

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

по выполнению лабораторных работ
по дисциплине «Системы реального времени»
для студентов направления подготовки
09.03.01 Информатика и вычислительная техника
Направленность (профиль)

Программное обеспечение средств вычислительной техники и
автоматизированных систем

СОДЕРЖАНИЕ

ПРЕДИСЛОВИЕ.....	5
ГЛАВА 1. ОСНОВЫ МУЛЬТИПОТОЧНОГО ПРОГРАММИРОВАНИЯ.....	9
1.1. Краткие теоретические сведения.....	9
1.2. Методические рекомендации.....	19
1.3. Лабораторная работа 1. Основы программирования мультиточечных приложений в среде Visual Programming Language	36
1.4. Контрольные вопросы.....	38
ГЛАВА 2. ОСОБЕННОСТИ МОДУЛЬНО-ИЕРАРХИЧЕСКОЙ АРХИТЕКТУРЫ СЕРВИСОВ.....	39
2.1. Краткие теоретические сведения.....	40
2.2. Методические рекомендации.....	43
2.3. Лабораторная работа 2. Изучение иерархической архитектуры сервисов ..	51
2.4. Контрольные вопросы.....	55
ГЛАВА 3. ОСОБЕННОСТИ ОРГАНИЗАЦИИ МНОГОФУНКЦИОНАЛЬНЫХ ПРОГРАММНЫХ МОДУЛЕЙ	57
3.1. Краткие теоретические сведения.....	58
3.2. Методические рекомендации.....	59
3.3. Лабораторная работа 3. Изучение модульной организации программ	66
3.4. Контрольные вопросы.....	68
ГЛАВА 4. АВТОМАТИЗИРОВАННОЕ УПРАВЛЕНИЕ РОБОТОМ.....	69
4.1. Краткие теоретические сведения.....	71
4.2. Методические рекомендации.....	75
4.3. Лабораторная работа 4. Моделирование автоматизированного управления мобильным роботом	81
4.4. Контрольные вопросы.....	85
ГЛАВА 5. АВТОМАТИЧЕСКОЕ УПРАВЛЕНИЕ РОБОТОМ	86
5.1. Краткие теоретические сведения.....	86
5.2. Методические рекомендации.....	90
5.3. Лабораторная работа 5. Моделирование автоматического управления мобильным роботом.....	98
5.4. Контрольные вопросы.....	102
ГЛАВА 6. ОСНОВЫ СУПЕРВИЗОРНОГО И ИНТЕРАКТИВНОГО УПРАВЛЕНИЯ МОБИЛЬНЫМ РОБОТОМ И МАНИПУЛЯТОРОМ.....	103
6.1. Краткие теоретические сведения.....	103
6.2. Методические рекомендации.....	105
6.3. Лабораторная работа 6. Моделирование супервизорного и интерактивного управления	115
6.4. Контрольные вопросы.....	117
ГЛАВА 7. СОЗДАНИЕ МОДЕЛИ ВИРТУАЛЬНОЙ СРЕДЫ	119
7.1. Краткие теоретические сведения.....	119
7.2. Методические рекомендации.....	121

7.3. Лабораторная работа 7. Основы работы в среде симуляции Visual Simulation Environment.....	133
7.4. Контрольные вопросы.....	135
ГЛАВА 8. УПРАВЛЕНИЕ СРЕДОЙ ВИРТУАЛИЗАЦИИ.....	136
8.1. Краткие теоретические сведения.....	136
8.2. Методические рекомендации.....	147
8.3. Лабораторная работа 8. Управление средой симуляции Visual Simulation Environment	154
8.4. Контрольные вопросы.....	156
ГЛАВА 9. СОЗДАНИЕ МОДЕЛИ РОБОТА В СРЕДЕ ВИРТУАЛИЗАЦИИ.....	158
9.1. Краткие теоретические сведения.....	158
9.2. Методические рекомендации.....	162
9.3. Лабораторная работа 9. Основы разработки модели робота в среде симуляции Visual Simulation Environment	168
9.4. Контрольные вопросы.....	169
ГЛАВА 10. УПРАВЛЕНИЕ РОБОТОМ ПРИ ДВИЖЕНИИ ПО ЛИНИИ С ИСПОЛЬЗОВАНИЕМ ДАТЧИКОВ.....	171
10.1. Краткие теоретические сведения.....	171
10.2. Методические рекомендации.....	187
10.3. Лабораторная работа 10. Моделирование движения робота по линии с использованием датчиков.....	201
10.4. Контрольные вопросы	203
ГЛАВА 11. УПРАВЛЕНИЕ РОБОТОМ ПРИ ОБХОДЕ ЛАБИРИНТА	204
11.1. Краткие теоретические сведения.....	204
11.2. Методические рекомендации.....	211
11.3. Лабораторная работа 11. Моделирование движения робота при обходе лабиринта	217
11.4. Контрольные вопросы	218
ГЛАВА 12. УПРАВЛЕНИЕ ДВИЖЕНИЕМ РОБОТА С ИСПОЛЬЗОВАНИЕМ ПИД-РЕГУЛЯТОРА.....	219
12.1. Краткие теоретические сведения.....	219
12.2. Методические рекомендации.....	225
12.3. Лабораторная работа 12. Моделирование движения робота с использованием ПИД-регулятора.....	232
12.4. Контрольные вопросы	233
ГЛОССАРИЙ	234
РЕКОМЕНДУЕМАЯ ЛИТЕРАТУРА И ИСТОЧНИКИ	240
ПРИЛОЖЕНИЕ 1. СОДЕРЖАНИЕ, ФОРМА И ПРАВИЛА ОФОРМЛЕНИЯ ОТЧЕТА ПО ЛАБОРАТОРНОЙ РАБОТЕ.....	243
ПРИЛОЖЕНИЕ 2. КРАТКОЕ ОПИСАНИЕ СРЕДЫ MICROSOFT ROBOTICS DEVELOPER STUDIO.....	244

ПРЕДИСЛОВИЕ

Задачи реального времени представляют собой одну из важных областей использования вычислительной техники. Системы реального времени (СРВ) разработаны специально для управления технологическими процессами сложных производственных и информационных систем. В системах реального времени накладываются жёсткие ограничения времени выполнения задач, связанных с обработкой вводимых данных и выдачей результатов.

Применение системы реального времени всегда связано с аппаратурой, с объектом, с событиями, происходящими на объекте. Система реального времени, как аппаратно-программный комплекс, включает в себя датчики, регистрирующие события на объекте, модули ввода-вывода, преобразующие показания датчиков в цифровой вид, пригодный для обработки этих показаний на компьютере, и, наконец, компьютер с программой, реагирующей на события, происходящие на объекте.

Классическим примером задачи, где требуется СРВ, является управление роботом-манипулятором, берущим деталь с ленты конвейера. Деталь движется и робот имеет лишь маленький промежуток времени, когда он может её взять. Если он опоздает, то деталь уже не будет на нужном участке конвейера, и следовательно, работа не будет сделана, несмотря на то, что робот находится в правильном месте. Если он завершит позиционирование раньше, то деталь ещё не успеет подъехать, и он заблокирует ей путь.

Другим примером СРВ может являться мобильный робот, оснащенный датчиками, вычислительным устройством с организацией параллельной обработки потоков, а также исполнительными механизмами. Осуществляя движение в среде, робот должен своевременно реагировать на внешние воздействия.

Робототехнические комплексы популярны в сфере промышленности и в области образования как современные высокотехнологичные исследовательские инструменты. Их использование в различных учебных

заведениях среднего и высшего профессионального образования позволяет реализовывать концепцию «обучение на проектах».

Применение возможностей робототехнических комплексов в инженерном образовании дает возможность одновременной отработки профессиональных навыков сразу по нескольким смежным дисциплинам: системы реального времени, теория управления, схмотехника, программирование, теория информации, механика. Востребованность комплексных знаний способствует развитию связей между исследовательскими коллективами. Кроме того студенты уже в процессе профильной подготовки сталкиваются с необходимостью решать реальные практические задачи.

В ходе подготовки квалифицированных ИТ-специалистов возникает проблема выбора робототехнических комплексов для учебных лабораторий и базовых инструментальных сред программирования.

Компания Microsoft предлагает многофункциональную среду – Robotics Developer Studio (RDS) , содержащую возможности графического и текстового программирования робототехнических платформ и среду симуляции (моделирование поведения робота в виртуальной среде). Начиная с релиза R3, пакет доступен для бесплатного использования. Это позволяет говорить о перспективах использования RDS для обучения студентов и школьников, изучающих основы робототехники. В настоящее время выпущен четвертый релиз среды разработки – RDS 4. Важным элементом новой системы стала поддержка контроллера Microsoft Kinect. Использование этого сенсора позволит роботу не только видеть окружающий мир, но и распознавать объекты вокруг себя.

Учебное пособие состоит из двенадцати глав, содержание которых позволяет достаточно подробно рассмотреть основополагающие принципы организации систем реального времени.

Глава 1 посвящена изучению основ мультипоточного программирования. В главе 2 изложены особенности модульно-иерархической архитектуры сервисов, практикум посвящен принципам разделения программного кода,

особенностями работы с нотификациями и модулями. Глава 3 содержит сведения об особенностях организации многофункциональных программных модулей.

В главе 4 рассмотрены основные принципы автоматизированного управления, для выполнения практических заданий используется задающий манипулятор и фиксированный набор команд управления автономным подвижным объектом. Глава 5 содержит сведения о принципах организации автоматического управления, для выполнения практических заданий используется автономный подвижный объект с различными траекториями движения. В главе 6 рассмотрены основы супервизорного и интерактивного управления мобильным роботом и манипулятором.

Глава 7 посвящена изучению принципов создания модели виртуальной среды, практические вопросы отрабатываются в среде симуляции VSE Microsoft RDS с использованием Microsoft Visual Studio. В главе 8 изложены принципы управления средой симуляции, практикум выполняется с использованием среды симуляции VSE и Microsoft Visual Studio для различных робототехнических систем. Глава 9 знакомит читателя с особенностями создания модели робота в среде виртуализации, при этом используется среда симуляции VSE и Microsoft Visual Studio.

В главе 10 рассмотрены основные принципы управления роботом с использованием сенсоров, практикум выполняется с использованием разного количества датчиков освещенности при движении робота по линии. Материал 11 главы содержит сведения о принципах выполнения задачи обхода односвязного и многосвязного лабиринтов, для выполнения практического задания используется правило «одной руки» и модель роботизированной платформы с лазерным дальномером. В заключительной 12 главе рассмотрены основы управления движением робота с использованием пропорционально-интегрально-дифференциального регулятора, практикум выполняется на модели роботизированной платформы Eddie Robot Platform оборудованной контроллером Kinect.

Учебное издание включает теоретический материал достаточный для выполнения лабораторного практикума. Каждая из глав завершается лабораторной работой, выполнение которой позволяет углубить понимание процессов, происходящих в системах реального времени.

Материалы практикума дополняют теоретические сведения по дисциплине «Системы реального времени» и могут быть использованы при проведении лекций.

Глоссарий содержит перечень ключевых узкоспециализированных терминов применяемых в данном учебном издании, которые сопровождаются их толкованием.

Библиографический список содержит перечень основной литературы, включающий источники, существенно расширяющие объем учебного материала, рекомендуемого авторами к ознакомлению в процессе изучения дисциплины «Системы реального времени».

В разделе дополнительной литературы имеется список использованных литературных источников и интернет-ресурсов. В тексте пособия приведены соответствующие ссылки, позволяющие получить дополнительную информацию, а также программное обеспечение необходимое для выполнения заданий лабораторного практикума.

В приложениях приведены минимальные требования к содержанию, форме отчета по лабораторной работе и правила его оформления, а также представлено краткое описание среды Microsoft Robotics Developer Studio 4.

Авторы выражают благодарность студентам специальности «Автоматизированных систем обработки информации и управления», оказавшим помощь в создании лабораторного практикума, принимавшим в разные годы активное участие в разработке методик проведения лабораторных работ, их постоянном обновлении и совершенствовании.

ГЛАВА 1. ОСНОВЫ МУЛЬТИПОТОЧНОГО ПРОГРАММИРОВАНИЯ

Повышенная технологическая сложность современного производства, сбора и анализа данных требует жесткого соблюдения временных рамок. Системы реального времени разработаны специально для управления технологическими процессами сложных производственных и информационных систем. В системах реального времени накладываются жёсткие ограничения времени выполнения задач, связанных с обработкой вводимых данных и выдачей результатов. Данные ограничения диктуются темпом развития процессов в контролируемом объекте [4].

Реальное время (real time) – время, в течение которого протекает процесс. Реальным временем (РВ) характеризуют такую реакцию объекта на входные сигналы, при которой он успевает достаточно быстро выработать выходные сигналы. Система работает в реальном времени, если ее быстродействие адекватно скорости протекания физических процессов на объектах контроля и управления. Здесь физические процессы – это процессы, непосредственно связанные с функциями, выполняемыми данной системой реального времени.

1.1. Краткие теоретические сведения

Главной особенностью СРВ является способность принять данные, как правило, с большого числа датчиков, преобразовать их в цифровую форму, обработать, выдать управляющие воздействия. В некоторых случаях необходимо архивировать данные и постоянно отображать состояние объекта за фиксированный интервал времени на устройстве вывода информации. Выполнение данных задач требует организации многопоточной организации процесса обработки информации.

Процесс – это динамическая сущность программы, ее код в процессе своего выполнения. В данном случае под программой понимается описание на некотором формализованном языке алгоритма, решающего поставленную

задачу. Программа является статической единицей, то есть неизменяемой с точки зрения операционной системы, ее выполняющей.

Процесс имеет:

- собственные области памяти, для хранения кода и данных;
- собственный стек;
- собственное отображение виртуальной памяти на физическую;
- собственное состояние.

Поток (нить или задача) – это как бы одна из ветвей исполнения процесса, которая характеризуется тем, что:

- разделяет с процессом область памяти под код и данные;
- имеет собственный стек;
- разделяет с процессом отображение виртуальной памяти на физическую (в системах с виртуальной памятью);
- имеет собственное состояние.

Таким образом, у двух потоков в одном процессе вся память является разделяемой и дополнительные расходы, связанные с разным отображением виртуальной памяти на физическую, сведены к нулю. Для потоков так же, как для процессов, определяются понятия **состояния потока** и **межпоточное взаимодействие**.

Отметим, что для двух процессов обычно требуется организовать что-то общее (память, канал и т. д.) для их взаимодействия, в то время как для двух потоков часто требуется организовать что-то (например, область памяти), имеющее свое значение в каждом из них.

Всякий процесс содержит один или несколько потоков. Операционная система позволяет потоку порождать новые потоки.

Потоки, по их особенностям, можно разделить на 3 категории:

1. Циклические потоки. Характерны для процессов управления и интерактивных процессов.

2. Периодические потоки. Характерны для многих технологических процессов и задач синхронизации.

3. Импульсные потоки. Характерны для задач сигнализации и асинхронных технологических процессов.

Используемый в данном практикуме программный продукт Microsoft Robotics Developer Studio (RDS) – Windows-ориентированная среда для управления роботами и их симуляции [7]. Данное программное обеспечение предназначено для разработки и поддерживает большое количество разнообразного аппаратного обеспечения роботов.

Microsoft RDS основана на библиотеке CCR (Concurrency and Coordination Runtime), .NET-реализации библиотеки для работы с параллельными и асинхронными потоками данных, используя обмен сообщениями, и DSS (Decentralized Software Services) – облегченное средство создания распределенных приложений на основе сервисов, которое предусматривает управление множеством сервисов для корректировки поведения в целом.

Среди особенностей RDS можно выделить:

- язык визуального программирования Microsoft Visual Programming Language для создания и отладки программных приложений для роботов;
- веб-ориентированные и Windows-ориентированные интерфейсы;
- симуляция 3D (включая аппаратное ускорение);
- упрощенный доступ к датчикам и исполнительным механизмам робота;
- поддержку нескольких языков, включая C#, Visual Basic .NET, JScript и IronPython.

Microsoft RDS поддерживает модульность для добавления новых сервисов в комплект и состоит из 4-х основных компонентов:

- CCR (Concurrency and Coordination Runtime);
- DSS (Decentralized Software Services);
- VPL (Visual Programming Language);
- VSE (Visual Simulation Environment).

Компоненты CCR и DSS также доступны отдельно для применения в коммерческих приложениях, требующих высокого уровня параллелизма и/или возможности взаимодействия между множеством узлов в сети.

Инструменты, позволяющие разрабатывать RDS-приложения, включают в себя графическую среду VPL; утилиты командной строки, позволяющие взаимодействовать с проектами Visual Studio (от версии VS Express) на языках C#, C++ или VB.NET; поддержку скриптовых языков, как например, IronPython; и инструменты симуляции в режиме 3D.

Visual Simulation Environment – среда 3D-симуляции поведения роботов в виртуальном мире на основе технологии NVIDIA PhysX, что позволяет использовать продвинутую физическую модель.

Visual Programming Language – графическая среда разработки, использующая каталог сервисов и действий. Они могут быть связаны графическим способом или при помощи мыши. VPL также позволяет вам генерировать код новых макро-сервисов на основе диаграмм, созданных пользователями. В VPL возможна простая настройка сервисов для различных аппаратных элементов. Рассмотрим элементы, использующиеся в среде визуального программирования VPL.

Блок данных (Data) расположен в окне базовых блоков – Basic Activities (рис. 1.1) и служит для **хранения постоянного значения (константы)**.

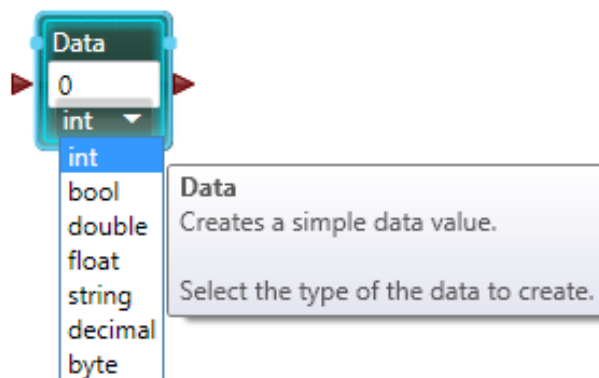


Рисунок 1.1– Блок данных

Для определения типа, хранимого значения, необходимо вызвать (нажать на стрелку в нижней части блока) список типов данных (рис. 1.1).

При использовании типа данных String, вводимое значение в кавычки не берется.

Блок переменной (Variable) служит для определения переменных (рис. 1.2).



Рисунок 1.2 – Блок переменной

Для определения переменной необходимо:

1. Переместить блок переменной в рабочее окно диаграммы, выделить его.
2. Через меню Edit, либо через окно свойств (Properties), либо нажав на символ многоточия в нижней части блока вызвать диалоговое окно добавления переменной (рис. 1.3).

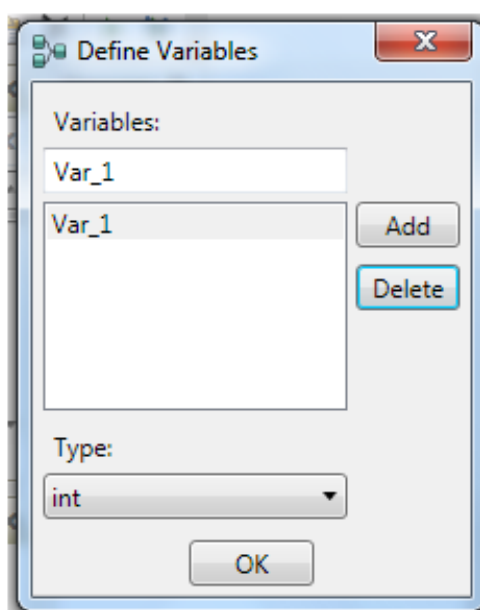


Рисунок 1.3 – Диалоговое окно определения переменной

3. Нажать на кнопку Add и ввести имя переменной (в имени должны присутствовать только латинские буквы, цифры и знак «_», при этом регистр учитывается).

4. Затем выбрать в списке тип данных и нажать кнопку ОК, либо продолжить добавлять переменные.

Редактирование переменных осуществляется с помощью тех же окон. Что бы отредактировать необходимую переменную, надо вызвать диалоговое окно добавления переменной, выбрать из списка переменную и нажать ОК. В переменную можно как **записать значение**, так и **считать значение** из нее. Осуществляется это следующим образом.

При соединении **выхода** какого-либо блока с **входом** блока переменных появляется окно связей (Connections) (рис. 1.4), в левой части которого находится окно входящих сообщений, а в правой части окно действий, реагирующих на данное сообщение. В данном случае в правой части находится два параметра: Get Value – считывание из переменной и Set Value – запись в переменную.

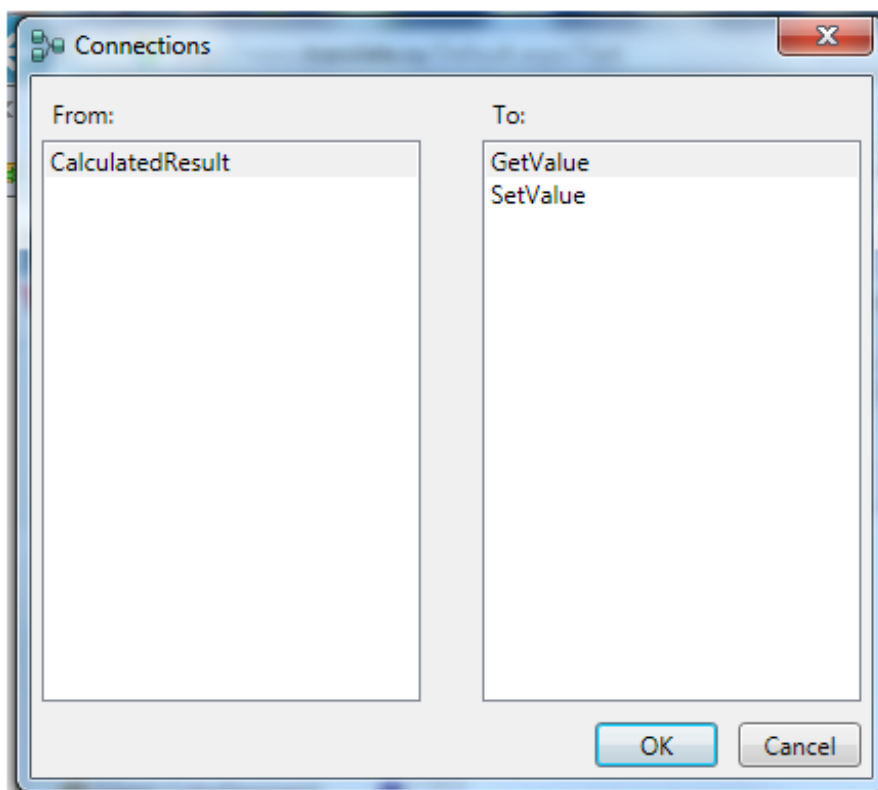


Рисунок 1.4 – Окно связей

Блок вычисления (Calculate) предназначен для выполнения простых арифметических или логических операций над числовыми или логическими значениями соответственно. В данном блоке в роли аргументов выражений

можно **использовать переменные** (рис. 1.5, а). Для этого необходимо ввести имя переменной, или ввести следующую команду «state.имя переменной».



Рисунок 1.5 – Блок вычисления с аргументами в виде: а) переменной; б) комбинации символьные строки с числовым значением

Так же в блоке вычисления можно **скомбинировать символьные строки с числовыми значениями**, например для вывода на экран, в данном случае символьная строка заключается в кавычки, далее ставится знак «+» и числовое значение, выражение или переменная (рис. 1.5, б). При использовании десятичных чисел, дробная часть отделяется точкой. В блоке вычисления реализуются следующие арифметические и логические операции:

+	– сложение;	/	– деление;	-	– вычитание;
%	– остаток от деления	*	– умножение;	&&	– И;
	– ИЛИ;	!	– логическое отрицание;		

Сервис простого диалога (Simple Dialog) (рис. 1.6) выводит на экран форму с текстом и кнопкой подтверждения (рис. 1.7). Данный сервис имеет и другие опции вывода различных форм, о которых будет рассказано позже.



Рисунок 1.6 – Сервис простого диалога

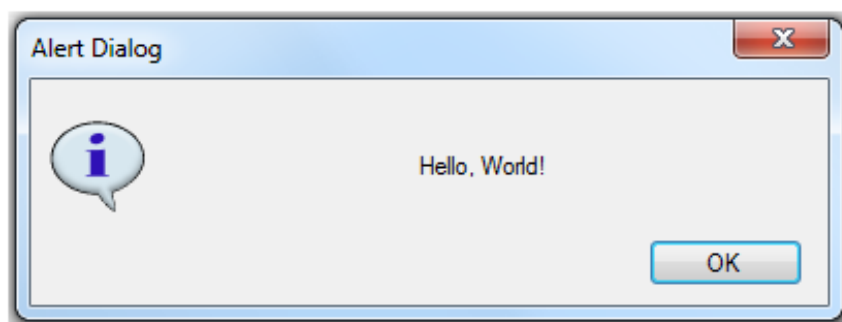


Рисунок 1.7 – Результат работы сервиса простого диалога

Сервис симуляции консольного интерфейса (рис. 1.8) выводит на экран окно консоли (рис. 1.9), с помощью которой можно осуществлять ввод и вывод информации. Консоль имеет и ряд опций работы (рис. 1.10).



Рисунок 1.8 – Сервис симуляции консольного интерфейса

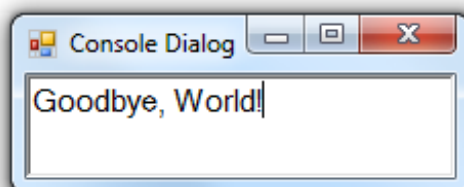


Рисунок 1.9 – Результат работы сервиса симуляции консольного интерфейса

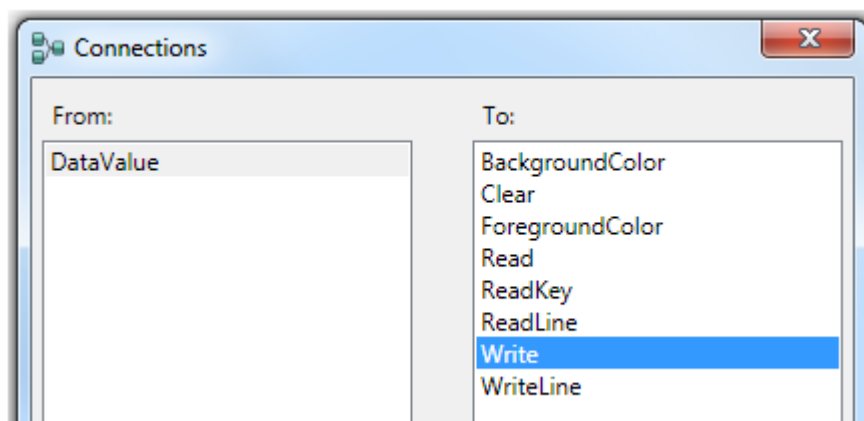


Рисунок 1.10 – Окно диалога связей сервиса симуляции консольного интерфейса

Блок слияния (Join) (рис. 1.11) предназначен для **комбинирования** различных переменных, типов данных (потоков данных) в один поток. В данном блоке каждому потоку задается ячейка с именем. Для создания ячейки потока, необходимо либо, вызвать контекстное меню блока и нажать на пункт Add Element (рис. 1.12, слева), либо отредактировать данный блок через окно свойств (рис.1.12, справа).

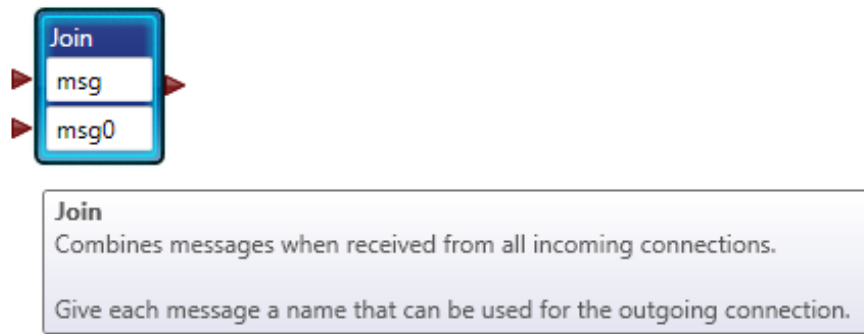


Рисунок 1.11 – Блок слияния

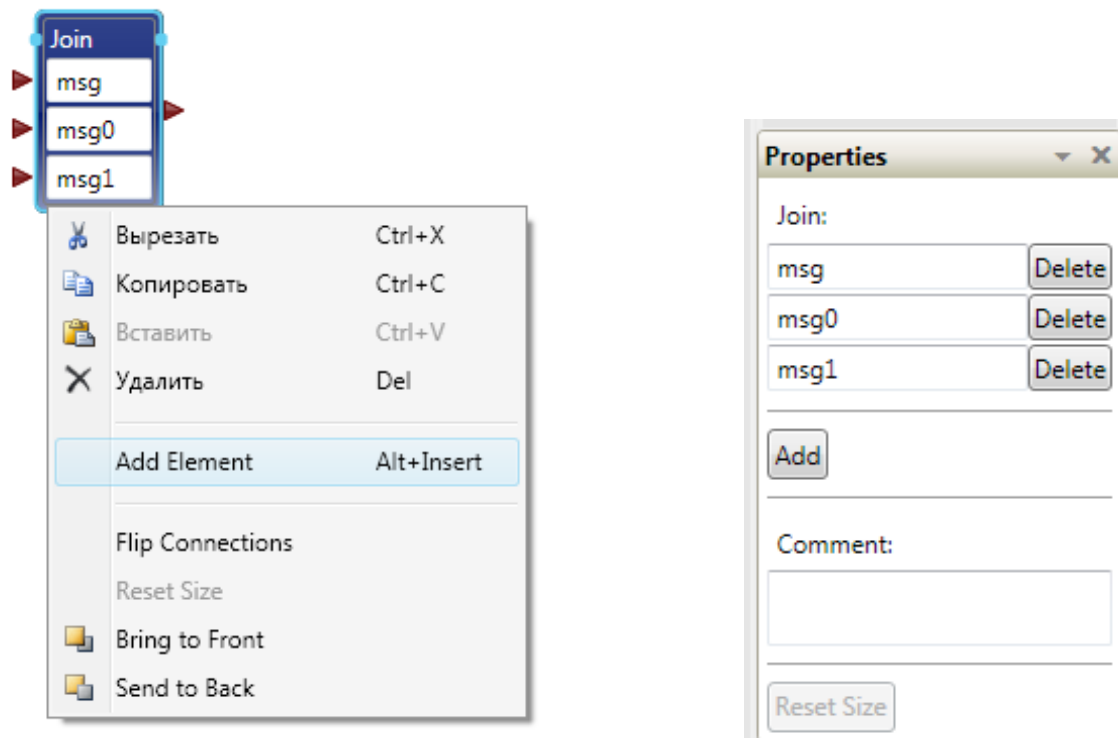


Рисунок 1.12 – Добавление новой ячейки потока

Блок условия (If) (рис. 1.13) реализует выполнение определённых команд при условии, что некоторое логическое выражение (условие) принимает значение «истина» (true) или «ложь» (false).

В блоке условия реализуются следующие логические операции:

= **или** == – равно;
 != **или** < > – не равно;
 < – меньше чем;
 > – больше чем;
 <= – меньше или равно чем;
 >= – больше или равно чем.

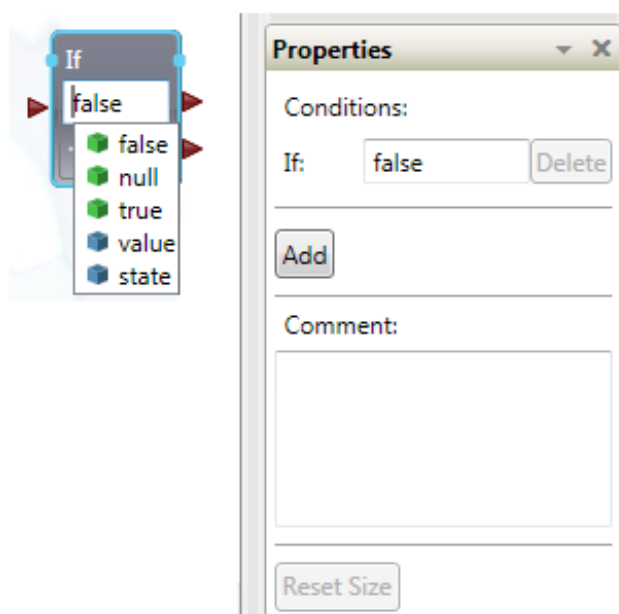


Рисунок 1.13 – Блок условия

Блок простого слияния (Merge) (рис. 1.14) служит для использования сервиса несколькими потоками данных, данный блок просто образует связку потоков, которые используют один и тот же сервис.

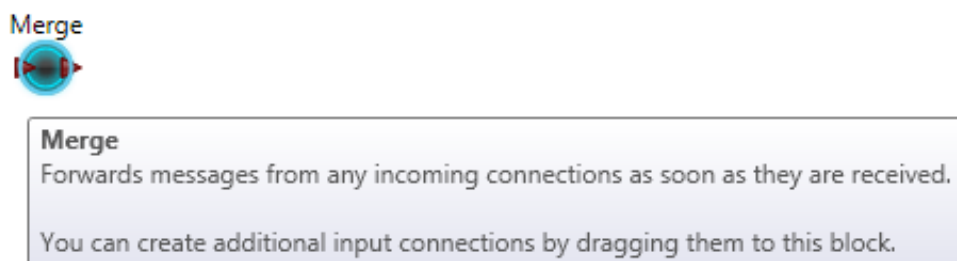


Рисунок 1.14 – Блок простого слияния

1.2. Методические рекомендации

1.2.1. Методика организации вывода текста на экран

1. Для организации вывода текста на экран необходимо добавить блок данных, выбрать тип данных – string и ввести необходимый текст в соответствующее поле. Далее нужно добавить сервис простого диалога и соединить блок данных с сервисом (рис. 1.15).

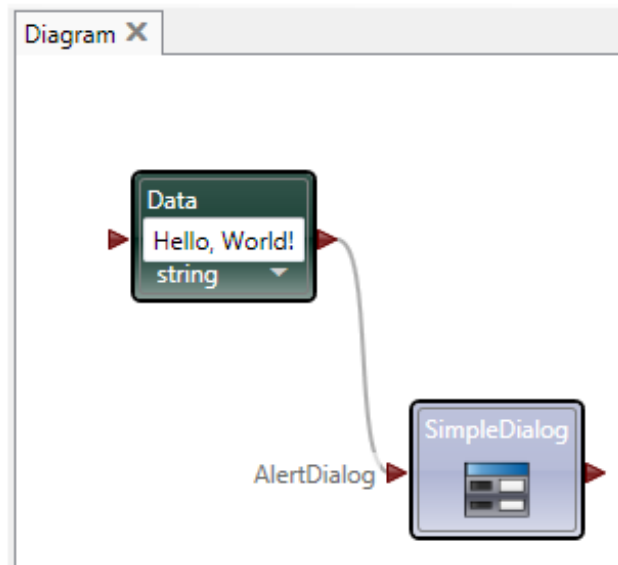


Рисунок 1.15 – Диаграмма вывода текста на экран

2. При организации соединения элементов диаграммы запустится окно диалога связей, где мы выбираем AlertDialog (рис. 1.16).

3. Далее появляется окно данных (рис. 1.17), в нем редактируются данные пересылаемые между блоками и сервисами. Выбираем из списка пункт value, это означает, что в сервис будет пересылаться какая-либо величина (в нашем случае эта величина определена блоком данных).

4. Запустим на выполнение написанную программу, нажав кнопку Start (Запустить) на кнопочной панели. При запуске программы появляется окно выполнения (рис. 1.18) со ссылкой на web-страницу отладки. В данном окне отображается ход выполнения программы, а так же ошибки, возникающие при выполнении программы. Что бы остановить выполнение программы нажмите кнопку Stop.

