

Министерство образования и науки Российской Федерации  
Федеральное государственное автономное образовательное учреждение высшего образования «Уральский федеральный университет имени первого Президента России Б.Н. Ельцина» (УрФУ)

Институт радиоэлектроники и информационных технологий – РТФ

Кафедра «Радиоэлектроники информационных систем»

Оценка

Руководитель курсового

проектирования: Мирвода Сергей Геннадьевич

Члены комиссии

Дата защиты 13.01.2024

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

по теме: «Разработка игры про сражение 18 века»

по дисциплине: «Языки и методы программирования»

Студенты: Лагуткин Андрей Александрович

Минеев Анатолий Сергеевич

Группа: РИ-311055

Екатеринбург

2024



Министерство образования и науки Российской Федерации  
Федеральное государственное автономное образовательное учреждение высшего образования «Уральский федеральный университет имени первого Президента России Б.Н. Ельцина» (УрФУ)

Институт радиоэлектроники и информационных технологий – РТф

Кафедра «Радиоэлектроники информационных систем»

**Задание**

**на курсовой проект/работу**

Студенты Лагуткин Андрей Александрович, Минеев Анатолий Сергеевич

Группа РИ-311055

Специальность/направление подготовки Информационно-аналитические системы безопасности

1.Тема курсового проекта/ работы Разработка игры про сражение 18 века

2.Содержание проекта/работы, в том числе состав графических работ и расчетов Архитектурные диаграммы (архитектурная, функциональная, пакетов, последовательности), физическая и концептуальная модель БД, ключевые моменты кода \_

3.Дополнительные сведения \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

4. План выполнения курсового проекта/работы

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование элементов  проектной работы | Сроки | Примечания | Отметка о выполнении |
| Постановка задачи | 05.11.2023-20.11.2023 |  |  |
| Оформление архитектурных диаграмм | 21.11.2023-14.12.2023 | Архитектурная, функциональная диаграммы; диаграммы пакетов, последовательности |  |
| Создание БД | 15.12.2023-17.12.2023 | Также заполнение БД данными |  |
| Реализация продукта | 18.12.2023-08.01.2024 |  |  |

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/С. Г. Мирвода/

Министерство образования и науки Российской Федерации

ФГАОУ ВО «УрФУ имени первого Президента России Б.Н.Ельцина»

**РЕЦЕНЗИЯ**

на курсовую работу (проект)

Студента Лагуткина Андрея Александровича, Минеева Анатолия Сергеевича\_\_\_\_ группы РИ-311055\_\_\_\_

(фамилия имя отчество)

Тема курсовой работы: Разработка игры про сражение 18 века \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Модуль/дисциплина Языки и методы программирования

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

1 Соответствие результатов выполнения работы целям и задачам курсового проектирования, результатам обучения по дисциплине/модулю\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_

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

2 Оригинальность и самостоятельность выполнения работы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

3 Полнота и глубина проработки разделов \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

4 Общая грамотность и качество оформления текстового документа и графических материалов\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

5 Вопросы и замечания \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

6 Общая оценка работы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

Сведения о рецензенте:

Ф.И.О. \_\_\_\_\_\_

Должность \_\_\_\_

Место работы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Уч. звание \_\_\_\_\_  Уч. степень \_\_\_\_\_

**Подпись \_\_\_\_\_\_\_\_\_\_\_\_\_ Дата**

# Содержание

[Введение 5](#_Toc156000459)

[Видение проекта 5](#_Toc156000460)

[Функциональная диаграмма 5](#_Toc156000461)

[1 Проектирование 6](#_Toc156000462)

[1.1 Постановка задачи 6](#_Toc156000463)

[1.2 Архитектура продукта 6](#_Toc156000464)

[1.3 База данных 7](#_Toc156000465)

[1.4 Функциональность продукта 8](#_Toc156000466)

[1.5 Требования к системному ПО 9](#_Toc156000467)

[2 Разработка 10](#_Toc156000468)

[2.2 Создание базы данных 10](#_Toc156000469)

