Мартьянов Никита Андреевич
студент
Зиганшина Юлия Афляховна
студентка
Кузнецов Сергей Андреевич
студент
Салов Данил Дмитриевич
студент
ФГАОУ ВО «Южно-Уральский государственный университет (НИУ)»
г. Челябинск, Челябинская область

ОРГАНИЗАЦИЯ КАНАЛА ПЕРЕДАЧИ ДАННЫХ ИЗ SIMULINK В UNITY

Аннотация*: в данной статье приведен алгоритм настройки канала связи между программами Simulink и Unity для использования графической оболочки последнего при моделировании движения объектов в Simulink. В результате проделанной работы было промоделировано движение космического аппарата вблизи астероида.*

Ключевые слова*: Simulink, Unity, моделирование движения.*

Трехмерное моделирование процессов позволяет наглядно продемонстрировать движение объекта управления(ОУ) в пространстве, что позволяет наблюдать особенности поведения ОУ, которые могли быть неочевидными или неучтёнными при работе только с отображением сигналов в виде графиков. Кроме того, трехмерное моделирование является неотъемлемой частью демонстрации проделанной работы, особенно, если ОУ представляет собой некоторое средство передвижения (автомобиль, космический аппарат, водное судно, квадракоптер, самолет и т. д.).

Одним из самых популярных средств моделирования процессов является пакет Simulink математической системы MATLAB, реализующий имитационное

блочное визуально-ориентированное моделирование систем и устройств как самого общего, так и конкретного назначения [1, с. 34]. Одно из решений визуализации – использование стандартного средства Simulink 3D Animation. Однако может потребоваться более профессиональная настройка трехмерной сцены (освещение, тени, отражения и т. п.) или другие особенности, которые нельзя реализовать в Simulink 3D Animation или их реализация чрезмерно трудоемкая.

Предлагаемым решением для трехмерной визуалиции является среда разработки Unity [2, с. 20]. На сегодняшний день Unity является одним из самых популярных средств для разработки компьютерных игр и симуляций и уже зарекомендовала себя среди пользователей за удобство и простоту использования.

В данной статье будет рассказано о настройке канала связи между пакетом Simulink и среды Unity по протоколу TCP/IP и передаче данных из Similink в Unity для дальнейшей визуалиции процесса. Такими данными, например, могут являться координаты, углы поворота, состояние двигателей и т. д.

Использование TCP/IP для организации связи

Для возможности коммутации двух независимых программных продуктов можно использовать сетевые технологии, такие как UDP или TCP/IP. И Simulink, и Unity позволяют организовывать подобные соединения. Будем использовать технологии TCP/IP [3, с. 12], где Unity будет являться сервером, а Simulink – клиентом.

В Simulink основным блоком для передачи данных по сети является TCP/IP Send из библиотеки Instrument Control Toolbox. В Unity имеется поддержка языка C# и технологий Microsoft.NET, поэтому серверная часть реализована с использованием библиотек System.Net и System.Net.Sockets [4, с. 77].

При запуске Unity сервер будет ожидать подключения Simulink, после успешного подключения Unity будет принимать данные и использовать их для визуализации, пока Simulink не прекратит работу и не завершит сетевое подключение.

Так как трехмерная визуализация представляет собой процесс в реальном времени, то в Simulink существует задержка на шаг моделирования. Таким образом частота вычислений в Simulink и частота отправки данных в Unity одинакова.

Настройка Simulink

В первую очередь в настройках проекта необходимо выбрать моделирование в дискретном виде и задать шаг моделирования, например, 0,1, как это показано на рис.1.

