Санкт-Петербургский государственный университет

Математическое обеспечение и администрирование информационных систем

Небогатиков Иван Юрьевич

Поддержка программирования симулятора AirSim в среде REAL.NET

Курсовая работа

Научный руководитель:
к. т. н., доцент Литвинов Ю. В.

Санкт-Петербург
2018

Оглавление

Введение

Для решения узкоспециализированных задач во многих случаях создаются особые языки, позволяющие решить проблему, используя методы и термины, характерные для данной области. Но часто обучение людей новым средствам программирования происходит слишком долго. Для облегчения работы с программами применяется визуальное программирование [1], помогающее понять логику программы большему числу людей. Данный способ популярен в образовательной робототехнике, в частности, при программировании дронов.

На кафедре системного программирования СПбГУ разрабатывается несколько сред [2, 3, 4], позволяющих создавать программы с помощью визуальных средств. Одной из них является среда REAL.NET, позволяющая создавать предметно-ориентированные визуальные языки. На данный момент среда находится в фазе разработки, поэтому в ней не реализованы некоторые важные функции, например, пользователю удобно рисовать только небольшые программы, поскольку нельзя объеденить в подпрограмму некоторую последовательность команд.

Компания Microsoft разрабатывает симулятор дронов и автомобилей AirSim для экспериментов с компьютерным зрением и машинным обучением, но в нем не реализовано визуальное программирование. Его внедрение позволит разработчикам сконцентрироваться на вопросах обучения дронов, а не на тонкостях реализации команд квадрокоптеру. Также визуальное программирование может быть использовано в образовательной робототехнике как аналог среды TRIK Studio [4], используемой для программирования роботов, для дронов. С помощью него люди смогут научиться программировать дронов, не боясь повредить настоящий квадрокоптер.

Целью данной работы является создание языка визуального программирования в REAL.NET для AirSim. Для достижения цели были выделены следующие задачи:

- создание библиотеки на C++ для управления дроном в AirSim;

- создание .NET-обертки над библиотекой C++;
- реализация визуального языка для REAL.NET;
- реализация интерпретатора кода по визуальному языку;
- апробация на демо-программе;
- реализация возможности создавать подпрограммы в среде REAL.NET.

1. Обзор существующих решений

1.1. QReal

На кафедре системного программирования СПбГУ разрабатывается среда QReal, предназначенная для создания специализированных сред визуального программирования [2].

В состав QReal входит метаредактор, который позволяет графически задавать метамодели разрабатываемых языков, а также редактор форм, с помощью которого возможно задать форму элементов на диаграммах. Создаваемые редакторы встраиваются в QReal, наследуя всю его функциональность — визуальные отладчик и интерпретатор, версионирование моделей, разделение графической и логической моделей, распознавание жестов мышью, API генераторов кода и т.п.

REAL.NET является наследником данного инструмента и использует некоторые базовые идеи, реализованные в QReal.

1.2. TRIK Studio

На базе проекта QReal была создана среда TRIK Studio, разработанная для визуального программирования робототехнических конструкторов [5]. Среда позволяет рисовать программы и исполнять их не только на настоящих роботах, передавая команды по Bluetooth, USB или прошивая их в робота, но и на эмулируемых роботах в этой же среде.

Среда TRIK Studio позловяет компилировать код из визуальной программы. В данной работе было принято решение интерпретировать программу пошагово, поскольку она исполняется на симуляторе, который работает на этом же компьютере и ее не надо переносить на другое устройство.

2. Описание используемых инструментов и библиотек

2.1. REAL.NET

Для создания визуального предметно-ориентированного языка используется разрабатываемая на кафедре системного программирования СПбГУ среда REAL.NET. Эта платформа создана специально для экспериментов и исследований с визуальным моделированием и для использования в качестве встраиваемой .NET библиотеки.

В REAL.NET реализуется идея "глубокого метамоделирования", описанная в статье [6].

