МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение   
высшего образования

«КРЫМСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ им. В. И. ВЕРНАДСКОГО»

ФИЗИКО-ТЕХНИЧЕСКИЙ ИНСТИТУТ

Кафедра компьютерной инженерии и моделирования

**4 В РЯД**

Курсовая работа

по дисциплине «Программирование»

студента 1 курса группы ПИ-б-о-201

Чахмарова Османа Рустемовича

направления подготовки 09.03.04 «Программная инженерия»

|  |  |  |
| --- | --- | --- |
| Научный руководитель  старший преподаватель кафедры компьютерной инженерии и моделирования | \_\_\_\_\_\_\_\_\_\_\_\_  (оценка)  \_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) | Чабанов В.В. |

Симферополь, 2021

**РЕФЕРАТ**

Чахмаров О.Р. 4 в ряд // Курсовая работа (уровень бакалавриата) по специальности 09.04.03 Программная инженерия / Кафедра компьютерной инженерии и моделирования Физико-технического института Крымского федерального университета им. В. И. Вернадского. – Симферополь, 2021. – 18 с., 3 рис., 1 прил., 7 ист.

*Объект* разработки – игра, управлять которой можно используя свой телефон.

*Цель* разработки – создание рабочей игры с реализацией клиент-серверной системы. Создание сервера с использованием языка С++, создание клиентов с использованием языка Python, HTML, реализация обмена данными между клиентом и сервером посредством GET, POST запросов.

ЯЗЫК ПРОГРАММИРОВАНИЯ, PYTHON, C++, ИГРА, СРЕДА РАЗРАБОТКИ

**О**ГЛАВЛЕНИЕ

[Список сокращений и условных обозначений 4](#_Toc74166963)

[ВВЕДЕНИЕ 5](#_Toc74166964)

[ГЛАВА 1 ПОСТАНОВКА ЗАДАЧИ 6](#_Toc74166965)

[**1.1 Цель проекта** 6](#_Toc74166966)

[**1.2 Существующие аналоги** 6](#_Toc74166967)

[**1.3 Основные отличия от аналогов** 8](#_Toc74166968)

[1.4 Техническое задание 8](#_Toc74166969)

[ГЛАВА 2 ПРОГРАММНАЯ РЕАЛИЗАЦИЯ ПРИЛОЖЕНИЯ 10](#_Toc74166970)

[2.1 Анализ инструментальных средств 10](#_Toc74166971)

[2.2 Описание алгоритмов 10](#_Toc74166972)

[2.4 Описание основных модулей 13](#_Toc74166973)

[ГЛАВА 3 ТЕСТИРОВАНИЕ ПРОГРАММЫ 18](#_Toc74166974)

[**3.1 Тестирование исходного кода** 18](#_Toc74166975)

[**3.2 Тестирование интерфейса пользователя и юзабилити** 19](#_Toc74166976)

[ГЛАВА 4 ПЕРСПЕКТИВЫ ДАЛЬНЕЙШЕГО РАЗВИТИЯ ПРОЕКТА 23](#_Toc74166977)

[**4.1 Перспективы технического развития** 23](#_Toc74166978)

[**4.2 Перспективы монетизации** 23](#_Toc74166979)

[ЗАКЛЮЧЕНИЕ 24](#_Toc74166980)

[ЛИТЕРАТУРА 25](#_Toc74166981)

[**ПРИЛОЖЕНИЕ 1 КОД ОСНОВНЫХ МОДУЛЕЙ ПРОЕКТА** 26](#_Toc74166982)

Список сокращений и условных обозначений

PS игровая консоль PlayStation

HTML язык гипертекстовой разметки

ПК персональный компьютер

ВВЕДЕНИЕ

Наблюдая за игрой брата на консоли PS с геймпадом, мне в голову пришла идея воссоздать похожий процесс взаимодействия с игрой, используя те знания, которые я приобрел во время моего обучения в университете. Ведь не у всех людей есть дома консоль и парочка геймпадов. Вместо них мы будем использовать ПК и смартфоны

Суть проекта заключается в создании игры с удаленным доступом к кнопкам управления.

Цель работы: сделать игру таким образом, чтобы пользователь мог управлять игрой, заходя на сайт и нажимая там на кнопки, которые отправляют данные на сервер. Игровое поле отрисовывается, беря данные с сервера. Для реализации цели понадобилось изучить библиотеку PyGame для создания игрового поля. Так же был изучен HTML для создания сайта.

ГЛАВА 1  
ПОСТАНОВКА ЗАДАЧИ

**1.1 Цель проекта**

Создание рабочего проекта, оттачивание навыков работы с серверами. Получение опыта работы с изображениями. Изучение библиотеки PyGame. Закрепление знаний и навыков по использованию HTML для верстки сайта.

**1.2 Существующие аналоги**

Игру 4 в ряд уже портировали на ПК немало людей:

* <https://logic-games.spb.ru/connect4>

На данном сайте вы можете поиграть в данную игру:

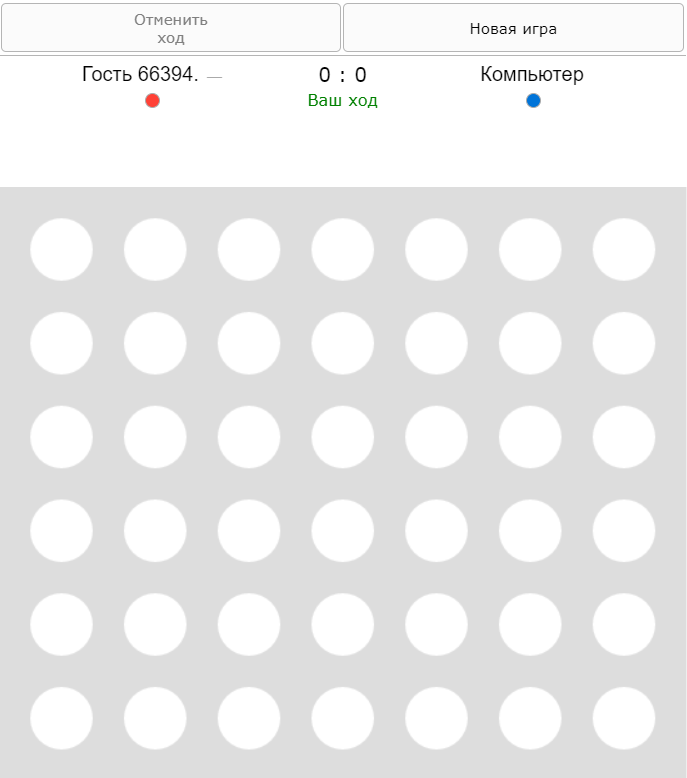


Рисунок 1.1. Интерфейс игры у logic-games.

