ГУАП

КАФЕДРА №51

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ
ПРЕПОДАВАТЕЛЬ

_____	_____	_____
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЕТ ПО КУРСОВОЙ РАБОТЕ

СЕТЕВАЯ ИГРА ТЕННИС

по курсу: ОСНОВЫ ПРОГРАММИРОВАНИЯ II

РАБОТУ ВЫПОЛНИЛ
СТУДЕНТ ГР. № 5511

_____	_____
подпись, дата	инициалы, фамилия

Санкт-Петербург 2017

1 Функциональная спецификация

1.1 Цель работы

Создание сетевой игры теннис (пинг-понг). Две полоски с двух сторон корта — игроки. Игрок может перемещаться вверх или вниз. Корт сверху и снизу окружен кирпичной стеной. Мячик отскакивает от стены с полным сохранением импульса.

Программа состоит из двух частей — серверной и клиентской. Серверная часть обеспечивает связь между игроками и служит как модель (Model-View) приложения. Клиентская часть выполняет функции отображения процесса игры и передачи ввода клиента серверу.

2 Руководство пользователя

1. При запуске программы в появившемся окне пользователю предлагается выбор — создать новую игру или присоединиться к существующей. Для любого из действий необходимо указать желаемое имя. Если имя уже занято, то при попытке начать игру пользователя попросят выбрать другое.

2. Для создания новой сессии пользователь должен ввести имя, счет, до которого будет вестись игра, и нажать New Session. Если подключение прошло успешно, пользователь должен ожидать подключения второго игрока.

3. Для подключения к существующей игре пользователь должен нажать Update для запроса информации о сессиях на сервер и затем выбрать одну из появившегося списка, ввести имя и нажать Play.

4. Для того чтобы подключиться к игре по имени другого игрока, пользователь должен ввести его и нажать Connect. Если свободная игра с игроком с таким именем найдена, игра начнется. Если нет — покажет соответствующее сообщение.

5. При начале игры появляется игровое поле. Поле представляет собой прямоугольник, по краям которого находятся прямоугольные платформы — "ракетки". Каждый клиент управляет своей, может перемещать ее вверх или вниз кнопками W и S соответственно.

6. Игра начинается с запуска мячика из центра в случайном направлении. Цель игры — двигая свою платформу не дать мячику коснуться своего края поля. Пропустивший очко начинает новый раунд запуском мяча путем нажатия пробела.

7. Для выхода из игры нужно нажать ESC.

Ниже приведены скриншоты, расширяющие руководство пользователя.

2.1 Главное меню игры

При запуске программы пользователя встретит следующее окно.

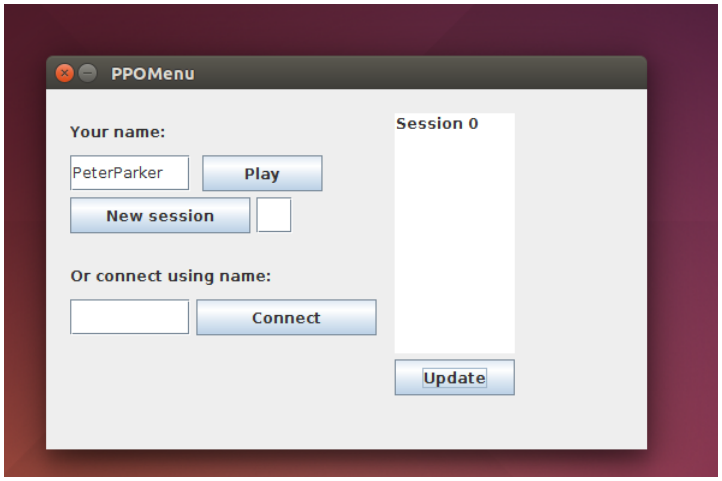

Рис. 1: Меню, открывающееся при запуске программы

2.2 Ввод данных

На картинке снизу приведен пример заполнения всех полей. При нажатии New Session будет создана новая игра до трех очков. При нажатии Play будет осуществлена попытка подключиться к сесси с именем Session 2. При нажатии Connect клиент попытается присоединиться к сессии с игроком под именем Batman.

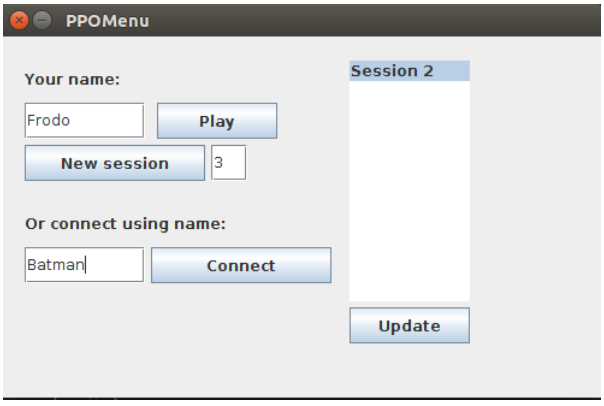

Рис. 2: Пример заполнения данных

2.3 Ожидание подключения

После создания новой сессии откроется окно с игрой, как показано на картинке снизу. Игра будет в режиме ожидания, что отображается надписью Awaiting connection на месте имени противника, пока не подключится второй игрок.

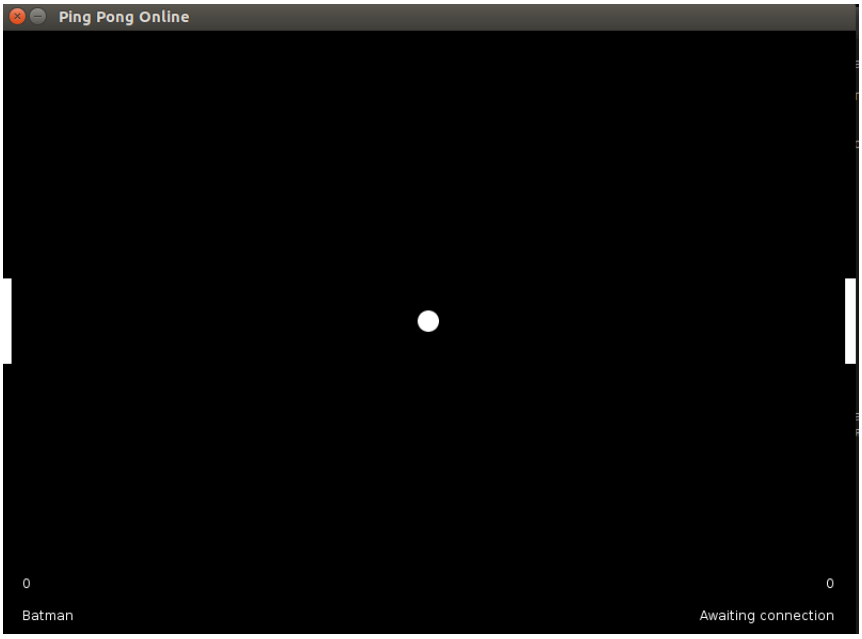

Рис. 3: Ожидание подключения

2.4 Игровой процесс

После присоединения второго игрока, надпись Awaiting connection заменится его именем. Тогда, игрок, создавший сессию, может начать игру нажатием пробела. На скриншоте изображен момент, когда оба игрока забили по два очка и теперь забивший последним должен нажать пробел чтобы развести мяч.

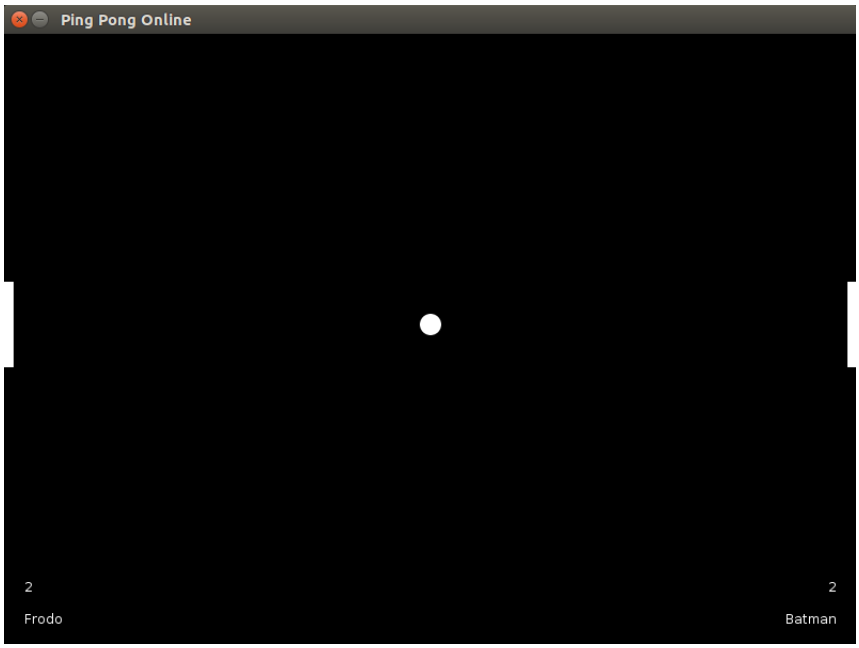

Рис. 4: Процесс игры

3 Архитектура программы

Программа разбита на три модуля — Server, Client и Model.

3.1 Сервер

- Исполнение функций модели игры
- Обеспечение связи между игроками
- Поддержка нескольких игровых сеансов одновременно
- Использует протокол UDP

3.2 Клиент

- Создание игровой сессии с другим клиентом через сервер
- Отображение присылаемых сервером данных о состоянии игры при помощи подмодуля View
- Передача ввода клиента серверу для обработки
- Возможность присоединиться к существующей игре или начать новую
- Использование протокола UDP

3.3 Протокол подключения клиента к серверу

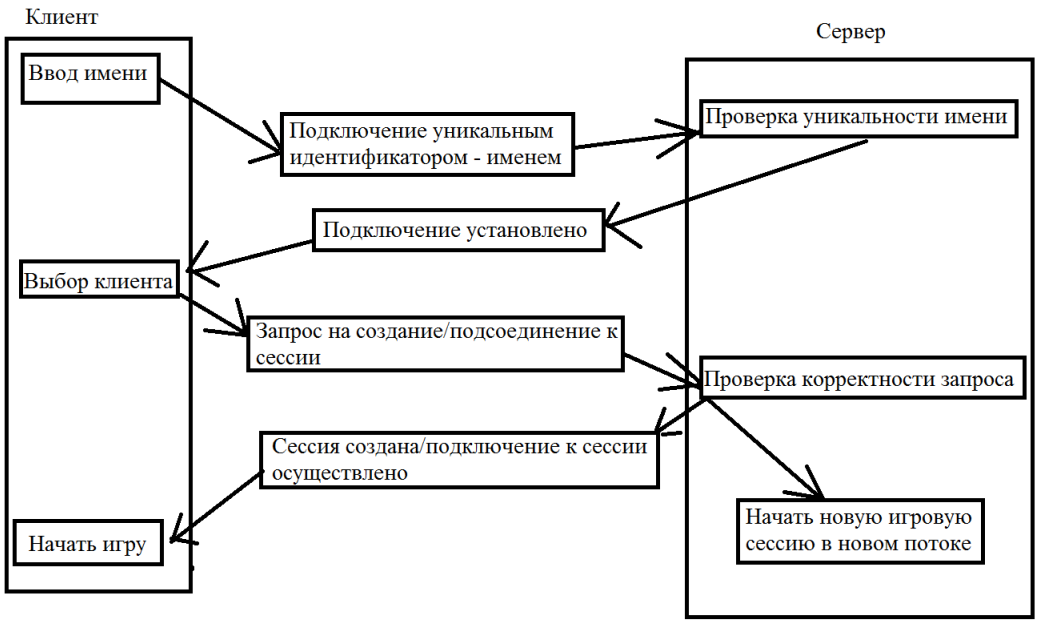

Рис. 5: Схема протокола подключения к игре

Изначально клиент посылает запрос на подключение серверу. Если подключение успешно, сервер сообщает об этом клиенту. Затем, в зависимости от выбора пользователя, клиент посылает серверу запрос на создание сессии или подключение к существующей. Если запрос принят, сервер сообщает об этом клиенту и создает сессию/подключает клиента к существующей. Далее идет обмен информацией между клиентом и сервером.

3.4 Структура связи клиента и сервера во время игры

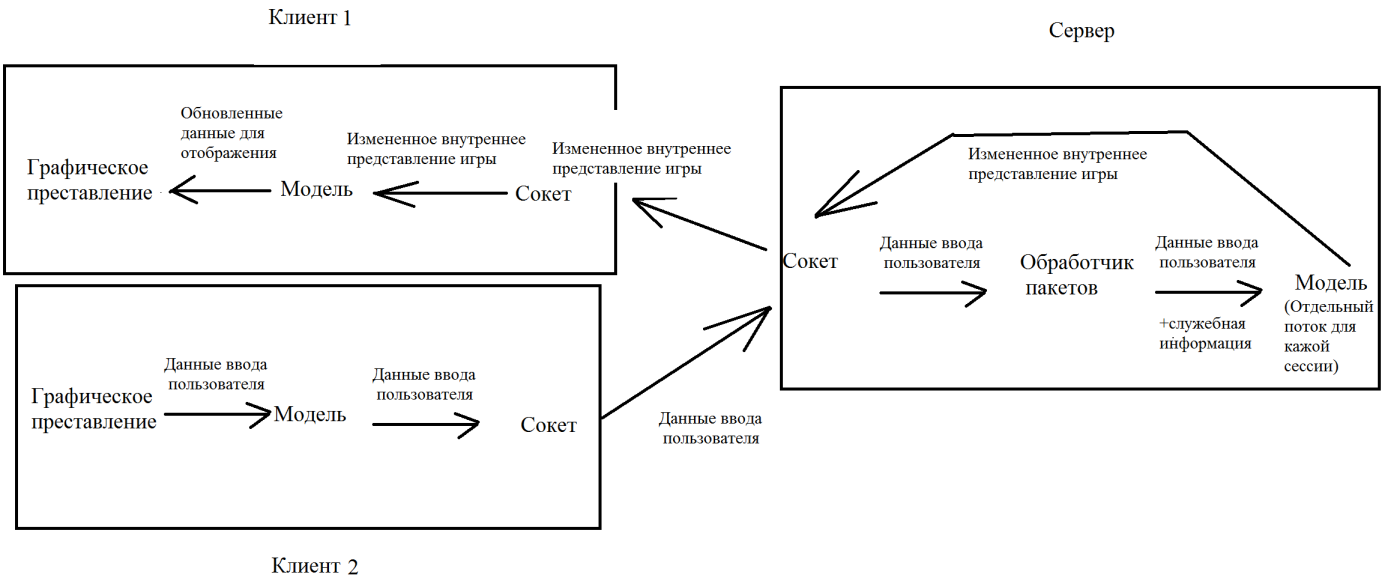

Рис. 6: Схема протокола общения клиента и сервера во время игры

На картинке сверху изображен пример взаимодействия двух участвующих в игре клиентов и сервера. Клиент отсылает серверу информацию о вводе пользователя. Тот, в свою очередь, обрабатывает пакет, устанавливая от кого он пришел и какой сессии его передать. Затем, сессия обновляет модель игры и информация об измененном состоянии посылается другому клиенту для отображения.

4 Основные классы и методы в реализации

4.1 Model

Класс — основной класс модели. Хранит информацию о поле, игроках, мяче, счете и прочем.

- `public PPOModel(int maxScore, int totalSpeed, int playerSpeed, String name)` — конструктор класса. Принимает параметры игры.
- `public void update()` — метод для обновления состояния игры. Симулирует движение игроков и мяча.

4.2 Server

Класс `PPOServer` — главный класс программы сервера

- `public static void main` — входная точка сервера, создает и запускает все необходимые службы.

Класс `InputHandler` — обработчик пакетов. Распределяет пришедшие пакеты по сессиям и обрабатывает соединения с клиентами. Работает в отдельном потоке.

- `public void handlePacket(String str))` — метод для передачи пакетов обработчику. Записывает пришедшие пакеты в LinkedBlockingQueue — очередь, предназначеную для многопоточных программ.
- `public void run()` — главный метод обработчика, забирает пакеты из очереди и обрабатывает их.

Класс `SocketWrapper` — обертка над сокетом. Принимает пакеты и передает их обработчику. Так же используется для отправки данных клиентам. Прослушивание сокета идет в отдельном потоке.

- `public void run()` — основной метод, слушает сокет и передает данные обработчику.
- `public static void send(InetAddress addr, int port, String message)` — метод для отправки сообщения клиенту.

Класс `GameSession` — класс, хранящий информацию о отдельной игровой сессии. Хранит в себе модель игры и обновляет ее, так же отвечает за отправку изменений состояния игры клиентам. Передача пакетом этому классу так же происходит при помощи LinkedBlockingQueue.

- `public void run()` — основной метод, обновляет состояние игры, передает ввод клиентов модели. Переодически отправляет данные о состоянии игры клиентам.
- `public static void addPacket(String packet)` — метод для добавления пакета в очередь на обработку.
- `addConnection(UDPConnection connection)` — метод для установления связи клиента с сессией.

Класс `UDPConnection` — класс, хранящий информацию об отдельном клиенте, включая адрес, порт, имя, сессию и прочее.

- `UDPConnection(String addr, int port, String name)` — конструктор, принимает адрес и порт клиента, а так же его имя.
- `void handlePacket(String packet)` — передает пакет на обработку сессии, к которой подключен.
- `void setSession(GameSession session)` — метод для установки связи с сессией.

4.3 Client

Класс `UDPConnection` — главный класс программы клиента

- `public static void main` — входная точка клиента, создает и запускает все необходимые службы.

Класс `Connector` — класс для связи с сервером. Отправляет и принимает пакеты.

- `Connector(InetAddress addr, int port)` — конструктор, принимает адрес и порт сервера.
- `public void send(String str)` — метод для отправки пакета серверу.
- `public String receive()` — метод, принимающий пакеты от сервера.

Класс `PPOGame` — класс, запускающий игру.

- `public void run()` — запуск потоков, отвечающих за отображение и обновление локальной модели игры на основи информации, поступающей от сервера.

Класс `ModelUpdater` — класс, обрабатывающий пакеты от сервера и обновляющий модель на их основе. Работает в отдельном потоке

- `public void run()` — принимает пакеты и на их основе обновляет модель.

Класс `ViewUpdater` — класс, обновляющий отображение. Так же является слушаетелем ввода пользователя, который и передает серверу.

- `public void run()` — обновляет отображение.
- `public void keyPressed(KeyEvent e))` и `public void keyPressed(KeyEvent e))` — методы, вызывающиеся при нажатии пользователем кнопок. Передают информацию о нажатиях серверу.

4.4 Подмодуль View

Класс `PPOMenuView` — класс, отвечающий за интерфейс главного меню

- `public void run()` — создает окно главного меню.

Класс `PPOGameView` — класс, отвечающий за отображение игры.

- `public PPOGameView(PPOModel model, int w, int h)` — конструктор класса, принимает модель, которую будет отрисовывать, и размеры окна. Создает окно с игрой.
- `public void render()` — метод, производящий рендеринг по модели.
- `public void draw()` — метод отрисовки состояния игры.