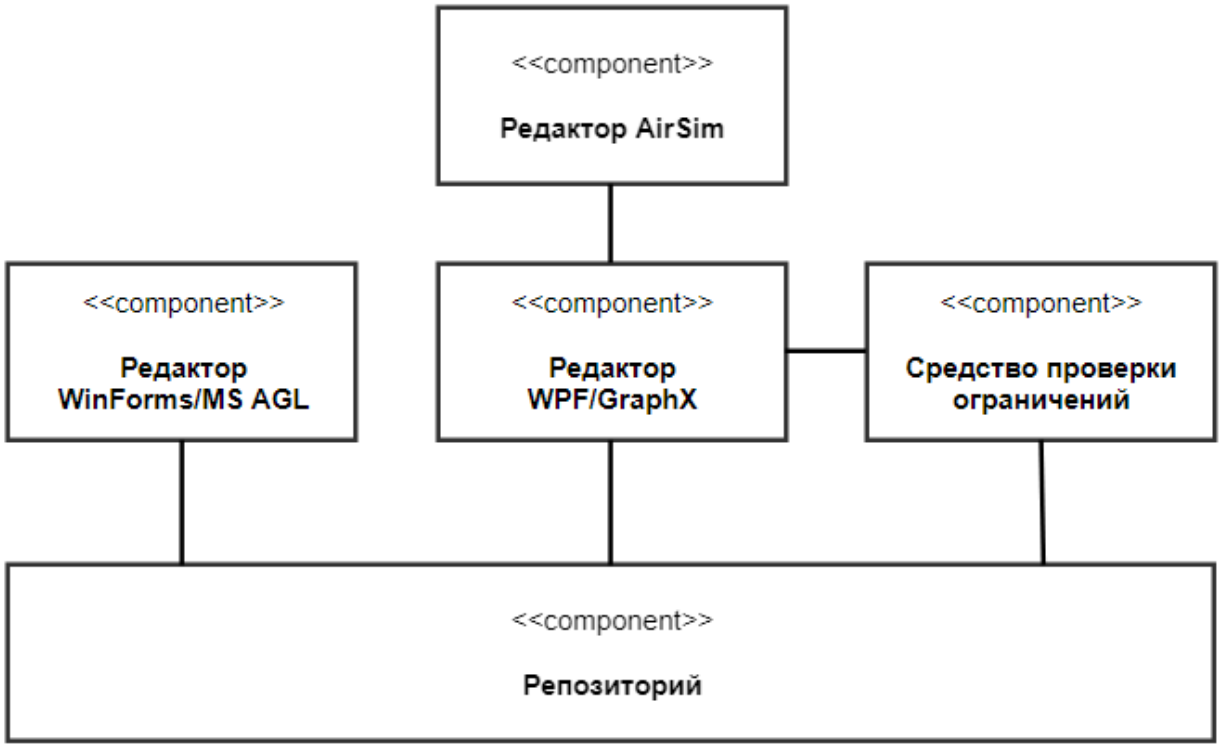

Рис. 1: Общая структура REAL.NET

Архитектура среды REAL.NET представлена на рис. 1. Метамодель визуального языка хранится в репозитории. Поскольку программы на визуальном языке представляют собой графы, то метамодель описывает все виды вершин и ребер, которые они могут содержать. К нему

подключены редакторы, визуализирующие данную модель и позволяющие пользователю вносить изменения и средство проверки ограничений, проверяющее модель и ее изменения на корректность. Редактор AirSim, в котором можно не только нарисовать программу, но и запустить ее на исполнение на дроне, подключен к WPF редактору, поскольку он является более функциональным.

2.2. AirSim

AirSim — симулятор дронов и автомобилей, разрабатываемый компанией Microsoft для экспериментов с глубоким обучением, компьютерным зрением и обучением с подкреплением [7]. Платформа поддерживает программное и аппаратное управление. Симуляция производится в игровом движке Unreal Engine. Проект разрабатывается как плагин для среды Unreal и может быть встроен в любую программу, работающую в этой среде. Исходный код проекта находится в свободном доступе.

Для взаимодействия с пользователем в приложении запускается сервер, обрабатывающий входящие сообщения. Для общения используется протокол msgpack-rpc, работающий поверх TCP/IP и позволяющий быстро упаковывать и передавать сообщения между сервером и клиентом. Данный протокол реализован на многих популярных языках программирования, что позволяет пользователям работать с дроном из разных сред разработки.