[2.3 Реализация продукта 10](#_Toc156000470)

[2.3.1 Использованные библиотеки 10](#_Toc156000471)

[2.3.2 Комментарии к поставленной задаче 10](#_Toc156000472)

[2.3.3 Пользовательский интерфейс 11](#_Toc156000473)

[2.3.4 Нереализованные функции 12](#_Toc156000474)

[2.4 Диаграмма пакетов 12](#_Toc156000475)

[Заключение 13](#_Toc156000476)

[Приложение А 14](#_Toc156000477)

# Введение

## Видение проекта

Kriegspiel – пошаговая стратегия для двух игроков, позволяющая им почувствовать себя в роли полководцев 18 века. У пользователей будет возможность управлять армией, состоящей из пехотных отрядов и отрядов артиллерии. Благодаря нашему продукту игроки смогут получить знания об оружии прошлых лет. В основе нашего проекта стоят мотивы интеллектуального удовлетворения пользователя, так как каждое принятое решение влияет на исход сражения. Мы стремимся подарить игрокам возможность стать частью великих битв, военных стратегий, создавая эпические моменты и впечатления.

## Функциональная диаграмма

Функциональная диаграмма продукта как он видится представлена на рисунке 1.

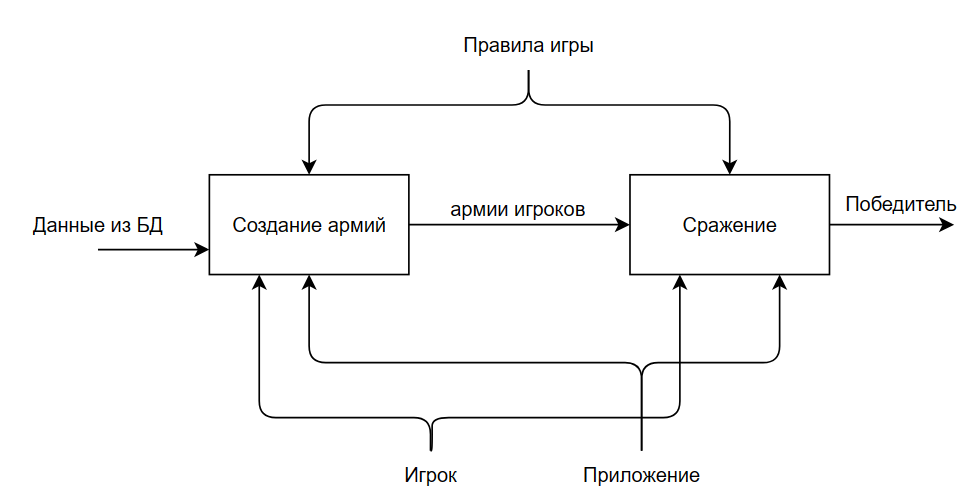


Рисунок 1 – Функциональная диаграмма продукта как он видится

# 1 Проектирование

## 1.1 Постановка задачи

Каждый игрок выбирает страну, за которую он будет играть. Каждая страна имеет уникальные отряды, а также цвет (по форме мундиров).

Армии игроков состоят из отрядов, которые могут быть пехотными или артиллерийскими. Пехотный отряд состоит из солдат, расположенных в шеренгах по 30 человек. Артиллерийский отряд состоит из трёх орудий и их расчётов, состоящих из 6 человек.

К пехотным отрядам относятся линейная пехота, лёгкая пехота и лейб-гвардия. Отряд линейной пехоты состоит из 90 человек. Одновременно стрелять могут только первые две шеренги (60 человек). Отряд лёгкой пехоты состоит из 60 человек. От линейной пехоты он отличается большей дальностью стрельбы. Лейб-гвардия представляет собой линейную пехоту с лучшими характеристиками. Одновременно стрелять могут первые три шеренги (90 человек).

Игроки по очереди управляют своими армиями: в свой ход игрок может совершить предусмотренные действия с каждым из своих отрядов. Пехотный отряд в течение хода может выполнить залп либо переместиться и (или) развернуться. Артиллерийский отряд не может передвигаться или поворачиваться. В течение хода игрока он может произвести залп ядрами или картечью.

