

УТВЕРЖДАЮ

Руководитель практики от Университета

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*(должность, ФИО)*

МП \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*(Подпись)*

Дата « \_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 202\_ год

**ОТЧЕТ**

**ПО УЧЕБНОЙ ПРАКТИКЕ**

ПМ.05 Проектирование и разработка информационных систем

|  |  |
| --- | --- |
| Караваев Александр Сергеевич | |
| *(Ф.И.О. обучающегося)* | |
| 09.02.07 Информационные системы и программирование | |
| *(специальность)* | |
|  | |
| Учебная группа | ИСПк-202-52-00 |
|  |  |
| Место прохождения практики | ФГБОУ ВО «Вятский государственный университет», |
| Колледж ВятГУ | *(наименование организации, структурного подразделения организации)* |
|  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Итоговая оценка: |  | | | | |
| Руководитель  практики от университета |  |  |  |  |  |
|  | *(дата)* |  | *(подпись)* |  | *(Ф.И.О.)* |

Киров, 2024

**ОТЗЫВ РУКОВОДИТЕЛЯ ПРАКТИКИ**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Ф.И.О. обучающегося | | Караваев Александр Сергеевич | | | |
| Специальность | | 09.02.07 Информационные системы и программирование | | | |
| Учебная группа | | ИСПк-202-52-00 | | | |
| Вид практики | | учебная практика | | | |
| Сроки прохождения практики с | | | 15.01.2024 | по | 19.06.2024 |
| Место прохождения практики | | | ФГБОУ ВО «Вятский государственный университет», Колледж ВятГУ | | |
|  | (наименование организации, структурного подразделения организации) | | | | |

ВИДЫ И КАЧЕСТВО ВЫПОЛНЕНИЯ РАБОТ

|  |  |  |  |
| --- | --- | --- | --- |
| Вид работ | Критерий выполнения работ | | |
| Выполнены полностью самостоятельно | Выполнены с незначительной помощью наставника | Выполнены с помощью наставника |
| Постановка задачи | V |  |  |
| Настройка рабочего окружения | V |  |  |
| Анализ предметной области, постановка расширенного технического задания | V |  |  |
| Проектирование | V |  |  |
| Реализация программного кода | V |  |  |
| Тестирование и отладка полученного кода | V |  |  |
| Подготовка отчетной документации, включая детальное описание разработанных и использованных алгоритмов | V |  |  |
| Подготовка к сдаче и сдача промежуточной аттестации | V |  |  |

Обучающийся ознакомлен с правилами внутреннего трудового распорядка, охраны труда, техники безопасности, противопожарной безопасности, санитарно-эпидемиологическими правилами и гигиеническими нормативами, а также прошел вводный инструктаж и инструктаж на рабочем месте.

Во время прохождения учебной практики обучающимся освоены следующие профессиональные и общие компетенции:

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование компетенции | Показатели оценки | Оценка | |
| Освоена | Не освоена |
| ПК 5.1. Собирать исходные данные для разработки проектной документации на информационную систему. | Способен выполнять анализ предметной области, выявляя существенные элементы, оказывающие влияние на проектируемую систему | V |  |
| ПК 5.2. Разрабатывать проектную документацию на разработку информационной системы в соответствии с требованиями заказчика. | Способен выполнять декомпозицию сущностей с целью получения наиболее полной картины о целесообразной структуре разработки | V |  |
| ПК 5.3. Разрабатывать подсистемы безопасной информационной системы в соответствии с техническим заданием. | Способен прогнозировать потенциально некорректные действия пользователя и предусматривать соответствующие реакции со стороны системы | V |  |
| ПК 5.4. Производить разработку модулей информационной системы в соответствии с техническим заданием. | Способен создавать программный код, отвечающий предъявляемым требованиям | V |  |
| ПК 5.5. Осуществлять тестирование информационной системы на этапе опытной эксплуатации с фиксацией выявленных ошибок кодирования в разрабатываемых модулях информационной системы. | Способен выполнять оценку корректности функционирования системы | V |  |
| ПК 5.6. Разрабатывать техническую документацию на эксплуатацию информационной системы. | Способен разрабатывать проектную, техническую и пользовательскую документации | V |  |
| ПК 5.7. Производить оценку информационной системы для выявления возможности ее модернизации. | Способен оценивать перспективы дальнейшего развития программной системы | V |  |
| ПК 5.8. Осуществлять интеграцию информационной системы с другими программными продуктами. | Способен повторно использовать готовые шаблонные решения при разработке программного продукта | V |  |
| ОК 01. Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам. | Способен анализировать специализированную информацию и находить оптимальные пути решения задач профессиональной деятельности | V |  |
| ОК 02. Использовать современные средства поиска, анализа и интерпретации информации, и информационные технологии для выполнения задач профессиональной деятельности. | Способен анализировать специализированную информацию и находить оптимальные пути решения задач профессиональной деятельности | V |  |
| ОК 03. Планировать и реализовывать собственное профессиональное и личностное развитие, предпринимательскую деятельность в профессиональной сфере, использовать знания по финансовой грамотности в различных жизненных ситуациях. | Способен находить пути улучшения имеющихся решений, позволяющих повысить их общий качественный уровень | V |  |
| ОК 04. Эффективно взаимодействовать и работать в коллективе и команде. | Способен конструктивно обмениваться информацией с коллегами, грамотно формулировать запросы в целях получения разъясняющей информации | V |  |
| ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста. | Способен конструктивно обмениваться информацией с коллегами, грамотно формулировать запросы в целях получения разъясняющей информации | V |  |
| ОК 06. Проявлять гражданско-патриотическую позицию, демонстрировать осознанное поведение на основе традиционных общечеловеческих ценностей, в том числе с учетом гармонизации межнациональных и межрелигиозных отношений, применять стандарты антикоррупционного поведения. | Способен конструктивно обмениваться информацией с коллегами, грамотно формулировать запросы в целях получения разъясняющей информации | V |  |
| ОК 07. Содействовать сохранению окружающей среды, ресурсосбережению, применять знания об изменении климата, принципы бережливого производства, эффективно действовать в чрезвычайных ситуациях. | Способен эффективно использовать компьютерное время, а также материальные ресурсы, необходимые для решения поставленных задач | V |  |
| ОК 08. Использовать средства физической культуры для сохранения и укрепления здоровья в процессе профессиональной деятельности и поддержания необходимого уровня физической подготовленности. | Способен соблюдать требования внутреннего трудового распорядка организации, охраны труда и техники безопасности в целях сохранения собственного здоровья | V |  |
| ОК 09. Пользоваться профессиональной документацией на государственном и иностранном языках. | Способен использовать в своей работе специализированную документацию | V |  |

**Краткая характеристика работы обучающегося**

|  |
| --- |
| Программа практики выполнена в полном объеме. Все виды работ выполнялись в срок, |
| без существенных замечаний. В достаточной степени была проявлена самостоятельность |
| и умение грамотно пользоваться источниками информации, находящимися в свободном |
| доступе. |

|  |  |  |
| --- | --- | --- |
|  |  | Руководитель практики  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Подпись ФИО  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (должность)  Дата «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 202\_\_ год |

**СОДЕРЖАНИЕ**

[1. СВЕДЕНИЯ О РАБОТЕ, ВЫПОЛНЕННОЙ В ПЕРИОД ПРОХОЖДЕНИЯ УЧЕБНОЙ ПРАКТИКИ 4](#_Toc1921211499)

[2. НАСТРОЙКА РАБОЧЕГО ОКРУЖЕНИЯ 5](#_Toc1098142238)

[3. ФОРМУЛИРОВКА ИНДИВИДУАЛЬНОГО ЗАДАНИЯ 7](#_Toc938218772)

[4. ОПИСАНИЕ ВЫПОЛНЕНИЯ ИНДИВИДУАЛЬНОГО ЗАДАНИЯ 8](#_Toc909690683)

[4.1 Анализ предметной области и обзор аналогов 8](#_Toc2141738082)

[4.2 Разработка структуры приложения и алгоритмов функционирования 13](#_Toc477362636)

[4.3 Реализация программы 29](#_Toc1883139767)

[4.4 Тестирование программного обеспечения 36](#_Toc19945828)

[ЗАКЛЮЧЕНИЕ 44](#_Toc271616305)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 45](#_Toc1950692267)

[ПРИЛОЖЕНИЕ А 46](#_Toc520108844)

[ПРИЛОЖЕНИЕ Б 68](#_Toc1458846392)

[ПРИЛОЖЕНИЕ В 69](#_Toc1718317841)

[ПРИЛОЖЕНИЕ Г 70](#_Toc885345677)

**ВВЕДЕНИЕ**

Учебная практика ПМ.05 проходила на базе Колледжа ФГБОУ ВО «Вятский государственный университет» в период с 15.01.2024 г. по 19.06.2024 г. по субботам.

Цель практики: сформировать у обучающихся представление о процессе разработки прикладного программного обеспечения.

Задачи практики:

– дать представление о различных методах проектирования прикладного программного обеспечения;

– сформировать навыки написания документации, требуемой при разработке прикладного программного обеспечения;

– дать представление о различных инструментах, используемых при разработке прикладного программного обеспечения.

# **1. СВЕДЕНИЯ О РАБОТЕ, ВЫПОЛНЕННОЙ В ПЕРИОД ПРОХОЖДЕНИЯ УЧЕБНОЙ ПРАКТИКИ**

В период с 15.01.2024 по 19.06.2024 при прохождении учебной практики ПМ.05 на базе Колледжа ФГОБУ ВО «Вятский государственный университет» был выполнен следующий перечень работ, представленный в таблице 1.

Таблица 1 – Сведения о работе, выполненной в период практики

| Дата | Краткое содержание выполненных работ |
| --- | --- |
| 15.01.2024 | Ознакомление с правилами внутреннего трудового распорядка, охраны труда, техники безопасности, противопожарной безопасности, санитарно-эпидемиологическими правилами и гигиеническими нормативами, а также вводный инструктаж и инструктаж на рабочем месте |
| 16.01.2024-29.01.2024 | Постановка задачи |
| 30.01.2024-12.02.2024 | Настройка рабочего окружения |
| 13.02.2024-26.02.2024 | Анализ предметной области, постановка расширенного технического задания |
| 27.02.2024-01.04.2024 | Проектирование |
| 02.04.2024-31.05.2024 | Реализация программного кода |
| 01.06.2024-06.06.2024 | Тестирование и отладка полученного кода |
| 07.06.2024-13.06.2024 | Подготовка отчетной документации, включая детальное описание разработанных и использованных алгоритмов |
| 17.06.2024-19.06.2024 | Подготовка к сдаче и сдача промежуточной аттестации |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(дата) (подпись)

# **2. НАСТРОЙКА РАБОЧЕГО ОКРУЖЕНИЯ**

В ходе прохождения учебной практики использовался язык программирования Python, среда разработки (IDE) PyCharm

Python — высокоуровневый язык программирования общего назначения с динамической строгой типизацией и автоматическим управлением памятью, ориентированный на повышение производительности разработчика, читаемости кода и его качества, а также на обеспечение переносимости написанных на нём программ. Язык является полностью объектно-ориентированным в том плане, что всё является объектами. Необычной особенностью языка является выделение блоков кода отступами.

Основными достоинствами языка является простота и читаемость кода. Синтаксис ядра языка минималистичен, за счёт чего на практике редко возникает необходимость обращаться к документации. Сам же язык известен как интерпретируемый и используется в том числе для написания скриптов.

Недостатками языка являются зачастую более низкая скорость работы и более высокое потребление памяти написанных на нём программ по сравнению с аналогичным кодом, написанным на компилируемых языках, таких как C или C++.



Рисунок 1 – логотип Python.

PyCharm — это кроссплатформенная интегрированная среда разработки для языка программирования Python, которая предоставляет пользователю комплекс средств для написания кода и визуальный отладчик. Продукт доступен в двух версиях: PyCharm Community Edition — бесплатная версия и PyCharm Professional Edition — расширенная версия продукта, обладающая дополнительной функциональностью.

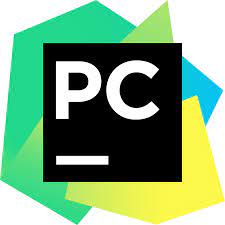


Рисунок 2 – логотип PyCharm.

2) Описание используемых библиотек. (в данный момент можно оставить пустым и заполнить по ходу разработки ПО). Обоснование использования именно этих библиотек.

Скриншоты работы в выбранной среде приведены на рисунках ….

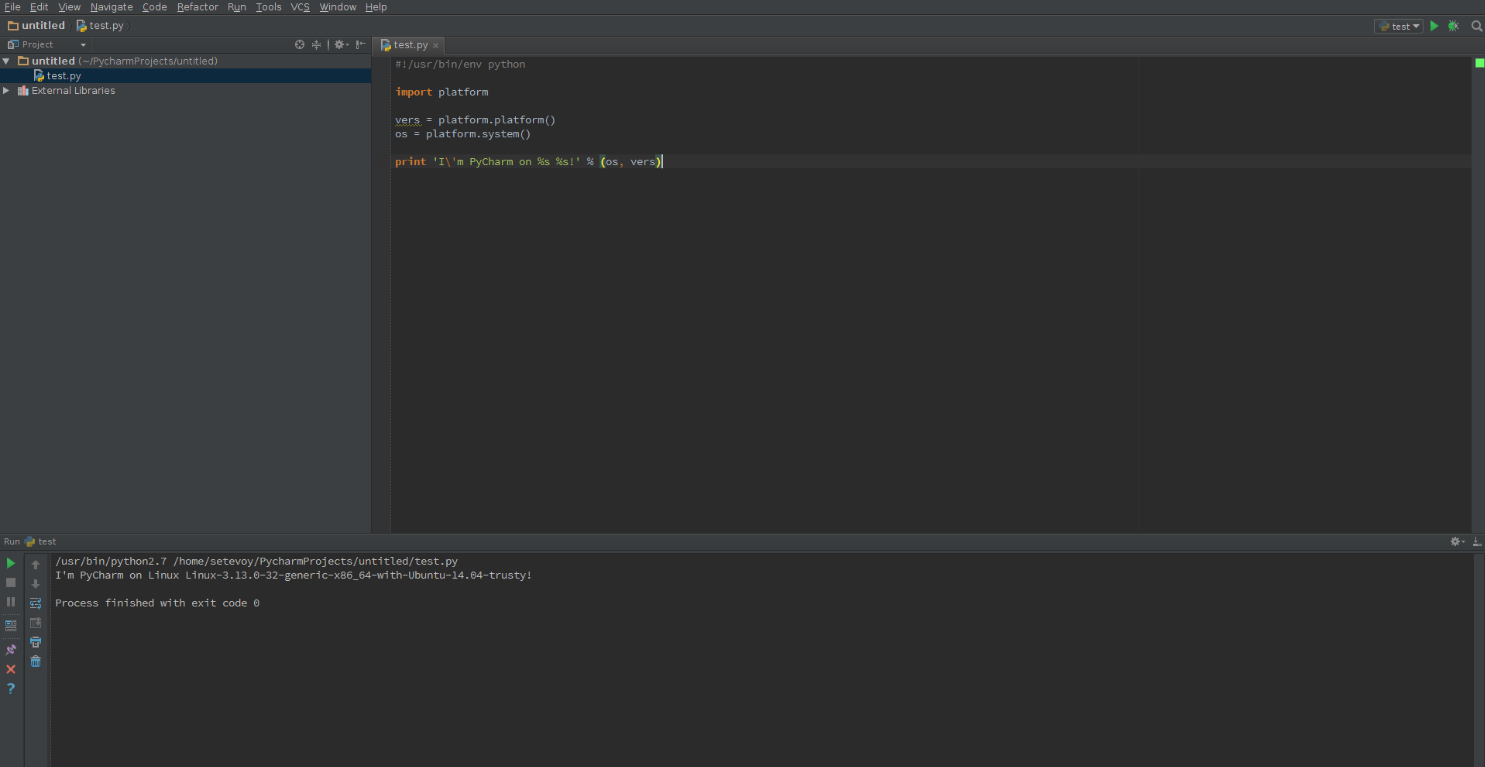


Рисунок 3 – Интерфейс PyCharm

Для системы контроля версий в соответствии с заданием использована система GitHub.

Имя пользователя: rondow 31

Адрес: <https://github.com/rondow31>

# **3. ФОРМУЛИРОВКА ИНДИВИДУАЛЬНОГО ЗАДАНИЯ**

В соответствии с индивидуальным заданием на учебную практику необходимо:

* разработать видеоигру «Поле Чудес», в соответствии с требованиями, указанными в техническом задании (см. Приложение Б);
* написать и подготовить к сдаче комплект документации, включающий в себя: данный отчет по учебной практике, техническое задание, руководство пользователя и программу и методику испытаний.

# **4. ОПИСАНИЕ ВЫПОЛНЕНИЯ ИНДИВИДУАЛЬНОГО ЗАДАНИЯ**

# **4.1 Анализ предметной области и обзор аналогов**

Компьютерная игра — компьютерная программа, служащая для организации игрового процесса (геймплея), связи с партнёрами по игре, или сама выступающая в качестве партнёра. В настоящее время, в ряде случаев, вместо термина компьютерная игра может использоваться видеоигра, то есть данные термины могут употребляться как синонимы и быть взаимозаменяемыми. Компьютерные игры могут создаваться на основе фильмов и книг, но есть и обратные случаи.

Головоломка (англ. *puzzle*) — название жанра компьютерных игр, целью которых является решение логических задач, требующих от игрока задействования логики, стратегии и интуиции. Предтечей жанра являлись настольные, графические и механические головоломки — от кроссвордов до кубика Рубика. Эти головоломки требовали от игрока логики и ловкости в решении, которые также стали играть важную роль в прототипах жанра, таких, как Q\*bert и Boulder Dash. Эталоном же жанра стала игра Тетрис, появившаяся в 1985 году и сочетавшая в себе простой и захватывающий игровой процесс.

