**Создание новой игры:**

* После накопления 8 игроков очередь автоматически создает новую комнату с интерфейсом RoyalRoom и передает ей свойства объекта виде QueueGamer (в базовом виде ссылка на ClientServer ник и пароль для возвращения в аккаунт)
* Передает созданную комнату в хранилище активных комнат Rooms
* Запускает комнату вызовом метода StartRoom

**Конструктор IRoom**

* Создает объект IRoomLogic
* Передаем объект IRoomLogic в качестве параметра конструктора INetwork

Класс RoomNetwork – управляет отправкой данных клиенту, содержит в себе список клиентов

**Движение игрока:**

* Начинается с получения сообщения GoToMsg (содержит в себе направление движения)
* При получении GoToMsg оно трансформируется в StartMoveGamer содержащее направление движения и добавляется в очередь сообщений объекта игрока
* После истечения кванта времени оно передается компоненту Movement
* Обработчик этого сообщения в Movement добавляет направление переданное в StartMoveGamer в currentDirection, компоненте переходим в состоянии active
* После перехода в активное состоянии, компонент на каждое сообщение TimeQuantPassed передвигает игрока на некоторое расстояние согласно текущему выбранному направлению.
* При получении сообщения EndMoveGamer компонент Movement переводит active в false(движение игрока прекращается)

**Механизм работы срочных сообщений**

* Базовое сообщение создается на уровне компонента и реализуется от IComponentMsg. после чего генерируется событие HappenedComponentMsg, вызывающий обработчик HandlerComponentEvent у игрового объекта
* Если это необходимо игровой объект собирает информацию у других компонентов и создается класс интерфейса IGameObjectEvent. После чего генерируется событие HappenedObjectEvent
* Событие обрабатывается методом HandlerObjectEvent класса RoyalRoomLogic и генерируется событие HappenedGameEvent
* При получении такого сообщения INetwork определяет, игроков, которые оказались в границах видимости данного события и уведомляет их об этом событии (конвертируя в нужный тип IMessage)

**Инвентарь. Поднятие предметов**

* Представлен компонентом Collector
* Содержит в себе две коллекции Модификаторы и Оружие
* При получении сообщения PickUpLootMsg мы получаем координаты игрока и обращаемся к карте. Если на площади тела игрока находится игровой объект (подходящего типа) мы добавляем его в инвентарь и удаляем его с карты. Отправляем этому объекту сообщение о том, что он подобран (содержит ссылка на объект хранящий этот объект).

**Оружие**

Содержит следующие компоненты:

* SolidBody – прозрачный
* Shot для создания выстрела(добавляет пулю в коллекцию игровых объектов) содержит в себе компонент Magazin
* Magazin – выполняет все действия происходящие с магазином

**Реализация создания игровой комнаты:**

* Справочная информация по работе комнаты: <https://habr.com/ru/post/147431/>
* Создать объект RoyalRoom и передать ему список подключившихся игроков
* Удалить ссылки на игроков в очереди
* Добавить объект в Rooms

**При создании RoyalRoom должны выполняться следующие действия:**

* Создание объекта RoyalRoomLogic
* Создание RoomNetwork

**При создании RoyalRoomLogic должны выполняться следующие действия:**

* Создание контекста данных RoyalGameModel
* Создание таймера запускающегося каждые 50 милисекунд и вызывающего метод TickQuantTimer
* Cоздание класса для хранения времени прошедшего с предыдущего запуска TickQuantTimer
* Реализовать метод TickQuantTimer

**Создание контекста данных RoyalGameModel:**

* Создаем массив клеток размером lengthOfSide. В каждую клетку помещаем ссылку на CellField
* Запускаем метод CreateGameObject (необходимо реализовать по умолчанию) – создаем объекты и добавляет их в список объектов и на игровую карту в соответствии с координатами
* Создаем метод CreatePlayers() – добавляет игроков в спиcки Players, GameObjects и на игровую карту

**При создании RoomNetwork должны выполняться следующие действия**:

* Сохранение ссылки на игровую логику
* Запуск метода CreateClients – создает INetworkClient и записывает в него ссылку на игрока из RoyalRoomLogic
* Создание таймер, который срабатывает каждые 3 сек и вызывает метод HandlerTotalSinch