# 1 Сценарий

Главная цель игры передвигаться по полю, поедая корм и других игроков, которые имеют меньший размер, чем он. После того, как шарик съел некоторое количество, он начинает расти в диаметре. Основная задача выжить и попасть в турнирную таблицу.

При заходе в игру пользователь попадает в меню, где может:

* Зарегистрировать и начать игру;
* Просмотреть игровое поле;
* Подключить бота.

Серверная часть должна отвечать за обработку данных о координатах всех пользователей и рассылать всем клиентам обновленные результаты. Как только сервер будет запущен, он должен ожидать клиентов.

После старта клиента игроку нужно управлять шариком с помощью клавиатуры (↓, →, ↑, ←, s, d, w, a). Поле, по которому движется шарик, ограничено. В режиме демонстрации поля передвижение по карте будет происходить аналогично с помощью стрелок и выше представленных букв.   
Вместо человека в игре могут соревноваться боты. Для этого бот должен обладать простейшим искусственным интеллектом. Клиенты - боты и люди, могут связываться с сервером (для обмена информации между собой) через API.

# 2 Архитектура



Рисунок 1 — Архитектура приложения

# 3 Словарь

При разработке игры «Agar.io» используются следующие понятия:

* Игрок;
* Бот;
* Клавиатура;
* Дисплей;
* Игра;
* Рейтинговая таблица;
* Баллы.

## 3.1 Игрок

Взаимодействует с программой, обеспечивая тем самым выполнение своих целевых функций. Взаимодействие осуществляется через внешние устройства компьютера: клавиатуру и дисплей.

Основные действия игрока можно разделить на:

● Управление ходом игры;

● Регистрация (ввод имени);

● Просмотр игрового поля.

Прямого взаимодействие игрока с программой не происходит. Между ними существует посредник в виде внешних устройств компьютера: клавиатуры, монитора, которые обеспечивают преобразования физических воздействий человека в программные события посредством использования клавиатуры. Обратная связь осуществляется за счет визуализации изменения состояния программы на экране дисплея.

## 3.2 Бот

Взаимодействует с сервером через API. Главным отличием от игрока является заложение искусственного интеллекта, в котором будут использованы простейшие алгоритмы передвижения и взаимодействия с другими игроками.

## 3.3 Игра

Основной программный модуль, решающий целевую задачу. Взаимодействует с клавиатурой, реагируя изменением внутреннего состояния на посылаемые воздействия игрока. Игра является достаточно сложным понятием, которое можно рассматривать как композицию следующих дополнительных понятий:

● модель игры;

● вид игры;

● контроллер игры.

Подобное видение определяется одним из наиболее распространенных в настоящее время подходом к реализации интерактивных приложений на основе концепции модель-вид-контроллер.

## 3.4 Рейтинговая таблица

Рейтинговая таблица будет содержать в себе лучших 5 игроков, которые имеют большее количество баллов (очков). Игроки в таблице нужно расположить в порядке убывания их полученных очков, которые можно набрать, поедая на игровом поле пищу либо других игроков.

# 4 Диаграмма прецедентов



Рисунок 2 — Диаграмма прецедентов

# 5 Алгоритм работы программы

Описание состояний продемонстрировано на рисунке 3.

