Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

«Московский государственный технический университет имени Н.Э. Баумана»

(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ Информатика и системы управления

КАФЕДРА Программное обеспечение ЭВМ и информационные технологии

Р А С Ч Ё Т Н О - П О Я С Н И Т Е Л Ь Н А Я З А П И С К А

к курсовому проекту на тему:

Разработка сетевого игрового приложения

Студент _____ Шилов А.И.
(Подпись, дата) (И.О.Фамилия)

Руководитель курсового проекта _____ Рогозин Н.О.
(Подпись, дата) (И.О.Фамилия)

Москва, 2019

Оглавление

Введение

Разработка компьютерных игр занимает особую нишу в IT-отрасли. Валовой оборот игровой индустрии в 2017 году составил около $116 млрд. и этот показатель продолжает увеличиваться с каждым годом. В индустрии насчитывается около 14 тысяч компаний, от стартапов до крупнейших игроков, имеющих известность по всему миру, в которых заняты сотни тысяч человек, причем многие из них имеют профессии, специфичные для данной области разработки: игровой программист, игровой дизайнер, дизайнер уровней, игровой продюсер, игровой художник и тестер игр. Результатами их труда пользуются около двух миллиардов человек, играющих в разработанные ими игры.

Индустрия имеет множество различных направлений. В первую очередь, различают разработку под различные платформы: для персональных компьютеров, для игровых приставок (Xbox, PlayStation, Nintendo и т.д.), для мобильных устройств и для браузеров. В случае коммерческого успеха игры компания, ее создавшая, может принять решение о поддержке игры для платформы, для которой она изначально не разрабатывалась, таким образом игра становится мультиплатформенной.

Ключевым аспектом игры является ее жанр. От него во многом зависит то, каким будет процесс игры в целом, какова будет его механика. Для каждого жанра существуют свои практики и приемы для создания атмосферы игры, увлекающей игрока, интересной сюжетной линии, зрелищных сражений и прочих игровых действий.

Еще одним немаловажным свойством игры является количество игроков. В настоящее время наиболее популярными вариантами являются одиночные игры или многопользовательские онлайн-игры. Также существуют многопользовательские на одном компьютере, по локальной сети или оффлайн.

Многопользовательские онлайн игры являются наиболее сложными с точки зрения реализации, так как необходимо принять во внимание

эффективность взаимодействия множества компьютеров, гарантировать корректность данных в условиях перебоев в работе сети и т.д. В рамках данного курсового проекта будет реализована многопользовательская игра по локальной сети, что по данной классификации наиболее приближено к онлайн-играм. С точки зрения жанра игра будет являться имитацией пошаговой настольной игры, а целевой платформой будут персональные компьютеры.

1. Аналитический раздел

В данном разделе рассматриваются достоинства и недостатки различных топологий сетей и протоколов транспортного уровня передачи данных.

1.1 Топологии сетей

В настоящее время в локальных сетях используются следующие физические топологии:

- Шина;
- Звезда;
- Кольцо.

1.1.1 Шина

Топология данного типа представляет собой общий кабель (называемый шина или магистраль), к которому подсоединены все рабочие станции. На концах кабеля находятся терминаторы, для предотвращения отражения сигнала.

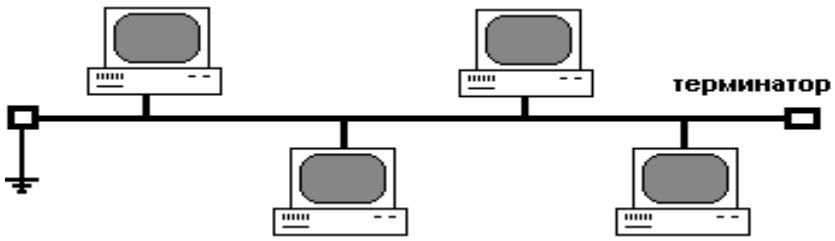

Рис. 1. Топология «шина»

Преимущества сетей шинной топологии:

- расход кабеля существенно уменьшен
- сеть легко настраивать и конфигурировать;
- отказ одного из узлов не влияет на работу сети в целом;

Недостатки сетей шинной топологии:

- разрыв кабеля может повлиять на работу всей сети;
- ограниченная длина кабеля и количество рабочих станций;

1.1.2 Звезда

В сети, построенной по топологии типа «звезда», каждая рабочая станция подсоединяется к концентратору, или хабу (рис. 2). Концентратор обеспечивает параллельное соединение ПК и, таким образом, все компьютеры, подключенные к сети, могут общаться друг с другом.