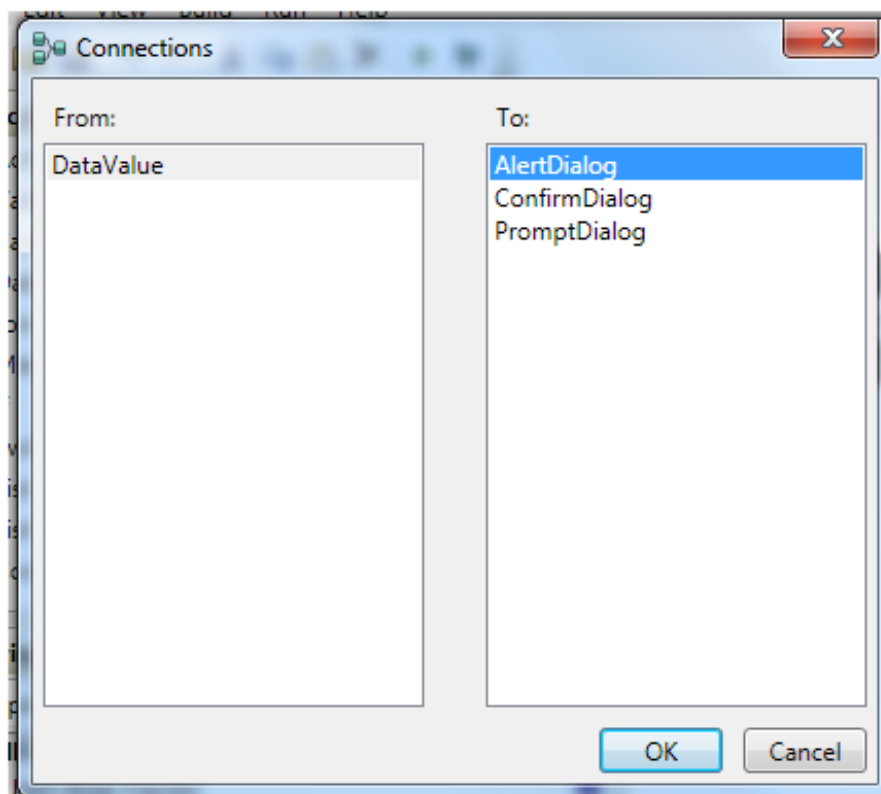


Рисунок 1.16 – Окно диалога связей

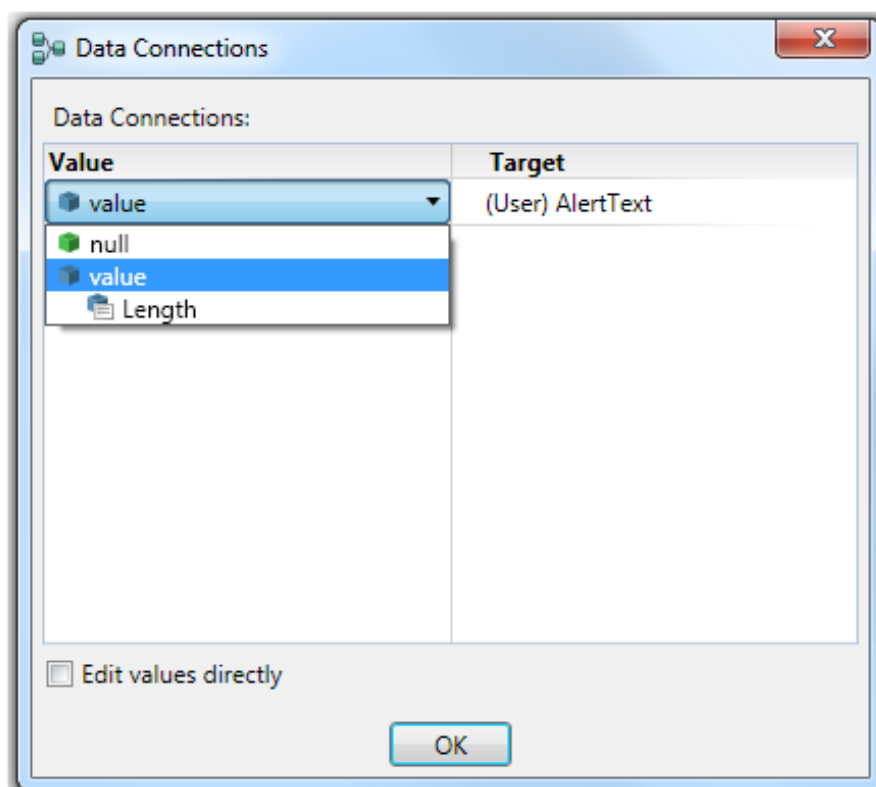


Рисунок 1.17 – Окно данных

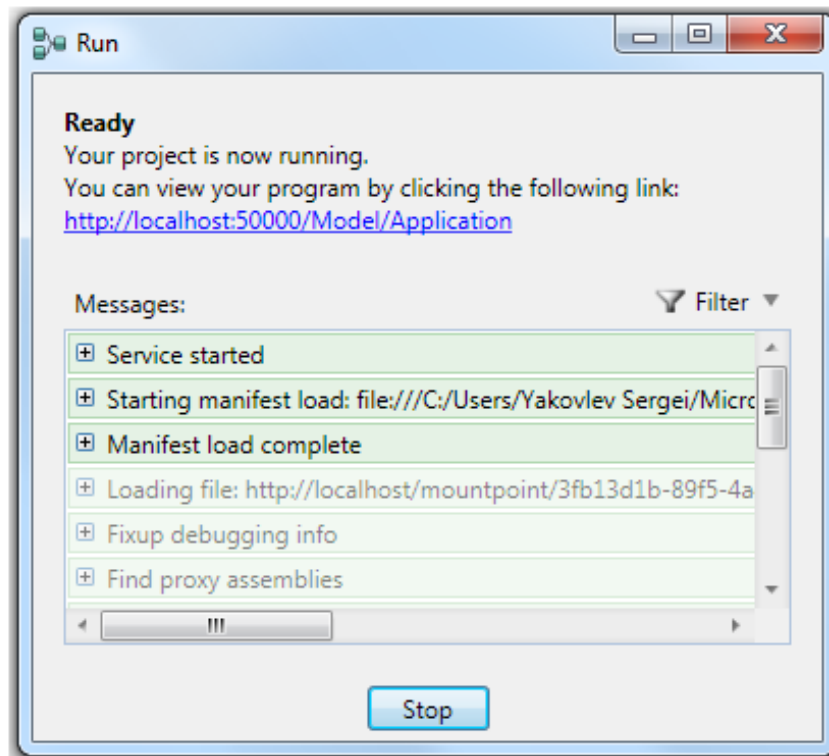


Рисунок 1.18 – Окно выполнения

1.2.2. Методика установления значения переменной

1. Для установления значения переменной необходимо добавить блок данных, выбрать тип данных – int и ввести необходимые данные в соответствующее поле. Далее нужно добавить блок переменной и вызвав диалоговое окно определения переменной, определить имя и тип переменной (рис. 1.19).

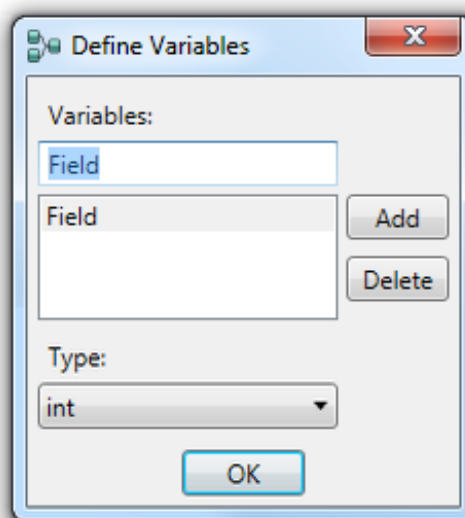


Рисунок 1.19 – Диалоговое окно определения переменной

2. Установить значение переменной «а», соединив блоки. В появившемся окне связей определить действие SetValue для входящего сообщения (рис. 1.20).

3. Конечный вид диаграммы устанавливающей значение переменной а равное 100 представлена на рис. 1.21.

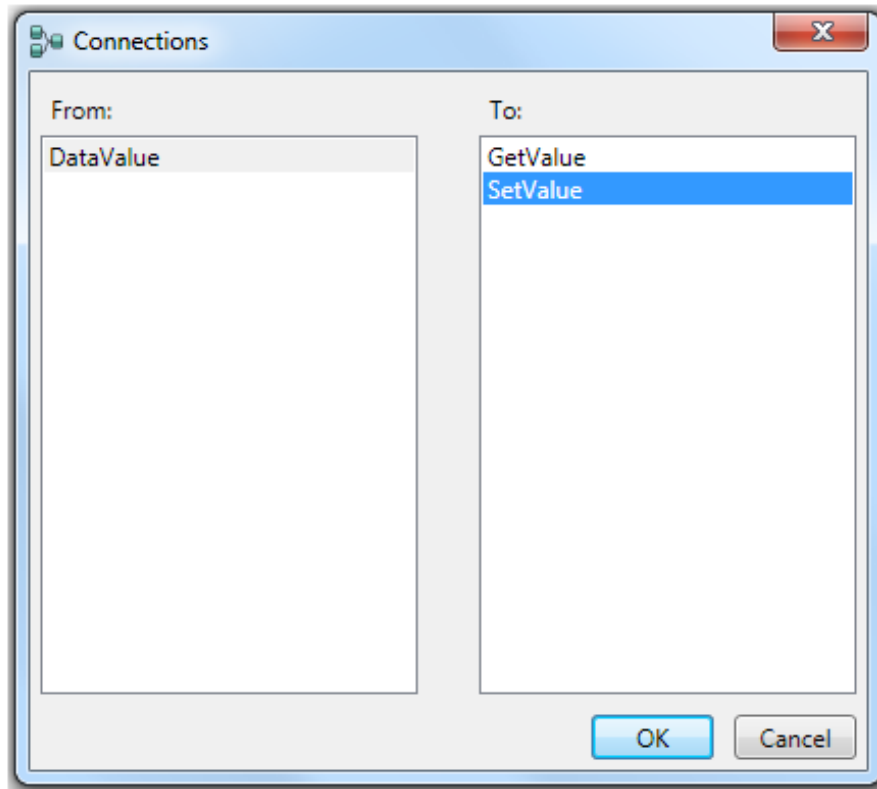


Рисунок 1.20 – Окно связей

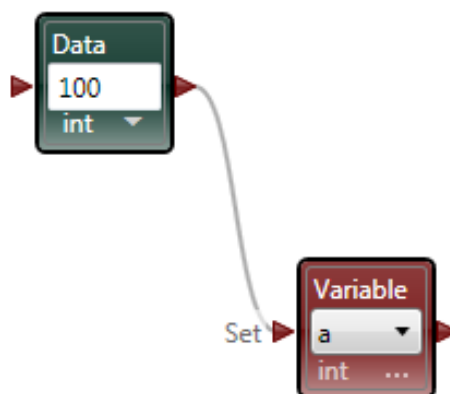


Рисунок 1.21 – Окно готовой диаграммы

1.2.3. Методика организации считывания значения переменной

1. Для организации считывания значения переменной необходимо в предыдущей диаграмме добавить блок вычисления и соединить его с блоком переменной.

2. В блоке вычисления в списке выбрать пункт state (рис. 1.22), что означает работу с состоянием переменной.

3. В блоке вычисления после state добавьте «.» и в появившемся списке выберите переменную a (рис. 1.23).

4. Добавим сервис простого диалога и соединим блок вычисления с сервисом. Выберем в окне связей опцию AlertDialog. В окне данных выберем из списка пункт value. в итоге получим диаграмму представленную на рис. 1.24.

5. Запустим на выполнение полученную программу. Результат её выполнения представлен на рис. 1.25.

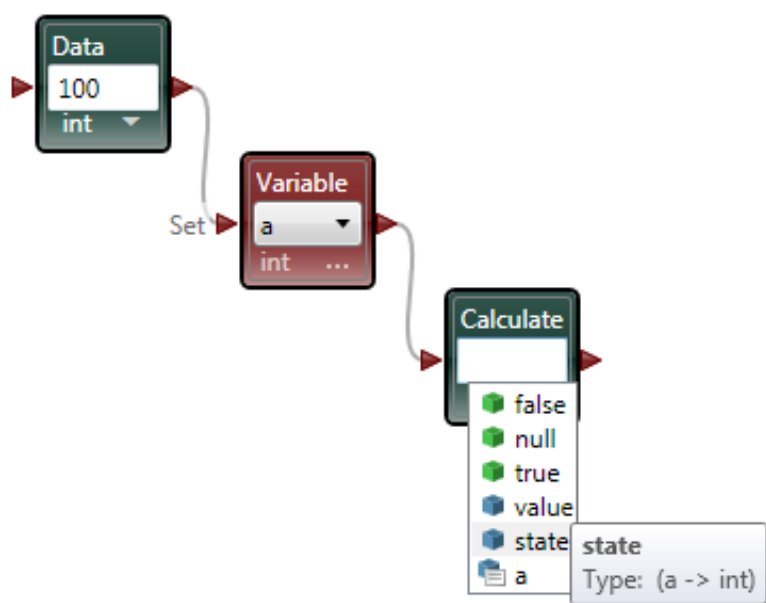


Рисунок 1.22 – Выбор параметра блока вычисления

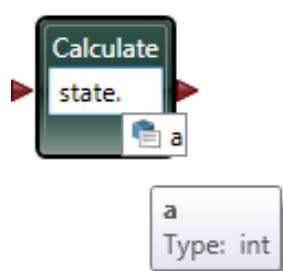


Рисунок 1.23 – Определение переменной в блоке вычисления

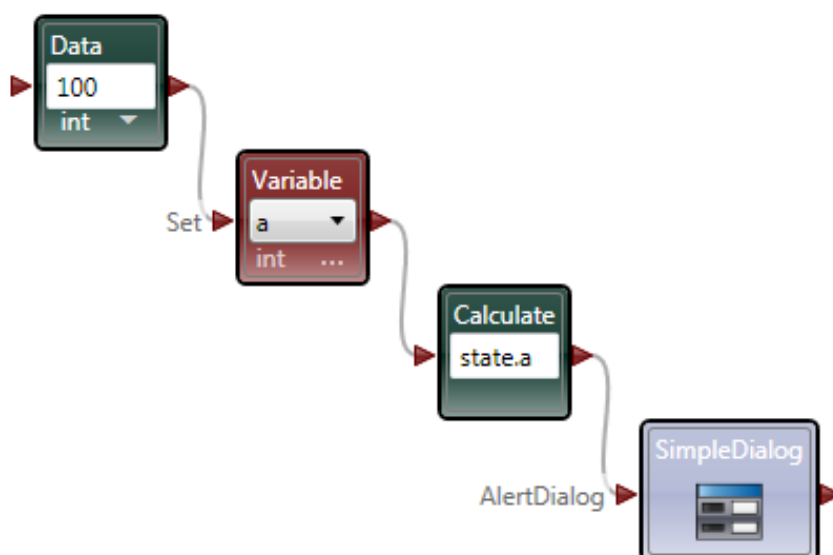


Рисунок 1.24 – Диаграмма считывания значения переменной

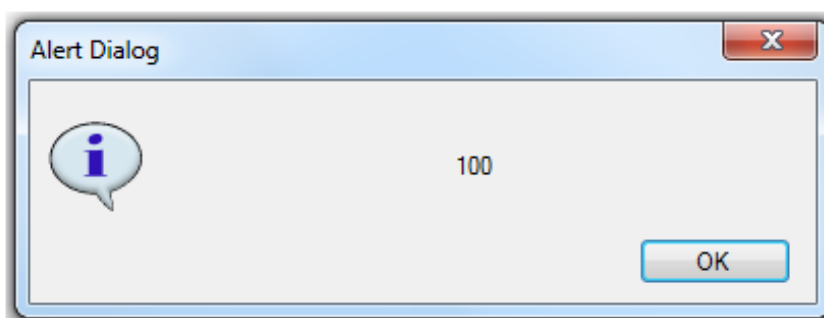


Рисунок 1.25 – Результат работы программы считывания значения переменной

1.2.4. Методика организации ввода-вывода данных с помощью блока вычисления

1. Для организации вывода текста с помощью блока вычислений подготовьте блоки, представленные на рис. 1.26.

2. Введите в блок данных свое имя (поддерживается кириллица), соедините блок данных с блоком вычисления, затем в блоке вычисления введите следующую строку: «Мое имя» + value.

3. Соедините блок вычисления с сервисом, и выполните необходимые операции. В результате работы программы должен быть следующий результат (рис. 1.27).

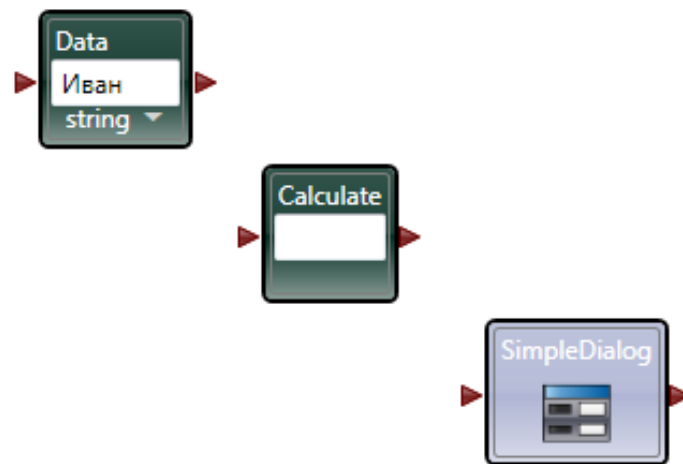


Рисунок 1.26 – Блоки диаграммы вывода текста

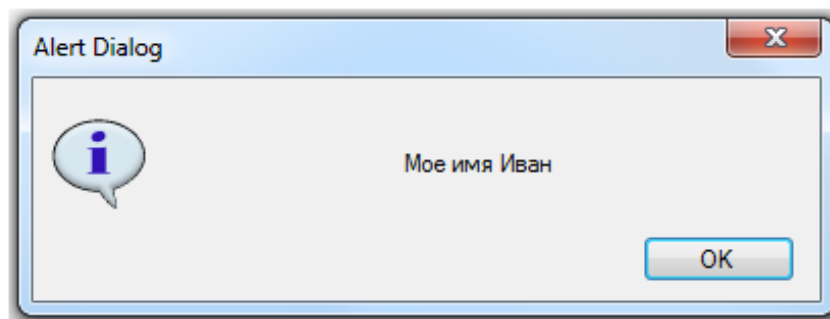


Рисунок 1.27 – Результат работы диаграммы вывода текста

4. Для организации ввода данных поместите в рабочее окно диаграммы блок данных и 3 сервиса симуляции консольного интерфейса. Введите в блоке данных следующую строку «Введите ваше имя: ». Соедините блок данных и первый сервис. В появившемся окне выберите опцию Write, далее в списке выберите значение value.

5. Соедините первый и второй сервисы. Выберите исходящее сообщение Write – Success и опцию Read Line.

6. Соедините первый и второй сервисы. Выберите исходящее сообщение Read Line – Success и опцию Write Line, далее в списке выберите Value. В итоге должна получиться следующая диаграмма (рис. 1.28).

7. Запустим на выполнение полученную программу. Результат её выполнения представлен на рис. 1.29.

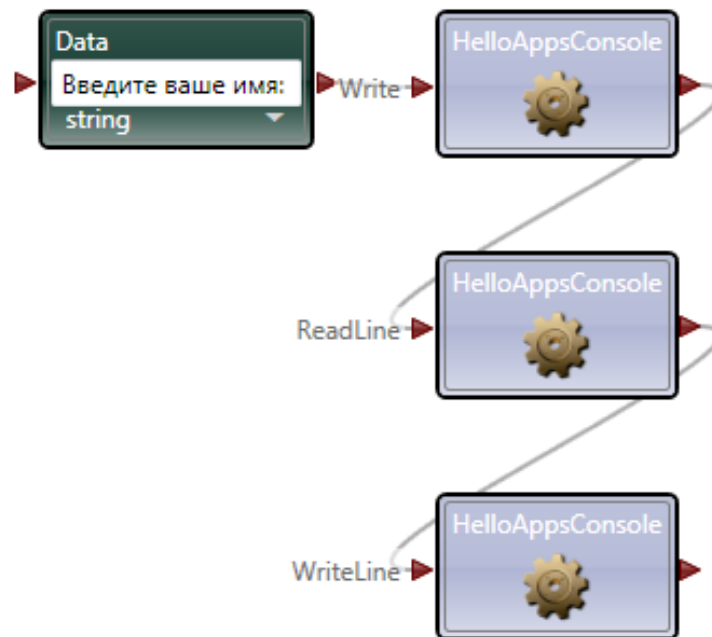


Рисунок 1.28 – Диаграмма ввода данных

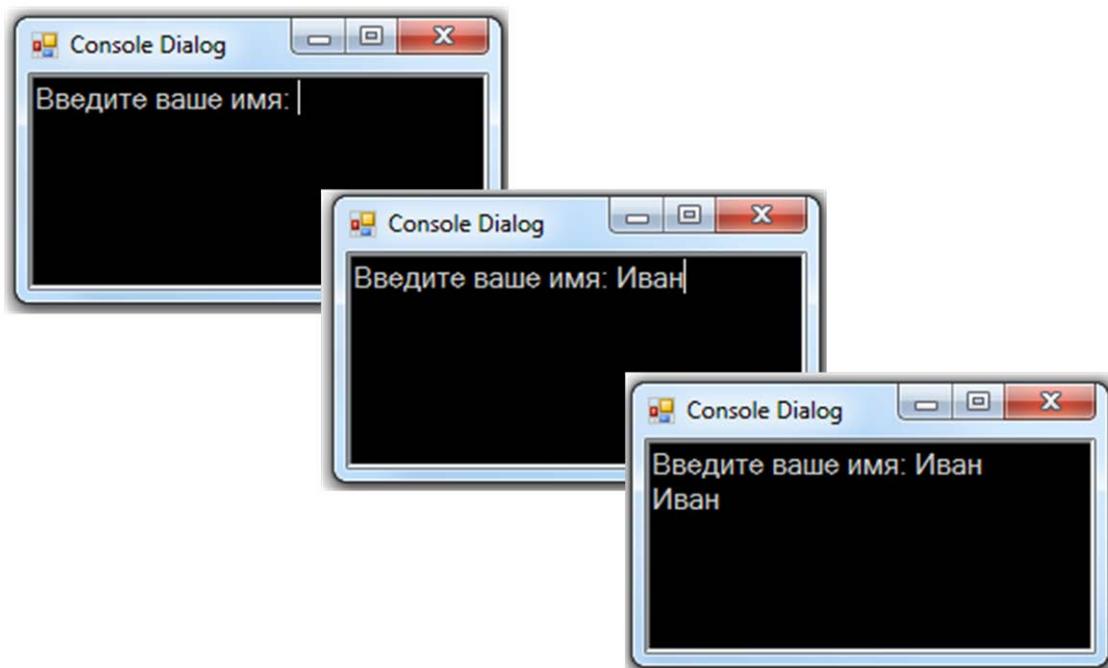


Рисунок 1.29 – Результат работы диаграммы ввода данных

8. Для организации ввода-вывода данных модифицируйте предыдущую диаграмму. Добавьте блок вычисления, включив его после второго сервиса и выбрав Read Line – Success.

9. Введите в блоке вычисления следующую строку **«Ваше имя: »** +. Далее выберите из списка значение Value. Затем соедините блок вычисления с сервисом, выберите опцию Write Line, установите значение Value из списка.

В итоге должна получиться следующая диаграмма (рис. 1.30). Запустив на выполнение программу, получим результат представленный на рис. 1.31.

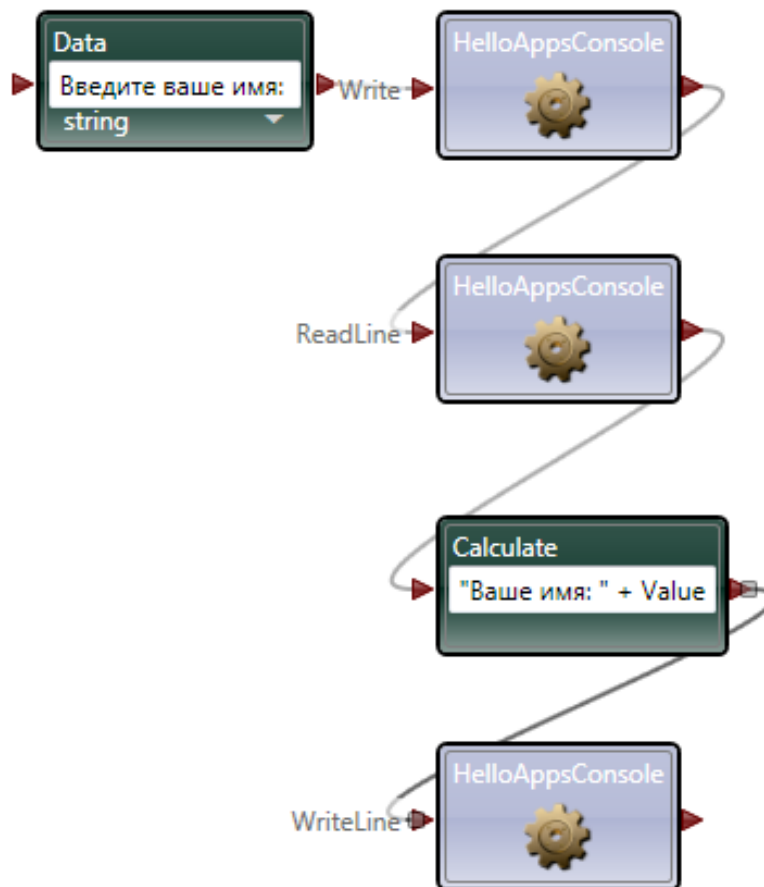


Рисунок 1.30 – Диаграмма ввода-вывода данных

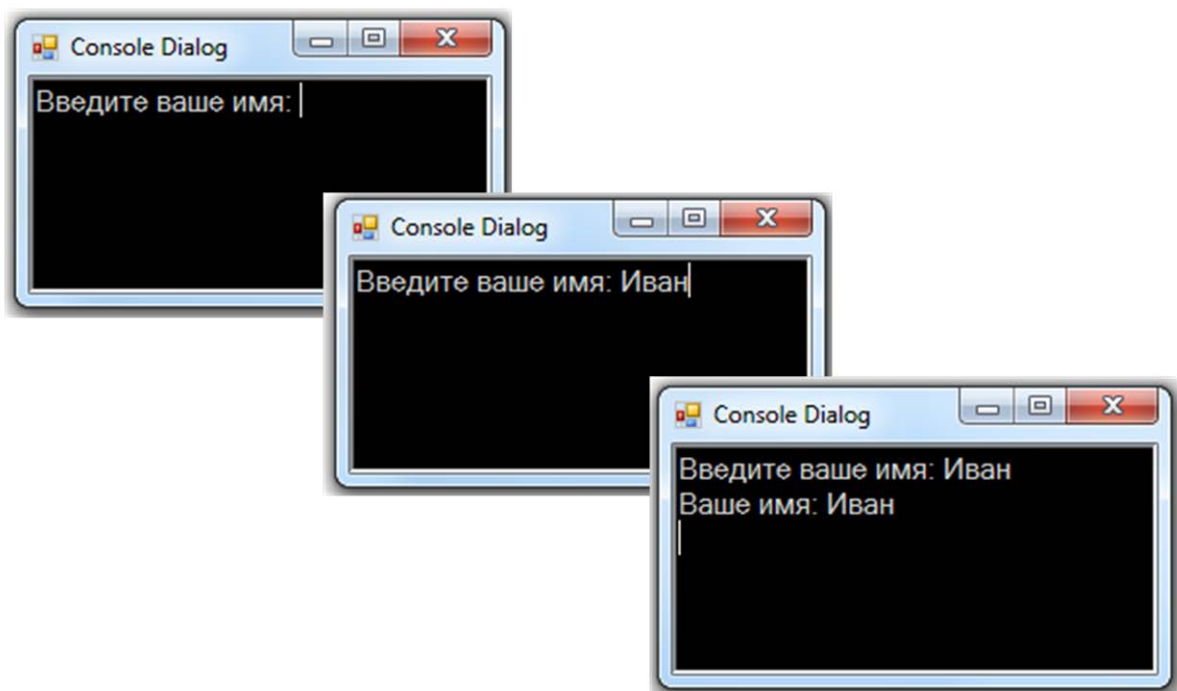


Рисунок 1.31 – Результат работы диаграммы ввода-вывода данных

1.2.5. Методика организации вычисления суммы

1. Для организации вычисления суммы двух констант добавьте два блока данных, блок слияния, блок вычисления и сервис симуляции консольного интерфейса.

2. Заполните поля блоков – определите константы, ячейки и операции для работы с данными. Соедините блок вычислений с сервисом. Задайте параметры соединения и данных (рис. 1.32).

3. Запустите на выполнение программу (рис. 1.33).

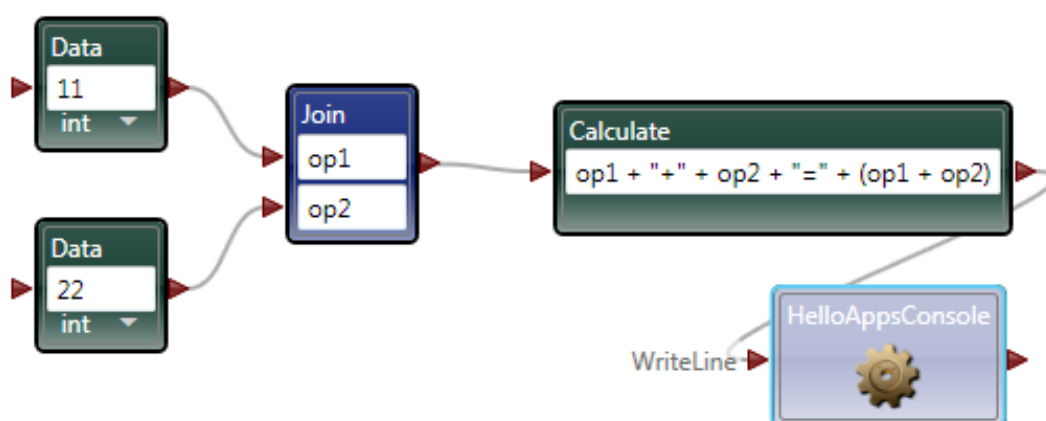


Рисунок 1.32 – Диаграмма вычисления суммы двух констант

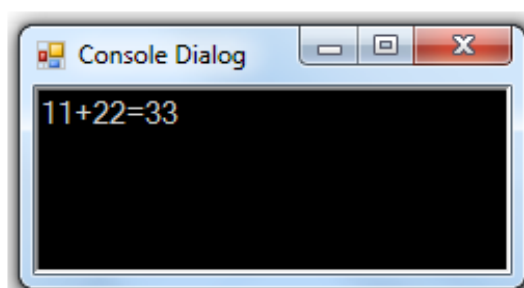


Рисунок 1.33 – Результат вычисления суммы двух констант

4. Для организации вычисления суммы двух переменных добавьте блок данных, блок вычислений, блок переменной и два сервиса симуляции консольного интерфейса.

5. Определите переменные `op1` и `op2`, тип переменных «int». Соедините блоки и определите все операции для работы с данными (рис. 1.34). Строка «(int)Value» в блоке вычисления выполняет функцию конвертирования символьных значений, введенных в консоль, в тип `int`.

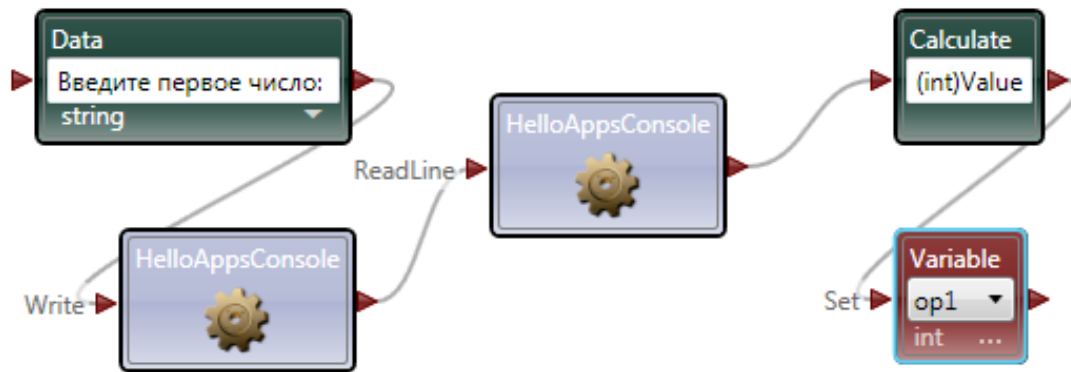


Рисунок 1.34 – Первый фрагмент диаграммы вычисления суммы двух переменных

6. Создайте второй фрагмент диаграммы по структуре идентичный первому. Соедините блок переменной первой диаграммы с блоком данных второй диаграммы (рис. 1.35).

7. Добавьте блок вычисления и сервис симуляции консольного интерфейса. Соедините блок переменной второго фрагмента диаграммы с блоком вычисления, затем блок вычисления с сервисом консолю (рис. 1.36). Конечный вид диаграммы вычисления суммы двух переменных представлен на рис. 1.37.

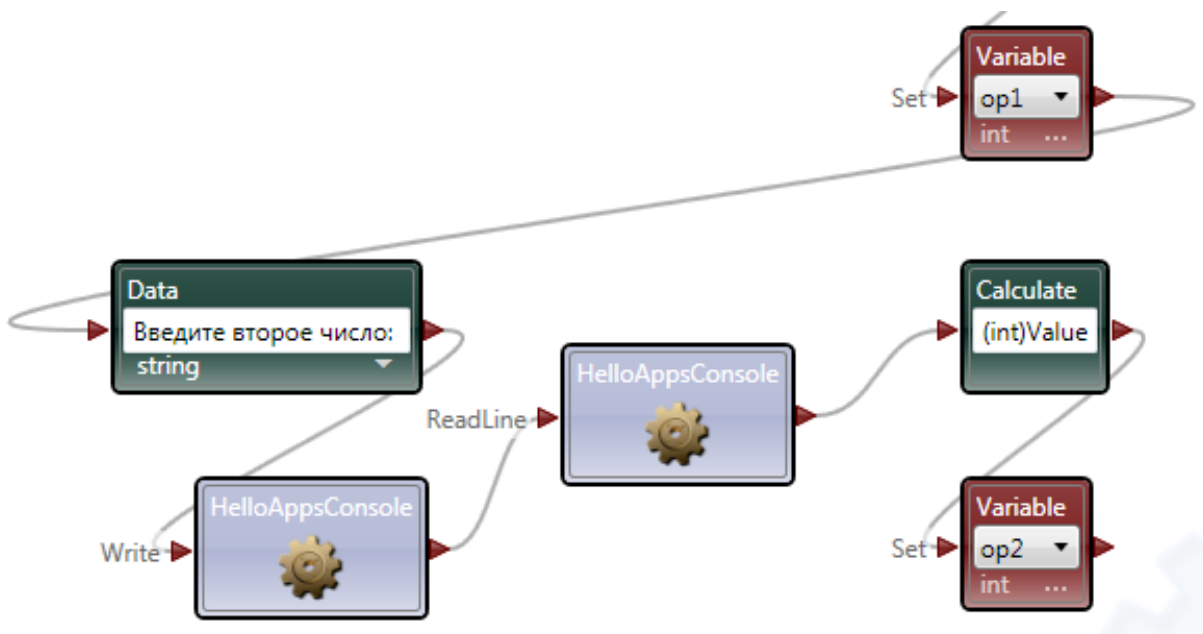


Рисунок 1.35 – Второй фрагмент диаграммы

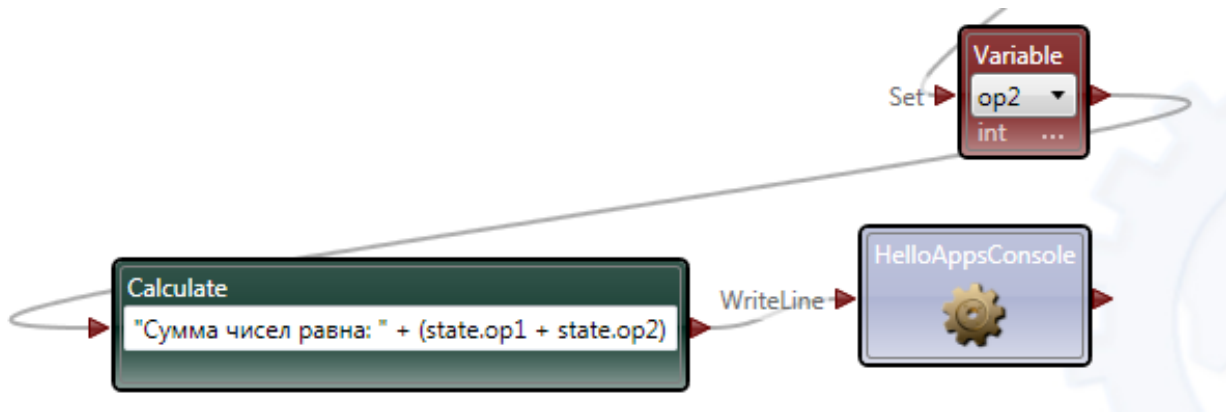


Рисунок 1.36 – Третий фрагмент диаграммы

8. Запустите программу на выполнение. Результат выполнения программы вычисления суммы двух переменных отражен на рис. 1.38.

