазине до начала игры. Диаграмма построена по линейной структуре, где каждый последующий этап логически вытекает из предыдущего.

1. **Начальный этап: «Пользователь заходит в магазин приложений для Android»**

* Магазин приложений (например, Google Play) является основным источником получения мобильных приложений на платформе Android.
* Пользователь начинает взаимодействие с системой именно с поиска приложения в этом магазине.

1. **Действие: «Пользователь вводит в поисковую строку название игры "Переливашка"»**

* Пользователь должен найти нужное приложение среди множества других, введя его название в поисковую строку.

1. **Действие: «Пользователь скачивает и устанавливает установочный пакет»**

* Любое приложение перед использованием должно быть загружено и установлено.
* Этот этап преобразует программное обеспечение из доступного в магазине в установленное на устройстве.

1. **Действие: «Пользователь заходит в приложение»**

* После установки необходимо запустить приложение, чтобы начать взаимодействие с ним.

1. **Действие: «Пользователь настраивает приложение в меню настроек»**

* Для более удобного использования пользователь может захотеть изменить настройки (например, выбрать язык, включить звуковые сигналы или вибрацию).
* Некоторые настройки могут быть критически важны для полноценной работы приложения.

1. **Действие: «Пользователь выбирает уровень»**

* Пользователь выбирает уровень, с которого он хочет начать игру.

1. **Действие: «Пользователь проходит уровни, переливая цвета»**

* Основной игровой процесс, где пользователь взаимодействует с игрой, решая головоломки и переливая цвета.

1. **Завершающий этап: «Пользователь сохраняет прогресс и выходит из игры»**

* После завершения игры пользователь сохраняет свой прогресс, чтобы продолжить с того же места в следующий раз.
* Пользователь может выйти из игры, завершив игровой сеанс.

****

Рисунок - Диаграмма развертывания

Диаграмма деятельности (Рисунок - 5) описывает последовательность действий пользователя в игровом приложении, начиная с открытия приложения и заканчивая завершением игры. Диаграмма отображает логику взаимодействия пользователя с системой, включая выбор уровней, игровые действия и возможные варианты развития событий.

Основные этапы диаграммы:

1. **Открытие приложения:**

* Пользователь запускает приложение.
* Отображается главное меню.

1. **Выбор действия в главном меню:**

* Пользователь выбирает опцию "Игра" или "Другое действие".
* Если выбрана "Игра", пользователь переходит к выбору уровня.
* Если выбрано "Другое действие", пользователь выполняет соответствующее действие (например, настройки или просмотр статистики).

1. **Выбор уровня:**

* Пользователь выбирает уровень для игры.
* Отображается игровое поле.

1. **Игровой процесс:**

* Пользователь выбирает бутылку и целевую бутылку для перелива.
* Система проверяет возможность перелива.
* Если перелив возможен, выполняется анимация перелива, обновляется состояние игры, и проверяется завершение уровня.
* Если перелив невозможен, игра продолжается без изменений.

1. **Проверка завершения уровня:**

* Если уровень завершен, отображаются результаты уровня.
* Если это последний уровень, игра завершается.
* Если это не последний уровень, разблокируется следующий уровень.
* Если уровень не завершен, игра продолжается.

1. **Завершение просмотра:**

* Пользователь завершает просмотр результатов или игрового процесса.
* Происходит выход из игры или возврат в главное меню.