В случае столкновения пули и солдата оба объекта исчезают, в случае столкновения ядра и солдата, исчезает только солдат. Также ядро может уничтожить артиллерийское орудие.

Если в пехотном отряде осталось меньше 20 солдат, отряд разбегается. Если в армии игрока не осталось ни одного пехотного отряда, игрок считается проигравшим.

## 1.2 Архитектура продукта

Продукт состоит из 2-х частей: приложения и базы данных. Подобный подход позволяет упростить и ускорить создание продукта, а также является достаточным для выполнения необходимых функций продуктом.

Приложение включает в себя пользовательский интерфейс, модуль доступа к базе данных и вычислительный модуль.

Пользовательский интерфейс должен выводить пользователю информацию о положении отрядов на карте, а также осуществлять взаимодействие пользователя со своей армией.

Вычислительный модуль должен вычислять координаты объектов на карте, обрабатывать взаимодействие этих объектов.

Модуль доступа к базе данных должен обеспечивать передачу данных из базы данных в приложение.

Использование базы данных позволяет добавлять расширения в готовый продукт (например, новые страны и отряды).

## 1.3 База данных

Концептуальная схема базы данных, выполненная в нотации Мартина, представлена на рисунке 2.1.

Изображение выглядит как текст, диаграмма, снимок экрана, линия

Автоматически созданное описание

Рисунок 2.1 – Концептуальная схема БД

Таблица countries содержит информацию о странах, которые игрок может выбрать. В ней имеются следующие атрибуты:

* country\_id – идентификационный номер страны;
* country\_name – название страны;
* r
* g
* b

Атрибуты r, g, b служат для описания цвета отрядов данной страны в модели RGB, где r, g, b – это соответственно красный, зелёный, синий.

Таблица infantry содержит информацию об отрядах пехоты. В ней имеются следующие атрибуты:

* infantry\_id – идентификационный номер отряда;
* infantry\_name – название отряда;
* country\_id – идентификационный номер страны, которой принадлежит отряд;
* count – количество солдат в отряде;
* distance – дальность стрельбы:
* way – максимальное расстояние, которое отряд может пройти за ход;
* error – максимальное отклонение пули в градусах;
* guard – указатель, определяющий, является ли отряд гвардейским.

Таблица artillery содержит информацию об отрядах пехоты. В ней имеются следующие атрибуты:

* artillery\_id – идентификационный номер отряда;
* artillery\_name – название отряда;
* country\_id – идентификационный номер страны, которой принадлежит отряд;
* core\_distance – дальность стрельбы ядрами;
* core\_error – максимальное отклонение ядра в градусах;
* grapeshot\_count – количество пуль в заряде картечи;
* grapeshot\_distance – дальность стрельбы картечью;
* grapeshot\_error – максимальное отклонение пуль в градусах.

## 1.4 Функциональность продукта

Продукт должен соответствовать функциональной диаграмме и диаграмме последовательности, представленным на рисунках 2.2 и 2.3 соответственно.