Для программного управления квадрокоптером в библиотеке AirSim находится класс MultirotorRpcLibClient, позволяющий подключиться к серверу, созданному при включении симуляции, и предоставляющий набор команд для управления квадрокоптера. Основные методы работы с клиентом описаны в списке ниже:

1. конструктор класса принимает IP-адрес и порт, на котором расположен сервер, и создает экземпляр клиента;

2. confirmConnection останавливает выполнение программы, пока соединение между клиентом и сервером не будет установлено;

3. armDisarm принимает булево значение — включить(true) или выключить(false) дрона;

4. enableApiControl включает программное управление квадрокоптера. Данный метод необходимо вызвать перед выполнением других функций управления дроном, иначе они будут игнорироваться, а в окне симуляции появится предупреждение о неправильной последовательности действий;

5. takeoff позволяет взлететь с заданным таймаутом;

6. land отдает команду дрону приземлиться;

7. hover позволяет остановить все движения и зависнуть над землей;

8. moveByVelocity — квадрокоптер двигается вдоль указанного вектора скорости в течение заданного времени;

9. moveToPosition принимает координаты точки назначения и скорость движения. Позволяет переместиться в указанную точку с указанной скоростью;

10. moveOnPath принимает список точек и скорость движения. Дрон последовательно посещает все указанные точки;

11. goHome отдает команду лететь домой, то есть к точке, которая возвращается методом getHomeGeoPoint.

Также квадрокоптер позволяет получить следующую информацию со своих датчиков:

1. getPosition — возвращает трехмерный вектор координат в пространстве;

2. getGpsLocation — возвращает gps координаты;

3. getCollisionInfo — предоставляет информацию о касаниях дроном окружающего мира;

4. getOrientation — возвращает кватернион, показывающий текущую ориентацию квадрокоптера в пространстве;
5. getVelocity — возвращает трехмерный вектор скорости;
6. getHomeGeoPoint — возвращает координаты домашней точки в пространстве;
7. simGetImages — по заданному запросу возвращает список изображений, полученных с камер квадрокоптера;
8. getDebugInfo — возвращает строку с отладочной информацией.

3. Реализация

3.1. Поток управления

Взаимодействие приложения с пользователем осуществляется по принципу, показанному на рис. 2.

Рис. 2: Взаимодействие пользователя со средой

Программа, нарисованная пользователем, интерпретируется с помощью команд для .NET-обертки библиотеки. Этот код вызывает функции клиента симулятора AirSim. Клиент отправляет полученные команды на сервер, работающий на квадрокоптере. Дрон исполняет инструкции, полученные от клиента.

В следующих разделах каждый шаг исполнения будет рассмотрен более подробно.

3.2. Создание библиотек для работы с дроном из среды .NET

Для взаимодействия между .NET приложением и частью симулятора, написанной на языке C++, была создана библиотека, реализующая функции симулятора AirSim, описанные в разделе с обзором используемых технологий. Библиотека подключается в .NET приложения и исполняет функции управления дроном, написанные разработчиками компании Microsoft на языке C++. Библиотечная функция createClient создает экземпляр C++ класса MultirotorRpcLibClient и возвращает указатель на него. Во все методы управления квадрокоптером передается данный указатель и аргументы, необходимые для вызова функции класса MultirotorRpcLibClient. Метод disposeClient вызывает деструктор созданного класса.

Во время создания и инициализации класса MultirotorClient, являющегося .NET-оберткой библиотеки, вызывается функция createClient и указатель на созданный класс сохраняется в виде приватного поля класса. Класс реализует интерфейс IDisposable и при его удалении вызывается деструктор C++ класса, завершающий работу клиента. Методы класса вызываются с помощью вызова соответствующих функций библиотеки и передачи туда созданного ранее указателя на класс.

3.3. Описание метамодели визуального языка

В репозитории среды REAL.NET была описана метамодель визуального языка управления квадрокоптером. Для каждого метода класса MultirotorClient был создан отдельный тип вершин, который будет использоваться в графе визуального языка. Каждый тип хранит в себе параметры, требуемые соответствующей функцией класса. Также были созданы специальные виды узлов initialNode, finalNode и ifNode. Вершины первого типа отвечают за создания экземпляра класса MultirotorClient, подключения к серверу и включению программного управления квадрокоптера. Вершина типа finalNode вызывает метод Dispose созданного

класса. Узлы типа ifNode имеет параметр Condition, представляющий собой булево выражение, которое будет проверяться при достижении данной вершины и дальнейший порядок действий будет зависеть от результата данного выражения.

Метамодель содержит описания двух видов дуг, соединяющих вершины. Первые отличаются от вторых наличием параметра Value, которое имеет тип bool. Ребра первого типа предназначенны для использования в качестве исходящих дуг из вершин вида ifNode, а второго типа в остальных случаях.

