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

ИГРА ПЛАТФОРМА

Курсовая работа
по дисциплине «Программирование»
студента 1 курса группы ИВТ-б-о-201
Роик Владислав Вадимович
направления подготовки 09.03.01 «Информатика и вычислительная техника»

Научный руководитель _____
(оценка)
старший преподаватель кафедры
компьютерной инженерии и моделирования _____ Чабанов В.В.
(подпись, дата)

Симферополь, 2021

РЕФЕРАТ

Игра Платформа – Симферополь: ФТИ КФУ им. В. И. Вернадского, 2020. – 48 с., 23 ил., 6 ист.

Объект разработки – игра «Китайские шашки», сервер игры.

Цель работы – создание рабочего игрового проекта с реализацией клиент-сервер системы. Создание сервера с использованием языка C++, реализация обмена данными приложения с сервером.

Было рассмотрено многочисленное количество вариантов реализации клиент-серверных систем, в ходе которых было принято решение использовать API подход для взаимодействия Python клиента и C++ сервера, в виду легкости развертывания и понимания.

ПРОГРАММИРОВАНИЕ, PYTHON, C++, ИГРА, АРКАНОИД, PYGAME, КЛИЕНТ-СЕРВЕР, БАЗА ДАННЫХ

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Данная курсовая работа является примером реализации задачи игры «Платформа», написанной с применением программирования на языках Python и C++. На экране отображается игровое поле, а также текущая статистика игры.

Цель курсовой работы по дисциплине «Программирование» состоит в закреплении и углублении знаний и навыков, полученных при изучении дисциплины. Курсовая работа представляет собой индивидуальное задание повышенной сложности по разработке программы на изучаемом языке высокого уровня.

Для выполнения курсовой работы потребовалось изучение дополнительных разделов языка, таких как графика и ее использование в Python и C++.

ГЛАВА 1
ПОСТАНОВКА ЗАДАЧИ

1.1 Цель проекта

Создание работающего проекта, получение опыта в разработке проектов, получение опыта в создании API систем. Закрепление навыков использования и внедрения сторонних C++ и Python библиотек, их изучение. Укрепление навыков наладки работы клиент – серверных систем, написание сервера с использованием языка C++.

1.2 Существующие аналоги

- Brick Breaker — популярная классическая аркадная головоломка, в которой шариком нужно разбивать блоки с различными свойствами. Вас ожидает более ста захватывающих уровней, много улучшений и яркая картинка. Также доступен локальный мультиплеер, с возможностью одновременной игры четырех человек на одном экране.
- Super Destronaut DX — классический арканоид с видом сбоку в научно-фантастическом сеттинге. Игра вдохновлена популярными аркадами из 90-х, где вам предстоит сопротивляться вторжению инопланетных захватчиков. Пилотируя космический штурмовик, вам нужно уничтожить как можно больше пришельцев, при этом зарабатывая очки.
- Blockbuster (1987) — классический арканоид, где вам предстоит управлять небольшой платформой, с помощью которой отбивать снаряд. Тот, в свою очередь, разбивает расположенные на противоположной стороне поля кирпичи. У них у всех разный запас прочности. Из каждого кирпича может выпасть какой-либо бонус. Некоторые из них упрощают игру, другие усложняют. Цель игры — очистить игровое поле.
- Enemy:Unknown — аркадный фантастический шутер с видом сверху. Это классическая игра про полет на космическом корабле, с отстрелом инопланетян, в современной обработке.

- INV - научно-фантастический арканоид с видом сверху. Вам предстоит отстреливать инопланетных существ и спасти Землю от захватчиков.

1.3 Основные отличия от аналогов

Главной особенностью данного проекта является дополнительная игра в вопросы, а также наличие рейтинговой таблицы и временных ограничений.

1.4 Техническое задание

Программа должна корректно интерпретировать и соблюдать следующие основные правила игры:

- Игровое поле — прямоугольник с границами.
- Игровой шарик перемещается внутри игрового поля, разбивая блоки и отскакивая от упругих стенок.
- Игрок может передвигать платформу стрелками «вправо» и «влево» в соответствующие стороны.
- В процессе игры игроку предлагается отвечать на вопросы, которые появляются рядом с шариков в определенные временные интервалы. С этими вопросами предлагаются и варианты ответа, перемещение по которым выполняется стрелками «вверх» и «вниз», ответ даётся при помощи клавиши «Enter».
- Сначала игры пользователь вводит свой никнейм.
- Очки платформы и интеллектуальные баллы должны подсчитываться отдельно друг от друга.
- Игроку даётся 6 секунд для ответа на вопрос.
- Интеллектуальные вопросы появляются с перерывом в 6-10 секунд.
- Вопросы создавались на основе блиц опросов и игры «0 или 100».
- На игровом поле игрок может наблюдать за собственным рекордом и своей позицией в турнирной таблице рекордов пользователей игры «Платформа».

- Турнирная таблица должна постоянно обновлять данные игр пользователей.
- Программа должна иметь звуковое сопровождение.

ГЛАВА 2
ПРОГРАММНАЯ РЕАЛИЗАЦИЯ ПРИЛОЖЕНИЯ

2.1 Анализ инструментальных средств

- Среда Microsoft Visual Studio 2019 – была выбрана в пользу знакомого интерфейса и удобных функций интегрирования с сервисом GitHub;
- C++ – как изучаемый язык программирования;
- SQL – декларативный язык программирования, применяемый для создания, модификации и управления данными в реляционной базе данных, управляемой соответствующей системой управления базами данных, для, соответственно, создания, модификации и управления данными в базе данных.
- Python – как изучаемый язык программирования.
- Pygame – набор модулей (библиотек) языка программирования Python, предназначенный для написания компьютерных игр и мультимедиа-приложений. Pygame базируется на мультимедийной библиотеке SDL. Был выбран в пользу ее доступности и большого количества возможностей.
- Библиотека JSON for Modern C++ – header-only библиотека для работы с JSON информацией на стороне приложения.

2.2 Описание алгоритмов

Основной цикл игры запускается и обновляет экран через фиксированные промежутки времени. Это ваша частота кадров, и она определяет, насколько плавными будут изображения. Обычно игры обновляют экран от 30 до 60 раз в секунду. Если вы будете двигаться медленнее, объекты на экране будут дергаться.

Внутри основного цикла есть три основных действия: обработка событий, обновление состояния игры и рисование текущего состояния экрана.

События в игре состоят из всего, что происходит вне контроля игрового кода, но имеет отношение к работе игры. Например, если в Breakout игрок нажимает клавишу со стрелкой влево, игре необходимо переместить ракетку влево.

Ядром каждой игры является ее состояние: то, что она отслеживает и рисует на экране. В Breakout состояние включает в себя расположение всех кирпичей, положение и скорость мяча, положение ракетки, а также жизни и счет.

В Платформе мяч отскакивает от объектов и имеет очень грубую систему физики твердого тела (если ее можно так назвать).

Класс Game - это ядро игры(Рис.1). Он запускает основной цикл. В нем много полезного функционала.

__init__(). Метод инициализирует сам Pygame, систему шрифта, и аудио микшера. Причина, по которой вам нужно сделать три разных вызова, заключается в том, что не все игры Pygame используют все компоненты, поэтому вы контролируете, какие подсистемы вы используете, и инициализируете только те, с их конкретными параметрами. Он создает фоновое изображение, основную поверхность (где все нарисовано) и игровые часы с правильной частотой кадров.

Член self.objects сохранит все игровые объекты, которые необходимо отрисовать и обновить. Различные обработчики управляют списками функции обработчика, которая должна вызываться при возникновении определенных событий.

```
01  import pygame
02  import sys
03
04  from collections import defaultdict
05
06
07  class Game:
08      def __init__(self,
09                   caption,
10                   width,
11                   height,
12                   back_image_filename,
13                   frame_rate):
14          self.background_image = \
15              pygame.image.load(back_image_filename)
16          self.frame_rate = frame_rate
17          self.game_over = False
18          self.objects = []
19          pygame.mixer.pre_init(44100, 16, 2, 4096)
20          pygame.init()
21 год       pygame.font.init()
22          self.surface = pygame.display.set_mode((width, height))
23          pygame.display.set_caption(caption)
24          self.clock = pygame.time.Clock()
25          self.keydown_handlers = defaultdict(list)
26          self.keyup_handlers = defaultdict(list)
27          self.mouse_handlers = []
```

Рисунок 1. Класс Game

Update() и draw() методы очень просты. Они просто перебирают все управляемые игровые объекты и вызывают их соответствующие методы. Если два игровых объекта перекрываются, порядок в списке объектов определяет, какой объект будет отображаться первым, а другой будет частично или полностью покрывать его(Рис.2).

```
1  def update(self):
2      for o in self.objects:
3          o.update()
4
5  def draw(self):
6      for o in self.objects:
7          o.draw(self.surface)
```

Рисунок 2. Используемые функции

Handle_events()(Рис.3). Метод прослушивает события, генерируемые Pygame, как ключ и мышь событий. Для каждого события он вызывает все функции-обработчики, которые зарегистрированы для обработки этого типа события.

```
def handle_events(self):
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            sys.exit()
        elif event.type == pygame.KEYDOWN:
            for handler in self.keydown_handlers[event.key]:
                handler(event.key)
        elif event.type == pygame.KEYUP:
            for handler in self.keydown_handlers[event.key]:
                handler(event.key)
        elif event.type in (pygame.MOUSEBUTTONDOWN,
                            pygame.MOUSEBUTTONUP,
                            pygame.MOUSEMOTION):
            for handler in self.mouse_handlers:
                handler(event.type, event.pos)
```

Рисунок 3. Функция handle_events().

Наконец, run() метод запускает основной цикл. Он выполняется, пока game_overчлен не станет True. В каждой итерации, она делает фоновое изображение и вызывает в порядке handle_events(), update() и draw()методах(Рис.4).