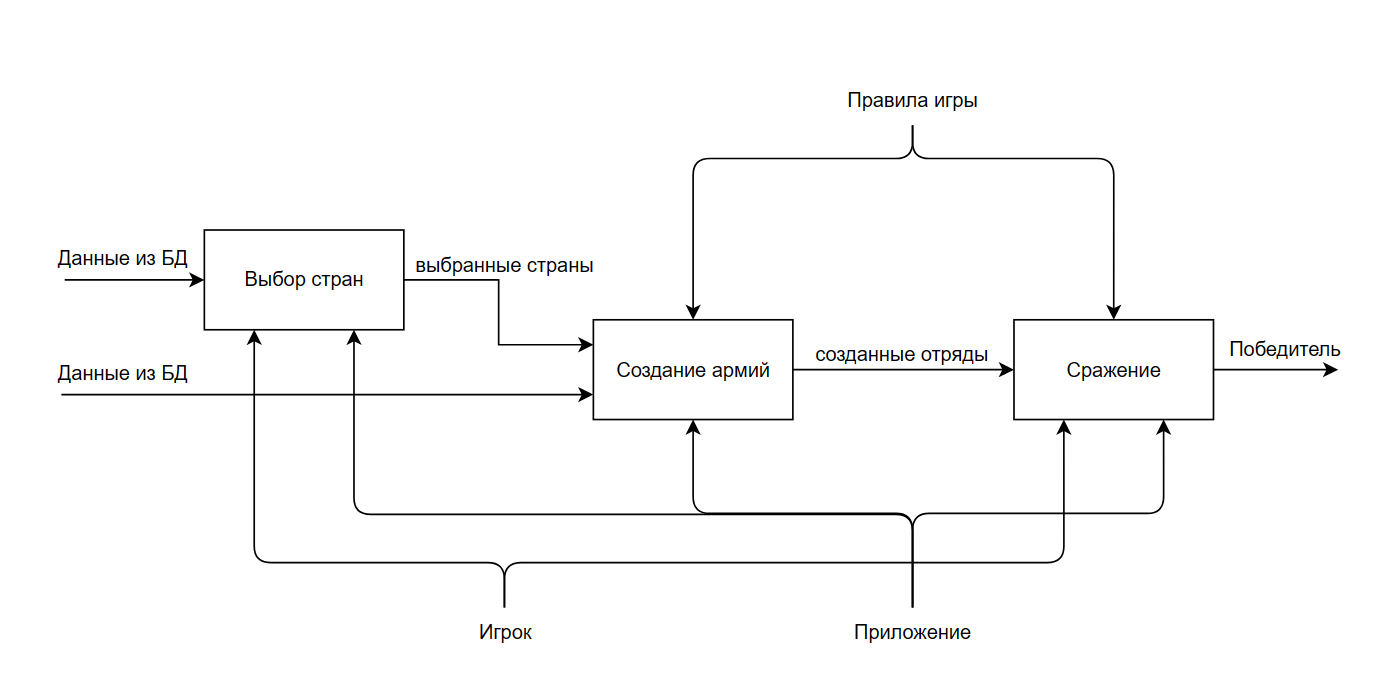


Рисунок 2.2 – Функциональная диаграмма

Изображение выглядит как текст, диаграмма, Параллельный, План

Автоматически созданное описание

Рисунок 2.3 – Диаграмма последовательности.

## 1.5 Требования к системному ПО

Продукт предназначен для использования на персональных компьютерах с операционной системой Windows 10 или Windows 11. Для использования продукта необходима реляционная СУБД.

# 2 Разработка

**2.1 Выбор языков программирования**

В качестве основного языка программирования был выбран Python. Данный язык программирования поддерживает ООП, а его относительная простота позволит существенно ускорить процесс разработки продукта.

В качестве СУБД был выбран PostgreSQL. Данная СУБД поддерживает стандарты SQL, является реляционной, что полностью отвечает требованиям к системному ПО.

## 2.2 Создание базы данных

Диаграмма физической модели базы данных представлена на рисунке 3.1.

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

Рисунок 3.1 – Физическая модель БД

## 2.3 Реализация продукта

### 2.3.1 Использованные библиотеки

При реализации продукта были использованы следующие библиотеки:

* Pygame – в качестве игрового движка;
* Psycopg2 – для связи приложения с базой данных;
* Numpy – для работы с тригонометрическими функциями;
* Random – для генерации случайных чисел.

Код продукта представлен в приложении А.

### 2.3.2 Комментарии к поставленной задаче

Пехотный отряд может поворачиваться на 45°, при этом используется 25 очков перемещения, таким образом за ход пехотный отряд может развернуться на 180°.

Гвардейские отряды ведут стрельбу поочерёдно каждой шеренгой, в остальных отрядах стрельбу ведут одновременно первые две шеренги.

