Московский государственный университет имени М.В. Ломоносова

Факультет вычислительной математики и кибернетики

Кафедра алгоритмических языков

Вербин Олег

Программная реализация обучающей игры на базе беспроводной локальной сети

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

Научный руководитель:

м.н.с. Н.В.Баева

Москва, 2021

Аннотация

В рамках данной выпускной квалификационной работы было необходимо перевести уже существующую обучающую игру в цифровой вид, и разработать способ по созданию и запуску других вариаций этой игры. Для решения этой задачи было разработано компьютерное кроссплатформенное приложение и сетевое мобильное приложение под Android. Компьютерное приложение позволяет создавать сценарии игр, а мобильное приложение позволяет эти сценарии запускать. Для создания сети внутри мобильного приложения используется технология WiFi Direct.

Оглавление

1 Введение

В современном мире спрос на обучающие программы только растёт [7]. Это связано с тем, что скорость жизни значительно возросла, и количество окружающей нас информации повысилось в разы. Люди больше не хотят долгое время сидеть за чтением книг - для обретения нужных навыков им значительно проще получать новые знания в игровой форме.

В данной выпускной квалификационной работе рассмотрено насколько обучающие приложения на самом деле могут помогать в обучении, а также описаны некоторые примеры существующих обучающих игр.

Основная часть работы была посвящена разработке кроссплатформенного компьютерного приложения, для создания множества различных шаблонов пошаговых сетевых игр, и мобильного приложения под Android, для реализации выбранного шаблона. Был придуман гибкий способ задания шаблона, и его дальнейшего вычисления.

Для разработки приложений были использованы языки Java[4] и Python[10], и среды разработки Android Studio[2] и PyCharm[5].

Также в рамках работы были рассмотрены методы создания беспроводной связи на близкой дистанции.

Дополнительно в работе происходит краткий обзор Android Studio, а также описано каким образом можно организовывать меж-потоковое взаимодействие в Android Studio[3].

2 Постановка задания

Реализовать мобильное приложение для географического факультета МГУ им.М.В.Ломоносова, позволяющее проводить сетевые обучающие игры. В ходе работы требовалось:

- изучить и формализовать правила обучающей игры, проводимой на географическом факультете;
- разработать и реализовать сетевое приложение, позволяющее проводить обучающую игру на мобильных устройствах;
- разработать язык сценариев, позволяющий проводить схожие игры без изменение кода основной программы;
- спроектировать и реализовать приложение, позволяющее создавать сценарии игр;
- предусмотреть возможность загрузки созданного сценария в приложение ведущего игры.

3 Правила обучающей игры

На географическом факультете иногда проводится игра – некоторое количество (4-6) студентов собираются в одном месте. Каждый игрок выбирает одну из доступных **ролей**: *Глава комиссии по ЧС, представитель медицины, представитель географии и т.д.*

Игрокам сообщается о том, что в больницу поступили заболевшие люди с подозрительными симптомами, и, что существует вероятность эпидемии. С этого момента задача игроков – общаясь друг с другом**, определить и вылечить болезнь**, и сделать это с наименьшими потерями

Изначально игрокам доступно несколько **карточек больных** и выделено некоторое количество **монет**. Также игрокам известно о **количестве заболевших и умерших**.

В игре участвует **ведущий**, который может устраивать случайные события и давать подсказки

Игра является **пошаговой**. С каждым ходом **количество больных увеличивается.** Ведущий в начале хода, в зависимости от нынешней ситуации, сам пересчитывает количество больных и сообщает новое значение игрокам.

Каждой роли доступны свои **действия**, которые **стоят монет**. В течении хода игроки предлагают свои действия главе комиссии по ЧС. В конце хода **глава комиссии по ЧС решает** какие из предложенных действий стоит предпринять

В зависимости от действий игроков им может открываться **дополнительная информация**: *новые карточки больных, эпидемиологические карты, и т.д.* Заражаемость также может изменяться в зависимости от предпринятых действий.

Основная задача игры заключается в том, чтобы научить людей общаться друг с другом. Предпринять действие может только глава комиссии по ЧС, а количество монет ограничено, поэтому игроки обязаны вступать друг с другом в дискуссию и объяснять почему их действие является наилучшим.

4 Игровое обучение

«Игра практически с древних времён выступает как форма обучения».

Ян Коменский

Все дети любят играть. Потребность в игре одна из базовых потребностей человека. Многие считают, что игра — это прерогатива детей дошкольного возраста. Это не совсем верно. Исчезновение игры в школьном возрасте происходит не от того, что дети не хотят играть, а скорее всего из-за отсутствия объективных возможностей её осуществления. Конечно же потребность в игре сохраняется на протяжении всей жизни. Как маме легче уговорить ребёнка выполнить то или иное поручение, прибегнув к игре, так и преподавателю эффективнее и проще преподнести любой материал, используя игровой момент. Обучающая игра выступает как средство побуждения, стимулирования и реализации поставленной задачи в игровой форме.

Но насколько игра может быть эффективной в обучении? Масштабные исследования в середине 2000-х доказали: многопользовательские онлайн-игры – это эффективный тренажер для отработки навыков управления людьми [9]. В процессе других исследований [11] было установлено, что участники игры «Мафия» отличаются от своих сверстников более развитыми коммуникативными и интеллектуальными способностями, у них выше лидерский потенциал, они в большей степени склонны доверять людям и не беспокоиться из-за мелочей. Также при возникновении трудностей они предпочитают искать новые пути их решения, а не возвращаться к проверенным способам. Итог: игры – учат. Поэтому можно говорить об игре, как об **обучающем средстве**.

Согласно исследованиям рынка технологий [7], в 2018 году потребители потратили примерно $1,9 млрд на приложения для тренировки мозга, такие как Lumosity, Peak и Elevate, — в четыре раза больше по сравнению с $475 млн в 2012 году. Это говорит о том, **что спрос на обучающие игры с каждым годом только растёт**.

Стоит помнить, что игры развиваются вместе с обществом - у каждого поколения свои игры. Сейчас, когда произошла цифровизация общества - игры также перешли в цифровой вид. Мобильные и компьютерные приложения являются мощным средством повышения разносторонности личности и значительно расширяют возможности

человека по реализации собственных идей путем моделирования различных ситуаций в игровой среде. Примеры современных обучающих игр приведены далее.

Lightbot

Lightbot - это мобильная игра, которая создана, чтобы научить детей составлять базовые алгоритмы. Игроку показывается игровое поле, которое состоит из плиток, расположенных на разной высоте. Также на поле присутствует робот, а некоторые плитки помечены специальным цветом.

Игрок должен составить из доступных действий (шаг вперёд, поворот по часовой стрелке, прыжок, и т.д.) алгоритм, который проводит робота из его начальной позиции по всем помеченным плиткам (см. Рисунок 1). Со временем сложность заданий растёт, игроку даётся возможность создавать циклы, а количество действий на уровень становится ограниченным

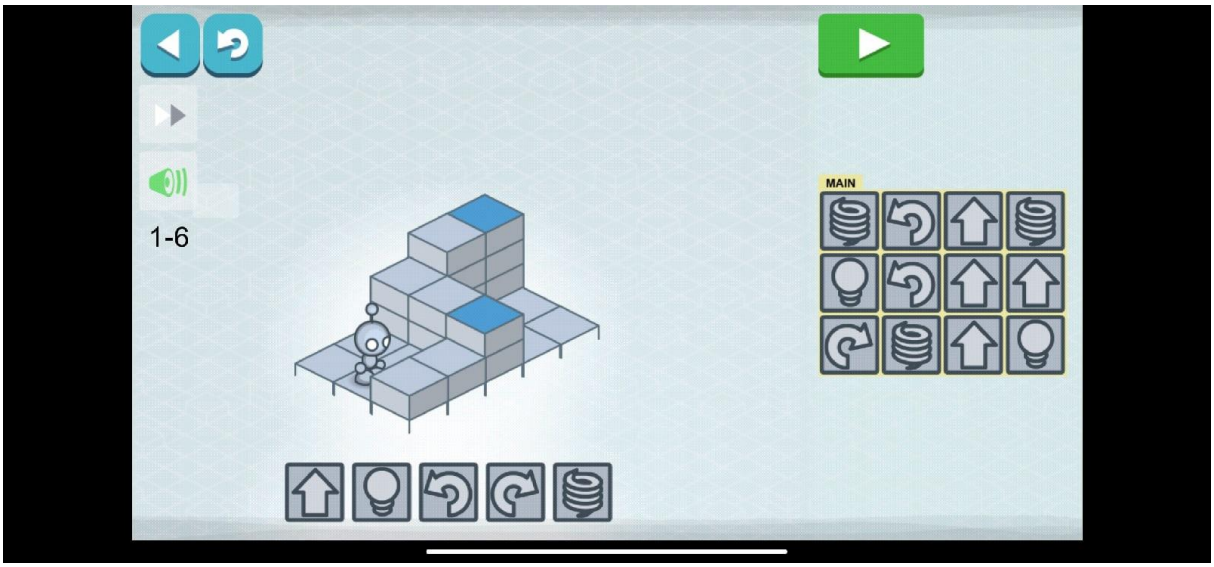


Рисунок 1 Lightbot

Lightbot позволяет в игровой форме научить детей алгоритмам. Интерфейс крайне простой и отзывчивый, картинка приятная глазу - это факторы, которые помогут привлечь внимание ребёнка. В малом возрасте очень сложно научить ребёнка чему-то подобному, но благодаря этой игре он будет обучаться, получая при этом удовольствие.