Затем он обновляет дисплей, который фактически обновляет физический дисплей со всем содержимым, которое было отображено во время этой итерации. И последнее, но не менее важное: он вызывает clock.tick()метод, чтобы контролировать, когда будет вызвана следующая итерация.

```
def run(self):
    while not self.game_over:
        self.surface.blit(self.background_image, (0, 0))

        self.handle_events()
        self.update()
        self.draw()

        pygame.display.update()
        self.clock.tick(self.frame_rate)
```

Рисунок 4. Функция run()

В данном проекте сервер играет роль посредника между базой данных, содержащей рекорды игроков, и самим игроком. Для общения клиента с сервером, клиент отправляет post запрос серверу (Рис.5).

```
if int(self.lastScore) !=  int(self.intScores)+int(self.score):
    print("Запрос отправлен")
    r = requests.post('http://localhost:3000/rating', data={'SCORE': int(self.intScores)+int(self.score),'na
    jsona = r.text
    self.record = json.loads(jsona)["record_score"]
    self.position = json.loads(jsona)["position"]
    self.lastScore = int(self.intScores)+int(self.score)
```

Рисунок 5. POST запрос на сервер для получения своей позиции и своего рекорда

На рисунке ниже(Рис.6) показан post запрос на сервер, который выполняется по окончании игры, цель этого запроса сохранить свой рекорд в базе данных.

```
self.draw()
message.draw(self.surface, centralized)
if end == 1:
    r = requests.post('http://localhost:3000/set_rating', data={'SCORE': int(self.intScores)+int(se
    jsona = r.text
```

Рисунок 6. POST запрос для сохранения своего рекорда

2.3 Описание структур данных

Играм необходимо управлять большим объемом информации и выполнять аналогичные операции со многими объектами. Платформа - это мини-игра, но пытаться управлять всем в одном файле было бы непросто. Вместо этого я решил создать файловую структуру и архитектуру(Рис.7), которые подходили бы для гораздо более крупных игр.

```
├── Pipfile
├── Pipfile.lock
├── README.md
├── ball.py
├── breakout.py
├── brick.py
├── button.py
├── colors.py
├── config.py
├── game.py
├── game_object.py
├── images
│   └── background.jpg
├── paddle.py
├── sound_effects
│   ├── brick_hit.wav
│   ├── effect_done.wav
│   ├── level_complete.wav
│   └── paddle_hit.wav
└── text_object.py
```

Рисунок 7. Каталог и файловая структура

Pipfile и Pipfile.lock - это современный способ управления зависимостями в Python. Каталог изображений содержит изображения, используемые игрой (только фоновое изображение в этом воплощении), а каталог sound_effects содержит короткие аудиоклипы, используемые в качестве (как вы уже догадались) звуковых эффектов.

Файлы ball.py, paddle.py и brick.py содержат код, специфичный для каждого из этих объектов Breakout. Я расскажу о них более подробно позже в этой серии. Файл text_object.py содержит код для отображения текста на экране, а файл background.py содержит игровую логику, специфичную для Breakout.

GameObject представляет собой визуальный объект, который знает, как отображать себя, поддерживать свои границы и перемещаться(Рис.8).

```python
from pygame.rect import Rect


class GameObject:
    def __init__(self, x, y, w, h, speed=(0,0)):
        self.bounds = Rect(x, y, w, h)
        self.speed = speed

    @property
    def left(self):
        return self.bounds.left

    @property
    def right(self):
        return self.bounds.right

    @property
    def top(self):
        return self.bounds.top

    @property
    def bottom(self):
        return self.bounds.bottom

    @property
    def width(self):
        return self.bounds.width

    @property
    def height(self):
        return self.bounds.height

    @property
    def center(self):
        return self.bounds.center

    @property
    def centerx(self):
        return self.bounds.centerx

    @property
    def centery(self):
        return self.bounds.centery

    def draw(self, surface):
        pass

    def move(self, dx, dy):
        self.bounds = self.bounds.move(dx, dy)

    def update(self):
        if self.speed == [0, 0]:
            return

        self.move(*self.speed)
```

Рисунок 8. Класс GameObject

GameObject предназначен для использования в качестве базового класса для других объектов. Он напрямую предоставляет множество свойств своего прямоугольника self.bounds и в своем update()методе перемещает объект в соответствии с его текущей скоростью. Он ничего не делает в своем draw()методе, который должен быть переопределен подклассами.

Сервер на C++ принимает HTTP запроса используя библиотеку cpp-httplib. Ниже на рисунке(Рис.9) показана часть кода, запускающая сервер.

```cpp
Server svr;
svr.Post("/rating", gen_response);
svr.Post("/set_rating", gen_response_set);
std::cout << "Start server... OK\n";
svr.listen("localhost", 3000);
```

Рисунок 9. Часть кода, отвечаюшая за запуск сервера

Перед тем как запускать сам сервер, идет проверка на наличие базы данных, в случае если база данных отсутствует, создается новая (Рис.10).

```
sqlite3* DB;
std::string sql = "CREATE TABLE PERSON("
    "LOGIN          TEXT    NOT NULL, "
    "RECORD         TEXT     NOT NULL );";
int exit = 0;
exit = sqlite3_open("persons.db", &DB);
char* messaggeError;
exit = sqlite3_exec(DB, sql.c_str(), NULL, 0, &messaggeError);
sqlite3_close(DB);
```

Рисунок 10. Часть кода, ответственная за создание базы данных

Когда сервер получил ожидаемый запрос, вызывается соответствующая функция (Рис. 11). Внутри функции на рисунке ниже, идет получение данных переданных post запросом, а после подключение к базе данных и выполнение команды, находящейся в строковой переменной stmt. Эта команда означает получение всех данных из таблицы PERSON. Метод библиотеки sqlite3 "sqlite3_exec" принимает своим аргументом функцию select_callback и адрес records.

```
58
59  void gen_response(const Request& req, Response& res) {
60      string score = req.get_param_value("SCORE");
61      string name = req.get_param_value("name");
62      int position=1;
63      int record_score=-1;
64
65      sqlite3* DB;
66      int exit = sqlite3_open("persons.db", &DB);
67      char* errmsg = 0;
68      string stmt = "SELECT * FROM PERSON";
69      Records records;
70      int ret = sqlite3_exec(DB, stmt.c_str(), select_callback, &records, &errmsg);
71
```

Рисунок 11. Функция gen_response

2.4 Описание основных модулей

Функция select_callback в C++ на рисунке ниже принимает первым параметром указатель p_data, этот указатель будет хранить адрес records. Таким образом функция select_callback, нужна для сохранения полученных из базы данных(Рис.12).

```
int select_callback(void* p_data, int num_fields, char** p_fields, char** p_col_names)
{
    Records* records = static_cast<Records*>(p_data);
    try {
        records->emplace_back(p_fields, p_fields + num_fields);
    }
    catch (...) {
        return 1;
    }
    return 0;
}
```

Рисунок 12. Функция select_callback

Когда сервер получил готовые данные, он их обрабатывает и возвращает полученные значения клиенту (Рис. 13).

```
json j = { {"position",position},{"record_score",record_score} };
sqlite3_close(DB);
res.set_content(j.dump(), "text/json; charset=UTF-8");// Вывод позиции и рекорда
```

Рисунок 13. Формирование ответа сервера клиенту

На рисунке ниже показаны части кода, которые ответственны за добавление или обновление позиции в базе данных (Рис.14).

```
if (is_available) // update
{
    cout << score << last_record;
    if (stoi(score) > last_record)
    {
        std::stringstream UpdateQuery;
        UpdateQuery << "UPDATE PERSON "
                "SET RECORD = '" << score
            << "' WHERE LOGIN = '" << name<< "'";
        sqlite3_stmt* UpdateStmt;
        sqlite3_prepare(DB, UpdateQuery.str().c_str(), UpdateQuery.str().size(), &UpdateStmt, NULL);
        if (sqlite3_step(UpdateStmt) != SQLITE_DONE) cout << "Ошибка" << endl;
    }
}
else { // insert

    std::stringstream insertQuery;
    insertQuery << "INSERT INTO PERSON (LOGIN, RECORD)"
        " VALUES ('" << name
        << "', " << score << ")";
    sqlite3_stmt* insertStmt;
    sqlite3_prepare(DB, insertQuery.str().c_str(), insertQuery.str().size(), &insertStmt, NULL);
    if (sqlite3_step(insertStmt) != SQLITE_DONE) cout << "Ошибка" << endl;

}
```

Рисунок 14. Часть кода из функции gen_response_set

ГЛАВА 3 ТЕСТИРОВАНИЕ ПРОГРАММЫ

3.1 Тестирование проекта

В моем проекте 5 модулей: главный модуль, модуль уведомлений, модуль поиска, модуль обновлений, модуль очистки. Тестами покрыты все модули.

3.1.1 Тестирование кнопок

Кнопки "Play" и "Quit" работают без проблем (Рис.15).

Рисунок 15. Кнопки "Play" и "Quit"

3.1.2 Тестирование платформы

Платформа успешно выполняет свою роль отбивания шарика(Рис.16).

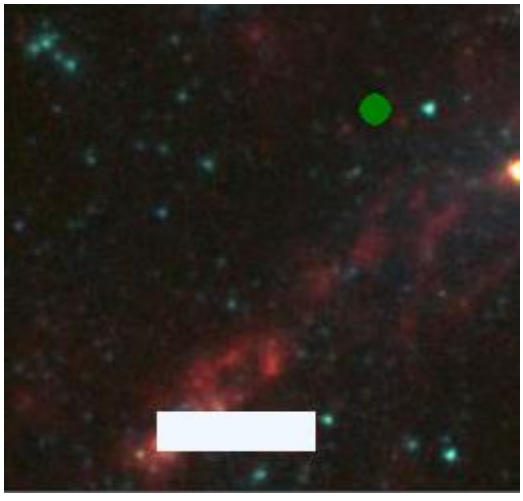

Рисунок 16. Платформа

3.1.3 Тестирование срабатывания кнопок

Платформа, как и задумывалось может передвигаться в стороны(Рис.17).