**«**Поле чудес**»** — компьютерная игра жанра головоломки. Задачей игрока является угадывание слова. Игроки поочерёдно вращают барабан, который имеет 16 секторов, от 5 до 25 очков, а также сектора х2 и сектор х4, отсутствующий в телевизионной игре. Игрок может назвать букву или слово целиком. Есть также сектор «плюс», позволяющий открыть любую букву, и сектор «приз», при попадании на который игрок может или продолжить игру, назвав букву, или забрать «приз», закончив игру. Есть также два специальных сектора: «0» (переход хода) и «банкрот» (обнуление очков). Игра проходит в три тура: четвертьфинал, полуфинал и финал. Для каждого тура ведущий объявляет тему, к которой относится угадываемое слово, не задавая никакого вопроса.

**4.1.1 Обзор аналогов.**

Первый аналог «Поле чудес: Капитал шоу» (1993г.) — это самая первая компьютерная игра «Поле чудес» для IBM-совместимых персональных компьютеров под управлением DOS, разработанная российским программистом Вадимом Башуровым в 1993 году и получила огромную популярность по всей России. Игра является неофициальной компьютерной адаптацией известной телепередачи «Поле чудес». Сам Башуров получил за свою игру всего 500 долларов.

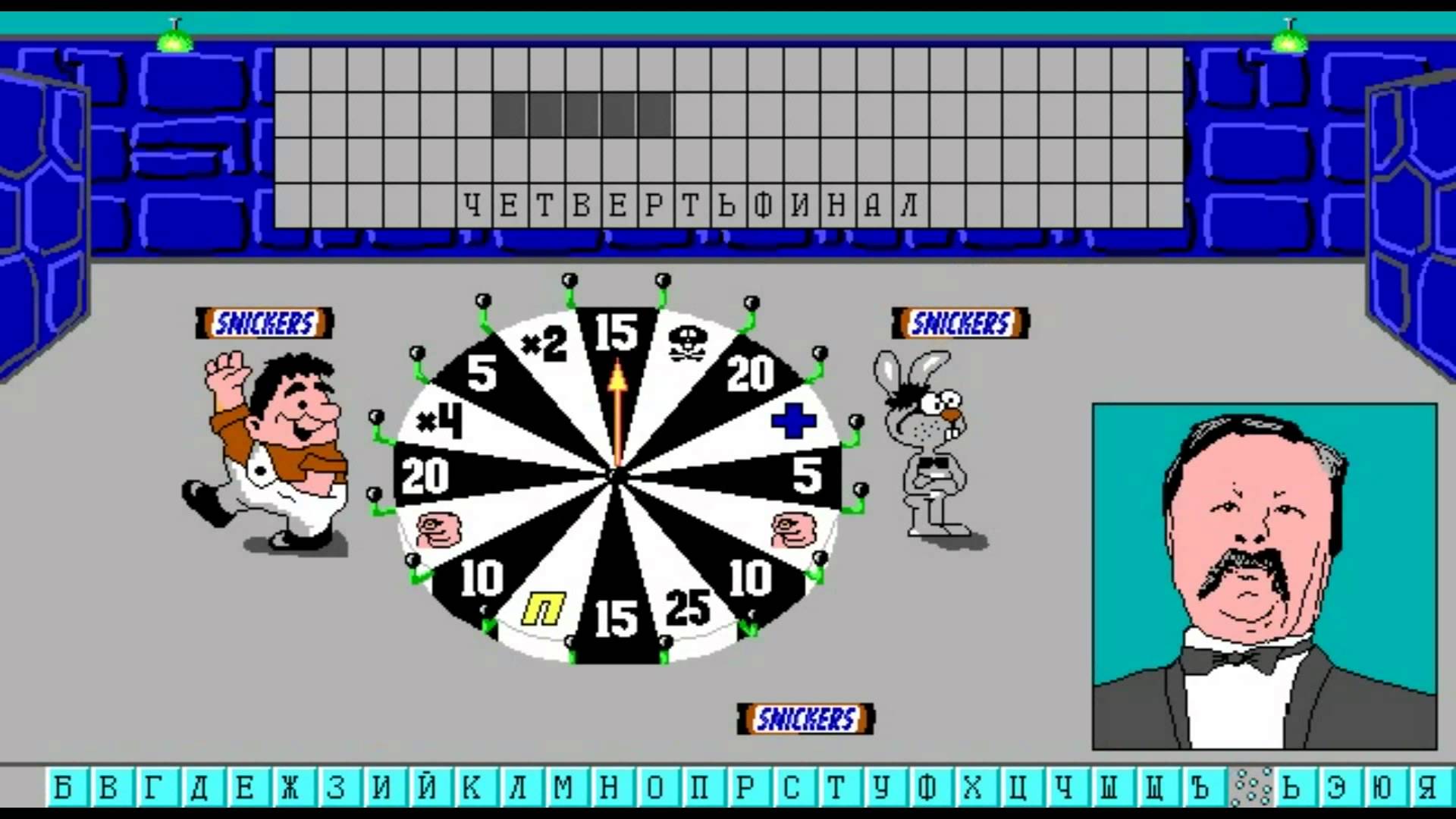


Рисунок 1.

Плюсы:

* Охват большого количества пользователей.
* Удобный и простой интерфейс

Минусы:

* Однотипный геймплей, который со временем может надоесть.

Следующий рассматриваемый аналог «Вращайте барабан» (2018г.) — симулятор нестареющей игры с озвучкой известного телеведущего. Имеет более десяти миллионов скачиваний в Google Play.



Рисунок 2.

Плюсы:

* Доступна онлайн игра.
* Вопросы распределены по возрастным категориям.
* Добавлен голос ведущего.

Минусы:

* Реклама после каждого раунда.

Ещё один аналог игра «Поле Чудес Плюс» (2018г.) — ремейк старой версии игры. Игра переписана с нуля, добавлено много новых персонажей, призов и пасхалок. Присутствуют все любимые персонажи из оригинальной DOS-версии 90-х годов. Игра разработана на собственном "движке", и имеет уникальную систему анимации, поэтому работает быстро и занимает совсем немного места. Более миллиона скачиваний в Google Play.

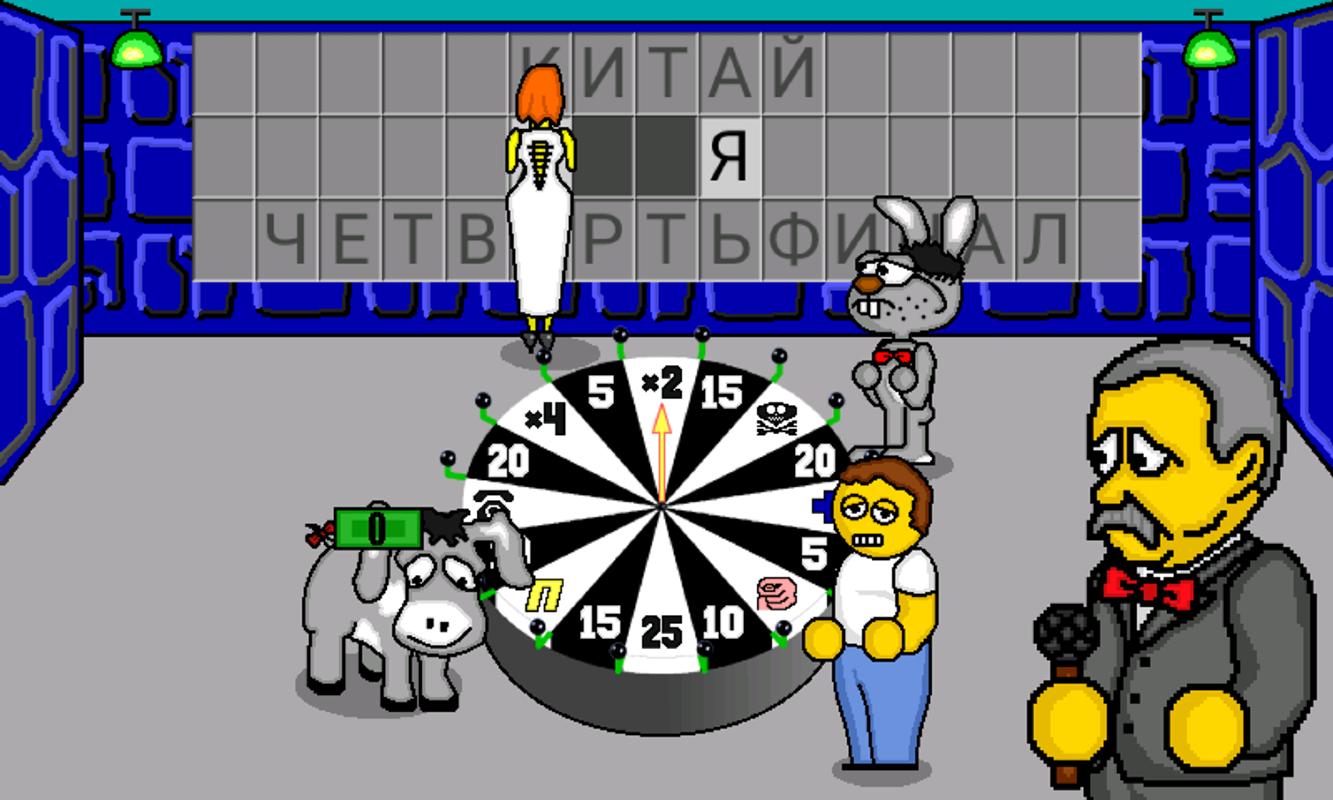


Рисунок 3.

Плюсы:

* Много различных уровней и этапов.
* Позволяет значительно улучшить словарный запас пользователя.

Минусы:

* Иногда загаданное слово не соответствует теме.
* Присутствует реклама.

Последний рассматриваемый аналог игра «Виселица» — эта игра схожа принципом игры с Полем чудес. Загадывается слово и отмечаются места для букв, например чертами. Также рисуется виселица с петлёй. Игрок предлагает букву, которая может входить в это слово. Если такая буква есть в слове, то буква появляется над соответствующими этой букве чертами — столько раз, сколько она встречается в слове. Если такой буквы нет, то к виселице добавляется круг в петле, изображающий голову. За каждый неправильный ответ добавляется одна часть туловища к виселице. Если туловище в виселице нарисовано полностью, то игрок проигрывает, считается повешенным. Если игроку удаётся угадать слово, он выигрывает. Игра имеет более пяти миллионов скачиваний в Google Play.

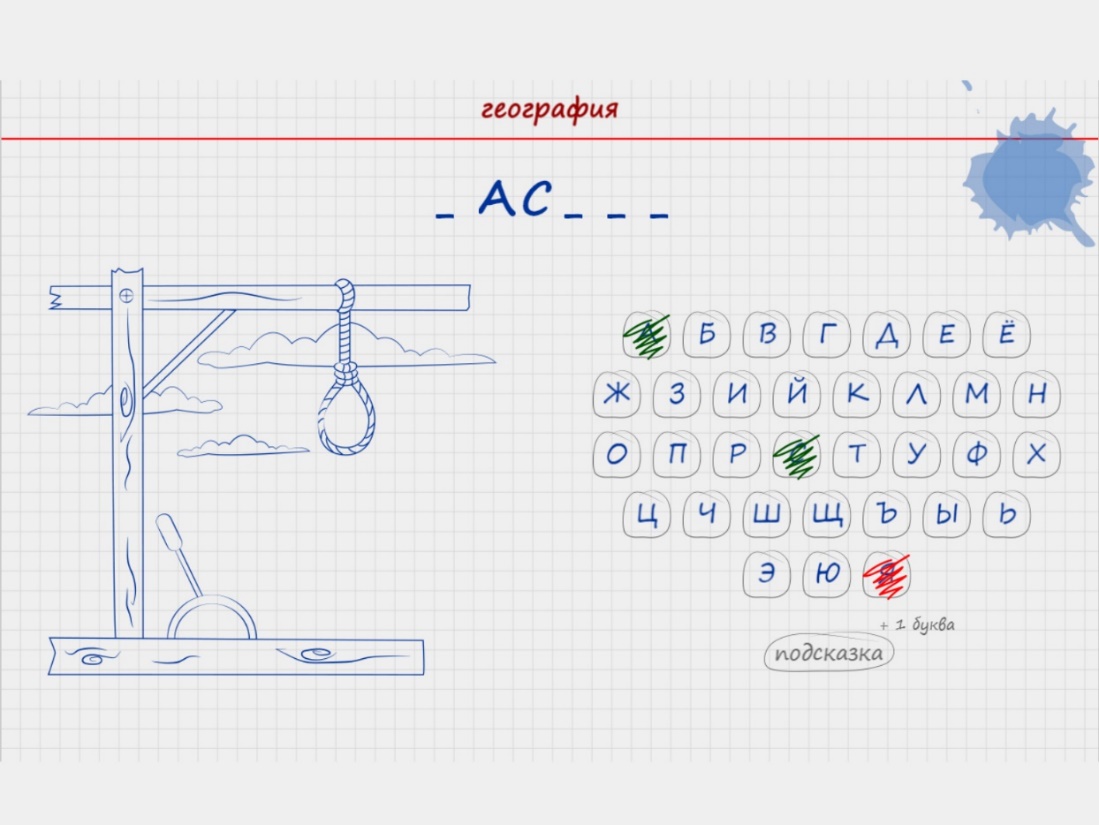


Рисунок 4.

Плюсы:

* Простота геймплея и правил.
* Есть онлайн режим.

Минусы:

* Большое количество рекламы.
* Иногда загаданное слово не соответствует теме.

Таким образом, данные игры, как и любая игра жанра «головоломка» значительно увеличивают словарный запас пользователей.

Из проведённых нами исследованиями можно отметить такие недостатки как:

* В некоторых из данных игр присутствует большое количество рекламы.
* Иногда загаданные слова в данных приложениях не совпадают с темой.

## **4.2 Разработка структуры приложения и алгоритмов функционирования**

В данном подразделе настоящего отчета по учебной практике приведены этапы разработки структуры приложения и алгоритмов функционирования.

**4.2.1 Разработка алгоритмов функционирования.**

4.2.1 Разработка алгоритмов функционирования

Алгоритм работы игры поле чудес представлен на рисунке 4.2.1.1.

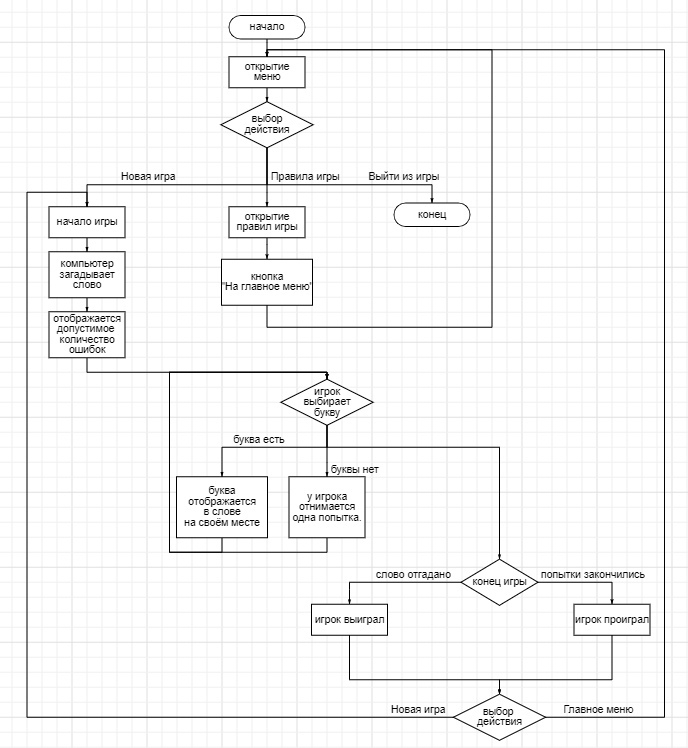


Рисунок 4.2.1.1 – Алгоритм работы игры «Поле чудес».

Алгоритм представляет собой загрузку интерфейса с последующими действиями пользователя. При нажатии на кнопку «Новая игра» будет открыто игровое окно, где игрок сможет сыграть в «Поле чудес». При нажатии на кнопку «Правила игры» будет показано окно с правилами игры.

При нажатии на кнопку «Выйти из игры» приложение закроется.

Алгоритм выбора слова представлен на рисунке 4.2.1.2.



Рисунок 4.2.1.2 – Алгоритм выбора слова

Алгоритм работы игрового поля представлен на рисунке 4.2.1.3.

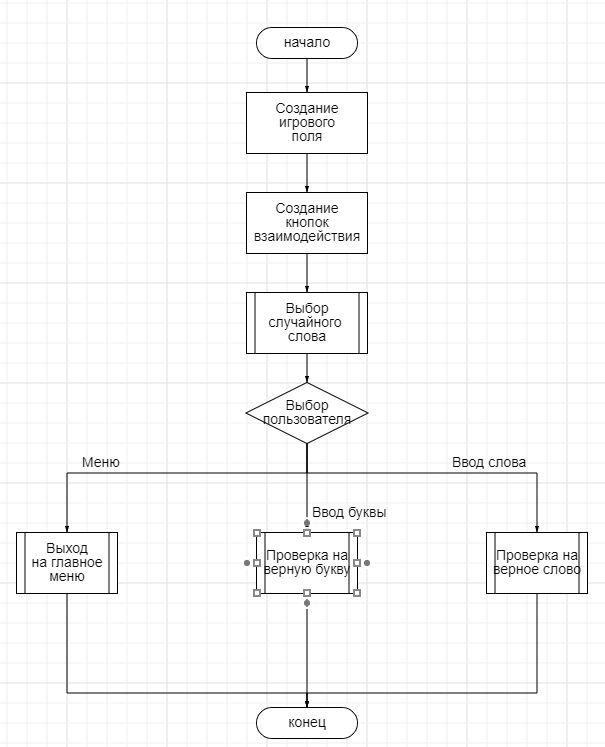


Рисунок 4.2.1.3 – Алгоритм игрового поля.

Алгоритм определения исхода игры представлен на рисунке 4.2.1.4.