* <https://vseigru.net/igry-na-dvoikh/26436-igra-chetyre-v-ryad-na-dvoikh.html>

На этом сайте также находится такая же игра, но более проработанная:

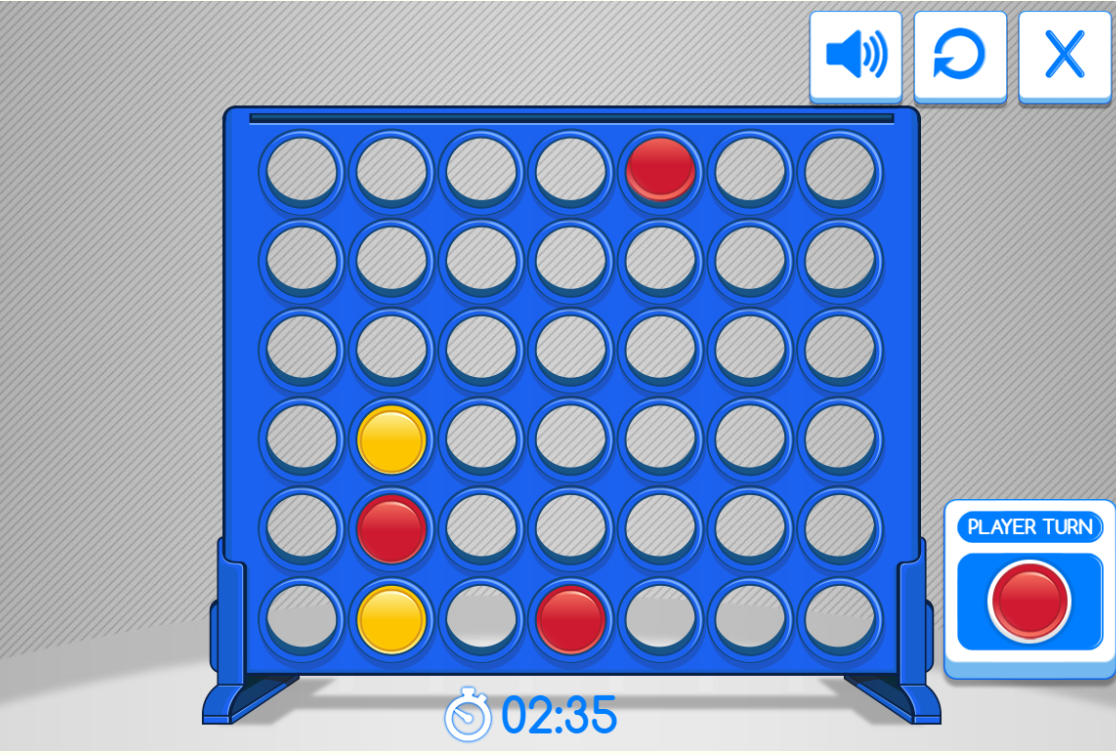


Рисунок 1.2. Интерфейс игры у Vseigru.net.

Так же в Play Market существует большое количество приложений с функционалом очень похожим на мой способ взаимодействия с игрой. С их помощью можно подключиться к ПК и использовать смартфон в качестве мыши или клавиатуры. Самый популярный из данных приложений на данный момент: Remote Mouse.

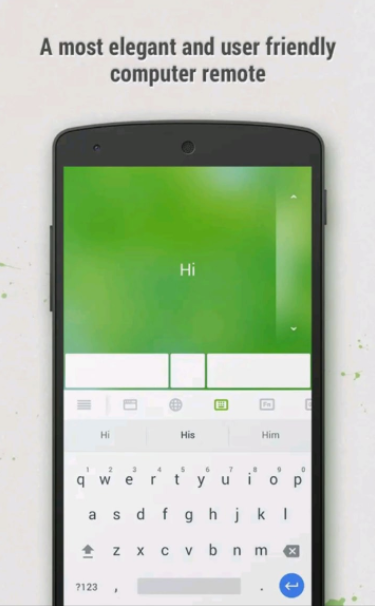


Рисунок 1.3. Интерфейс приложения Remote Mouse.

**1.3 Основные отличия от аналогов**

Мой проект, в отличие от приведенных выше аналогов, сочетает в себе и игровой процесс первых двух, и способ взаимодействия с ПК как у последнего.

## 1.4 Техническое задание

1. Проект состоит из нескольких частей:
   1. Клиент (Python) с запущенной игрой;
   2. Сайт, где вы сможете зайти в свою комнату для игры или посмотреть рейтинг игроков;
   3. Сервер (C++), соединяющий клиент с сайтом и проводящий все расчеты;
2. Что из себя представляет клиент:
   1. Отрисовка всех элементов игрового поля.
   2. Данные берутся с сервера.
3. Принцип работы сайта:
   1. При входе на сайт нас встречает страница, где требуется выбрать свою комнату.
   2. После появляются все элементы управления игрой, доступные для игрока.
   3. Сайт считывает ваш ход и отправляет данные серверу.
4. Принцип работы сервера:
   1. Получение данных о игроках и их действиях с сайта.
   2. Расчет всех событий и отправка данных для отрисовки игрового поля клиенту.

5. Требования к информационной и программной совместимости

Программная система должна функционировать под операционной системой Microsoft Windows 8, 10.

Язык пользовательского интерфейса программы — русский.

ГЛАВА 2  
ПРОГРАММНАЯ РЕАЛИЗАЦИЯ ПРИЛОЖЕНИЯ

## 2.1 Анализ инструментальных средств

Вторым клиентом у меня является сайт. Пользователю куда удобнее зайти на сайт, чем скачивать приложение для игры. Тем более это легче сделать. Для создания сайта мне понадобилось изучить HTML.

Сервер у меня на языке C++, так как в процессе обучения в университете мне уже приходилось работать с ним.

Интегрированная среда разработки Visual Studio 2019 была выбрана в связи со знакомым интерфейсом и возможностями по разработке программ на языке С++.

Интегрированная среда разработки JetBrains PyCharm была выбрана из-за ее приспособленности к разработке программ на языке Python.

Для отображения игрового поля я использовал язык программирования Python с библиотекой PyGame. Даже потратив время на изучение нового для меня набора модулей, я не пожалел, что отказался от Tkinter. На PyGame разработка игрового поля была более приятной и быстрой.

Для верстки сайта я использовал текстовый редактор Sublime text 3.

C++ и Python – как изучаемые языки программирования.

Библиотеки nlohmann/json и cpphttplib из-за их наличия в программе обучения и простоты работы.

