**Пояснительная Записка**

**Бакулин И.Д. и Кириллов И.Д**

Целью проекта является реализация знакомой игры Тетрис на языке python с возможностью играть онлайн через создание собственного отдельного сервера. Проект был вдохновлен популярным аналогом игры Тетриса [TETR.IO](https://tetr.io/).

Были использованы следующие библиотеки: pygame, socket, threading, pickle и random.

Проект разделён на 2 главные части: одиночная игра (основной игровой процесс) и серверная составляющая игры. Также дополнительно добавлены настройки для улучшения игрового процесса.

Игровой процесс полностью сходен с оригинальным Тетрисом. Инструкцию по игре можно найти в свободном доступе, но есть небольшие изменения в управлении:

* Перемещение происходит на стрелочки, изменение положения фигуры на стрелочку “вверх”.
* При нажатии на “пробел” фигура становится на позицию, которая обозначается тенью на игровом поле.
* Справа от основного поля находится очередь, из которой будут браться фигуры для постановки на поле, считая сверху вниз.
* При нажатии клавиши “shift” фигура перемещается на удержание, таким образом, при повторном нажатии фигуру можно будет поставить вместо той, что была взята в очереди справа от игрового
* Так же в поле «Очки» подсчитываются очки за полностью заполненные линии. А поле «Уровень» показывает сложность, с которой будут опускаться фигуры.

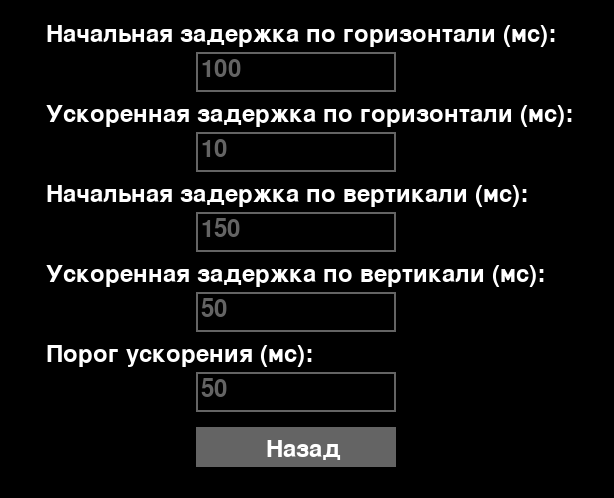
**Интерфейс одиночной игры:**



**Настройки**

Игра считывает зажатие клавиш движения фигуры горизонтально и вертикально. Это улучшает пользовательский опыт и более точной установки блоков. Но так как при зажатии клавиши, каждый кадр в который нажата клавиша будет считываться игрой нужно ограничить количество считываний. Мы решили ограничить количество движений на кадр используя таймер, который бы считал разницу между временем начала зажатия клавиши и текущим временем – таким образом мы бы получили время, которое клавиша была зажата. Если мы знаем время, которое клавиша была зажата, то можно сравнить это с «задержкой зажатия», чтобы начать повторное движение блока, после единичного переноса. Так же, про прохождению большего количества времени зажатия клавиши, фигура должна ускорять свое движение, что так же добавило точности в управление фигуры. Отсюда появляется новая переменная: «Ускоренная задержка».  
Чтобы пользователь мог настроить необходимую ему задержку, мы добавили меню настроек, в котором можно установить необходимые значения задержки в миллисекундах. Класс InputBox имеет в себе аргумент allowed\_chars, который фильтрует нажатые клавиши и воспринимает только разрешенные.

**Интерфейс окна настроек:**



**Серверная часть**

Серверная часть состоит из нескольких функций и из двух частей – клиентской и серверной, серверная часть состоит из:

* start()
* notify\_waitiing()
* handle\_client()
* broadcast()
* remove\_client()
* stop()

Функция start() отвечает за запуск и дальнейшее провождение игры по мере подключения двух игроков, а так же после нее.

notify\_waiting() отвечает за сообщения клиенту об ожидании подключения второго игрока

handle\_client() отвечает за пересылку данных между двумя игроками во время игры

broadcast() функция, которая отвечает за пересылку сообщений между двумя клиентами, пересылая сообщение всем клиентам, кроме отправителя

remove\_client() отвечает за успешное отключение клиента от сервера

stop() отвечает за выключение сервера, после отключения двух игроков

И вторая часть – клиентская, в которой основой класс, который работает с сетью BattleClient, а класс игры BattleTetrisGame работает с pygame и синхронизацией поля противника. В BattleClient присутствуют базовые класс для работы с подключением к серверу, функции для подключения, приема данных, отправления данных и отключения.

Для синхронизации данных нужно отправлять данные на сервер, далее с сервера на клиент – для этого мы решили использовать библиотеку pickle для кодирования данных для передачи по сети. Данная библиотека передает объекты ЯП python напрямую, после декодирования которые можно использовать без дополнительных изменений данных. Так же, при декодировании данных на сервере или клиенте, мы столкнулись с проблемой, при которой данные не передавались равномерно, и не весь байтовый код доходил до получателя в заданной длине передачи. Чтобы это исправить, мы использовали следующий способ:

Сначала, отправить кодирует длину отправляемых данных в первых 10 байтах.  
Получатель считывает первые 10 байтов и получает данные необходимой длины, чтобы данные доставились в целости.

Для запуска сервера необходимо запустить файл server\_run.py, в котором аргумент класса BattleServer - host=0.0.0.0, для того чтобы айпи привязался ко всем возможным айпи-адресам в сети, указать заведомо открытый порт в port=? (Мы использовали «эмулятор» локальной сети Radmin VPN, через который можно было подключиться через любой порт)