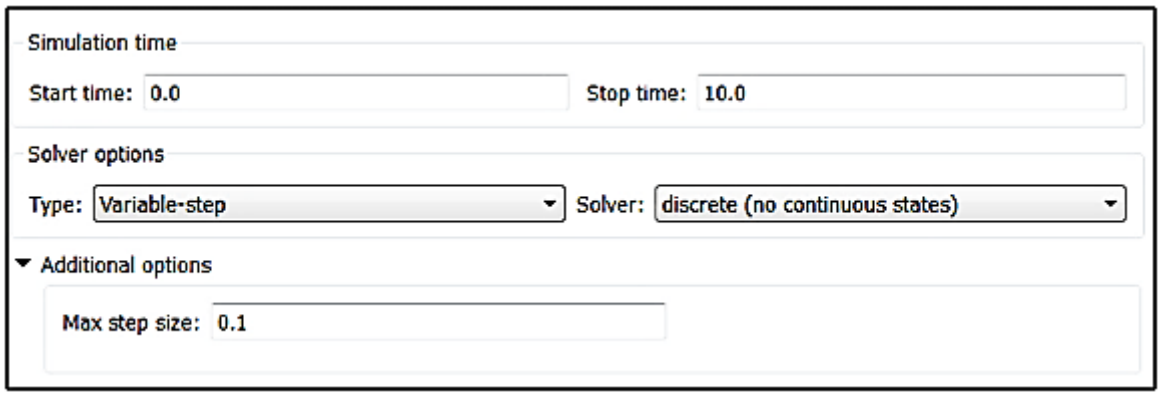

Рис. 1. Настройка проекта

Схема передачи данных представлена на рис. 2. Показан пример с передачей сигналов от источников Data1 и Data2, далее в блоке module1 они преобразуются к типу single, а также формируется служебный сигнал flag, который всегда равен единице. В общем случае количество сигналов может быть гораздо большим.

Сигнал flag необходим для надежного фиксирования того, что Simulink закончил работу, потому что после завершения симуляции клиент может не сразу завершить работу и присылать вместо истинных значений сигналов нули. Unity, в свою очередь, отслеживает состояние сигнала flag, и обрабатывает ситуацию, когда он равен нулю.

На рис. 3 показано, что сигналы источников изменяются с шагом 0,1, который указывается в поле Sample time.

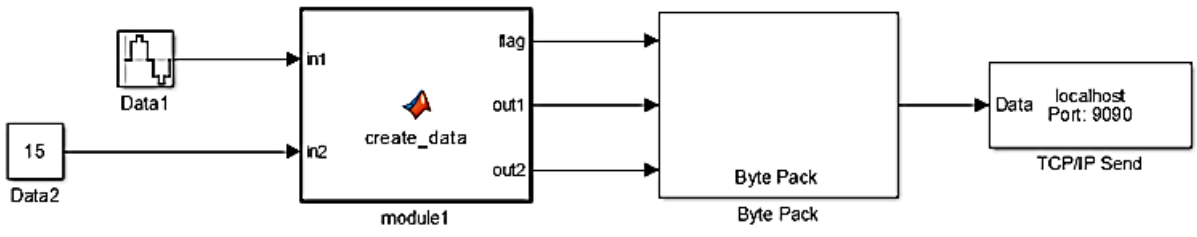

Рис. 2 Пример схемы передачи значений двух сигналов

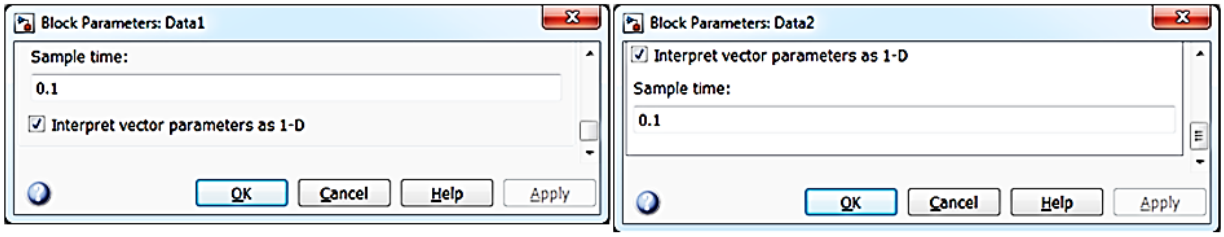

Рис. 3. Задание шага в источниках сигнала

Блок module1 – это блок MATLAB Function, где реализована задержка для обеспечения изменения сигнала симуляции в реальном времени, а также преобразование сигналов к типу single. Преобразование связано с тем, что Unity использует вещественные числа с одинарной точностью (single), а в Simulink по умолчанию используются с двойной (double). На рис. 4 представлен программный код блока module 1.

```
1   function [flag, out1, out2] = create_data(in1, in2)
2 -     pause(0.1);
3
4 -     flag = single(1);
5 -     out1 = single(in1);
6 -     out2 = single(in2);
7   end
```

Рис. 4. Блок задержки и преобразования типа сигнала

Блок Byte Pack принимает на вход сигналы и преобразует их в последовательность байтов для дальнейшей передачи этой последовательности по сети. На рисунке 5 показано, что в поле Input port data types (cell array) необходимо указать количество переменных и тип каждой из них.