## 2.2 Описание алгоритмов

1. Описание алгоритмов клиента
   1. Получение данных с сервера

Чтобы отобразить игровое поле, клиент делает get-запрос на сервер, который отправляет клиенту json-файл.

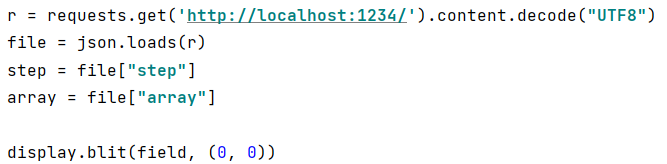


Рисунок 2.1. Алгоритм получения данных с сервера.

* 1. Отрисовка игрового поля.

Для этого был создан массив, который заполнен числами от 0 до 2.

0 – пустая ячейка

1 – красная фишка в ячейке

2 – желтая фишка в ячейке

Смотря на значение элемента массива, программа отображает нужную фишку если она там стоит.

Также, вместе с массивом сервер предает информацию о состоянии игры, то есть, кто сейчас ходит или кто победил.

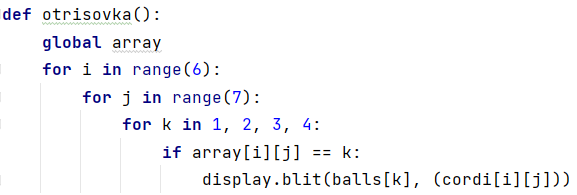


Рисунок 2.2. Алгоритм отображения фишек.

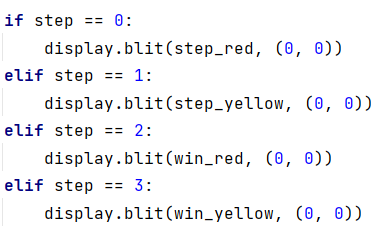


Рисунок 2.3. Алгоритм отображения стадии игры.

1. Описание алгоритмов сервера
   1. Создание json-файла со стартовыми данными.

При запуске сервера создается файл, прочитав который, клиент отобразит пустое поле, ходит первым красный игрок.

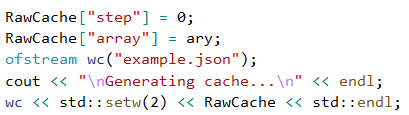


Рисунок 2.4. Алгоритм создания json-файла.

* 1. Передача данных клиенту для отображения игрового поля.

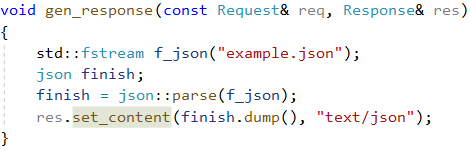


Рисунок 2.5. Алгоритм передачи данных клиенту.

* 1. Открытие страниц в браузере, при соответственных get-запросах

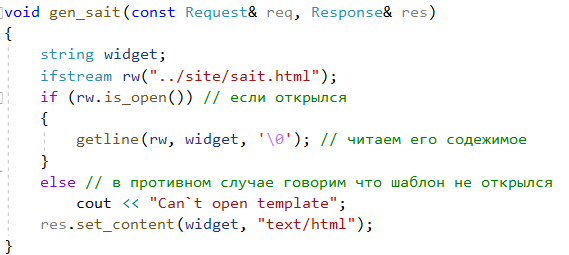


Рисунок 2.6. Алгоритм для открытия страниц в браузере.

* 1. Изменение json-файла при нажатии на игровые кнопки



Рисунок 2.7. Алгоритм изменения json-файла при нажатии на игровые кнопки.

1. Описание работы сайта

Сайт имеет форму, где каждая кнопка названа универсальным именем. При нажатии на кнопку, сайт отправляет post-запрос серверу, с информацией о кнопке. Сервер, считывая эту информацию, проводит расчеты и изменяет json-файл.

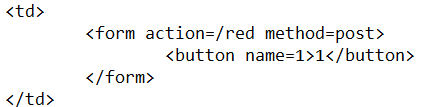


Рисунок 2.8. Отправка данных серверу через форму на сайте.

## 2.4 Описание основных модулей

1. Сервер

После запуска сервера в терминале выходят три сообщения, каждый из которых говорит о успешном запуске отдельных процессов.

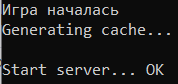


Рисунок 2.9. Сообщения в консоли.

1. Клиент

После запуска клиента открывается окно с игровым полем. Оно обновляется каждую секунду.

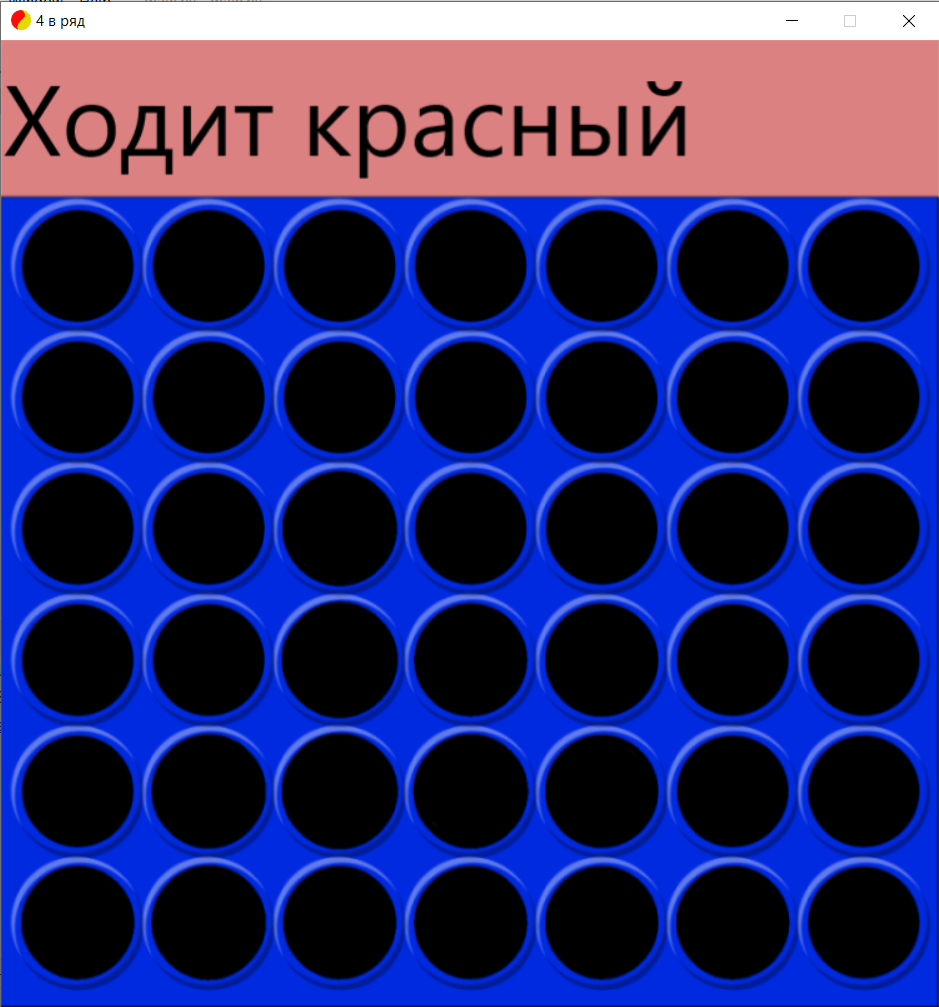


Рисунок 2.10. Игровое поле.