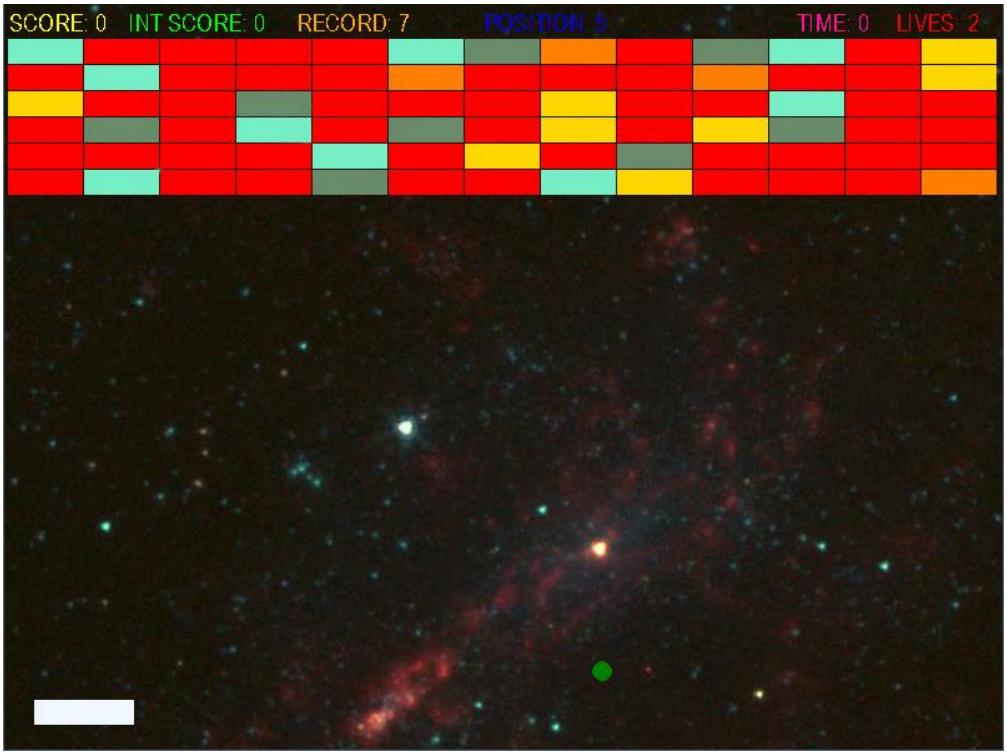

Рисунок 17. Передвижение платформы

3.1.4 Тестирование поля для ввода ника

Поле для ввода никнейма работает исправно(Рис.18).

Рисунок 18. Поле для ввода никнейма

3.1.5 Тестирование счетчиков

Как видно на рисунке ниже, все счетчики выполняют свое предназначение(Рис.19).

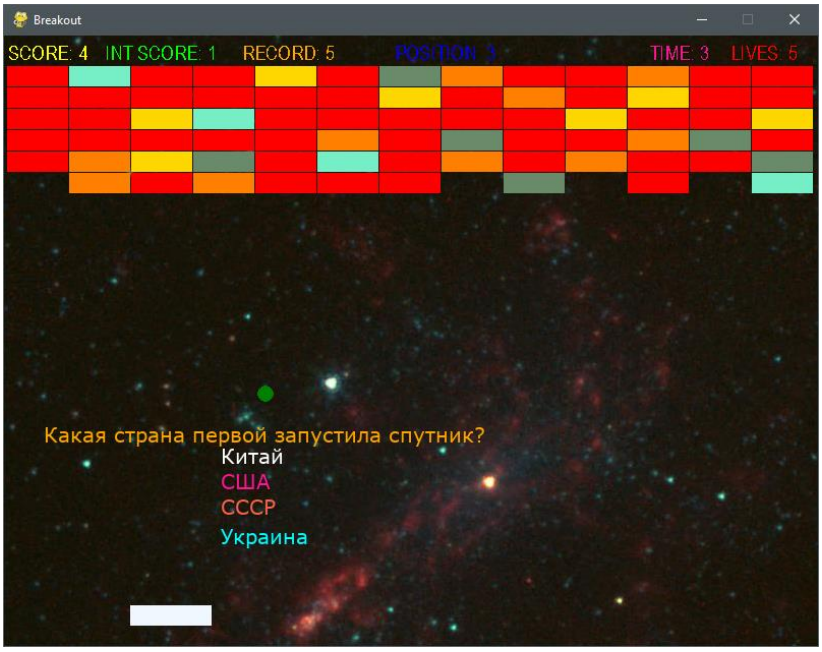

Рисунок 19. Интерфейс игры

3.1.6 Тестирование передвижения шарика

Шарик передвигается в соответствии с задуманным(Рис.20).

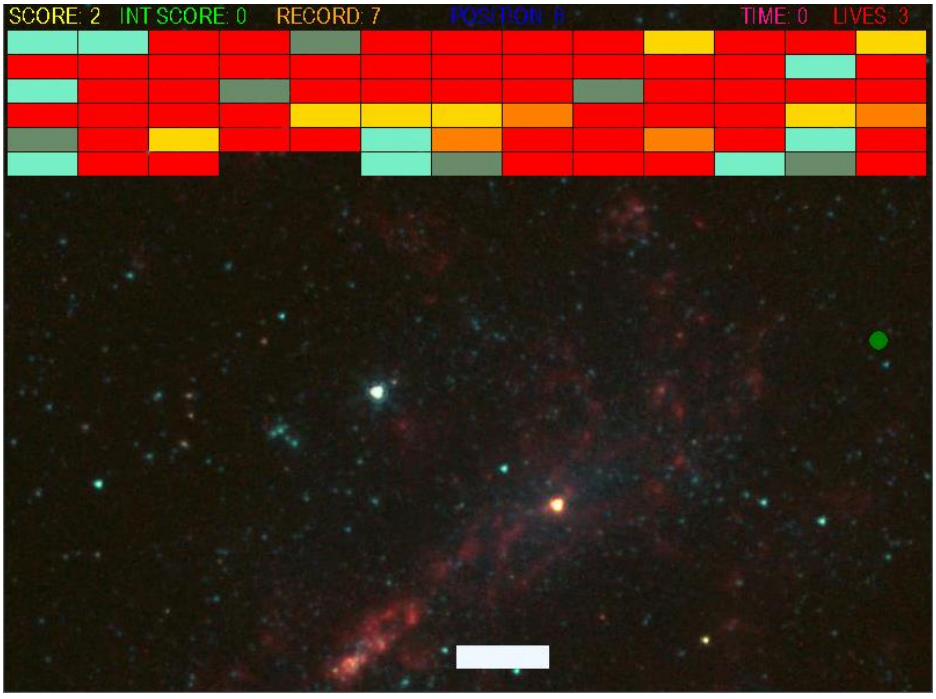

Рисунок 20. Движение шарика

3.1.7 Тестирование разрушаемости блоков

Ниже представлен результат проведенного тестирования(Рис.21).

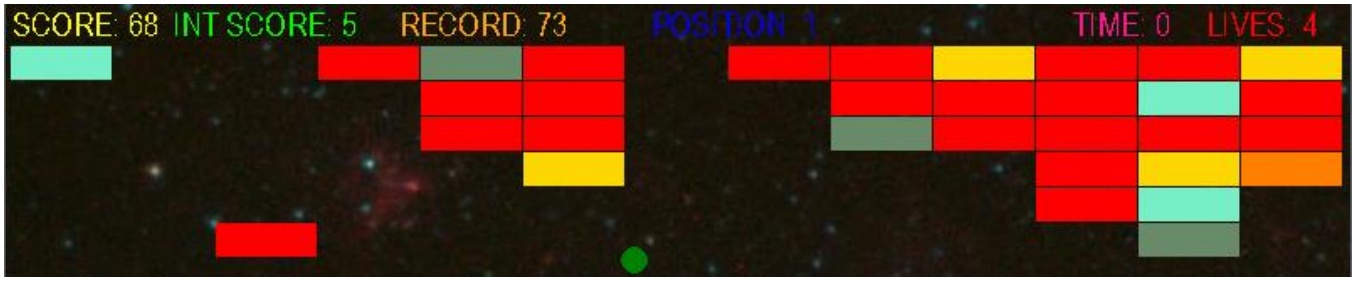

Рисунок 21. Разрушаемость блоков

3.1.8 Тестирование взаимодействия клиента с сервером

Каждый раз, когда изменяется количество очков, отправляется запрос на сервер. При получении сервером запроса, выполняется определенный алгоритм, который в результате возвращает клиенту рекорд пользователя и его позицию среди всех ранее игравших в эту игру игроков.

Когда игра закончена, на сервер поступает запрос о добавлении игрока в рейтинговую таблицу или обновлении его рекорда.

Ниже(Рис.22) показана рейтинговая таблица из базы данных, в которой видно, что рекорд действительно устанавливается, это нам позволяет утверждать, что сервер тоже исправен.

Таблица: PERSON

	LOGIN	RECORD
	Фильтр	Фильтр
1	Player	73
2	Player2	3
3	Player3	4
4	Vlad	5
5	NameTest	6

Рисунок 22. Программа DB Browser

3.1.9 Тестирование конца игры

Когда игра окончена, выводится статистика за игру, количество очков за разрушение блоков и количество очков за ответы на всплывающие вопросы. Тест пройден успешно (Рис.23).

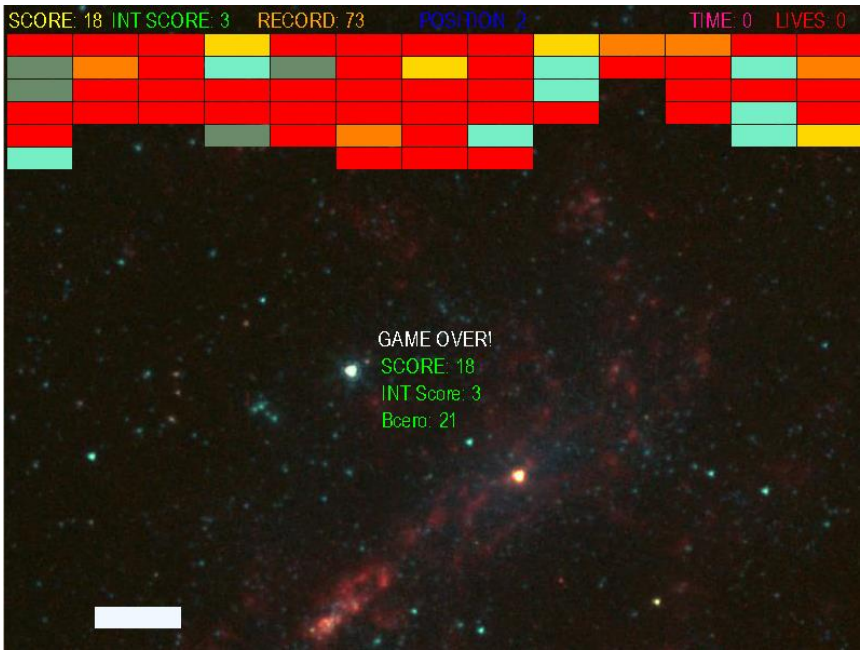

Рисунок 23. Конец игры

ГЛАВА 4
ПЕРСПЕКТИВЫ ДАЛЬНЕЙШЕГО РАЗВИТИЯ ПРОЕКТА

4.1 Перспективы технического развития

Из идей, которых не удалось реализовать в процессе разработки, можно отметить сцену общих настроек игры. Еще одной из значимых недостатков – это малое количество многопоточной оптимизаций.