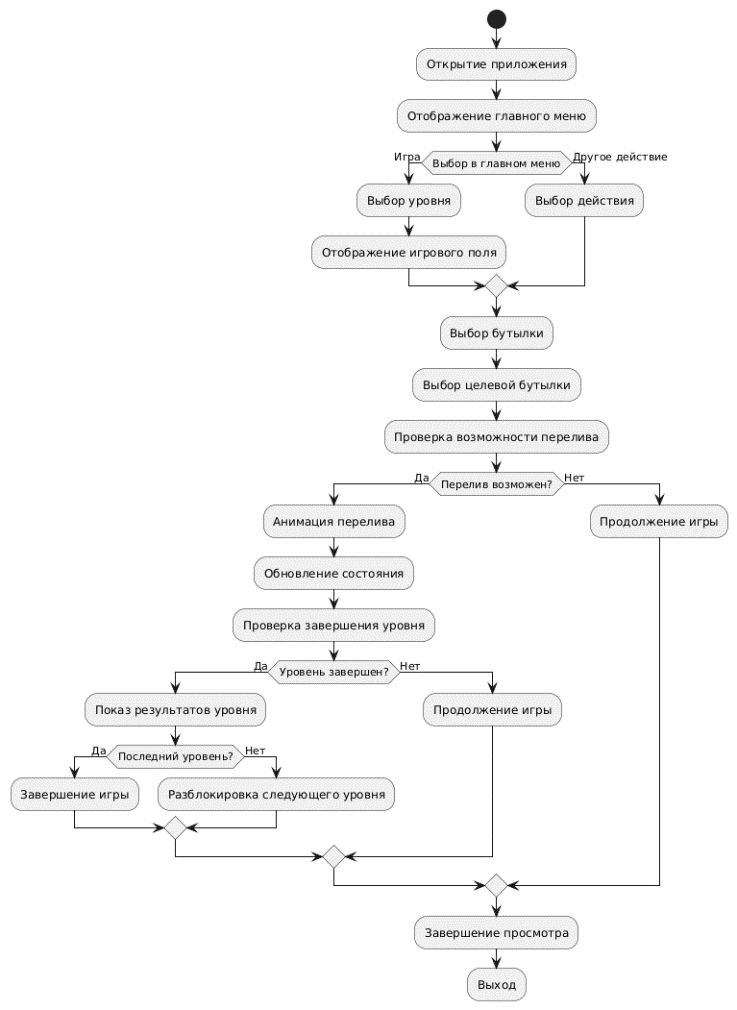


Рисунок - Диаграмма деятельности

Диаграмма состояний (Рисунок – 6) моделирует поведение игрового приложения в терминах различных состояний системы и переходов между ними. Она демонстрирует, как пользователь взаимодействует с приложением и как система реагирует на его действия.

Основные компоненты диаграммы:

1. **Состояния:**

* **ожидание Выбора** – Приложение ожидает действий пользователя, например, выбора уровня или настройки;
* **завершение Уровня** – Состояние, в котором уровень завершен, и система ожидает дальнейших инструкций;
* **настройки** – Состояние, в котором пользователь может управлять настройками приложения;
* **выход** – Состояние, в котором пользователь завершает работу с приложением.

1. **Переходы:**

* **выбор уровня** – Пользователь выбирает уровень для игры, что приводит к переходу в состояние игрового процесса;
* **повторять** – Пользователь решает повторить уровень, что возвращает систему в состояние игрового процесса;
* **управление Настройками** – Пользователь изменяет настройки приложения, что приводит к переходу в состояние "Настройки";
* **проверка Завершения** – Система проверяет, завершен ли уровень, и переходит в соответствующее состояние.

1. **Начальное и конечное состояния:**

* **начальное состояние** – Приложение запущено и ожидает выбора пользователя;
* **конечное состояние** – Пользователь завершает работу с приложением.

Переход между состояниями:

* **запуск приложения** → **Ожидание Выбора** – Приложение загружено, и пользователь видит главное меню;
* **ожидание Выбора** → **Выбор уровня** – Пользователь выбирает уровень для игры;
* **выбор уровня** → **Игровой процесс** – Пользователь начинает играть на выбранном уровне;
* **игровой процесс** → **Проверка Завершения** – Система проверяет, завершен ли уровень;
* если уровень завершен, система переходит в состояние **Завершение Уровня**;
* если уровень не завершен, система возвращается в состояние **Игровой процесс**;
* **завершение Уровня** → **Повторять** – Пользователь решает повторить уровень;
* **завершение Уровня** → **Выход** – Пользователь завершает игру и выходит из приложения.