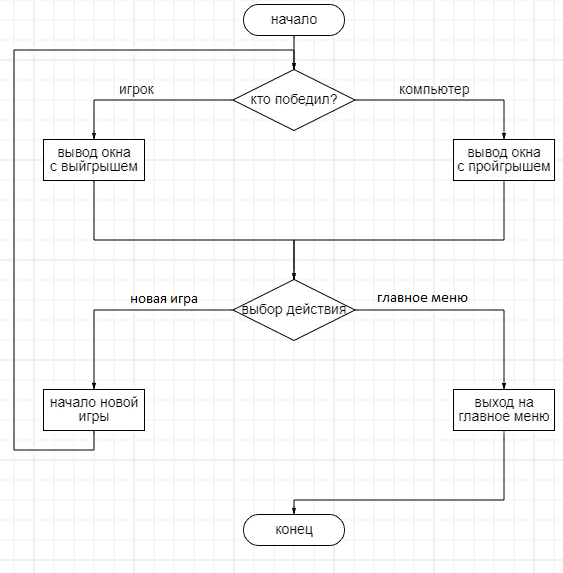


Рисунок 4.2.1.4 – Алгоритм определения исхода игры.

**4.2.2. Разработка структурных решений**

Информационная структура игры приведены на рисунке 4.2.2.1.

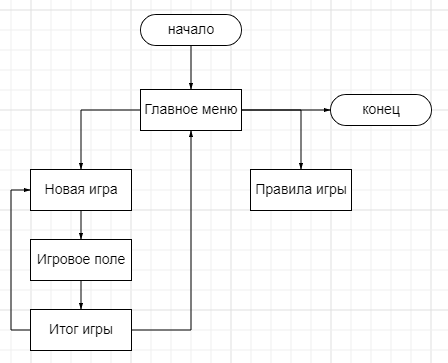


Рисунок 4.2.2.1 – Информационная структура игры.

Модульная структура игры приведена на рисунке 4.2.2.2



Рисунок 4.2.2.2 – Модульная структура игры.

**4.2.3 Проектирование интерфейса**

Интерфейс программы был разработан с помощью сервиса Figma. Результаты проектирования интерфейса приложения представлены на рисунках 4.2.3.1-4.2.3.5, а также расположены по ссылке: <https://www.figma.com/file/ke1HEPExK5moS7IyOA17F3/Figma-basics?type=design&node-id=601-12&mode=design&t=EVItQBl7kT6IcY0K-0>

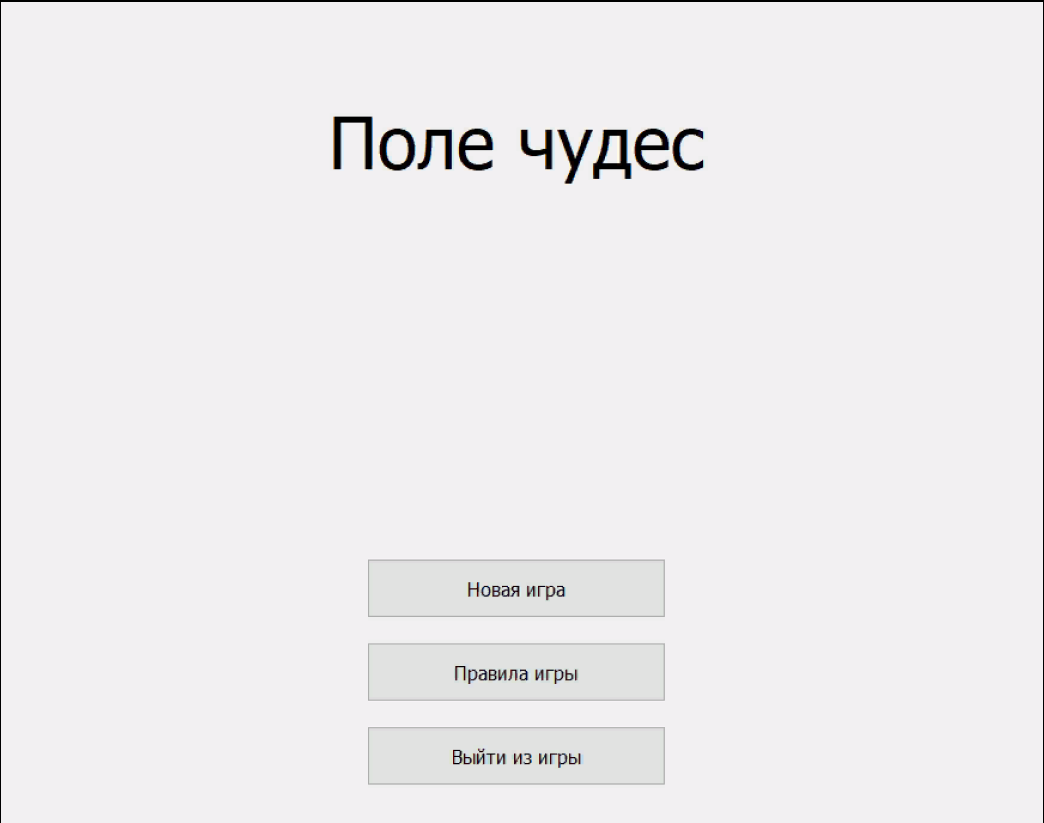


Рисунок 4.2.3.1 – Главное меню.

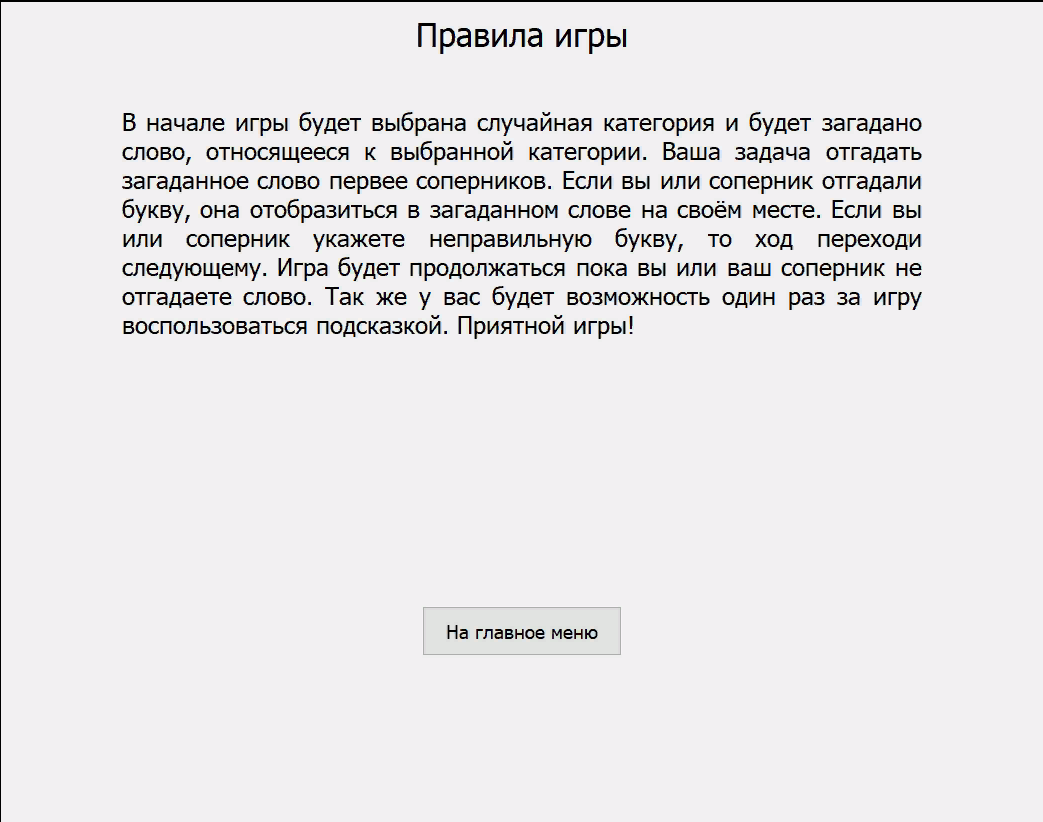


Рисунок 4.2.3.2 – Правила игры.

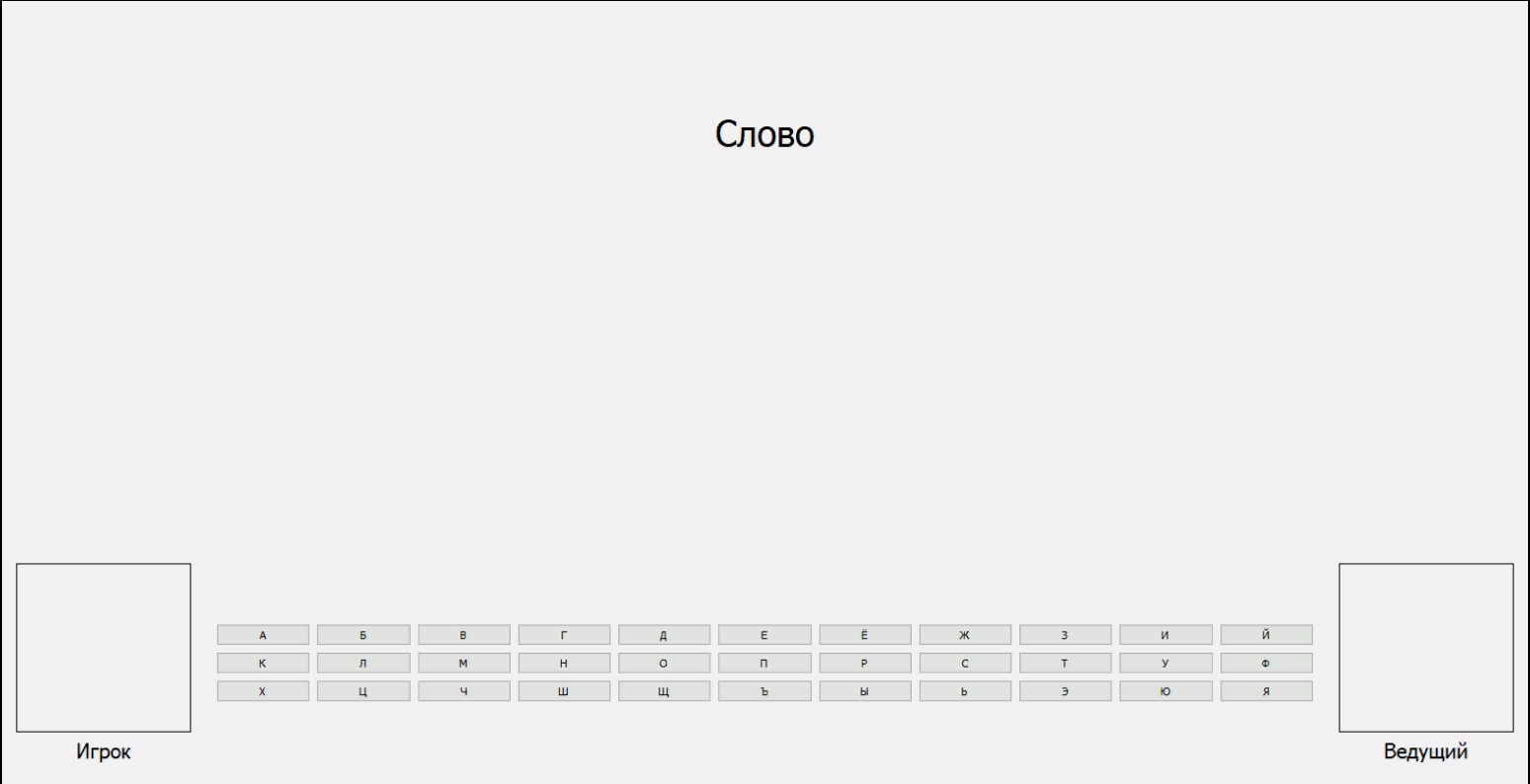


Рисунок 4.2.3.3 – Игровое окно.

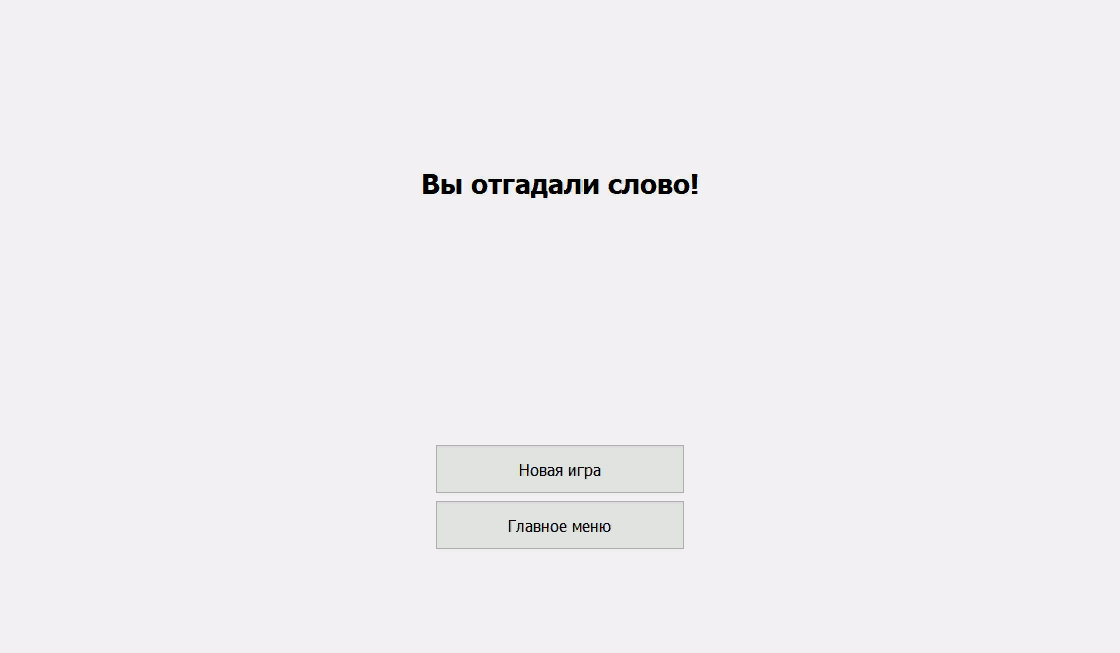


Рисунок 4.2.3.4 – Итог игры с выигрышем.

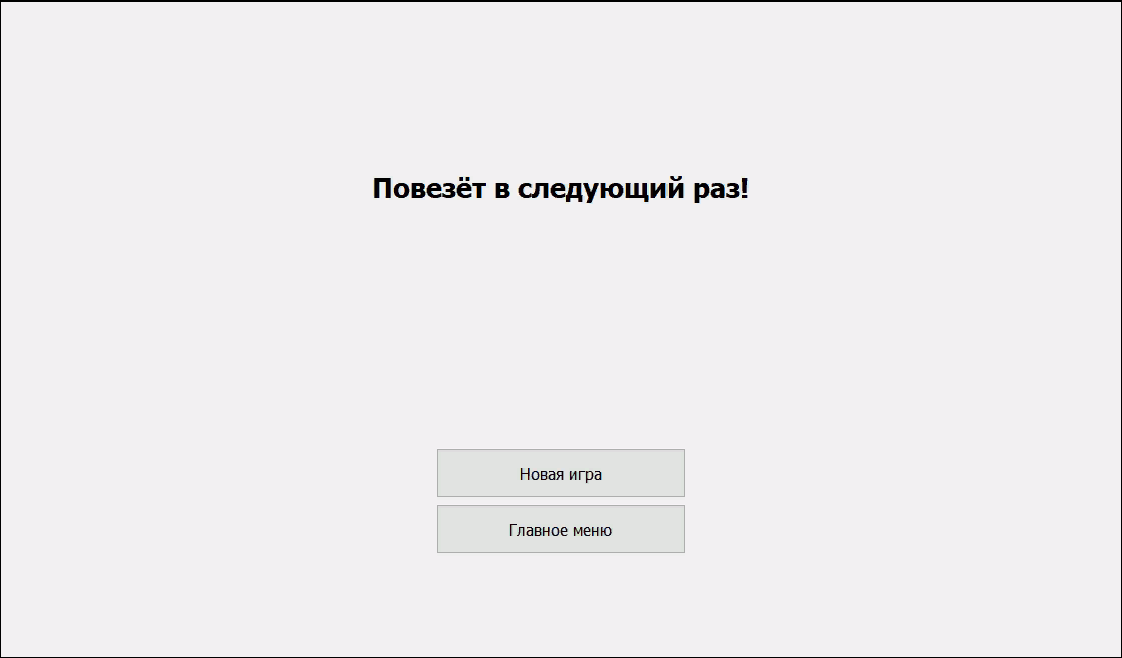


Рисунок 4.2.3.5 – Итог игры с проигрышем.

В данном разделе рассмотрены алгоритмы работы видеоигры “Поле чудес”. Рассмотрены алгоритмы работы меню, игрового поля и выводе результата игры.

Представлен прототип интерфейса, созданный в Figma, содержащий конечные интерфейсные решения.

## **4.3 Реализация программы**

В данном разделе представлены результаты разработки алгоритмов функционирования программы и интерфейса программы, а также реализации основных функций программы.

**4.3.1. Программная реализация**

Программное обеспечение разработано в среде PyCharm на языке

Python с использованием библиотек PyQt5, random, re, sys.

Программное обеспечение состоит из семи классов: Меnu\_page, GameWindow, Rules, Win, Lose, Ui\_SecondaryWindows, changer\_page.

Класс Меnu\_page создаёт окно меню.

Класс GameWindow создаёт окно игры и содержит следующие основные функции: функция создания виртуальной клавиатуры; функции обработки нажатых клавиш и кнопок; функции для нахождения и отключения кнопок, соответствующих названной букве; функции работы ботов, функция обновления загаданного слова; функции, позволяющие сканировать словарь и передавать возможные слова ботам для дальнейшего отгадывания; функции, определяющие исход игры; функция для создания сообщения паузы.

Класс Rules создаёт окно с правилами игры.

Класс Win создаёт окно с выигрышным результатом игры.

Класс Lose создаёт окно с проигрышным результатом игры.

Класс Ui\_SecondaryWindows задаёт иконку и фон игры.

Класс changer\_page служит для переключения между окнами.

Исходный код программного обеспечения представлен в приложении А.

Функция KeyPressEvent является функцией для отгадывания букв с помощью механической клавиатуры. Эта функция реализует основную логику обработки ввода пользователя в игре, обрабатывая нажатия и обновляя графический интерфейс в зависимости от состояния игры.

Исходный код функции KeyPressEvent представлен на рисунке 4.3.1.1.