Ниже приведены основные ограничения на нарисованную пользователем программу:

- программа представляется в виде связного ориентированного графа;
- вершина типа initialNode должна присутствовать в единственном экземпляре;
- узлы вида initialNode не должны содержать входящих дуг, а вида finalNode — исходящих;
- из вершин всех типов, кроме ifNode, должно исходить только одно ребро, при этом оно должно быть первого типа, то есть не содержать параметра Value;
- узлы вида ifNode должны содержать две исходящих дуги второго типа, причем значение параметра Value у одной должно иметь значение true, а у другой — false.

3.4. Интерпретация графа

Для исполнения нарисованной пользователем программы в редактор AirSim были добавлены две кнопки: Execute и Stop. При нажатии кнопки Execute создается отдельный поток и начинается интерпретация графа.

Изначально токен управления передается в вершину типа initialNode, далее на каждом шаге алгоритма токен передаётся по единственной исходящей связи, либо связь выбирается исходя из семантики условного блока.

На каждом шаге проверяется корректность программы, удовлетворяет ли она ограничениям, описанным в метамодели. Если некоторые требования не выполнены, то об этом сообщается в поле вывода редактора, вызываются команды приземления квадрокоптера и отключения от сервера. Иначе, в окно сообщений записывается информация о выполненной команде и происходит переход в смежную вершину по нужной дуге. При нажатии кнопки Stop, происходит посадка дрона, завершение работы клиента и потока, интерпретирующего граф.

3.5. Поддержка подпрогамм

Для поддержки создания подпрограмм, вершинам в метамодели было добавлено поле, которое может содержать функцию или быть пустым. Пустым оно является только в тех вершинах, которые были определены как основные блоки в метамодели (например, основными блоками в языке AirSim, которые были описаны выше). Остальные вершины представляют собой функции, которые могут использовать вершины основного вида и другие функции.

Интерпретатор визуальных программ корректо работает с подпрограммами, при переходе на блок функции при исполнении рекурсивно вызывается исполнение графа подпрограммы, потом продолжается выполнение основной программы.

Реализованный принцип работы метамодели показан на рис. 3. Получается, что метамодель распадается на два "слоя": основные функции и пользовательские функции.

3.6. Тестирование

Для апробации результата были нарисованы программы, которые тестируют блоки по отдельности или взаимодействие различных бло-

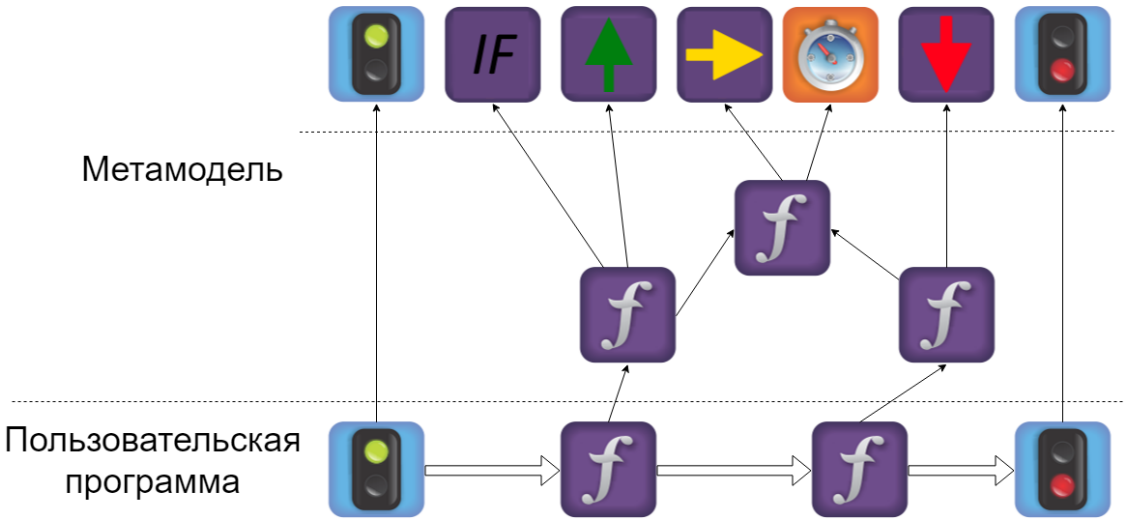

Рис. 3: Структура репозитория

ков.

Пример одной из программ изображен на на рис. 4. Приложение подключается к серверу и отправляет команды квадрокоптеру. Дрон взлетает, и в зависимости от текущего местоположения, которое проверяется в вершине ifNode, либо пролетает некоторое расстояние и садится, либо садится сразу. При завершении работы приложения происходит отключение от сервера.