1. Сайт

Зайдя в браузере по ссылке localhost:1234/sait можно будет увидеть сайт,

Где вас просят выбрать цвет. Выбрав комнату, появляются 7 кнопок, с помощью которых и происходит взаимодействие с игрой.

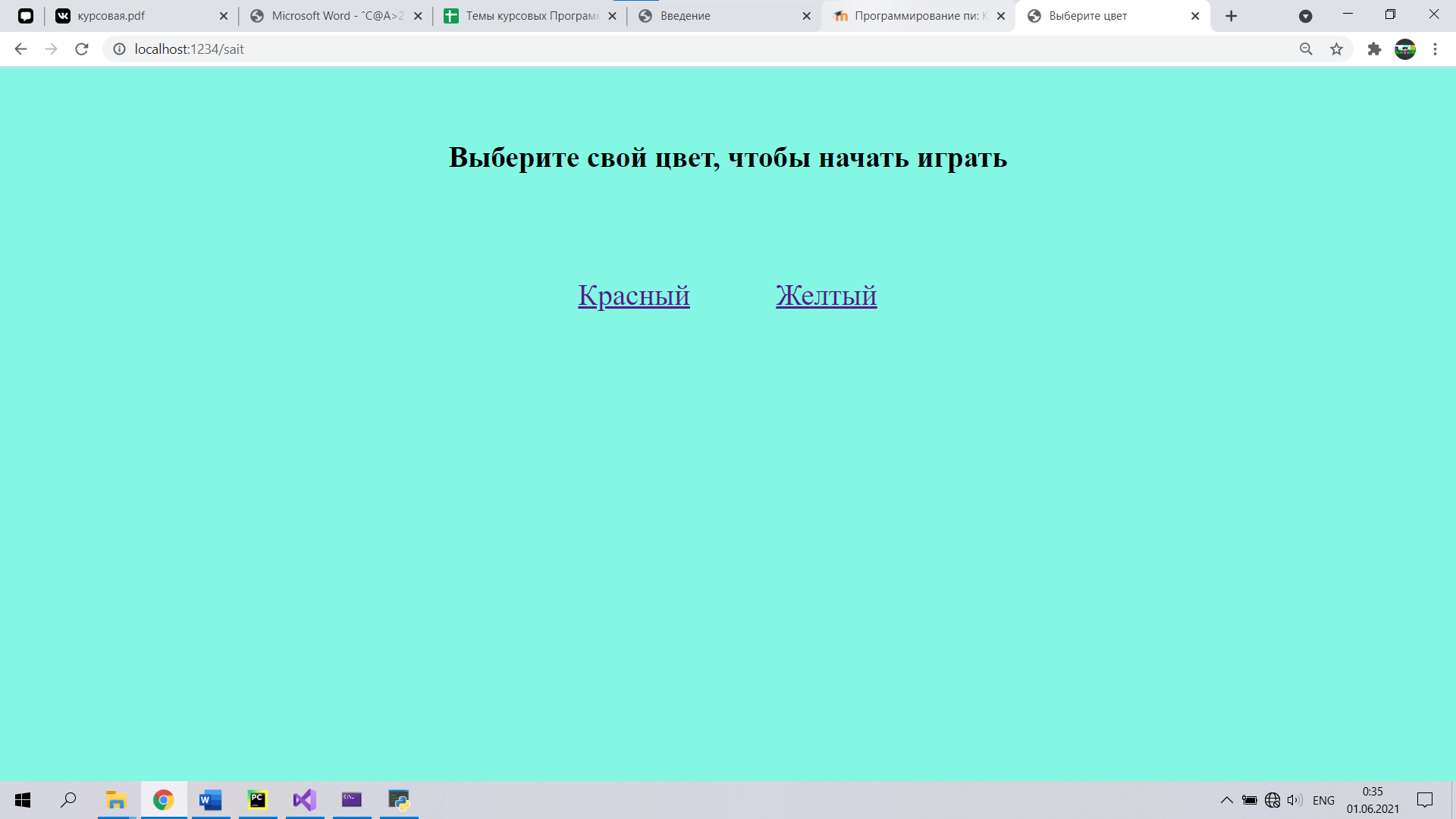


Рисунок 2.11. Сайт для выбора комнаты.

