OXD_TCP

Описание протокола «Open XD TCP»
v. 0.0.05a

Цель протокола

Организовать двустороннюю передачу данных между приложением на базе игрового движка (ниже, условно называем как **клиент**) и приложением, которое управляет динамической платформой (далее условно называем как **сервер**).

Обмен данных осуществляется по средствам TCP протокола. Адрес и порт соединения должен быть настраиваемый. По умолчанию используется IP 127.0.0.1 и порт 11000.

Протокол состоит из набора сообщений, которые могут распознаются как команды, отчеты, поток данных.

Команды получаемые клиентом

Команда	Параметры	Описание
Управление основным треком анимации (камера или транспорт)		
START		Запуск проигрывания
STOP		Остановка с возвратом на начало проигрывания
PAUSE		Остановка
QUIT		Закрытие клиента, выход из программы
Управление уровнями		
LEVELS		Возвращает список доступных уровней в формате **JSON***, пример: **{'Super Bomb','Fly Fire','Mad Attack'}**
LOAD_LEVEL	Имя уровня	Загружает указанный уровень. Имя уровня можно получить при помощи **LEVELS_GET**, если уровень не найден,

		возвращает **LEVEL name NOT_FOUND**
Справочная информация		
NAME		Возвращает имя приложения
TITLE		Возвращает заголовок приложения
DESCRIPTION		Возвращает описание приложения
AUTHOR		Возвращает имя автора приложения
ENGINE		Возвращает имя игрового движка, на базе которого построено приложение
OXD_TCP		Возвращает версию протокола OXD_TCP
ID		Идентификационный номер или контрольная сумма
OCULOS_SDK		Возвращает номер версии SDK Oculos Rift, если не поддерживает, возвращает **NONE**
Настройки клиента		
DEBUG		Устанавливает или отменяет режим отладки TCP клиента. В этом режиме на экран выводятся строки, получаемые или отправляемые серверу. В некоторых случаях функция может быть не доступной, например если приложение на базе UE4, собрано без параметра -Debug.
CMD	Любая поддерживаемая консольная команда, например запрос для UE4: **CMD quit**, приведет к закрытию приложения.	Запуск консольных команд клиента. В некоторых случаях часть консольных команд может быть не доступна или приводить к нежелательным последствиям. Например клиент на базе UE4, собранный без команды -Debug, может зависнуть, получив команду **CMD Stat FPS**
MODE_SET	**SYNCH_FRAME** **STREAM** (возможно расширение списка режимов)	Установка режима работы клиента. В режиме **SYNCH_FRAME** клиент будет возвращать опорные сигналы, для синхронизации с уже прописанным треком движения.

		В режиме **STREAM**, клиент будет возвращать координаты перемещения, и в первом и во втором случае интервал передачи данных можно настраивать (см. ниже **TIMER_GET**, **TIMER_SET**) Ответом на изменение режима будет строка **MODE SYNCH_FRAME** или **MODE STREAM,** а если режим не поддерживается возвращает **MODE NOT_SUPPORT.** Подробнее о данных передаваемых в этих режимах см. ниже.
MODE_GET		Команда возвращает строку **MODE SYNCH_FRAME** или **MODE STREAM**, для того, чтобы проверить, какой режим активен в данный момент.
SCALE_GET		Возвращает переменную масштабирования, по умолчанию — 1.0. Пример ответа: **SCALE 1.0**. Масштабирование актуально только для **MODE STREAM**
SCALE_SET	Переменная **float**	Устанавливает размер масштаба. Параметр влияет на размер передаваемых координат.
TIMER_GET		Возвращает значение интервала отправки данных, пример ответа: **TIMER 1.0**, по умолчанию интервал равен 1.0 (секунд)
TIMER_SET	Переменная **float**	Устанавливает интервал отправки данных, по умолчанию значение = 1.0
SAVE_CONFIG		Сохраняет значения измененных параметров: **TIMER, SCALE, MODE, DEBUG, TRACKING, LOCATION, ROTATION**
RELOAD_CONFIG		Производит обновление конфигурации. Эта функция полезна, если вы вручную редактировали файл **Connector.ini****
FULLSCREEN		Переход в полноэкранный режим и обратно

DIMENSION	**[width]x[height][w|f]**	Изменяет разрешение экрана, пример: **DIMENSION 1920x1080f** – установит разрешение 1920x1080 с полноэкранным режимом, **DIMENSION 800x600w** – установит разрешение 800x600 в оконном режиме.
TRACKING_GET		Возвращает строку **TRACHING ACTOR** или **TRACHING CAMERA**
TRACHING_SET	**ACTOR** **CAMERA**	Устанавливает режимы слежения за главным объектом. **ACTOR** – в этом режиме будут передаваться данные о перемещение главного героя или транспортного средства. Этот режим больше подходит для использования приложения совместно с Oculos Rift **CAMERA** – в этом режиме будут передаваться координаты камеры, которая следит за главным героем (персонаж или транспортное средство). Этот режим больше подходит для использования приложения в экранном режиме, по аналогии с обычным райдом.
Данные в режиме **SYNCH_FRAME**		
Данные отправляются клиентом через промежуток времени, установленный при помощи **TIMER_SET.** Передача начинается после отправки команды **START** и заканчивается после передачи команд **STOP, PAUSE** **{F:[int],DT:[float],T:[float]}**		Набор величин в формате **JSON***, где: • **F** – номер кадра (frame) • **DT** – промежуток времени между пред идущим вызовом (delta time) в секундах. • **T** – общее время с момента запуска проигрывания в секундах Пример: **{F:125,DT:1.002856,T:5.678946}**
Данные в режиме **STREAM**		
Данные отправляются клиентом через промежуток времени, установленный при помощи **TIMER_SET.** Передача начинается после отправки команды **START** и		Набор величин в формате **JSON***, где по умолчанию: • **DT** – промежуток времени между пред идущим вызовом (delta time), **DT** напрямую зависит от FPS и