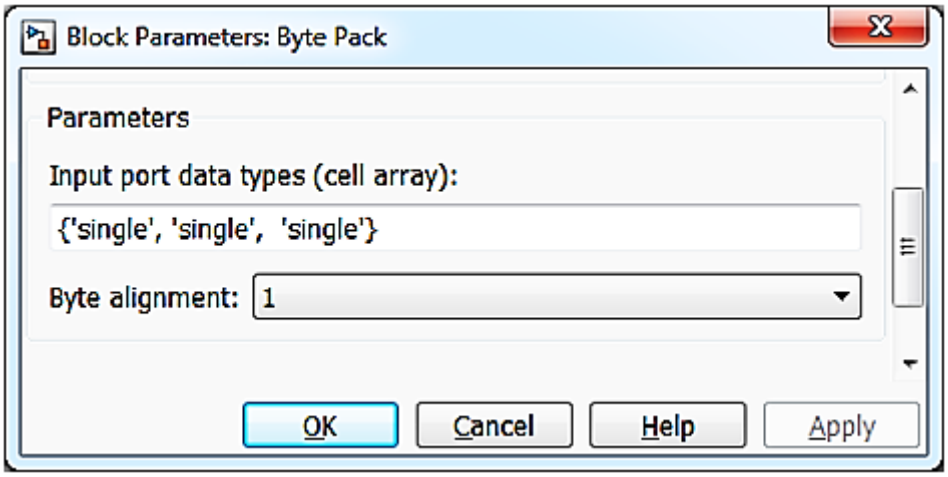

Рис. 5. Блок формирования последовательности байтов

Последовательность байтов подается на блок TCP/IP Send, где передается на сервер по указанному адресу и порту. На рисунке 6 показано, что сервер находится на той же рабочей машине, что и клиент (Значение local host в поле Remote address), порт 9090, на котором запущен сервер, задается в Unity и в общем случае является произвольным.

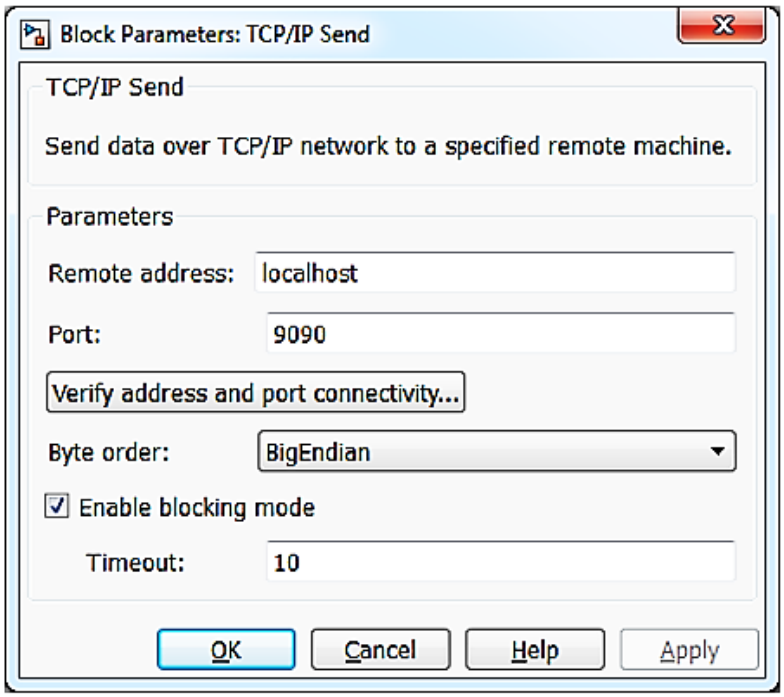

Рис. 6. Настройка блока TCP-IP Send

Настройка Unity

Создадим класс TCP, который будет запускать TCP-сервер в отдельном потоке и ожидать подключение клиента Simulink.

Создадим класс main, который будет наследовать от основного класса MonoBehaviour, который описан в библиотеке UnityEngine. Это позволит перегрузить метод Update(), который вызывается каждый раз, когда происходит перерисовка экрана. Создадим объект класса TCP, который запустит отдельный от основного поток для ожидания клиента. Как только сетевое подключение будет установлено данные будут приниматься методом Read(), где поток байтов будет обратно преобразован к переменным соответствующего типа.

В методе Update() будем получать последние данные, пришедшие от клиента с помощью метода GetData() класса TCP. Дальнейшее использование этих данных зависит от конкретной задачи.

Пример передачи сигнала

В качестве примера будем передавать сигнал синусоиды с амплитудой 1 и частотой 3,14 рад/с. Шаг моделирования примем 0,1, а длительность 1. На рисунке 7 показана схема моделирования в Simulink.