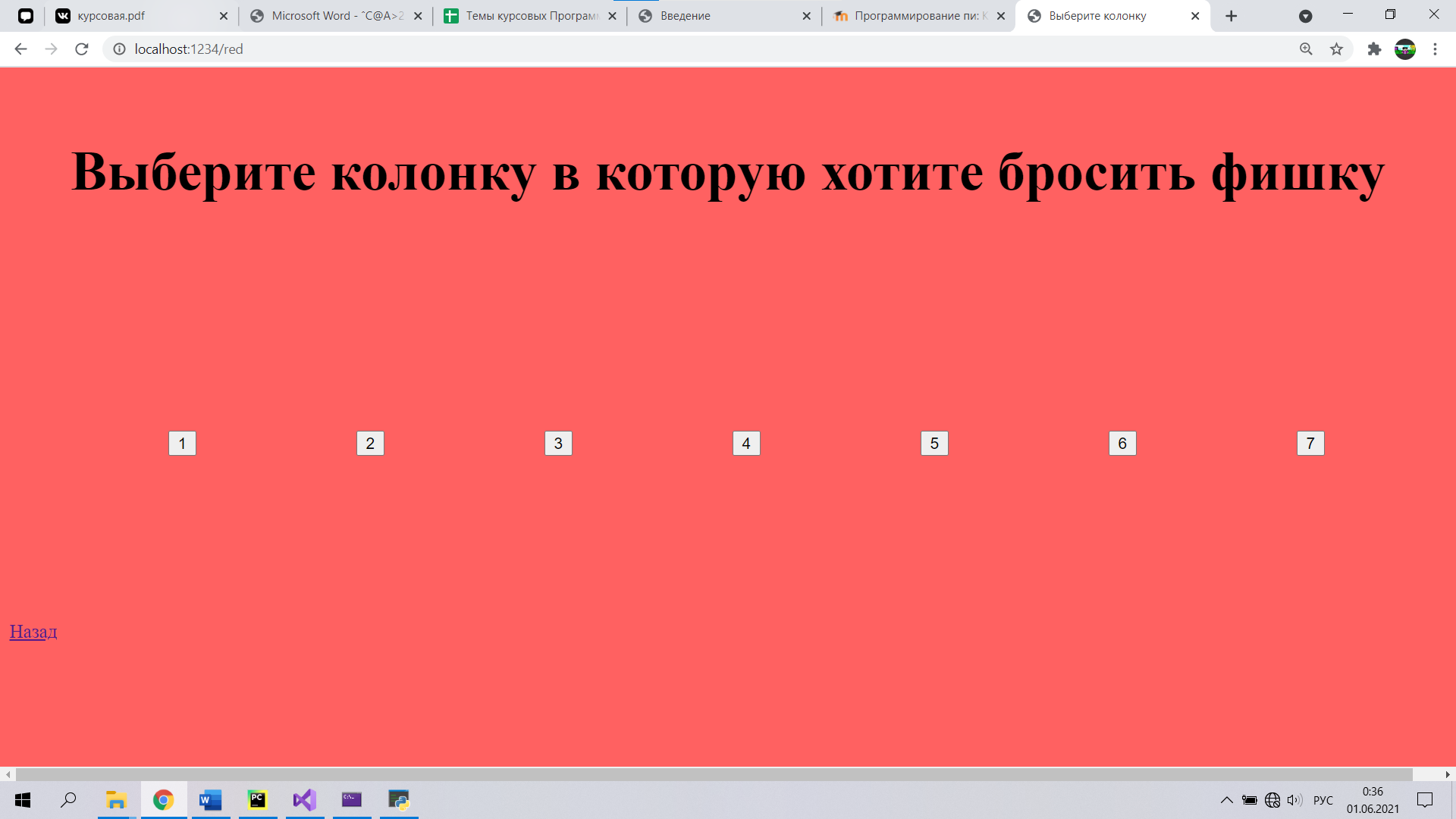


Рисунок 2.12. Сайт для управления красными фишками.

1. Игровой процесс

Нажимая на кнопки клиент будет менять игровое поле, беря данные из сервера. Например: нажав на кнопку 1, играя красными фишками. Клиент отобразит ее в левом нижнем углу.



Рисунок 2.13. Игровое поле с красной фишкой.

1. Конец игры

Игрок, собрав 4 фишки в ряд по диагонали или вертикали или горизонтали, является победителем. Клиент покажет, кто именно выиграл.



Рисунок 2.14. Игровое поле после победы одного из игроков.

ГЛАВА 3  
ТЕСТИРОВАНИЕ ПРОГРАММЫ

**3.1 Тестирование исходного кода**

Для клиента

При статическом анализе кода грубых ошибок замечено не было. При выполнении Debug серьезных ошибок, вызывающих сбой программы, обнаружено не было.



Рисунок 3.1. Рабочий клиент.

Для сервера

При статическом анализе кода грубых ошибок замечено не было. Однако было обнаружено отсутствие компонентов httplib которое вызывает ряд ошибок. Однако если их добавить вручную, то все становится на свои места и сервер запускается корректно.

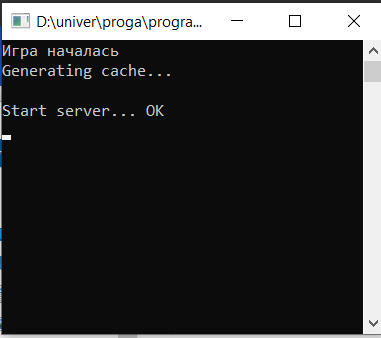


Рисунок 3.2. Рабочий сервер.

**3.2 Тестирование интерфейса пользователя и юзабилити**

Проект делится на три части: сервер, клиент и сайт. Для каждой части было выделено несколько тестов, состоящих из пары задач.

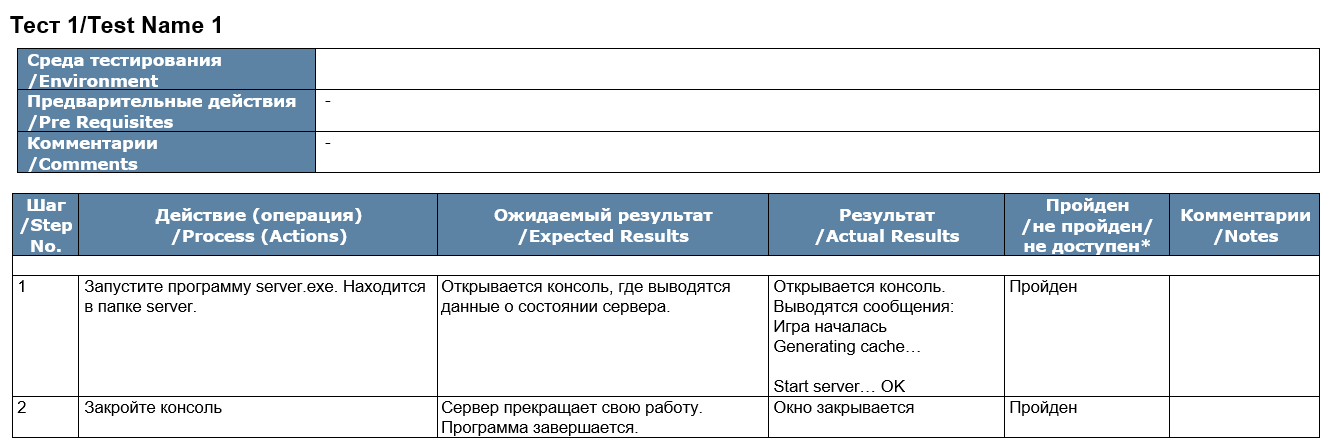


Рисунок 3.3. Тест 1: Запуск и завершение работы сервера.