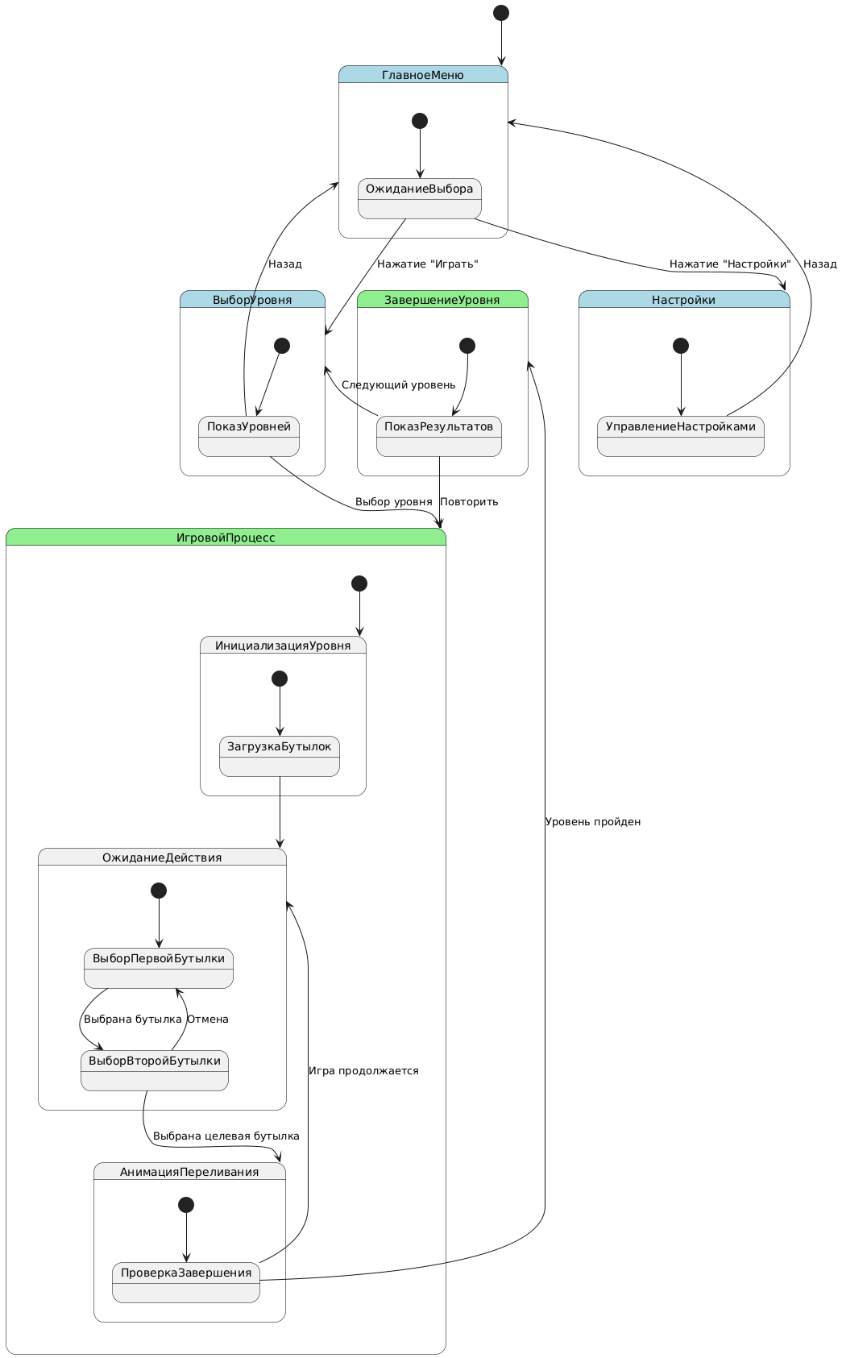


Рисунок - Диаграмма состояний

3.2 Руководство разработчика

Диаграмма классов (Рисунок - 7) иллюстрирует архитектуру классов игрового приложения, включая основные компоненты, такие как уровни, состояние игры, бутылки и управление анимациями. Она показывает взаимосвязи между классами и их функциональность.

Основные классы и их описание:

1. **Level (Уровень):**

**Атрибуты:**

* number: Int – Номер уровня;
* requiredMoves: Int – Количество необходимых ходов для завершения уровня;
* timeLimit: Int? – Ограничение по времени для уровня (опционально);
* initialBottles: List<Bottle> – Начальное состояние бутылок на уровне.

**Методы:**

* isCompleted(): Boolean – Проверяет, завершен ли уровень;
* getStars(): Int – Возвращает количество звезд, полученных за уровень.

1. **GameApplication (Игровое приложение):**

**Атрибуты:**

* gameState: GameState – Текущее состояние игры.

**Методы:**

* onCreate() – Инициализация приложения.

1. **GameState (Состояние игры):**

**Атрибуты:**

* currentLevel: Int – Текущий уровень;
* moves: Int – Количество сделанных ходов;
* timer: Timer – Таймер для отслеживания времени;
* bottles: List<Bottle> – Список бутылок в текущем уровне.

**Методы:**

* resetLevel() – Сбрасывает уровень;
* undoMove() – Отменяет последний ход;
* checkWinCondition(): Boolean – Проверяет, выполнены ли условия победы.

1. **Bottle (Бутылка):**

**Атрибуты:**

* colors: Stack<Color> – Стек цветов в бутылке;
* capacity: Int – Вместимость бутылки.