Scratch

Scratch - это визуальная событийно-ориентированная среда программирования, а также сайт, который позволяет достаточно просто создавать интерактивные истории, игры и мультфильмы, а затем делиться своими творениями с другими участниками онлайн-сообщества.

Scratch состоит из множества командных блоков, например: передвинуться вперёд на n шагов, повернуться по часовой стрелке на n градусов, и т.д. Пользователь может загрузить свои изображения в scratch и создать сложную логику по перемещению этих изображений, посредством командных блоков (см. Рисунок 2). Также блоки позволяют организовывать циклы и проверять условия, благодаря чему функционал значительно расширяется

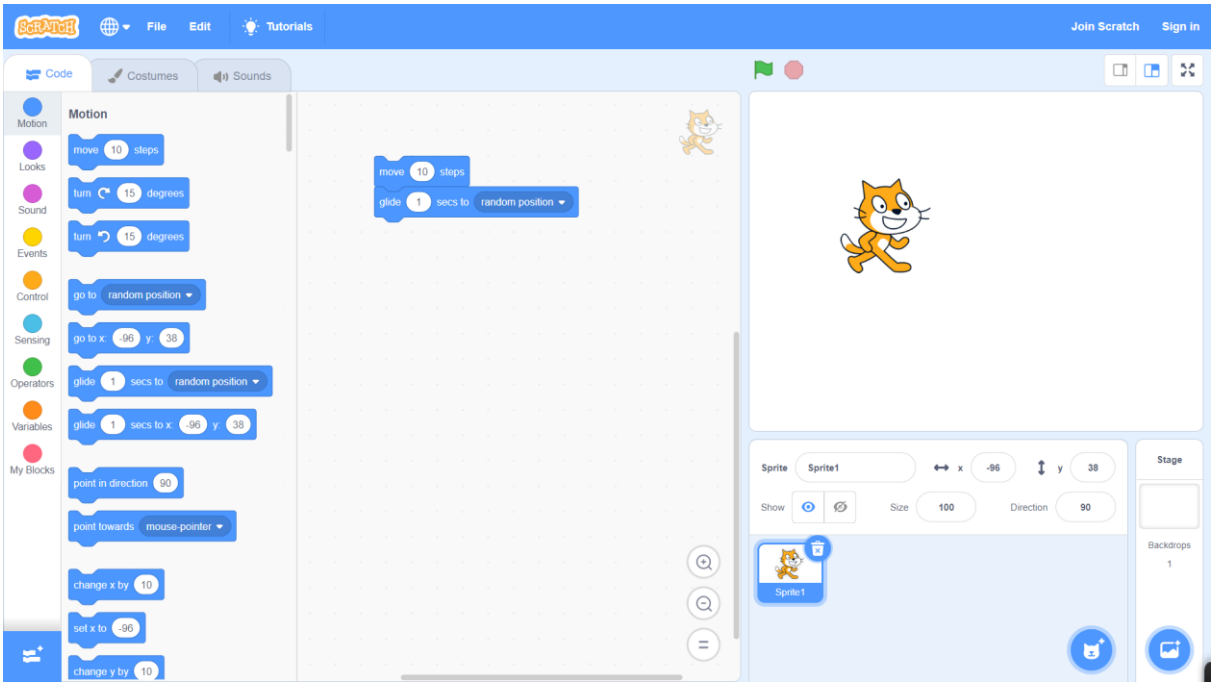


Рисунок 2 Scratch

Scratch помогает молодым людям учиться мыслить творчески, рассуждать системно и работать совместно — это необходимые навыки для жизни в 21 веке. Возможность загружать свои творения на сайт дополнительно мотивирует детей развиваться в этой области. Scratch - это тот самый первый шаг, который необходим ребёнку, чтобы проявить себя и в будущем стать экспертом в этой области.

LEGO MINDSTORMS

LEGO MINDSTORMS - это несколько роботов построенных из LEGO, а также приложение, которое позволяет использовать Scratch, чтобы программировать поведение этих роботов.

Роботы оснащены моторами и различными датчиками, и устроены таким образом, что их можно разбирать и использовать для своих целей. После сборки робота необходимо, используя язык Scratch и специальное приложение LEGO MINDSTORMS, запрограммировать поведение робота (см. Рисунок 3)

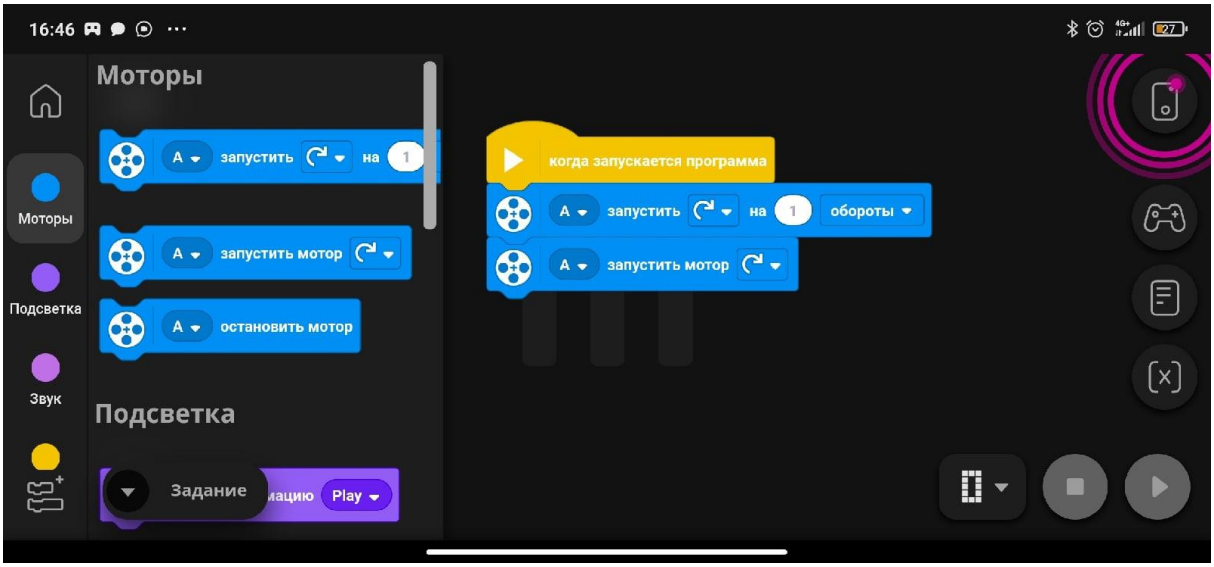


Рисунок 3 LEGO MINDSTORMS

Конструируя и оживляя роботов, дети проявляют не только свои творческие способности, но и через игру приобретают базовые навыки в области робототехники.

Duolingo

Duolingo – это приложение, которое позволяет в игровой форме учить иностранный язык.

Приложение состоит из большого количества уроков с постепенно повышающейся сложностью. Duolingo следит за твоим прогрессом и мотивирует тебя различными бонусами. В игре есть рейтинг и система очков, которые также позволяют оставаться мотивированным.

Каждый урок состоит из теоретической части и практических упражнений, в которых нужно переводить предложения, воспринимать слова на слух и практиковать говорение (см. Рисунок 4).