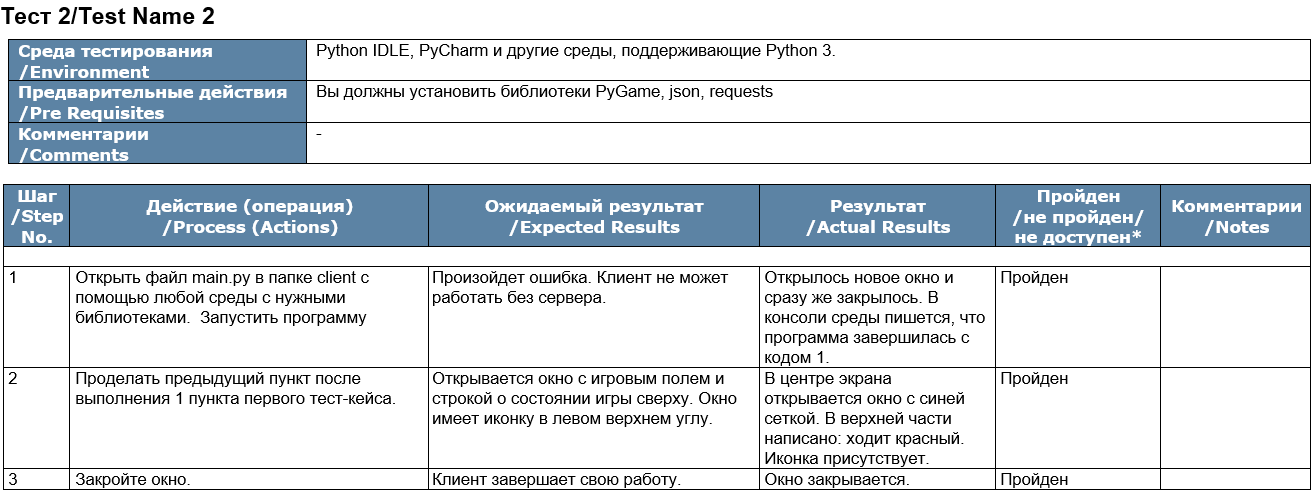


Рисунок 3.4. Тест 2: Запуск и завершение работы клиента с игровым полем.



Рисунок 3.5. Тест 3: Проверка работоспособности сайта.

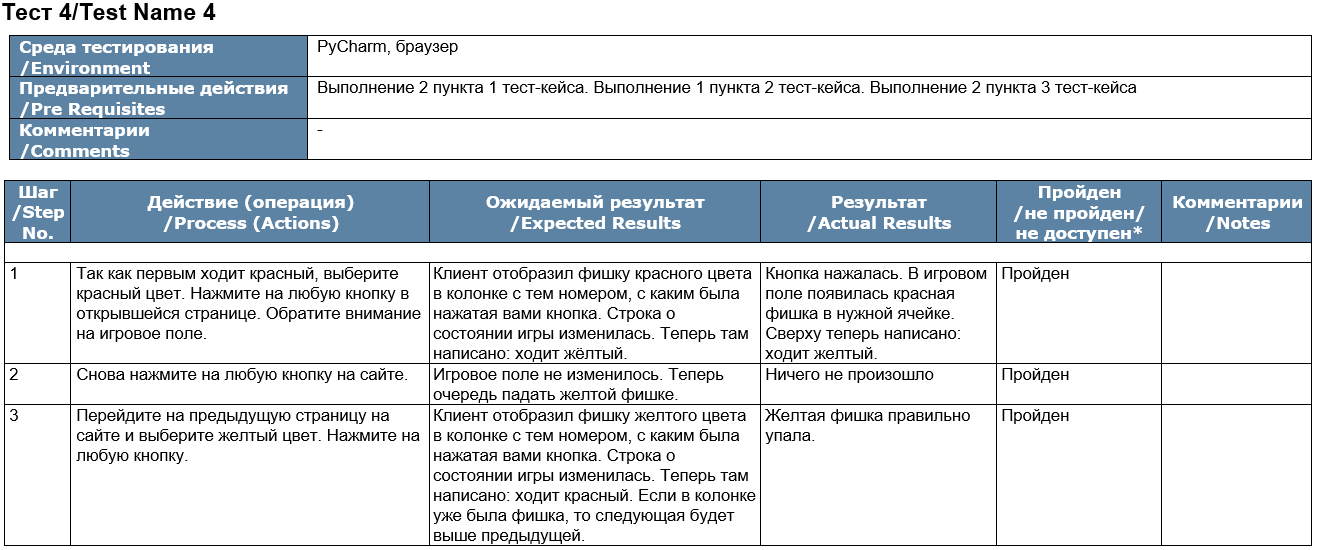


Рисунок 3.6. Тест 4: Проверка корректного начала игры, перехода хода к другому игроку.

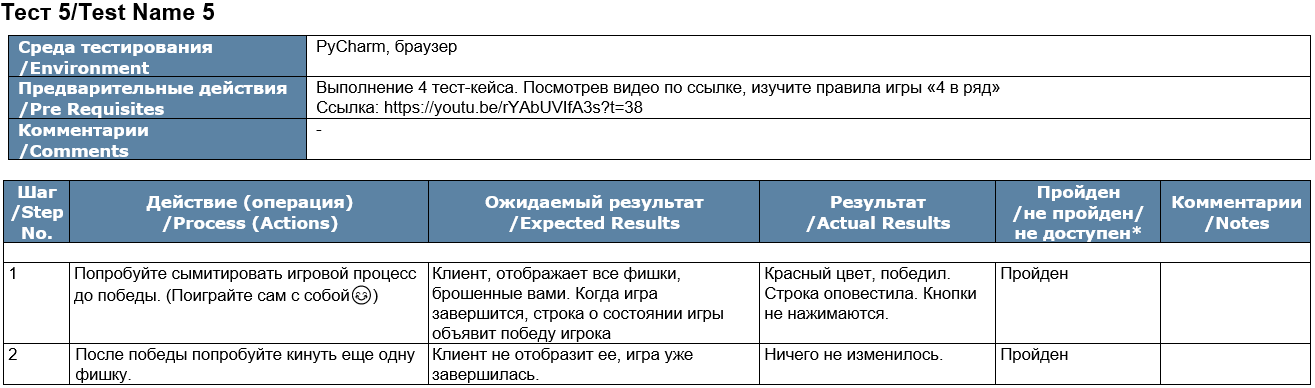


Рисунок 3.7. Тест 5: Проверка игрового процесса, корректного завершения игры.