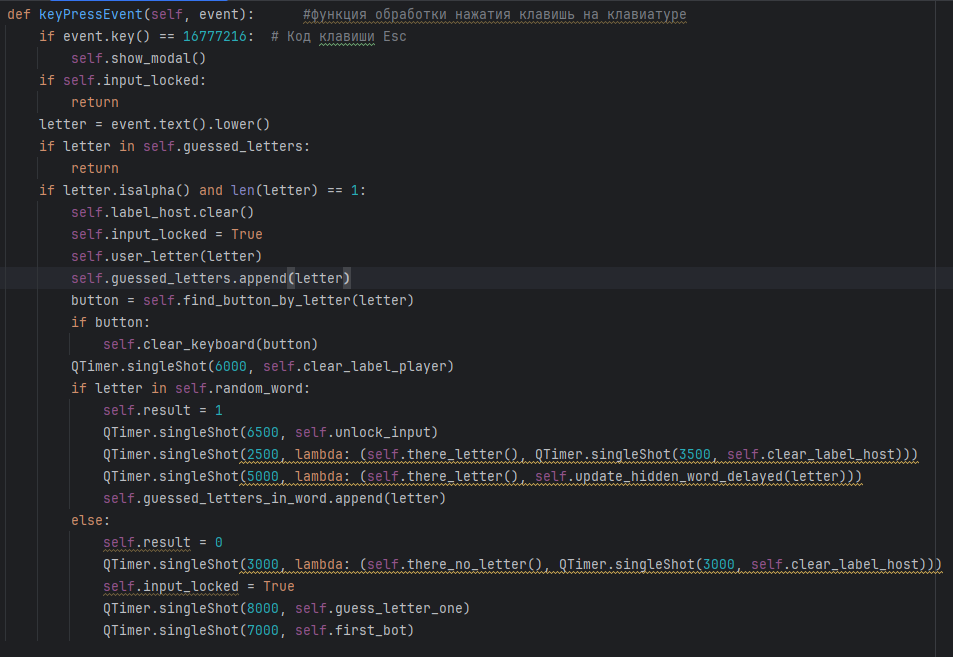


Рисунок 4.3.1.1 – Исходный код функции KeyPressEvent

Функция guess\_letter\_one является функцией для реализации простого бота для отгадывания букв на основе функции KeyPressEvent. Сначала бот получает информацию о возможных словах, сканируя словарь. В зависимости от количества не отгаданных букв и шанса, бот может гарантировано называть правильные буквы.

Исходный код функции guess\_letter\_one представлен рисунках 4.3.1.2 и 4.3.1.3.

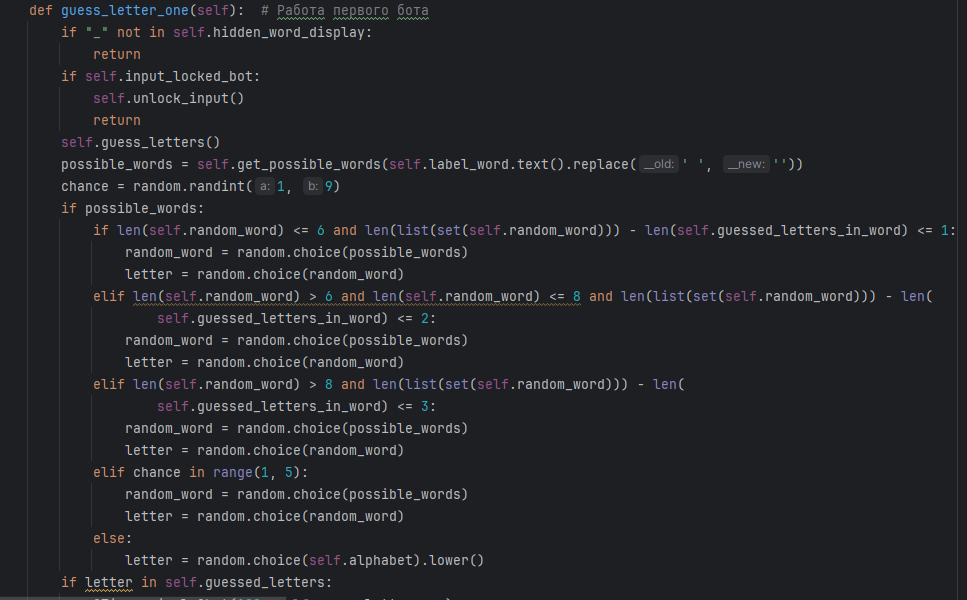


Рисунок 4.3.1.2 – Исходный код функции guess\_letter\_one

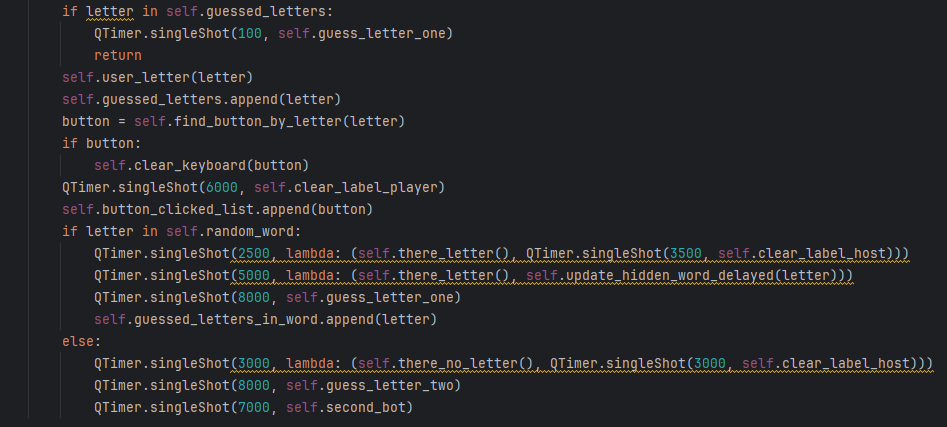


Рисунок 4.3.1.3 – Исходный код функции guess\_letter\_one

**4.3.2 Разработка пользовательского интерфейса**

Окно с главным меню игры представлено на рисунке 4.3.2.1.

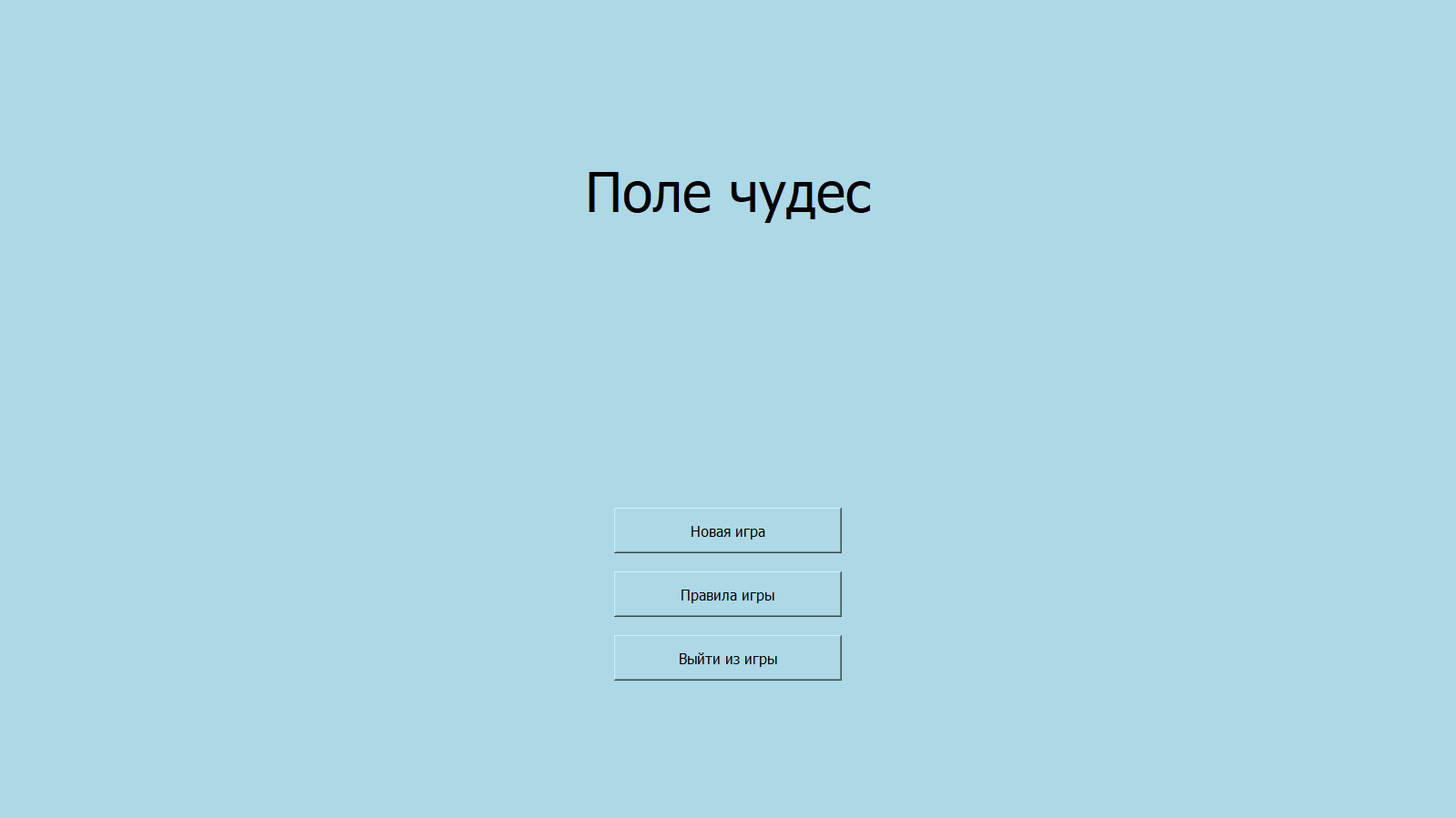


Рисунок 4.3.2.1 – Окно с главным меню игры

Кнопка «Новая игра» служит для перехода к окну игры.

Кнопка «Правила игры» служит для показа правил игры.

Кнопка «Выйти из игры» служит для выхода из программы.

Кнопки сформированы через объект QPushButton, надписи с помощью объекта QLabel.

Окно с правилами игры представлено на рисунке 4.3.2.2.

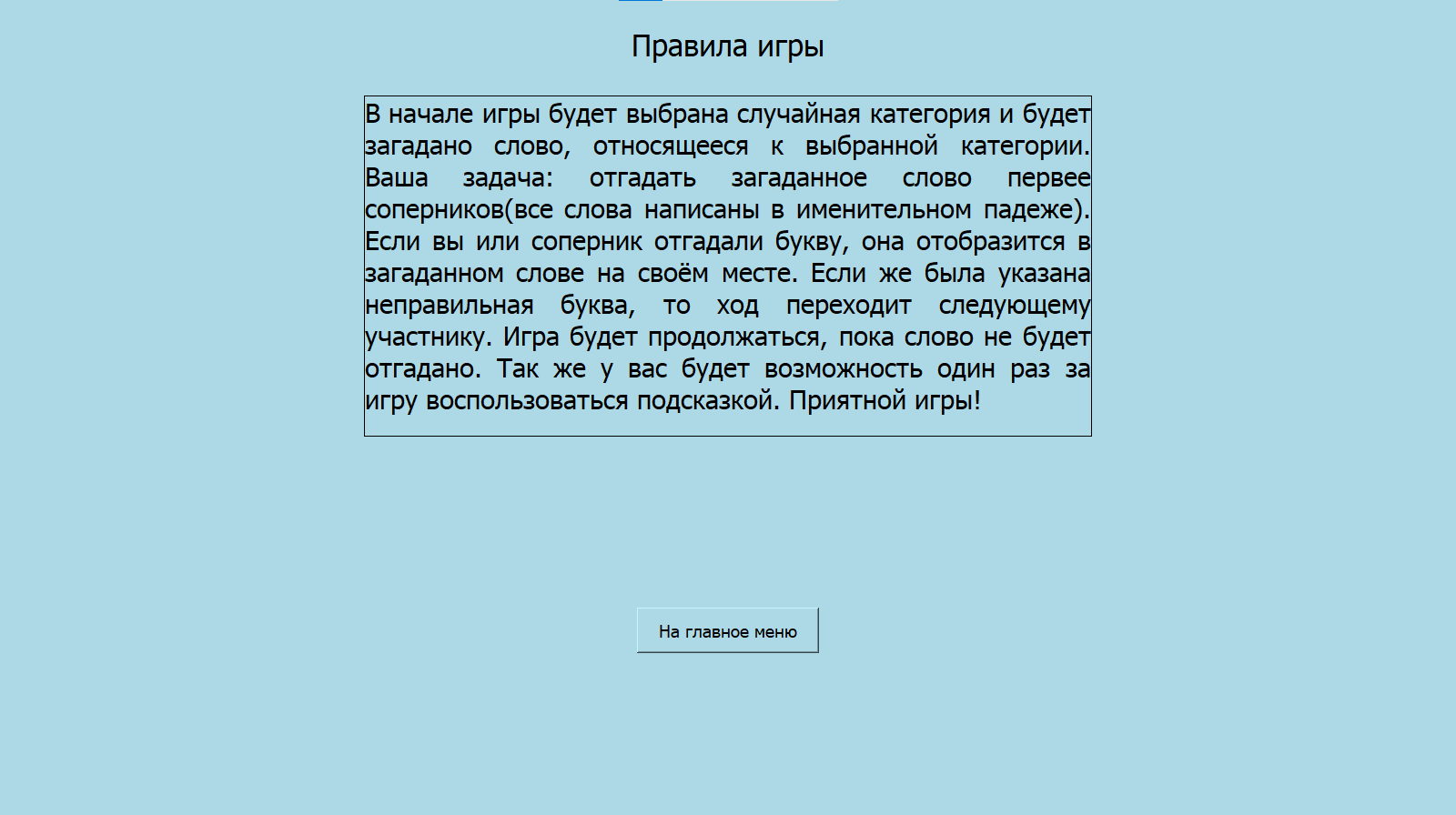


Рисунок 4.3.2.2 – Правила игры

Кнопка «На главное меню» служит для перехода в меню.

Кнопки сформированы через объект QPushButton, надписи с помощью объекта QLabel.

Окно с полем игры представлено на рисунке 4.3.2.3.

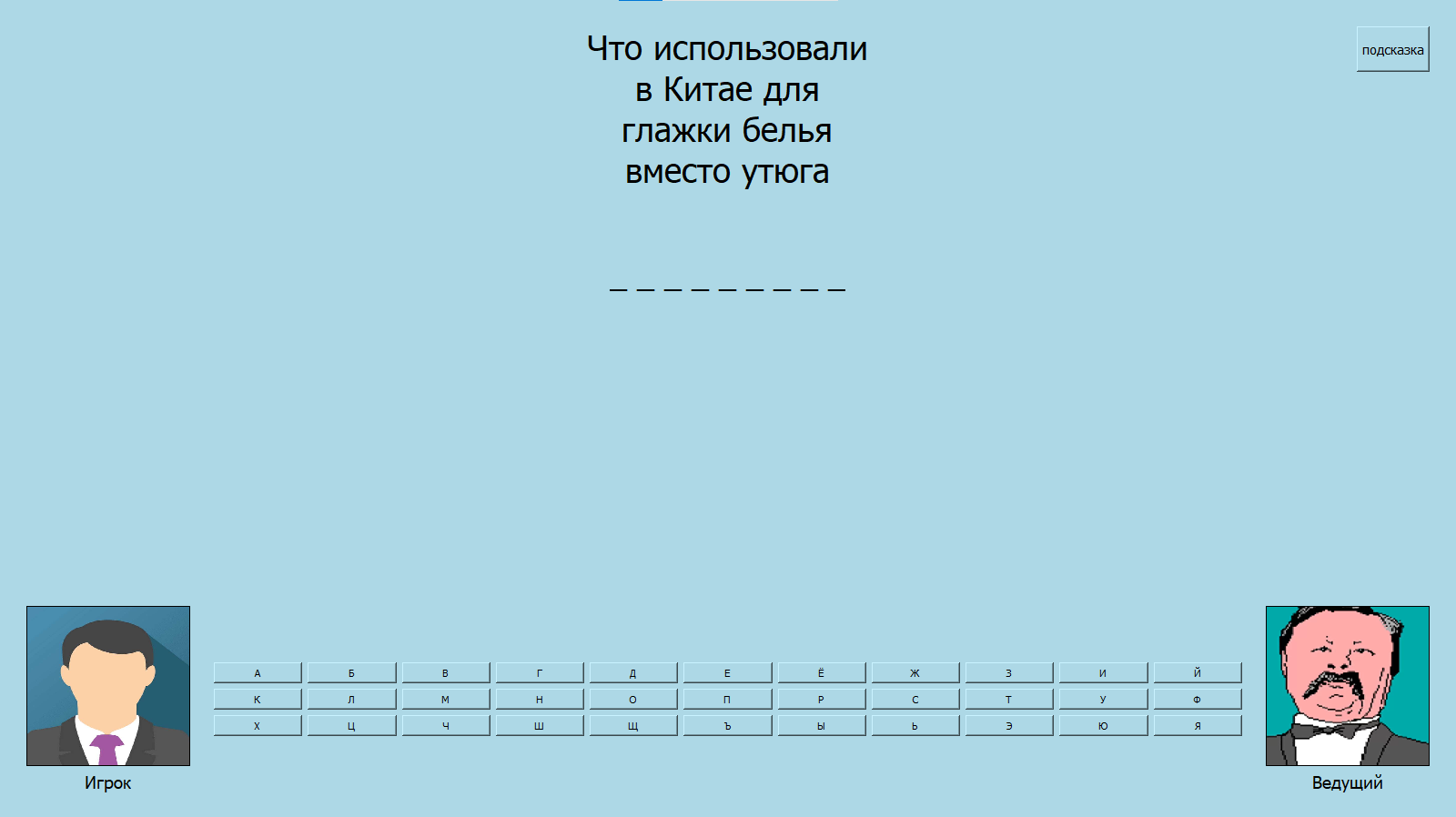


Рисунок 4.3.2.3 – Поле игры

Виртуальная клавиатура и кнопка «подсказка» представлена в виде множества QButtone и служат для отгадывания букв в слове.

Надписи и картинки сформированы с помощью объекта QLabel.

Окна с результатами игры представлены на рисунках 4.3.2.4 и 4.3.2.5.