Данные от передающей станции сети передаются через хаб по всем линиям связи всем ПК. Информация поступает на все рабочие станции, но принимается только теми станциями, которым она предназначается.

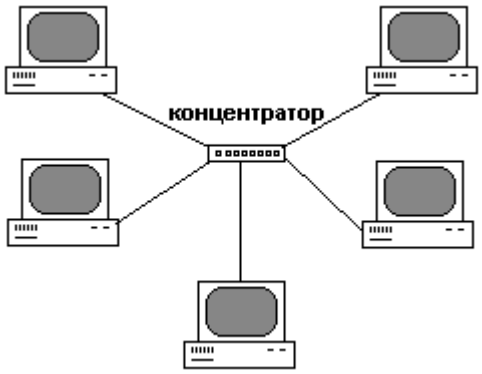

Рис. 2. Топология «звезда»

Преимущества сетей топологии звезда:

- легко подключить новый ПК;
- имеется возможность централизованного управления;
- сеть устойчива к неисправностям отдельных ПК и к разрывам соединения отдельных ПК.

Недостатки сетей топологии звезда:

- отказ хаба влияет на работу всей сети;
- большой расход кабеля.

1.1.3 Кольцо

В сети с топологией типа «кольцо» все узлы соединены каналами связи в неразрывное кольцо, по которому передаются данные (рис. 3). Выход одного ПК соединяется со входом другого ПК. Начав движение из

одной точки, данные, в конечном счете, попадают на его начало. Данные в кольце всегда движутся в одном и том же направлении.

Принимающая рабочая станция распознает и получает только адресованное ей сообщение. В сети с топологией типа физическое кольцо используется маркерный доступ, который предоставляет станции право на использование кольца в определенном порядке.

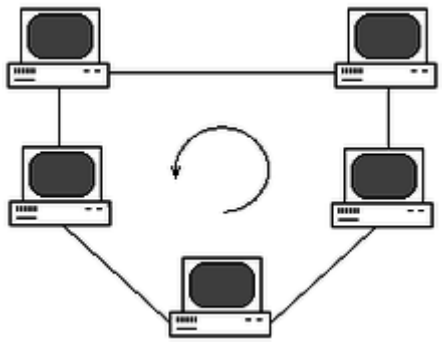

Рис. 3. Топология «кольцо»

Преимущества сетей топологии кольцо:

- сеть легко создать и настраивать.

Недостатки сетей топологии звезда:

- повреждение линии связи в одном месте приводит к неработоспособности всей сети.

1.1.4 Смешанная топология

В крупных сетях с произвольными связями между компьютерами можно выделить отдельные произвольно связанные фрагменты (подсети), имеющие типовую топологию, поэтому их называют сетями со смешанной топологией (рис. 4).