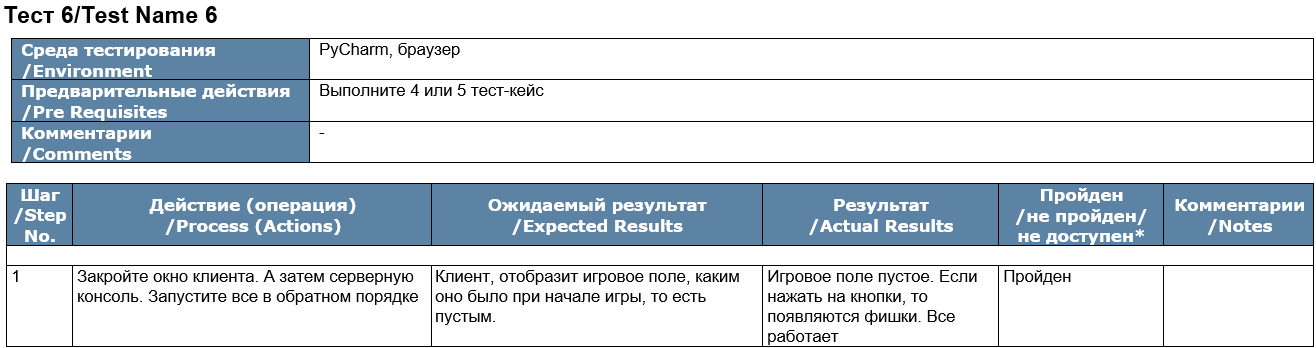


Рисунок 3.8. Тест 5: Перезапуск программы.

ГЛАВА 4  
ПЕРСПЕКТИВЫ ДАЛЬНЕЙШЕГО РАЗВИТИЯ ПРОЕКТА

**4.1 Перспективы технического развития**

Остается еще большое количество вещей, которые не были реализованы в данном проекте. В качестве примера можно привести систему рейтинга, которую в начале разработки планировалось добавить.

**4.2 Перспективы монетизации**

Так как основной целью проекта было получение знаний и навыков для работы в этой отрасли в будущем, смотреть на этот проект с точки зрения монетизации бессмысленно: программа не закончена, хоть и выполняет почти все свои задачи. Внешний вид не доработан.

ЗАКЛЮЧЕНИЕ

В настоящее время мало кто еще додумался до создания такого способа взаимодействия с игрой. Поняв, что никто еще не делал подобное, я решил проверить себя в неизведанной сфере. Итогом выполненной курсовой работы является продукт, позволяющий взаимодействовать с игрой на ПК с помощью своих смартфонов.

Во время выполнения данного проекта был получен новый опыт в разработке крупных проектов, распределении времени и выставления приоритетов с учетом дедлайнов. Так же был получен опыт в поиске, выборе и установке библиотек, в особенности на языке С++. Был получен опыт в работе с PyGame, HTML

Некоторые части продукта все еще нуждаются в полировке, однако весь функционал в программе является рабочим и почти полностью соответствует техническому заданию.

ЛИТЕРАТУРА

1. ГОСТ 19.002-80 Схемы алгоритмов и программ. Правила выполнения [Текст] – Введ. с 01.07. 1981 г. М.: Изд-во стандартов, 1981. – 9 с.
2. ГОСТ 19.003-80 Схемы алгоритмов и программ. Обозначение условные графические [Текст] – Введ. с 01.07. 1981 г. М.: Изд-во стандартов, 1981. – 9 с.
3. Оформление выпускной квалификационной работы на соискание квалификационного уровня «Магистр» («Бакалавр»): методические рекомендации. / сост. Бержанский В.Н., Дзедолик И.В., Полулях С.Н. – Симферополь: КФУ им. В.И.Вернадского, 2017. – 31 с.
4. Любанович Б. Простой Python. Современный стиль программирования / Б. Любанович. СПб.: Питер, 2019. — 194 с.
5. Страуструп Б. Программирование. Принципы и практика с использованием C++ - 2 изд. / Б. Страуструп. М.: Вильямс, 2018. — С. 254.
6. Документация по языку С++ [электронный ресурс] – режим доступа: https://ru.cppreference.com/w/, дата обращения: 10.04.2021;
7. Хабр [электронный ресурс] – режим доступа: https://www.habr.com/, дата обращения: 10.04.2021;

**ПРИЛОЖЕНИЕ 1  
КОД ОСНОВНЫХ МОДУЛЕЙ ПРОЕКТА**

Файл server.cpp

#include <iostream>

#include <string>

#include <iomanip>

#include <cpp\_httplib/httplib.h>

#include <nlohmann/json.hpp>

using std::cout;

using std::string;

using std::ofstream;

using std::ifstream;

using namespace httplib;

using json = nlohmann::json;

using std::endl;

json config;

ofstream logger("log.txt");

const int x = 6;

const int y = 7;

int ary[x][y] =

{

{0, 0, 0, 0, 0, 0, 0},

{0, 0, 0, 0, 0, 0, 0},

{0, 0, 0, 0, 0, 0, 0},

{0, 0, 0, 0, 0, 0, 0},

{0, 0, 0, 0, 0, 0, 0},

{0, 0, 0, 0, 0, 0, 0}

};

int end = 0;

int step = 1;

int a;

bool udod = 0;

bool ending = 0;

json RawCache;

void rascheti(int a)

{

if (step == 1)

{

std::cout << "\nХодит красный\n";

if (a > 7)

return;

}

else

{

std::cout << "\nХодит желтый\n";

if (a < 8)

return;

else

a = a - 7;

}

a = a - 1;

udod = 0;

if ((a > 6) or (a < 0))

udod = 1;

if (udod == 1)

return;

for (int i = 1; i < 7; i++) // Падение фишки

{

if (ary[6 - i][a] == 0)

{

ary[6 - i][a] = step;

break;

}

if (i == 6)

udod = 1;

}

if (udod == 1)

return;

for (int i = 0; i < 6; i++) // Вычисление победителя по горизонтали

{

for (int j = 0; j < 4; j++)

{

if ((ary[i][j] == step) and (ary[i][j + 1] == step) and (ary[i][j + 2] == step) and (ary[i][j + 3] == step))

ending = 1;

}

}

for (int i = 0; i < 4; i++) // Вычисление победителя по вертикали

{

for (int j = 0; j < 7; j++)

{

if ((ary[i][j] == step) and (ary[i + 1][j] == step) and (ary[i + 2][j] == step) and (ary[i + 3][j] == step))

ending = 1;

}

}

for (int i = 0; i < 4; i++) // Вычисление победителя по диагонали вниз

{

for (int j = 0; j < 4; j++)

{

if ((ary[i][j] == step) and (ary[i + 1][j + 1] == step) and (ary[i + 2][j + 2] == step) and (ary[i + 3][j + 3] == step))

ending = 1;

}

}

for (int i = 3; i < 6; i++) // Вычисление победителя по диагонали вверх

{

for (int j = 0; j < 4; j++)

{

if ((ary[i][j] == step) and (ary[i - 1][j + 1] == step) and (ary[i - 2][j + 2] == step) and (ary[i - 3][j + 3] == step))

ending = 1;

}

}

if (ending == 1) // Объявление победителя

{

if (step == 1)