Действия с отрядом осуществляются в следующем порядке:

1. Выбор отряда с помощью клавиш «1», «2», …;
2. Выбор действия (для пехотного отряда: «q» - левый разворот, «e» - правый разворот, «w» - движение вперёд, «f» - выстрел; для артиллерийского отряда: «g» - выстрел картечью, ЛКМ в пределах сектора обстрела – выстрел ядром).

При смене хода отряды перестраиваются: солдаты с задних шеренг встают на места убитых солдат из передних шеренг. Перестроение синего отряда представлено на рисунке 3.3.

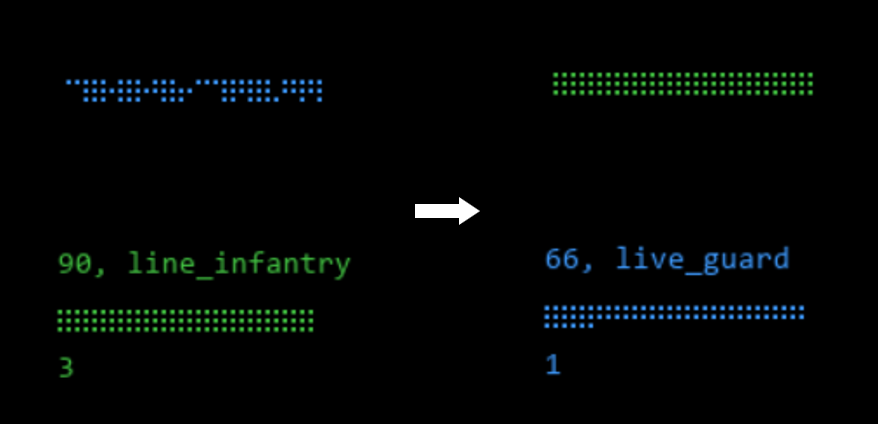


Рисунок 3.2 – Перестроение отрядов

### 2.3.3 Пользовательский интерфейс

В левой части экрана выводится список, показывающий состояние отрядов: 0 – отряд не совершал действие; 0.5 – отряд начал движение, есть возможность двигаться дальше; 1 – отряд выполнил действие. Под списком выводится номер выбранного в данный момент пехотного отряда.

Изображение выглядит как текст, Шрифт, часы, черный

Автоматически созданное описание

Рисунок 3.3 – Вывод состояния отрядов

В момент хода игрока пехотные отряды его армии подписываются следующим образом: число сверху – количество солдат в отряде, далее – тип отряда; число внизу – номер отряда.

Изображение выглядит как Шрифт, снимок экрана, дизайн

Автоматически созданное описание

Рисунок 3.4 – Подписи пехотного отряда

При выборе артиллерийского отряда на карте отображаются линии, показывающие сектор обстрела каждого орудия.

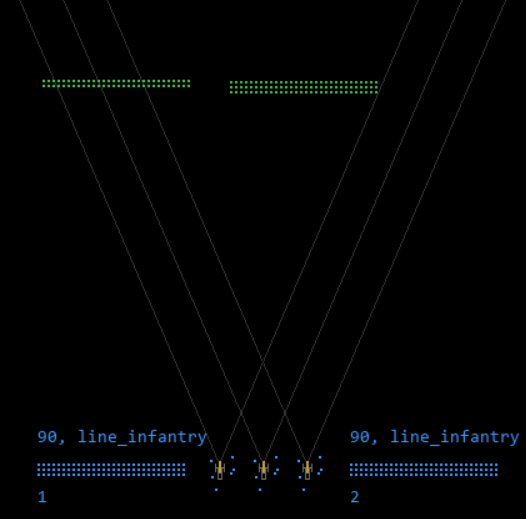


Рисунок 3.5 – Секторы стрельбы орудий

### 2.3.4 Нереализованные функции

1. Рукопашный бой – в данный момент можно пройти сквозь отряд противника;
2. Связь расчёта и орудия – в данный момент орудие продолжает стрелять даже при отсутствии артиллеристов;
3. Разворот артиллерийского отряда и его перемещение.

## 2.4 Диаграмма пакетов

Диаграмма пакетов представлена на рисунке 3.6.