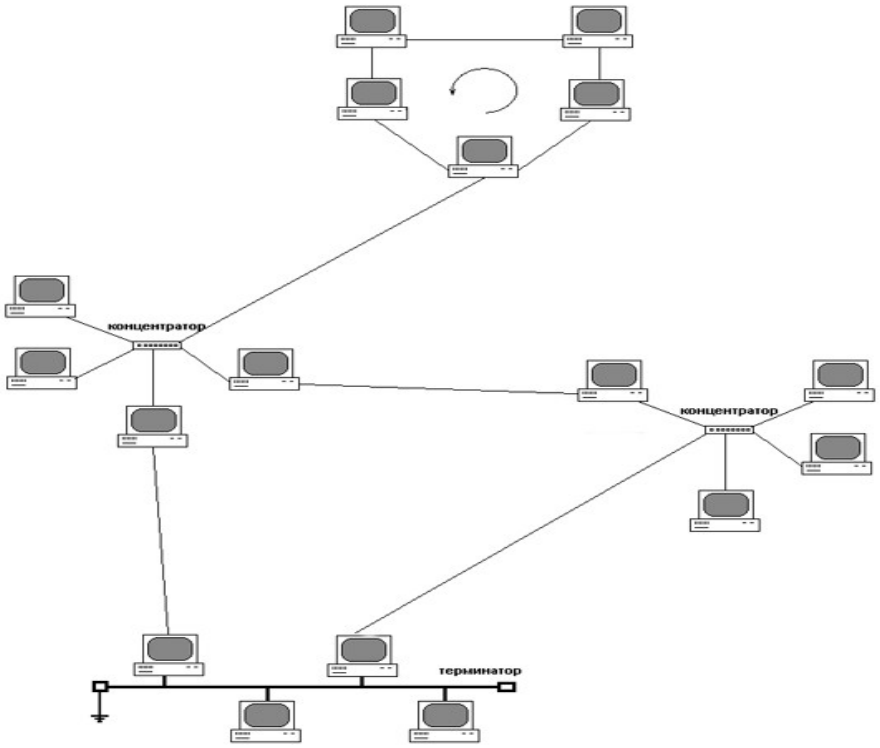

Рис. 4. Смешанная топология

1.2 Протоколы передачи данных транспортного уровня

Протоколы транспортного уровня модели OSI отвечают за обеспечение надёжной передачи данных от отправителя к получателю, защиту от потери или дублирования пакетов данных, контроль целостности данных, порядок поступления пакетов.

Для локальной сети в основном используют два протокола:

- TCP;
- UDP.

1.2.1 Протокол TCP

Протокол, ориентированный на наличие соединения между участниками обмена информацией, для создания которого им требуется выполнить «тройное рукопожатие». Как только соединение установлено, пользователи могут отправлять данные в обоих направлениях. Обладает следующими преимуществами:

- Надёжность — TCP управляет подтверждением, повторной передачей и тайм-аутом сообщений. Производятся многочисленные попытки доставить сообщение. Если оно потеряется на пути, сервер вновь запросит потерянную часть. В TCP нет ни пропавших данных, ни (в случае многочисленных тайм-аутов) разорванных соединений.
- Упорядоченность — если два сообщения последовательно отправлены, первое сообщение достигнет приложения-получателя первым. Если участки данных прибывают в неверном порядке, TCP отправляет неупорядоченные данные в буфер до тех пор, пока все данные не могут быть упорядочены и переданы приложению.

Ценой этих преимуществ является один существенный недостаток:

- Тяжеловесность — TCP необходимо три пакета для установки сокет-соединения перед тем, как отправить данные. TCP следит за надёжностью и перегрузками, что тоже создает дополнительные накладные расходы.

1.2.2 Протокол UDP

Простейший протокол, реализующий минимальный функционал отправки и получения сообщений, гарантирующий целостность данных только в пределах одного пакета. В противоположность протоколу TCP имеет недостатки:

- Ненадёжный — когда сообщение посылается, неизвестно, достигнет ли оно своего назначения — оно может потеряться по пути. Нет таких понятий, как подтверждение, повторная передача, тайм-аут.
- Неупорядоченность — если два сообщения отправлены одному получателю, то порядок их достижения цели не может быть предугадан.

Единственным преимуществом перед TCP является большая скорость обмена данными, что, однако, в некоторых ситуациях является более весомым аргументом при выборе используемого протокола.

1.3 Выводы

Для реализации данного курсового проекта в качестве используемой топологии сети выбрана смешанная топология на основе топологии звезды. Клиенты будут подключаться к серверу и взаимодействать с ним для авторизации, поиска оппонентов и сохранения результатов игры, а в процессе игры будут подключаться друг к другу напрямую. Такая реализация выбрана, во-первых, в целях изучения возможностей построения топологии более сложной, чем типовая «звезда», во-вторых, потому что передаваемые между клиентами данные не требуют синхронизации и обработки со стороны сервера.

В качестве протокола передачи данных будет использован TCP. Разрабатываемое приложение не имеет никаких требований относительно быстродействия, поэтому нет никакой необходимости в выборе протокола UDP.