1. **LevelManager (Менеджер уровней):**

**Атрибуты:**

* levels: List<Level> – Список всех уровней;
* currentProgress: Map<int, Int> – Текущий прогресс по уровням.

**Методы:**

* loadLevel(number: Int): Level – Загружает уровень по номеру;
* saveProgress() – Сохраняет прогресс игры.

1. **AnimationManager (Менеджер анимаций):**

**Методы:**

* playPourAnimation(from: Bottle, to: Bottle) – Проигрывает анимацию переливания;
* playWinAnimation() – проигрывает анимацию победы;
* playStarAnimation() – проигрывает анимацию получения звезд.

1. **Color (Цвет):**

* перечисление цветов, используемых в игре: RED, BLUE, GREEN, YELLOW, PURPLE, ORANGE, PINK.

Взаимосвязи между классами:

* **gameApplication** управляет общим состоянием игры через **GameState**;
* **gameState** содержит текущий уровень и список бутылок, а также методы для управления игровым процессом;
* **level** описывает конкретный уровень, включая начальное состояние бутылок и условия завершения;
* **bottle** представляет собой бутылку с цветами, которые можно переливать;
* **levelManager** отвечает за загрузку уровней и сохранение прогресса;
* **animationManager** управляет анимациями, такими как переливание, победа и получение звезд.

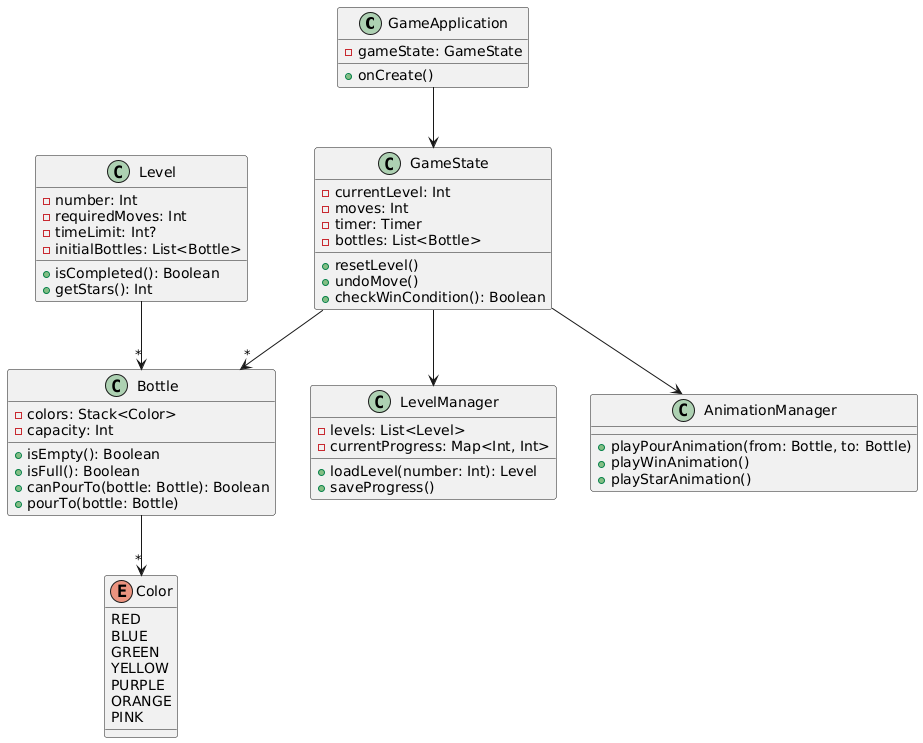


Рисунок - диаграмма классов

# Заключение

В рамках данного курсового проекта было разработано мобильное приложение для планирования задач с функцией уведомления. Целью проекта было создание удобного и эффективного инструмента для организации задач пользователя с возможностью напоминания о предстоящих делах.

В процессе разработки были выполнены следующие задачи:

* Определена концепция и архитектура приложения, включающая функциональные и нефункциональные требования.
* Разработан интерфейс пользователя, обеспечивающий удобный доступ к функциям добавления, редактирования и удаления задач.
* Реализована система уведомлений, позволяющая пользователям получать напоминания о задачах в заданное время.
* Выполнено тестирование приложения для обеспечения стабильной работы и удобного пользовательского опыта.

В результате проделанной работы было создано мобильное приложение, которое упрощает процесс управления личными задачами и помогает пользователям более эффективно организовать своё время. Разработанный проект демонстрирует возможности Android Studio и языка программирования Kotlin для создания удобных и функциональных мобильных приложений.