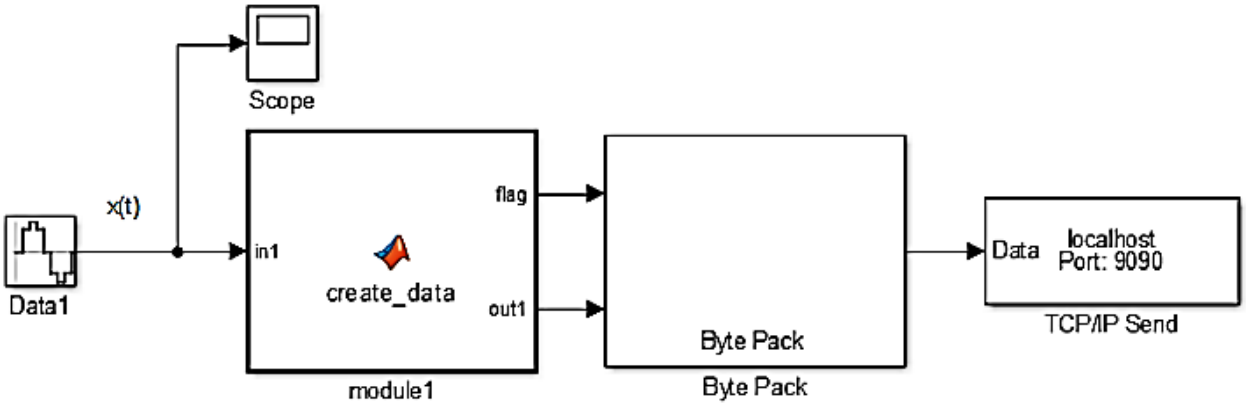

Рис. 7. Схема моделирования

На рис. 8 справа показан график изменения сигнала, а слева – значения, которые были переданы в Unity по сети. Видим, что данные передаются корректно и без потерь. В качестве примера для моделирования было рассмотрено движение космического аппарата в космосе. Была создана трехмерная сцена с 3D-объектами. На рис. 9 показан пример возможной визуализации в Unity.