Для посыла запроса на сервер и получение ответа требуется некоторое количество времени, так как сервер проекта размещен на бесплатном хостинге, эти недостатки четко видны в процессе эксплуатации приложения.

Кроме этого, в проекте сделано много из того, что планировалось сделать, однако есть некоторые идеи, которые хотелось бы реализовать или изменить. Во-первых, это возможность конфигурации настроек игры, таких как сложность игры, выдаваемых игроку на ввод, различные модификаторы для усложнения или упрощения игры, уровень сложности блиц вопросов и т.п.

Во-вторых, визуальная часть проекта также до конца не реализована. Хотелось бы добавить текстуры блокам, которые одновременно будут как привлекать внимание, так и создавать незагруженное эстетичное цельное полотно игры. Шрифт, цвет и расположение балльно-рейтинговой информации тоже должно быть изменено.

В-третьих, для усовершенствования и монетизации проекта необходимо наполнить игру стандартным игровым набором: валютой, магазином, бонусами и тд.

4.2 Перспективы монетизации

На данном этапе никаких способов монетизации не реализовано, однако идеи реализации присутствуют. Одной из перспектив является продажа на онлайн-сервисах цифрового распространения компьютерных игр и программ, таких как «Steam», «Epic Games Store», «GOG». Выставление проекта на продажу

на популярной площадке поможет ему получить хотя бы минимум внимания, и шанс заинтересовать человека, просматривающего списки игр.

Существует огромное количество способов получить доход от своей игры. И дело ограничивается не только прямыми продажами контента.

Одна из самых типовых моделей монетизации — так называемый free-to-play. Популярность этого способа растёт день ото дня. Концепция такова: игра предоставляется бесплатно, но за персонализацию игры, а также внутри игровые предметы пользователь платит.

Как правило, эта модель преобладает в мобильном пространстве, но также встречается в играх в социальных сетях, MMO и компьютерных играх.

Один из лучших способов обеспечить себе постоянный доход. Когда пользователь приобретает не расходуемый предмет, он платит один раз. Когда ему нужны расходуемые предметы, он будет совершать покупки регулярно. Для этого, можно добавить внутри игровой магазин, в котором будет представлена возможность покупки визуальной составляющей платформы, а также улучшений её характеристик.

Настроив систему «ачивок» для игроков, значительно увеличим их желание получить какой-либо предмет. Случайные награды создают ожидание возможности. Исследования показывают, что такой путь получения достижений более привлекателен, чем получение предсказуемой награды, что, конечно, положительно сказывается на монетизации.

Реклама и интернет — связаны на протяжении многих лет. Реклама является одним из лучших способов заработать на игре, не задействуя при этом заработок игроков.

Полноэкранная реклама и межстраничные объявления. Такая реклама заполняет собой весь экран, закрывая интерфейс игры, и держится на экране определённый непродолжительный период, а затем исчезает. В отличие от баннера, который может растворяться на заднем фоне после непосредственного использования, пользователь успевает полностью ознакомиться с межстраничным объявлением перед тем, как оно исчезнет.

ЗАКЛЮЧЕНИЕ

В результате выполнения данной работы, потребовались высокие знания языков программирования. В ходе написания программы требовалось обращаться к дополнительным источникам, чтобы разобраться с такими важными аспектами программирования: работа с файлами, символьными данными, графическим режимом, умение пользоваться конструкторами, деструкторами, наследованием, а также ознакомится с правилами игры «Платформа».

Был получен опыт в разработке крупных проектов, работе с мультифайловыми проектами в среде разработки Microsoft Visual Studio 2019, распределении времени и выставления приоритетов в процессе разработки. Так же был получен опыт в поиске и выборе библиотек, позволяющих реализовать функционал, нужный приложению. Был получен опыт работы со сторонними библиотеками. Освоен принцип реализации работы клиент-серверных структур.

Данная работа помогла лучше освоить языки программирования C++ и Python и улучшить свои навыки программирования.

Продукт, будучи на ранних этапах, все еще нуждается в полировке, в некоторых местах все еще можно найти грубоватые заготовки или не полностью продуманные функции, однако весь функционал в программе рабочий и отвечает целям, поставленным в начале разработки.

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. Стасышин, В. М. Базы данных: технологии доступа: учеб. пособие для СПО / В. М. Стасышин, Т. Л. Стасышина. — 2-е изд., испр. и доп. — М.: Издательство Юрайт, 2018. — 164 с.

5. Федоров, Д. Ю. Программирование на языке высокого уровня Python: учеб. пособие для СПО / Д. Ю. Федоров. — М.: Издательство Юрайт, 2019. — 126 с.

6. Федоров, Д. Ю. Программирование на языке высокого уровня python : учеб. пособие для прикладного бакалавриата / Д. Ю. Федоров. — 2-е изд., перераб. и доп. — М.: Издательство Юрайт, 2019. — 161 с.

ПРИЛОЖЕНИЕ 1
КОД ОСНОВНЫХ МОДУЛЕЙ ПРОЕКТА

ball.py

```
import pygame

from game_object import GameObject
from PIL import ImageFont # new
import config as c

# print(Q)
# Q = [
#       [
#       ['Как называется ближайшая к Солнцу планета?','orange'],
#       ['Венера','red'],
#       ['Марс','deeppink'],
#       ['Меркурий','tomato'],
#       ['Земля','cyan']
#       ],

#
class Ball(GameObject):
        def __init__(self, x, y, r, color, speed):
                # print("Создан Ball") # 3 раза - 3 жизни
                GameObject.__init__(self, x - r, y - r, r * 2, r * 2, speed)
                self.radius = r
                self.diameter = r * 2
                self.color = color
                # self.ClTextQ = Texts_quest(Q,0)
                # print("myclass, создан")


        def draw(self, surface): # отрисовывает каждое положение
                r = pygame.draw.circle(surface, self.color, self.center,
self.radius)
                # print(Breakout.is_game_running)
                # if self.is_game_running:
                #       self.ClTextQ.spawn(surface,self.center)

        def update(self):
                super().update()
```

breakout.py

```
import random
from datetime import datetime, timedelta

import os
import time
import pygame
from pygame.rect import Rect

import config as c
from ball import Ball
```

```
from brick import Brick
from button import Button
from game import Game
from paddle import Paddle
from text_object import TextObject


from texts_quest import Texts_quest as TextsQuest

import colors

import json
```

brick.py

```
import pygame

from game_object import GameObject


class Brick(GameObject):
    def __init__(self, x, y, w, h, color, special_effect=None):
        GameObject.__init__(self, x, y, w, h)
        self.color = color
        self.special_effect = special_effect

    def draw(self, surface):
        pygame.draw.rect(surface, self.color, self.bounds)
```

button.py

```
import pygame

from game_object import GameObject
from text_object import TextObject
import config as c


class Button(GameObject):
    def __init__(self, x, y, w, h, text, on_click=lambda x: None, padding=0):
        super().__init__(x, y, w, h)
        self.state = 'normal'
        self.on_click = on_click

        self.text = TextObject(x + padding, y + padding, lambda: text,
c.button_text_color, c.font_name, c.font_size)

    @property
    def back_color(self):
        return dict(normal=c.button_normal_back_color,
                    hover=c.button_hover_back_color,
                    pressed=c.button_pressed_back_color)[self.state]

    def draw(self, surface):
        pygame.draw.rect(surface, self.back_color, self.bounds)
        self.text.draw(surface)
```