Проект предоставляет основу для дальнейшего развития и усовершенствования, включая добавление новых функций и оптимизацию интерфейса для лучшего взаимодействия с пользователем.

# Список использованной литературы

1. Официальная документация Android [Электронный ресурс] - https://developer.android.com/?hl=ru, 2025
2. Сообщество разработчиков Android на Reddit [Электронный ресурс] - https://www.reddit.com/r/androiddev/, 2025
3. Форум Stack Overflow для разработчиков [Электронный ресурс] - https://stackoverflow.com/, 2025
4. Ray Wenderlich, руководства и статьи по Android Studio [Электронный ресурс] - https://www.kodeco.com/library, 2025
5. Официальная документация Kotlin [Электронный ресурс] - https://kotlinlang.org/docs/home.html, 2025
6. Руководства и гайды по Android Studio [Электронный ресурс] - https://www.linode.com/docs/guides, 2021
7. «Android Programming: The Big Nerd Ranch Guide» Билл Филипс, Крис Стюарт и Кристин Марсикано, 2016
8. «Kotlin в действии» Дмитрий Жемеров, Светлана Исакова, 2017
9. «Head First. Программирование для Android на Kotlin. 3-е изд.» Дон Гриффитс, Дэвид Гриффитс, 2023

# Приложение А

package com.example.perelevaica

import android.app.Activity

import androidx.compose.animation.AnimatedVisibility

import androidx.compose.animation.core.EaseOutBack  
import androidx.compose.animation.core.Spring  
import androidx.compose.animation.core.animateFloatAsState  
import androidx.compose.animation.core.spring  
import androidx.compose.animation.core.tween  
import androidx.compose.animation.expandVertically  
import androidx.compose.animation.fadeIn  
import androidx.compose.animation.fadeOut  
import androidx.compose.animation.scaleIn  
import androidx.compose.animation.scaleOut  
import androidx.compose.animation.shrinkVertically  
import androidx.compose.foundation.background  
import androidx.compose.foundation.clickable  
import androidx.compose.foundation.layout.Arrangement  
import androidx.compose.foundation.layout.Box  
import androidx.compose.foundation.layout.Column  
import androidx.compose.foundation.layout.PaddingValues  
import androidx.compose.foundation.layout.Row  
import androidx.compose.foundation.layout.Spacer  
import androidx.compose.foundation.layout.aspectRatio  
import androidx.compose.foundation.layout.fillMaxSize  
import androidx.compose.foundation.layout.height  
import androidx.compose.foundation.layout.padding  
import androidx.compose.foundation.layout.width  
import kotlinx.coroutines.delay  
  
@Composable  
fun MainMenu(onStartGame: () -> Unit, onSettings: () -> Unit) {  
 val context = *LocalContext*.current  
 var visible by remember **{** *mutableStateOf*(false) **}** LaunchedEffect(Unit) **{** visible = true  
 **}** Box(  
 modifier = Modifier  
 .*fillMaxSize*()  
 .*background*(  
 Brush.verticalGradient(  
 colors = *listOf*(*Color*(0xFF1A237E), *Color*(0xFF3F51B5))  
 )  
 )  
 ) **{** AnimatedVisibility(  
 visible = visible,  
 enter = *fadeIn*(animationSpec = *tween*(1000)) + *expandVertically*(  
 animationSpec = *tween*(1000, easing = *EaseOutBack*)  
 ),  
 exit = *fadeOut*() + *shrinkVertically*()  
 ) **{** Column(  
 modifier = Modifier  
 .*fillMaxSize*()  
 .*padding*(32.*dp*),  
 horizontalAlignment = Alignment.CenterHorizontally,  
 verticalArrangement = Arrangement.Center  
 ) **{** Text(  
 "Paint Sort: Coloring Puzzle",  
 style = MaterialTheme.typography.headlineLarge,  
 fontWeight = FontWeight.Bold,  
 color = Color.White  
 )  
 Spacer(modifier = Modifier.*height*(48.*dp*))  
 MenuButton(  
 text = "Start Game",  
 onClick = onStartGame  
 )  
 Spacer(modifier = Modifier.*height*(24.*dp*))  
 MenuButton(  
 text = "Settings",  
 onClick = onSettings  
 )  
 Spacer(modifier = Modifier.*height*(24.*dp*))  
 MenuButton(  
 text = "Exit",  
 onClick = **{** (context as? Activity)?.finish() **}** )  
 **}  
 }  
 }**