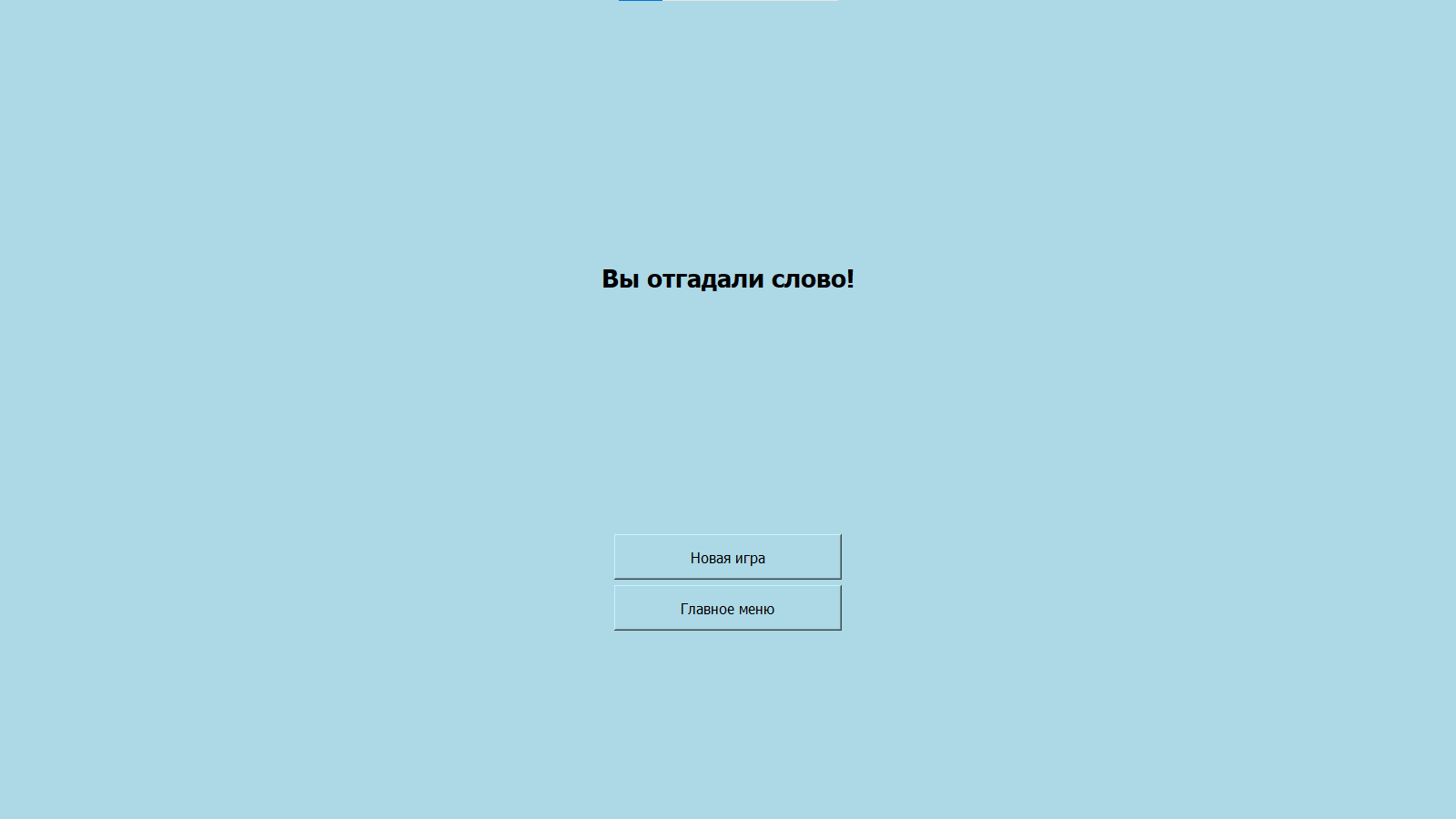


Рисунок 4.3.2.4 – Окно выигрыша

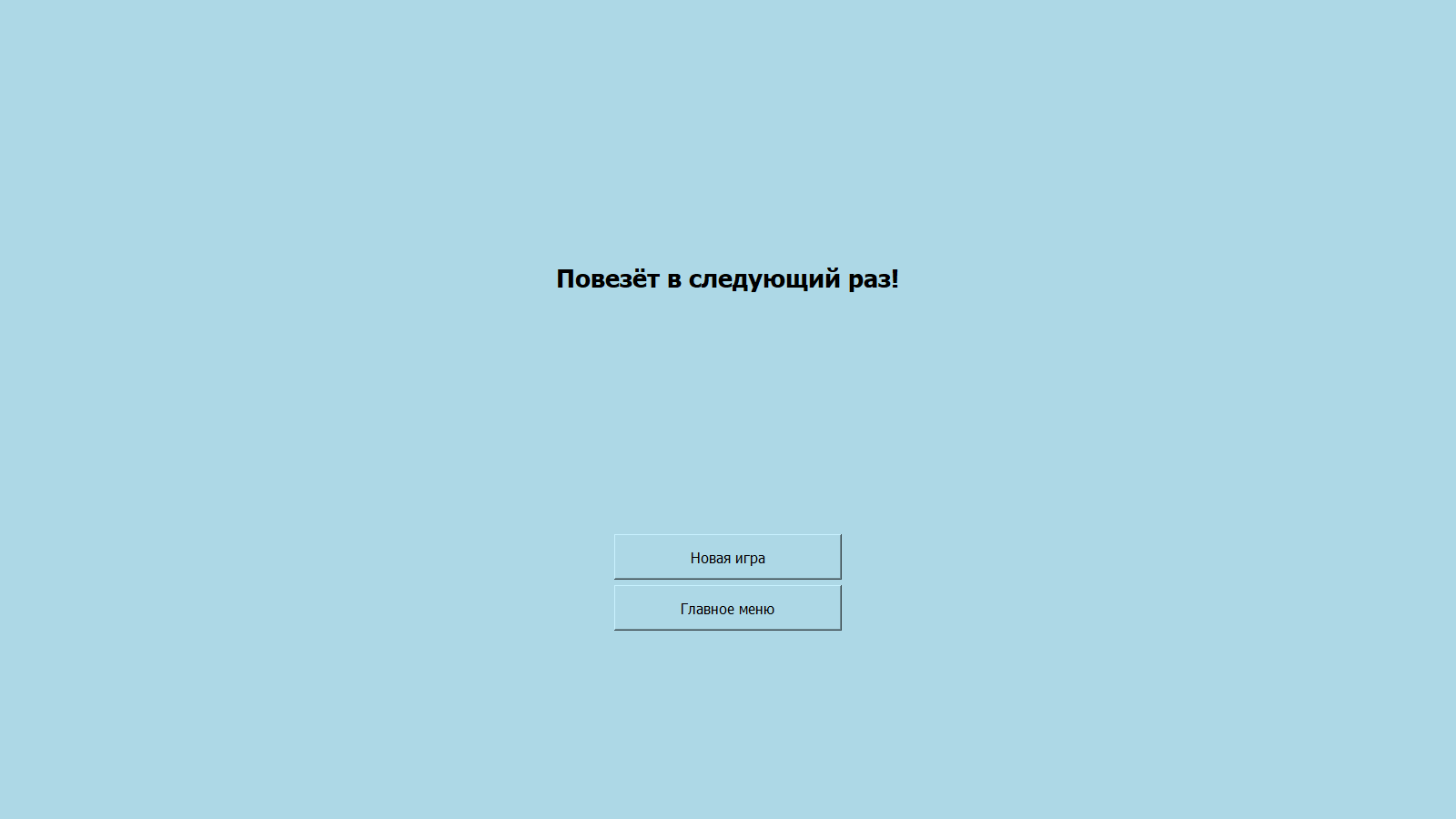


Рисунок 4.3.2.5 – Окно проигрыша

Кнопка «Новая игра» служит для начала новой игры

Кнопка «Главное меню» служит для выхода в окно меню

Кнопки сформированы через объект QPushButton, надпись с помощью объекта QLabel.

Таким образом, в данном разделе были произведены разработки алгоритмов функционирования программы. Были реализованы основные функции программы, а также рассмотрены функции для отгадывания слова.

Был разработан интерфейс программы, рассмотрены основные элементы и объекты, с помощью которых они созданы.

## **4.4 Тестирование программного обеспечения**

Тестирование приложения было проведено в соответствии с программой и методикой испытаний. Результаты испытаний внесены в таблицу 1.

Таблица 1 – Результаты испытаний

| **№** | **Шаг испытаний (проверок)** | **№ пункта**  **Методики** | **Отметка о прохождении (да/нет)** | **Примечания** |
| --- | --- | --- | --- | --- |
| **1** | Проверка состава и качества документации сопроводительной документации | **1** | **Да** |  |
| **2** | Проверка корректности запуска игры | **2** | **Да** |  |
| **3** | Проверка возможности просмотра правил | **3** | **Да** |  |
| **4** | Проверка генерации окна с игровым полем и загаданным словом | **4** | **Да** |  |
| **5** | Проверка возможности возврата на окно меню | **5** | **Да** |  |
| **6** | Проверка возможности ввода и блокировки букв с помощью клавиш клавиатуры | **6** | **Да** |  |
| **7** | Проверка возможности ввода и блокировки букв с помощью кнопок на виртуальной клавиатуре | **7** | **Да** |  |
| **8** | Проверка возможности один раз воспользоваться подсказкой с помощью кнопки «подсказка» | **8** | **Да** |  |
| **9** | Проверка на корректный ввод и блокировку букв, выполняемый двумя простыми ботами | **9** | **Да** |  |
| **10** | Проверка на правильное заполнение загаданного слова и его отображение на экране | **10** | **Да** |  |
| **11** | Проверка вывода окна с результатом игры | **11** | **Да** |  |
| **12** | Проверка на возможность начать новую игру или выхода в окно меню из окна с результатом игры | **12** | **Да** |  |

Результаты проверки сценариев:

1. В результате выполнения сценария №2 пользователь увидит окно меню (рис. 4.4.1);
2. В результате сценария №3 пользователь увидит окно с правилами игры (рис. 4.4.2);
3. Результатом сценария №4 является смена окна меню на окно игры и генерация загаданного слова (рис. 4.4.3);
4. Результатом сценария №5 является смена окна игры на окно меню (рис. 4.4.1);
5. Результатом сценария №11 является смена окна игры на окно результата игры (рис. 4.4.4 – 4.4.5)