colors.py

```
"""
https://www.webucator.com/blog/2015/03/python-color-constants-module/
"""

ALICEBLUE = (240, 248, 255)
ANTIQUEWHITE = (250, 235, 215)
ANTIQUEWHITE1 = (255, 239, 219)
ANTIQUEWHITE2 = (238, 223, 204)
ANTIQUEWHITE3 = (205, 192, 176)
ANTIQUEWHITE4 = (139, 131, 120)
AQUA = (0, 255, 255)
AQUAMARINE1 = (127, 255, 212)
AQUAMARINE2 = (118, 238, 198)
AQUAMARINE3 = (102, 205, 170)
AQUAMARINE4 = (69, 139, 116)
AZURE1 = (240, 255, 255)
AZURE2 = (224, 238, 238)
AZURE3 = (193, 205, 205)
AZURE4 = (131, 139, 139)
BANANA = (227, 207, 87)
BEIGE = (245, 245, 220)
BISQUE1 = (255, 228, 196)
BISQUE2 = (238, 213, 183)
BISQUE3 = (205, 183, 158)
BISQUE4 = (139, 125, 107)
BLACK = (0, 0, 0)
BLANCHEDALMOND = (255, 235, 205)
BLUE = (0, 0, 255)
BLUE2 = (0, 0, 238)
BLUE3 = (0, 0, 205)
BLUE4 = (0, 0, 139)
BLUEVIOLET = (138, 43, 226)
BRICK = (156, 102, 31)
BROWN = (165, 42, 42)
BROWN1 = (255, 64, 64)
BROWN2 = (238, 59, 59)
BROWN3 = (205, 51, 51)
BROWN4 = (139, 35, 35)
BURLYWOOD = (222, 184, 135)
BURLYWOOD1 = (255, 211, 155)
BURLYWOOD2 = (238, 197, 145)
BURLYWOOD3 = (205, 170, 125)
BURLYWOOD4 = (139, 115, 85)
BURNTSIENNA = (138, 54, 15)
BURNTUMBER = (138, 51, 36)
CADETBLUE = (95, 158, 160)
CADETBLUE1 = (152, 245, 255)
CADETBLUE2 = (142, 229, 238)
CADETBLUE3 = (122, 197, 205)
CADETBLUE4 = (83, 134, 139)
CADMIUMORANGE = (255, 97, 3)
CADMIUMYELLOW = (255, 153, 18)
CARROT = (237, 145, 33)
CHARTREUSE1 = (127, 255, 0)
CHARTREUSE2 = (118, 238, 0)
CHARTREUSE3 = (102, 205, 0)
CHARTREUSE4 = (69, 139, 0)
```

```
CHOCOLATE = (210, 105, 30)
CHOCOLATE1 = (255, 127, 36)
CHOCOLATE2 = (238, 118, 33)
CHOCOLATE3 = (205, 102, 29)
CHOCOLATE4 = (139, 69, 19)
COBALT = (61, 89, 171)
COBALTGREEN = (61, 145, 64)
COLDGREY = (128, 138, 135)
CORAL = (255, 127, 80)
CORAL1 = (255, 114, 86)
CORAL2 = (238, 106, 80)
CORAL3 = (205, 91, 69)
CORAL4 = (139, 62, 47)
CORNFLOWERBLUE = (100, 149, 237)
CORNSILK1 = (255, 248, 220)
CORNSILK2 = (238, 232, 205)
CORNSILK3 = (205, 200, 177)
CORNSILK4 = (139, 136, 120)
CRIMSON = (220, 20, 60)
CYAN2 = (0, 238, 238)
CYAN3 = (0, 205, 205)
CYAN4 = (0, 139, 139)
DARKGOLDENROD = (184, 134, 11)
DARKGOLDENROD1 = (255, 185, 15)
DARKGOLDENROD2 = (238, 173, 14)
DARKGOLDENROD3 = (205, 149, 12)
DARKGOLDENROD4 = (139, 101, 8)
DARKGRAY = (169, 169, 169)
DARKGREEN = (0, 100, 0)
DARKKHAKI = (189, 183, 107)
DARKOLIVEGREEN = (85, 107, 47)
DARKOLIVEGREEN1 = (202, 255, 112)
DARKOLIVEGREEN2 = (188, 238, 104)
DARKOLIVEGREEN3 = (162, 205, 90)
DARKOLIVEGREEN4 = (110, 139, 61)
DARKORANGE = (255, 140, 0)
DARKORANGE1 = (255, 127, 0)
DARKORANGE2 = (238, 118, 0)
DARKORANGE3 = (205, 102, 0)
DARKORANGE4 = (139, 69, 0)
DARKORCHID = (153, 50, 204)
DARKORCHID1 = (191, 62, 255)
DARKORCHID2 = (178, 58, 238)
DARKORCHID3 = (154, 50, 205)
DARKORCHID4 = (104, 34, 139)
DARKSALMON = (233, 150, 122)
DARKSEAGREEN = (143, 188, 143)
DARKSEAGREEN1 = (193, 255, 193)
DARKSEAGREEN2 = (180, 238, 180)
DARKSEAGREEN3 = (155, 205, 155)
DARKSEAGREEN4 = (105, 139, 105)
DARKSLATEBLUE = (72, 61, 139)
DARKSLATEGRAY = (47, 79, 79)
DARKSLATEGRAY1 = (151, 255, 255)
DARKSLATEGRAY2 = (141, 238, 238)
DARKSLATEGRAY3 = (121, 205, 205)
DARKSLATEGRAY4 = (82, 139, 139)
DARKTURQUOISE = (0, 206, 209)
DARKVIOLET = (148, 0, 211)
```

```
DEEPPINK1 = (255, 20, 147)
DEEPPINK2 = (238, 18, 137)
DEEPPINK3 = (205, 16, 118)
DEEPPINK4 = (139, 10, 80)
DEEPSKYBLUE1 = (0, 191, 255)
DEEPSKYBLUE2 = (0, 178, 238)
DEEPSKYBLUE3 = (0, 154, 205)
DEEPSKYBLUE4 = (0, 104, 139)
DIMGRAY = (105, 105, 105)
DIMGRAY = (105, 105, 105)
DODGERBLUE1 = (30, 144, 255)
DODGERBLUE2 = (28, 134, 238)
DODGERBLUE3 = (24, 116, 205)
DODGERBLUE4 = (16, 78, 139)
EGGSHELL = (252, 230, 201)
EMERALDGREEN = (0, 201, 87)
FIREBRICK = (178, 34, 34)
FIREBRICK1 = (255, 48, 48)
FIREBRICK2 = (238, 44, 44)
FIREBRICK3 = (205, 38, 38)
FIREBRICK4 = (139, 26, 26)
FLESH = (255, 125, 64)
FLORALWHITE = (255, 250, 240)
FORESTGREEN = (34, 139, 34)
GAINSBORO = (220, 220, 220)
GHOSTWHITE = (248, 248, 255)
GOLD1 = (255, 215, 0)
GOLD2 = (238, 201, 0)
GOLD3 = (205, 173, 0)
GOLD4 = (139, 117, 0)
GOLDENROD = (218, 165, 32)
GOLDENROD1 = (255, 193, 37)
GOLDENROD2 = (238, 180, 34)
GOLDENROD3 = (205, 155, 29)
GOLDENROD4 = (139, 105, 20)
GRAY = (128, 128, 128)
GRAY1 = (3, 3, 3)
GRAY10 = (26, 26, 26)
GRAY11 = (28, 28, 28)
GRAY12 = (31, 31, 31)
GRAY13 = (33, 33, 33)
GRAY14 = (36, 36, 36)
GRAY15 = (38, 38, 38)
GRAY16 = (41, 41, 41)
GRAY17 = (43, 43, 43)
GRAY18 = (46, 46, 46)
GRAY19 = (48, 48, 48)
GRAY2 = (5, 5, 5)
GRAY20 = (51, 51, 51)
GRAY21 = (54, 54, 54)
GRAY22 = (56, 56, 56)
GRAY23 = (59, 59, 59)
GRAY24 = (61, 61, 61)
GRAY25 = (64, 64, 64)
GRAY26 = (66, 66, 66)
GRAY27 = (69, 69, 69)
GRAY28 = (71, 71, 71)
GRAY29 = (74, 74, 74)
GRAY3 = (8, 8, 8)
```

```
GRAY30 = (77, 77, 77)
GRAY31 = (79, 79, 79)
GRAY32 = (82, 82, 82)
GRAY33 = (84, 84, 84)
GRAY34 = (87, 87, 87)
GRAY35 = (89, 89, 89)
GRAY36 = (92, 92, 92)
GRAY37 = (94, 94, 94)
GRAY38 = (97, 97, 97)
GRAY39 = (99, 99, 99)
GRAY4 = (10, 10, 10)
GRAY40 = (102, 102, 102)
GRAY42 = (107, 107, 107)
GRAY43 = (110, 110, 110)
GRAY44 = (112, 112, 112)
GRAY45 = (115, 115, 115)
GRAY46 = (117, 117, 117)
GRAY47 = (120, 120, 120)
GRAY48 = (122, 122, 122)
GRAY49 = (125, 125, 125)
GRAY5 = (13, 13, 13)
GRAY50 = (127, 127, 127)
GRAY51 = (130, 130, 130)
GRAY52 = (133, 133, 133)
GRAY53 = (135, 135, 135)
GRAY54 = (138, 138, 138)
GRAY55 = (140, 140, 140)
GRAY56 = (143, 143, 143)
GRAY57 = (145, 145, 145)
GRAY58 = (148, 148, 148)
GRAY59 = (150, 150, 150)
GRAY6 = (15, 15, 15)
GRAY60 = (153, 153, 153)
GRAY61 = (156, 156, 156)
GRAY62 = (158, 158, 158)
GRAY63 = (161, 161, 161)
GRAY64 = (163, 163, 163)
GRAY65 = (166, 166, 166)
GRAY66 = (168, 168, 168)
GRAY67 = (171, 171, 171)
GRAY68 = (173, 173, 173)
GRAY69 = (176, 176, 176)
GRAY7 = (18, 18, 18)
GRAY70 = (179, 179, 179)
GRAY71 = (181, 181, 181)
GRAY72 = (184, 184, 184)
GRAY73 = (186, 186, 186)
GRAY74 = (189, 189, 189)
GRAY75 = (191, 191, 191)
GRAY76 = (194, 194, 194)
GRAY77 = (196, 196, 196)
GRAY78 = (199, 199, 199)
GRAY79 = (201, 201, 201)
GRAY8 = (20, 20, 20)
GRAY80 = (204, 204, 204)
GRAY81 = (207, 207, 207)
GRAY82 = (209, 209, 209)
GRAY83 = (212, 212, 212)
GRAY84 = (214, 214, 214)
```

```
GRAY85 = (217, 217, 217)
GRAY86 = (219, 219, 219)
GRAY87 = (222, 222, 222)
GRAY88 = (224, 224, 224)
GRAY89 = (227, 227, 227)
GRAY9 = (23, 23, 23)
GRAY90 = (229, 229, 229)
GRAY91 = (232, 232, 232)
GRAY92 = (235, 235, 235)
GRAY93 = (237, 237, 237)
GRAY94 = (240, 240, 240)
GRAY95 = (242, 242, 242)
GRAY97 = (247, 247, 247)
GRAY98 = (250, 250, 250)
GRAY99 = (252, 252, 252)
GREEN = (0, 128, 0)
GREEN1 = (0, 255, 0)
GREEN2 = (0, 238, 0)
GREEN3 = (0, 205, 0)
GREEN4 = (0, 139, 0)
GREENYELLOW = (173, 255, 47)
HONEYDEW1 = (240, 255, 240)
HONEYDEW2 = (224, 238, 224)
HONEYDEW3 = (193, 205, 193)
HONEYDEW4 = (131, 139, 131)
HOTPINK = (255, 105, 180)
HOTPINK1 = (255, 110, 180)
HOTPINK2 = (238, 106, 167)
HOTPINK3 = (205, 96, 144)
HOTPINK4 = (139, 58, 98)
INDIANRED = (205, 92, 92)
INDIANRED1 = (255, 106, 106)
INDIANRED2 = (238, 99, 99)
INDIANRED3 = (205, 85, 85)
INDIANRED4 = (139, 58, 58)
INDIGO = (75, 0, 130)
IVORY1 = (255, 255, 240)
IVORY2 = (238, 238, 224)
IVORY3 = (205, 205, 193)
IVORY4 = (139, 139, 131)
IVORYBLACK = (41, 36, 33)
KHAKI = (240, 230, 140)
KHAKI1 = (255, 246, 143)
KHAKI2 = (238, 230, 133)
KHAKI3 = (205, 198, 115)
KHAKI4 = (139, 134, 78)
LAVENDER = (230, 230, 250)
LAVENDERBLUSH1 = (255, 240, 245)
LAVENDERBLUSH2 = (238, 224, 229)
LAVENDERBLUSH3 = (205, 193, 197)
LAVENDERBLUSH4 = (139, 131, 134)
LAWNGREEN = (124, 252, 0)
LEMONCHIFFON1 = (255, 250, 205)