2 Конструкторский раздел

Данный раздел содержит описание реализованного самостоятельно протокола прикладного уровня и его применения для обмена данными между сервером и клиентским приложением и между клиентскими приложениями.

2.1 Описание протокола прикладного уровня

Основной идеей протокола является передача данных в формате JSON. Преимуществом этого формата является его распространенность и поддержка со стороных многих языков программирования, возможность представления сложных многоуровневых структур данных. В протоколе HTTP для определения длины поступившего сообщения используется заголовок Content-Length, по аналогии с этим механизмом было решено перед посылкой сообщения произвольной длины посылать сообщение фиксированной длины 4 байта, содержащее количество байтов в следующем сообщении. Таким образом, алгоритм посылки сообщения можно представить в виде следующей схемы (рис. 5):

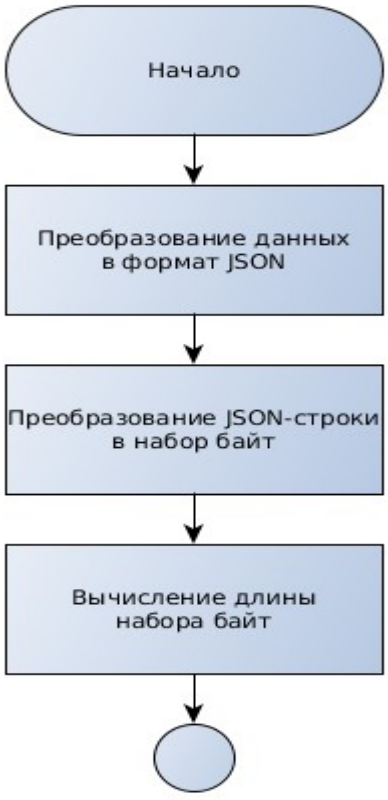

Рис. 5. Схема алгоритма посылки сообщения

В алгоритме получения сообщения происходят обратные действия и преобразования (рис. 6):

Рис. 6. Схема алгоритма получения сообщения

2.2 Описание применения протокола прикладного уровня

Посылаемые сообщения разделяются на два типа: команды и ответы на выполнение команды. Оба типа являются JSON-объектами. Для сообщений-команд единственным обязательным полем которого является поле command, остальные поля опциональны и содержат информацию, необходимую для выполнения команды принимающей стороной. Для сообщений-ответов все поля являются опциональными. В случае, когда ответ является пустым, т.е. не содержит полей, он может не посылаться вовсе. Пример листинга программы, демонстрирующего использование данного протокола, приведен на рисунке 7:

```
# вызывающая сторона
command = {
    'command': 'SOME_COMMAND',
    'some_structured_data': {
        'array': [1, 2, 3],
        'str': 'str'
    }
}
protocol.send_data(socket, command)
result = protocol.recv_data(socket)

# принимающая сторона
command = protocol.recv_data(socket)
if command['command'] == 'SOME_COMMAND':
    result = work_with_data(command['some_structured_data'])
    protocol.send_data(socket, result)
elif command['command'] == 'ANOTHER_COMMAND':
    ...
```

Рис. 7. Пример использования протокола прикладного уровня

3 Технологический раздел

Данный раздел содержит обоснованный выбор средств программной реализации и руководство пользователя к клиентскому приложению.

3.1 Выбор средств реализации

Для реализации проекта был выбран язык программирования Python. Он чрезвычайно удобен как в процессе чтения, так и написания программ благодаря изящному минималистичному синтаксису, богатой стандартной библиотеке, а также множеству сторонних модулей и расширений.

В качестве базы данных была выбрана MongoDB. Она является документоориентированной и не требует описания схемы таблиц, поэтому предоставляет большую гибкость чем традиционные SQL базы данных.

Для реализации графического интерфейса клиентского приложения был использован фреймворк PyQt. Она является адаптацией под язык Python известного фреймворка Qt, написанной на C++. PyQt поддерживает практически весь функционал Qt и соответствует его документации, благодаря чему является мощным и удобным в использовании графическим фреймворком при разработке на Python.

3.2 Руководство пользователя

Клиентское приложение содержит возможности регистрации и авторизации (рис. 10), просмотра таблицы чемпионов (рис. 11), интерфейс игрового поля (рис. 12):

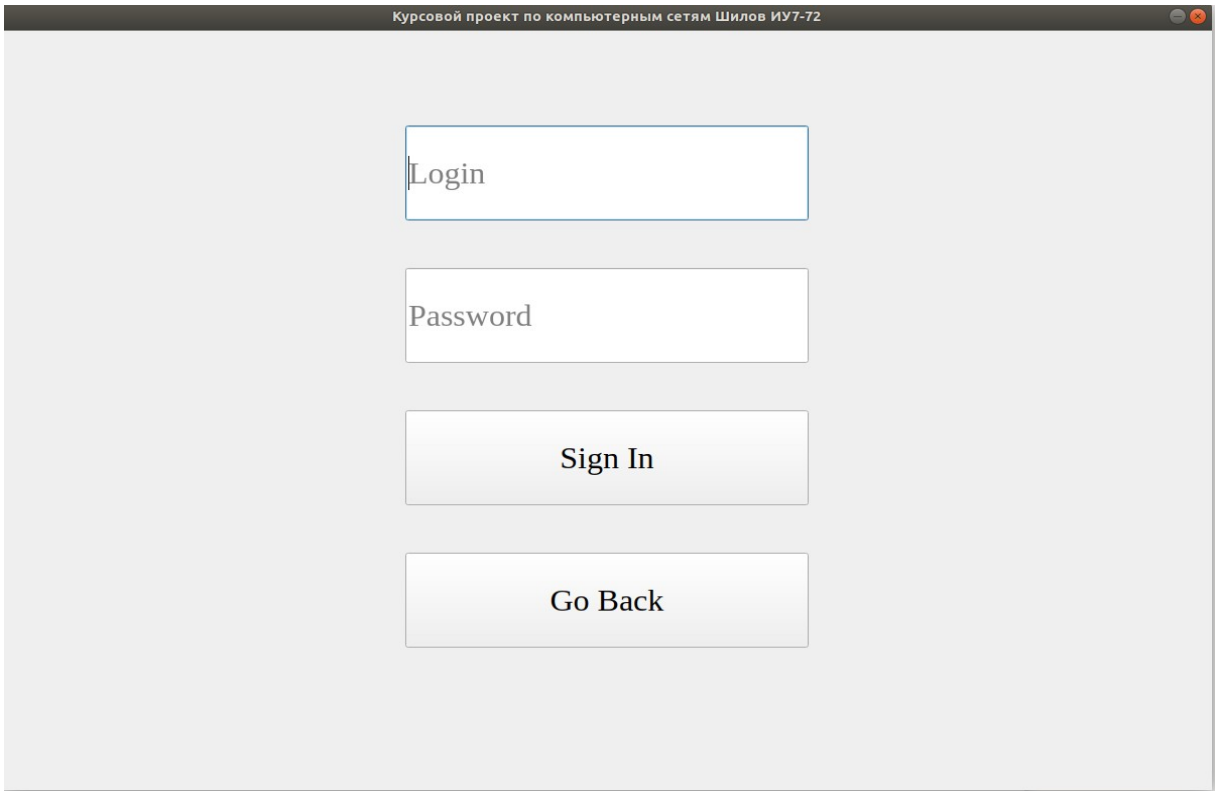

Рис. 10. Окно авторизации

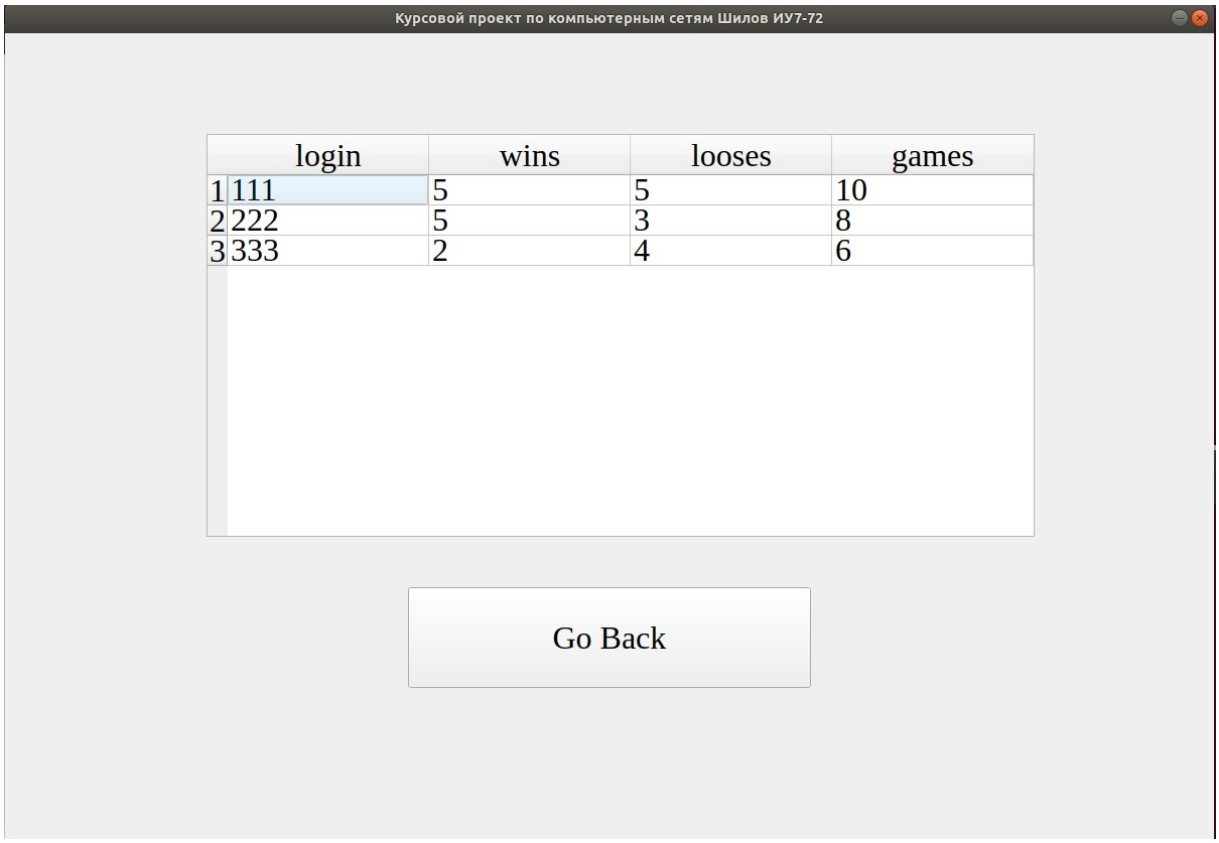

Рис. 11. Таблица чемпионов

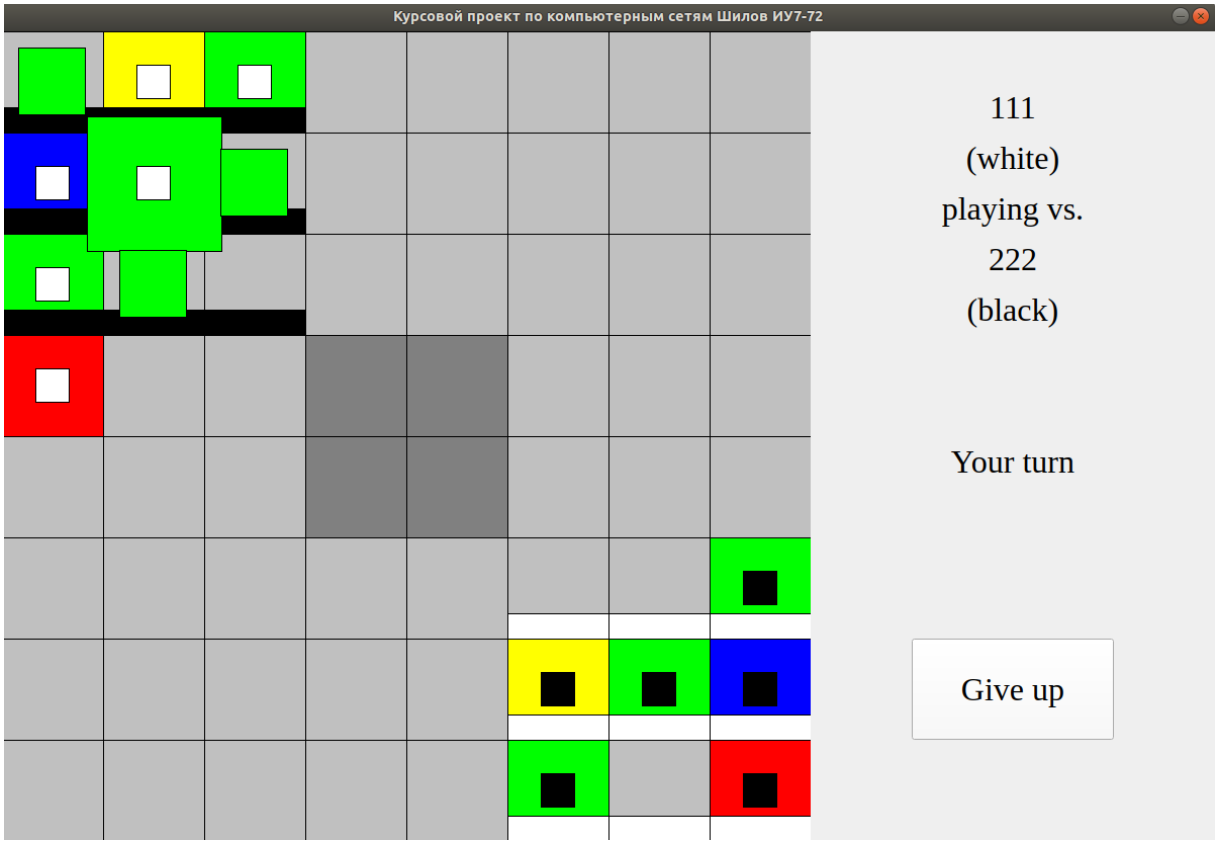

Рис. 12. Игровое поле

Цель игры состоит в том, чтобы передвинуть все подконтрольные игроку фигуры, помеченные квадратом черного или белого цвета, на клетки соответствующего им цвета, помеченные прямоугольником. За один ход имеется возможность передвинуть одну фигуру.

Существуют фигуры четырех типов, обозначенные различными цветами. Правила их передвижения таковы:

- зеленая фигура может передвинуться на соседнюю клетку, с которой соприкасается ребром одна или более других дружественных фигур;
- желтая фигура может передвинуться на соседнюю клетку, с которой соприкасаются ребром две или более других дружественных фигур;

- синяя фигура может перепрыгнуть на клетку через одну или более дружественных фигур;
- красная фигура может перепрыгнуть на клетку через две или более дружественных фигур.

Кроме того, не все клетки игрового поля, не занятые фигурами, являются пустыми: они могут заняты препятствиями.

Интерфейс имеет подсказки для выделенной фигуры (ее размер увеличивается в 1.33 раза при выделении), для возможных ходов выделенной фигуры (обозначается фигурой того же цвета масштабом 0.66), выводится информация об игроках и о том, чей сейчас ход, в правой панели окна.

Заключение

В результате проделанной работы было реализовано сетевое игровое приложение, работающее в пределах локальной сети. Рассмотрены и проанализированы преимущества и недостатки различных топологий сетей и сетевых протоколов транспортного уровня. Реализован самостоятельно протокол прикладного уровня, обеспечивающий гибкий и удобный обмен сложноструктурированными данными. Разработан графический интерфейс клиентского приложения и работа с базой данных на стороне сервера в соответствии с техническим заданием.

Список использованной литературы

1. Рогозин Н.О. Курс лекций по дисциплине «Компьютерные сети»
2. Python documentation. URL: https://docs.python.org/3
3. Qt documentation. URL: http://doc.qt.io/qt-5.11/index.html
4. MongoDB documentation. URL: https://docs.mongodb.com