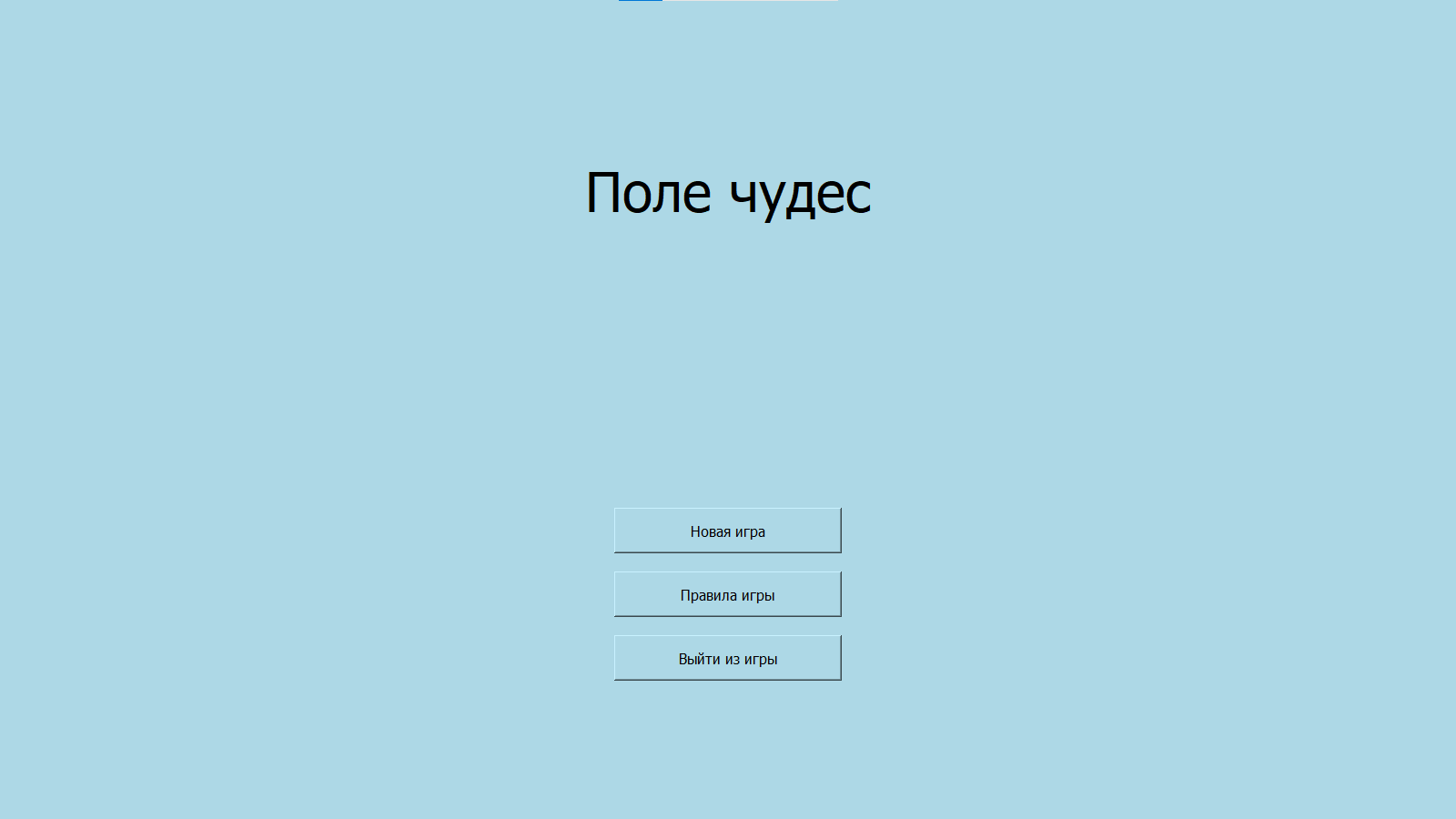


Рисунок 4.4.1

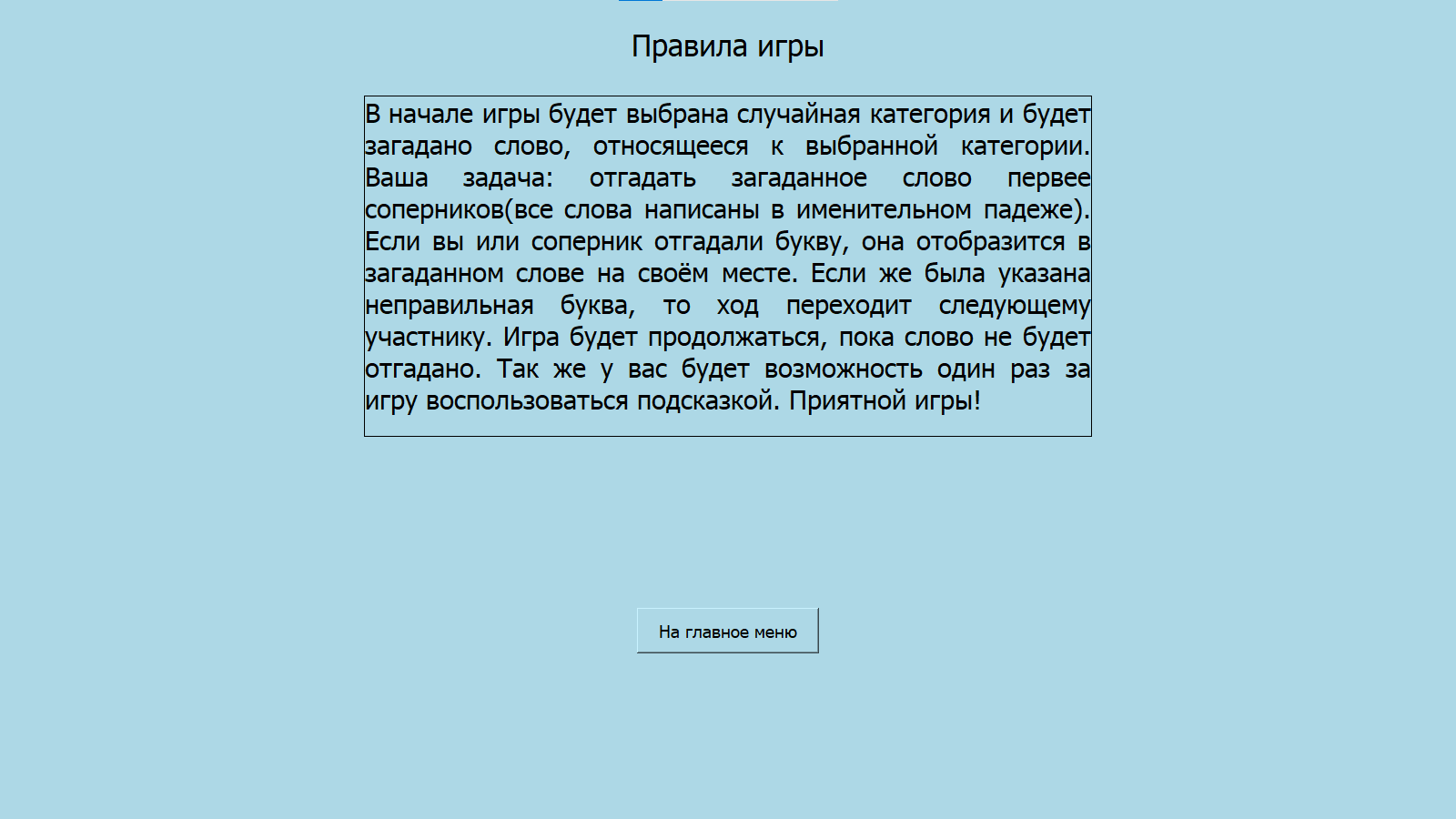


Рисунок 4.4.2

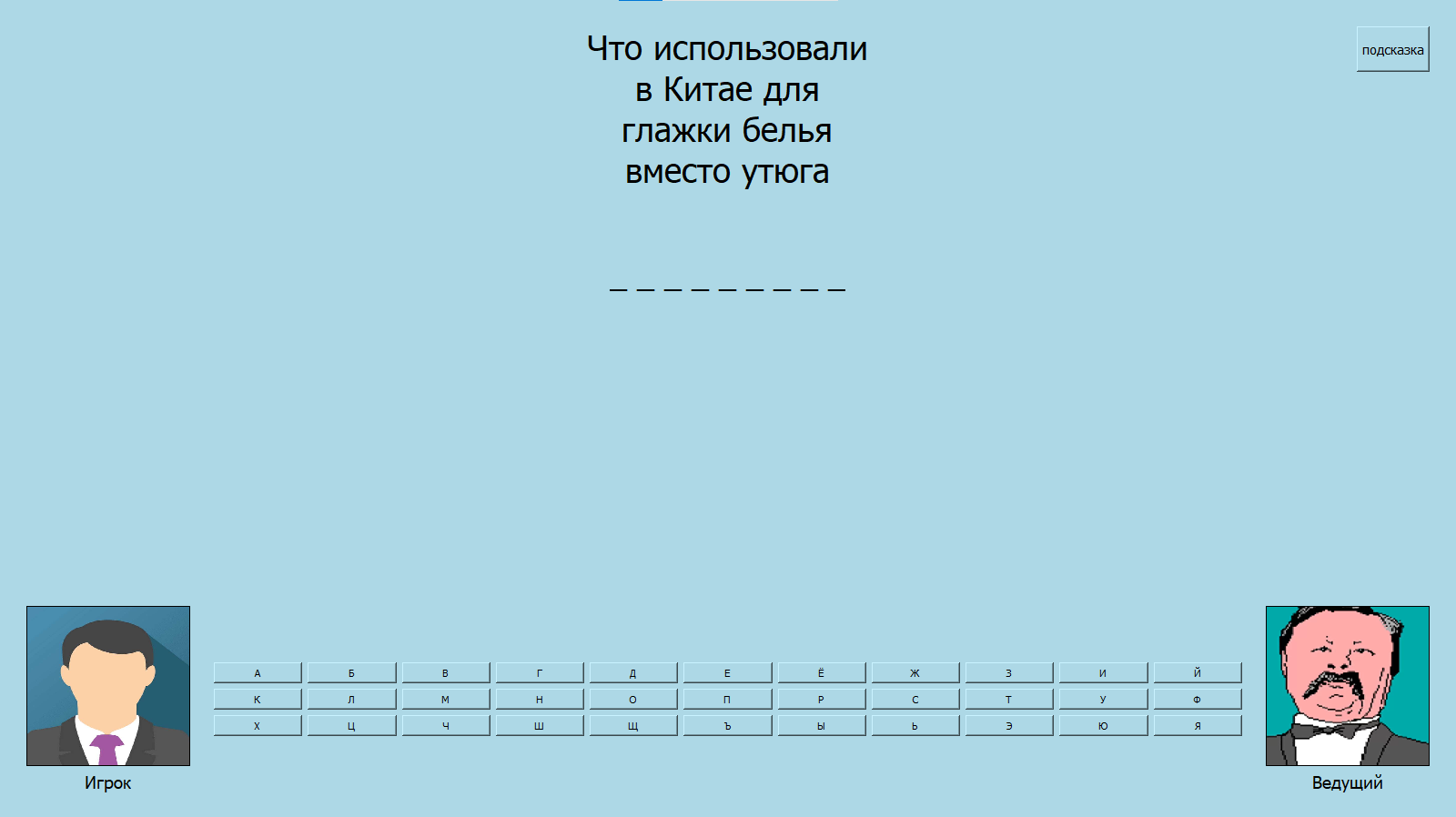


Рисунок 4.4.3

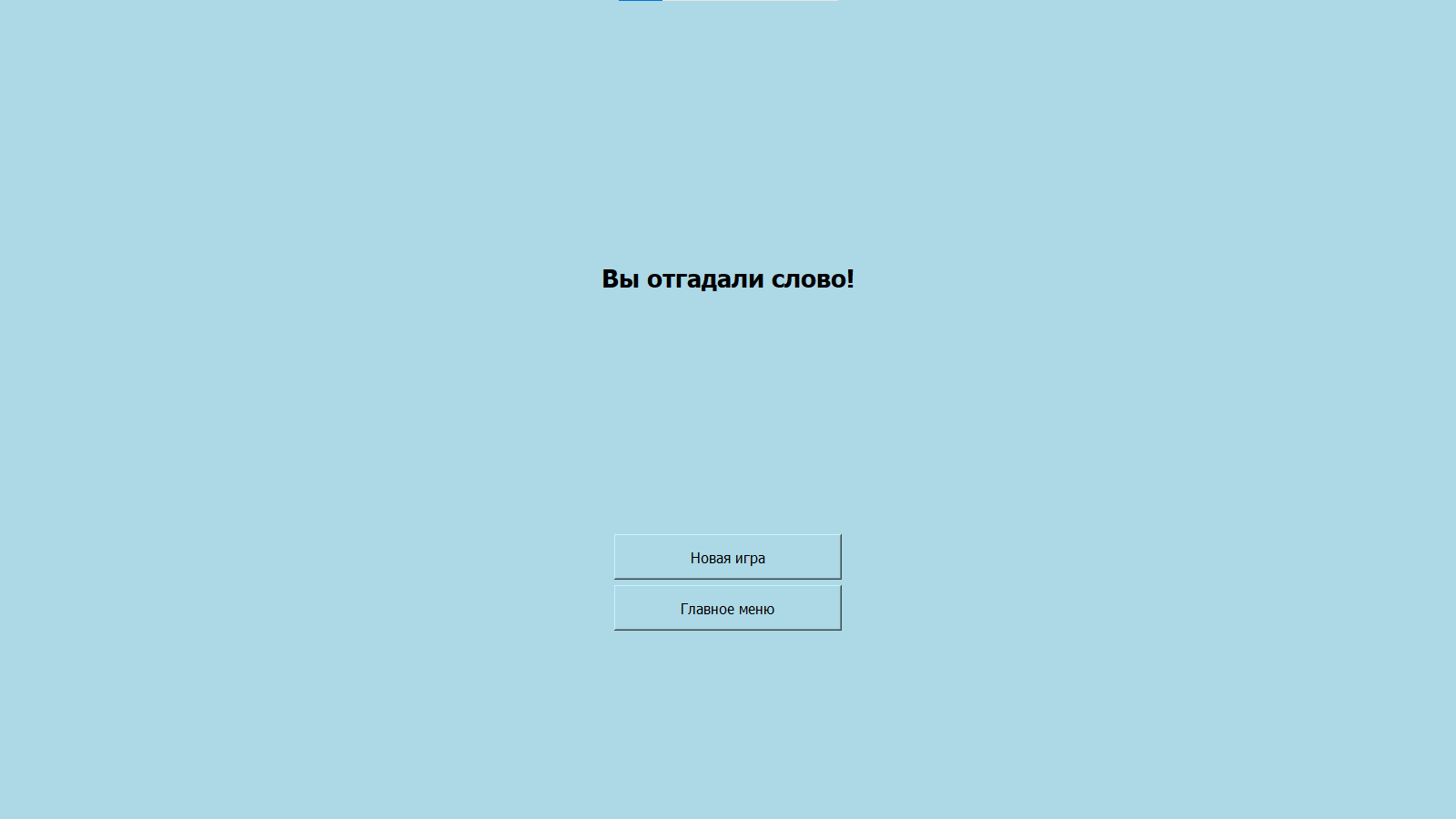


Рисунок 4.4.4

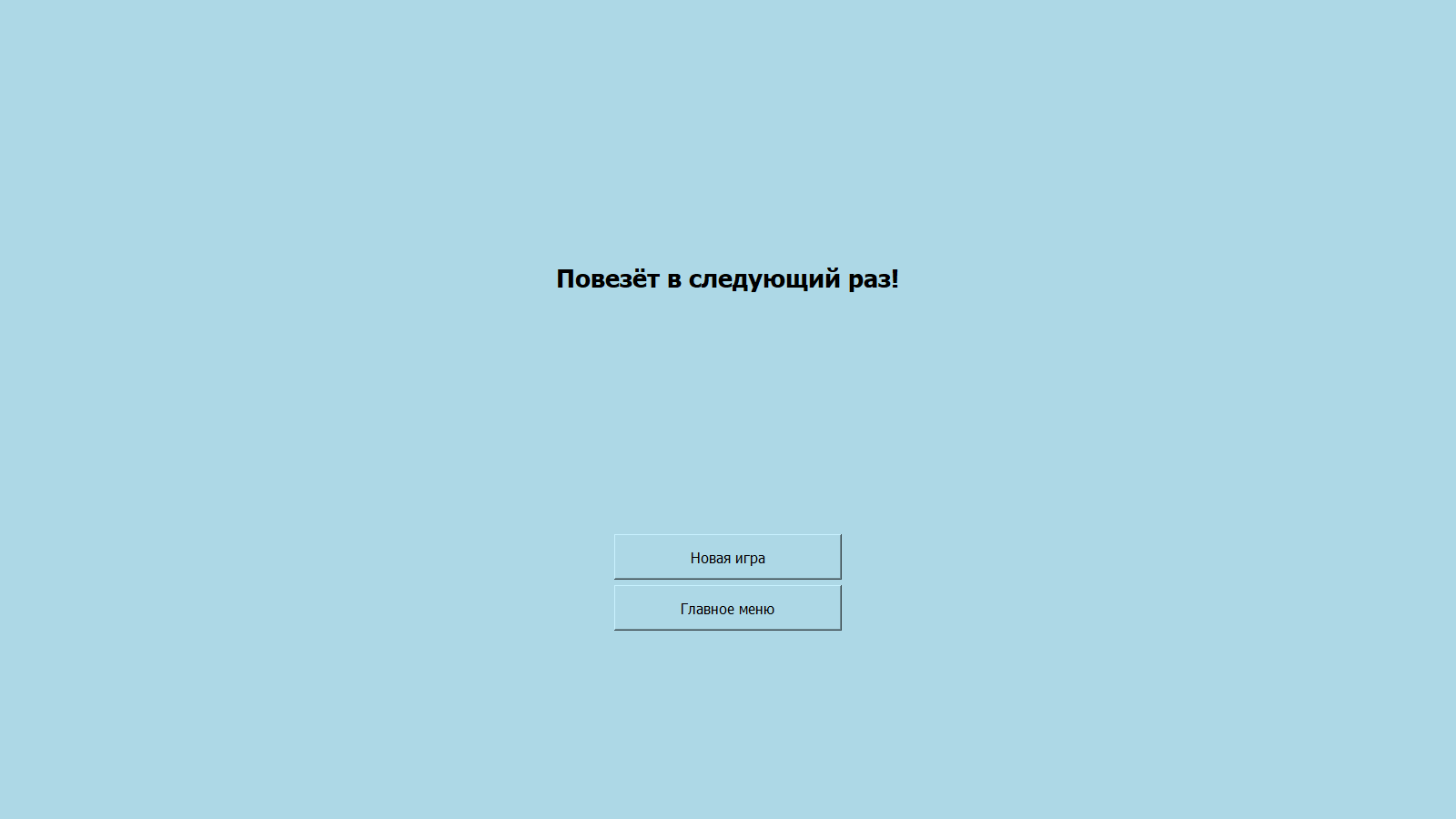


Рисунок 4.4.5

# **ЗАКЛЮЧЕНИЕ**

В ходе учебной практики, в соответствии с индивидуальным заданием, была разработана видеоигра «Поле чудес» и закреплены навыки использования языка программирования Python и библиотеки PyQt5. Были проведены анализ предметной области, разработка программного кода, тестирование, а также написана отчетная документация.

Таким образом, программа учебной практики ПМ.05 в период с 15.01.2024 по 19.06.2024 была выполнена мной в полном объеме.

При выполнении работ, предусмотренных заданием на практику, в организации ФГБОУ ВО «Вятский государственный университет» мной, Караваевым Александром Сергеевичем, соблюдались правила внутреннего трудового распорядка, охраны труда, техники безопасности, противопожарной безопасности, санитарно-эпидемиологические правила и гигиенические нормативы. Пройден вводный инструктаж и инструктаж на рабочем месте. Замечаний не имел.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Эл Свейгарт. Автоматизация рутинных задач с помощью Python: практическое руководство для начинающих: Пер. с англ. – Москва: ООО “И.Д. Вильямс”, 2017. – 592 с.
2. Васильев А. Н. Программирование на Python в примерах и задачах. – Москва: Эксмо, 2021. – 616 с.
3. Руководство по PyQt5 для начинающих – GUI Python. URL: <https://python-scripts.com/pyqt5?ysclid=lxewzzhplx179489292> (дата обращения 10.02.2024).
4. PyQt5 tutorial 2024, Create Python GUIs with Qt. URL: [https://www.pythonguis.com/pyqt5-tutorial/](https://www.pythonguis.com/pyqt5-tutorial/%20) (дата обращения: 20.03.2024).
5. PyQt5 Tutorial – Working with Qt Designer // Codeloop URL: <https://codeloop.org/pyqt5-tutorial-working-with-qt-designer/> (дата обращения 20.03.2024).

# **ПРИЛОЖЕНИЕ А**

**ИСХОДНЫЙ КОД**

**PoleChudes.py**

import PyQt5

from PyQt5 import QtCore, QtWidgets, QtGui

from PyQt5.QtCore import QTimer

from PyQt5.QtWidgets import QApplication, QMainWindow, QMessageBox, QStackedWidget, QWidget

from PyQt5.QtGui import QPixmap, QIcon

from generation\_word import word\_categories

from dictionary import word\_categories\_dictionary

from PyQt5.QtGui import QIcon

import random

import re

class Menu\_page(QWidget):

def \_\_init\_\_(self, changer):

super().\_\_init\_\_() # окно меню

self.changer = changer

self.gridLayout\_4 = QtWidgets.QGridLayout(self)

self.gridLayout\_4.setObjectName("gridLayout\_4")

self.frame\_2 = QtWidgets.QFrame(self)

self.frame\_2.setMaximumSize(QtCore.QSize(16777215, 400))

self.frame\_2.setFrameShape(QtWidgets.QFrame.StyledPanel)

self.frame\_2.setFrameShadow(QtWidgets.QFrame.Raised)

self.frame\_2.setObjectName("frame\_2")

self.gridLayout\_3 = QtWidgets.QGridLayout(self.frame\_2)

self.gridLayout\_3.setObjectName("gridLayout\_3")

self.label = QtWidgets.QLabel(self.frame\_2)

self.label.setMaximumSize(QtCore.QSize(16777215, 200))

font = QtGui.QFont()

font.setPointSize(45)

self.label.setFont(font)

self.label.setObjectName("label")

self.gridLayout\_3.addWidget(self.label, 0, 0, 1, 1, QtCore.Qt.AlignHCenter)

self.gridLayout\_4.addWidget(self.frame\_2, 0, 0, 1, 1)

self.frame = QtWidgets.QFrame(self)

self.frame.setFrameShape(QtWidgets.QFrame.StyledPanel)

self.frame.setFrameShadow(QtWidgets.QFrame.Raised)

self.frame.setObjectName("frame")

self.gridLayout\_2 = QtWidgets.QGridLayout(self.frame)

self.gridLayout\_2.setObjectName("gridLayout\_2")

self.gridLayout = QtWidgets.QGridLayout()

self.gridLayout.setVerticalSpacing(20)

self.gridLayout.setObjectName("gridLayout")

self.Rules\_Button = QtWidgets.QPushButton(self.frame)

self.Rules\_Button.setMaximumSize(QtCore.QSize(250, 50))

font = QtGui.QFont()

font.setPointSize(12)

self.Rules\_Button.setFont(font)

self.Rules\_Button.setObjectName("Rules\_Button")

self.gridLayout.addWidget(self.Rules\_Button, 1, 0, 1, 1)

self.Play\_Button = QtWidgets.QPushButton(self.frame)

self.Play\_Button.setMaximumSize(QtCore.QSize(250, 50))

font = QtGui.QFont()

font.setPointSize(12)

self.Play\_Button.setFont(font)

self.Play\_Button.setObjectName("Play\_Button")

self.gridLayout.addWidget(self.Play\_Button, 0, 0, 1, 1)

self.Exit\_Button = QtWidgets.QPushButton(self.frame)

self.Exit\_Button.setMaximumSize(QtCore.QSize(250, 50))

font = QtGui.QFont()

font.setPointSize(12)

self.Exit\_Button.setFont(font)

self.Exit\_Button.setObjectName("Exit\_Button")

self.gridLayout.addWidget(self.Exit\_Button, 2, 0, 1, 1)

self.gridLayout\_2.addLayout(self.gridLayout, 0, 0, 1, 1)

self.gridLayout\_4.addWidget(self.frame, 1, 0, 1, 1)

# self.setCentralWidget(self.centralwidget)

# Интерфейс под себя

self.retranslateUi\_Menu()

QtCore.QMetaObject.connectSlotsByName(self)

def retranslateUi\_Menu(self):

\_translate = QtCore.QCoreApplication.translate

self.setWindowTitle(\_translate("MainWindow", "Поле чудес"))

self.label.setText(\_translate("MainWindow", "Поле чудес"))

self.Rules\_Button.setText(\_translate("MainWindow", "Правила игры"))

self.Rules\_Button.clicked.connect(self.changer.rules)

self.Play\_Button.setText(\_translate("MainWindow", "Новая игра"))

self.Play\_Button.clicked.connect(self.changer.game)

self.Exit\_Button.setText(\_translate("MainWindow", "Выйти из игры"))

self.Exit\_Button.clicked.connect(self.exit\_game)

def exit\_game(self):