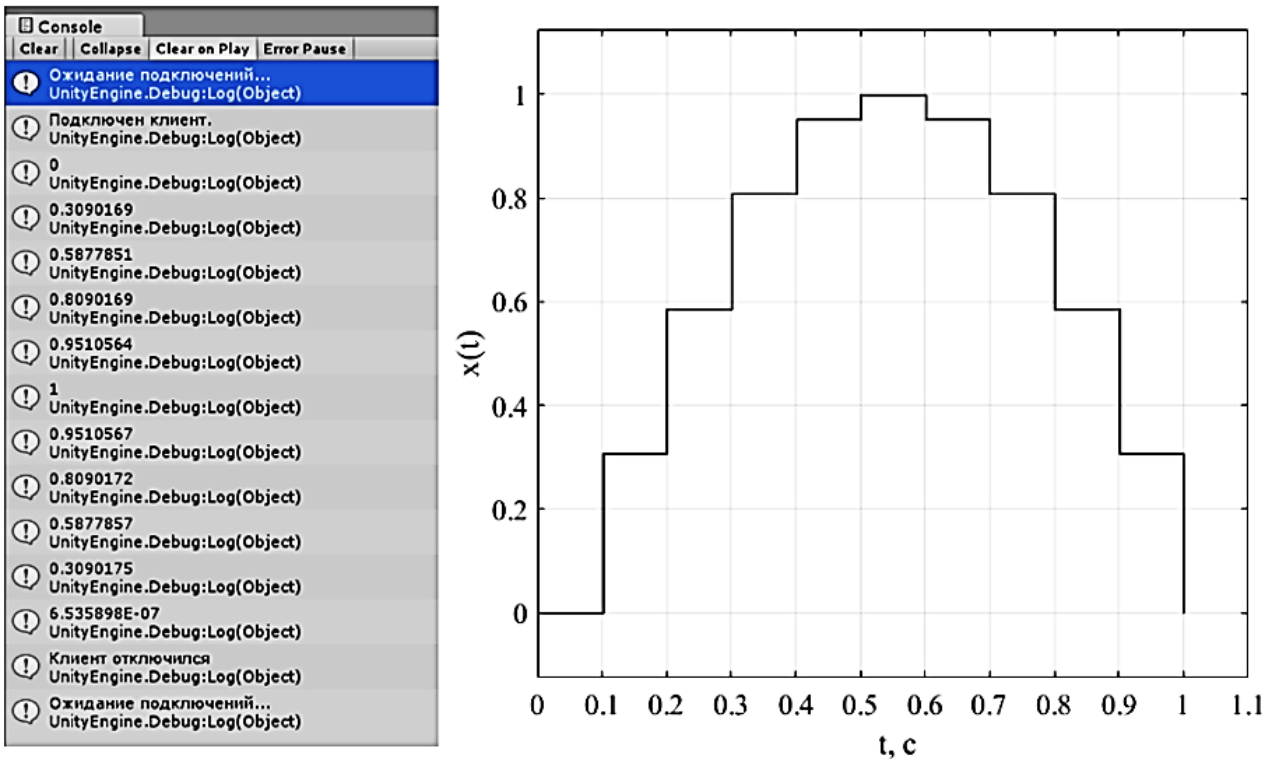

Рис. 8. Переданный и принятый сигналы

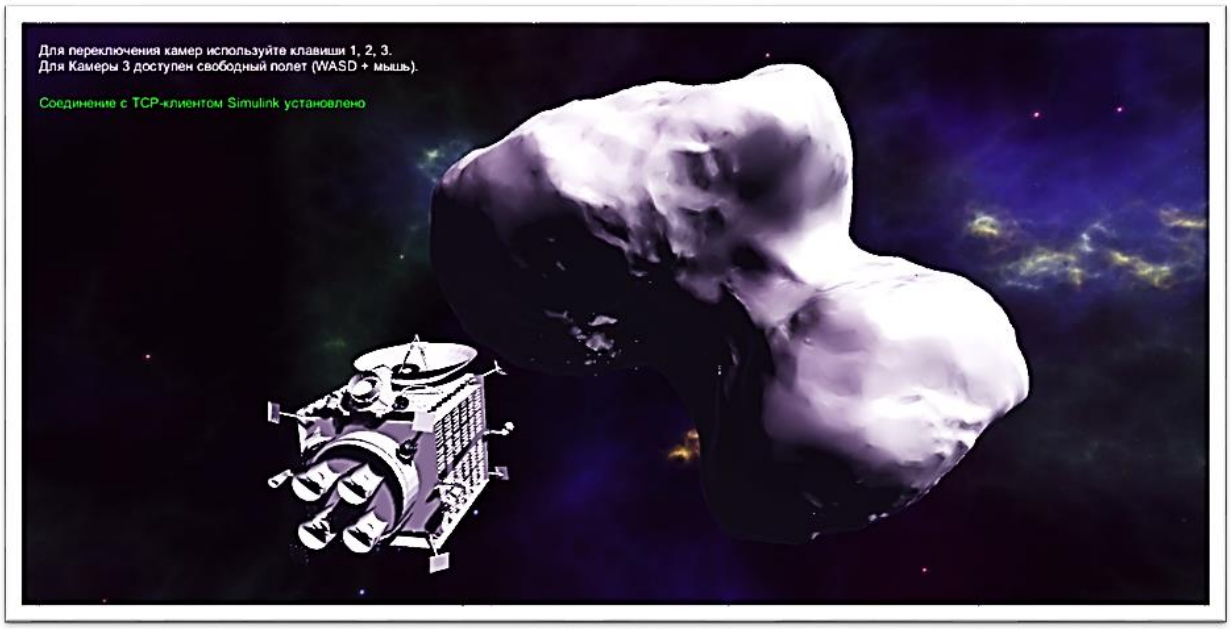

Рис. 9. Визуализация движения космического аппарата

Заключение

В результате проделанной работы, был настроен канал передачи данных между программами Simulink и Unity. Таким образом, появляется возможность при моделировании систем управления, различных механических систем использовать так же и графические возможности Unity. При небольших

изменениях программного кода появляется возможность пересылать пакеты данных обратно в Simulink, что может быть полезно, для моделирования систем, где управление реализовано в Simulink, а, например, уравнения движения или соударения симулируются в физическом движке Unity. Таким образом, реализованный метод подходит для задач управления подвижными объектами.

Список литературы

1. Дьяконов В.П. Simulink 5/6/7: Самоучитель / В.П. Дьяконов. – М.: ДМК Пресс, 2008. – 784 с.

2. Хокинг Д. Unity в действии. Мультиплатформенная разработка на C#: Монография / Пер. с англ. И. Рузмайкиной. – СПб.: Питер, 2016. – 336 с.

3. Чеппел Л. TCP/IP: Учебный курс / Л. Чеппел, Э. Типел. – СПб.: БХВ–Петербург, 2003. – 976 с.

4. Net. Сетевое программирование для профессионалов / Э. Кровчик, В. Кумар, Н. Лагари, А. Мунгале, К. Нагел, Т. Паркер, Ш. Шивакумар – М.: Изд-во Лори, 2005. – 417 с.