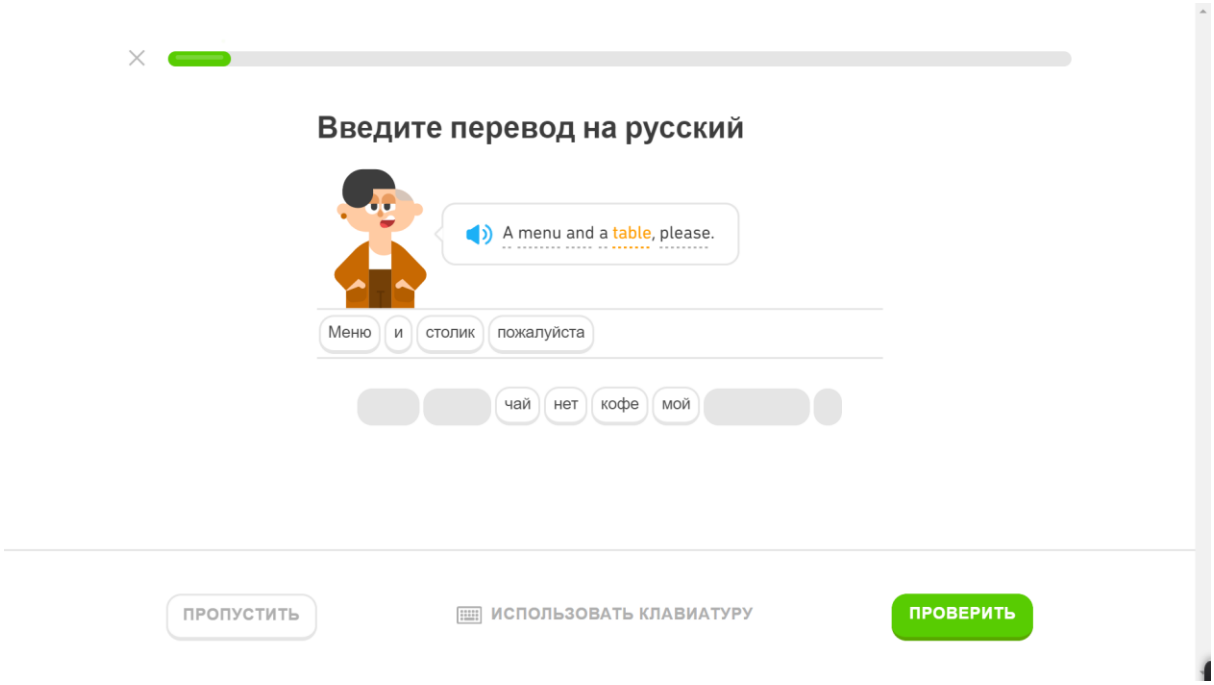


Рисунок 4 Duolingo

Исследование, проведённое профессорами Городского университета Нью-Йорка и Университета Южной Каролины [6], показало, что 34 часа на Duolingo дают столько же навыков чтения и письма, сколько даёт начальный семестровый курс в американском высшем учебном заведении, занимающий около 130 часов.

Итог

Все рассмотренные игры позволяют в простой форме получить ранее неизвестные навыки. Через игру человек может учиться с удовольствием, а это в свою очередь значительно повышает скорость усвоения информации. Также игры зачастую используют простые визуальные образы, а многие люди проще воспринимают информацию визуально. Дополнительным немаловажным фактом является то, что игры в большинстве своём интерактивны, поэтому пользователь может сразу посмотреть интересующие его моменты, провести нужные тесты и сразу-же увидеть, где он ошибается. Благодаря всем этим фактором обучающие игры очень эффективны, и в будущем, скорее всего, будут всё активнее развиваться.

5 Язык описания сценария

Вместо того, чтобы разрабатывать приложение, которое будет способно запустить одну конкретную игру, лучше разработать приложение, которое сможет запускать большое множество схожих игр.

Для этого необходимо разработать конструктор сценариев и определить способ задания различных сценариев для игр. В конструкторе обязательно должна быть возможность задать вышеописанную игру, при этом, чем больше различных игр можно задать в конструкторе, тем он лучше.

5.1 Формализация игры

В первую очередь необходимо понять, из каких элементов будет состоять игра. На основе анализа вышеописанной игры были выделены следующие элементы:

- **Переменные** (*число заболевших и монеты*) – это объекты, которые хранят в себе действительное значение
- **Листы** (*карточки больных, эпидемиологические карты*) – это объекты, которые хранят в себе изображение и флаг видимости (доступно ли данное изображение игрокам, или нет)
- **Роли** (*глава комиссии по ЧС, представитель медицины*) – это объекты, которые определяют список возможных действий игрока
- **Функции** – это **правила** по которым значение переменных изменяется в начале каждого хода (*число больных увеличивается*)
- **Действия** (*купить вакцину*) – это объекты, которые хранят в себе условие на действие, реакцию, варианты ответов, последний выбранный ответ и количество использований
- **Условие на действие** (*у игрока должно быть >= N монет*) – это предикат от переменных, листов и действий, который говорит, можно ли выполнять действие
- **Реакция** (*отнять N монет, показать эпидемиологические карты*) – это **правило** по которому значения переменных и флаги видимости листов изменяются при активации действия

- **Условие на завершение игры** – это предикат от переменных, листов и действий, который говорит, завершилась ли игра

Правило – это сущность, которая должна позволять изменять переменные, листы и действия, опираясь на их нынешнее значение. Пусть в игре есть n переменных, m листов и d действий. Составим вектор из n значений переменных, m значений флагов видимости листов и 2d выбранных ответов и количеств использований действий. Тогда правило – это отображение $f: \mathbb{R}^{n+m+2d} \to \mathbb{R}^{n+m+2d}$

Условие на действие (завершение игры) $f: \mathbb{R}^{n+m+2d} \to \mathbb{R}^{n+m+2d}$

Далеко не все отображения можно точно представить в памяти компьютера, поэтому нужно ввести ограничения, чтобы работать только с удобными отображениями.

Рассмотрим такие $f: \mathbb{R}^{n+m+2d} \to \mathbb{R}^{n+m+2d}$, что для них можно разбить пространство $\mathbb{R}^n$ на конечное число непересекающихся множеств $R_1 \cup R_2 \cup \ldots \cup R_k = \mathbb{R}^n$, таким образом, чтобы

$\forall i \in [1,2,\ldots,k] \; \forall x \in R_i \; f(x) = \vec{f_i}(x)$, где $\vec{f_i}$ задаётся **вектор формулой** из $\mathbb{R}^n$, и

$\forall i \in [1,2,\ldots,k-1] \; R_i$ задаётся **неравенством** из $\mathbb{R}^n$

То есть пространство $\mathbb{R}^n$ разбивается на подмножества таким образом, что на каждом из этих подмножеств отображение f имеет простой вид, и сами подмножества задаются простым способом. Теперь, чтобы задать подобное отображение достаточно хранить k **вектор формул** для $\vec{f_i}$ и k-1 **неравенств** для R_i

Вектор формула $\vec{f} = \overrightarrow{\{f_1, f_2, \ldots, f_n\}}$, f_i - является **формулой**, $\vec{f}: \mathbb{R}^n \to \mathbb{R}^n$

$\vec{f}(x) = \overrightarrow{\{f_1(x), f_2(x), \ldots, f_n(x)\}}, \qquad x \in \mathbb{R}^n, f_i: \mathbb{R}^n \to \mathbb{R}$

$f_i(x) = g_1(X) \; op \; g_2(x) \; op \; \ldots \; op \; g_m(x), \qquad x \in \mathbb{R}^n, g_m: \mathbb{R} \to \mathbb{R}$

$g_m(x) =$ либо $h_j(x)$, либо $t_p\big(s_1(x), s_2(x), \ldots, s_p(x)\big)$

$h_j(x) =$ либо x_j, либо $const \;\; x \in \mathbb{R}^n, const \in \mathbb{R}$

$t_p(x) =$ одна из заданных функций $t: \mathbb{R}^p \to \mathbb{R}$ (sin, cos, max и т.д.)

$s_i(x)$ - **формула** $s: \mathbb{R}^n \to \mathbb{R}, \; x \in \mathbb{R}^n$

op - {+,-,/,*,&,|} математические и логические операции

$x_1 \,\&\, x_2 \;=\; (x_1 \neq 0)$ "логическое и" $(x_2 \neq 0), \qquad x_1, x_2 \;\in \mathbb{R}$

$x_1 \mid x_2 \;=\; (x_1 \neq 0)$ "логическое или" $(x_2 \neq 0), \qquad x_1, x_2 \;\in \mathbb{R}$

Другими словами, формула – это функция от n вещественных переменных, состоящая из вещественных чисел, заданных функций и операций из op.

Неравенство $e(x)$ из $R^n \rightarrow \{0,1\}$

$$e(x) = 0\ cond\ \ s(x)\ \textit{или}\ e_1(x)\&e_2(x)\ \textit{или}\ e_1(x)|e_2(x), \qquad s(x) - \textit{формула} : R^n \rightarrow R$$

$$e(x), e_2(x)\ -\ \textit{неравенство из}\ R^n \rightarrow \{0,1\}$$

cond – $\{=, \neq, <, >, \leq, \geq\}$

Пример формулы:

f(x1,x2,x3) = x1 + x2

*f(x1,x2,x3) = x1*x2+sin(cos(max(25/x1,x3)))*

f(x1,x2,x3) = x1|x2&x3

Пример задания области с помощью неравенства:

*e(x1,x2) –> x1*x1 + x2*x2 < 25 – внутренность круга*

*e(x1,x2) –> 3*x1-x2>0 - полуплоскость*

*e(x1,x2) –> x1*x1 + x2*x2 <1 & x1>0 & x2>0 – положительная четверть круга*

5.2 Конструктор сценариев

В получившемся представлении можно изменять каждый элемент. Варьируя эти параметры получим всё **множество возможных игр**. При этом чем больше функций находится в заданных функциях (t_p: $\mathbb{R}^p \rightarrow \mathbb{R}$), тем больше множество возможных игр.

Так-как данное представление было получено из анализа вышеописанной игры - эта игра принадлежит множеству возможных игр.

Чтобы задавать сценарии можно использовать нынешнее представление игры, но это не очень удобно. Для облегчения создания сценария были введены следующие изменения:

В конструкторе всегда существует переменная, в которой хранится номер текущего раунда, а также роль хоста (игрока, создавшего игру, имеющего возможность изменять переменные и листы по своему усмотрению) и лидера (игрока, к которому будут отправляться действия остальных, для подтверждения)

С переменными ассоциируется начальное, минимальное и максимальное значение. Теперь, если значение переменной выходит за заданные границы, то оно изменяется на ближайшее граничное значение. Дополнительно вводятся невидимые переменные – переменные, которые не видны игроку, но их всё ещё можно использовать в правилах.