Изображение выглядит как текст, диаграмма, План, Прямоугольник

Автоматически созданное описание

Рисунок 3.6 – Диаграмма пакетов

# Заключение

В ходе курсовой работы была разработана игра про сражение 18 века. Были созданы архитектурные диаграммы, была продумана архитектура продукта, осуществлён выбор языков программирования в соответствии с требованиями к ПО. По результатам проектирования был реализован продукт.

# Приложение А

# Выбор стран игроками

cur.execute("""SElECT country\_id, country\_name

FROM countries""")

query\_res = cur.fetchall()

for text in query\_res:

print(f'{text[0]} - {text[1]}')

print('Страна первого игрока (номер): ')

p1 = int(input())

p1\_country = query\_res[p1 - 1][1]

print('Страна второго игрока (номер): ')

p2 = int(input())

p2\_country = query\_res[p2 - 1][1]

cur.execute("""SELECT r, g, b from countries where country\_id = (%s)""", (p1,))

C1 = cur.fetchall()[0]

cur.execute("""SELECT r, g, b from countries where country\_id = (%s)""", (p2,))

C2 = cur.fetchall()[0]

running = True

units\_list = []

act\_list = [0, 0, 0, 0]

line\_list = [(0, 0), (0, 0), (0, 0)]

all\_sprites = pygame.sprite.Group()

units = pygame.sprite.Group()

units1 = pygame.sprite.Group()

units2 = pygame.sprite.Group()

units3 = pygame.sprite.Group()

units4 = pygame.sprite.Group()

units5 = pygame.sprite.Group()

units6 = pygame.sprite.Group()

batery1 = pygame.sprite.Group()

batery2 = pygame.sprite.Group()

cannons1 = pygame.sprite.Group()

cannons2 = pygame.sprite.Group()

bullets = pygame.sprite.Group()

cores = pygame.sprite.Group()

cannons = pygame.sprite.Group()

# создание отрядов пехоты первого игрока

add\_unit(units1, 450, 600, C1, 'live\_guard', p1)

dict\_1[0] = units1

add\_unit(units2, 700, 600, C1, 'line\_infantry', p1)

dict\_1[1] = units2

add\_unit(units3, 850, 600, C1, 'light\_infantry', p1)

dict\_1[2] = units3

# создание отрядов пехоты второго игрока

add\_unit(units4, 550, 600, C2, 'line\_infantry', p2)

dict\_2[0] = units4

add\_unit(units5, 800, 600, C2, 'line\_infantry', p2)

dict\_2[1] = units5

add\_unit(units6, 950, 600, C2, 'line\_infantry', p2)

dict\_2[2] = units6

for a in dict\_2.values():

for unit in a:

unit.rect.left = size[0] - unit.rect.left

unit.rect.bottom = size[1] - unit.rect.bottom

selected\_unit = units1

dict\_0 = dict\_1

dict\_n = dict\_2

cur.execute("""SELECT core\_distance, core\_error, grapeshot\_count, grapeshot\_distance, grapeshot\_error from artillery where country\_id = (%s)""", (p1,))

c\_dist, c\_er, g\_co, g\_dist, g\_er = cur.fetchall()[0]

cannon1 = Cannon(600, 600, 'resources/canon.png', c\_dist, c\_er, g\_co, g\_dist, g\_er)

add\_cannon(cannon1, cannons1)

cannon2 = Cannon(635, 600, 'resources/canon.png', c\_dist, c\_er, g\_co, g\_dist, g\_er)

add\_cannon(cannon2, cannons1)

cannon3 = Cannon(670, 600, 'resources/canon.png', c\_dist, c\_er, g\_co, g\_dist, g\_er)

add\_cannon(cannon3, cannons1)

cur.execute("""SELECT core\_distance, core\_error, grapeshot\_count, grapeshot\_distance, grapeshot\_error from artillery where country\_id = (%s)""", (p2,))

c\_dist, c\_er, g\_co, g\_dist, g\_er = cur.fetchall()[0]