QApplication.quit()

class GameWindow(QWidget):

def \_\_init\_\_(self, changer):

super().\_\_init\_\_() # игровое окно

# self.main\_window = main\_window

self.changer = changer

self.hidden\_word\_display = [] # стображение слова пользователю

self.guessed\_letters = [] # список использованных букв

self.guessed\_letters\_in\_word = [] # список отгаданных букв

self.random\_word = "" # случайно выбранное слово

self.timer\_host = QTimer(self)

self.timer\_host.setSingleShot(True)

self.timer\_host.timeout.connect(self.clear\_label\_host)

self.gridLayout = QtWidgets.QGridLayout(self)

self.gridLayout.setObjectName("gridLayout")

self.frame = QtWidgets.QFrame(self)

self.frame.setFrameShape(QtWidgets.QFrame.StyledPanel)

self.frame.setFrameShadow(QtWidgets.QFrame.Raised)

self.frame.setObjectName("frame")

self.gridLayout\_2 = QtWidgets.QGridLayout(self.frame)

self.gridLayout\_2.setObjectName("gridLayout\_2")

self.frame\_3 = QtWidgets.QFrame(self.frame)

self.frame\_3.setMaximumSize(QtCore.QSize(200, 225))

self.frame\_3.setFrameShape(QtWidgets.QFrame.StyledPanel)

self.frame\_3.setFrameShadow(QtWidgets.QFrame.Raised)

self.frame\_3.setObjectName("frame\_3")

self.verticalLayout\_2 = QtWidgets.QVBoxLayout(self.frame\_3)

self.verticalLayout\_2.setObjectName("verticalLayout\_2")

self.verticalLayout = QtWidgets.QVBoxLayout()

self.verticalLayout.setObjectName("verticalLayout")

self.label\_2 = QtWidgets.QLabel(self.frame\_3)

self.label\_2.setAlignment(QtCore.Qt.AlignCenter)

self.label\_2.setObjectName("label\_2")

self.verticalLayout.addWidget(self.label\_2)

self.label\_3 = QtWidgets.QLabel(self.frame\_3)

font = QtGui.QFont()

font.setPointSize(14)

self.label\_3.setFont(font)

self.label\_3.setMaximumSize(QtCore.QSize(16777215, 25))

self.label\_3.setObjectName("label\_3")

self.verticalLayout.addWidget(self.label\_3, 0, QtCore.Qt.AlignHCenter)

self.verticalLayout\_2.addLayout(self.verticalLayout)

self.gridLayout\_2.addWidget(self.frame\_3, 2, 0, 1, 1)

self.frame\_4 = QtWidgets.QFrame(self.frame)

self.frame\_4.setMaximumSize(QtCore.QSize(200, 225))

self.frame\_4.setFrameShape(QtWidgets.QFrame.StyledPanel)

self.frame\_4.setFrameShadow(QtWidgets.QFrame.Raised)

self.frame\_4.setObjectName("frame\_4")

self.verticalLayout\_4 = QtWidgets.QVBoxLayout(self.frame\_4)

self.verticalLayout\_4.setObjectName("verticalLayout\_4")

self.verticalLayout\_3 = QtWidgets.QVBoxLayout()

self.verticalLayout\_3.setObjectName("verticalLayout\_3")

self.label\_4 = QtWidgets.QLabel(self.frame\_4)

self.label\_4.setAlignment(QtCore.Qt.AlignCenter)

self.label\_4.setObjectName("label\_4")

self.verticalLayout\_3.addWidget(self.label\_4)

self.label\_5 = QtWidgets.QLabel(self.frame\_4)

font = QtGui.QFont()

font.setPointSize(14)

self.label\_5.setFont(font)

self.label\_5.setMaximumSize(QtCore.QSize(16777215, 25))

self.label\_5.setObjectName("label\_5")

self.verticalLayout\_3.addWidget(self.label\_5, 0, QtCore.Qt.AlignHCenter)

self.verticalLayout\_4.addLayout(self.verticalLayout\_3)

self.gridLayout\_2.addWidget(self.frame\_4, 2, 2, 1, 1)

self.frame\_5 = QtWidgets.QFrame(self.frame)

self.frame\_5.setFrameShape(QtWidgets.QFrame.StyledPanel)

self.frame\_5.setFrameShadow(QtWidgets.QFrame.Raised)

self.frame\_5.setObjectName("frame\_5")

self.gridLayout\_5 = QtWidgets.QGridLayout(self.frame\_5)

self.gridLayout\_5.setObjectName("gridLayout\_5")

self.gridLayout\_4 = QtWidgets.QGridLayout()

# создание алфавита

self.alphabet = [chr(i) for i in range(1040, 1072)]

self.alphabet.insert(6, chr(1025))

self.create\_keyboard()

self.gridLayout\_4.setObjectName("gridLayout\_4")

self.gridLayout\_5.addLayout(self.gridLayout\_4, 0, 0, 1, 1)

self.gridLayout\_2.addWidget(self.frame\_5, 2, 1, 1, 1)

self.frame\_2 = QtWidgets.QFrame(self.frame)

self.frame\_2.setFrameShape(QtWidgets.QFrame.StyledPanel)

self.frame\_2.setFrameShadow(QtWidgets.QFrame.Raised)

self.frame\_2.setObjectName("frame\_2")

self.horizontalLayout = QtWidgets.QHBoxLayout(self.frame\_2)

self.horizontalLayout.setObjectName("horizontalLayout")

self.gridLayout\_3 = QtWidgets.QGridLayout()

self.gridLayout\_3.setObjectName("gridLayout\_3")

font = QtGui.QFont()

font.setPointSize(26)

self.label\_player = QtWidgets.QLabel(self.frame)

self.label\_player.setFont(font)

self.label\_player.setAlignment(QtCore.Qt.AlignCenter | QtCore.Qt.AlignBottom)

self.gridLayout\_2.addWidget(self.label\_player, 1, 0, 1, 1)

self.label\_host = QtWidgets.QLabel(self.frame)

self.label\_host.setFont(font)

self.label\_host.setAlignment(QtCore.Qt.AlignCenter | QtCore.Qt.AlignBottom)

self.gridLayout\_2.addWidget(self.label\_host, 1, 2, 1, 1)

self.label\_category = QtWidgets.QLabel(self.frame\_2)

self.label\_category.setWordWrap(True)

self.label\_category.setAlignment(QtCore.Qt.AlignJustify)

self.label\_category.setFont(font)

self.gridLayout\_3.addWidget(self.label\_category, 0, 0, 0, 0, QtCore.Qt.AlignHCenter | QtCore.Qt.AlignTop)

self.label\_word = QtWidgets.QLabel(self.frame\_2)

self.label\_word.setFont(font)

self.label\_word.setObjectName("label\_word")

self.gridLayout\_3.addWidget(self.label\_word, 0, 0, 0, 0, QtCore.Qt.AlignHCenter | QtCore.Qt.AlignBottom)

self.horizontalLayout.addLayout(self.gridLayout\_3)

self.gridLayout\_2.addWidget(self.frame\_2, 0, 0, 1, 3)

self.frame\_6 = QtWidgets.QFrame(self.frame)

self.frame\_6.setFrameShape(QtWidgets.QFrame.StyledPanel)

self.frame\_6.setFrameShadow(QtWidgets.QFrame.Raised)

self.frame\_6.setObjectName("frame\_6")

self.gridLayout\_2.addWidget(self.frame\_6, 1, 1, 1, 1)

self.gridLayout.addWidget(self.frame, 0, 0, 1, 1)

self.button\_frame = QtWidgets.QFrame(self.frame)

self.button\_frame.setFrameShape(QtWidgets.QFrame.StyledPanel)

self.button\_frame.setFrameShadow(QtWidgets.QFrame.Raised)

self.button\_layout = QtWidgets.QHBoxLayout(self.button\_frame)

self.button\_help = QtWidgets.QPushButton("подсказка", self.button\_frame)

self.button\_help.setFixedSize(80, 50)

font = QtGui.QFont()

font.setPointSize(11)

self.button\_help.setFont(font)

self.button\_layout.addWidget(self.button\_help)

self.gridLayout\_2.addWidget(self.button\_frame, 0, 2, 1, 1, QtCore.Qt.AlignTop | QtCore.Qt.AlignRight)

self.button\_clicked\_list = []

# self.setCentralWidget(self.centralwidget)

self.retranslateUi\_game()

QtCore.QMetaObject.connectSlotsByName(self)

def create\_keyboard(self):

for i in range(3):

for j in range(11):

btn = QtWidgets.QPushButton()

btn.setText(self.alphabet[i \* 11 + j])

btn.clicked.connect(self.button\_clicked)

self.gridLayout\_4.addWidget(btn, i, j)

def button\_clicked(self): # функция для обработки нажатия кнопок на виртуальной клавиатуре

button = self.sender() # кнопка, которая была нажата

if self.input\_locked:

return

if button:

self.button\_clicked\_list.append(button)

self.label\_host.clear()

self.input\_locked = True

button.setEnabled(False)

letter = button.text().lower()

self.user\_letter(letter)

self.guessed\_letters.append(letter)

QTimer.singleShot(6000, self.clear\_label\_player)

if letter in self.random\_word:

self.result = 1

QTimer.singleShot(6500, self.unlock\_input)

QTimer.singleShot(2500, lambda: (self.there\_letter(), QTimer.singleShot(3500, self.clear\_label\_host)))

QTimer.singleShot(5000, lambda: (self.there\_letter(), self.update\_hidden\_word\_delayed(letter)))

self.guessed\_letters\_in\_word.append(letter)

else:

self.result = 0

QTimer.singleShot(3000, lambda: (self.there\_no\_letter(), QTimer.singleShot(3000, self.clear\_label\_host)))

self.input\_locked\_bot = False

QTimer.singleShot(7000, self.first\_bot)

QTimer.singleShot(8000, self.guess\_letter\_one)

def retranslateUi\_game(self):

\_translate = QtCore.QCoreApplication.translate

self.setWindowTitle(\_translate("MainWindow", "Поле чудес"))

self.label\_2.setText(\_translate("MainWindow", "TextLabel"))

self.label\_3.setText(\_translate("MainWindow", "Игрок"))

self.label\_4.setText(\_translate("MainWindow", "TextLabel"))

self.label\_5.setText(\_translate("MainWindow", "Ведущий"))

pixmap1 = QPixmap('картинки/аватар.jpg')

self.label\_2.setPixmap(pixmap1)

self.scaled\_pixmap1 = pixmap1.scaled(250, 200)

self.label\_2.setPixmap(self.scaled\_pixmap1)

pixmap2 = QPixmap('картинки/ведущий.jpeg')

self.label\_4.setPixmap(pixmap2)

self.scaled\_pixmap2 = pixmap2.scaled(250, 200)

self.label\_4.setPixmap(self.scaled\_pixmap2)

self.button\_help.clicked.connect(self.hint)

app.setStyleSheet('#label\_2, #label\_4{ border: 1px solid black}; #label{ font-size: 26pt}')

self.restart\_game()

def restart\_game(self): #обновление окна

# выбор случайной подсказки

self.random\_category = random.choice(list(word\_categories.keys()))

self.label\_category.setText(f"<center><span style='font-size: 28pt;'>{self.random\_category}</span></center>")

# выбор слова из выбранной подсказки

self.hidden\_word\_display.clear()

self.random\_word = random.choice(word\_categories[self.random\_category])

self.hidden\_word\_display = list("\_" \* len(self.random\_word))

self.label\_word.setText(' '.join(self.hidden\_word\_display)) # установка скрытого слова в label

self.unlock\_input()

self.label\_host.clear()

self.button\_help.setEnabled(True)

self.guessed\_letters.clear()

self.guessed\_letters\_in\_word.clear()

if len(self.button\_clicked\_list) != 0:

for i in range(len(self.button\_clicked\_list)):

self.button\_clicked\_list[i].setEnabled(True)

self.button\_clicked\_list.clear()

def hint(self): # подсказка игроку

if self.input\_locked:

return

unopened\_letters = set(self.random\_word) - set(self.guessed\_letters\_in\_word)

if unopened\_letters:

letter = random.choice(list(unopened\_letters))

self.help = f"Назовите \n букву \"{letter}\""

self.label\_host.setText(self.help)

self.button\_help.setEnabled(False)

def user\_letter(self, letter): # Буква выбранная игроком/ботом

self.chosen\_letter = f"Буква \"{letter}\""

self.label\_player.setText(self.chosen\_letter)

def clear\_label\_host(self): # Очищение ответа ведущего

self.label\_host.clear()

def clear\_label\_player(self): # Очищение ответа игрока/бота

self.label\_player.clear()

def first\_bot(self): # Ходит первый компьютер

self.first = f"Первый компьютер"

self.label\_3.setText(self.first)

def second\_bot(self): # Ходит второй компьютер

self.second = f"Второй компьютер"

self.label\_3.setText(self.second)

def guess\_letter\_one(self): # Работа первого бота

if "\_" not in self.hidden\_word\_display:

return

if self.input\_locked\_bot:

self.unlock\_input()

return

self.guess\_letters()

possible\_words = self.get\_possible\_words(self.label\_word.text().replace(' ', ''))

chance = random.randint(1, 9)

if possible\_words:

if len(self.random\_word) <= 6 and len(list(set(self.random\_word))) - len(self.guessed\_letters\_in\_word) <= 1:

random\_word = random.choice(possible\_words)

letter = random.choice(random\_word)

elif len(self.random\_word) > 6 and len(self.random\_word) <= 8 and len(list(set(self.random\_word))) - len(

self.guessed\_letters\_in\_word) <= 2:

random\_word = random.choice(possible\_words)

letter = random.choice(random\_word)

elif len(self.random\_word) > 8 and len(list(set(self.random\_word))) - len(

self.guessed\_letters\_in\_word) <= 3:

random\_word = random.choice(possible\_words)

letter = random.choice(random\_word)

elif chance in range(1, 5):

random\_word = random.choice(possible\_words)

letter = random.choice(random\_word)

else:

letter = random.choice(self.alphabet).lower()

if letter in self.guessed\_letters:

QTimer.singleShot(100, self.guess\_letter\_one)

return

self.user\_letter(letter)

self.guessed\_letters.append(letter)

button = self.find\_button\_by\_letter(letter)

if button:

self.button\_clicked\_list.append(button)

self.clear\_keyboard(button)

QTimer.singleShot(6000, self.clear\_label\_player)

if letter in self.random\_word:

QTimer.singleShot(2500, lambda: (self.there\_letter(), QTimer.singleShot(3500, self.clear\_label\_host)))

QTimer.singleShot(5000, lambda: (self.there\_letter(), self.update\_hidden\_word\_delayed(letter)))

QTimer.singleShot(8000, self.guess\_letter\_one)

self.guessed\_letters\_in\_word.append(letter)

else:

QTimer.singleShot(3000, lambda: (self.there\_no\_letter(), QTimer.singleShot(3000, self.clear\_label\_host)))

QTimer.singleShot(8000, self.guess\_letter\_two)

QTimer.singleShot(7000, self.second\_bot)

def guess\_letter\_two(self): # Работа второго бота

if "\_" not in self.hidden\_word\_display:

return

if self.input\_locked\_bot:

self.unlock\_input()

return

self.guess\_letters()

possible\_words = self.get\_possible\_words(self.label\_word.text().replace(' ', ''))

chance = random.randint(1, 9)

if possible\_words:

if len(self.random\_word) <= 6 and len(list(set(self.random\_word))) - len(self.guessed\_letters\_in\_word) <= 1:

random\_word = random.choice(possible\_words)

letter = random.choice(random\_word)

elif len(self.random\_word) > 6 and len(self.random\_word) <= 8 and len(list(set(self.random\_word))) - len(

self.guessed\_letters\_in\_word) <= 2:

random\_word = random.choice(possible\_words)

letter = random.choice(random\_word)

elif len(self.random\_word) > 8 and len(list(set(self.random\_word))) - len(

self.guessed\_letters\_in\_word) <= 3:

random\_word = random.choice(possible\_words)

letter = random.choice(random\_word)

elif chance in range(1, 4):

random\_word = random.choice(possible\_words)

letter = random.choice(random\_word)

else:

letter = random.choice(self.alphabet).lower()

if letter in self.guessed\_letters:

QTimer.singleShot(100, self.guess\_letter\_two)

return

self.user\_letter(letter)

self.guessed\_letters.append(letter)

button = self.find\_button\_by\_letter(letter)

if button:

self.button\_clicked\_list.append(button)

self.clear\_keyboard(button)

QTimer.singleShot(6000, self.clear\_label\_player)

if letter in self.random\_word:

QTimer.singleShot(2500, lambda: (self.there\_letter(), QTimer.singleShot(3500, self.clear\_label\_host)))

QTimer.singleShot(5000, lambda: (self.there\_letter(), self.update\_hidden\_word\_delayed(letter)))

QTimer.singleShot(8000, self.guess\_letter\_two)

self.guessed\_letters\_in\_word.append(letter)

else:

QTimer.singleShot(3000, lambda: (self.there\_no\_letter(), QTimer.singleShot(3000, self.clear\_label\_host)))

QTimer.singleShot(7000, self.unlock\_input)

def there\_no\_letter(self): # Выбранной буквы нет в слове

self.absent = f"Нет такой \n буквы"

self.label\_host.setText(self.absent)

def there\_letter(self): # Выбранная буква есть в слове

self.present = f"Откройте!"

self.label\_host.setText(self.present)

def guess\_letters(self): # создание шаблона слова для дальнейшей работы с ним

pattern = self.label\_word.text().replace(' ', '')

possible\_words = self.get\_possible\_words(pattern)

def get\_possible\_words(self, pattern): # создаём список подходящих под щаблон слов

dictionary = word\_categories\_dictionary

possible\_words = []

for word in dictionary:

if len(word) == len(pattern) and self.matches\_pattern(word, pattern):

possible\_words.append(word)

return possible\_words

def matches\_pattern(self, word, pattern): # сравниваем буквы шаблона с буквами из словаря

for w\_char, p\_char in zip(word, pattern):

if p\_char != '\_' and w\_char != p\_char:

return False

return True

def unlock\_input(self): # Разрешение игроку выбирать букву

self.input\_locked = False

self.player = f"Игрок"

self.label\_3.setText(self.player)

def keyPressEvent(self, event): # функция обработки нажатия клавишь на клавиатуре

if event.key() == 16777216: # Код клавиши Esc

self.show\_modal()

if self.input\_locked:

return

letter = event.text().lower()

if re.fullmatch(r'[а-я-ё]', letter):

if letter in self.guessed\_letters:

return

if letter.isalpha() and len(letter) == 1:

self.guessed\_letters.append(letter)

self.label\_host.clear()

self.input\_locked = True

self.user\_letter(letter)

button = self.find\_button\_by\_letter(letter)

if button:

self.button\_clicked\_list.append(button)

self.clear\_keyboard(button)