LEMONCHIFFON2 = (238, 233, 191)
LEMONCHIFFON3 = (205, 201, 165)
LEMONCHIFFON4 = (139, 137, 112)
LIGHTBLUE = (173, 216, 230)
LIGHTBLUE1 = (191, 239, 255)
LIGHTBLUE2 = (178, 223, 238)
```

```
LIGHTBLUE3 = (154, 192, 205)
LIGHTBLUE4 = (104, 131, 139)
LIGHTCORAL = (240, 128, 128)
LIGHTCYAN1 = (224, 255, 255)
LIGHTCYAN2 = (209, 238, 238)
LIGHTCYAN3 = (180, 205, 205)
LIGHTCYAN4 = (122, 139, 139)
LIGHTGOLDENROD1 = (255, 236, 139)
LIGHTGOLDENROD2 = (238, 220, 130)
LIGHTGOLDENROD3 = (205, 190, 112)
LIGHTGOLDENROD4 = (139, 129, 76)
LIGHTGOLDENRODYELLOW = (250, 250, 210)
LIGHTGREY = (211, 211, 211)
LIGHTPINK = (255, 182, 193)
LIGHTPINK1 = (255, 174, 185)
LIGHTPINK2 = (238, 162, 173)
LIGHTPINK3 = (205, 140, 149)
LIGHTPINK4 = (139, 95, 101)
LIGHTSALMON1 = (255, 160, 122)
LIGHTSALMON2 = (238, 149, 114)
LIGHTSALMON3 = (205, 129, 98)
LIGHTSALMON4 = (139, 87, 66)
LIGHTSEAGREEN = (32, 178, 170)
LIGHTSKYBLUE = (135, 206, 250)
LIGHTSKYBLUE1 = (176, 226, 255)
LIGHTSKYBLUE2 = (164, 211, 238)
LIGHTSKYBLUE3 = (141, 182, 205)
LIGHTSKYBLUE4 = (96, 123, 139)
LIGHTSLATEBLUE = (132, 112, 255)
LIGHTSLATEGRAY = (119, 136, 153)
LIGHTSTEELBLUE = (176, 196, 222)
LIGHTSTEELBLUE1 = (202, 225, 255)
LIGHTSTEELBLUE2 = (188, 210, 238)
LIGHTSTEELBLUE3 = (162, 181, 205)
LIGHTSTEELBLUE4 = (110, 123, 139)
LIGHTYELLOW1 = (255, 255, 224)
LIGHTYELLOW2 = (238, 238, 209)
LIGHTYELLOW3 = (205, 205, 180)
LIGHTYELLOW4 = (139, 139, 122)
LIMEGREEN = (50, 205, 50)
LINEN = (250, 240, 230)
MAGENTA = (255, 0, 255)
MAGENTA2 = (238, 0, 238)
MAGENTA3 = (205, 0, 205)
MAGENTA4 = (139, 0, 139)
MANGANESEBLUE = (3, 168, 158)
MAROON = (128, 0, 0)
MAROON1 = (255, 52, 179)
MAROON2 = (238, 48, 167)
MAROON3 = (205, 41, 144)
MAROON4 = (139, 28, 98)
MEDIUMORCHID = (186, 85, 211)
MEDIUMORCHID1 = (224, 102, 255)
MEDIUMORCHID2 = (209, 95, 238)
MEDIUMORCHID3 = (180, 82, 205)
MEDIUMORCHID4 = (122, 55, 139)
MEDIUMPURPLE = (147, 112, 219)
MEDIUMPURPLE1 = (171, 130, 255)
MEDIUMPURPLE2 = (159, 121, 238)
```

```
MEDIUMPURPLE3 = (137, 104, 205)
MEDIUMPURPLE4 = (93, 71, 139)
MEDIUMSEAGREEN = (60, 179, 113)
MEDIUMSLATEBLUE = (123, 104, 238)
MEDIUMSPRINGGREEN = (0, 250, 154)
MEDIUMTURQUOISE = (72, 209, 204)
MEDIUMVIOLETRED = (199, 21, 133)
MELON = (227, 168, 105)
MIDNIGHTBLUE = (25, 25, 112)
MINT = (189, 252, 201)
MINTCREAM = (245, 255, 250)
MISTYROSE1 = (255, 228, 225)
MISTYROSE2 = (238, 213, 210)
MISTYROSE3 = (205, 183, 181)
MISTYROSE4 = (139, 125, 123)
MOCCASIN = (255, 228, 181)
NAVAJOWHITE1 = (255, 222, 173)
NAVAJOWHITE2 = (238, 207, 161)
NAVAJOWHITE3 = (205, 179, 139)
NAVAJOWHITE4 = (139, 121, 94)
NAVY = (0, 0, 128)
OLDLACE = (253, 245, 230)
OLIVE = (128, 128, 0)
OLIVEDRAB = (107, 142, 35)
OLIVEDRAB1 = (192, 255, 62)
OLIVEDRAB2 = (179, 238, 58)
OLIVEDRAB3 = (154, 205, 50)
OLIVEDRAB4 = (105, 139, 34)
ORANGE = (255, 128, 0)
ORANGE1 = (255, 165, 0)
ORANGE2 = (238, 154, 0)
ORANGE3 = (205, 133, 0)
ORANGE4 = (139, 90, 0)
ORANGERED1 = (255, 69, 0)
ORANGERED2 = (238, 64, 0)
ORANGERED3 = (205, 55, 0)
ORANGERED4 = (139, 37, 0)
ORCHID = (218, 112, 214)
ORCHID1 = (255, 131, 250)
ORCHID2 = (238, 122, 233)
ORCHID3 = (205, 105, 201)
ORCHID4 = (139, 71, 137)
PALEGOLDENROD = (238, 232, 170)
PALEGREEN = (152, 251, 152)
PALEGREEN1 = (154, 255, 154)
PALEGREEN2 = (144, 238, 144)
PALEGREEN3 = (124, 205, 124)
PALEGREEN4 = (84, 139, 84)
PALETURQUOISE1 = (187, 255, 255)
PALETURQUOISE2 = (174, 238, 238)
PALETURQUOISE3 = (150, 205, 205)
PALETURQUOISE4 = (102, 139, 139)
PALEVIOLETRED = (219, 112, 147)
PALEVIOLETRED1 = (255, 130, 171)
PALEVIOLETRED2 = (238, 121, 159)
PALEVIOLETRED3 = (205, 104, 137)
PALEVIOLETRED4 = (139, 71, 93)
PAPAYAWHIP = (255, 239, 213)
PEACHPUFF1 = (255, 218, 185)
```

```
PEACHPUFF2 = (238, 203, 173)
PEACHPUFF3 = (205, 175, 149)
PEACHPUFF4 = (139, 119, 101)
PEACOCK = (51, 161, 201)
PINK = (255, 192, 203)
PINK1 = (255, 181, 197)
PINK2 = (238, 169, 184)
PINK3 = (205, 145, 158)
PINK4 = (139, 99, 108)
PLUM = (221, 160, 221)
PLUM1 = (255, 187, 255)
PLUM2 = (238, 174, 238)
PLUM3 = (205, 150, 205)
PLUM4 = (139, 102, 139)
POWDERBLUE = (176, 224, 230)
PURPLE = (128, 0, 128)
PURPLE1 = (155, 48, 255)
PURPLE2 = (145, 44, 238)
PURPLE3 = (125, 38, 205)
PURPLE4 = (85, 26, 139)
RASPBERRY = (135, 38, 87)
RAWSIENNA = (199, 97, 20)
RED1 = (255, 0, 0)
RED2 = (238, 0, 0)
RED3 = (205, 0, 0)
RED4 = (139, 0, 0)
ROSYBROWN = (188, 143, 143)
ROSYBROWN1 = (255, 193, 193)
ROSYBROWN2 = (238, 180, 180)
ROSYBROWN3 = (205, 155, 155)
ROSYBROWN4 = (139, 105, 105)
ROYALBLUE = (65, 105, 225)
ROYALBLUE1 = (72, 118, 255)
ROYALBLUE2 = (67, 110, 238)
ROYALBLUE3 = (58, 95, 205)
ROYALBLUE4 = (39, 64, 139)
SALMON = (250, 128, 114)
SALMON1 = (255, 140, 105)
SALMON2 = (238, 130, 98)
SALMON3 = (205, 112, 84)
SALMON4 = (139, 76, 57)
SANDYBROWN = (244, 164, 96)
SAPGREEN = (48, 128, 20)
SEAGREEN1 = (84, 255, 159)
SEAGREEN2 = (78, 238, 148)
SEAGREEN3 = (67, 205, 128)
SEAGREEN4 = (46, 139, 87)
SEASHELL1 = (255, 245, 238)
SEASHELL2 = (238, 229, 222)
SEASHELL3 = (205, 197, 191)
SEASHELL4 = (139, 134, 130)
SEPIA = (94, 38, 18)
SGIBEET = (142, 56, 142)
SGIBRIGHTGRAY = (197, 193, 170)
SGICHARTREUSE = (113, 198, 113)
SGIDARKGRAY = (85, 85, 85)
SGIGRAY12 = (30, 30, 30)
SGIGRAY16 = (40, 40, 40)
SGIGRAY32 = (81, 81, 81)
```

```
SGIGRAY36 = (91, 91, 91)
SGIGRAY52 = (132, 132, 132)
SGIGRAY56 = (142, 142, 142)
SGIGRAY72 = (183, 183, 183)
SGIGRAY76 = (193, 193, 193)
SGIGRAY92 = (234, 234, 234)
SGIGRAY96 = (244, 244, 244)
SGILIGHTBLUE = (125, 158, 192)
SGILIGHTGRAY = (170, 170, 170)
SGIOLIVEDRAB = (142, 142, 56)
SGISALMON = (198, 113, 113)
SGISLATEBLUE = (113, 113, 198)
SGITEAL = (56, 142, 142)
SIENNA = (160, 82, 45)
SIENNA1 = (255, 130, 71)
SIENNA2 = (238, 121, 66)
SIENNA3 = (205, 104, 57)
SIENNA4 = (139, 71, 38)
SILVER = (192, 192, 192)
SKYBLUE = (135, 206, 235)
SKYBLUE1 = (135, 206, 255)
SKYBLUE2 = (126, 192, 238)
SKYBLUE3 = (108, 166, 205)
SKYBLUE4 = (74, 112, 139)
SLATEBLUE = (106, 90, 205)
SLATEBLUE1 = (131, 111, 255)
SLATEBLUE2 = (122, 103, 238)
SLATEBLUE3 = (105, 89, 205)
SLATEBLUE4 = (71, 60, 139)
SLATEGRAY = (112, 128, 144)
SLATEGRAY1 = (198, 226, 255)
SLATEGRAY2 = (185, 211, 238)
SLATEGRAY3 = (159, 182, 205)
SLATEGRAY4 = (108, 123, 139)
SNOW1 = (255, 250, 250)
SNOW2 = (238, 233, 233)
SNOW3 = (205, 201, 201)
SNOW4 = (139, 137, 137)
SPRINGGREEN = (0, 255, 127)
SPRINGGREEN1 = (0, 238, 118)
SPRINGGREEN2 = (0, 205, 102)
SPRINGGREEN3 = (0, 139, 69)
STEELBLUE = (70, 130, 180)
STEELBLUE1 = (99, 184, 255)
STEELBLUE2 = (92, 172, 238)
STEELBLUE3 = (79, 148, 205)
STEELBLUE4 = (54, 100, 139)
TAN = (210, 180, 140)
TAN1 = (255, 165, 79)
TAN2 = (238, 154, 73)
TAN3 = (205, 133, 63)
TAN4 = (139, 90, 43)
TEAL = (0, 128, 128)
THISTLE = (216, 191, 216)
THISTLE1 = (255, 225, 255)
THISTLE2 = (238, 210, 238)
THISTLE3 = (205, 181, 205)
THISTLE4 = (139, 123, 139)
TOMATO1 = (255, 99, 71)
```

```
TOMATO2 = (238, 92, 66)
TOMATO3 = (205, 79, 57)
TOMATO4 = (139, 54, 38)
TURQUOISE = (64, 224, 208)
TURQUOISE1 = (0, 245, 255)
TURQUOISE2 = (0, 229, 238)
TURQUOISE3 = (0, 197, 205)
TURQUOISE4 = (0, 134, 139)
TURQUOISEBLUE = (0, 199, 140)
VIOLET = (238, 130, 238)
VIOLETRED = (208, 32, 144)
VIOLETRED1 = (255, 62, 150)
VIOLETRED2 = (238, 58, 140)
VIOLETRED3 = (205, 50, 120)
VIOLETRED4 = (139, 34, 82)
WARMGREY = (128, 128, 105)
WHEAT = (245, 222, 179)
WHEAT1 = (255, 231, 186)
WHEAT2 = (238, 216, 174)
WHEAT3 = (205, 186, 150)
WHEAT4 = (139, 126, 102)
WHITE = (255, 255, 255)
WHITESMOKE = (245, 245, 245)
YELLOW1 = (255, 255, 0)
YELLOW2 = (238, 238, 0)
YELLOW3 = (205, 205, 0)
YELLOW4 = (139, 139, 0)
```