cannon4 = Cannon(750, 185, 'resources/canon2.png', c\_dist, c\_er, g\_co, g\_dist, g\_er)

add\_cannon(cannon4, cannons2)

cannon5 = Cannon(785, 185, 'resources/canon2.png', c\_dist, c\_er, g\_co, g\_dist, g\_er)

add\_cannon(cannon5, cannons2)

cannon6 = Cannon(820, 185, 'resources/canon2.png', c\_dist, c\_er, g\_co, g\_dist, g\_er)

add\_cannon(cannon6, cannons2)

cannon4.down = True

cannon5.down = True

cannon6.down = True

artillery(600, 600, batery1, C1)

artillery(635, 600, batery1, C1)

artillery(670, 600, batery1, C1)

artillery(763, 600, batery2, C2)

artillery(728, 600, batery2, C2)

artillery(693, 600, batery2, C2)

selected\_canons = cannons1

c\_count = 0

for canon in selected\_canons:

line\_list[c\_count] = (canon.rect.left + 3, canon.rect.top + 2)

c\_count += 1

for unit in batery2:

unit.rect.left = size[0] - unit.rect.left

unit.rect.bottom = size[1] - unit.rect.bottom

score\_font = pygame.font.SysFont('consolas', 14)

score\_text = []

number\_text = []

action\_text = score\_font.render(str(act\_list), True, (White))

for unit in list(dict\_0.values()):

score\_text.append(score\_font.render(f'{len(unit.sprites())}', True, unit.sprites()[0].color))

number\_text.append(score\_font.render(f'{list(dict\_0.keys())[list(dict\_0.values()).index(unit)]}', True, unit.sprites()[0].color))

while running:

clock.tick(FPS)

k = list(dict\_0.keys())[list(dict\_0.values()).index(selected\_unit)]

for event in pygame.event.get():

if event.type == QUIT:

running = False

# Выстрел из пушки

elif event.type == MOUSEBUTTONDOWN and canon\_select == True and act\_list[3] == 0:

ziel = event.pos

for canon in selected\_canons:

canon.fire = True

canon.comand = pygame.time.get\_ticks()

canon\_select = False

act\_list[3] = 1

elif event.type == KEYDOWN:

# Смена хода

if event.key == K\_RETURN:

for unit in dict\_0.values():

new\_position(unit)

for sprite in all\_sprites:

sprite.rect.left = size[0] - sprite.rect.left

sprite.rect.bottom = size[1] - sprite.rect.bottom

for unit in dict\_n.values():

new\_position(unit)

for canon in cannons:

if canon.down == False:

canon.image = pygame.image.load('resources/canon2.png')

canon.down = True

else:

canon.image = pygame.image.load('resources/canon.png')

canon.down = False

canon.rect.left -= 8

canon.rect.bottom += 15

if (player1 == True):

dict\_0 = dict\_2

dict\_n = dict\_1

selected\_unit = units4

selected\_canons = cannons2

player1 = False

else:

dict\_0 = dict\_1

dict\_n = dict\_2

selected\_unit = units1

selected\_canons = cannons1

player1 = True

for a in dict\_0.values():

for u in a:

u.way = 0

act\_list = [0, 0, 0, 0]

line\_list = []

for canon in selected\_canons:

line\_list.append((canon.rect.left + 3, canon.rect.top + 2))

# Выбор отряда

elif event.key == K\_1:

selected\_unit = dict\_0[0]

elif event.key == K\_2:

selected\_unit = dict\_0[1]

elif event.key == K\_3:

selected\_unit = dict\_0[2]

elif event.key == K\_4 and act\_list[3] == 0:

canon\_select = True

# Выстрел отряда пехоты

elif event.key == K\_f and act\_list[k] == 0 and canon\_select == False:

j = 0

for u in selected\_unit:

if u.guard == True:

u.time = random.randint(0, 50)

u.fire = True

u.comand = pygame.time.get\_ticks() + 800 \* (j // 30)

j += 1

elif random.randint(0, 100) >= 15 and j < 60:

u.fire = True

u.comand = pygame.time.get\_ticks()

j += 1

act\_list[k] = 1

# Выстрел картечью

elif event.key == K\_g and canon\_select == True and act\_list[3] == 0:

for canon in selected\_canons:

canon.comand = pygame.time.get\_ticks()

canon.grapeshot\_fire = True

canon\_select = False

act\_list[3] = 1

elif event.key == K\_w and act\_list[k] != 1 and canon\_select == False:

go\_check = True

elif event.key == K\_e and act\_list[k] != 1 and canon\_select == False:

i\_un = 0

for u in selected\_unit:

if u.way <= 75 and i\_un == 0:

povorot(selected\_unit)

act\_list[k] = 0.5

u.way += 25

elif u.way <= 75:

u.way += 25

i\_un += 1

elif event.key == K\_q and canon\_select == False:

i\_un = 0

for u in selected\_unit:

if u.way <= 75 and i\_un == 0:

left\_povorot(selected\_unit)

act\_list[k] = 0.5

u.way += 25

elif u.way <= 75:

u.way += 25

i\_un += 1

elif event.type == KEYUP:

if event.key == K\_w:

go\_check = False

all\_sprites.update()

if go\_check == True:

act\_list[k] = 0.5

for u in selected\_unit:

if (u.way < u.max\_way):

u.x\_d, u.y\_d = u.rect.center

u.go()

else:

go\_check = False

act\_list[k] = 1

# Проверка столкновений:

pygame.sprite.groupcollide(units, bullets, True, True)

pygame.sprite.groupcollide(units, cores, True, False)

pygame.sprite.groupcollide(cannons, cores, True, False)

for i in range(len(dict\_0.values())):

if (len(list(dict\_0.values())[i].sprites())) > 0:

score\_text[i] = (score\_font.render(f'{len(list(dict\_0.values())[i].sprites())}, {list(dict\_0.values())[i].sprites()[0].type}', True, list(dict\_0.values())[i].sprites()[0].color))

number\_text[i] = (score\_font.render(f'{i + 1}', True, list(dict\_0.values())[i].sprites()[0].color))

for group in units\_list:

if len(group) < 20:

for unit in group:

unit.image.fill(White)

unit.rect = unit.rect.move (unit.run\_speed)

if unit.rect.top <= 0 or unit.rect.left <= 0 or unit.rect.right >= size[0]:

unit.kill()

p1\_count = len(units1) + len(units2) + len(units3)

p2\_count = len(units4) + len(units5) + len(units6)

if (p2\_count < 20):

running = False

print(f'{p1\_country} has won this battle!')

if (p1\_count < 20):

running = False

print(f'{p2\_country} has won this battle!')

screen.fill(BLACK)

if (canon\_select == True):

for line in line\_list:

pygame.draw.line(screen, Gray2, line, (line[0] - 0.43 \* line[1], 0), 1)

pygame.draw.line(screen, Gray2, line, (line[0] + 0.43 \* line[1], 0), 1)

all\_sprites.draw(screen)

for i in range(len(dict\_0.values())):

if (len(list(dict\_0.values())[i].sprites())) > 0:

screen.blit(score\_text[i], (list(dict\_0.values())[i].sprites()[0].rect.left, list(dict\_0.values())[i].sprites()[0].rect.bottom - 30))

screen.blit(number\_text[i], (list(dict\_0.values())[i].sprites()[0].rect.left, list(dict\_0.values())[i].sprites()[-1].rect.bottom + 10))

action\_text = score\_font.render(str(act\_list), True, (White))

p1\_text = score\_font.render(str(p1\_count), True, (C1))

p2\_text = score\_font.render(str(p2\_count), True, (C2))

k\_text = score\_font.render(str(k + 1), True, (White))

screen.blit(action\_text, (30, 20))

screen.blit(p1\_text, (1400, 20))

screen.blit(p2\_text, (1400, 40))

screen.blit(k\_text, (30, 40))

pygame.display.update()

pygame.time.delay(500)

pygame.quit()