{

std::cout << "\nВыиграл красный\n";

step = 3;

end = 1;

}

else

{

std::cout << "\nВыиграл желтый\n";

end = 1;

step = 4;

}

}

for (int i = 0; i < 6; i++) //Вывод массива

{

for (int j = 0; j < 7; j++)

{

std::cout << ary[i][j] << " ";

}

std::cout << std::endl;

}

if (step == 1) // Переход хода

{

step = 2;

}

else

{

if (step == 2)

{

step = 1;

}

}

RawCache["step"] = step - 1;

RawCache["array"] = ary;

ofstream wc("example.json");

cout << "Generating cache..." << endl;

wc << std::setw(2) << RawCache << std::endl;

}

void gen\_red(const Request& req, Response& res)

{

string widget;

ifstream rw("../site/red.html");

if (rw.is\_open()) // если открылся

{

getline(rw, widget, '\0'); // читаем его содежимое

}

else // в противном случае говорим что шаблон не открылся

cout << "Can`t open template";

res.set\_content(widget, "text/html");

}

void gen\_yellow(const Request& req, Response& res)

{

string widget;

ifstream rw("../site/yellow.html");

if (rw.is\_open()) // если открылся

{

getline(rw, widget, '\0'); // читаем его содежимое

}

else // в противном случае говорим что шаблон не открылся

cout << "Can`t open template";

res.set\_content(widget, "text/html");

}

void gen\_sait(const Request& req, Response& res)

{

string widget;

ifstream rw("../site/sait.html");

if (rw.is\_open()) // если открылся

{

getline(rw, widget, '\0'); // читаем его содежимое

}

else // в противном случае говорим что шаблон не открылся

cout << "Can`t open template";

res.set\_content(widget, "text/html");

}

void gen\_redpost(const Request& req, Response& res)

{

string widget;

ifstream rw("../site/red.html");

if (rw.is\_open()) // если открылся

{

getline(rw, widget, '\0'); // читаем его содежимое

}

else // в противном случае говорим что шаблон не открылся

cout << "Can`t open template";

string a = req.body.c\_str();

a.erase(a.find("="), 1);

cout << a << std::endl;

int i = std::stoi(a);

rascheti(i);

res.set\_content(widget, "text/html");

}

void gen\_yellowpost(const Request& req, Response& res)

{

string widget;

ifstream rw("../site/yellow.html");

if (rw.is\_open()) // если открылся

{

getline(rw, widget, '\0'); // читаем его содежимое

}

else // в противном случае говорим что шаблон не открылся

cout << "Can`t open template";

string a = req.body.c\_str();

a.erase(a.find("="), 1);

cout << a << std::endl;

int i = std::stoi(a);

rascheti(i);

res.set\_content(widget, "text/html");

}

void gen\_response(const Request& req, Response& res)

{

std::fstream f\_json("example.json");

json finish;

finish = json::parse(f\_json);

res.set\_content(finish.dump(), "text/json");

}

int main()

{

setlocale(LC\_ALL, "rus");

std::cout << "Игра началась";

RawCache["step"] = 0;

RawCache["array"] = ary;

ofstream wc("example.json");

cout << "\nGenerating cache...\n" << endl;

wc << std::setw(2) << RawCache << std::endl;

Server svr; // Создаём сервер (пока-что не запущен)

svr.Get("/red", gen\_red);

svr.Get("/yellow", gen\_yellow);

svr.Get("/sait", gen\_sait);

svr.Get("/", gen\_response);

svr.Post("/red", gen\_redpost);

svr.Post("/yellow", gen\_yellowpost);

std::cout << "Start server... OK\n";

svr.listen("localhost", 1234); // Запускаем сервер на localhost и порту 1234

}

Рисунок П.1. Файл «server.cpp»

Файл client.py:

**import** pygame  
**import** json  
**import** requests  
  
pygame.init()  
  
display\_width = 750  
display\_height = 774  
  
display = pygame.display.set\_mode((display\_width, display\_height))  
pygame.display.set\_caption(**'4 в ряд'**)  
  
icon = pygame.image.load(**'icon.png'**)  
pygame.display.set\_icon(icon)  
  
balls = [**None**, pygame.image.load(**'red.png'**), pygame.image.load(**'yellow.png'**), pygame.image.load(**'red\_strih.png'**),  
 pygame.image.load(**'yellow\_strih.png'**)]  
balls\_options = [90, 80]  
  
step\_red = pygame.image.load(**'step\_red.png'**)  
step\_yellow = pygame.image.load(**'step\_yellow.png'**)  
win\_red = pygame.image.load(**'win\_red.png'**)  
win\_yellow = pygame.image.load(**'win\_yellow.png'**)  
  
clock = pygame.time.Clock()  
  
cordi = [  
 [[15, 137], [121, 137], [226, 137], [331, 137], [436, 137], [541, 137], [645, 137]],  
 [[15, 242], [121, 242], [226, 242], [331, 242], [436, 242], [541, 242], [645, 242]],  
 [[15, 346], [121, 346], [226, 346], [331, 346], [436, 346], [541, 346], [645, 346]],  
 [[15, 451], [121, 451], [226, 451], [331, 451], [436, 451], [541, 451], [645, 451]],  
 [[15, 556], [121, 556], [226, 556], [331, 556], [436, 556], [541, 556], [645, 556]],  
 [[15, 661], [121, 661], [226, 661], [331, 661], [436, 661], [541, 661], [645, 661]]  
]  
  
  
**def** otrisovka():  
 **global** array  
 **for** i **in** range(6):  
 **for** j **in** range(7):  
 **for** k **in** 1, 2, 3, 4:  
 **if** array[i][j] == k:  
 display.blit(balls[k], (cordi[i][j]))  
  
  
**def** run\_game():  
 game = **True** field = pygame.image.load(**'field.png'**)  
  
 **while** game:  
 **global** array  
 **for** event **in** pygame.event.get():  
 **if** event.type == pygame.QUIT:  
 pygame.quit()  
 quit()  
  
 r = requests.get(**'http://localhost:1234/'**).content.decode(**"UTF8"**)  
 file = json.loads(r)  
 step = file[**"step"**]  
 array = file[**"array"**]  
  
 display.blit(field, (0, 0))  
  
 otrisovka()  
  
 **if** step == 0:  
 display.blit(step\_red, (0, 0))  
 **elif** step == 1:  
 display.blit(step\_yellow, (0, 0))  
 **elif** step == 2:  
 display.blit(win\_red, (0, 0))  
 **elif** step == 3:  
 display.blit(win\_yellow, (0, 0))  
  
 pygame.display.update()  
 clock.tick(1)  
  
  
run\_game()

Рисунок П.2. Файл «client.py»