config.py

```
import colors

screen_width = 800
screen_height = 600
background_image = 'images/background.jpg'

frame_rate = 90

row_count = 6
brick_width = 60
brick_height = 20
brick_color = colors.RED1
offset_y = brick_height + 10

ball_speed = 3
ball_radius = 8
ball_color = colors.GREEN

paddle_width = 80
paddle_height = 20
paddle_color = colors.ALICEBLUE
paddle_speed = 6

status_offset_y = 5

text_color = colors.YELLOW1
initial_lives = 3
lives_right_offset = 85
lives_offset = screen_width - lives_right_offset
```

```
score_offset = 5

font_name = 'Arial'
font_size = 20

effect_duration = 20

sounds_effects = dict(
    brick_hit='sound_effects/brick_hit.wav',
    effect_done='sound_effects/effect_done.wav',
    paddle_hit='sound_effects/paddle_hit.wav',
    level_complete='sound_effects/level_complete.wav',
)

message_duration = 2

button_text_color = colors.WHITE,
button_normal_back_color = colors.INDIANRED1
button_hover_back_color = colors.INDIANRED2
button_pressed_back_color = colors.INDIANRED3

menu_offset_x = 20
menu_offset_y = 300
menu_button_w = 80
menu_button_h = 50
```

game.py

```
import pygame
import sys

from collections import defaultdict


class Game:
    def __init__(self, caption, width, height, back_image_filename, frame_rate):
        self.background_image = pygame.image.load(back_image_filename)
        self.frame_rate = frame_rate
        self.game_over = False
        self.objects = []
        pygame.mixer.init(44100, -16, 2, 4096)
        pygame.init()
        pygame.font.init()
        self.surface = pygame.display.set_mode((width, height))
        pygame.display.set_caption(caption)
        self.clock = pygame.time.Clock()
        self.keydown_handlers = defaultdict(list)
        self.keyup_handlers = defaultdict(list)
        self.mouse_handlers = []

        self.choose = 1
        self.last_choose = 1
        self.is_answere_true = False

    def update(self):
        for o in self.objects:
            o.update()

    def draw(self):
```

```
        for o in self.objects:
            o.draw(self.surface)

    def handle_events(self):
        global choose,texts
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                pygame.quit()
                sys.exit()
            elif event.type == pygame.KEYDOWN:
                for handler in self.keydown_handlers[event.key]:
                    handler(event.key)



                if self.is_game_running:
                    if event.key == pygame.K_DOWN:
                        print("up")
                        # font = pygame.font.SysFont('Verdana', 20)# вынести в
конфиг size
                        # texts[choose][0] = font.render(texts[choose][1], True,
texts[choose][2])# Истина цвет
                        if self.choose == 4:
                            self.choose = 1
                        else:
                            self.choose +=1
                        # texts[choose][0] = font.render(texts[choose][1], True,
"white")# Белый цвет

                    if event.key == pygame.K_UP:
                        print("down")
                        # font = pygame.font.SysFont('Verdana', 20)# вынести в
конфиг size
                        # texts[choose][0] = font.render(texts[choose][1], True,
texts[choose][2])# Истина цвет

                        if self.choose == 1:
                            self.choose = 4
                        else:
                            self.choose -=1
                        # texts[choose][0] = font.render(texts[choose][1], True,
"white")# Белый цвет


                    if event.key == pygame.K_RETURN and len(self.TextQuest.texts)
!= 0:
                        if self.TextQuest.texts[self.choose][3]==str(1):
                            self.is_answere_true = True

                            print("Верно")
                        else:
                            self.is_answere_False = True
                            print("неверно")




            elif event.type == pygame.KEYUP:
```

```
                for handler in self.keyup_handlers[event.key]:
                    handler(event.key)
            elif event.type in (pygame.MOUSEBUTTONDOWN, pygame.MOUSEBUTTONUP,
pygame.MOUSEMOTION):
                for handler in self.mouse_handlers:
                    handler(event.type, event.pos)



    def run(self):
        while not self.game_over:
            self.surface.blit(self.background_image, (0, 0))

            self.handle_events()
            self.update()
            self.draw()

            pygame.display.update()
            self.clock.tick(self.frame_rate)
        import time
        time.sleep(5) # Задержка перед закрытием, чтобы посомтреть на результат
```

game_object.py

```
from pygame.rect import Rect


class GameObject:
    def __init__(self, x, y, w, h, speed=(0,0)):
        self.bounds = Rect(x, y, w, h)
        self.speed = speed

    @property
    def left(self):
        return self.bounds.left

    @property
    def right(self):
        return self.bounds.right

    @property
    def top(self):
        return self.bounds.top

    @property
    def bottom(self):
        return self.bounds.bottom

    @property
    def width(self):
        return self.bounds.width

    @property
    def height(self):
        return self.bounds.height

    @property
    def center(self):
```

```
        return self.bounds.center
```

paddle.py

```
import pygame

import config as c
from game_object import GameObject


class Paddle(GameObject):
    def __init__(self, x, y, w, h, color, offset):
        GameObject.__init__(self, x, y, w, h)
        self.color = color
        self.offset = offset
        self.moving_left = False
        self.moving_right = False

    def draw(self, surface):
        pygame.draw.rect(surface, self.color, self.bounds)

    def handle(self, key):
        if key == pygame.K_LEFT:
            self.moving_left = not self.moving_left
        else:
            self.moving_right = not self.moving_right

    def update(self):
        if self.moving_left:
            dx = -(min(self.offset, self.left))
        elif self.moving_right:
            dx = min(self.offset, c.screen_width - self.right)
        else:
            return

        self.move(dx, 0)
```