Вводятся **вкладки** – это объекты, которые хранят в себе список листов и флаг видимости (видны ли игроку все листы внутри вкладки). Функциональный смысл вкладок заключается в группировании листов. Теперь все листы распределены по вкладкам, а в правилах можно использовать флаг видимости вкладки

Действие дополнительно хранит в себе положительный целочисленный атрибут «максимальное количество использований» - действие может быть использовано максимум столько раз. Реакция разбивается на **условия на реакцию** (предикаты от переменных, листов и действий) и **подреакции** (правила). Подреакция выполняется, только, если выполнилось соответствующее условие на реакцию.

Функция разбивается на **условия на функцию** (предикаты от переменных, листов и действий) и **подфункции** (правила), при этом задан порядок на подфункции. При выполнении функции, происходит проход по заданному порядку по условиям на функцию, и, в случае срабатывания условия на функцию, выполняется соответствующая подфункция, и выполнение функции на этом завершается.

По сути условие на реакцию/функцию задаёт R_i (разбиение $\mathbb{R}^n$), а подреакция/подфункция задаёт $\overrightarrow{f_i}$ (вектор формулу). При этом теперь можно задавать пересекающиеся R_i, тогда, если $x \in$ пересечению, то выполнятся все соответствующие подреакции и первая по порядку подфункция

Функции ассоциируются с переменными – каждая переменная хранит в себе ровно одну функцию. Теперь в начале хода происходит проход по всем переменным и выполнение соответствующей функции, при этом изменение переменных, листов и действий происходит только по завершению всех функций.

Таким образом функция не может изменить значение переменной и повлиять этим на работу другой функции, так что можно считать, что функции работают с переменными, листами и действиями такими, какими они были в начале раунда, при этом все функции выполняются одновременно. Такая же логика с одновременным выполнением распространяется и на реакции

Добавленные изменения никоим образом не изменяют множество возможных игр.

Для того, чтобы задавать формулы, необходимо задать функции $h_j(x) = x_j$, то есть нужно научиться получать соответствующее значение переменной, листа или действия. **Для этих целей каждому элементу состояния игры выдаётся некоторый уникальный идентификационный номер (id)**. Теперь, если внутри формулы использовать id, то вернётся соответствующее значение.

Для переменной – это значение переменной, для листа и вкладки – это значение флага видимости, для действия – это количество использований. Чтобы получить последний вариант ответа для нужного действия, используется функция ans(id), которая возвращает целое число (порядковый номер ответа, или -1, если действие ещё не было использовано, или оно не предполагает выбор из нескольких вариантов)

Теперь для задания формулы осталось лишь определить функции $t_p(x)\colon R^p \to R$. Чем больше этих функций описано, тем больше множество возможных игр. Список заданных функций приведён в **приложении А**.

Для того, чтобы задать вектор формулу, необходимо задать каждую её компоненту, однако на практике обычно необходимо изменять лишь какое-то небольшое количество элементов, поэтому задавать все компоненты – излишне. Для задания нужной компоненты вектор формулы введены функции eq и show.

eq(id,value), необходимо для того, чтобы менять значение переменной. Использование eq(id,value) **внутри подфункции или подреакции** равносильно заданию

компоненты, которая соответствует переменной с заданным id, для вектор формулы ($f_i = value$). При этом, если внутри подфункции или подреакции используется несколько eq с одинаковым id, то для задания компоненты для id используется только последняя функция eq. eq(id,value) возвращает 1, если value не превышает граничные значения соответствующей переменной, иначе возвращает 0.

show(n,id0,id1,…idk), нужна для того, чтобы показывать игрокам листы, то есть выставлять значения флагов видимости. Использование show(n,id0,id1,…idk) **внутри подфункции или подреакции** равносильно заданию компонент, которые соответствуют листам и вкладкам с заданными id, для вектор формулы ($f_i = 1$). Число n означает, что среди заданных листов и вкладок выбирается случайным образом не больше n листов, которых игрок ещё не видел, и для них происходит изменение флага видимости. Если n = 0, то выбираются все листы из перечисленных. show всегда возвращает 1.

При активации действия происходит следующее:

- Проверяется условие на действие. Если оно не проходит, то действие нельзя применить, иначе выполнение продолжается.
- Действие применяется, то есть обновляется выбранный ответ и количество использований.
- Применяется реакция, то есть выполняются все подреакции, для которых верно условие на подреакцию.
- Проверяется условие на победу и условие на проигрыш (в таком порядке).

В начале хода происходит следующее:

- Переменная отвечающая за номер раунда увеличивается на 1.
- Для всех переменных одновременно выполняются соответствующие функции.
- Проверяется условие на победу и условие на проигрыш.

Пример задания сценария приведен в **приложении В**

6 Проектирование и разработка

6.1 Выбор технологии передачи данных

Так-как все игроки будут находиться в одной комнате – организовывать сетевое взаимодействие через интернет излишне. Для реализации сети на близкой дистанции существует несколько вариантов:

- Bluetooth – метод сетевого взаимодействия, основанный на использовании радиоволн с частотами 2.402-2.48 ГГц
- WiFi - технология беспроводной локальной сети с устройствами на основе стандартов IEEE 802.11. Работает на больших расстояниях и с большей скоростью, чем Bluetooth, но потребляет больше энергии и требует общей точки доступа.
- WiFi Direct – стандарт, позволяющий двум и более Wi-Fi-устройствам общаться друг с другом без маршрутизаторов и общих точек доступа

WiFi Direct быстрее Bluetooth и работает на больших расстояниях, не требует общей точки доступа, по сравнению с обычным WiFi, а так-же тратит меньше всего зарядки, за счёт умного распределения ресурсов. По этим причинам будет использоваться WiFi Direct

6.2 Инструменты разработки конструктора сценариев

В конструкторе сценариев необходимо задавать большое количество элементов и набирать много текста, поэтому удобнее всего реализовать конструктор в качестве компьютерного приложения.

Существует множество инструментов разработки под компьютер. Логичнее всего выбрать те инструменты, которые позволяют создавать кроссплатформенные приложения. Одним из таких инструментов является нативно встроенная в python графическая библиотека Tkinter [1].

Tkinter является одной из первых графических библиотек для python, и она до сих пор поддерживается, так что её эффективность была проверена временем. Например, *интегрированная среда разработки и обучения IDLE была создана с помощью библиотеки Tkinter*

Итог:

- Операционная система – **Windows, MacOs, Linux**
- Язык программирования - **Python**
- Среда разработки - **PyCharm**
- Графическая библиотека – **Tkinter**

6.3 Инструменты разработки приложения

Для работы приложения необходимо, чтобы все игроки находились в одной комнате. Запускать множество ноутбуков, и тем более компьютеров, в одной комнате неудобно, поэтому приложение должно работать на мобильных устройствах.

Существует две доминирующие на мобильном пространстве операционные системы – Android и IOS. Разработка под них довольно сильно отличается, а кроссплатформенные решения не обеспечивают той гибкости, которую можно получить, разрабатывая под конкретную операционную систему, поэтому нужно выбрать какую-то одну операционную систему и разрабатывать приложение только под неё.

Операционная система Android на 2021 занимает около 70% мирового рынка всех операционных систем, используемых на мобильных устройствах, для России этот показатель может быть даже выше, так как устройства именно с этой операционной системой являются наиболее доступными по цене. К тому-же установка сторонних приложений в Android значительно проще, чем в IOS. По этим причинам разработка будет вестись только под Android.

Итог:

- Операционная система – **Android**
- Язык программирования - **Java**
- Среда разработки - **Android Studio**
- Графические библиотеки – **внутренние библиотеки Android**
- Организация сети – **WiFi Direct (WiFi P2P) [8]**

6.4 Обзор Android Studio

В Android Studio для создания приложения необходимо работать с 3 типами файлов: **Manifest.xml, Layout.xml** и **Activity.java.**

- **Manifest** – здесь описываются: разрешения, которые могут понадобиться чтобы приложение заработало; технические и программные составляющие, которые необходимы для корректного запуска приложения; название пакета, в котором содержится весь код приложения; все компоненты из которых состоит приложение (все activity и прочее)

- **Layout.xml** – здесь описывается внешний вид приложения. Существует множество объектов (кнопки, текстовые поля, списки с объектами и т.д.), которые можно здесь использовать. Для размещения в пространстве существует несколько базовых слоёв, например, линейные слои (вертикальные и горизонтальные, всё размещается линейно друг за другом), constraint слои (необходимо связывать объекты горизонтальными и вертикальными линиями к другим объектам. Можно привязывать объекты к краям экрана) и др. Ещё каждому объекту здесь задаётся уникальный id.

- **Activity.java** – с точки зрения Java, это просто класс, унаследованный от AppCompatActivity. Activity является ключевым классом для приложения. Всё взаимодействие с пользователем происходит именно здесь.