заканчивается после передачи команд **STOP, PAUSE** **{DT:[float],X:[float],Y:[float],Z:[float],R:[float],P:[float],W:[float]}**		установленного параметра **TIMER**. Например если **TIMER** установлен на 1.0000 сек, **DT** может отличаться в небольших пределах. Это связано с производительностью. **DT** является фактическим значением, которое высчитывается между вызовами. • **X** – изменение координаты X за промежуток **DT** • **Y** – изменение координаты Y за промежуток **DT** • **Z** – изменение координаты Z за промежуток **DT** • **R** – текущее значение Roll в градусах • **P** - текущее значение Pitch в градусах • **W** – текущее значение Yaw в градусах Пример: **{DT:1.005895,X:1.556446,Y:0.523495,Z:0.413259,R:5.123495,P:20.493178,W:1.379545}** см. ниже, как перенастроить вывод для **STREAM**
Настройка режима **STREAM**		
LOCATION_SET	**DELTA** **GLOBAL** **VECTOR**	Перенастройка вывода отправки данных о перемещении. **DELTA** – данные являются разницей между предыдущими координатами объекта с текущей позиций, за промежуток времени **DT** **GLOBAL** — глобальные координаты объекта, без какой-либо обработки **VECTOR** — координаты описаны вектором перемещения, подробное описание появится позже.
ROTATION_SET	**DELTA** **GLOBAL** **VECTOR**	Перенастройка вывода отправки данных о вращении. **DELTA** – данные являются разницей между предыдущими положением объекта с текущей позиций, за промежуток времени **DT** **GLOBAL** — текущее положение объекта, без какой-либо обработки **VECTOR** — положение объекта описанное вектором

		вращения, подробное описание появится позже.
LOCATION_GET		Возвращает одно из установленных значений (**DELTA**, **GLOBAL**, **VECTOR**)
ROTATION_GET		Возвращает одно из установленных значений (**DELTA**, **GLOBAL**, **VECTOR**)
Управление локальным многопользовательским режимом		
SPLIT_2		Создает второго пользователя и разбивает экран на 2 части по вертикали
SPLIT_4		Создает +3 пользователя и разбивает экран на 4 части
SPLIT_DEL		Удаляет всех добавленных пользователей, оставляет один общий экран
Управление эффектами		
Данные отправляются клиентом в те моменты, которые прописаны сценарием приложения**.** Передача возможна только после отправки команды **START** и заканчивается после передачи команд **STOP, PAUSE** **{FX:[string],P:[float]:,T:[float]}** **Пример.** По пути следования попадаем в водопад, тригеры расставленные в приложении срабатывают и отправляют серверу строку: **{FX:3,P:1.0:T:3.0}** Что должно интерпретироваться сервером как: «Включить эффект брызги на 3 секунды на полной мощности». Если устройство создающее эффекты, не управляет мощностью, то поступить можно несколькими способами: 1. Игнорировать мощность. 2. Использовать дискредитацию, давать серию импульсов с	Строка в формате **JSON***, где: • **FX** — имя эффекта 1. **WIND** (ветер) 2. **LIGHT** (стробоскоп) 3. **SPRAY** (брызги) 4. **SMOKE** (дым) 5. **SNAKE** (змеи, «мышиные хвостики») 6. **SOAP** (пузыри) 7. **SNOW** (снег) 8. **SHAK** (вибрация) Эффекты от 9 номера до 255 не зарезервированы и могут быть использованы как угодно. Для того, чтобы пояснить как работать с новыми эффектами, или изменить номера эффектов, используйте команду **FX_MAP**	

определенной длительностью и частотой.	• **P** — мощность (от 0 до 1) • **T** — промежуток времени (секунды) в котором работает эффект

Примечания:

JSON*

В протоколе **OXD_TCP**, JSON используется в слегка видоизмененном состоянии (облегченная версия). Отсутствую любые кавычки, лишние пробелы, знаки переносов строки.
Предполагается использование встроенного разбора строк. Наличие фигурных скобок говорит о начале или завершении сообщения JSON, запятые разделяют пары «параметр:значение». Двоеточие отделяет параметр от значения.
Упрощенный формат JSON, не содержит вложенных структур, как правило это всегда небольшие массивы состоящие из пар **ИМЯ_КЛЮЧА:ЗНАЧЕНИЕ**.

Connector.ini**

Конфигурационный файл, в котором содержатся только параметры протокола **OXD_TCP**. Его редактирование во внешнем редакторе может быть полезно, для изменения IP или адреса порта TCP соединения, так как эти два параметра удаленно не изменяются. Перенастройка адреса и порта может понадобиться, если программа управляющая динамической платформой (сервер) и приложение (клиент) установлены на разных машинах с общей локальной сетью.