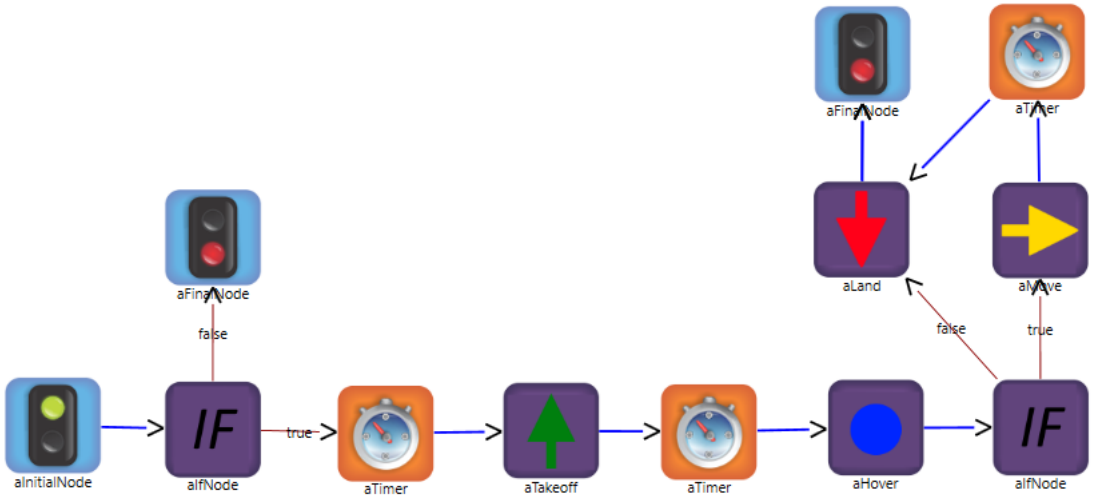

Рис. 4: Пример программы

Заключение

В рамках данной работы были решены следующие задачи:

- создана библиотеки на C++ для управления дроном в AirSim;
- создана .NET-обертка над библиотекой C++;
- реализован визуальный язык для REAL.NET;
- реализован интерпретатор кода визуального языка;
- библиотеки были апробированы на демо-программе;
- реализована возможность создания подпрограмм в среде REAL.NET.

Текущие результаты представлены в GitHub-репозитории [8].

Список литературы

[1] Kelly S., Tolvanen J.-P. Visual domain-specific modeling Benefits and experiences of using metaCASE tools. — International Workshop on Model Engineering, at ECOOP., 2000. — URL: `http://dsmforum.org/papers/Visual_domain-specific_modelling.pdf` (дата обращения: 14.05.2018).

[2] A. Kuzenkova, A. Deripaska, T. Bryksin, Y. Litvinov, V. Polyakov, QReal DSM Platform: An Environment for Creation of Specific Visual IDEs // Proceedings of 8th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2013), SCITEPRESS, 2013, pp. 251-257.

[3] Среда предметно-ориентированного визуального моделирования REAL.NET / Литвинов Ю.В., Кузьмина Е.В., Небогатиков И.Ю., Алымова Д.А. — Всероссийская научная конференция по проблемам информатики СПИСОК-2017, 2017. — URL: `https://github.com/yurii-litvinov/articles/blob/master/2017-realNet/realNet.pdf` (дата обращения: 12.05.2017).

[4] Dmitry Mordvinov, Yurii Litvinov, Timofey Bryksin. TRIK Studio: Technical Introduction // Proceedings of the FRUCT'20, 2017, ISSN 2305-7254, ISBN 978-952-68653-0-0. pp 296-308. — URL: `https://fruct.org/publications/fruct20/files/Mor.pdf` (дата обращения: 14.05.2018)

[5] Литвинов Ю.В. Реализация визуальных средств программирования роботов для изучения информатики в школах // Компьютерные инструменты в образовании, СПб., 2013, № 1, С. 36-45. — URL: `https://goo.gl/2zJdhq` (дата обращения: 14.05.2018)

[6] C. Atkinson, T. Kühne, In defence of deep modelling //Information and Software Technology. – 2015. – Т. 64. – С. 36-51.

[7] Wiki страница AirSim. — URL: https://github.com/Microsoft/AirSim/wiki (дата обращения: 14.05.2018).

[8] Репозиторий REAL.NET. — URL: https://github.com/yurii-litvinov/REAL.NET (дата обращения: 14.05.2018).