Q.json

```
[[{
     "namea":"Как называется ближайшая к Солнцу планета?",
     "color":"orange",
      "status": "0"

    },{
      "namea":"Венера",
      "color":"red",
      "status": "0"
    },{
      "namea":"Марс",
      "color":"deeppink",
      "status": "0"
    },{
      "namea":"Меркурий",
      "color":"tomato",
      "status": "1"
    },{
      "namea":"Земля",
```

```
      "color":"cyan",
      "status": "0"
    }],[{
     "namea":"Какая страна первой запустила спутник?",
     "color":"orange",
      "status": "0"

    },{
      "namea":"Китай",
      "color":"red",
      "status": "0"
    },{
      "namea":"США",
      "color":"deeppink",
      "status": "0"
    },{
      "namea":"СССР",
      "color":"tomato",
      "status": "1"
    },{
      "namea":"Украина",
      "color":"cyan",
      "status": "0"
    }]


    ]
```

text_object.py

```
import pygame


class TextObject:
    def __init__(self, x, y, text_func, color, font_name, font_size):
        self.pos = (x, y)
        self.text_func = text_func
        self.color = color
        self.font = pygame.font.SysFont(font_name, font_size)
        self.bounds = self.get_surface(text_func())

    def draw(self, surface, centralized=False):
        text_surface, self.bounds = self.get_surface(self.text_func())
        if centralized:
            pos = (self.pos[0] - self.bounds.width // 2, self.pos[1])
        else:
            pos = self.pos
        surface.blit(text_surface, pos)

    def get_surface(self, text):
        text_surface = self.font.render(text, False, self.color)
        return text_surface, text_surface.get_rect()

    def update(self):
        pass
```

text_quest.py

```
texts=[]
choose = 1
from PIL import ImageFont # new
import pygame
import config as c
class Texts_quest():
        def __init__(self,Q): # r_int вопрос в Q
                # print("init start")
                self.Q = Q
                self.fsize = 20
                self.font_gsize = ImageFont.truetype('verdana.ttf', self.fsize)
                self.texts = []
                self.font = pygame.font.SysFont('Verdana', self.fsize)

                self.sizes = []
                # print("start create")
                # self.texts.clear()
                choose = 1
                # self.create_text()
                # print("init fin")

        def get_font_size(self,text):
                return self.font_gsize.getsize(text)

        def create_text(self,r_int=0):
                self.r_int = r_int # от 0 до размера Q
                p=0
                for quest in self.Q[self.r_int]:
                        # print(quest)

                        color = quest['color']

                        if p==1:
                                color = "white"
                        # self.texts.append(self.font.render(quest['namea'], True,
quest['color']))
                        self.texts.append([self.font.render(quest['namea'], True,
color),quest['namea'], quest['color'],quest['status']])
                        self.sizes.append(self.get_font_size(quest['namea']))
                        j=0
                        self.Max_h = 0
                        self.Max_w = 0
                        # print("cickle")
                        while j<len(self.sizes):
                                # print("inC")
                                if j==0:
                                        j+=1
                                        continue

                                if self.sizes[j][0]>self.Max_w:
                                        self.Max_w = self.sizes[j][0]
                                if self.sizes[j][1]>self.Max_h:
                                        self.Max_h = self.sizes[j][1]
                                j+=1
                        p+=1

        def delete_text(self):
```

```
                self.texts.clear()
                self.sizes.clear()

        def spawn(self,surface,XY):
                self.surface = surface
                self.XY = XY
                i = 0
                for text in self.texts:

                        # Границы, если слева

                        hals_sz_text = self.Max_w/2
                        if i==0:
                                hals_sz_text = self.sizes[i][0]/2



                        if self.XY[0]<c.screen_width/2: # левая часть

                                #Левая часть
                                if self.XY[0] - hals_sz_text <0:
                                        pos  = (0 ,self.XY[1] +
self.sizes[i][1]+i*self.Max_h)
                                        self.surface.blit(text[0], pos)
                                else:
                                        pos  = (self.XY[0]- hals_sz_text,self.XY[1]
+ self.sizes[i][1]+i*self.Max_h)
                                        self.surface.blit(text[0], pos)
                        else: # Правая часть

                                #Правая часть
                                if self.XY[0] + hals_sz_text >c.screen_width:
                                        pos  = (c.screen_width- hals_sz_text*2
,self.XY[1] + self.sizes[i][1]+i*self.Max_h)
                                        self.surface.blit(text[0], pos)
                                else:
                                        pos  = (self.XY[0]- hals_sz_text,self.XY[1]
+ self.sizes[i][1]+i*self.Max_h)
                                self.surface.blit(text[0], pos)

                        i+=1
```

server.cpp

```
#include <iostream>
#include <cpp_httplib/httplib.h>
using namespace std;
using namespace httplib;
#include <string>
#include <cmath>
#include <fstream>
#include <iomanip>
#include <vector>

#include <nlohmann/json.hpp>
#define SQLITE_API __declspec(dllimport)
#include <sqlite3/sqlite3.h>
using namespace std;
```

```
using json = nlohmann::json;

sqlite3* db;



using Record = std::vector<std::string>;
using Records = std::vector<Record>;



int select_callback(void* p_data, int num_fields, char** p_fields, char**
p_col_names)
{
	Records* records = static_cast<Records*>(p_data);
	try {
		records->emplace_back(p_fields, p_fields + num_fields);
	}
	catch (...) {
		// abort select on failure, don't let exception propogate thru
sqlite3 call-stack
		return 1;
	}
	return 0;
}

vector<vector<string>>  sort_12(vector<vector<string>> arr, int size)
{
	vector<string> tmp;
	for (int i = 0; i < size - 1; ++i) // i - номер прохода
	{
		for (int j = 0; j < size - 1; ++j) // внутренний цикл прохода
		{
			if (stoi(arr[j + 1][1]) > stoi(arr[j][1]))
			{
				tmp = arr[j + 1];
				arr[j + 1] = arr[j];
				arr[j] = tmp;
			}
		}
	}
	return arr;
}



void gen_response(const Request& req, Response& res) {
	//cout << "================================================="<<endl;
	string score = req.get_param_value("SCORE");
	string name = req.get_param_value("name");
	int position=1;
	int record_score=-1;

	// Проверка на созданность бд
	sqlite3* DB;
	int exit = sqlite3_open("persons.db", &DB);

	char* errmsg = 0;
	string stmt = "SELECT * FROM PERSON";
```

```
        Records records;
        //cout << "start" << endl;
        int ret = sqlite3_exec(DB, stmt.c_str(), select_callback, &records,
&errmsg);
        vector<vector<string>> list_player;
        //cout << "start_1" << endl;
                        for (auto& record : records) { // ПОЛУЧЕНИЕ ПОЗИЦИИ
                                //cout << record[0] <<" "<<record[2]<< endl;
                                if (record[0]==name) // Если текущий игрок
                                {
                                        record_score = stoi(record[1]);
                                }
                                string r1 = record[0];
                                string r2 = record[1];
                                vector<string> trm;
                                trm.push_back(r1);
                                trm.push_back(r2);
                                list_player.push_back(trm);
                        }
                        //cout << "start_2" << endl;
                        if (record_score == -1) //  Значит новый игрок
                        {
                                record_score = stoi(score);
                        }
                        if (stoi(score)>record_score)
                        {
                                record_score = stoi(score);
                        }
                        vector<vector<string>>new_players_list =
sort_12(list_player, list_player.size());

                        //cout << "$$$$$$$$$$"<< endl;
                        int i = 1;
                        position = list_player.size();

                        //cout << "start2" << endl;
        for (auto& record : new_players_list) { // ПОЛУЧЕНИЕ ПОЗИЦИИ
                //cout << record[1] << " = "<<score << " = "<< i << " |
"<<(stoi(score) >= stoi(record[1])) <<endl;

                if (stoi(score) >= stoi(record[1]))
                {
                        position = i;
                        break;
                }
                i++;
        }

        //cout << "position" << position << endl;
        json j = { {"position",position},{"record_score",record_score} };

        sqlite3_close(DB);
                        res.set_content(j.dump(), "text/json; charset=UTF-8");//
Вывод позиции и рекорда


}
```

```
void gen_response_set(const Request& req, Response& res) {
	auto score = req.get_param_value("SCORE");
	string name = req.get_param_value("name");

	sqlite3* DB;
	int exit = sqlite3_open("persons.db", &DB);
	int last_record=0;

	char* errmsg = 0;
	string stmt = "SELECT * FROM PERSON";
	Records recordsD;
	int ret = sqlite3_exec(DB, stmt.c_str(), select_callback, &recordsD,
&errmsg);

	bool is_available = false;
	for (auto& record : recordsD) { // ПОЛУЧЕНИЕ ПОЗИЦИИ
		cout << name << record[0] << endl;
		if (name == record[0])
		{
			last_record = stoi(record[1]);
			is_available = true;
		}
		}
	cout <<is_available << "is" << endl;

	if (is_available) // update
	{
		cout << score << last_record;
		if (stoi(score) > last_record)
		{
			cout << "update" << endl;



			std::stringstream UpdateQuery;
			UpdateQuery << "UPDATE PERSON "
					"SET RECORD = '" << score
				<< "' WHERE LOGIN = '" << name<< "'";
			//cout << UpdateQuery.str();

			sqlite3_stmt* UpdateStmt;
			sqlite3_prepare(DB, UpdateQuery.str().c_str(),
UpdateQuery.str().size(), &UpdateStmt, NULL);

			cout << "Stepping Update Statement" << endl;
			if (sqlite3_step(UpdateStmt) != SQLITE_DONE) cout <<
"Didn't UpdateItem!" << endl;
		}
	}
	else { // insert
		cout << "insert" << endl;



		std::stringstream insertQuery;
		insertQuery << "INSERT INTO PERSON (LOGIN, RECORD)"
			" VALUES ('" << name
			<< "', " << score << ")";
		sqlite3_stmt* insertStmt;
```

```
                sqlite3_prepare(DB, insertQuery.str().c_str(),
insertQuery.str().size(), &insertStmt, NULL);

                cout << "Stepping Insert Statement" << endl;
                if (sqlite3_step(insertStmt) != SQLITE_DONE) cout << "Didn't
Insert Item!" << endl;

        }

        sqlite3_close(DB);
        res.set_content("1", "text/html");// Вывод статуса успешной операции


}

int main()
{
        setlocale(LC_ALL, "Russian");
        // Проверка на созданность бд
        sqlite3* DB;
        std::string sql = "CREATE TABLE PERSON("
                "LOGIN          TEXT    NOT NULL, "
                "RECORD         TEXT     NOT NULL );";
        int exit = 0;
        exit = sqlite3_open("persons.db", &DB);
        char* messaggeError;
        exit = sqlite3_exec(DB, sql.c_str(), NULL, 0, &messaggeError);

        printf("Autocommit: %d\n", sqlite3_get_autocommit(DB));
        //if (exit != SQLITE_OK) {
        //      std::cerr << "Создание таблицы" << std::endl;
        //      sqlite3_free(messaggeError);
        //}
        //else {
        //      std::cout << "Таблица уже создана" << std::endl;
        //}
        sqlite3_close(DB);


        //string str = "SELECT * " + to_string(exit) + "WHERE";
        //cout << str;



        Server svr;
        svr.Post("/rating", gen_response);    // Виджет
        svr.Post("/set_rating", gen_response_set);    // Виджет
        std::cout << "Start server... OK\n";
        svr.listen("localhost", 3000);


        }
```