У activity есть жизненный цикл, и чтобы понять, что происходит, необходимо его разобрать:

- **onCreate**() вызывается, как только activity было создано, или при повторном обращении к activity, при его уничтожении.
- **onStart**() вызывается всегда после onCreate() или после onRestart().
- **onResume**() вызывается всегда после onStart() или после onPause(), в случае, если пользователь вернулся к activity, и оно ещё не было уничтожено.
- **onPause**() вызывается после onResume(), если другое activity вышло на передний план.
- **onStop**() вызывается после onPause(), если activity больше не видно пользователю.
- **onRestart**() вызывается после onStop(), если пользователь вернулся к activity, и оно ещё не было уничтожено.
- **onDestroy**() вызывается после onStop(), если activity завершено или убивается системой.

Таким образом onResume() будет вызываться очень часто, поэтому здесь нежелательно размещать тяжёлые элементы, с другой стороны onCreate() вызывается редко, поэтому всю тяжёлую инициализацию необходимо выполнять именно здесь. Также в onCreate() необходимо позаботиться о визуальном виде приложения, то есть отобразить окно с одним из существующих Layout.xml. В **Приложении С** приведена схема, для лучшего понимания жизненного цикла activity.

Чтобы отобразить layout нужно внутри activity обратиться к функции setContentView(). Чтобы найти объект внутри layout нужно вызвать findViewById(). Далее для объектов можно изменить свойства (например, выставить текст) и зарегистрировать **callback** – действие, которое будет происходить при взаимодействии с объектом (например, при нажатии на кнопку)

6.5 Сокеты, handler и looper

Передача данных между устройствами реализована через **сокеты**. Как только устройства наладят связь через WiFi Direct и окажутся в одной группе, необходимо обеспечить обмен информацией посредством сокетов

Общение между сокетами организовано классическим образом: хост в вечном цикле принимает новые подключения, и на каждого клиента создаются два потока - один для чтения из сокета, один для записи в сокет.

Также необходима возможность действовать на эти потоки со стороны activity и наоборот. В Android, если объект был создан каким-то activity, то только этот activity может внести какие-либо изменения в объект. *Например, сторонний поток не сможет изменить какой-либо объект в layout.*

Для решений этой проблем в android существует **handler** и **looper**.

- **Looper** - при запуске приложения создаётся главный поток и главный looper (По сути - очередь с кодом). Главный поток в вечном цикле выполняет код из looper-а (в порядке очереди). Не рекомендуется занимать главный поток тяжёлыми операциями, так как он отвечает за интерфейс. Необходимо, чтобы пользователь мог взаимодействовать с приложением, даже если на фоне производятся вычисления.
- **Handler** - это объект, который можно ассоциировать с конкретным looper-ом и он позволяет отправлять в looper свой код на выполнение.

Таким образом каждому потоку нужно ассоциировать handler и работать через него. То есть, если нужно изменить какой-то объект внутри layout из внешнего потока – нужно отправить соответствующий код на handler, ассоциируемый с главным looper-ом.

6.6 Архитектура приложения

Хосту необходимо ожидать подключения игроков и налаживать взаимодействие с этими игроками (см. Рисунок 5).

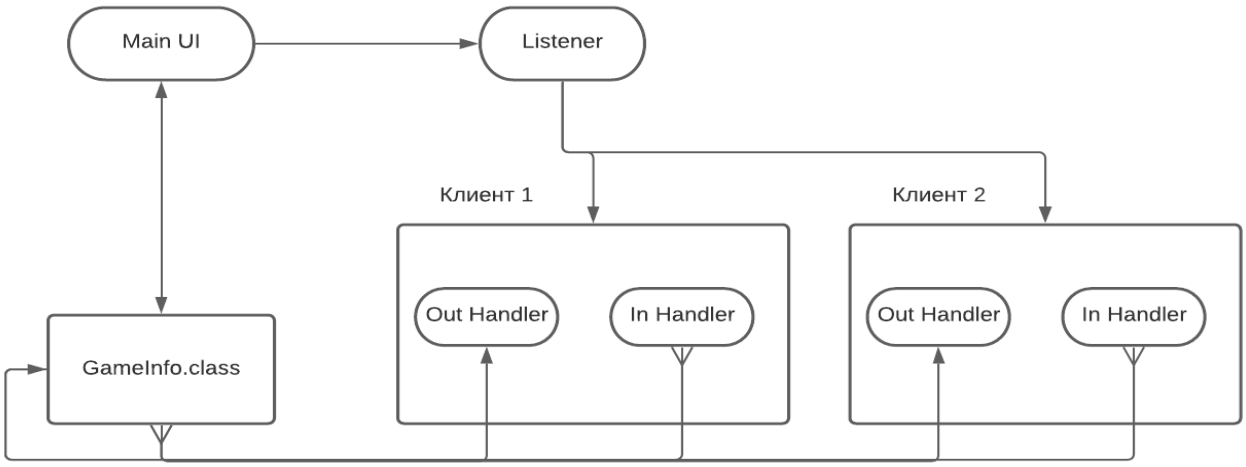


Рисунок 5 Архитектура Хоста

Клиенту необходимо подключиться к хосту и взаимодействовать с ним (см. Рисунок 6).

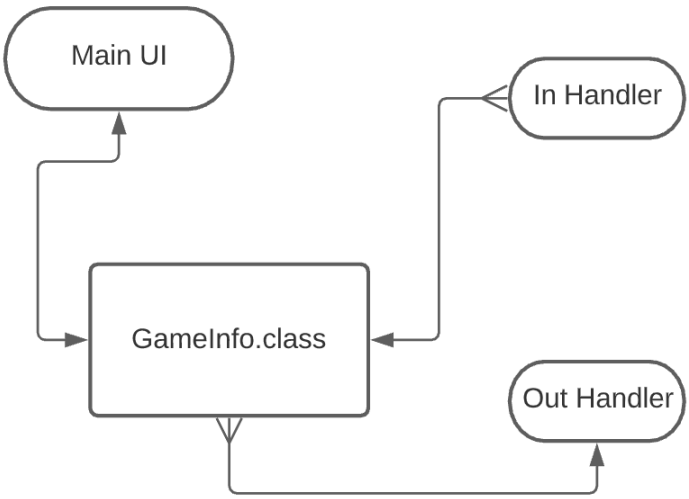


Рисунок 6 Архитектура Клиента

Main UI – основной поток. Создаётся при запуске приложения. Отвечает за отображение всего интерфейса.

Listener – поток, который принимает новые соединения на сокете. При налаживании нового соединения, создаёт In Handler и Out Handler.

In Handler – поток, который используется, чтобы прочитать данные с сокета. Читает данные с сокета построчно и вызывает метод GameInfo.parseLine(string)

Out Handler – поток, который используется, чтобы записать данные в сокет. Если главный поток хочет что-то передать по сети, он вызывает метод GameInfo.print(string,to)

GameInfo.class – класс, который хранит в себе всё состояние игры, а также методы, необходимые для взаимодействия между потоками. Метод parseLine(string) обрабатывает строку и делает соответствующие изменения в главном потоке. Метод print(string,to) заставляет соответствующий (to) Out Handler записать строку в сокет

Для переменных, функций, вкладок, изображений, ролей и действий существует свой класс. Во всех этих классах есть метод перевода в строку – он представляет объект в виде строки, готовой к передаче по сокету.

Такой подход значительно упрощает взаимодействие между потоками. Теперь, чтобы передать объект, достаточно вызвать его метод, для получения правильного строкового представления, и затем передать эту строку в метод GameInfo.print. Чтобы принять объект, достаточно написать соответствующий код в GameInfo.parseLine.

7 Интерфейс конструктора сценариев

Было разработано компьютерное приложение для создания и редактирования сценария (см. Рисунок 7). Приложение позволяет задавать все элементы из которых состоит сценарий и автоматически проставляет для них id. Интерфейс приложения разделён на 7 областей, по одной области на переменные/вкладки/роли/функции/листы/действия и отдельная область, которая находится справа и служит для создания/редактирования выбранного объекта.

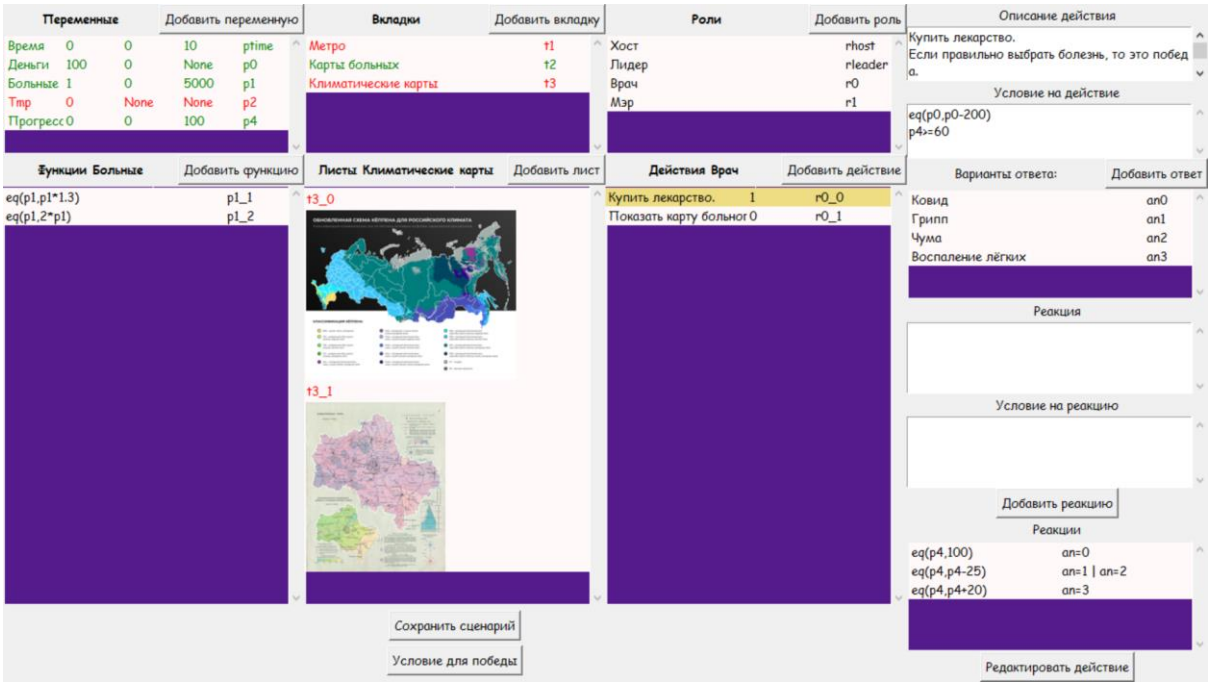


Рисунок 7 Интерфейс конструктора сценария

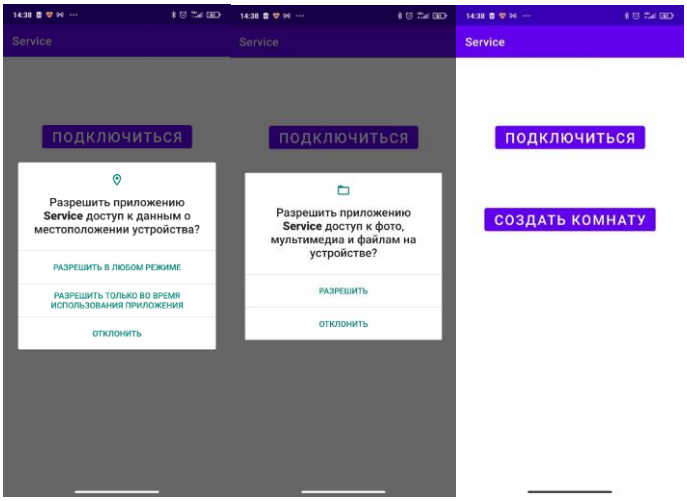
- При старте программы можно либо выбрать существующий сценарий и отредактировать его, либо создать новый сценарий.
- Esc - выход из полноэкранного режима.
- При клике на предмет – он выделяется специальным цветом.
- Del – удаляет выделенный объект.
- При нажатии на кнопку “Добавить переменную/вкладку/..” создается соответствующий объект. Для переменной и вкладки создаётся стандартный элемент и сразу-же добавляется в интерфейс. Для функций/ролей/действий

заполняется самая правая область, и в ней уже можно задать нужный значения для элемента и поместить его в интерфейс.

- Двойной клик открывает редактирование.
- При клике на переменные/вкладки/роли в соответствующей области показываются соответствующие функции/листы/действия.
- При нажатии на кнопку “Добавить лист” открывается файловый менеджер в котором можно выбрать изображение (.jpg или .png).
- Двойной клик на картинку внутри листа открывает её в новом окне. Здесь можно изменять размер окна, чтобы лучше разглядеть картинку.
- Двойной клик на id изменяет видимость объекта. Видимые объекты выделяются зелёным цветом, невидимые – красным.
- Внутри функций можно выделить кликом предмет, а затем использовать стрелки (Up и Down) чтобы изменить местоположение объекта (функции выполняются сверху вниз, при этом выполнение обрывается при первом срабатывании, поэтому порядок важен).
- Минимальное/максимальное значение переменной может быть только числом или none, что означает отсутствие предельного значения.
- Начальное значение в переменной с индексом ptime показывает продолжительность раунда в секундах (0 – значит не ограничено по времени). Максимальное значение показывает количество раундов (0 или none – значит нет ограничения по раундам).
- Кнопка “Условие на победу” заполняет самую правую область таким образом, что в ней можно задать название сценария, условие на победу и условие на проигрыш.
- Автоматически проверяется корректность формул и указываются ошибки.
- Кнопка “Сохранить сценарий” открывает файловый менеджер в котором можно указать путь, по которому будет сохранён сценарий

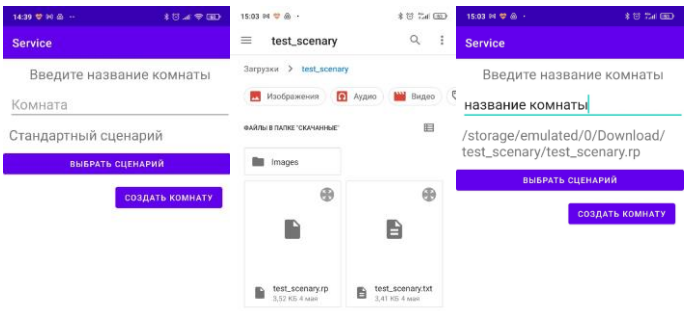
8 Интерфейс мобильного приложения

Запуск



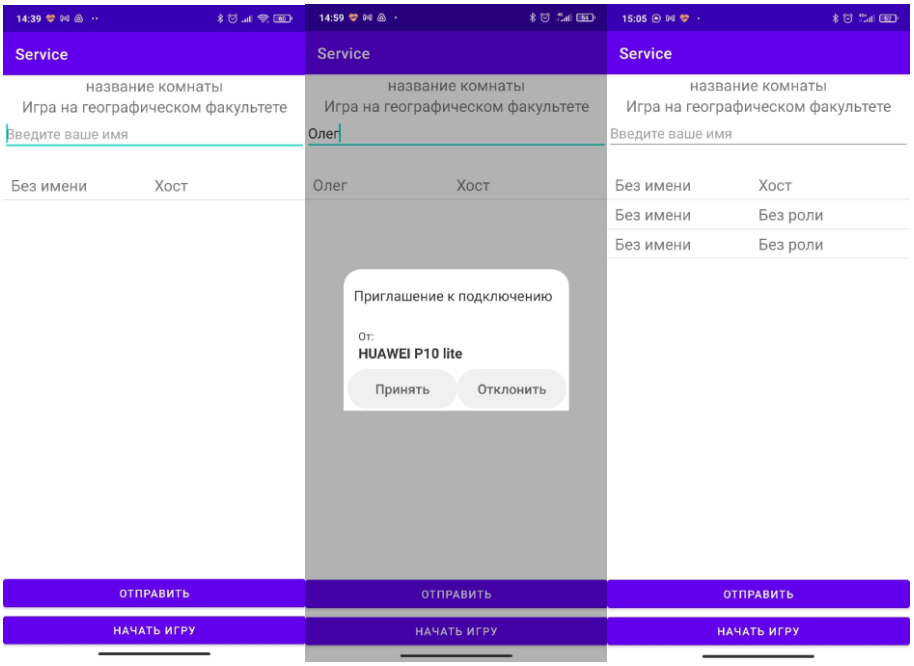
Приложению, для корректной работы необходим доступ к файлам на устройстве (для загрузки сценария) и доступ к геоданным (для работы WiFi Direct). При запуске приложение потребует эти разрешения. Также необходимо включить WiFi и геоданные, иначе сеть будет невозможно создать и приложение будет сигнализировать о выключенном WiFI.

Создание комнаты



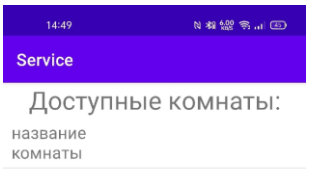
Здесь необходимо указать название комнаты и указать путь к файлу со сценарием (*.rp). При нажатии на кнопку «выбрать сценарий» открывается файловый менеджер, в котором можно выбрать нужный файл.

Комната ожидания (Хост)



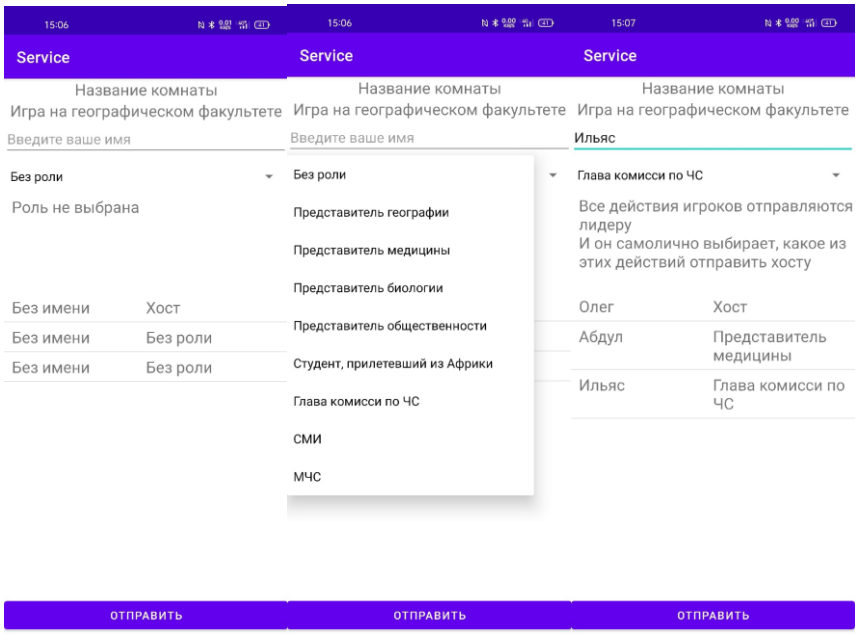
При создании комнаты открывается данное окно. Здесь хост может видеть подключённых игроков (их роль и имя), а также может переименоваться сам. При подключении каждого игрока всплывает соответствующее окно. Когда наберётся достаточное количество игроков хост может начать игру, к этому моменту все игроки должны выбрать роль, среди которых должен быть хотя-бы один лидер.

Подключение



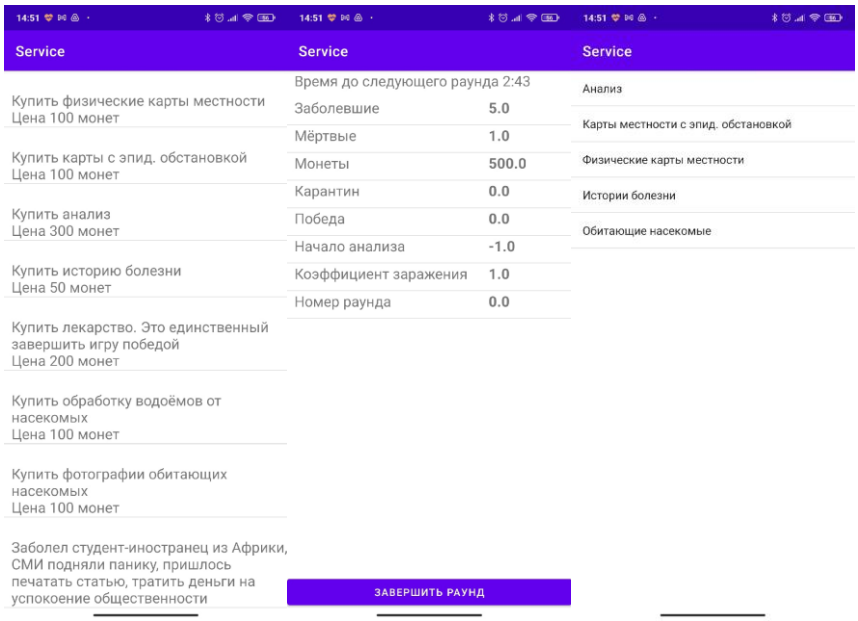
Здесь отображаются все доступные на данный момент комнаты. При нажатии на комнату, происходит подключение к ней.

Комната ожидания (Клиент)



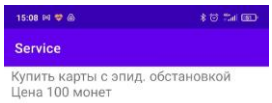
При подключении к комнате открывается это окно. Здесь пользователь видит всех игроков в комнате и может установить себе роль и имя.

Игра



При старте игры перед пользователем открываются окно с действиями, окно с основной информацией и окно с вкладками. Между окнами можно переключаться через свайпы влево-вправо.

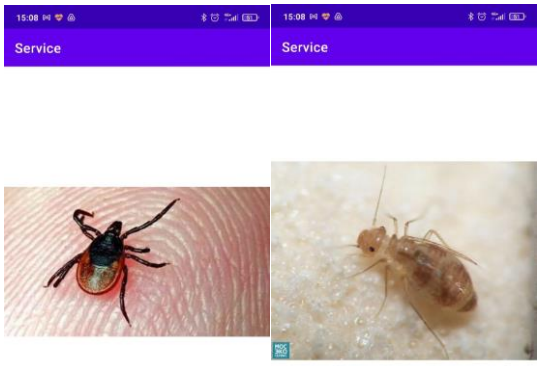
Действия



В окне с действиями отображаются все действия для данной роли. Хост видит все свои действия + действия всех остальных ролей. Действие может быть недоступно, в этом случае всплывёт соответствующее окно. Хосту доступны все действия.

При клике на действие, если оно доступно, открывается окно, в котором можно выбрать один из вариантов ответа, если действие предполагает несколько вариантов ответа.

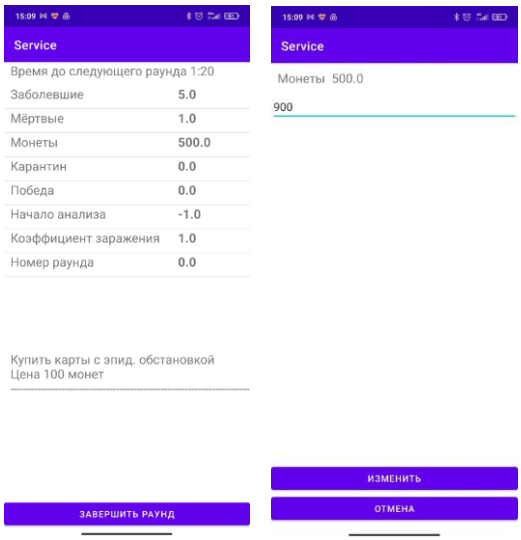
Вкладки



В окне с вкладками отображаются все видимые игроку вкладки. Хосту отображаются все вкладки в игре.

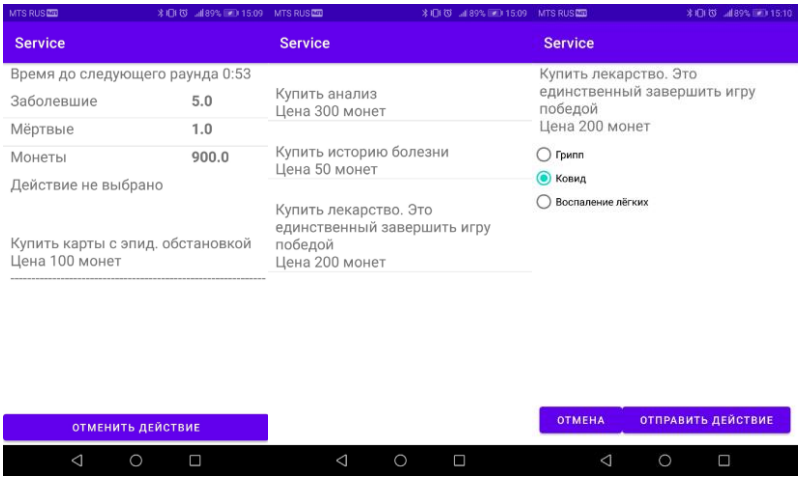
При клике на вкладку открывается её содержимое. Свайпами влево-вправо можно переключаться между картинками. Кнопка «отправить игрокам» есть только у хоста, и отображается, только если данного изображения нет у игроков.

Переменные

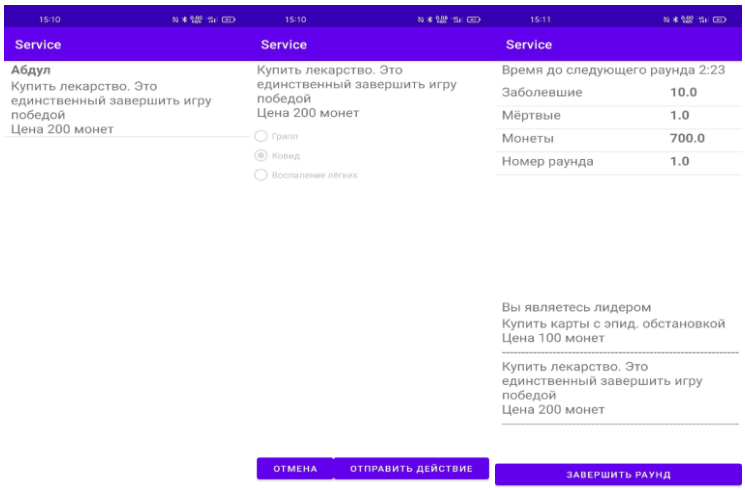


В центральном окне отображаются значения переменных и время до следующего раунда. Внизу отображаются все действия, которые были совершены к данному моменту. Хост может кликать на переменные, чтобы изменять их

Отправка действий



Если пользователь (не лидер) отправляет действие, то оно передаётся лидеру. Теперь лидер может это действие подтвердить, и тогда оно отправится хосту и начнётся новый раунд.



Конец игры

Когда таймер отсчитывает заданное время – начинается новый раунд. Если игроки не успеют победить за отведённое им время (в раундах), то они проиграют.

При победе или поражении соответствующая информация записывается в журнал действий, и игра перестаёт отвечать на действия игрока.

9 Результаты работы

Требовалось разработать приложение, позволяющее проводить сетевые обучающие игры. В результате выполнения данной выпускной квалификационной работы поставленная задача была решена. В ходе работы были выполнены следующие пункты:

- Формализованы правила обучающей игры, проводимой на географическом факультете. Выделена структура игры и её основные элементы.
- Разработано сетевое мобильное приложение под Android с использованием технологии WiFi Direct. Приложение позволяет запускать загруженные сценарии обучающих игр.
- Определено множество возможных игр и разработан способ, позволяющий создать сценарий любой игры из множества возможных игр.
- Разработано кроссплатформенное компьютерное приложение для создания сценариев игр. Приложение написано на языке Python с использованием графической библиотеки Tkinter. Приложение позволяет создавать, редактировать и сохранять сценарии для игр.
- Реализована возможность загрузки любого созданного сценария в мобильное приложение ведущего игры. Ведущий может запустить этот сценарий, и другие игроки смогут присоединиться к созданной игре.

10 Литература

1. Alan Moore. Python GUI Programming with Tkinter: Develop Responsive and Powerful GUI Applications with Tkinter.- Бирмингем: Packt Publishing,2018.- С. 454
2. Android Studio [Электронный ресурс].- Электрон.дан. - URL: https://developer.android.com/studio/intro. (дата обращения: 08.12.2020).
3. Handler [Электронный ресурс].- Электрон.дан. - URL: https://betterprogramming.pub/a-detailed-story-about-handler-thread-looper-message-queue-ac2cd9be0d78. (дата обращения: 08.12.2020).
4. Joshua Bloch. Effective Java.- Boston : Addison-Wesley, 2018.-С. 392
5. PyCharm [Электронный ресурс].- Электрон.дан. - URL: https://www.jetbrains.com/pycharm/. (дата обращения: 26.03.2021).
6. Roumen Vesselinov, John Grego. Изучение эффективности Дуолинго (англ.) = Duolingo Effectiveness Study. — Нью-Йорк, Колумбия (Южная Каролина), 2012. — С. 25.
7. Trend: Consumers spend significantly more on digital brain health and neurotechnology apps [Электронный ресурс].- Электрон.дан. - URL: https://sharpbrains.com/blog/2019/05/24/trend-consumers-spend-significantly-more-on-digital-brain-health-and-neurotechnology-apps/. (дата обращения: 03.04.2021).
8. WiFi Direct [Электронный ресурс].- Электрон.дан. - URL: https://developer.android.com/training/connect-devices-wirelessly. (дата обращения: 08.12.2020).
9. Гроголева О. Ю, Анкудинова В. С. Типологические особенности личности участников интеллекутально-психологической игры "Мафия"// Вестник Омского университета. Серия "Психология".-2014.№2. С.4-12
10. Марк Лутц. Изучаем Python. :5-е изд. Том 1. - СПб.: Символ-Плюс, 2011
11. Ривз БайронМалоун ТомасО’Дрисколл Тони. Виртуальный мир как кузница руководящих кадров [Электронный ресурс].- Электрон.дан. - URL: https://hbr-russia.ru/liderstvo/psikhologiya/a9572. (дата обращения: 03.04.2021).

Приложение А

Список заданных функций

- sin(x) = синус
- cos(x) = косинус
- max(a,b) = наибольшее значение
- min(a,b) = наименьшее значение
- eq(id,value) = 1, если value не превышает граничные значения соответствующей переменной, иначе возвращает 0.
- show(n,t1,t2,..,tn) = 1
- ans(id) = порядковый номер ответа, или -1, если действие еще не было использовано, или оно не предполагает выбор из нескольких вариантов
- exp(x,y) = x^y
- not(x) = 1, если x==0, иначе 0

Приложение В

Пример задания сценария

Для примера рассмотрим частичную реализацию игры с географического факультета. В качестве переменных должно быть число заболевших и монеты. Обе переменные имеют ограничение снизу в 0 и не имеют ограничения сверху.

Дополнительно введем невидимую переменную «победа» в качестве индикатора, что игра завершилась победой, и невидимую переменную «начало анализа» смысл которой будет объяснен позже.

Переменные					
Название	**Начальное значение**	**Минимальное значение**	**Максимальное значение**	**Флаг видимости**	**Id**
Номер раунда	0	0	None	Видно	Ptime
Число заболевших	5	0	None	Видно	P0
Монеты	500	0	None	Видно	P1
Победа	0	0	1	Не видно	P2
Начало анализа	-1	None	None	Не видно	P3

Создаём и заполняем вкладки.

Вкладки	
Название	**Id**
Карты местности с эпид. обстановкой	T0
Физические карты местности	T1
Истории болезни	T2
Обитающие насекомые	T3
Анализ	T4

Создаём роли.

Роли	
Название	**Id**
Хост	R0
Лидер	R1
Представитель географии	R2
Представитель медицины	R3

Игра заканчивается поражением, если игроки потратили все деньги, при этом так и не купив лекарство, или число заболевших превысило установленный лимит. Запишем это в условие на поражение p1=0 | p0>300

Игра заканчивается победой, если игроки купили лекарство от правильного заболевания. Введём действие «купить лекарство» для роли «Представитель медицины». Лекарство стоит 200 монет. У лекарства есть несколько вариантов ответа «грипп», «ковид», «воспаление лёгких». В случае, если игроком был выбран правильный ответ, игра должна завершиться победой.

<table>
<tr><td colspan="2">Действие «купить лекарство»</td></tr>
<tr><td>Вариант ответа</td><td>Id</td></tr>
<tr><td>Грипп</td><td>An0</td></tr>
<tr><td>Ковид</td><td>An1</td></tr>
<tr><td>Воспаление лёгких</td><td>An2</td></tr>
<tr><td colspan="2">Условие на действие</td></tr>
<tr><td colspan="2">p1>=200 //проверяем, что у игрока есть 200 монет</td></tr>
<tr><td colspan="2">Реакция</td></tr>
<tr><td>Условие на подреакцию</td><td>Подреакция</td></tr>
<tr><td>//Пустое условие выполняется всегда</td><td>eq(p1,p1-200) //отнимаем 200 монет за использование действия</td></tr>
<tr><td>An=2 // an – это функция, которая возвращает номер ответа, который выбрал игрок. Она равносильна ans(id), но для ans нужно знать id действия, а оно появится, только после создания самого действия, поэтому был введён такой обходной путь</td><td>eq(p2,1) //если выбран правильный ответ, то присваиваем переменной значение 1.</td></tr>
</table>

Теперь нужно записать условие на победу p2=1. Условие на победу всегда проверяется до условия на поражение, поэтому если игрок последним действием потратил все деньги, но купил нужное лекарство, то он победит.

Добавим действие «купить историю болезни» для роли «Представитель медицины». Это действие должно показывать игроку одну историю болезни и стоить 50 монет.

<table>
<tr><td colspan="2">Действие «купить историю болезни»</td></tr>
<tr><td>Вариант ответа</td><td></td></tr>
<tr><td>Пусто</td><td>Пусто</td></tr>
<tr><td colspan="2">Условие на действие</td></tr>
<tr><td colspan="2">p1>=50 & not(t2) (проверяем, что у игрока есть 50 монет, и что он ещё не видел все истории болезни)</td></tr>
<tr><td colspan="2">Реакция</td></tr>
<tr><td>Условие на подреакцию</td><td>Подреакция</td></tr>
<tr><td></td><td>eq(p1,p1-50) & show(1,t2) //отнимаем 50 монет и показываем одну историю болезни</td></tr>
</table>

Добавим действие «купить физические карты» для роли «представитель географии». Действие даёт доступ к физическим картам и стоит 100 монет.

Действие «купить физические карты»	
Вариант ответа	
Пусто	Пусто
Условие на действие	
p1>=100 & not(t1) (проверяем, что у игрока есть 100 монет, и что он ещё не видел карты)	
Реакция	
Условие на подреакцию	**Подреакция**
	eq(p1,p1-100) & show(0,t1) //отнимаем 100 монет и показываем все карты

Введём функцию для переменной число заболевших.

Функция для переменной «число заболевших»	
Условие на подфункцию	**Подфункция**
	Eq(p0,p0*2) //С каждым ходом количество больных увеличивается вдвое

Введём действие «заказать анализ» для роли «представитель медицины». Это действие стоит 300 монет, и оно должно через 3 хода после покупки сообщать игроку пришедшие анализы.

Чтобы реализовать подобную логику, введём переменную «начало анализа», куда будем записывать раунд, в котором начался анализ. Теперь в начале каждого хода отслеживаем сколько прошло времени с начала анализа, и ждём, когда пройдёт N ходов. Действие можно применять только один раз, поэтому запишем это в максимальное число использований

Действие «купить анализ»	
Вариант ответа	
Пусто	Пусто
Условие на действие	
p1>=300 (проверяем, что у игрока есть 300 монет)	
Реакция	
Условие на подреакцию	**Подреакция**
	eq(p1,p1-300) & eq(p3,ptime) //отнимаем 300 монет и обновляем значение переменной «начало раунда»

Осталось ввести функцию для переменной «начало раунда»

Функция для переменной «начало раунда»	
Условие на подфункцию	**Подфункция**

not(p3=-1) // использование переноса
// строки равносильно
//соединению через &
 	
(ptime-p3)=3 //прошло 3 раунда
	eq(p3,-1) //возвращаем исходное состояние
	
	show(0,t4) //показываем анализ

Приложение С

Жизненный цикл Activity