QTimer.singleShot(6000, self.clear\_label\_player)

if letter in self.random\_word:

self.result = 1

QTimer.singleShot(6500, self.unlock\_input)

QTimer.singleShot(2500, lambda: (self.there\_letter(), QTimer.singleShot(3500, self.clear\_label\_host)))

QTimer.singleShot(5000, lambda: (self.there\_letter(), self.update\_hidden\_word\_delayed(letter)))

self.guessed\_letters\_in\_word.append(letter)

else:

self.result = 0

QTimer.singleShot(3000, lambda: (self.there\_no\_letter(), QTimer.singleShot(3000, self.clear\_label\_host)))

self.input\_locked\_bot = False

QTimer.singleShot(8000, self.guess\_letter\_one)

QTimer.singleShot(7000, self.first\_bot)

def find\_button\_by\_letter(self, letter): #функция для нахождения клавиши с выбранной буквой

for i in range(3):

for j in range(11):

button = self.gridLayout\_4.itemAtPosition(i, j).widget()

if button and button.text().lower() == letter:

return button

return None

def clear\_keyboard(self, button): #блокируем букву

button.setEnabled(False)

def update\_hidden\_word\_delayed(self, letter): # обновляем отображаемый пользователю текст загаданного слова

for i in range(len(self.random\_word)):

if self.random\_word[i] == letter:

self.hidden\_word\_display[i] = letter

self.label\_word.setText(' '.join(self.hidden\_word\_display))

self.player\_win()

self.player\_lose()

def player\_win(self):

if "\_" not in self.hidden\_word\_display and self.result == 1:

QTimer.singleShot(3500, self.changer.win)

def player\_lose(self):

if "\_" not in self.hidden\_word\_display and self.result == 0:

self.label\_player.clear()

self.label\_host.clear()

QTimer.singleShot(3500, self.changer.lose)

def show\_modal(self):

msg\_box = QMessageBox(self)

msg\_box.setWindowTitle('Пауза')

msg\_box.setText('Вы хотите вернуться в главное меню?')

msg\_box.setStandardButtons(QMessageBox.Yes | QMessageBox.No)

# Меняем текст кнопок после установки стандартных кнопок

yes\_button = msg\_box.button(QMessageBox.Yes)

yes\_button.setText("Продолжить")

no\_button = msg\_box.button(QMessageBox.No)

no\_button.setText("Главное меню")

response = msg\_box.exec\_()

if response == QMessageBox.No:

self.input\_locked\_bot = True

# Создаем экземпляры окон

self.changer.to\_the\_main\_menu()

class Rules(QWidget):

def \_\_init\_\_(self, changer):

super().\_\_init\_\_() # окно правил

self.changer = changer

self.gridLayout\_6 = QtWidgets.QGridLayout(self)

self.gridLayout\_6.setObjectName("gridLayout\_6")

self.frame\_3 = QtWidgets.QFrame(self)

self.frame\_3.setMaximumSize(QtCore.QSize(16777215, 80))

self.frame\_3.setCursor(QtGui.QCursor(QtCore.Qt.SizeAllCursor))

self.frame\_3.setFrameShape(QtWidgets.QFrame.StyledPanel)

self.frame\_3.setFrameShadow(QtWidgets.QFrame.Raised)

self.frame\_3.setObjectName("frame\_3")

self.verticalLayout = QtWidgets.QVBoxLayout(self.frame\_3)

self.verticalLayout.setObjectName("verticalLayout")

self.label\_3 = QtWidgets.QLabel(self.frame\_3)

font = QtGui.QFont()

font.setPointSize(25)

self.label\_3.setFont(font)

self.label\_3.setAlignment(QtCore.Qt.AlignCenter)

self.label\_3.setObjectName("label\_3")

self.verticalLayout.addWidget(self.label\_3)

self.gridLayout\_6.addWidget(self.frame\_3, 0, 0, 1, 1)

self.frame\_4 = QtWidgets.QFrame(self)

self.frame\_4.setFrameShape(QtWidgets.QFrame.StyledPanel)

self.frame\_4.setFrameShadow(QtWidgets.QFrame.Raised)

self.frame\_4.setObjectName("frame\_4")

self.gridLayout\_5 = QtWidgets.QGridLayout(self.frame\_4)

self.gridLayout\_5.setObjectName("gridLayout\_5")

self.label\_2 = QtWidgets.QLabel(self.frame\_4)

self.label\_2.setMinimumSize(QtCore.QSize(550, 300))

self.label\_2.setMaximumSize(QtCore.QSize(800, 16777215))

font = QtGui.QFont()

font.setPointSize(20)

self.label\_2.setFont(font)

self.label\_2.setTextFormat(QtCore.Qt.AutoText)

self.label\_2.setScaledContents(False)

self.label\_2.setAlignment(QtCore.Qt.AlignLeading | QtCore.Qt.AlignLeft | QtCore.Qt.AlignVCenter)

self.label\_2.setWordWrap(True)

self.label\_2.setObjectName("label\_2")

self.gridLayout\_5.addWidget(self.label\_2, 0, 0, 1, 1)

self.gridLayout\_6.addWidget(self.frame\_4, 1, 0, 1, 1)

self.frame\_5 = QtWidgets.QFrame(self)

self.frame\_5.setFrameShape(QtWidgets.QFrame.StyledPanel)

self.frame\_5.setFrameShadow(QtWidgets.QFrame.Raised)

self.frame\_5.setObjectName("frame\_5")

self.gridLayout\_7 = QtWidgets.QGridLayout(self.frame\_5)

self.gridLayout\_7.setObjectName("gridLayout\_7")

self.main\_menu = QtWidgets.QPushButton(self.frame\_5)

self.main\_menu.setMaximumSize(QtCore.QSize(200, 50))

font = QtGui.QFont()

font.setPointSize(14)

self.main\_menu.setFont(font)

self.main\_menu.setObjectName("main\_menu")

self.gridLayout\_7.addWidget(self.main\_menu, 0, 0, 1, 1)

self.gridLayout\_6.addWidget(self.frame\_5, 2, 0, 1, 1)

self.retranslateUi\_Rules()

QtCore.QMetaObject.connectSlotsByName(self)

def retranslateUi\_Rules(self):

\_translate = QtCore.QCoreApplication.translate

self.setWindowTitle(\_translate("MainWindow", "Поле чудес"))

self.label\_3.setText(\_translate("MainWindow", "Правила игры"))

self.label\_2.setStyleSheet('font:22pt')

self.label\_2.setAlignment(QtCore.Qt.AlignJustify)

self.label\_2.setText(\_translate("MainWindow",

"В начале игры будет выбрана случайная категория и будет загадано слово, относящееся к выбранной категории. Ваша задача: отгадать загаданное слово первее соперников(все слова написаны в именительном падеже). Если вы или соперник отгадали букву, она отобразится в загаданном слове на своём месте. Если же была указана неправильная буква, то ход переходит следующему участнику. Игра будет продолжаться, пока слово не будет отгадано. Так же у вас будет возможность один раз за игру воспользоваться подсказкой. Приятной игры!"))

self.main\_menu.setText(\_translate("MainWindow", "На главное меню"))

self.main\_menu.clicked.connect(self.changer.to\_the\_main\_menu)

class Win(QWidget):

def \_\_init\_\_(self, changer):

super().\_\_init\_\_() # окно правил

self.changer = changer

self.gridLayout = QtWidgets.QGridLayout(self)

self.gridLayout.setObjectName("gridLayout")

self.frame = QtWidgets.QFrame(self)

self.frame.setFrameShape(QtWidgets.QFrame.StyledPanel)

self.frame.setFrameShadow(QtWidgets.QFrame.Raised)

self.frame.setObjectName("frame")

self.gridLayout\_2 = QtWidgets.QGridLayout(self.frame)

self.gridLayout\_2.setObjectName("gridLayout\_2")

self.frame\_2 = QtWidgets.QFrame(self.frame)

self.frame\_2.setFrameShape(QtWidgets.QFrame.StyledPanel)

self.frame\_2.setFrameShadow(QtWidgets.QFrame.Raised)

self.frame\_2.setObjectName("frame\_2")

self.verticalLayout\_3 = QtWidgets.QVBoxLayout(self.frame\_2)

self.verticalLayout\_3.setObjectName("verticalLayout\_3")

self.label = QtWidgets.QLabel(self.frame\_2)

self.label.setMaximumSize(QtCore.QSize(16777215, 200))

font = QtGui.QFont()

font.setPointSize(20)

font.setBold(True)

font.setItalic(False)

font.setUnderline(False)

font.setWeight(75)

self.label.setFont(font)

self.label.setAlignment(QtCore.Qt.AlignCenter)

self.label.setObjectName("label")

self.verticalLayout\_3.addWidget(self.label)

self.gridLayout\_3 = QtWidgets.QGridLayout()

self.gridLayout\_3.setContentsMargins(0, -1, -1, -1)

self.gridLayout\_3.setObjectName("gridLayout\_3")

self.pushButton = QtWidgets.QPushButton(self.frame\_2)

self.pushButton.setMaximumSize(QtCore.QSize(250, 50))

font = QtGui.QFont()

font.setPointSize(12)

self.pushButton.setFont(font)

self.pushButton.setObjectName("pushButton")

self.gridLayout\_3.addWidget(self.pushButton, 0, 0, 1, 1)

self.pushButton\_2 = QtWidgets.QPushButton(self.frame\_2)

sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Minimum, QtWidgets.QSizePolicy.Fixed)

sizePolicy.setHorizontalStretch(0)

sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(self.pushButton\_2.sizePolicy().hasHeightForWidth())

self.pushButton\_2.setSizePolicy(sizePolicy)

self.pushButton\_2.setMaximumSize(QtCore.QSize(250, 50))

self.pushButton\_2.setBaseSize(QtCore.QSize(0, 0))

font = QtGui.QFont()

font.setPointSize(12)

self.pushButton\_2.setFont(font)

self.pushButton\_2.setLayoutDirection(QtCore.Qt.LeftToRight)

self.pushButton\_2.setObjectName("pushButton\_2")

self.gridLayout\_3.addWidget(self.pushButton\_2, 1, 0, 1, 1)

self.verticalLayout\_3.addLayout(self.gridLayout\_3)

self.gridLayout\_2.addWidget(self.frame\_2, 0, 0, 1, 1)

self.gridLayout.addWidget(self.frame, 0, 0, 1, 1)

# self.setCentralWidget(self.centralwidget)

self.retranslateUi\_win()

QtCore.QMetaObject.connectSlotsByName(self)

def retranslateUi\_win(self):

\_translate = QtCore.QCoreApplication.translate

self.setWindowTitle(\_translate("MainWindow", "MainWindow"))

self.label.setText(\_translate("MainWindow", "Вы отгадали слово!"))

self.pushButton.setText(\_translate("MainWindow", "Новая игра"))

self.pushButton.clicked.connect(self.changer.game)

self.pushButton\_2.setText(\_translate("MainWindow", "Главное меню"))

self.pushButton\_2.clicked.connect(self.changer.to\_the\_main\_menu)

class Lose(QWidget):

def \_\_init\_\_(self, changer):

super().\_\_init\_\_()

self.changer = changer

self.gridLayout = QtWidgets.QGridLayout(self)

self.gridLayout.setObjectName("gridLayout")

self.frame = QtWidgets.QFrame(self)

self.frame.setFrameShape(QtWidgets.QFrame.StyledPanel)

self.frame.setFrameShadow(QtWidgets.QFrame.Raised)

self.frame.setObjectName("frame")

self.gridLayout\_2 = QtWidgets.QGridLayout(self.frame)

self.gridLayout\_2.setObjectName("gridLayout\_2")

self.frame\_2 = QtWidgets.QFrame(self.frame)

self.frame\_2.setFrameShape(QtWidgets.QFrame.StyledPanel)

self.frame\_2.setFrameShadow(QtWidgets.QFrame.Raised)

self.frame\_2.setObjectName("frame\_2")

self.verticalLayout\_3 = QtWidgets.QVBoxLayout(self.frame\_2)

self.verticalLayout\_3.setObjectName("verticalLayout\_3")

self.label = QtWidgets.QLabel(self.frame\_2)

self.label.setMaximumSize(QtCore.QSize(16777215, 200))

font = QtGui.QFont()

font.setPointSize(20)

font.setBold(True)

font.setItalic(False)

font.setUnderline(False)

font.setWeight(75)

self.label.setFont(font)

self.label.setAlignment(QtCore.Qt.AlignCenter)

self.label.setObjectName("label")

self.verticalLayout\_3.addWidget(self.label)

self.gridLayout\_3 = QtWidgets.QGridLayout()

self.gridLayout\_3.setContentsMargins(0, -1, -1, -1)

self.gridLayout\_3.setObjectName("gridLayout\_3")

self.pushButton = QtWidgets.QPushButton(self.frame\_2)

self.pushButton.setMaximumSize(QtCore.QSize(250, 50))

font = QtGui.QFont()

font.setPointSize(12)

self.pushButton.setFont(font)

self.pushButton.setObjectName("pushButton")

self.gridLayout\_3.addWidget(self.pushButton, 0, 0, 1, 1)

self.pushButton\_2 = QtWidgets.QPushButton(self.frame\_2)

sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Minimum, QtWidgets.QSizePolicy.Fixed)

sizePolicy.setHorizontalStretch(0)

sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(self.pushButton\_2.sizePolicy().hasHeightForWidth())

self.pushButton\_2.setSizePolicy(sizePolicy)

self.pushButton\_2.setMaximumSize(QtCore.QSize(250, 50))

self.pushButton\_2.setBaseSize(QtCore.QSize(0, 0))

font = QtGui.QFont()

font.setPointSize(12)

self.pushButton\_2.setFont(font)

self.pushButton\_2.setLayoutDirection(QtCore.Qt.LeftToRight)

self.pushButton\_2.setObjectName("pushButton\_2")

self.gridLayout\_3.addWidget(self.pushButton\_2, 1, 0, 1, 1)

self.verticalLayout\_3.addLayout(self.gridLayout\_3)

self.gridLayout\_2.addWidget(self.frame\_2, 0, 0, 1, 1)

self.gridLayout.addWidget(self.frame, 0, 0, 1, 1)

self.retranslateUi\_lose()

QtCore.QMetaObject.connectSlotsByName(self)

def retranslateUi\_lose(self):

\_translate = QtCore.QCoreApplication.translate

self.setWindowTitle(\_translate("MainWindow", "MainWindow"))

self.label.setText(\_translate("MainWindow", "Повезёт в следующий раз!"))

self.pushButton.setText(\_translate("MainWindow", "Новая игра"))

self.pushButton.clicked.connect(self.changer.game)

self.pushButton\_2.setText(\_translate("MainWindow", "Главное меню"))

self.pushButton\_2.clicked.connect(self.changer.to\_the\_main\_menu)

class Ui\_SecondaryWindows(QMainWindow):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.setWindowIcon(QIcon('картинки/иконка.jpg'))

self.setStyleSheet("background-color: lightblue;")

class changer\_page(QStackedWidget):

def \_\_init\_\_(self, parent=None):

super(changer\_page, self).\_\_init\_\_(parent)

# Добавляем окна в QStackedWidget

self.Menu = Menu\_page(self)

self.Game = GameWindow(self)

self.Rules = Rules(self)

self.Win = Win(self)

self.Lose = Lose(self)

self.addWidget(self.Menu)

self.addWidget(self.Game)

self.addWidget(self.Rules)

self.addWidget(self.Win)

self.addWidget(self.Lose)

# Устанавливаем текущее окно (меню)

self.setCurrentIndex(0)

def to\_the\_main\_menu(self):

self.setCurrentIndex(0)

def game(self):

self.Game.restart\_game()

self.setCurrentIndex(1)

def rules(self):

self.setCurrentIndex(2)

def win(self):

self.setCurrentIndex(3)

def lose(self):

self.setCurrentIndex(4)

if \_\_name\_\_ == "\_\_main\_\_":

import sys

app = QtWidgets.QApplication(sys.argv)

MainWindow = Ui\_SecondaryWindows()

changer = changer\_page()

MainWindow.setCentralWidget(changer)

MainWindow.setMinimumSize(1152, 864)

MainWindow.showFullScreen()

app.exec\_()

**generation\_word.py**

word\_categories = {

'Какое существо может задержать дыхание на 6 суток': ['скорпион'],

'Какая страна не имеет столицы': ['науру'],

'Какое живое существо может спать, не просыпаясь 3 года': ['улитка'],

'Какая кость самая маленькая в человеческом организме': ['стремя'],

'У какого живого существа кровь синего цвета': ['осьминог'],

'У какого животного три сердца': ['осьминог'],

'Какое животное имеет 3 глаза':['туатара'],

'Какое животное может переварить стальной гвоздь':['крокодил'],

'Какое животное одновременно дает молоко и откладывает яйца':['утконос'],

'Какая страна мира имеет две столицы':['боливия'],

'В этой стране живут белые слоны':['тайланд'],

'В процессе чего человек способен произвести воздушный поток силой 170 км/ч':['чихание'],

'Что использовали в Китае для глажки белья вместо утюга':['сковорода'],

'Как в старину называли сторожа городских ворот':['вратарь'],

'Как называется железная дорога с канатной тягой, устанавливаемая на крутых подъемах':['фуникулёр'],

'Какое животное дало название распространенному в Древнем Риме способу боевого построения':['черепаха'],

'Цитрусовое дерево с ароматными цветками':['бергамот'],

'Зная это, мы можем понять, как работает устройство':['структура'],

'В Чили находится самое крупное сооружение подобного типа, длина которого – 1 километр':['бассейн'],

'Единственное ядовитое млекопитающее в мире': ['утконос'],

}

**dictionary.py**

word\_categories\_dictionary = ['скорпион',

'науру',

'улитка',

'стремя',

'осьминог',

'туатара',

'крокодил',

'утконос',

'боливия',

'тайланд',

'чихание',

'сковорода',

'вратарь',

'фуникулёр',

'черепаха',

'бергамот',

'структура',

'бассейн',

]

# **ПРИЛОЖЕНИЕ Б**

**ТЕХНИЧЕСКОЕ ЗАДАНИЕ**

# **ПРИЛОЖЕНИЕ В**

**ПРОГРАММА И МЕТОДИКА ИСПЫТАНИЙ**

# **ПРИЛОЖЕНИЕ Г**

**РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ**