Лекция 6

Основы создания многопользовательских игр на Unity3D

Network Manager

- На каждом объекте должен висеть `Network Identity`, но не на `Network Manager`
- `Network Manager`очень удобный и гибкий. Все можно делать из инспектоора и не писать код
- `Spawn Management` - показывает где инстанцировать каждый `gameobject`
- `Scene Management` - показывает какая сцена грузится при онлайн или офлайн

Server, Client, Local Client

- `Авторитарность` отвечает за то, где и кто будет контролировать состояние объекта
  - Позиция и все подобное
  - если есть авторитарность, то клиент может менять своё положение у себя и на сервере
  - если нет авторитарности, то клиент, меняя своё положение, не меняет положение себя на сервере
- У каждого игрока свой мир. Состояние синхронизируется между мирами

NetworkBehaviour

- `NetworkBehaviour` - аналог `MonoBehaviour` для манипуляции с `Network`
- `SyncVar` - показывает какие поля будут синхронизоваться
  - Максимум 32 поля `SyncVar`
  - Обновление `SyncVar` идет в одну сторону `Server -> Client`
  - Структуры `Vector3` передает все переменные, даже если одна координата поменялась
  - Unity сама понимает какие поля изменились
- `SyncList` - используется для синхронизации списка переменных
  - Сериализуются только `public` поля
  - Можно серилизовать только базовые типы
  - Есть `callback` по обновлению списка

Способы общения между клиентом и сервером

1. `UnityCallbacks` - каллбеки по сетевым эвентам
2. `RPC` - позволяет вызвать клиенту какой-то метод у аналогичного объекта на сервере
   - `Command`. Например, мы хотим чтобы дверь открывалась только на сервере
3. `Messages` - сами определяем каие данные должны передаваться с сообщением
4. `onSerialise` - отправить данные по передаче в него `true`

Network Proximity Checker

- Отвечает за видимость других клиентов
- Позволяет уменьшить количество отправялемых данных
- На каждом `gameobject` должен быть коллайдер

Корутины

- Все скрипты, если мы иного не указали, грузятся в одном основном потоке. И если будет какая-то тяжелая операция при старте, то unity сначала выполнит тяжелую операцию, тем самым зависнет на время выполнения операции.
- `IEnumerator` делает функцию корутиной
- Будет сгенерирован отдельный класс, где будет храниться все локальные переменные функции - корутины
- `WaitForSecondRealtime()` - ожидание секунды по реальному времени
- `WaitForSecond()` - ожидание секунды по времени `Time.timeScale()`

Сцены

- Переключение между сценами делать в корутинах
- У AsyncOperation есть атрибут `progress`, который показывает какой прогресс выполнения операции