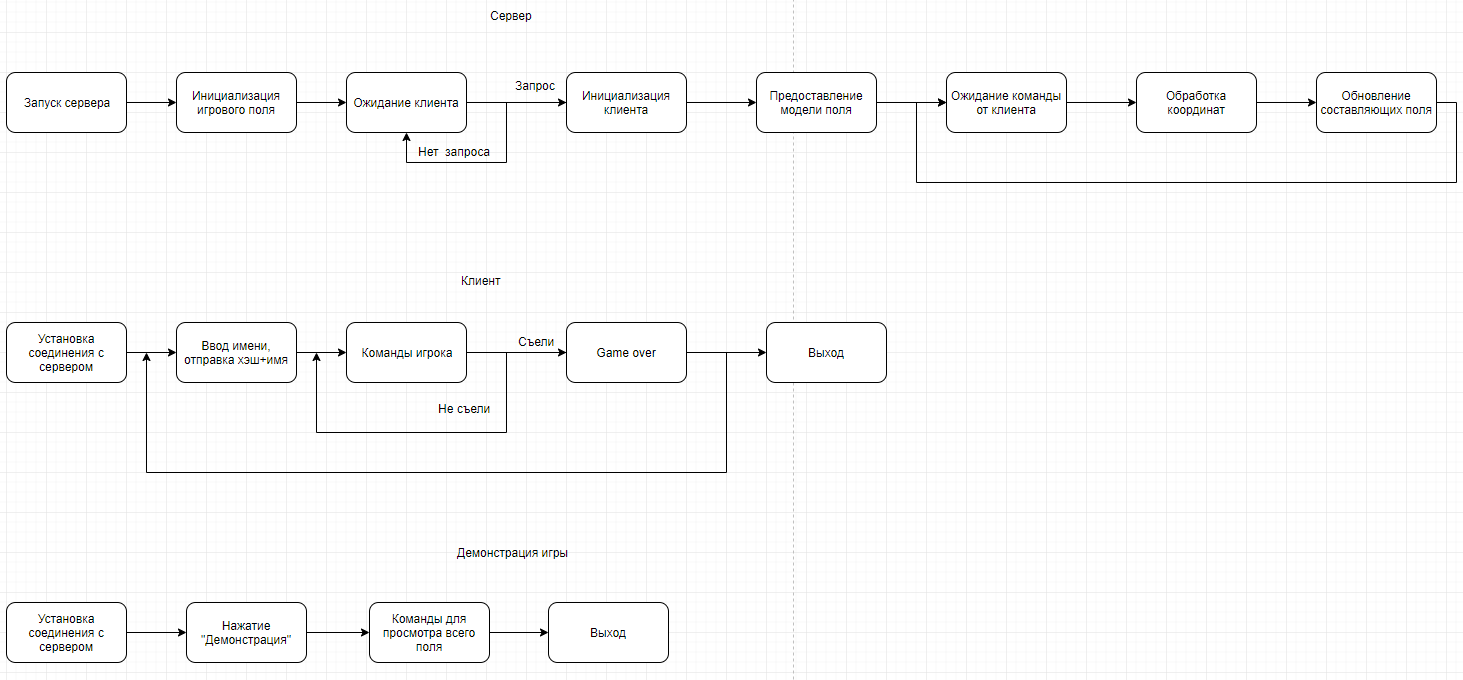


Рисунок 3 — Описание состояний

# 6 Инструкция по развертыванию системы

## 6.1 Установка программы и её компонентов

Скопировать репозиторий через ssh или https отсюда: https://gitlab.com/unidev/agario

Библиотека бота находится по адресу: <https://gitlab.com/prettyGoo/agario-python-bot>

## 6.2 Инсталлирование программного обеспечения для запуска а локальной системе

Для использования программы на локальной системе:

* Необходимо установить nodejs версии не ниже 8: <https://nodejs.org/en/> ;
* Установить все зависимости проекта: npm i;
* Запустить сервер: npm start.

## 6.3 Запуск внутри Docker

Docker — программное обеспечение для автоматизации развёртывания и управления приложениями в среде [виртуализации на уровне операционной системы](https://ru.wikipedia.org/wiki/%D0%92%D0%B8%D1%80%D1%82%D1%83%D0%B0%D0%BB%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F_%D0%BD%D0%B0_%D1%83%D1%80%D0%BE%D0%B2%D0%BD%D0%B5_%D0%BE%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%BE%D0%B9_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D1%8B).

Если вы используете Linux или macOS и не имеете nodejs, установленный локально, то вы можете запустить проект внутри Docker. Для этого необходимо выполнить команду docker-compose up, перед этим убедитесь, что Docker and Docker compose установлены на вашей системе. Вероятность, что докер заработает под Windows достаточно мала (но вы можете попытаться).

## 6.4 Запуск бота

PythonBot (<https://pypi.org/project/agario-bot/>)

Соответствующая папка может быть найдена в корне проекта. Она НЕ предназначена для запуска или прямого импорта, а лишь содержит для ознакомления код библиотеки для написания бота и содержит примеры. Если вы хотите использовать эту библиотеку для написания своего бота на Python, то она должна быть установлена через pip (лучше всего не засорять свой интерпретатор питона и устанавливать ее внутри virtualenv): pip install agario-bot. Примеры находятся в PythonBot/agario\_bot/examples/scary\_bot

Если вы хотите добавить какие-то изменения в саму библиотеку, то вам необходимо изменить setup.py (хотя бы название библиотеки), затем создать аккаунт на pypi.org, добавить логин и пароль в соотвествующий файл на системе для облегчения деплоя (о том, как это сделать, можно найти на сайте pypi) и выполнить команду make deploy. Затем вы можете установить свою измененную версию библиотеки через команду через pip install your-new-library-name