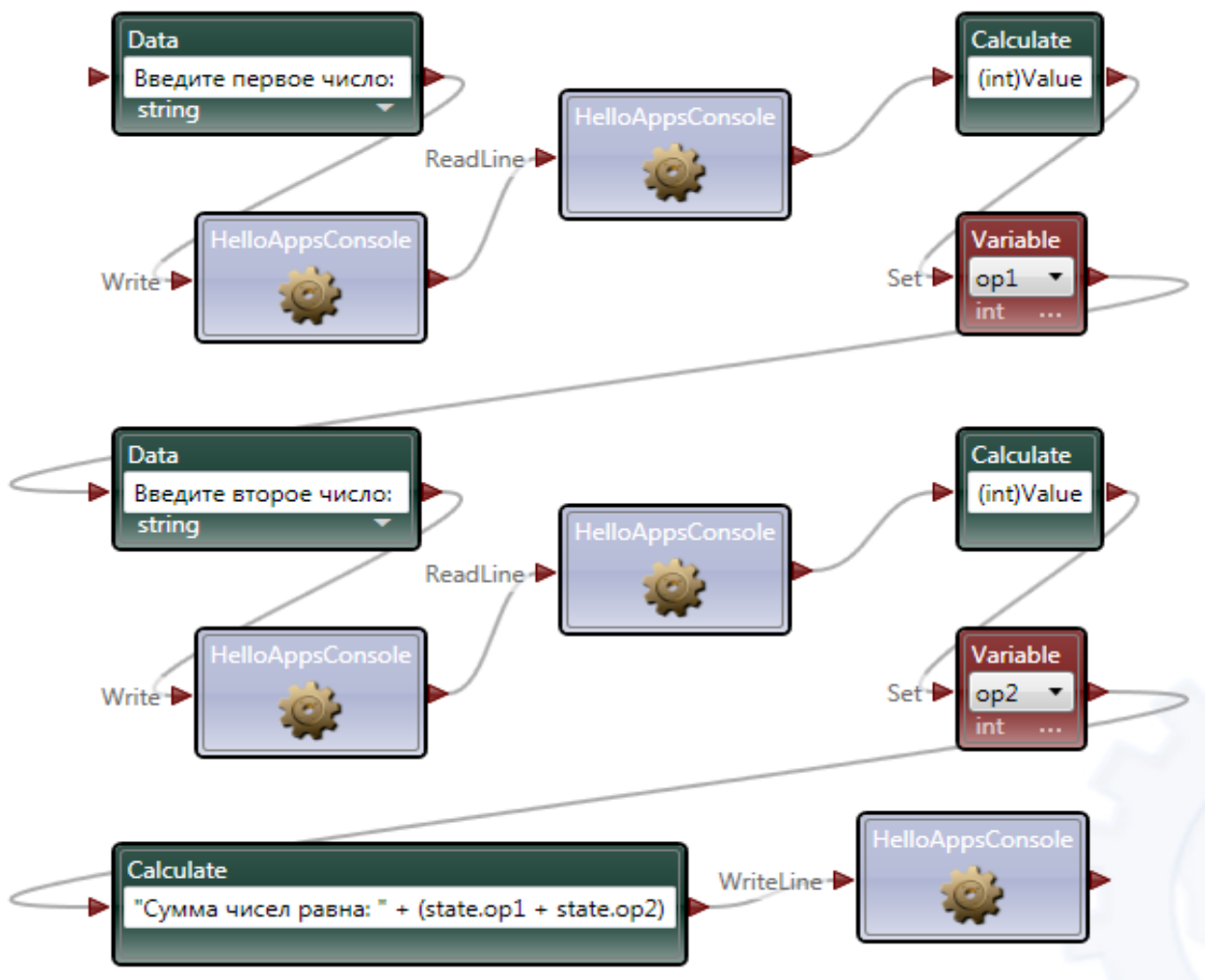


Рисунок 1.37 – Конечный вид диаграммы вычисления суммы двух переменных

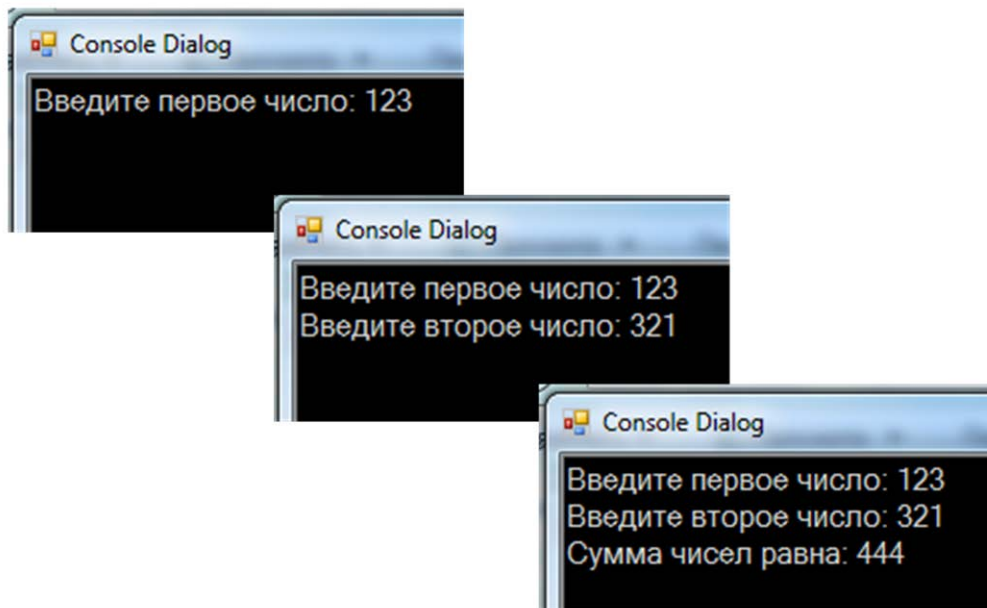


Рисунок 1.38 – Результат выполнения программы вычисления

1.2.6. Методика организации сравнения констант и переменных

1. Для организации сравнения двух констант добавьте блоки и сервисы согласно рис. 1.39. Заполните блоки данных, блок слияния. Соедините блоки и сервисы. Заполните блоки вычисления.

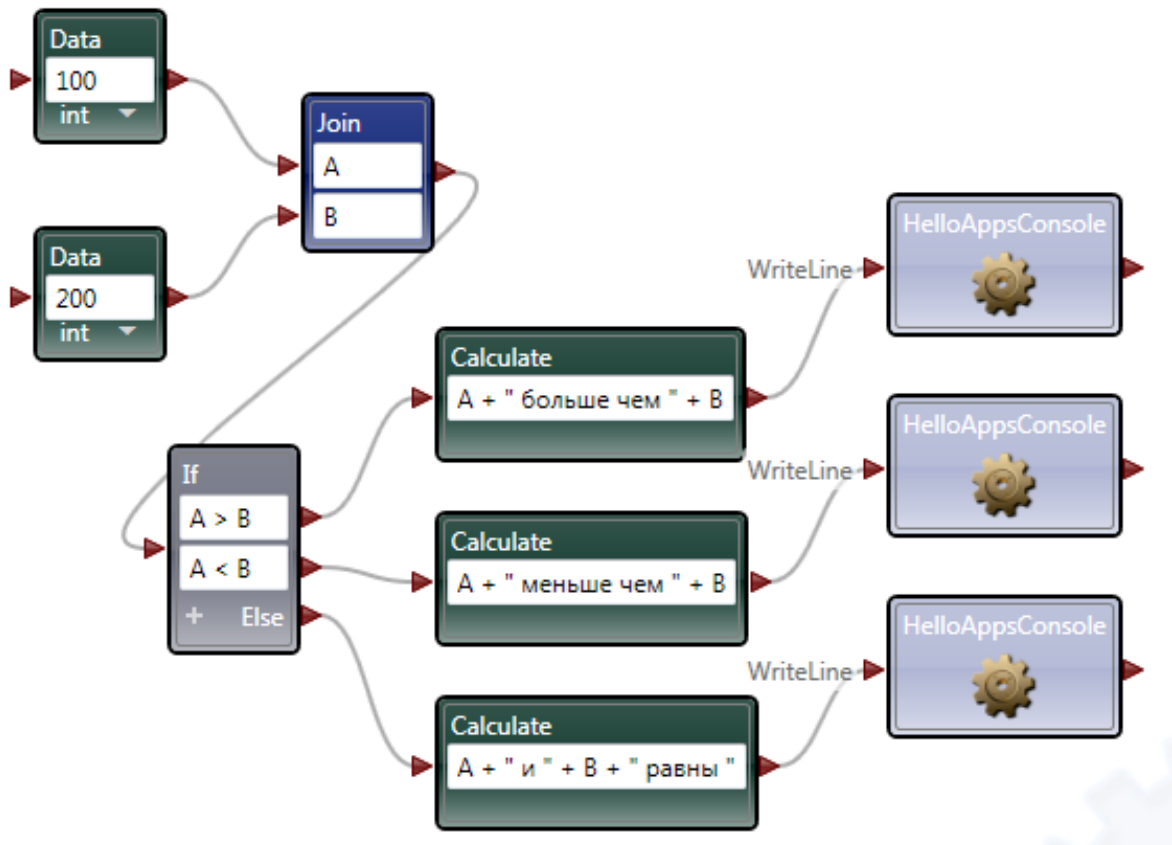


Рисунок 1.39 – Конечный вид диаграммы сравнения двух констант

2. Запустите программу на выполнение. Результат выполнения программы сравнения двух констант отражен на рис. 1.40.

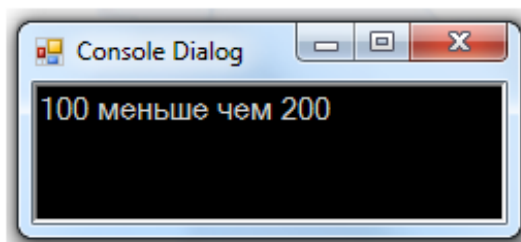


Рисунок 1.40 – Результат выполнения программы сравнения

3. Для организации сравнения двух переменных добавьте первый и второй фрагмент диаграммы из методики 1.1.5. Соедините блок переменной первой диаграммы с блоком данных второй диаграммы. Определите переменные `op1` и `op2`, тип переменных «int».

4. Добавьте блок условия и три блока вычисления, а так же три сервиса симуляции консольного интерфейса.

5. Соедините блок переменной второй части диаграммы с блоком условия, соедините между собой добавленные блоки и сервисы, заполните необходимые блоки (рис. 1.41).

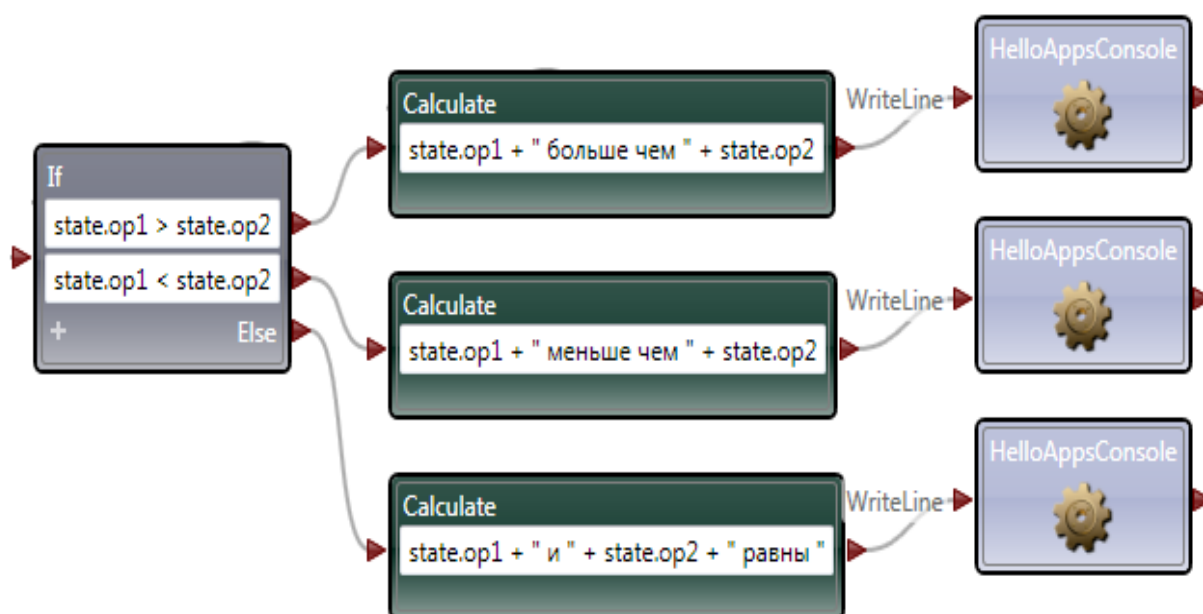


Рисунок 1.41 – Третий фрагмент диаграммы сравнения двух переменных

6. Конечный вид диаграммы сравнения двух переменных представлен на рис. 1.42. Запустите программу на выполнение. Результат выполнения программы сравнения двух переменных отражен на рис. 1.43.

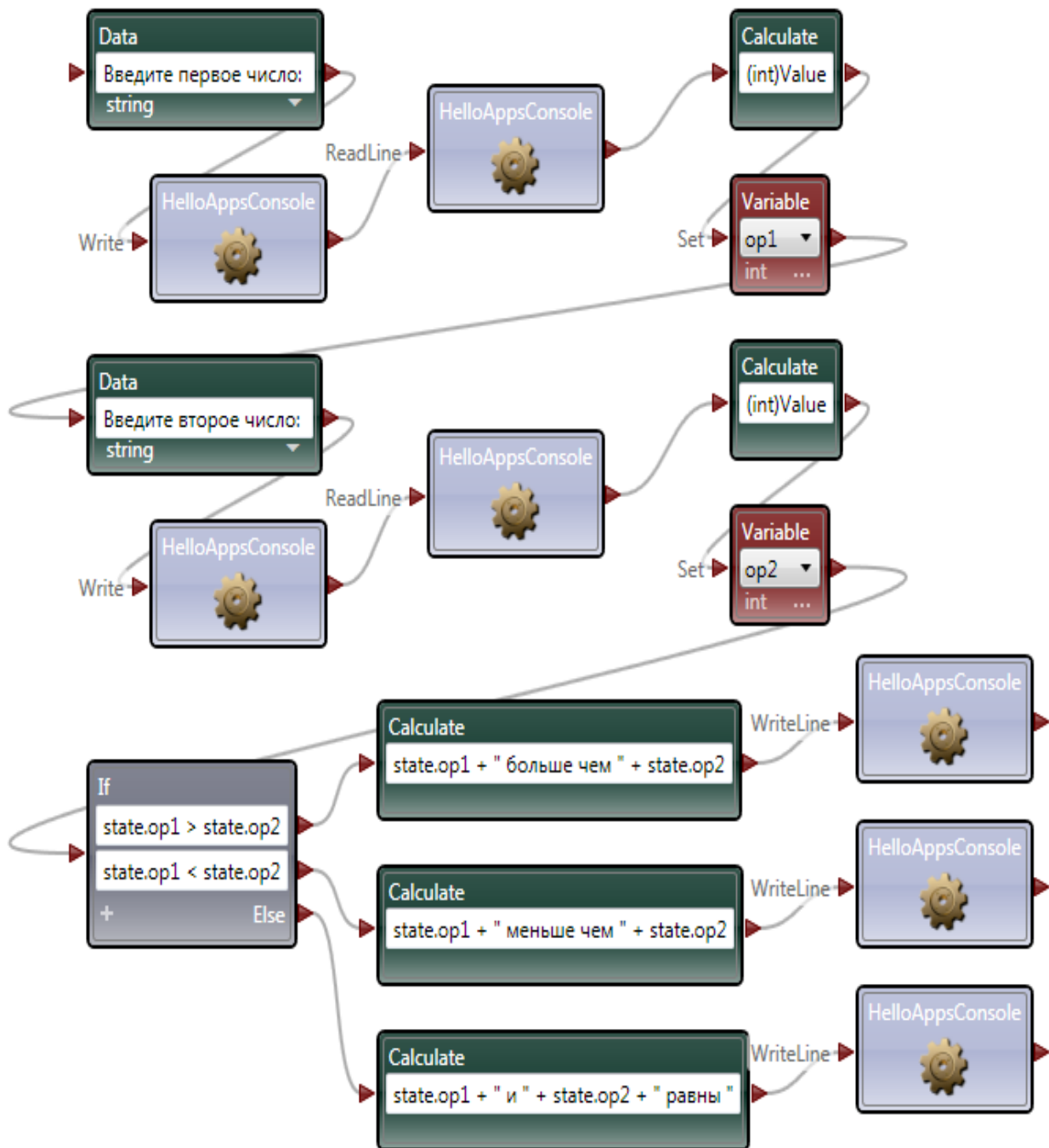


Рисунок 1.42 – Конечный вид диаграммы сравнения двух переменных

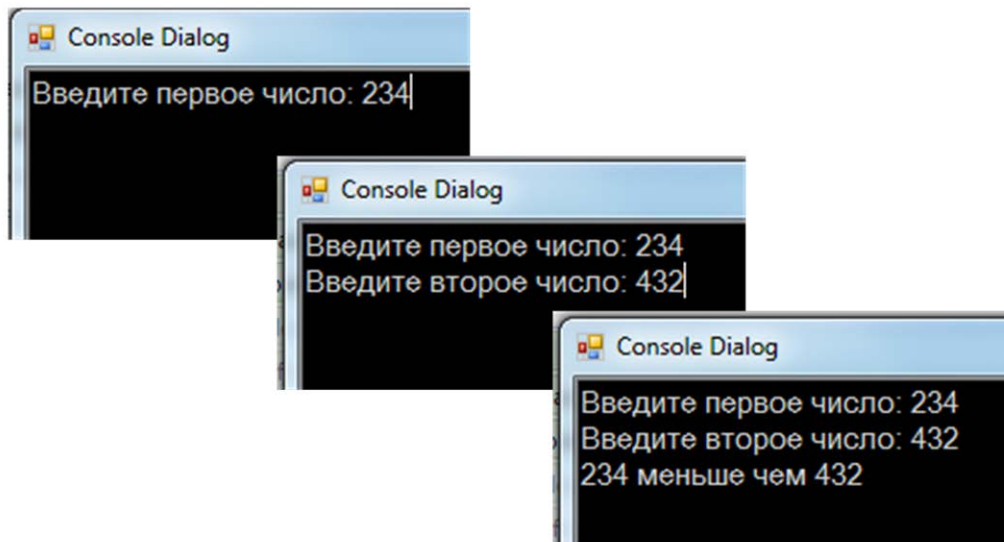


Рисунок 1.43 – Результат выполнения программы сравнения двух переменных

1.2.7. Методика организации сравнения символьных строк

1. Для организации сравнения символьных строк добавьте блоки и сервисы согласно рис. 1.44.

2. Заполните блоки данных, блок слияния и блок условия. Соедините блоки и сервисы. Заполните блоки вычисления.

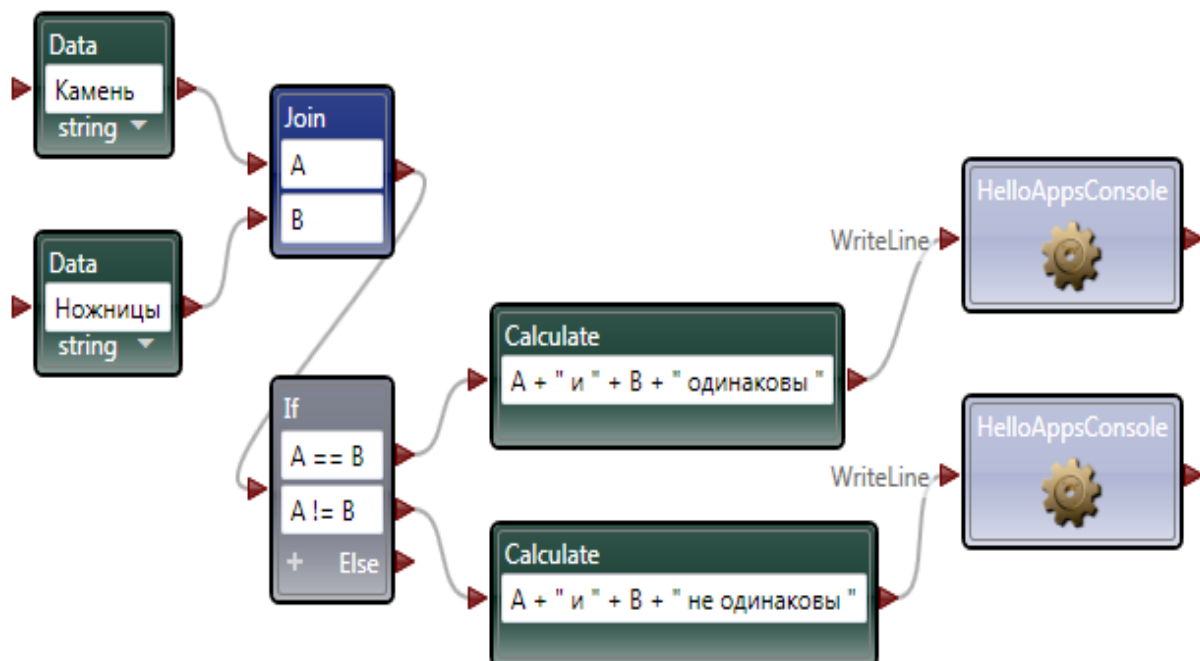


Рисунок 1.44 – Диаграмма сравнения символьных строк

3. Запустите программу на выполнение. Результат выполнения программы сравнения символьных строк отражен на рис. 1.45.

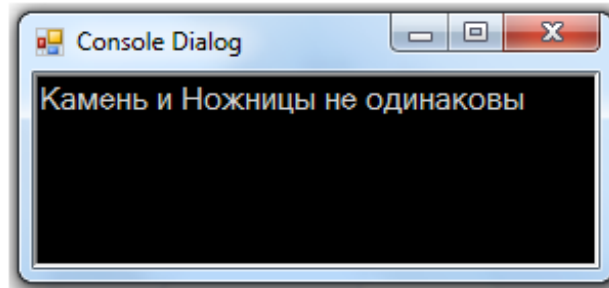


Рисунок 1.45 – Результат выполнения программы сравнения символьных строк

4. Модифицируйте диаграмму сравнения символьных строк введением блока простого слияния (рис. 1.46). Запустите программу на выполнение. Сравните полученный результат выполнения программы с предыдущим результатом.

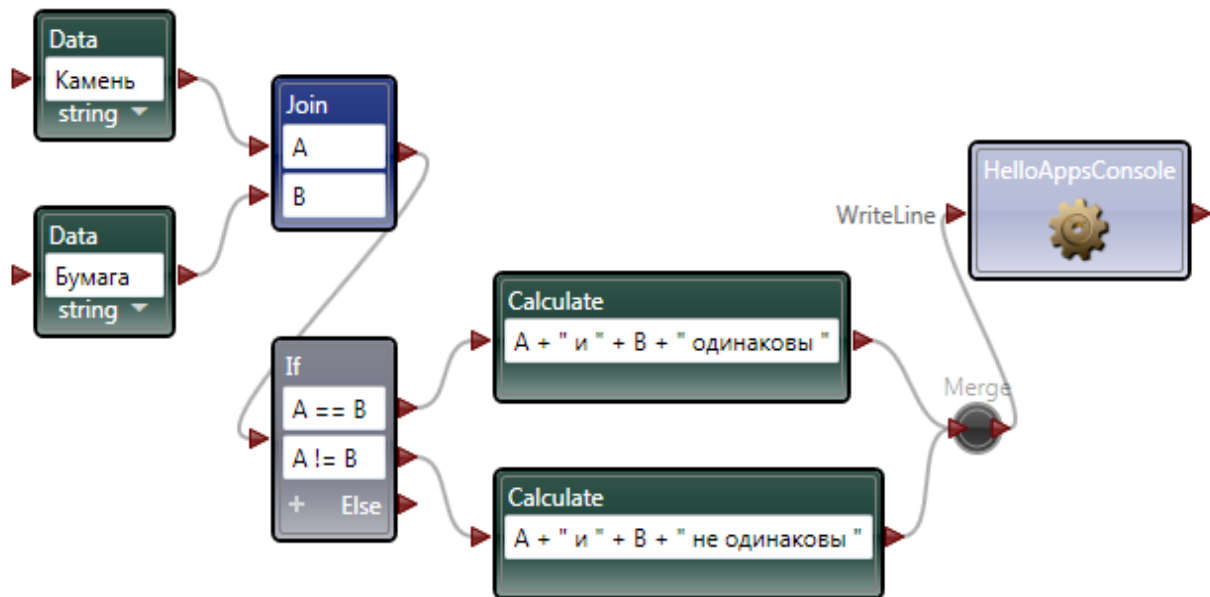


Рисунок 1.46 – Модифицированная диаграмма сравнения символьных строк

1.3. Лабораторная работа 1. Основы программирования мультиточечных приложений в среде Visual Programming Language

1.3.1. Цель и содержание

Целью лабораторной работы является приобретение студентами практических навыков работы в среде визуального программирования Visual Programming Language с блоками данных, переменной и вычисления, выполнения вывода информации на дисплей с помощью сервиса простого диалога и сервиса симуляции консольного интерфейса, а также использования блока сравнения и блока простого слияния.

Содержание

1. Организация считывания значения переменной с помощью сервиса простого диалога.
2. Организация ввода-вывода данных с помощью блока вычислений.
3. Изучение опций работы сервиса симуляции консольного интерфейса.
4. Изучение особенностей использования блока сравнения.

1.3.2. Аппаратура и материалы

Для изучения материалов этой лабораторной работы необходимо:

1. Компьютер, удовлетворяющий минимальным системным требованиям для установки Microsoft Robotics Developer Studio R4.
2. Установленная платформа RDS R4.

1.3.3. Порядок выполнения работы

1. Изучение теоретического материала и методических рекомендаций.
2. Выполнение задания, предлагаемого в лабораторной работе.

Задание:

1. Установите HelloApps SPL Tool, запустив файл SPLXSetup.msi.
2. Установите пакет обновлений HelloApps SPL Services, запустив файл SPL_Installation_Full_Pack.splx.
3. Выполните задание из таблицы 1.1 согласно своему варианту.

Таблица 1.1. Варианты заданий для самостоятельного решения

№ вар.	Задание
1.	1. Решите уравнение $z = \frac{(x+5) \times y}{2}$ при помощи блока вычислений. Вывод вычислений произведите при помощи сервиса простого диалога. 2. Произведите сравнение двух введенных с клавиатуры чисел.
2.	1. Решите уравнение $z = x^2 - 5xy - 3y - 7$ при помощи блока вычислений. Вывод вычислений произведите при помощи сервиса простого диалога. 2. Произведите сравнение трех введенных с клавиатуры чисел.
3.	1. Решите уравнение $z = (x-3)^2 + y^2$ при помощи блока вычислений. Вывод вычислений произведите при помощи сервиса простого диалога. 2. Определите, какое из двух введенных с клавиатуры чисел больше, если первое число равно $x = x - 10$, а второе $y = y + 14$?
4.	1. Решите уравнение $z = 3x + y^3 + 4$ при помощи блока вычислений. Вывод вычислений произведите при помощи сервиса простого диалога. 2. Произведите сравнение двух введенных с клавиатуры чисел, если первое число $x = 2x + 4$, а второе $y = (x-3)^2$.
5.	1. Решите уравнение $z = 8x + x^2 + 13y$ при помощи блока вычислений. Вывод вычислений произведите при помощи сервиса простого диалога. 2. Произведите сравнение двух введенных с клавиатуры чисел, если первое число $x = 2x^2 + x$, а второе $y = y^2 + 20$.
6.	1. Решите уравнение $z = (x+4)^2 + (y-6)^2$ при помощи блока вычислений. Вывод вычислений произведите при помощи сервиса простого диалога. 2. Произведите сравнение двух введенных с клавиатуры чисел, если первое число $x = (x-2)^3$, а второе $y = 21y$.
7.	1. Решите уравнение $z = (x+8)(y+2)$ при помощи блока вычислений. Вывод вычислений произведите при помощи сервиса простого диалога. 2. Произведите сравнение двух введенных с клавиатуры чисел, если первое число $x = (3x-5)^2$, а второе $y = 7y + 29$.
8.	1. Решите уравнение $z = \frac{(x^3+7) \times y^3}{2}$ при помощи блока вычислений. Вывод вычислений произведите при помощи сервиса простого диалога. 2. Произведите сравнение двух введенных с клавиатуры слов. Определите, в каком из них наибольшее количество знаков.

№ вар.	Задание
9.	1. Решите уравнение $z = \frac{(x \times y)^2}{2}$ при помощи блока вычислений. Вывод вычислений произведите при помощи сервиса простого диалога. 2. Произведите сравнение двух введенных с клавиатуры предложений. Определите, в каком из них наибольшее количество знаков.
10.	1. Решите уравнение $z = x^2 - y$ при помощи блока вычислений. Вывод вычислений произведите при помощи сервиса простого диалога. 2. Произведите сравнение двух введенных с клавиатуры слов. Сделайте это при двух условиях: в первом случае сравнивается количество знаков, во втором сами слова.
11.	1. Решите уравнение $z = (\frac{1}{6}x - y)(y^2 - 4)$ при помощи блока вычислений. Вывод вычислений произведите при помощи сервиса простого диалога. 2. Произведите сравнение трех введенных с клавиатуры слов. Определите, в каком из них наибольшее количество знаков
12.	1. Решите уравнение $z = \frac{(x + 5)(3 + y)}{2}$ при помощи блока вычислений. Вывод вычислений произведите при помощи сервиса простого диалога. 2. Произведите сравнение двух слов введенных с клавиатуры и одной константы. Определите, в каком из них наибольшее количество знаков.

1.4. Контрольные вопросы

1. Перечислите типы хранимого значения в блоке данных.
2. Перечислите типы переменных в блоке переменной.
3. Назовите арифметические и логические операции, реализуемые в блоке вычислений.
4. Охарактеризуйте опции работы сервиса симуляции консольного интерфейса.
5. Охарактеризуйте понятия процесса и потока.
6. Перечислите возможные пути создания ячейки потока в блоке слияния.
7. Назовите логические операции, реализуемые в блоке условия.
8. Охарактеризуйте особенности применения блока простого слияния.

ГЛАВА 2. ОСОБЕННОСТИ МОДУЛЬНО-ИЕРАРХИЧЕСКОЙ АРХИТЕКТУРЫ СЕРВИСОВ

Процесс проектирования архитектуры программного обеспечения состоит в проектировании структуры всех его компонент, функционально связанных с решаемой задачей, включая сопряжения между ними и требования к ним [29]. Во время разработки архитектуры программного обеспечения выполняется его модульно-иерархическое построение.

Модуль – это замкнутая программа, которую можно вызвать из любого другого модуля в программе и можно отдельно компилировать.

Стремление к созданию модульных программ объясняется следующими факторами:

- программные модули, как правило, решают небольшую функциональную задачу, используют на входе и на выходе немного данных; внутренние переменные модуля не связаны с внутренними переменными других модулей; поэтому отдельные модули могут создаваться и отлаживаться различными разработчиками независимо друг от друга;
- модульные программы легко читать, сопровождать и модифицировать; исправление отдельного модуля вызывает минимальные изменения в других модулях, связанных с ним по управлению и информации;
- модульные программы обладают повышенной надежностью, так как при их разработке существует возможность распределения работ по созданию модулей различной сложности и важности между программистами различной квалификации;
- можно создавать библиотеки наиболее употребительных подпрограмм, которые затем можно использовать в качестве комплектующих частей при разработке других приложений;
- процедура загрузки всей программы в оперативную память упрощается при использовании метода оверлейности;

– возникает много естественных контрольных точек для наблюдения за продвижением проекта по управлению и по информации.

2.1. Краткие теоретические сведения

Иерархия представляет собой свойство упорядоченного множества компонентов, между которыми установлено отношение приоритета [29]. Компоненты, между которыми отсутствует предпочтительность, образуют один иерархический уровень.

Архитектуре программного обеспечения как иерархической системе присущ ряд свойств, важнейшими из которых являются:

- вертикальная соподчиненность, заключающаяся в последовательном упорядоченном расположении взаимодействующих компонент, составляющих данный комплекс программ;
- компоненты одного уровня обеспечивают реализацию функций компонент следующего уровня;
- каждый уровень иерархии реализуется через функции компонент более нижних уровней;
- каждый компонент знает о компонентах более низких уровней и ничего не знает о компонентах более высоких уровней;
- право вмешательства и приоритетного воздействия на компоненты любых уровней со стороны компонент более высоких иерархических уровней;
- взаимозависимость действий компонент верхних уровней от реакций на воздействия и от функционирования компонент нижних уровней, информация о которых передается верхним уровням.

В среде Microsoft Robotics Developer Studio R4 при написании программ большого объема требуется разделение программного кода (взаимосвязанных блоков и сервисов) на несколько диаграмм. С помощью такого подхода удобно реализовать иерархию сервисов, когда один сервис вызывает работу других (рис. 2.1).



Рисунок 2.1 – Иерархия сервисов

На уровне сервисов данная концепция осуществляется с помощью, так называемых **нотификаций** (Notification) или извещений. Допустим, в диаграмме присутствует сервис таймера, с правой стороны сервиса находится элемент в виде круга (рис. 2.2), это соединительная точка для отправления нотификации.

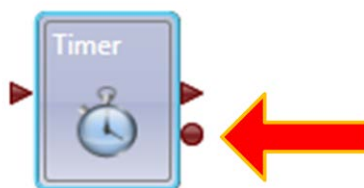


Рисунок 2.2 – Сервис таймера

Нотификация является сообщением о состоянии сервиса. Сервис изменяет свое состояние при поступлении в него сообщений от других сервисов или блоков. На примере таймера это может выглядеть как, например истечение установленного времени. В случае кнопочного интерфейса – нажатие на кнопку пользователем. В этом случае в сервисе формируется сообщение, которое может быть отправлено через разъем нотификации. Проиллюстрируем вышесказанное примером приведенном на рис. 2.3.

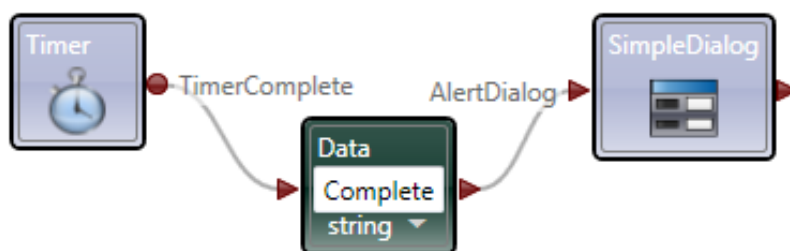


Рисунок 2.3 – Реализация работы с нотификацией

Через определенное время состояние таймера изменится и сформируется сообщение нотификации, которое будет послано в блок данных. В блоке данных формируется новое сообщение, содержащее символьную строку «Complete» (рис. 2.4). Обратите внимание, что для пересылки нотификации должны использоваться одни и те же сервисы, т. е. при добавлении сервиса (рис. 2.5) необходимо выбрать уже существующий. В противном случае разные сервисы будут выполняться отдельно.

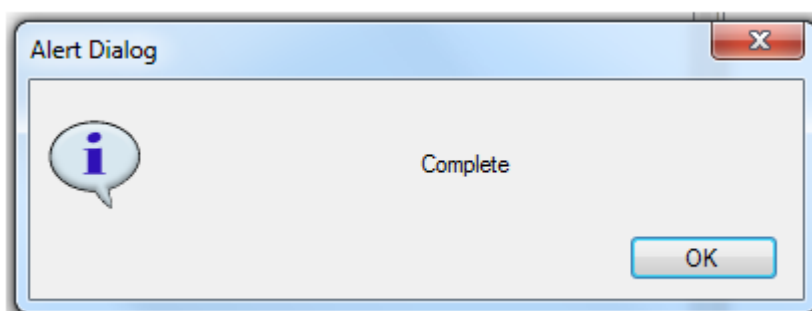


Рисунок 2.4 – Результат работы программы

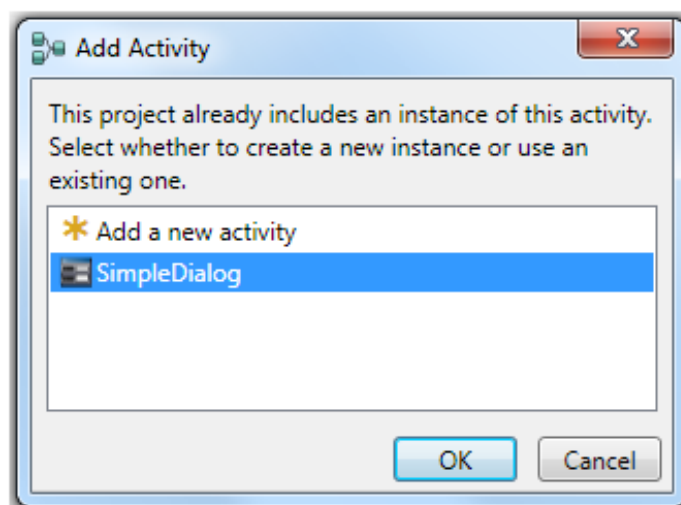


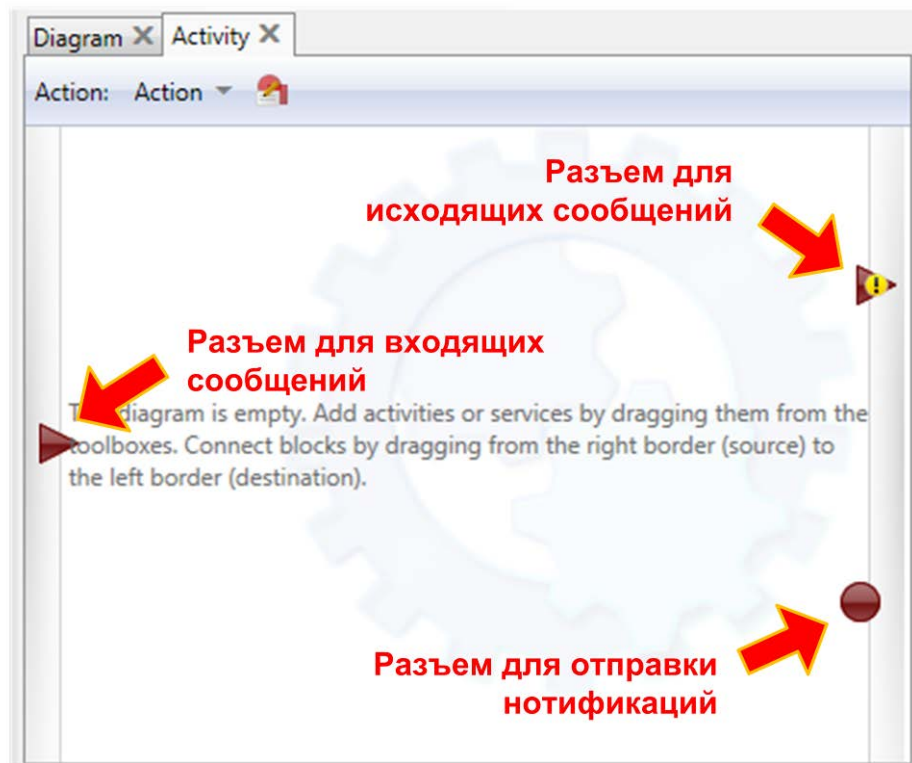
Рисунок 2.5 – Добавление сервиса

Модуль (Activity) в среде RDS 4 – это блок (рис. 2.6), внутри которого можно создавать диаграммы для экономии места в рабочей области диаграммы. В итоге мы получаем вложенные друг в друга диаграммы. Созданный модуль можно скомпилировать как сервис и использовать в дальнейшем написании программ.



Рисунок 2.6 – Блок модуля

При щелчке по блоку модуля открывается окно (рис. 2.7) в которое добавляются сервисы и блоки. Модуль имеет элементы входа и выхода сообщений, а так же выход для нотификаций. При создании модуля необходимо определить входящие и исходящие сообщения и нотификации.



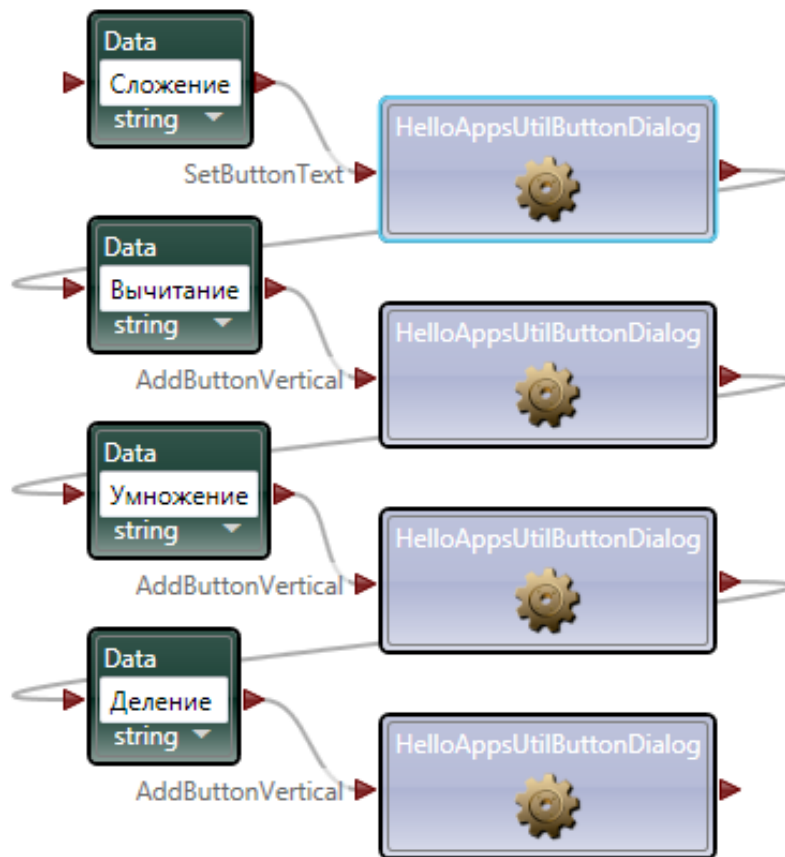


Рисунок 2.8 – Первая диаграмма

2. При соединении блоков данных с сервисами, в окне данных выберите пункт value для назначения Text первого сервиса и для назначений Name и Text последующих сервисов.

3. Запустите на выполнение диаграмму. Результат работы программы представлен на рис. 2.9.

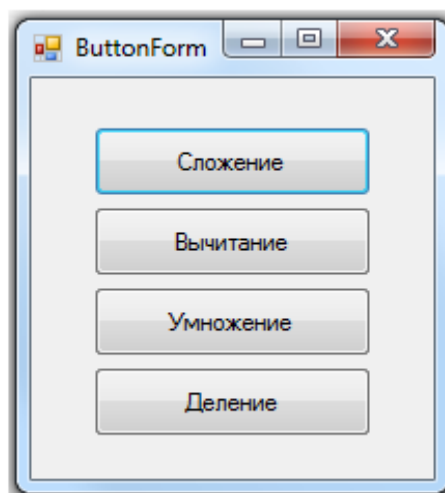


Рисунок 2.9 – Результат работы программы

4. Создайте две вкладки диаграммы. В окне одной из диаграмм добавьте блоки и сервисы, соедините их, заполните блоки (рис. 2.10). В другой вкладке создайте третью диаграмму (рис. 2.11).

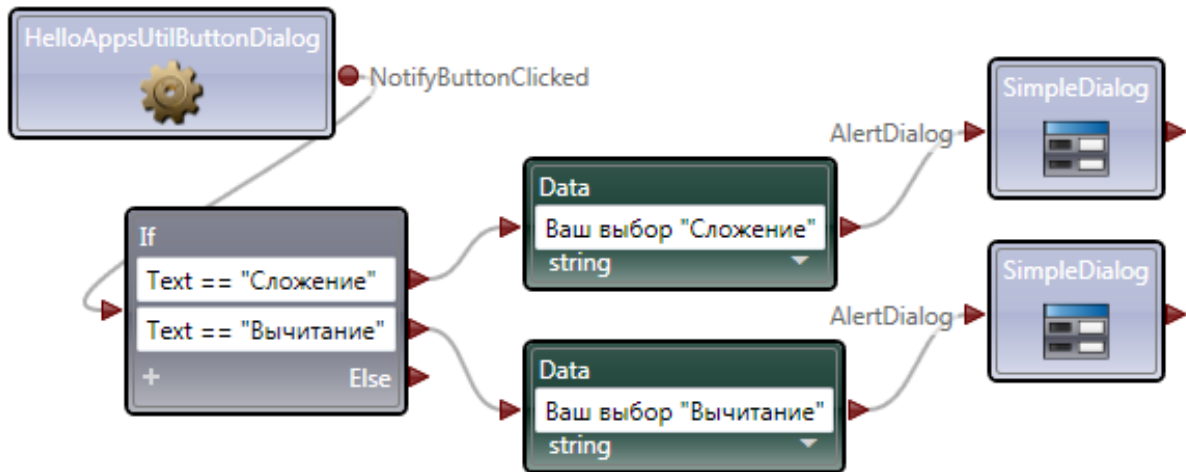


Рисунок 2.10 – Вторая диаграмма

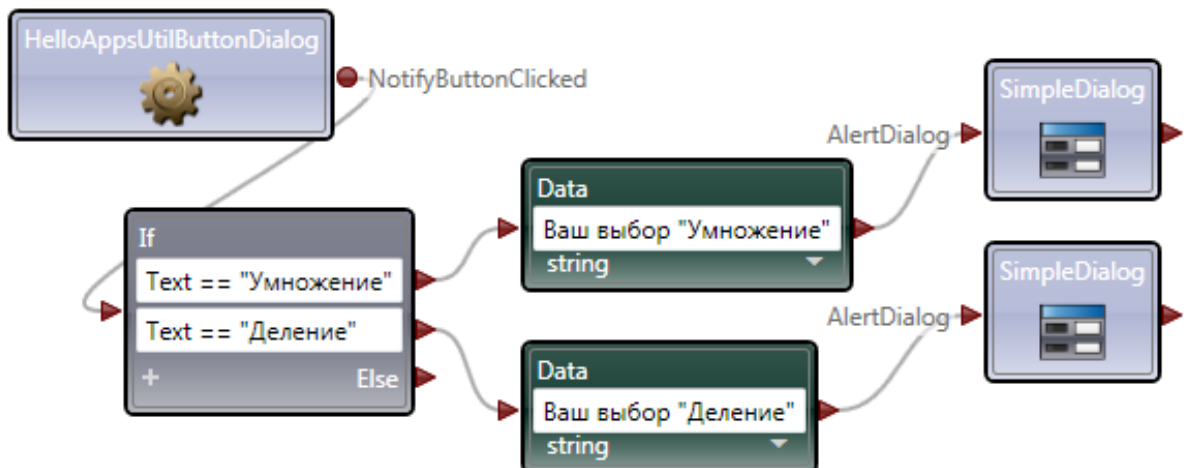


Рисунок 2.11 – Третья диаграмма

5. Запустите на выполнение программу. В зависимости от нажатой кнопки в окне диалога (рис. 2.9), генерируется сообщение (рис. 2.12).

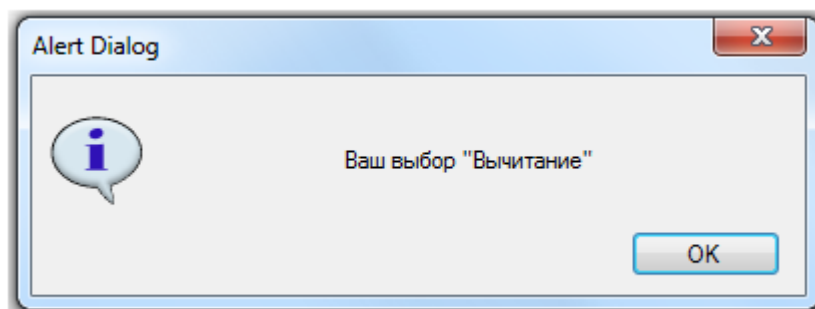


Рисунок 2.12 – Окно сообщения

2.2.2. Методика декомпозиции программы с помощью модуля

1. Рассмотрим программу работающую следующим образом (рис. 2.13).
2. Для организации декомпозиции программы добавьте сервис и модуль (рис. 2.14). Откройте модуль двойным щелчком.

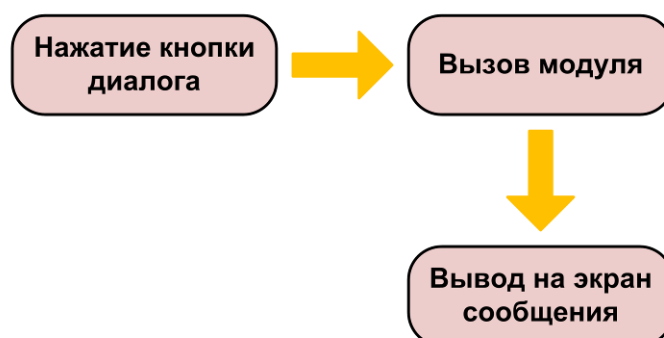


Рисунок 2.13 – Концепция работы программы

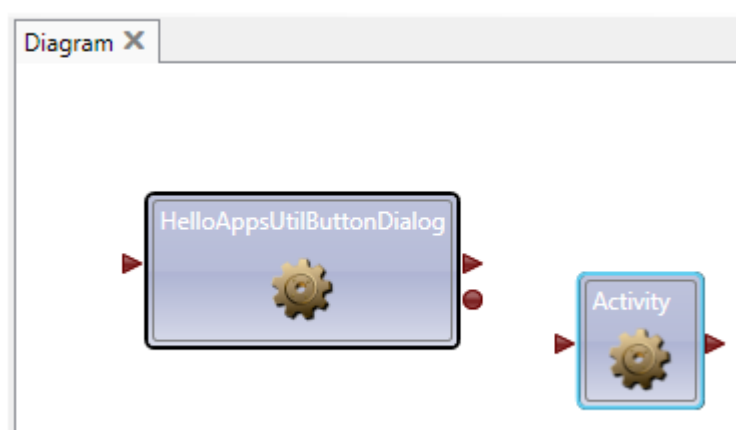


Рисунок 2.14 – Окно диаграммы

3. Добавьте в модуль блоки и сервисы, заполните блоки, соедините входящие и исходящие разъемы (рис. 2.15).
4. Соедините сервис с модулем (рис. 2.16).
5. Запустите на выполнение программу (рис. 2.17).

2.2.3. Методика добавления входящих сообщений модуля

1. Работа очередной программы будет организована следующим образом (рис. 2.18).

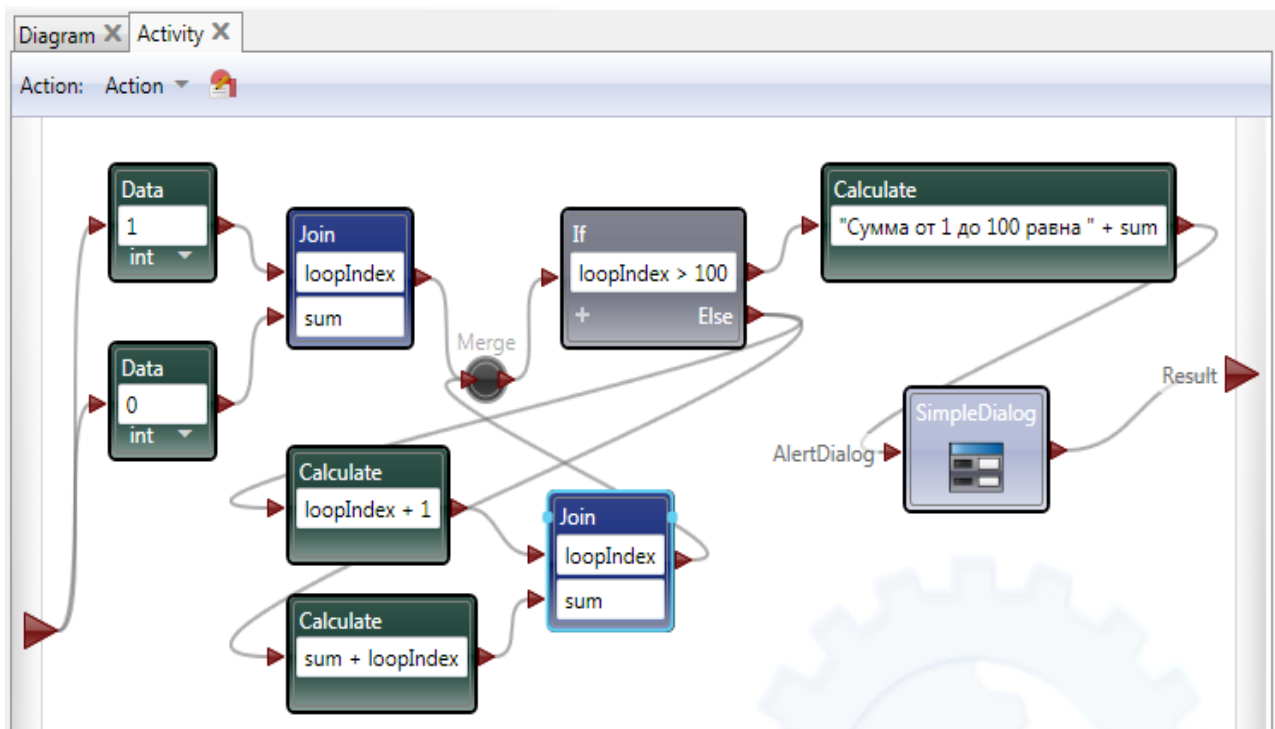


Рисунок 2.15 – Окно модуля

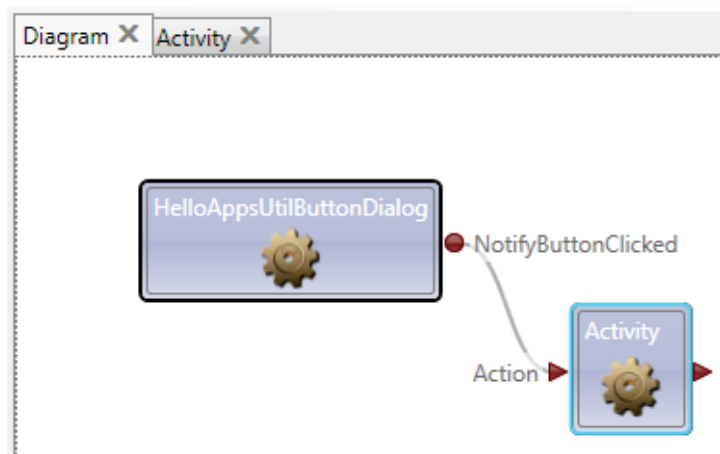


Рисунок 2.16 – Окно модифицированной диаграммы

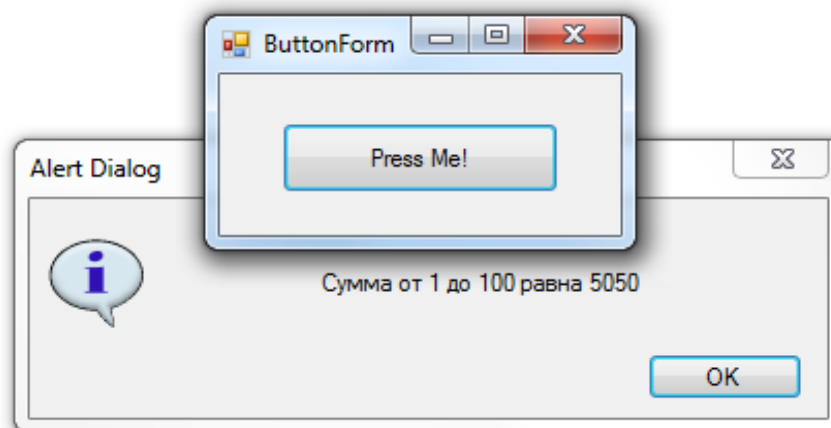


Рисунок 2.17 – Результат работы программы

2. Для организации добавления входящих сообщений в предыдущей диаграмме (рис. 2.16) удалите соединение между сервисом и модулем.

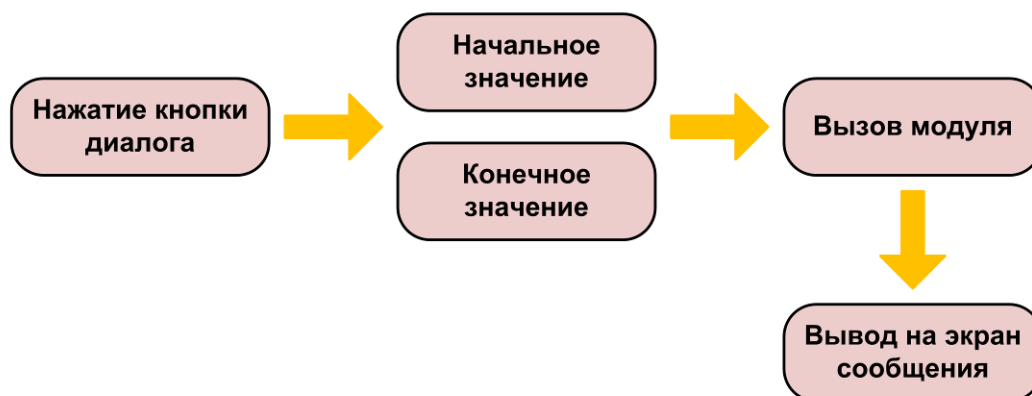


Рисунок 2.18 – Концепция работы программы

3. Задайте имя для модуля, например «MySumUp».
4. Откройте окно модуля и нажмите кнопку «Actions and Notifications».
5. В появившемся окне добавьте два входящих сообщения (рис. 2.19).

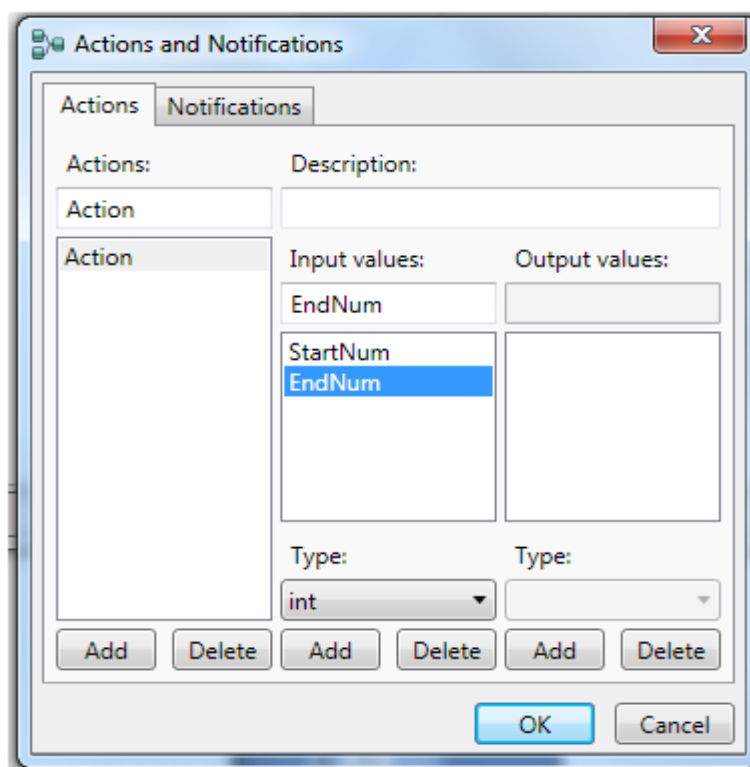


Рисунок 2.19 – Добавленные входящие сообщения

6. Удалите в модуле соединения на разъеме входящих сообщений. Измените модуль (рис. 2.20). Измените диаграмму (рис. 2.21).

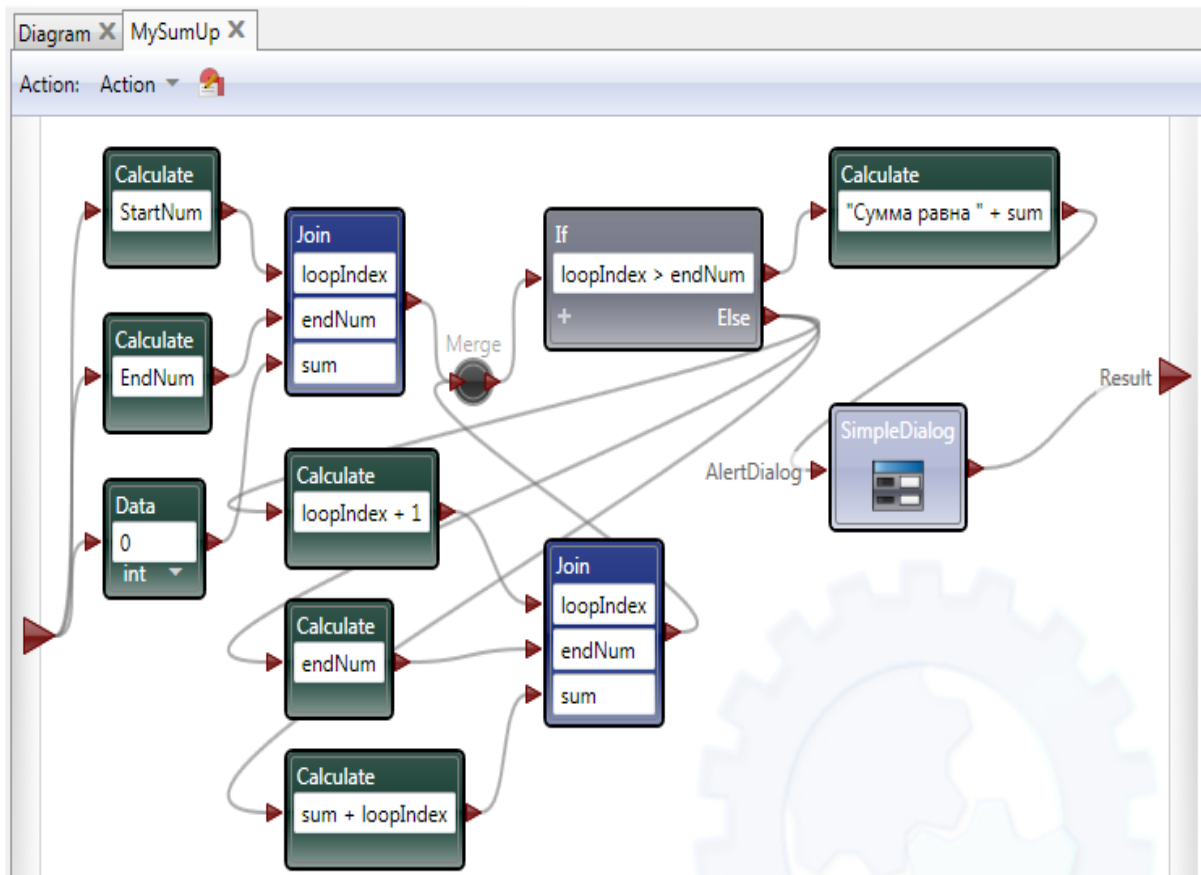


Рисунок 2.20 – Модифицированный модуль

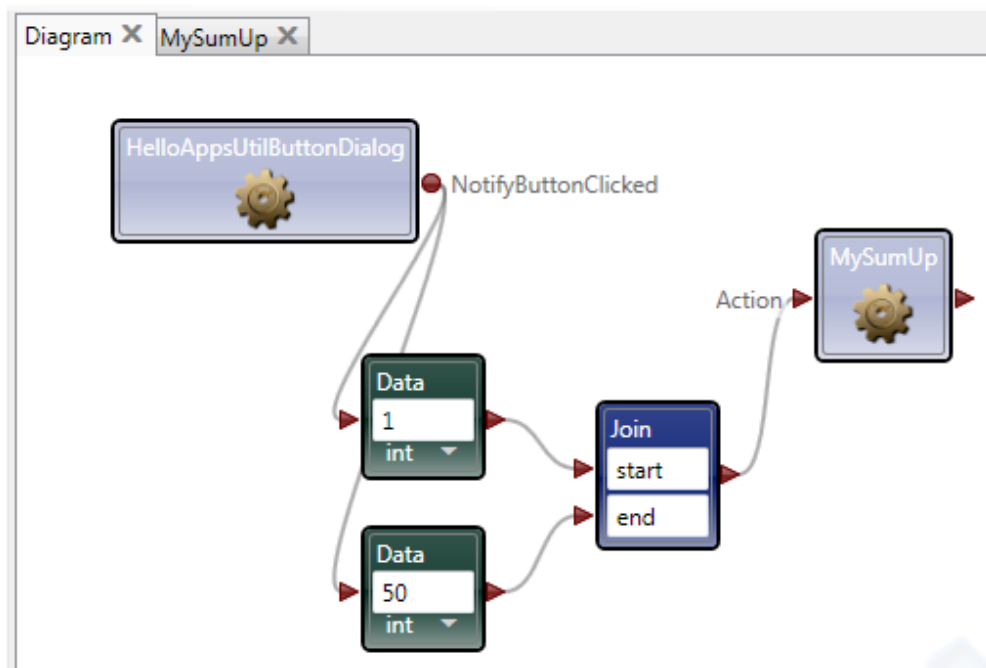


Рисунок 2.21 – Модифицированная диаграмма

7. При соединении блока слияния с модулем, в появившемся окне выберите соответствующие значения (рис. 2.22).
8. Запустите программу на выполнение (рис. 2.23).

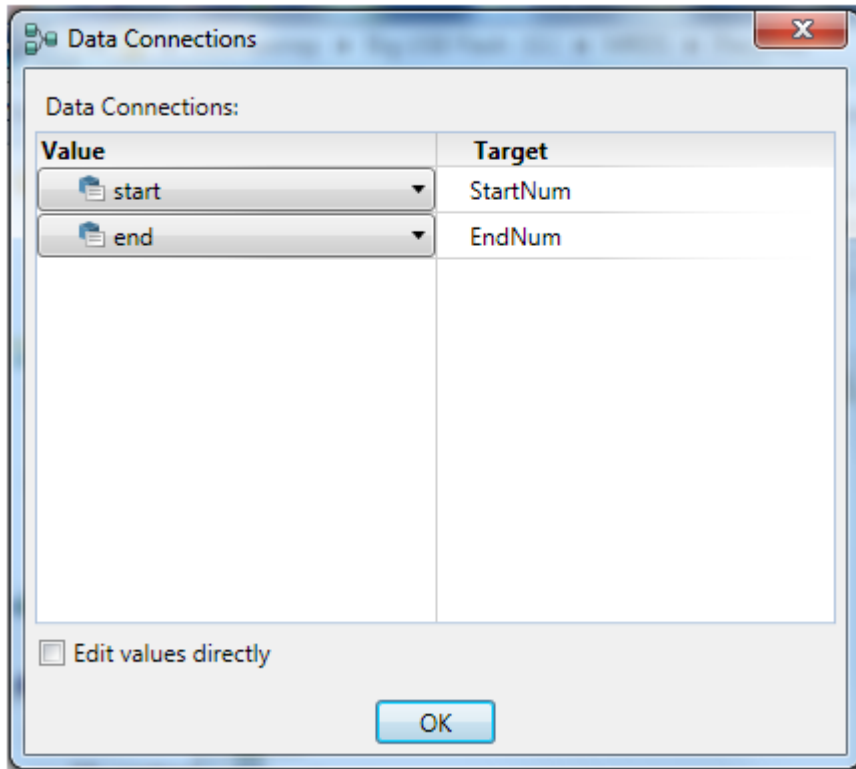


Рисунок 2.22 – Окно пересылаемых значений

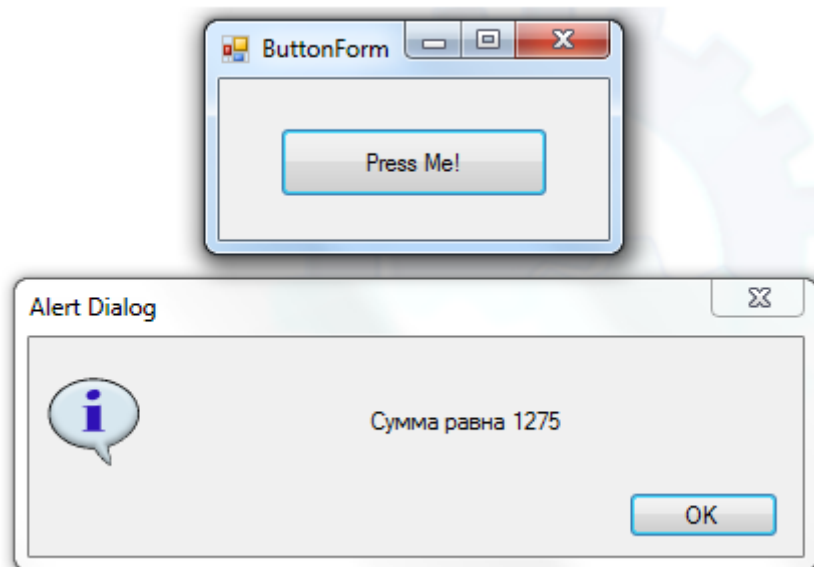


Рисунок 2.23 – Результат работы программы

2.2.4. Методика добавления исходящих сообщений модуля

1. Для организации добавления исходящих сообщений откройте окно модуля предыдущей диаграммы (рис. 2.20) и нажмите кнопку «Actions and Notifications». Добавьте исходящее сообщение (рис. 2.24).

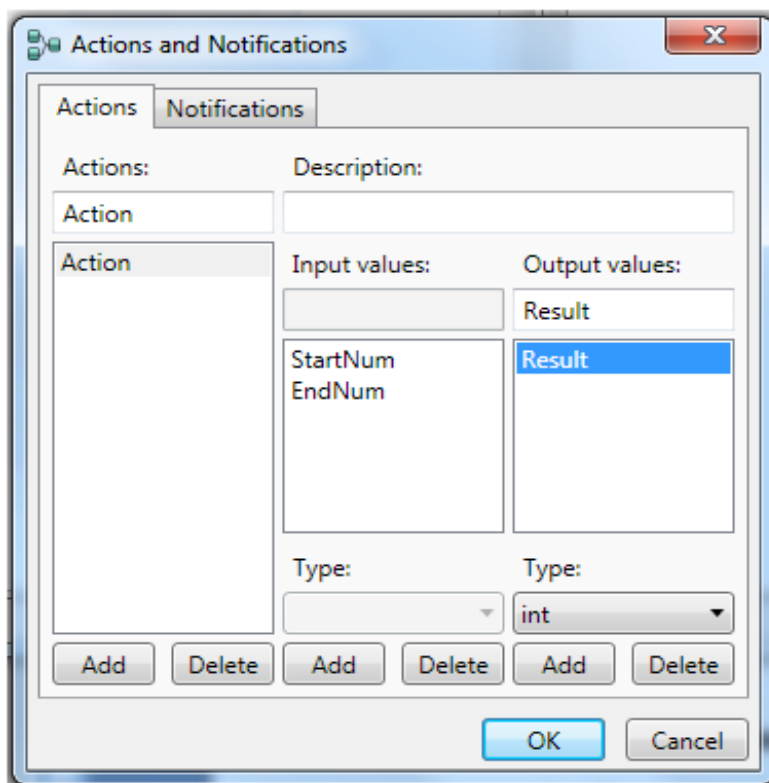


Рисунок 2.24 – Добавленное исходящее сообщение

2. В модуле удалите сервис простого диалога, модифицируйте блок вычисления и соедините его с выходным разъемом (рис. 2.25).

3. Добавьте в диаграмму блок вычисления и сервис простого диалога (рис. 2.26). Запустите программу на выполнение (рис. 2.27).

2.3. Лабораторная работа 2. Изучение иерархической архитектуры сервисов

2.3.1. Цель и содержание

Целью лабораторной работы является ознакомление студентов с принципами разделения программного кода, особенностями работы с нотификациями и модулями.

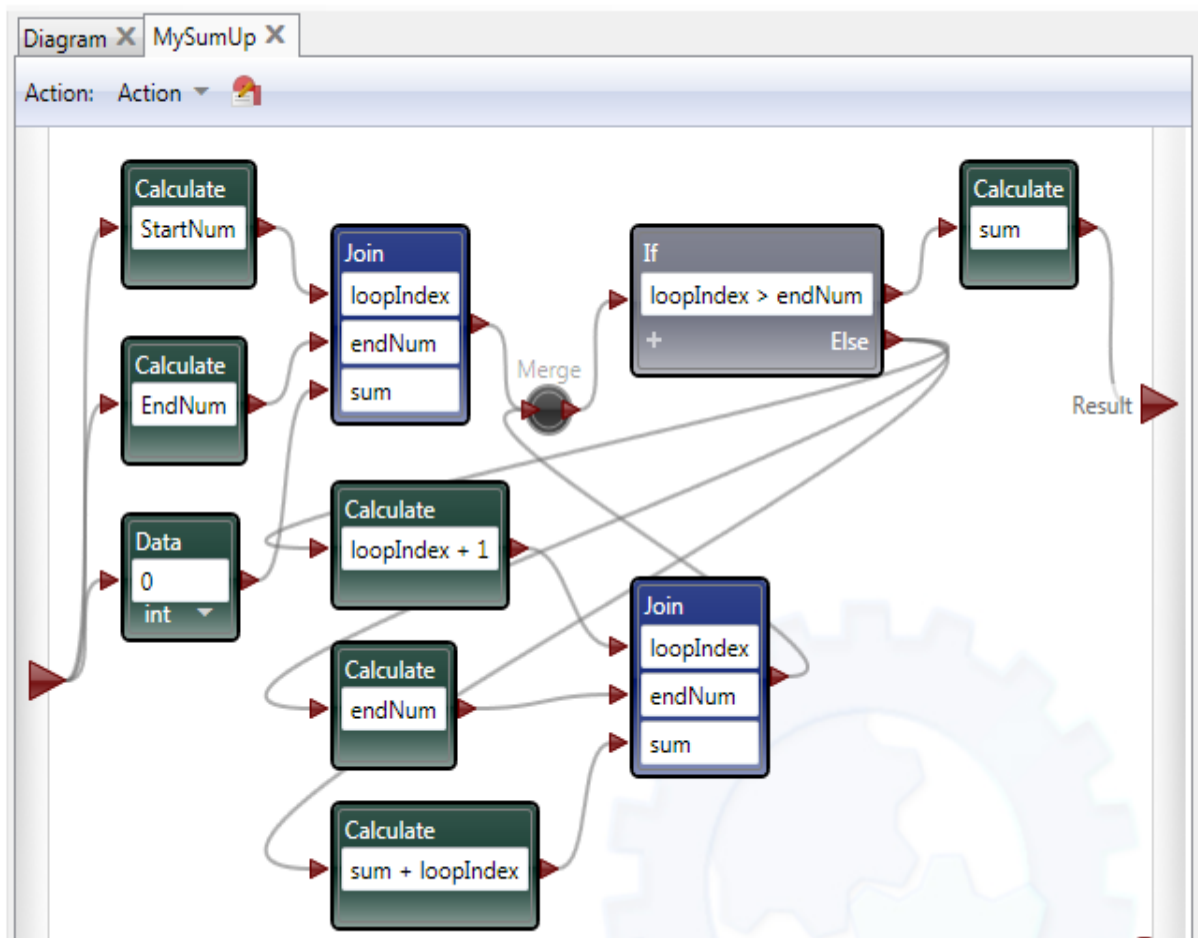


Рисунок 2.25 – Модифицированный модуль

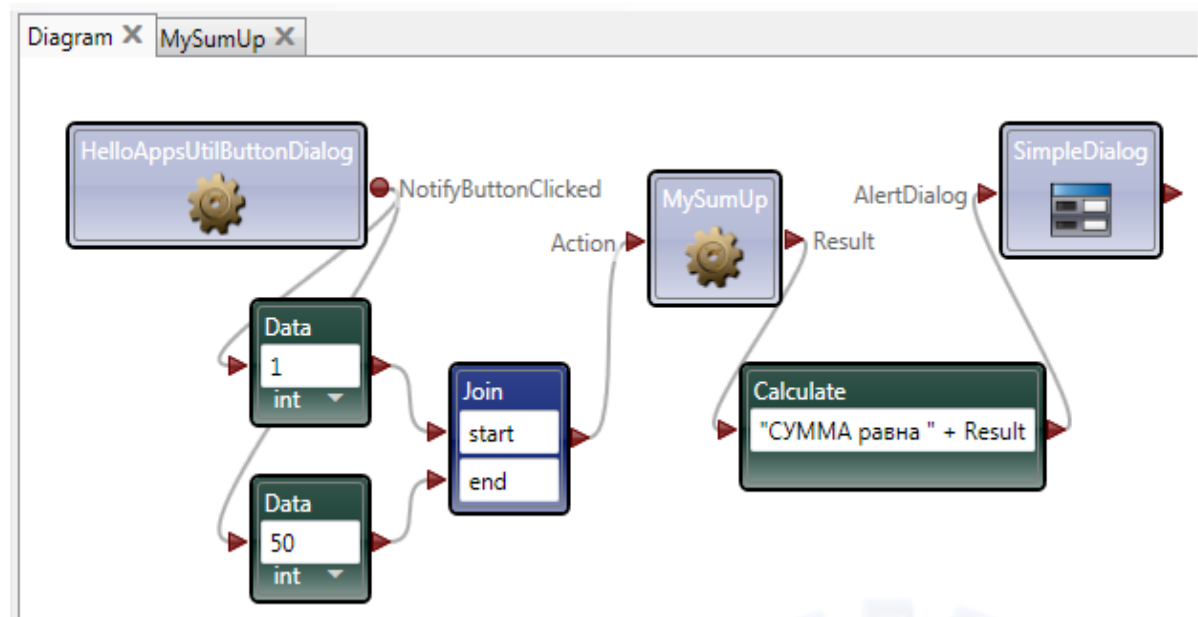


Рисунок 2.26 – Модифицированная диаграмма

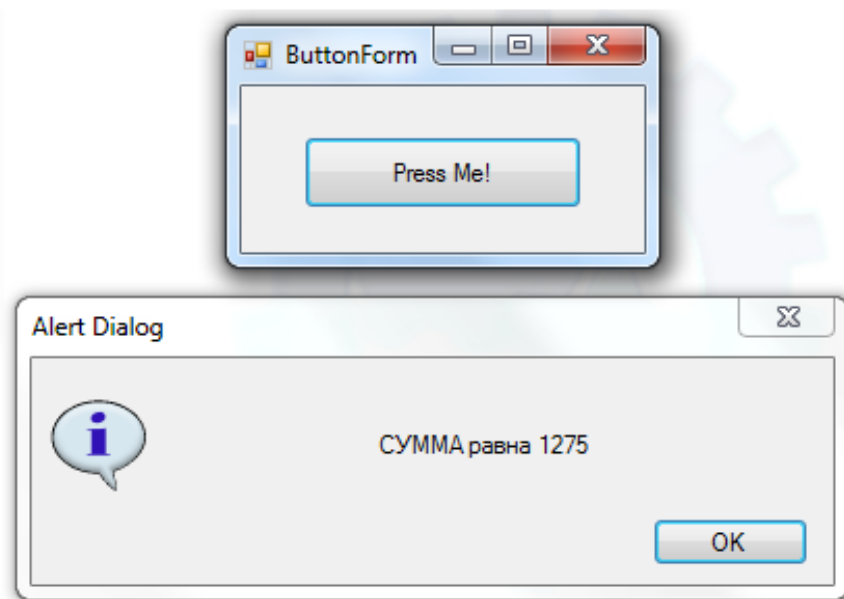


Рисунок 2.27 – Результат работы программы

Содержание

1. Организация декомпозиции программы с помощью нотификации.
2. Организация декомпозиции программы с помощью модуля.
3. Организация добавления входящих и исходящих сообщений модуля.

2.3.2. Аппаратура и материалы

Для изучения материалов этой лабораторной работы необходимо:

1. Компьютер, удовлетворяющий минимальным системным требованиям для установки Microsoft Robotics Developer Studio R4.
2. Установленная платформа RDS R4 с SPLX Launcher tool и SPL Service Pack.

2.3.3. Порядок выполнения работы

1. Изучение теоретического материала и методических рекомендаций.
2. Выполнение задания, предлагаемого в лабораторной работе.

Задание:

1. Создайте программы согласно методикам 2.2.1 – 2.2.4.
2. Выполните задание из таблицы 2.1 согласно своему варианту.

Таблица 2.1. Варианты заданий для самостоятельного выполнения

№ вар.	Задание
1.	Создайте программу с декомпозицией на основе использования нотификации, описание которой приведено далее. 1. Есть три кнопки, на каждой из них написано одно из слов: снег, солнце, бабочка. 2. При нажатии на кнопку – выдается сообщение со стихотворением на данную тему.
2.	Создайте программу с декомпозицией на основе использования нотификации, описание которой приведено далее. 1. Есть три кнопки, на каждой из них написано число. 2. При нажатии на кнопку генерируется сообщение, что выбранное число: больше, меньше или равно остальным числам.
3.	Создайте программу с декомпозицией на основе использования нотификации, описание которой приведено далее. 1. Есть четыре кнопки, каждая из кнопок означает цвет. 2. При нажатии на кнопку генерируется сообщение о том, какую позицию в радуге занимает выбранный цвет, и какие цвета стоят слева и справа от выбранного цвета.
4.	Создайте программу с декомпозицией на основе использования нотификации, описание которой приведено далее. 1. Есть шесть кнопок, каждая из кнопок обозначает закрытую коробку. 2. При нажатии на кнопку генерируется сообщение о том, что скрыто в выбранной коробке. Предусмотрите главный приз, утешительные призы и пустые коробки.
5.	Создайте программу с декомпозицией на основе использования нотификации, описание которой приведено далее. 1. На кнопке написано уравнение $((x^3)/3 - 4)^2$. 2. При нажатии на кнопку в модуле происходит вычисление уравнения и выдается результат (x – переменная, введенная с клавиатуры).
6.	Создайте программу с декомпозицией на основе использования модуля, описание которой приведено далее. 1. С помощью простого диалога задайте вопрос. На четырех кнопках помещены варианты ответа. 2. При нажатии на кнопку в модуле сравниваются выбор с правильным ответом: если ответ правильный – выдается поздравительное сообщение, иначе – правильный ответ и сообщение об ошибке.
7.	Создайте программу с декомпозицией на основе использования модуля, описание которой приведено далее. 1. Есть четыре кнопки: Север, Юг, Восток, Запад. 2. При нажатии на кнопку происходит вычисление, насколько градусов нужно повернуться, чтобы оказаться лицом в указанном направлении. Учитывать что изначальное направление строго на северо-восток, а поворачиваться можно только по часовой стрелке.
8.	Создайте программу с декомпозицией на основе использования модуля, описание которой приведено далее. 1. При нажатии на кнопку модуль рассчитывает факториал 10.

№ вар.	Задание
9.	Создайте программу с декомпозицией на основе использования модуля, описание которой приведено далее. 1. Организуйте добавление входящих и исходящих сообщений модуля. 2. С помощью блока вычислений решите уравнение $(x^3 - y^2)/4$. 3. Результат, используя входящее сообщение, проверьте на четность.
10.	Создайте программу с декомпозицией на основе использования модуля, описание которой приведено далее. 1. Организуйте добавление входящих и исходящих сообщений модуля. 2. Пусть есть две переменные x и y , введенные с клавиатуры. Данные в модуль передаются через входящее и исходящее сообщение. 3. В модуле рассчитывается сумма всех чисел с шагом x до значения суммы равной y .
11.	Создайте программу с декомпозицией на основе использования модуля, описание которой приведено далее. 1. Организуйте добавление входящих и исходящих сообщений модуля. 2. С помощью простого диалога введите свое имя. 3. При нажатии кнопки «Приветствие» в модуле составляется предложение: Здравствуйте, «Имя»! Как дела?
12.	Создайте программу с декомпозицией на основе использования модуля, описание которой приведено далее. 1. Организуйте добавление входящих и исходящих сообщений модуля. 2. С помощью простого диалога организуйте введение пароля. 3. При нажатии кнопки «Проверка» в модуле создайте реализацию проверки введенного пароля. 4. Если пароли совпадают, присвойте переменной значение 1, иначе 0. 5. На диаграмме произведите проверку – если 1 – то выводится сообщение о правильном вводе пароля, иначе – об ошибке.

2.4. Контрольные вопросы

1. Дайте определение модуля.
2. Перечислите факторы способствующие созданию модульных программ.
3. Дайте определение иерархии.
4. Назовите свойства присущие архитектуре программного обеспечения как иерархической системе.
5. Перечислите шаги позволяющие организовать декомпозицию программы с помощью нотификации.

6. Охарактеризуйте понятие и элементы модуля в VPL.

7. Перечислите шаги позволяющие организовать декомпозицию программы с помощью модуля.

ГЛАВА 3. ОСОБЕННОСТИ ОРГАНИЗАЦИИ МНОГОФУНКЦИОНАЛЬНЫХ ПРОГРАММНЫХ МОДУЛЕЙ

В большей степени программные продукты не являются монолитом и имеют конструкцию (архитектуру) построения – состав и взаимосвязь программных модулей.

Модуль – это самостоятельная часть программы, имеющая определенное назначение и обеспечивающая заданные функции обработки автономно от других программных модулей [14]. Таким образом, программный продукт обладает внутренней организацией, или же внутренней структурой, образованной взаимосвязанными программными модулями.

Структуризация программ выполняется в первую очередь для удобства разработки, программирования, отладки и внесения изменений в программный продукт. Как правило, программные комплексы большой алгоритмической сложности разрабатываются коллективом разработчиков. Таким образом, структуризация программных продуктов преследует основные цели:

- распределить работы по исполнителям, обеспечив приемлемую их загрузку и требуемые сроки разработки программных продуктов;
- построить календарные графики проектных работ и осуществлять их координацию в процессе создания программных изделий;
- контролировать трудозатраты и стоимость проектных работ и др.

Структурное разбиение программ на отдельные составляющие служит основой и для выбора инструментальных средств их создания, хотя имеет место и обратное влияние – выбор инструментальных средств разработчика программного обеспечения определяет типы программных модулей.

При создании программных продуктов выделяются многократно используемые модули, проводится их типизация и унификация, за счет чего сокращаются сроки и трудозатраты на разработку программного продукта в целом.

Некоторые программные продукты используют модули из готовых библиотек стандартных подпрограмм, процедур, функций, объектов, методов обработки данных.

3.1. Краткие теоретические сведения

В среде Microsoft Robotics Developer Studio R4 при написании программы в модуле можно определить множество операций, которые будут по-разному обрабатывать входящие сообщения. Это делает модуль универсальным элементом программы. Для каждого входящего сообщения можно поставить в соответствие определенную операцию, либо конкретную операцию связать с каким-либо условием (рис. 3.1).

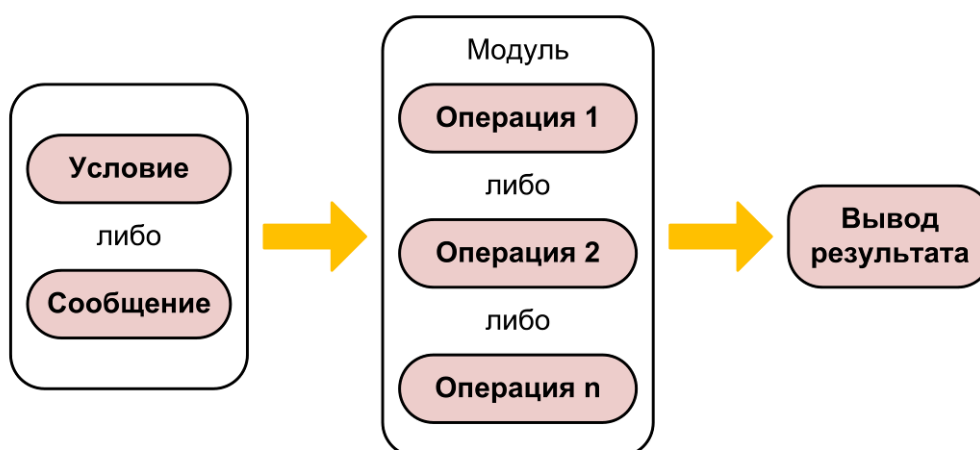


Рисунок 3.1 – Схема работы модуля

В модуле, как и в сервисах, можно организовать рассылку уведомлений (рис. 3.2). Данные сообщения создаются в модуле, с помощью блоков и сервисов, они могут относиться к различным типам данных. Уведомления должны быть предварительно определены в модуле (рис. 3.3).



Рисунок 3.2 – Модуль с разъемом уведомлений

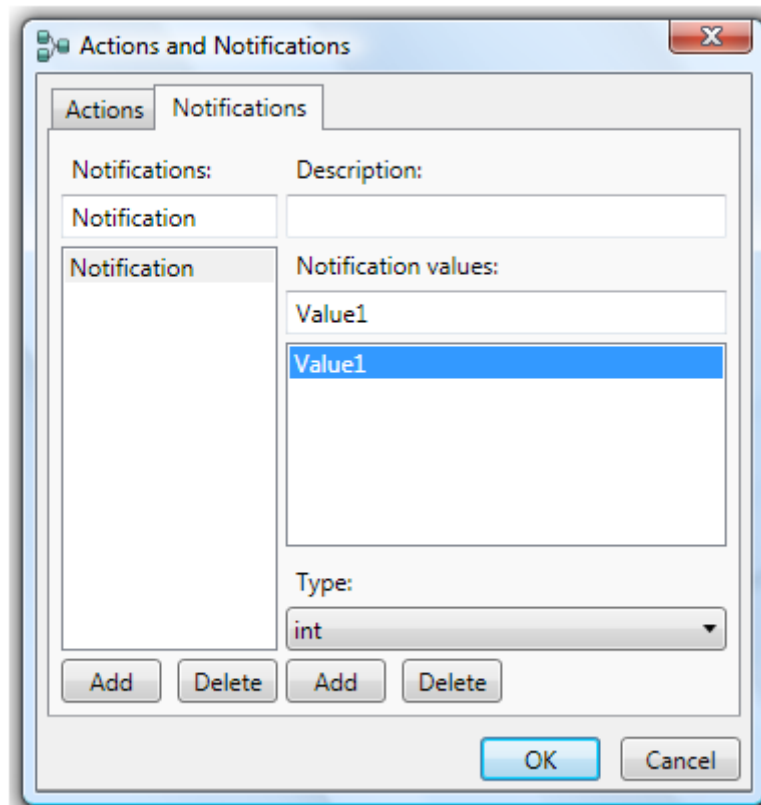


Рисунок 3.3 – Вкладка нотификаций

3.2. Методические рекомендации

3.2.1. Методика реализации в модуле многофункциональной обработки сообщений

1. Для реализации в модуле многофункциональной обработки сообщений создайте новую вкладку диаграммы – Diagram2. Добавьте блоки и сервисы, соедините их и заполните (рис. 3.4). Запустите программу на выполнение (рис. 3.5).
2. Добавьте в окне первой диаграммы сервис и модуль (рис. 3.6).
3. Откройте окно модуля, затем вызовите окно «Actions and Notifications» и определите 4 операции (рис. 3.7). Для каждой операции определите входящие и исходящие значения.
4. Задайте имя модуля «MyCalculator». Добавьте блоки для каждой операции модуля (рис. 3.8).

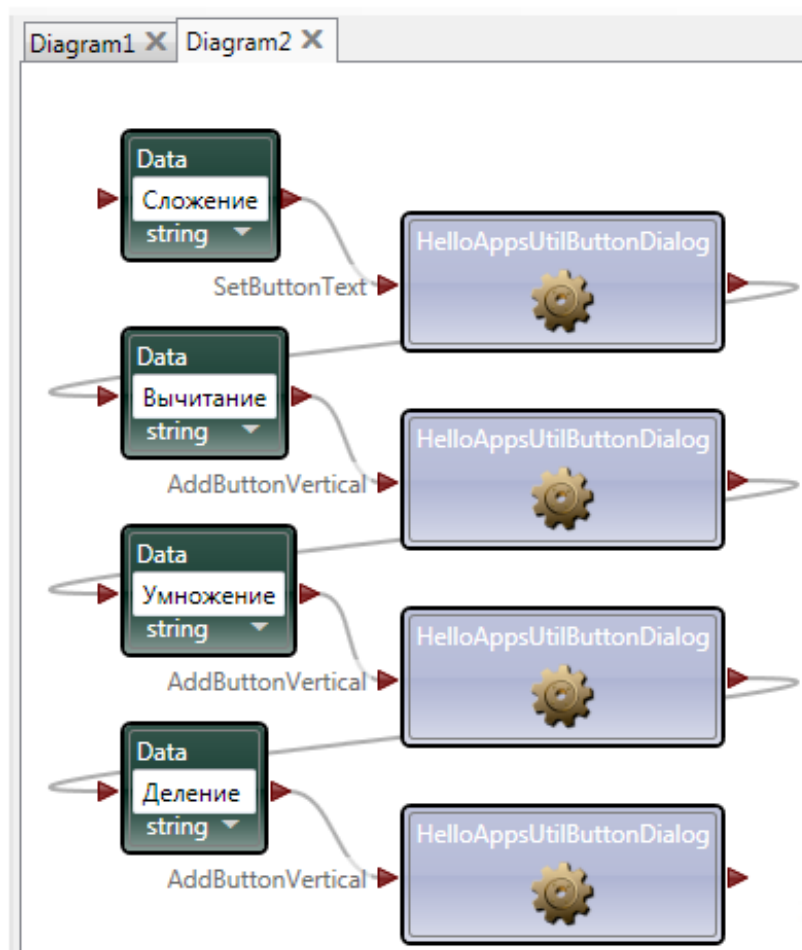


Рисунок 3.4 – Готовая вторая диаграмма

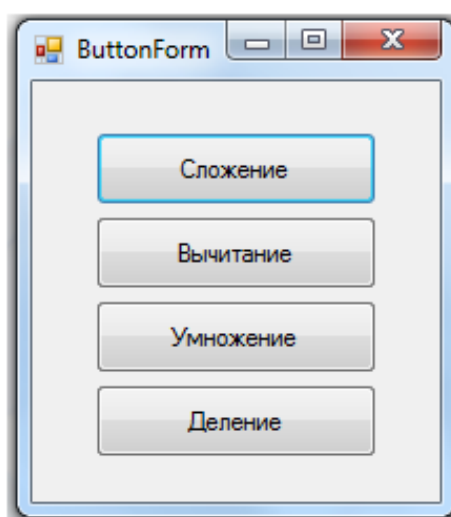


Рисунок 3.5 – Результат работы программы

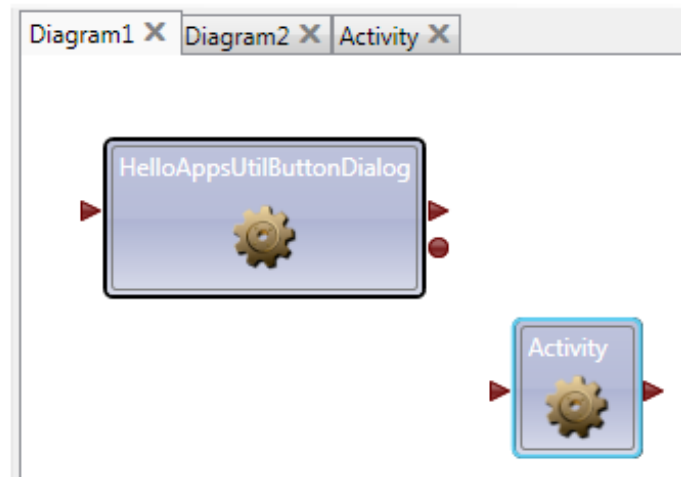


Рисунок 3.6 – Окно первой диаграммы

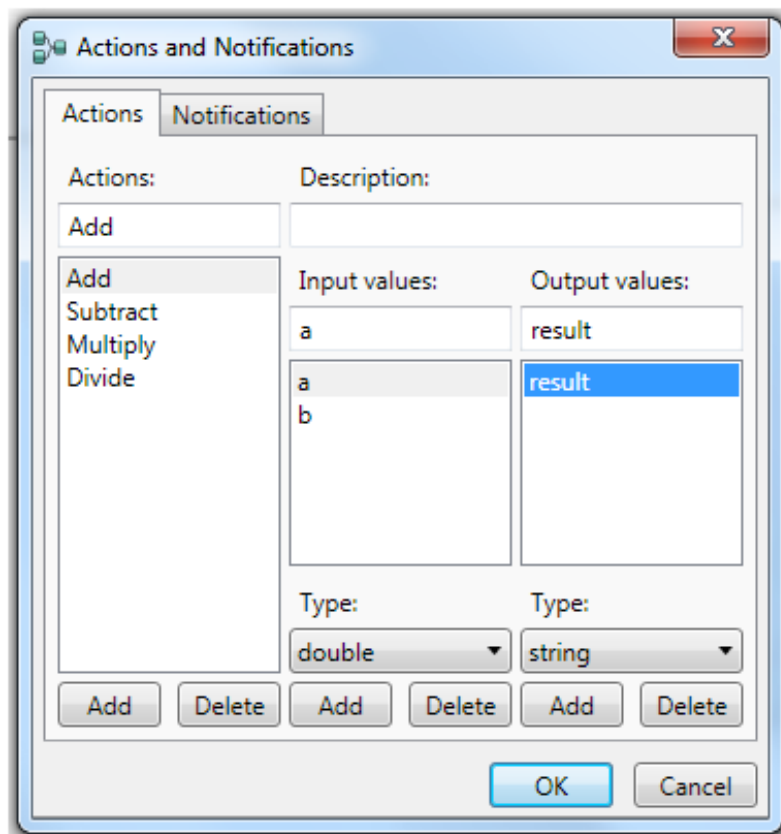


Рисунок 3.7 – Окно «Actions and Notifications» модуля

5. В диаграмму «Diagram1» добавьте блоки и сервисы, соедините их согласно рис. 3.9. При соединении блока Merge с сервисом простого диалога в появившемся окне в списке значений (Value) выберите result (рис. 3.10).

6. Запустите программу на выполнение (рис. 3.11).

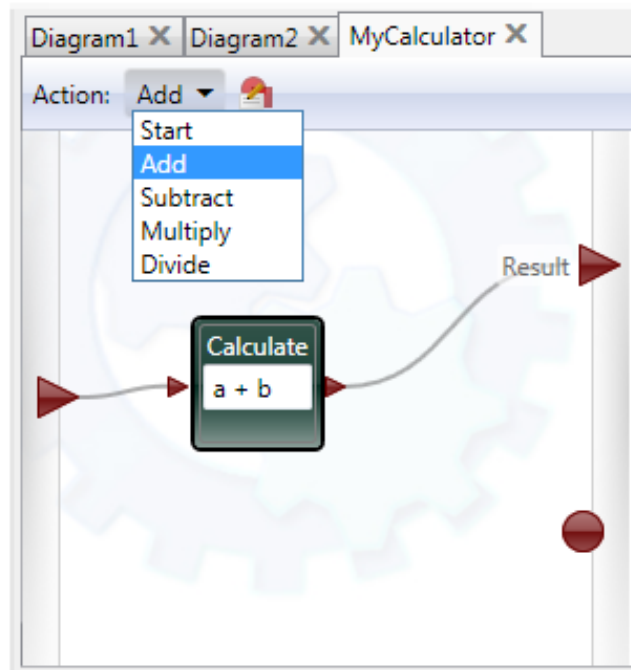


Рисунок 3.8 – Окно модуля с блоком операции «Add»

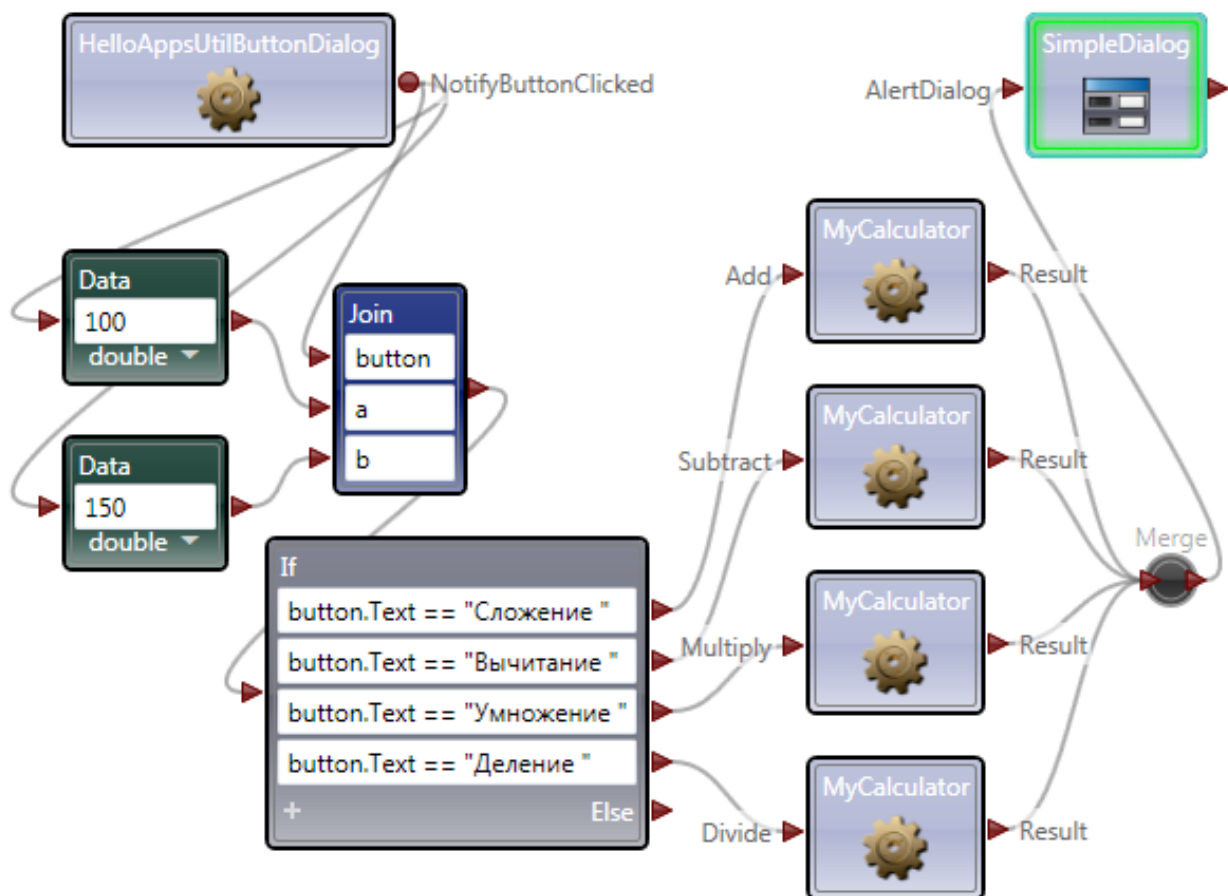


Рисунок 3.9 – Окно первой диаграммы

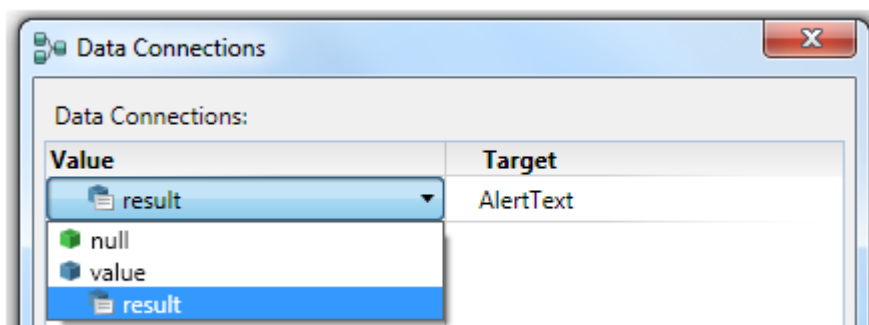


Рисунок 3.10 – Окно Data Connections

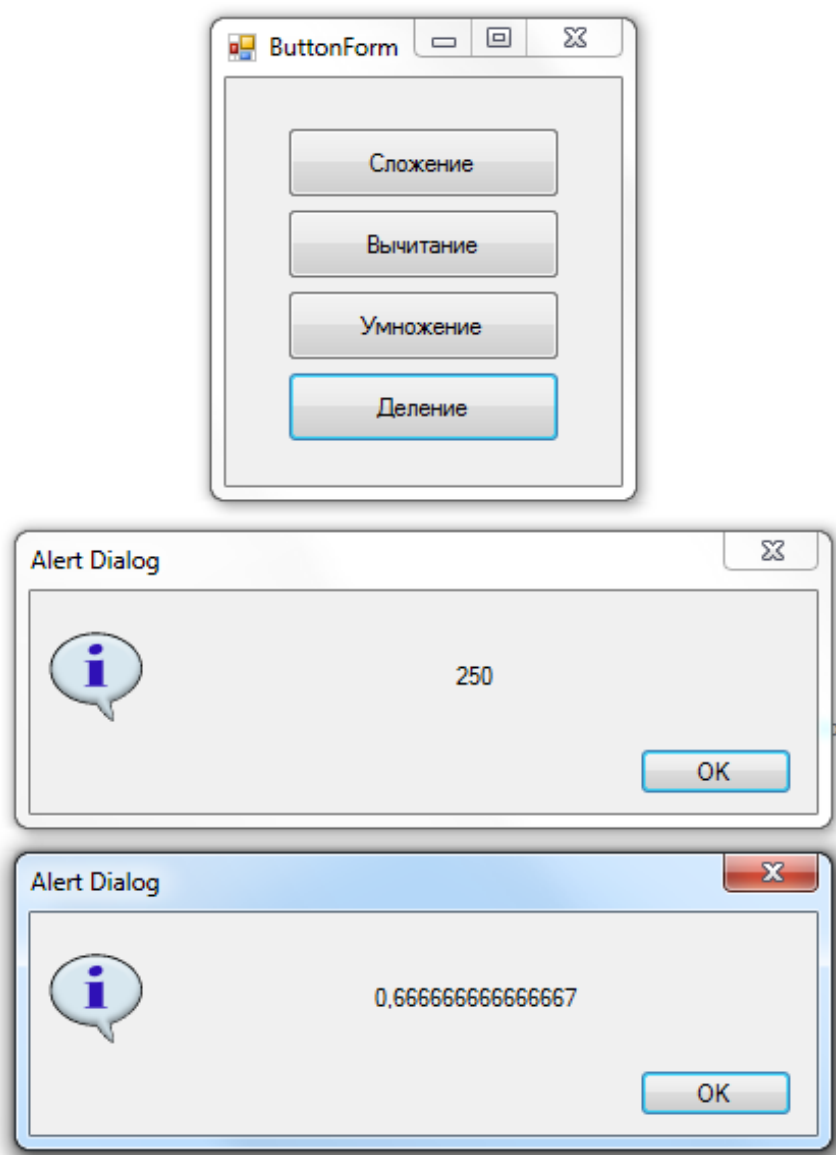


Рисунок 3.11 – Результат работы программы

3.2.2. Методика реализации в модуле нотификаций

1. Для реализации в модуле нотификаций напомним программу вычисления i -го элемента ряда Фибоначчи. Логика работы данной программы отражена на рис. 3.12.

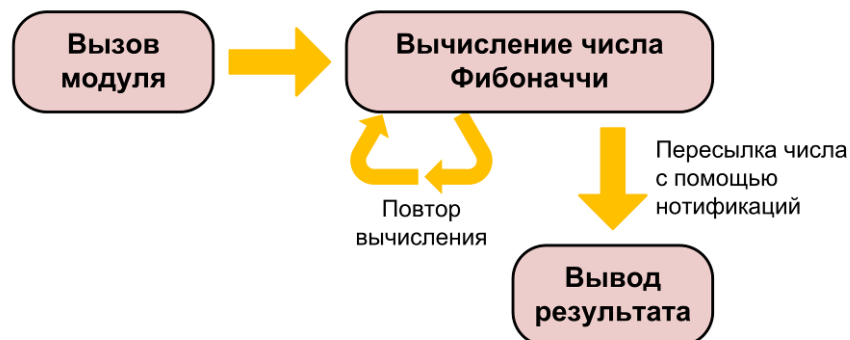


Рисунок 3.12 – Схема работы модуля

2. Добавьте в диаграмму сервис HelloAppsUtilButtonDialog и модуль. Дайте имя модулю «Fibonacci».

3. Откройте окно модуля и добавьте блоки данных и переменных, определите переменные n1 и n2 (рис. 3.13).

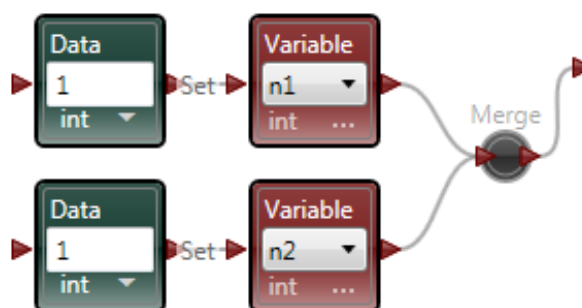


Рисунок 3.13 – Окно модуля

4. Дополните модуль необходимыми блоками, соедините их и заполните (рис. 3.14).

5. Определите в модуле нотификацию (рис. 3.15).

6. При соединении блока вычисления с разъемом нотификации выберите в первом окне Notification, а в следующем value.

7. Откройте вкладку Diagram, скопируйте модуль и повторно вставьте в диаграмму.

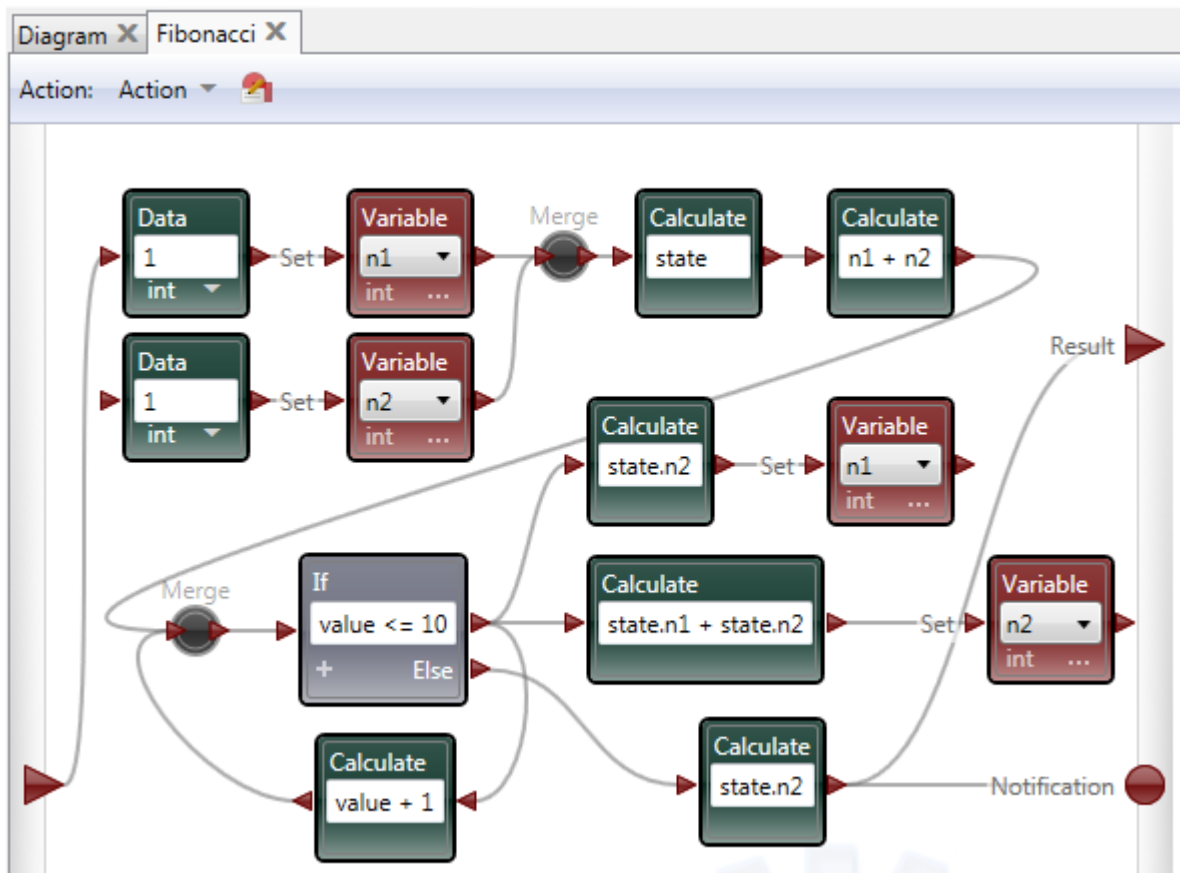


Рисунок 3.14 – Окно модифицированного модуля

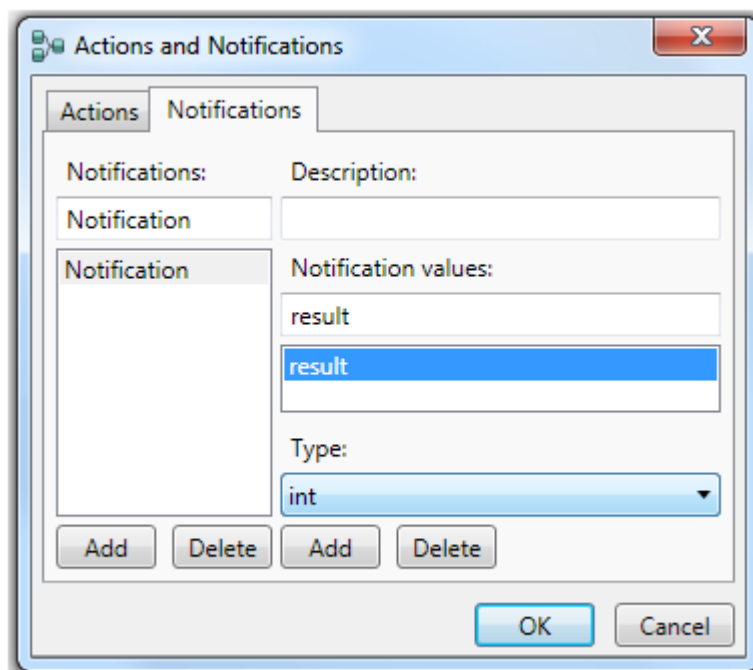


Рисунок 3.15 – Вкладка нотификаций

8. Соедините разъем нотификации сервиса с модулем, в появившемся окне выберите `NotifyButtonClicked`.

9. Добавьте в диаграмму сервис HelloAppsUtilListBoxDialog, соедините разъем нотификации модуля с добавленным сервисом, в появившихся окнах выберите поочередно AddItem и result. В законченном виде диаграмма должна выглядеть следующим образом (рис. 3.16).

10. Запустите программу на выполнение (рис. 3.17).

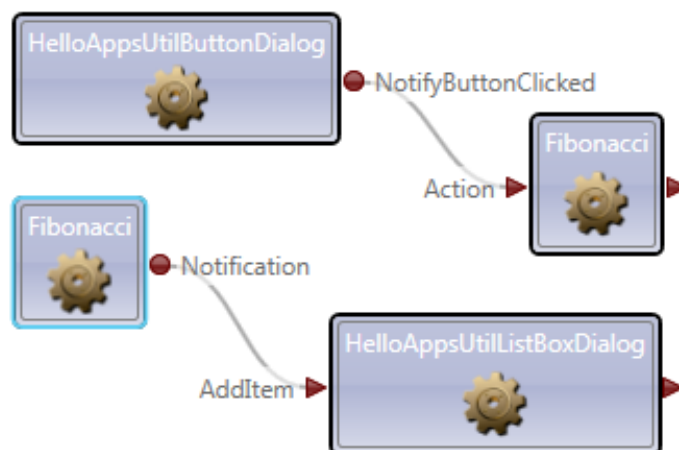


Рисунок 3.16 – Окно модифицированной диаграммы

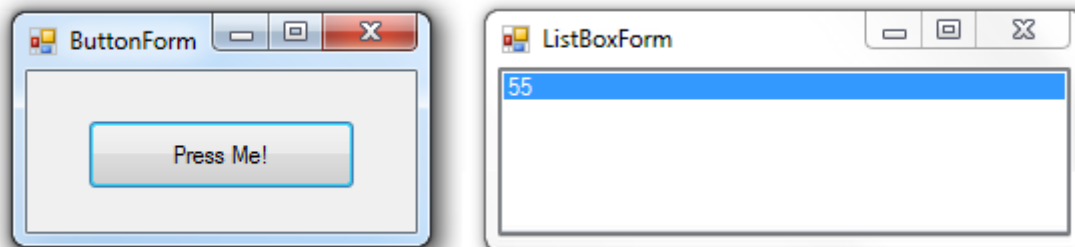


Рисунок 3.17 – Результат работы программы для $i = 10$

3.3. Лабораторная работа 3. Изучение модульной организации программ

3.3.1. Цель и содержание

Целью лабораторной работы является приобретение студентами практических навыков реализации в модуле многофункциональной обработки сообщений, а также использования в модуле нотификаций.

Содержание

1. Организация в модуле многофункциональной обработки сообщений.

2. Организация в модуле нотификаций.

3.3.2. Аппаратура и материалы

Для изучения материалов этой лабораторной работы необходимо:

1. Компьютер, удовлетворяющий минимальным системным требованиям для установки Microsoft Robotics Developer Studio R4.
2. Установленная платформа RDS R4 с SPLX Launcher tool и SPL Service Pack.

3.3.3. Порядок выполнения работы

1. Изучение теоретического материала и методических рекомендаций.
2. Выполнение задания, предлагаемого в лабораторной работе.

Задание:

1. Создайте программы согласно методикам 3.2.1 – 3.2.2.
2. Выполните задание из таблицы 3.1 согласно своему варианту.

Таблица 3.1. Варианты заданий для самостоятельного выполнения

№ вар.	Задание
1.	Используя как пример методику 3.2.1, создайте калькулятор для трех чисел.
2.	Используя как пример методику 3.2.1, создайте программу для вычисления: факториала, корня, квадрата и куба числа.
3.	Используя как пример методику 3.2.1, создайте программу для вычисления по формулам: а) $z = x^2 - 4$; б) $z = (x - y) \times 4$; в) $z = 6x - 9y$.
4	Используя как пример методику 3.2.1, найдите стороны прямоугольника, вписанного в окружность радиуса R и имеющего наибольшую площадь S , если величина R равна: 10, 15, 20, 25.
5	Используя как пример методику 3.2.1, определите, является ли введенное число: двузначным, четным, простым.
6	Используя как пример методику 3.2.1, определите, является ли данное число: отрицательным, кратное 3, трехзначным.
7	Используя как пример методику 3.2.2, вычислите сумму первых 20 членов последовательности, если их общий член последовательности равен $a_k = \frac{k+2}{k^2-2}$, где $k=1, 2, \dots, 20$.
8	Используя как пример методику 3.2.2, вычислите значение числа n с точностью до 0,001, используя разложение функции экспоненты в ряд.

№ вар.	Задание
9	Используя как пример методику 3.2.2, организуйте выведение на экран результата вычисления возведения в степень 3 чисел от x до y .
10	Используя как пример методику 3.2.2, вычислите факториал числа n . При этом число n вводится с клавиатуры.
11	Используя как пример методику 3.2.2, определите количество чисел из списка от 1 до n , которые являются простыми числами. При этом число n вводится с клавиатуры.
12	Используя как пример методику 3.2.2, определите сумму чисел из списка от 1 до n . При этом число n вводится с клавиатуры.

3.4. Контрольные вопросы

1. Назовите цели структуризации программных продуктов на этапе разработки.
2. Прокомментируйте особенности многофункциональной обработки сообщений в модуле.
3. Перечислите преимущества использования нотификаций.
4. Прокомментируйте содержание вкладок окна «Actions and Notifications» модуля.
5. Охарактеризуйте этапы организации выполнения различных операций в модуле.

ГЛАВА 4. АВТОМАТИЗИРОВАННОЕ УПРАВЛЕНИЕ РОБОТОМ

Автоматизированная система управления (АСУ) – комплекс аппаратных и программных средств, предназначенный для управления различными процессами в рамках технологического процесса, производства, предприятия [7]. АСУ применяются в различных отраслях промышленности, энергетике, транспорте и т. п. Термин «автоматизированная», в отличие от термина «автоматическая» подчёркивает сохранение за человеком-оператором некоторых функций, либо наиболее общего, целеполагающего характера, либо не поддающихся автоматизации.

Важнейшая задача АСУ – повышение эффективности управления объектом на основе роста производительности труда и совершенствования методов планирования процесса управления. Различают автоматизированные системы управления объектами (технологическими процессами – АСУТП, предприятием – АСУП, отраслью – ОАСУ) и функциональные автоматизированные системы, например, проектирование плановых расчётов, материально-технического снабжения и т. д.

Сегодня для технологичного высокоэффективного производства применение робототехники является обязательным условием [26]. Без роботов невозможно получить высокую производительность и обеспечить нужное качество готового изделия. Крайне сложно говорить об эффективности производства, если на нем нет интеграции робототехнических систем. На любых участках техпроцесса роботы позволяют заменить монотонный физический труд, повысить качество изделий, увеличить их выпуск.

Современные роботы оснащены большим количеством опций, позволяющих реализовывать самые невероятные решения. Роботы обладают высокой универсальностью базирования: их можно ставить на пол, крепить к стене, подвешивать на консоль или устанавливать на направляющих для линейного перемещения на участке. Робот может быть достаточно легким и

компактным для размещения его непосредственно на технологическом оборудовании: на станке, на транспортере и т. д.

Применение робототехники – универсальный путь автоматизации сварочной технологии не только в серийном, но и мелкосерийном производстве, так как для выполнения различных операций можно использовать того же робота, применяя для его управления следующую программу [26]. Роботизированная контактная сварка, представляет собой хороший пример экономически эффективного использования роботов, к тому же освобождающая человека от однообразной и тяжелой работы. Применение промышленных роботов для точечной сварки, было и есть одним из самых распространенных приложений робототехники.

Применительно к робототехнике термин «паллетирование» обобщает широкую гамму технологических операций промышленного производства. Для машиностроения это, в частности, позиционирование заготовок: подача заготовки в станок, съём обработанной детали, укладка на транспортер или в тару и подобные манипуляции. Учитывая высокую точность движений, выполняемых роботом, в среднем около 0,15 мм и высокую грузоподъемность – до 1,0 тонны у модели KUKA KR 1000 Titan которая способна на вытянутой руке позиционировать кузов легкового автомобиля или подавать заготовки в пресса можно представить насколько промышленный робот не заменим в современном производстве. Так же паллетирование при помощи робота может быть целесообразно при складировании, упаковке, сортировке.

Сегодня промышленные роботы уже не предъявляют специфических требований к технологии изготовления изделия: таких как высокая точность всех заготовок узла, стабильность положения сварного соединения в пространстве и высокое качество сварочных материалов. Разработаны и широко применяются различные программные и аппаратные методы контроля всех параметров техпроцесса. Применение, в том числе обслуживание и переналадка робота стали сегодня значительно проще и доступнее для любого вида производства.

4.1. Краткие теоретические сведения

При работе с реальными движущимися устройствами (чем, по сути, и является робот) мы сталкиваемся с необходимостью как можно скорее реагировать на внешние события [16]. Данное требование можно выполнить путем разработки эффективного программного обеспечения мобильным роботом.

Система командного управления, рассмотренная в данной главе, реализуется путем участия человека-оператора в управлении отдельными исполнительными устройствами (приводами) и осуществляется в релейном режиме.

В случае командного управления мобильным роботом используются **дискретные команды** «вперед-назад», «влево-вправо» с дискретным заданием скорости движения и команда «стоп». Управление с помощью задающей рукоятки – сегодня основной способ дистанционного управления (по кабелю или радиоканалу) мобильными роботами.

В среде Microsoft Robotics Developer Studio R4 командное управление может быть реализовано с помощью сервиса Desktop Joystick.

Desktop Joystick

Окно сервиса Desktop Joystick содержит элементы управления, которые **эмулируют возможности игрового контроллера или джойстика** (рис. 4.1). Перемещение трекбола осуществляется с помощью мыши. Значения координат X и Y показывают текущее положение трекбола. Кроме этого на панели данного сервиса помещена цифровая программируемая клавиатура.

Для организации перемещения трекбола наведите на него указатель мыши и нажмите левую кнопку, при перемещении мыши соответственно будет изменяться положение перекрестия на трекболе. Когда вы отпустите кнопку мыши, трекбол вернется в центральное положение. Обратите внимание, что диапазон значений X и Y изменяется от -1000 до +1000.

В дополнение к трекболу, вы можете использовать следующие клавиши на клавиатуре: **A** – влево, **D** – вправо, **W** – вперед, **S** – назад. При удерживании

одной из этих клавиш, трекбол будет двигаться в соответствующем направлении, отпускание удерживаемой клавиши возвращает трекбол в центральное положение. Нажатие на цифровые клавиши в правой части окна сервиса, или нажатие на соответствующие клавиши на клавиатуре может быть запрограммировано.

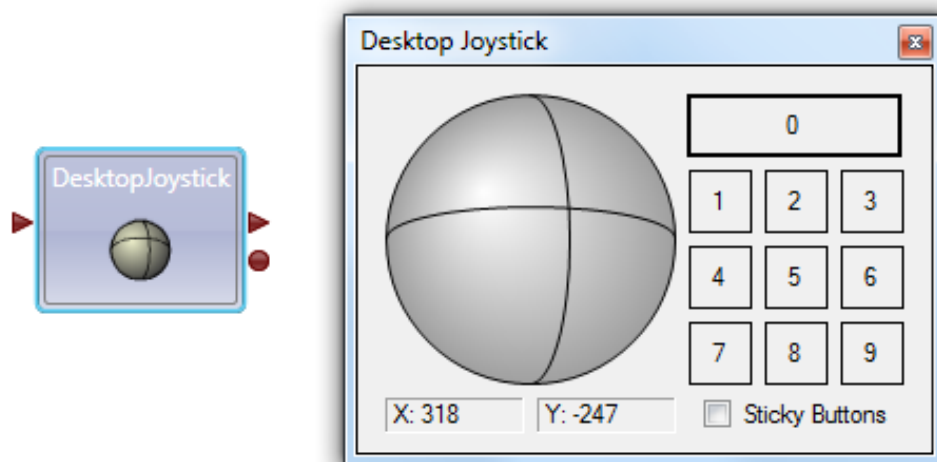


Рисунок 4.1 – Окно сервиса Desktop Joystick

Game Controller

При использовании задающей рукоятки необходимо добавить в диаграмму сервис Game Controller (рис. 4.2).



Рисунок 4.2 – Сервис Game Controller

Generic Differential Drive

Для управления дифференциальным приводом робота в VPL используется сервис Generic Differential Drive (общий дифференциальный привод) (рис. 4.3). Дифференциальный привод в общем виде состоит из двух независимо управляемых приводов. Для взаимодействия с дифференциальным приводом предусмотрены следующие операции: All Stop, Enable Drive, Drive Distance, Get, Set Drive Power, Set Drive Speed, RotateDegrees.



Рисунок 4.3 – Сервис Generic Differential Drive

В данной работе будет использована операция **Set Drive Power**, которая устанавливает мощность каждого привода в отдельности в диапазоне $[-1; 1]$ (тип данных **double**), при отрицательных значениях мощности привода робот будет двигаться назад, если значения приводов различны, то робот будет двигаться по кривой. Разъем Notification служит для отправки сообщений о состоянии сервиса привода.

Сервис Generic Differential Drive поддерживает основные операции для большинства устройств, но не связан по умолчанию не с одним из них. Главное преимущество платформы RDS состоит в том, что она позволяет писать программы для различных роботов. Для того чтобы связать базовое устройство с конкретной системой управления роботом необходимо дополнительно загрузить манифест (например, манифест для имитации робота iRobot Create – `IRobot.Create.Simulation.Manifest.xml`).



Рисунок 4.4 – Программируемый робот iRobot Create

Автоматизированное управление мобильным роботом может осуществляться, кроме джойстика, так же **фиксированным набором команд**. Такие команды задают перемещение в одной плоскости.

Реализуется данное управление либо аппаратно с помощью, например геймпада или с помощью узкоспециализированных пультов управления, либо программно с помощью соответствующего интерфейса оператора. Microsoft Robotics Developer Studio работает непосредственно на платформе или контролирует её из ОС Windows или Windows Embedded по беспроводному каналу связи (Wi-Fi или Bluetooth).

DirectionDialog

Рассмотрим сервис интерфейса оператора DirectionDialog (рис. 4.5). При запуске программы мы увидим диалоговое окно с кнопками, отвечающими за направление передвижения, и кнопку «Стоп». Данные кнопки можно запрограммировать как на элементарные действия (вперед, назад, поворот, так и на более сложные). Движения, выполняемые по данным командам, могут быть как **непрерывными**, так и **детерминированными** (с заданным расстоянием, углом поворота).

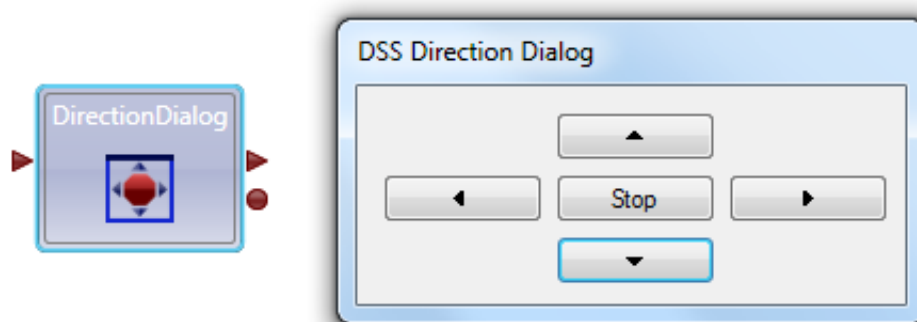


Рисунок 4.5 – Диалоговое окно сервиса DirectionDialog

XInputController

Возможно добавление сервиса геймпада рис. 4.6. Данный сервис реализует все возможности данного устройства (взаимодействие со всеми кнопками, тактильная отдача).

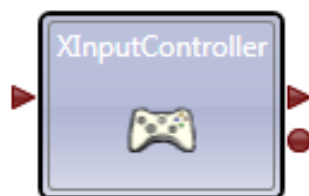


Рисунок 4.6 – Сервис XInputController

4.2. Методические рекомендации

4.2.1. Методика организации автоматизированного управления с помощью виртуального задающего манипулятора

1. Для организации автоматизированного управления добавьте на диаграмме сервисы Desktop Joystick и Generic Differential Drive.

2. Соедините разъем Notification сервиса Desktop Joystick с сервисом Generic Differential Drive. В появившемся окне Connection сделайте соответствующий выбор: UpdateAxes и SetDrivePower.

3. В следующем окне Data Connection поставьте галочку напротив надписи Edit values directly. Отредактируйте значения мощности левого и правого колеса, которые будут передаваться в привод, данные значения получаются при преобразовании координат трекбола при помощи формул (рис. 4.7). Готовая диаграмма изображена на рис. 4.8.

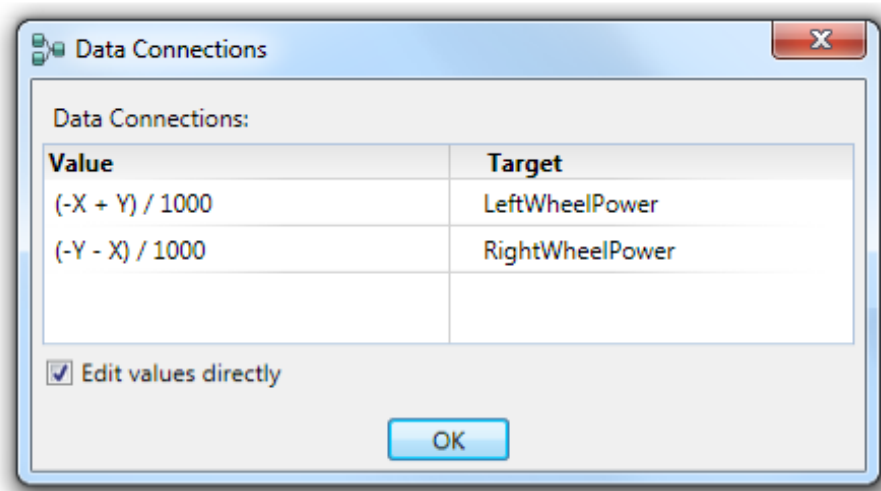


Рисунок 4.7 – Окно Data Connection

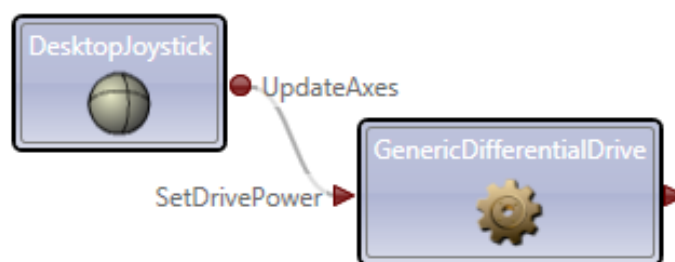


Рисунок 4.8 – Готовая диаграмма

4. Далее выделите сервис Generic Differential Drive и в окне Properties, в раскрывающемся списке Configuration выберите Use a manifest. Затем нажмите кнопку Import и выберите соответствующий манифест (рис. 4.9).

5. Запустите программу на выполнение (рис. 4.10).

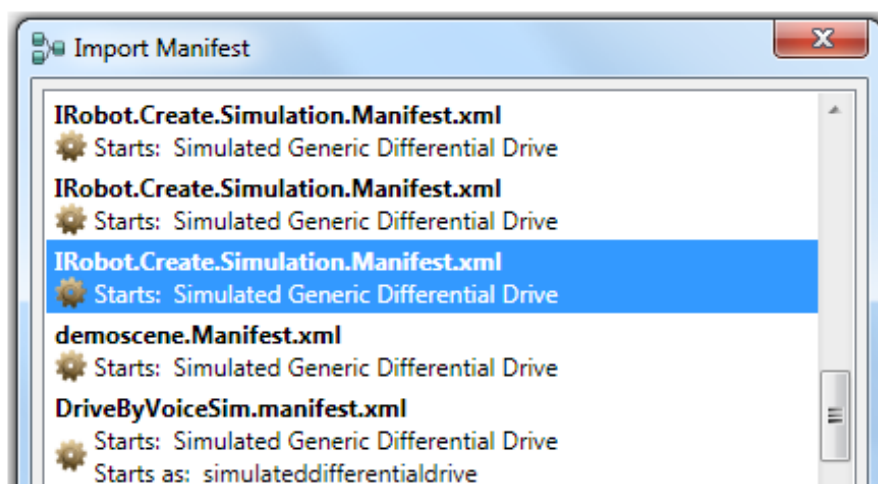


Рисунок 4.9 – Окно импорта манифестов

4.2.2. Методика организации автоматизированного управления с помощью фиксированного набора команд

1. Для организации автоматизированного управления добавьте на диаграмме сервис DirectionDialog и блок If (рис. 4.11).

2. Соедините выход блока If, соответствующий действию «Вперед» (Forwards), с сервисом дифференциального привода Generic Differential Drive. В появившемся окне Connections, выберите DriveDistance. В следующем окне заполните столбец Value (рис. 4.12).

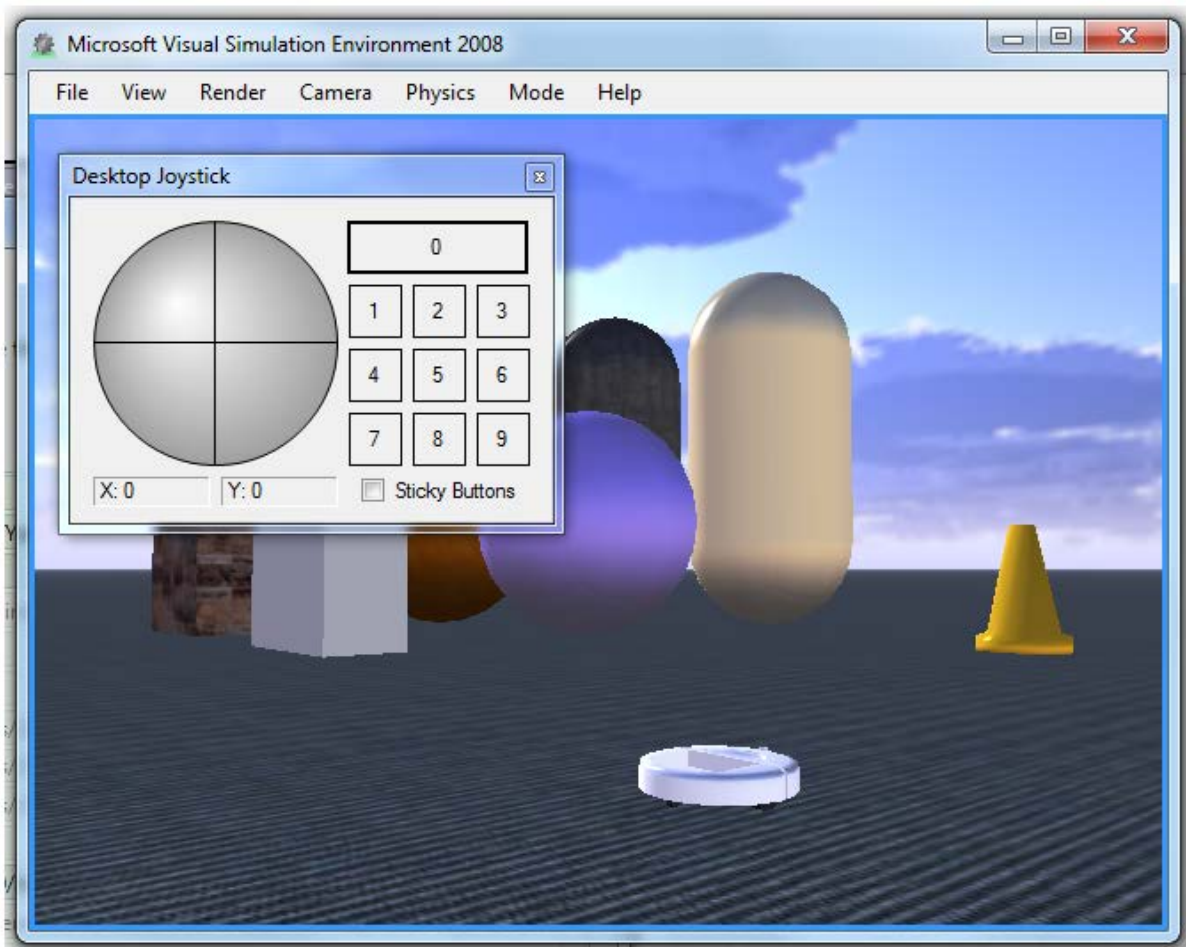


Рисунок 4.10 – Окно симуляции робота IRobot Create

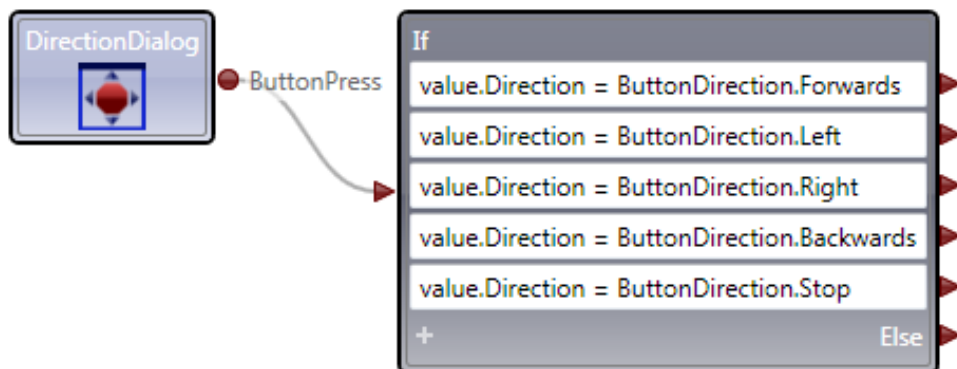


Рисунок 4.11 – Сервис DirectionDialog и блок If

3. Выполните те же операции для выхода блока If соответствующему действию «Влево» (Left). В появившемся окне Connections, выберите RotateDegrees.

4. В следующем окне заполните столбец Value (рис. 4.13).

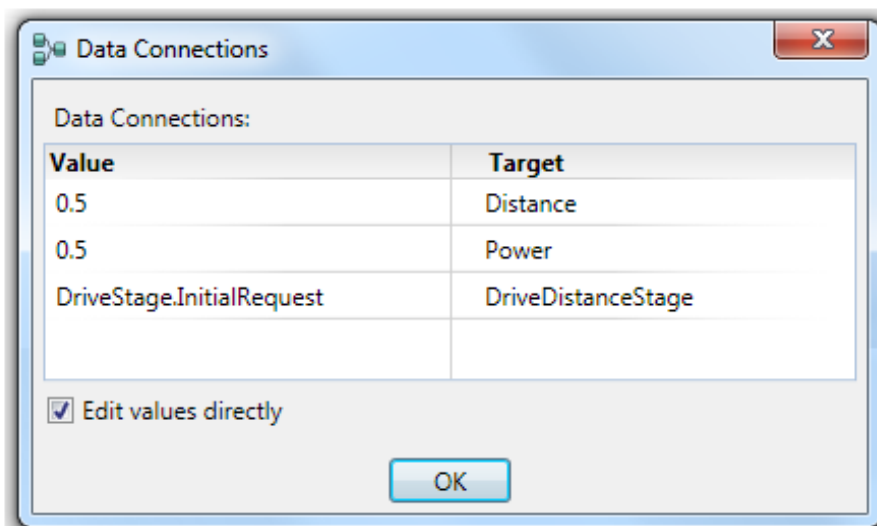


Рисунок 4.12 – Окно Data Connections для действия «Вперед»

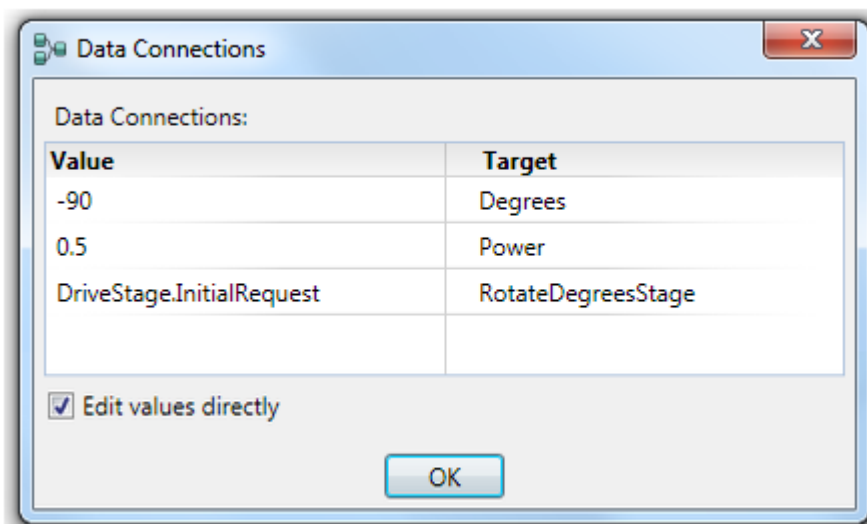


Рисунок 4.13 – Окно Data Connections для действия «Влево»

5. Соедините оставшиеся выходы блока If с дифференциальными приводами в соответствии с таблицей 4.1. Полученная диаграмма приведена на рис. 4.14. Импортируйте манифест и запустите программу на выполнение.

6. В созданной диаграмме задание параметров происходит непосредственно через редактирование сервиса привода. Возможен альтернативный вариант, в котором задание параметров происходит с помощью блоков констант Data (рис. 4.15).

Таблица 4.1. Описание параметров сервиса Generic Differential Drive

Направление	Запрос к сервису привода	Дистанция (Distance)	Поворот (Degrees)	Мощность (Power)
Вперед (Forwards)	DriveDistance	0.5		0.5
Назад (Backwards)	DriveDistance	0.5		– 0.5
Влево (Left)	RotateDegrees		90	0.5
Вправо (Right)	RotateDegrees		– 90	0.5
Стоп (Stop)	SetDrivePower			0.0

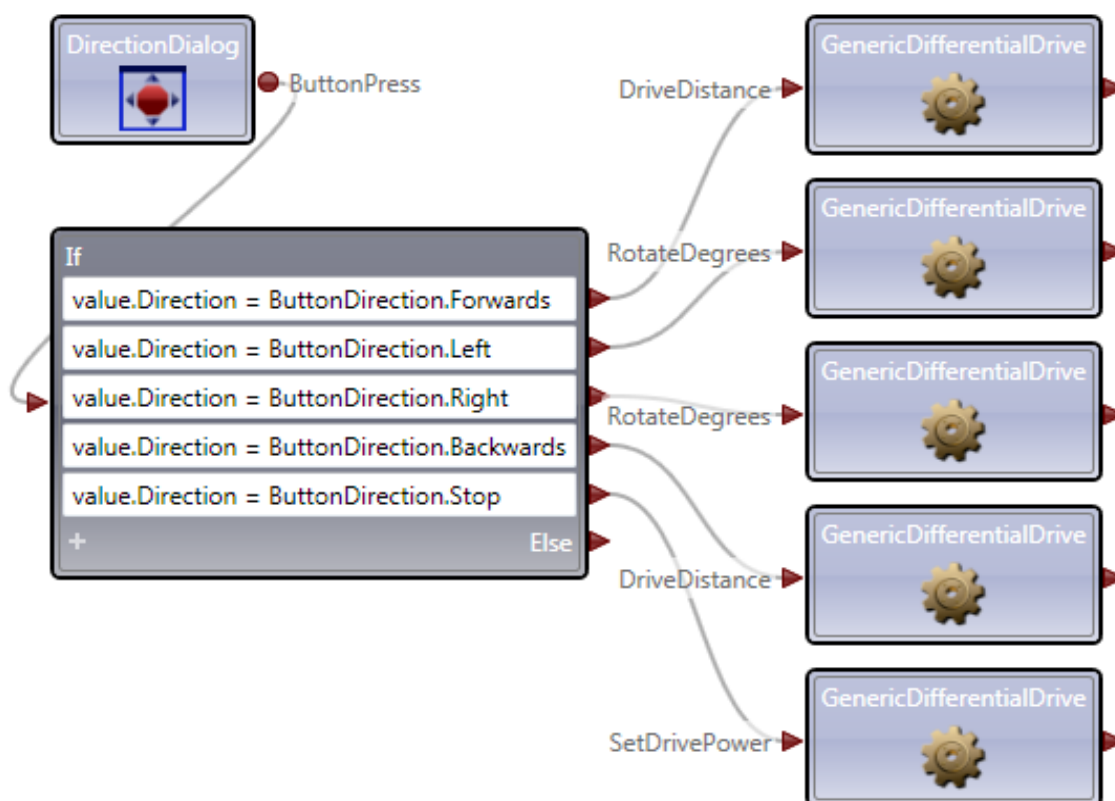


Рисунок 4.14 – Готовая диаграмма

7. Для реализации командного управления с непрерывным перемещением также можно использовать блок Data (рис. 4.16). В данном случае происходит управление по мощности привода, для задания параметров мощности необходимо редактировать сервис привода.

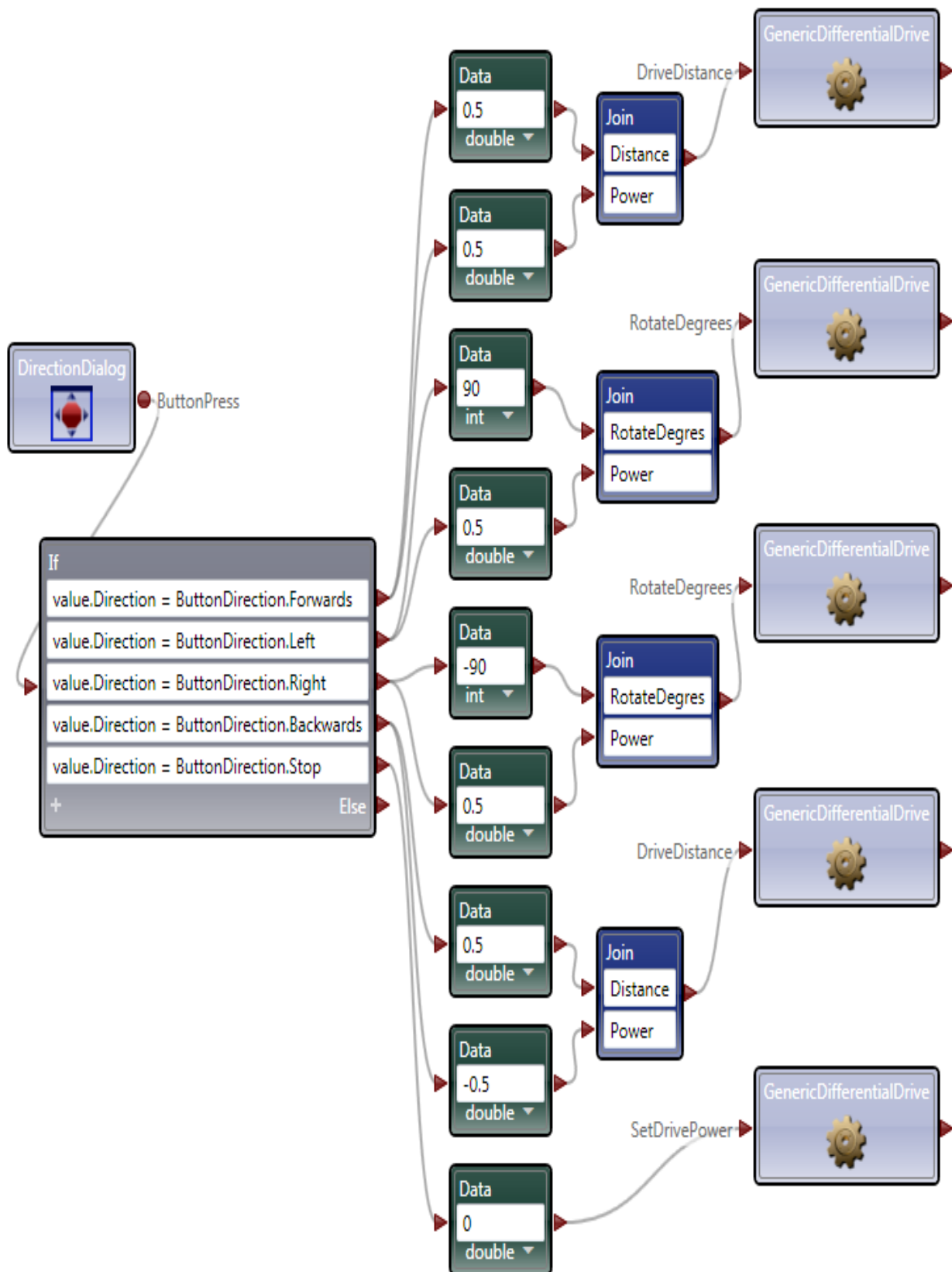


Рисунок 4.15 – Диаграмма с заданием параметров с помощью блока Data

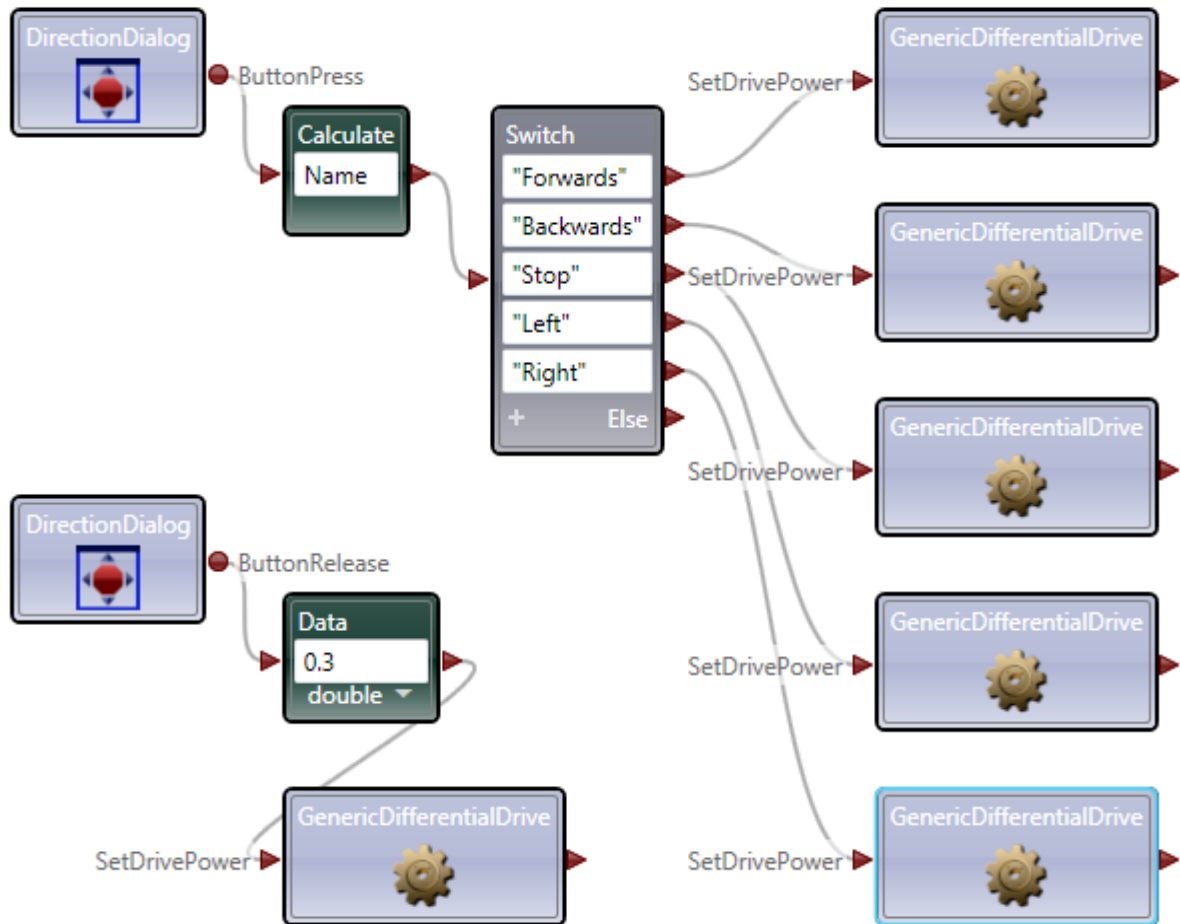


Рисунок 4.16 – Диаграмма командного управления с непрерывным перемещением

4.3. Лабораторная работа 4. Моделирование автоматизированного управления мобильным роботом

4.3.1. Цель и содержание

Целью лабораторной работы является приобретение студентами практических навыков организации автоматизированного управления автономным подвижным объектом с помощью задающего манипулятора и с использованием фиксированного набора команд.

Содержание

1. Организация автоматизированного управления с помощью виртуального задающего манипулятора.
2. Организация автоматизированного управления с помощью фиксированного набора команд.

4.3.2. Аппаратура и материалы

Для изучения материалов этой лабораторной работы необходимо:

1. Компьютер, удовлетворяющий минимальным системным требованиям для установки Microsoft Robotics Developer Studio R4.
2. Установленная платформа RDS R4.

4.3.3. Порядок выполнения работы

1. Изучение теоретического материала и методических рекомендаций.
2. Выполнение задания, предлагаемого в лабораторной работе.

Задание:

1. Организуйте автоматизированное управление виртуальным задающим манипулятором на основе методики 4.2.1 и выбора двух произвольных вариантов манифеста.
2. Организуйте автоматизированное управление с помощью фиксированного набора команд тремя методами на основе методики 4.2.2 и варианта задания из табл. 4.2 и 4.3.

Таблица 4.2. Варианты заданий для самостоятельного выполнения на основе методики 4.2.2 (первые два метода)

№ варианта	Направление	Дистанция (Distance)	Поворот (Degrees)	Мощность (Power)
1	Вперед (Forwards)	0,1	-	0,8
	Назад (Backwards)	0,8	-	0,1
	Влево (Left)	-	90	0,9
	Вправо (Right)	-	-90	0,9
2	Вперед (Forwards)	0,2	-	0,6
	Назад (Backwards)	0,6	-	0,2
	Влево (Left)	-	45	0,7
	Вправо (Right)	-	-45	0,7
3	Вперед (Forwards)	0,3	-	0,4
	Назад (Backwards)	0,4	-	0,3
	Влево (Left)	-	60	0,5
	Вправо (Right)	-	-60	0,5

Продолжение таблицы 4.2

№ варианта	Направление	Дистанция (Distance)	Поворот (Degrees)	Мощность (Power)
4	Вперед (Forwards)	0,4	-	0,2
	Назад (Backwards)	0,2	-	0,4
	Влево (Left)	-	75	0,3
	Вправо (Right)	-	-75	0,3
5	Вперед (Forwards)	0,5	-	0,9
	Назад (Backwards)	0,9	-	0,5
	Влево (Left)	-	15	0,1
	Вправо (Right)	-	-15	0,1
6	Вперед (Forwards)	0,6	-	0,1
	Назад (Backwards)	0,1	-	0,6
	Влево (Left)	-	30	0,8
	Вправо (Right)	-	-30	0,8
7	Вперед (Forwards)	0,7	-	0,3
	Назад (Backwards)	0,3	-	0,7
	Влево (Left)	-	-90	0,6
	Вправо (Right)	-	90	0,6
8	Вперед (Forwards)	0,8	-	0,5
	Назад (Backwards)	0,5	-	0,8
	Влево (Left)	-	-45	0,4
	Вправо (Right)	-	45	0,4
9	Вперед (Forwards)	0,9	-	0,7
	Назад (Backwards)	0,7	-	0,9
	Влево (Left)	-	-60	0,2
	Вправо (Right)	-	60	0,2
10	Вперед (Forwards)	1,0	-	1,0
	Назад (Backwards)	1,0	-	1,0
	Влево (Left)	-	-75	1,0
	Вправо (Right)	-	75	1,0
11	Вперед (Forwards)	0,75	-	0,25
	Назад (Backwards)	0,25	-	0,75
	Влево (Left)	-	-15	0,65
	Вправо (Right)	-	15	0,65
12	Вперед (Forwards)	0,35	-	0,65
	Назад (Backwards)	0,65	-	0,35
	Влево (Left)	-	-30	0,25
	Вправо (Right)	-	30	0,25

Таблица 4.3. Варианты заданий для самостоятельного выполнения на основе методики 4.2.2 (третий метод)

№ варианта	Направление	Мощность левого двигателя (Power)	Мощность правого двигателя (Power)
1	Вперед (Forwards)	0,1	0,1
	Назад (Backwards)	0,9	0,9
	Влево (Left)	0,0	1,0
	Вправо (Right)	1,0	0,0
2	Вперед (Forwards)	0,2	0,2
	Назад (Backwards)	0,8	0,8
	Влево (Left)	0,2	0,8
	Вправо (Right)	0,8	0,2
3	Вперед (Forwards)	0,3	0,3
	Назад (Backwards)	0,7	0,7
	Влево (Left)	0,3	0,7
	Вправо (Right)	0,7	0,3
4	Вперед (Forwards)	0,4	0,4
	Назад (Backwards)	0,6	0,6
	Влево (Left)	0,4	0,6
	Вправо (Right)	0,6	0,4
5	Вперед (Forwards)	0,5	0,5
	Назад (Backwards)	0,5	0,5
	Влево (Left)	0,6	0,4
	Вправо (Right)	0,4	0,6
6	Вперед (Forwards)	0,6	0,6
	Назад (Backwards)	0,4	0,4
	Влево (Left)	0,7	0,3
	Вправо (Right)	0,3	0,7
7	Вперед (Forwards)	0,7	0,7
	Назад (Backwards)	0,3	0,3
	Влево (Left)	0,8	0,2
	Вправо (Right)	0,2	0,8
8	Вперед (Forwards)	0,8	0,8
	Назад (Backwards)	0,2	0,2
	Влево (Left)	0,9	0,1
	Вправо (Right)	0,1	0,9
9	Вперед (Forwards)	0,9	0,9
	Назад (Backwards)	0,1	0,1
	Влево (Left)	1,0	0,0
	Вправо (Right)	0,0	1,0

Продолжение таблицы 4.3

№ варианта	Направление	Мощность левого двигателя (Power)	Мощность правого двигателя (Power)
10	Вперед (Forwards)	1,0	1,0
	Назад (Backwards)	1,0	1,0
	Влево (Left)	0,25	0,75
	Вправо (Right)	0,75	0,25
11	Вперед (Forwards)	0,35	0,35
	Назад (Backwards)	0,75	0,75
	Влево (Left)	0,45	0,55
	Вправо (Right)	0,55	0,45
12	Вперед (Forwards)	0,65	0,65
	Назад (Backwards)	0,45	0,45
	Влево (Left)	0,75	0,25
	Вправо (Right)	0,25	0,75

4.4. Контрольные вопросы

1. Охарактеризуйте особенности системы командного управления.
2. Назовите принципиальные отличия организации автоматизированного управления с помощью виртуального задающего манипулятора от варианта использования фиксированного набора команд.
3. Прокомментируйте возможности использования координат трекбола в сервисе Desktop Joystick.
4. Перечислите и прокомментируйте варианты организации автоматизированного управления с помощью фиксированного набора команд применяемые в данной работе.
5. Охарактеризуйте взаимосвязь сервиса Generic Differential Drive и манифеста.

ГЛАВА 5. АВТОМАТИЧЕСКОЕ УПРАВЛЕНИЕ РОБОТОМ

Существует чрезвычайно большое разнообразие автоматических систем, выполняющих те или иные функции по управлению самыми различными физическими процессами во всех областях техники [5]. В этих системах сочетаются весьма разнообразные по конструкции механические, электрические и другие устройства, составляя, в общем, сложный комплекс взаимодействующих друг с другом звеньев.

5.1. Краткие теоретические сведения

Система автоматического управления (САУ) – комплекс устройств, предназначенных для автоматического изменения одного или нескольких параметров объекта управления с целью установления требуемого режима его работы [6].

Примерами автоматических систем могут служить [5]:

- автомат включения освещения, в котором имеется фотоэлемент, реагирующий на силу дневного света, и специальное устройство для включения освещения, срабатывающее от определенного сигнала фотоэлемента;
- автомат, выбрасывающий какие-либо предметы (билеты, шоколад) при опускании в него определенной комбинации монет;
- автоматический регулятор скорости вращения двигателя, поддерживающий постоянную угловую скорость двигателя независимо от внешней нагрузки (аналогично — регуляторы температуры, давления, напряжения, частоты и пр.);
- автопилот, поддерживающий определенный курс и высоту полета самолета без помощи летчика;
- следящая система, на выходе которой с определенной точностью воспроизводится произвольное во времени изменение какой-нибудь величины, поданной на вход;

– система самонаведения снаряда на цель и пр.

Все эти и им подобные автоматические системы можно разделить на два больших класса:

1. Автоматы, выполняющие определенного рода одноразовые или многократные операции; сюда относятся, например, автомат включения освещения, билетный автомат и т. п.

2. Автоматические системы, которые в течение длительного времени нужным образом изменяют (поддерживают неизменными) какие-либо физические величины (координаты движущегося объекта, скорость движения, электрическое напряжение, частоту, температуру, давление, громкость звука и пр.) в том или ином управляемом процессе. Сюда относятся автоматические регуляторы, следящие системы, автопилоты, системы самонаведения и т. п.

Наиболее распространенными задачами, которые решают системы автоматического управления, являются стабилизация, выполнение заданной программы и слежение.

Системы, поддерживающие постоянное значение управляемой величины при изменяющихся возмущающих воздействиях называются **стабилизирующими системами** (стабилизация температуры, давления, напряжения, углового положения летательного аппарата и т. п.). Управляющее устройство таких систем иногда называют регулятором, а сами системы – системами автоматического регулирования.

Системы, изменяющие управляемую величину по заранее заданной программе, называются **программными системами**. Такая задача возникает, например, при выводе ракеты на заданную траекторию, при развороте зеркала телескопа с целью компенсации вращения Земли и в других случаях.

Системы, управляемая величина которых воспроизводит произвольно изменяющееся задающее воздействие, называются **следящими системами**. Так, антенна радиолокатора должна следить за маневрирующей целью, фреза копировально-фрезерного станка должна воспроизводить движение шупа по копиру и т. д.

В перечисленных выше системах требуемое значение управляемой величины оставалось постоянным, либо изменялось по определенной программе, либо задавалось извне. Однако в ряде случаев сама система в процессе управления должна производить поиск такого требуемого значения, которое необходимо в данный момент времени выдерживать, чтобы режим работы управляемого объекта был наивыгоднейшим. Такие системы автоматического управления называются **экстремальными**.

Экстремальное управление может применяться, например, для поддержания наиболее экономичной скорости полета самолета, соответствующей минимальному секундному расходу топлива при изменяющихся высоте полета, массе самолета, скорости и направлении ветра и т. д. При этом будет достигнута и максимальная дальность полета при заданном запасе топлива.

Управляющее устройство системы создается применительно к каждому конкретному управляемому объекту. Если свойства объекта и возмущающие воздействия хорошо известны, то можно заранее уверенно выбирать такие структуру и параметры управляющего устройства, при которых обеспечивается надлежащее качество процессов в проектируемой автоматической системе. Если же свойства самого объекта известны недостаточно достоверно и если к тому же они могут в процессе работы в некоторых пределах случайным образом меняться, то и параметры управляющего устройства можно подобрать лишь ориентировочно. В этом случае при отклонении параметров объекта от расчетных будет происходить ухудшение качества процессов в системе.

В тех случаях, когда закон изменения параметров объекта во времени заранее хорошо известен, можно рассчитать, как и когда нужно менять параметры управляющего устройства, чтобы качество работы автоматической системы в целом оставалось неизменно хорошим. Так делается, например, в некоторых системах управления ракетами, у которых в процессе полета из-за выгорания топлива изменяется масса, а из-за изменения плотности атмосферы – эффективность исполнительных органов.

Если же составление такой программы оказывается невозможным вследствие незнания истинного закона изменения хотя бы некоторых параметров объекта, то прибегают к построению так называемой **самонастраивающейся системы**. Для этого в систему вводятся дополнительные автоматические устройства, которые определяют отклонение какого-либо показателя качества от его требуемого значения и изменяют параметры управляющего устройства или даже его структуру с целью минимизации указанного отклонения.

Экстремальные и самонастраивающиеся системы относятся к так называемым адаптивным системам автоматического управления. Образцы для подражания дают процессы адаптации в живой природе. Так, стабилизация температуры тела человека и некоторых животных при низких температурах окружающей среды осуществляется за счет изменения притока крови к поверхностным тканям, а при высоких температурах включаются механизмы потоотделения и дыхания, обеспечивающие интенсивный отвод избытка тепла. Таким образом, имеет место изменение алгоритма управления.

В данной работе будет рассмотрено **автоматическое программное управление** роботом, которое по типу движения относится к непрерывному (контурному) управлению. **Непрерывное программное управление** заключается в том, что либо автономный подвижный объект, либо рабочий орган манипулятора движутся непрерывно по траектории, которая задана программно либо непрерывной функцией времени, либо дискретной функцией времени, т. е. последовательностью точек. При этом важно, что объект управления должен проходить эти точки без остановок.

В работе реализуется совместное управление двумя приводами (дифференциальный привод). Основная задача – координация приводов для обеспечения движения по заданной траектории. Составление уравнения кинематики для задания траектории движения и его алгоритмизация происходит на этапе программирования.

При движении по **траектории «от точки к точке»** в каждой точке происходит остановка робота и смена направления движения, либо выполнение каких-либо операций, например работа с датчиками и сенсорами.

5.2. Методические рекомендации

5.2.1. Методика организации автоматического управления при движении по окружности

1. Для организации автоматического управления при движении по окружности необходимо организовать блок TurningRadiusToWheelPowers, который будет отвечать за распределение мощности в дифференциальном двигателе и будет необходим при построении программ реализующих в том числе другие траектории движения. Добавьте блок Activity. На панели Properties задайте название данного блока TurningRadiusToWheelPowers.

2. В окне Actions and Notifications измените имя Action на CalculateWheelPowers (рис. 5.1).

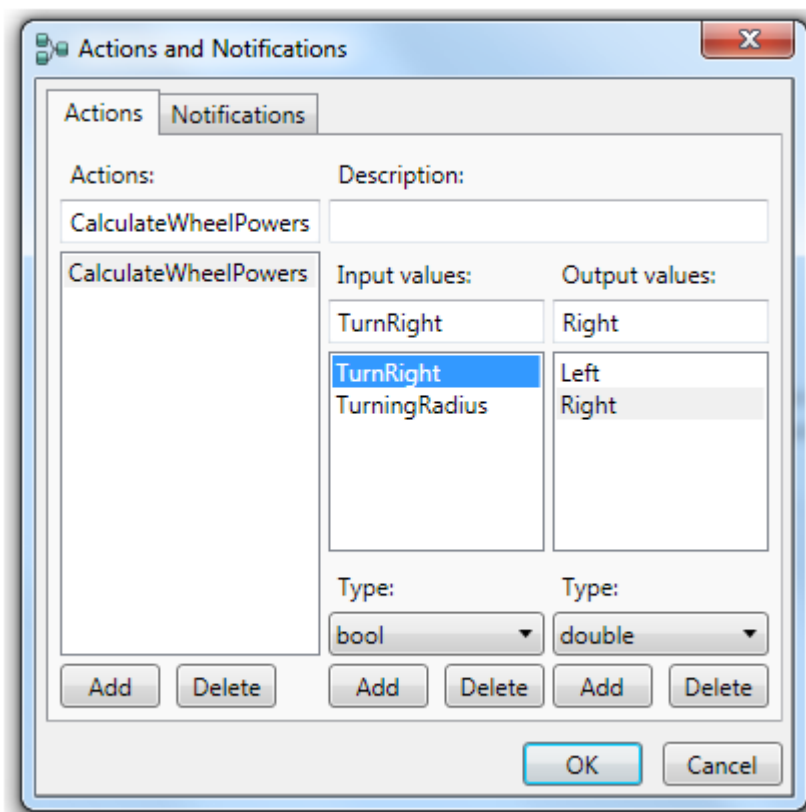


Рисунок 5.1 – Вкладка Actions блока TurningRadiusToWheelPowers

3. Добавьте входящие и исходящие сообщения, обратив внимание, что значение TurnRight должно иметь тип данных boolean, а значение TurningRadius, Left и Right тип данных double.

4. Добавьте в рабочее окно блока TurningRadiusToWheelPowers необходимые блоки, соедините их и заполните поля (рис. 5.2). Первый блок Calculate просто транслирует значение TurnRight в блок Join. Блок Data устанавливает мощность, передаваемую на левое колесо в размере 1.0. Второй блок Calculate выполняет вычисления для определения мощности на правом колесе. Все три значения передаются в блок Join, который объединяет все полученные значения в одну шину.

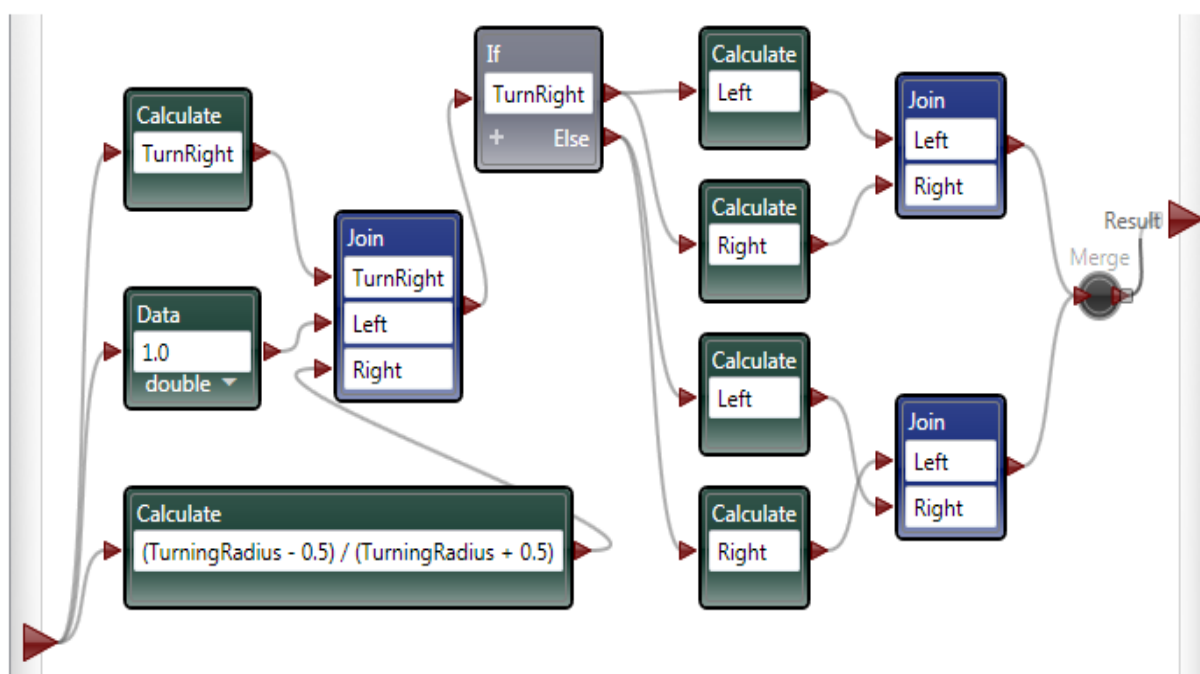


Рисунок 5.2 – Окно блока TurningRadiusToWheelPowers

5. Добавьте блок If для проверки значения TurnRight. Соедините выход Join с входом If. Если значение TurnRight истинно, то питание на левый и правый двигатель передаются без изменений.

6. Добавьте два блока Calculate, которые будут ретранслировать эти значения и еще один Join. Если значение TurnRight ложно, то значения передаваемой мощности на двигатели должны поменяться местами. Перейдите на диаграмму.

7. Добавьте на диаграмме блоки и сервис дифференциального привода, свяжите их с блоком TurningRadiusToWheelPowers (рис. 5.3). При выполнении данной программы, робот начнет движение непосредственно после запуска симулятора.

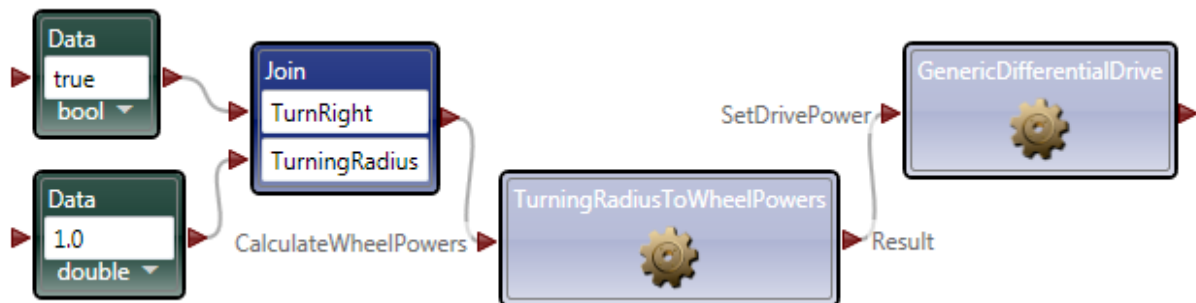


Рисунок 5.3 – Окно диаграммы движения по окружности

8. Для того, что бы взять на себя контроль над запуском робота, необходимо добавить блок Data и сервис SimpleDialog, выход которого соединить с входом имеющегося блока Data (рис 5.4). Данное программное решение представляет собой простейший элемент супервизорного управления.

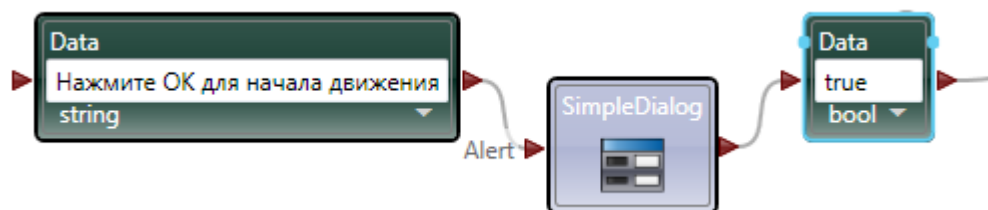


Рисунок 5.4 – Добавление сервиса SimpleDialog

5.2.2. Методика организации автоматического управления при движении по дуге

1. Для организации автоматического управления при движении по дуге необходимо модифицировать предыдущую программу. В определенный момент времени движения по окружности, когда робот опишет дугу, установить мощность двигателей в ноль. Что бы задать данный момент, необходимо использовать сервис Timer.

2. Добавьте один сервис Timer на диаграмму и соедините его входы с выходами GenericDifferentialDrive (соединив порт SetDrivePowerSuccess с

портом SetTimer (рис. 5.5). При соединении сервиса дифференциального привода с таймером, в окне Data Connections выберите Edit Values Directly и введите 1500 мс (рис. 5.6).

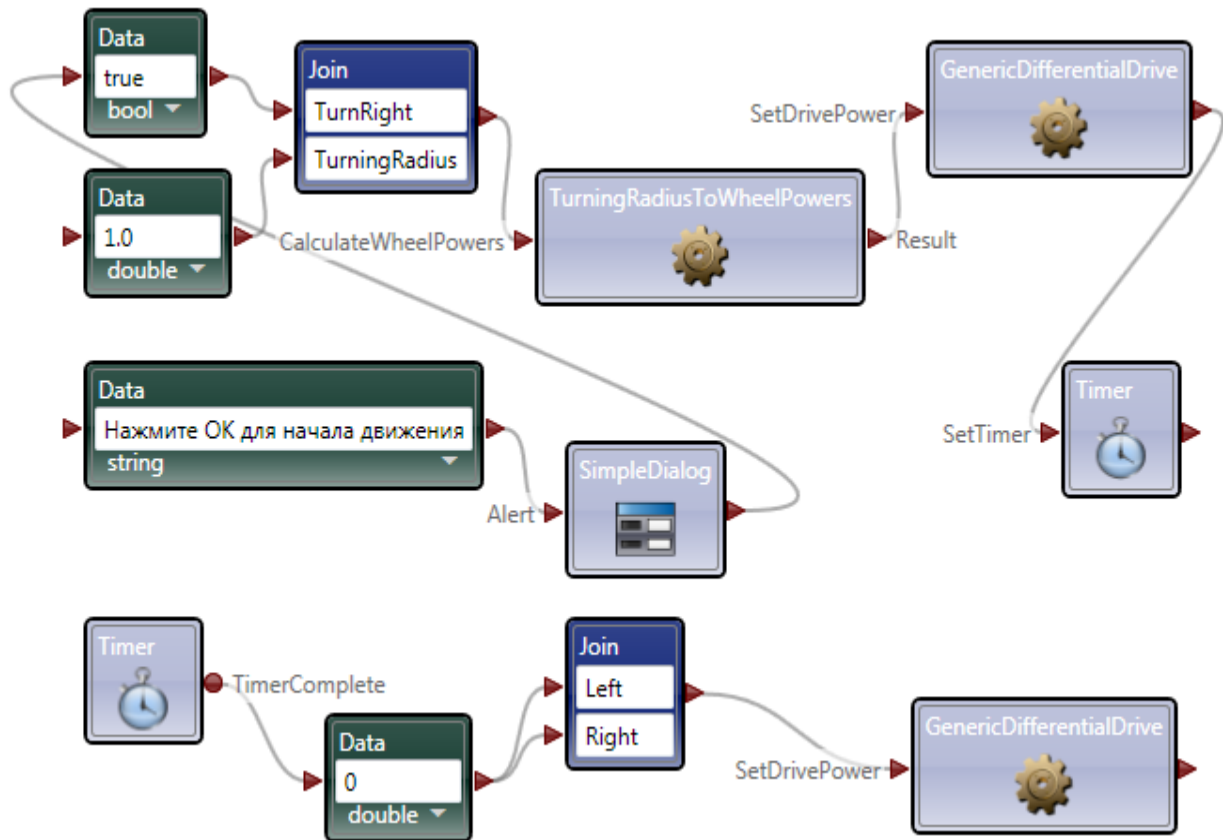


Рисунок 5.5 – Окно диаграммы движения по дуге

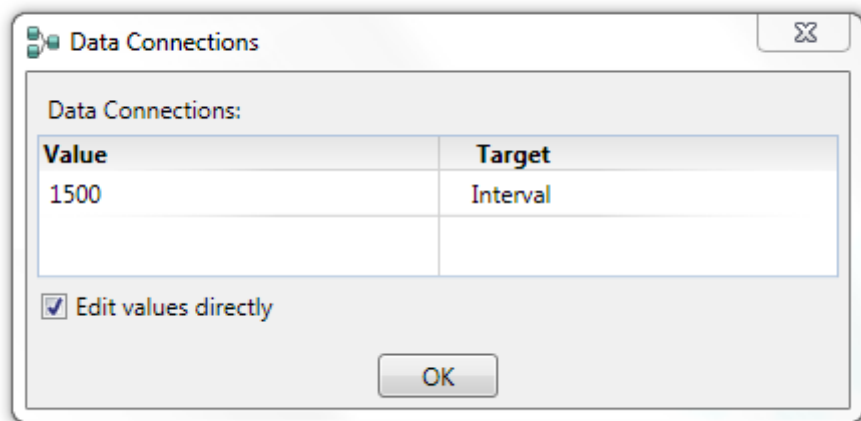


Рисунок 5.6 – Окно Data Connections

3. Скопируйте сервис Timer на диаграмму и соедините его с блоком Data, определяющим уровень мощности. Соедините выход Data с входами Join.

Соедините выход Join с входами GenericDifferentialDrive и установите LeftWheelPower для Left и RightWheelPower для Right.

5.2.3. Методика организации автоматического управления при движении по траектории в виде «восьмерки»

1. Для организации автоматического управления при движении по траектории в виде «восьмерки» необходимо составить соответствующую программу (рис. 5.7). При ее выполнении робот будет описывать окружность. Когда траектория замкнется, мощность на дифференциальном приводе перераспределится в другом соотношении, и робот начнет описывать вторую окружность, которая будет иметь одну точку пересечения с первой.

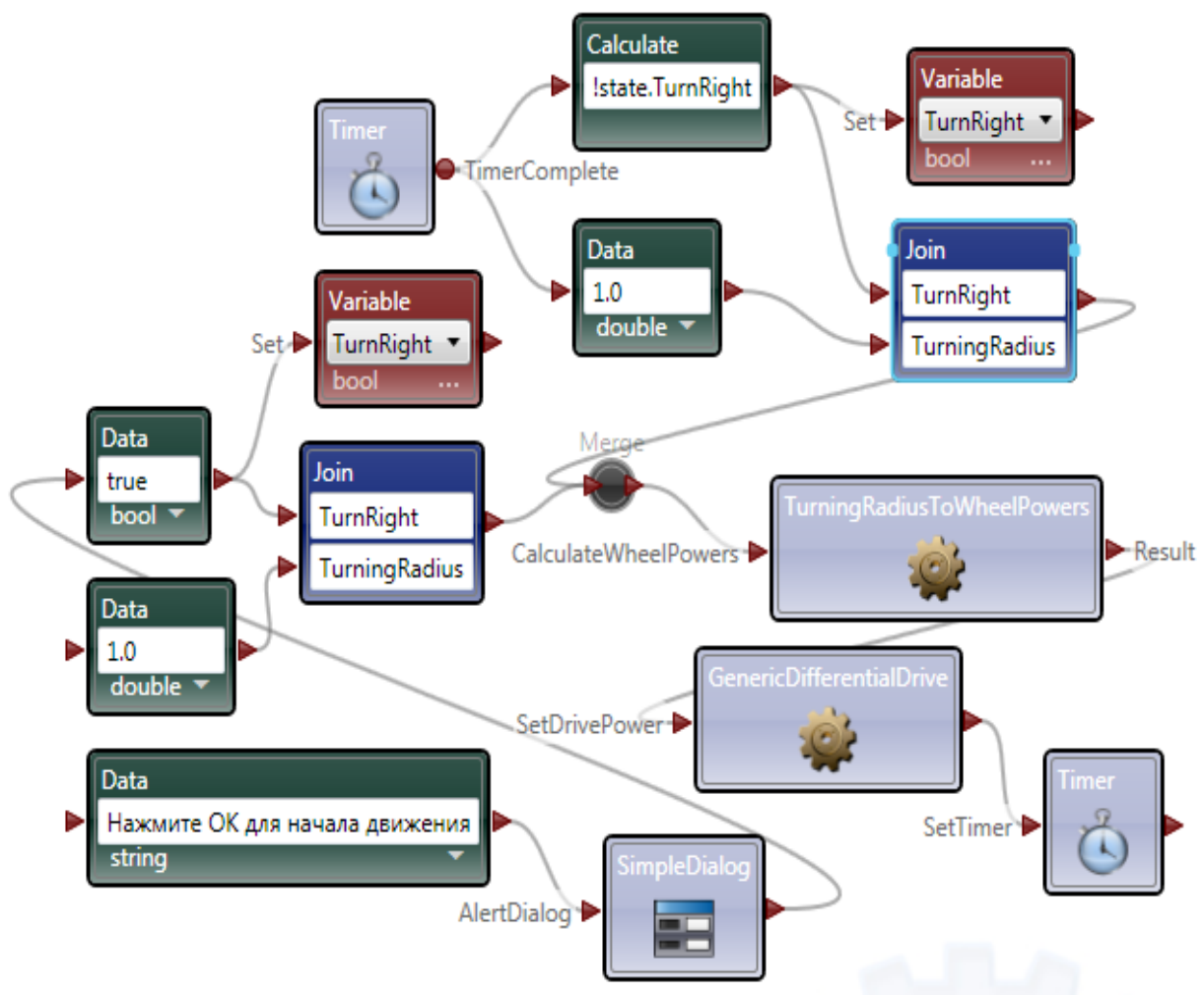


Рисунок 5.7 – Окно диаграммы движения по «восьмерке»

2. Программно это реализовано с помощью записи в переменную TurnRight значения true или false. Данная переменная отвечает за движение по часовой стрелке (true) и против часовой стрелки (false). Для определения момента перераспределения мощности, нужно установить таймер в 5000 мс, приблизительно столько времени необходимо роботу Create для прохождения круга с единичным радиусом.

3. Для организации изменения значения переменной TurnRight применяется блок Calculate с выражением **!state.TurnRight**.

5.2.4. Методика организации автоматического управления при движении по траектории с переменным радиусом

1. Для организации автоматического управления при движении по траектории с переменным радиусом необходимо изменять радиус поворота в каждом цикле выполнения кода. Для выполнения этой модификации нужно добавить новую переменную с названием TurningRadius и создать необходимые блоки по аналогии с TurnRight. Однако, в отличие от переключения логического значения TurnRight необходимо умножать значение TurningRadius на некоторое число больше 1 (например, на 1,1).

2. Составьте диаграмму для организации движения по спирали (рис. 5.8). Запустите ее на выполнение. Можно отметить, что робот не возвращается с точку начала движения. Причина в жестко заданном времени поворота – 5000 мс.

3. Попробуйте исправить эту ошибку, вводя новую Variable для значения времени поворота и увеличивая ее на некоторую величину одновременно с радиусом поворота. Этот подход, основанный на вычислениях, а не на датчиках получил название управление по обратной связи.

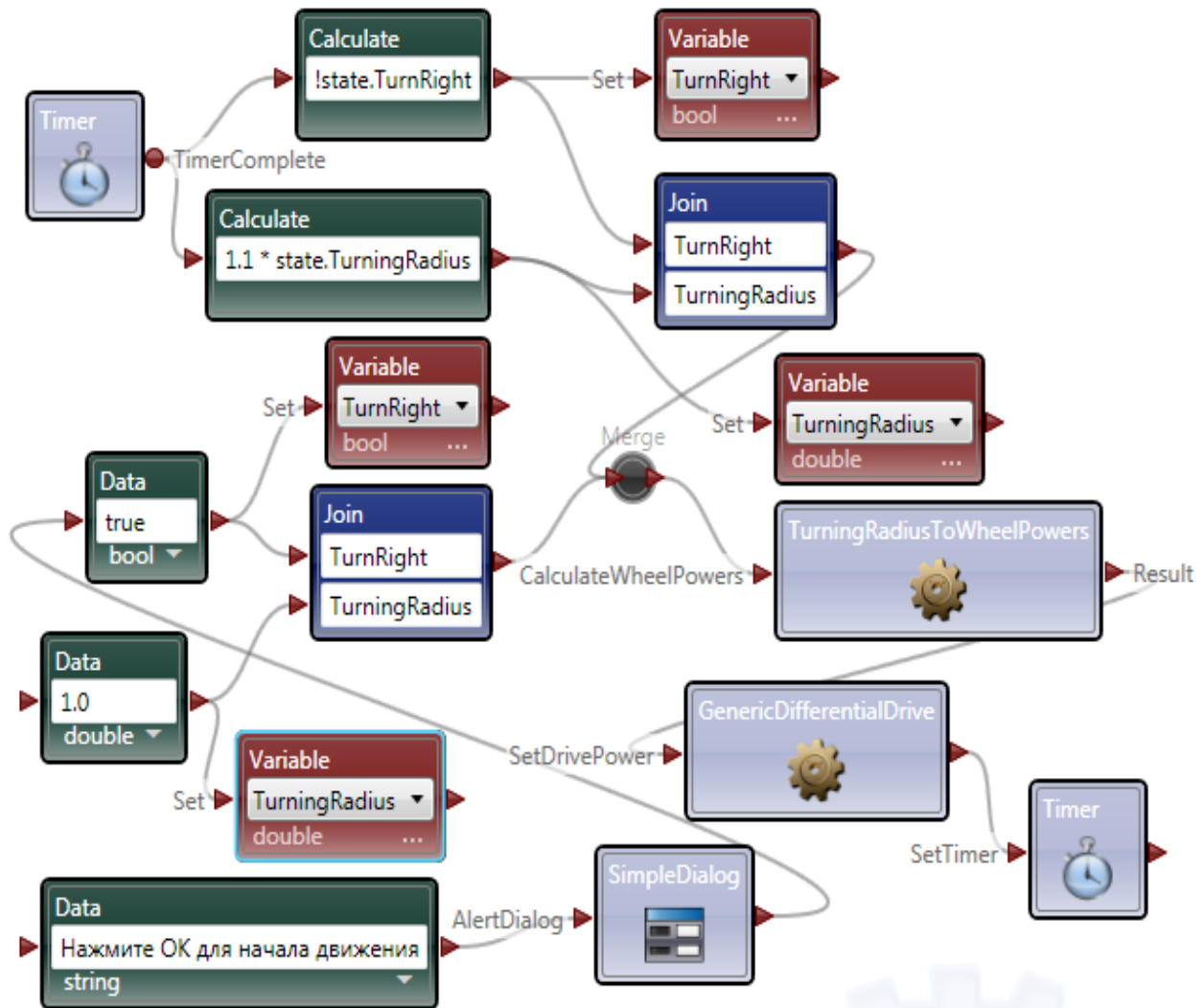


Рисунок 5.8 – Окно диаграммы движения по спирали

5.2.5. Методика организации автоматического управления при движении по траектории «от точки к точке»

1. Для организации автоматического управления при движении по траектории «от точки к точке» необходимо создать диаграмму (рис. 5.9).

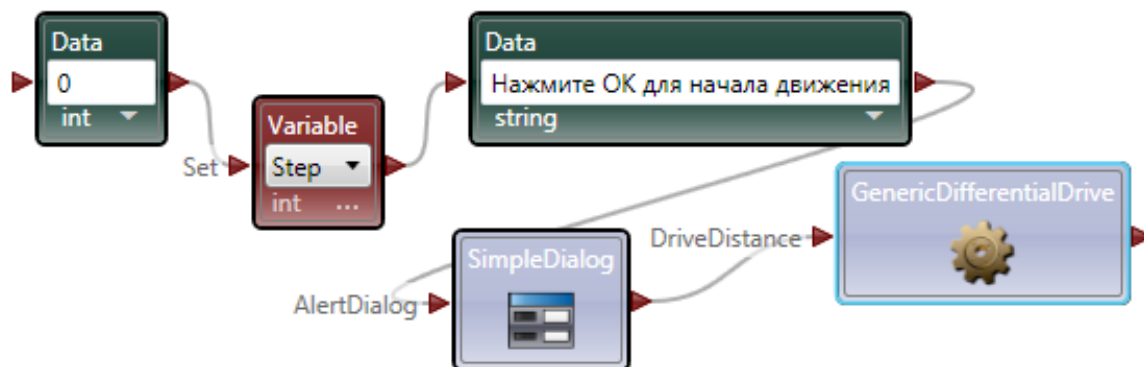


Рисунок 5.9 – Первый фрагмент диаграммы движения «от точки к точке»

2. Переменная Step необходима для организации пошагового выполнения алгоритма движения. При соединении SimpleDialog и GenericDifferentialDrive необходимо определить, что величина расстояния (Distance) составляет 0,5, мощность (Power) составляет 0,5 и DriveDistanceStage является DriveStage.InitialRequest. Данная диаграмма определит движение робота на первом этапе своей траектории.

3. Для организации последующих этапов движения робота необходимо добавить в диаграмму второй фрагмент, представленный на рис. 5.10.

4. В первом блоке If проверяется завершение выполнения движения на расстояние. Во втором блоке If проверяется завершение вращения.

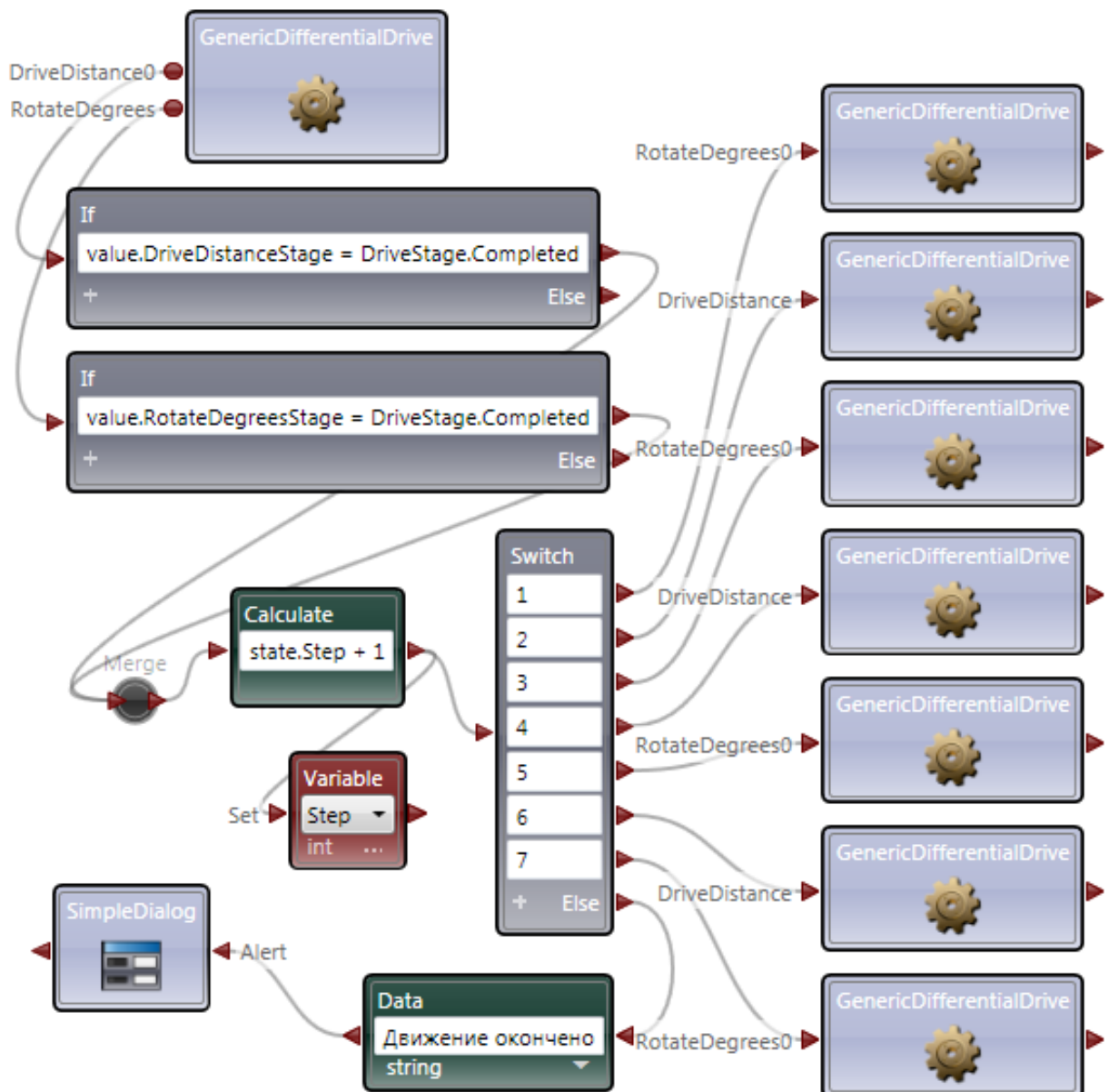


Рисунок 5.10 – Второй фрагмент диаграммы движения «от точки к точке»

5. В блоке Calculate происходит увеличение значения переменной Step на единицу.

6. Блок Switch определяет соответствие этапа работы программы и совершаемых действий. При соединении блока Switch и сервисов GenericDifferentialDrive необходимо выбрать поочередно RotateDegrees и DriveDistance. Для операции RotateDegrees выбирают значение 90 для параметра градусы (Degrees). Для операции DriveDistance выбирают значение 0.5 для параметра дистанции (Distance).

5.3. Лабораторная работа 5. Моделирование автоматического управления мобильным роботом

5.3.1. Цель и содержание

Целью лабораторной работы является приобретение студентами практических навыков организации автоматического управления автономным подвижным объектом.

Содержание

1. Организация автоматического управления при движении по окружности.
2. Организация автоматического управления при движении по дуге.
3. Организация автоматического управления при движении по траектории в виде «восьмерки».
4. Организация автоматического управления при движении по траектории с переменным радиусом.
5. Организация автоматического управления при движении по траектории «от точки к точке».

5.3.2. Аппаратура и материалы

Для изучения материалов этой лабораторной работы необходимо:

1. Компьютер, удовлетворяющий минимальным системным требованиям для установки Microsoft Robotics Developer Studio R4.
2. Установленная платформа RDS R4.

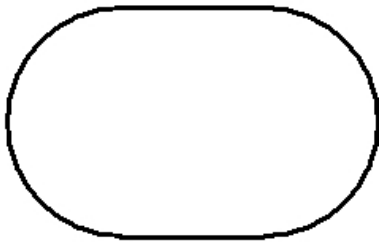
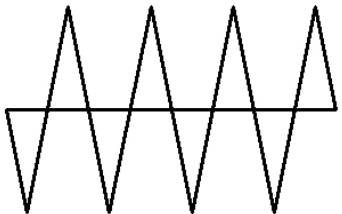
5.3.3. Порядок выполнения работы

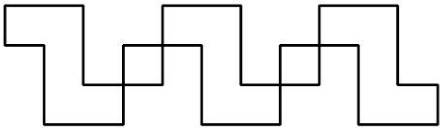
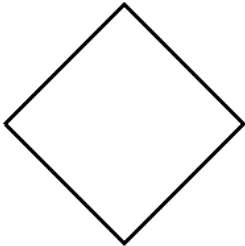
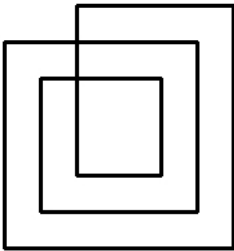
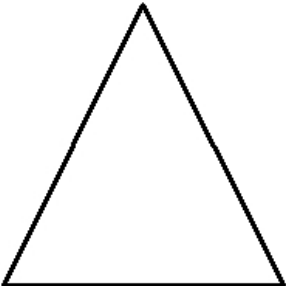
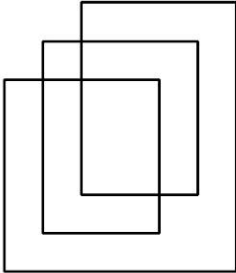
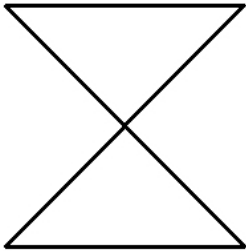
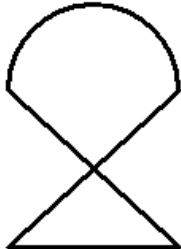
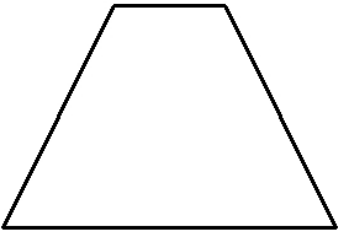
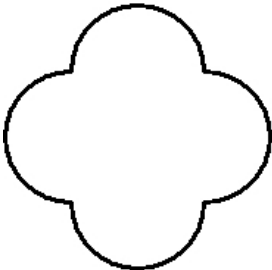
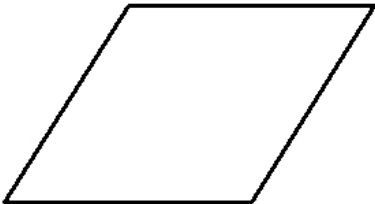
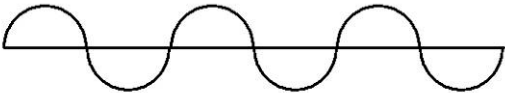
1. Изучение теоретического материала и методических рекомендаций.
2. Выполнение задания, предлагаемого в лабораторной работе.

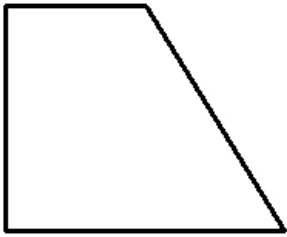
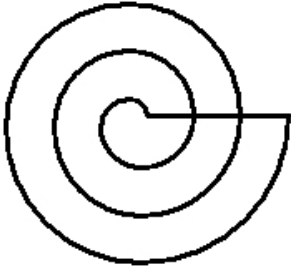
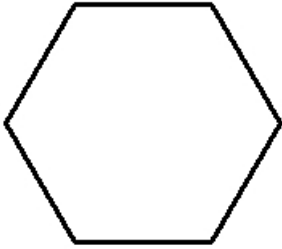
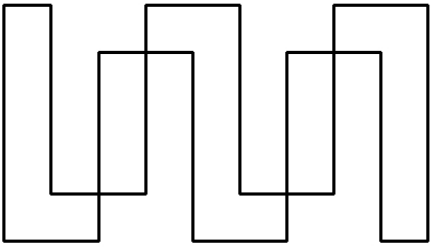
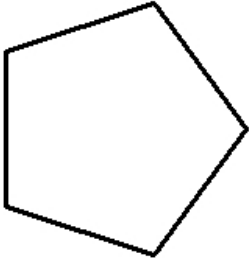
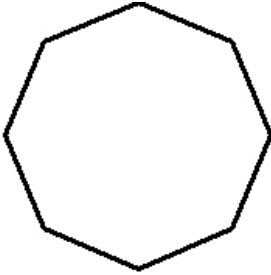
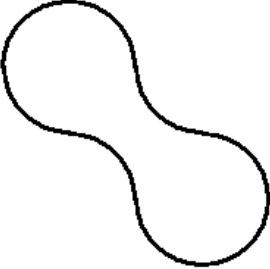
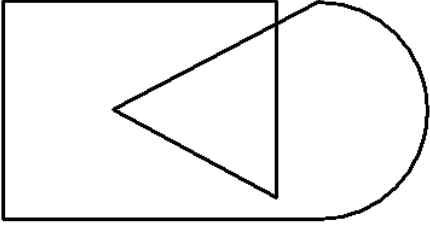
Задание:

1. Организуйте управление при движении по окружности согласно методике 5.2.1 с двумя разными вариантами скорости движения и радиуса поворота.
2. Организуйте управление при движении по дуге согласно методике 5.2.2 с двумя с разными вариантами длины дуги, скорости движения и радиуса поворота.
3. Организуйте управление при движении по траектории в виде «восьмерки» согласно методике 5.2.3 с двумя с разными вариантами длительности вращения, скорости движения и радиуса поворота.
4. Организуйте управление при движении по траектории с переменным радиусом согласно методике 5.2.4. Добейтесь совпадения точек перераспределения мощности.
5. Организуйте алгоритм движения робота по двум заданным траекториям, согласно методике 5.2.5 и варианта задания из табл. 5.1.

Таблица 5.1. Варианты заданий для самостоятельного выполнения

№ варианта	Траектория 1	Траектория 2
1		

№ варианта	Траектория 1	Траектория 2
2		
3		
4		
5		
6		
7		

№ варианта	Траектория 1	Траектория 2
8		
9		
10		
11		
12		

5.4. Контрольные вопросы

1. Дайте определение системы автоматического управления.
2. Назовите примеры и области применения систем автоматического управления.
3. Охарактеризуйте особенности функционирования блока TurningRadius ToWheelPowers.
4. Прокомментируйте роль блока Timer в организации управления при движении по дуге.
5. Назовите переменную, отвечающую за организацию перераспределения мощности на дифференциальном приводе при движении по траектории в виде «восьмерки».
6. Охарактеризуйте отличия переменных TurnRight и TurningRadius в программе движения по траектории с переменным радиусом.
7. Перечислите элементарные операции, выполняющиеся при движении по траектории «от точки к точке».

ГЛАВА 6. ОСНОВЫ СУПЕРВИЗОРНОГО И ИНТЕРАКТИВНОГО УПРАВЛЕНИЯ МОБИЛЬНЫМ РОБОТОМ И МАНИПУЛЯТОРОМ

6.1. Краткие теоретические сведения

Супервизорный способ управления (supervisory system) роботом представляет собой последовательность заранее заключенных в его память и автоматически выполняемых типовых операций, в промежутке между которыми оператор называет очередную операцию, обеспечивая, таким образом, выполнение определенной заданной сложной работы. Директивы оператора при этом включают в себя два вида информации: название (код) типовой операции и ввод численных значений некоторых заранее неопределенных параметров программы этой операции.

Типичным видом последней информации являются необходимые для выполнения операции координаты внешней среды (координаты предметов или места, куда требуется вывести рабочий орган манипулятора и т. п.).

Это осуществляется через систему целеуказания с помощью, например, задающей рукоятки, световой метки на экране дисплея, светового пера, мыши, средств виртуальной реальности в компьютерном трехмерном пространстве. Операция может задаваться через клавиатуру, в том числе виртуальную на экране дисплея, с помощью речевого командного устройства и других средств общения.

Более совершенное **интерактивное управление** предполагает двустороннюю, т. е. диалоговую, связь с роботом. Робот после получения определенного задания может запросить у оператора недостающую для его выполнения информацию или какие-либо уточнения, сообщить о возникновении неисправности или невыполнимости какой-то операции с указанием причин, т. п.

Оператор в свою очередь может запросить дополнительную информацию о состоянии отдельных систем робота, содержании памяти, дополнительную сенсорную информацию о внешней среде. Рассмотренные способы управления

человеком-оператором часто применяются совместно с автоматическим управлением.

Современные пульта управления (терминалы) робототехническими и технологическими системами могут быть стационарными (рис. 6.1) и переносными (рис. 6.2).



Рисунок 6.1 – Пульт управления манипулятором KUKA



Рисунок 6.2 – Переносной пульт оператора SINUMERIK NT 8 (по материалам сайта <http://support.automation.siemens.com/>)

В Microsoft Robotics Developer Studio R4 для реализации интерактивного управления используются следующие сервисы: SimpleDialog, FlexibleDialog, HelloAppsConsole, SimpleDashboard, SoundPlayer. Для реализации системы целеуказания используются следующие сервисы: DirectionDialog, DesktopJoystick, GameController, XInputController, SimpleDashboard, SpeechRecognizer.

В RDS терминал в общем виде представлен с помощью сервиса Simple Dashboard (рис. 6.3). Данный сервис представляет собой окно, в котором сосредоточены различные по своему назначению элементы управления.

Рассмотрим данные элементы:

- Direct Input Device – данный элемент позволяет подключить к виртуальному терминалу устройства позиционирования (джойстики как реальные, так и виртуальные), имеет встроенный трекбол;
- Remote Node – с помощью данного элемента происходит соединение с роботом через канал связи (порт);
- Differential Drive – индикатор состояние дифференциального привода;
- Logging – создание файла-отчета (лог-файла);
- Articulated Arm – элемент взаимодействия с манипулятором представляет собой окно с выбором звена и полем для ввода угла поворота, а так же кнопки для подтверждения ввода параметров (Apply Changes) и соединения с манипулятором (Connect);
- Laser Range Finder – данный элемент реализует взаимодействие с лазерным дальномером.

6.2. Методические рекомендации

6.2.1. Методика организации супервизорного управления с использованием сервисов

1. Для организации супервизорного управления будут использованы сервисы DirectionDialog и HelloAppsConsole. Создайте две диаграммы: Main

(рис. 6.4) и Supervisory System. Первая диаграмма отвечает за управление роботом. В диаграмме Main содержатся 4 блока Activity: Forwards, Backwards, Right и Left представленные на рис. 6.5 – 6.8.

2. На вкладках Actions необходимо задать исходящие значения Distance и Power для блоков Forwards, Backwards и исходящие значения RotateDegree и Power для блоков Right и Left.

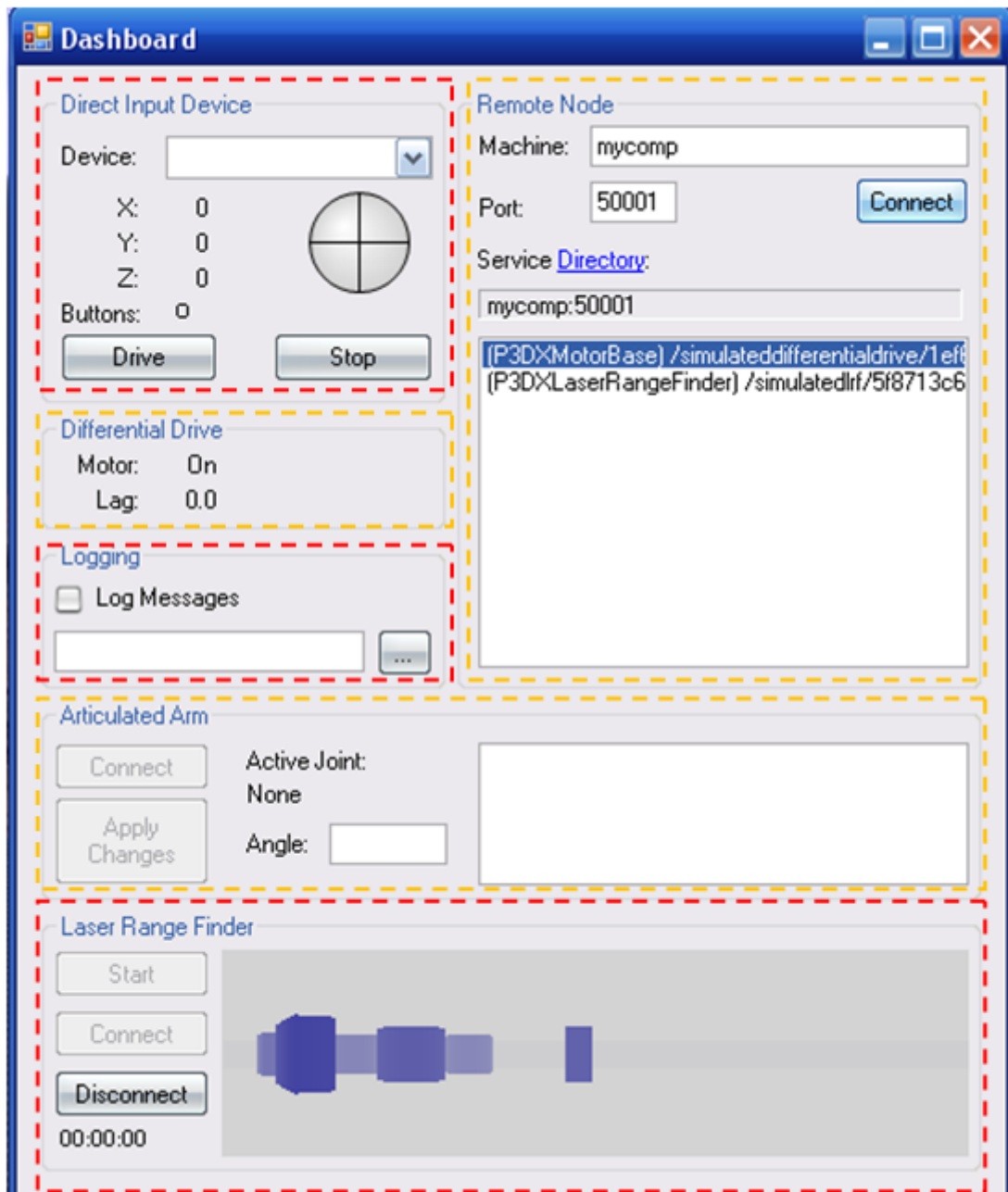


Рисунок 6.3 – Окно сервиса Simple Dashboard

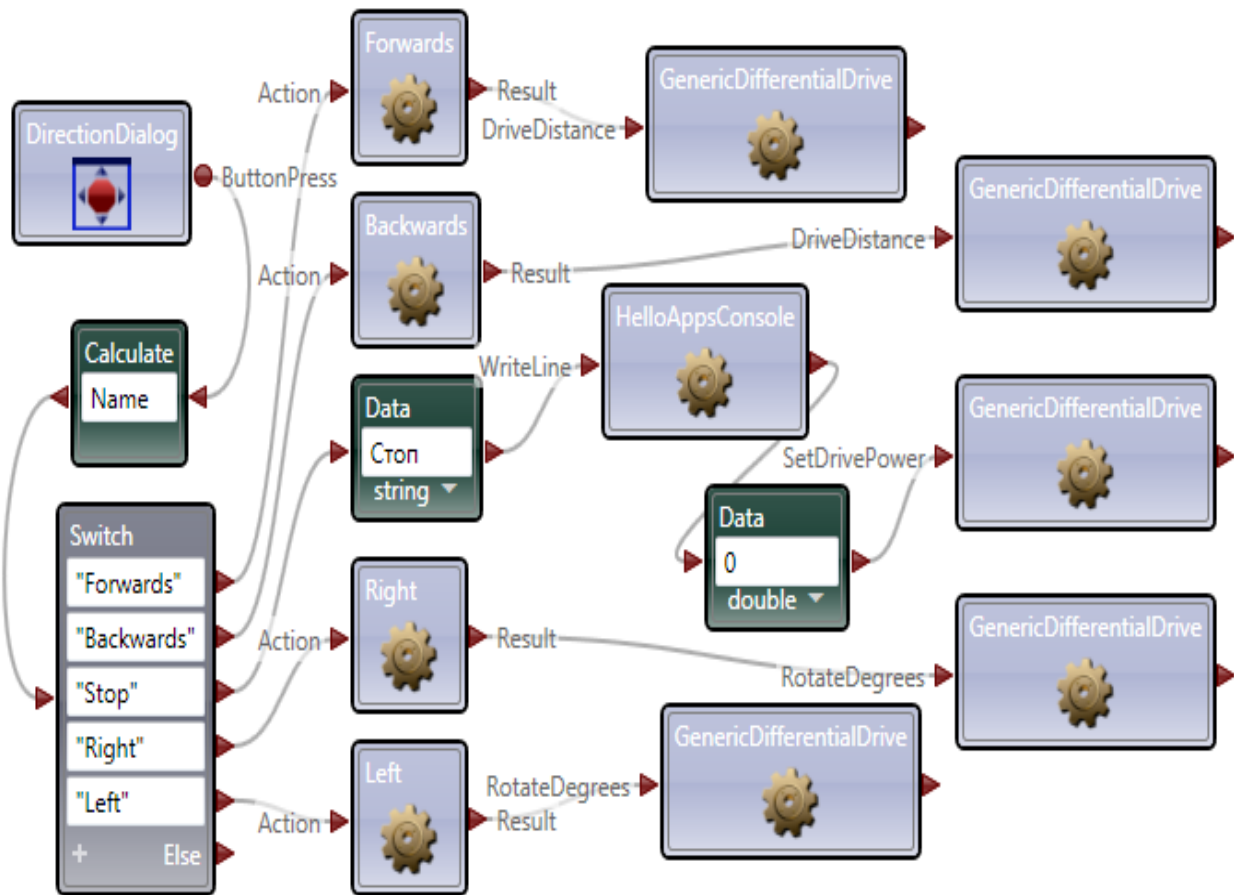


Рисунок 6.4 – Диаграмма Main

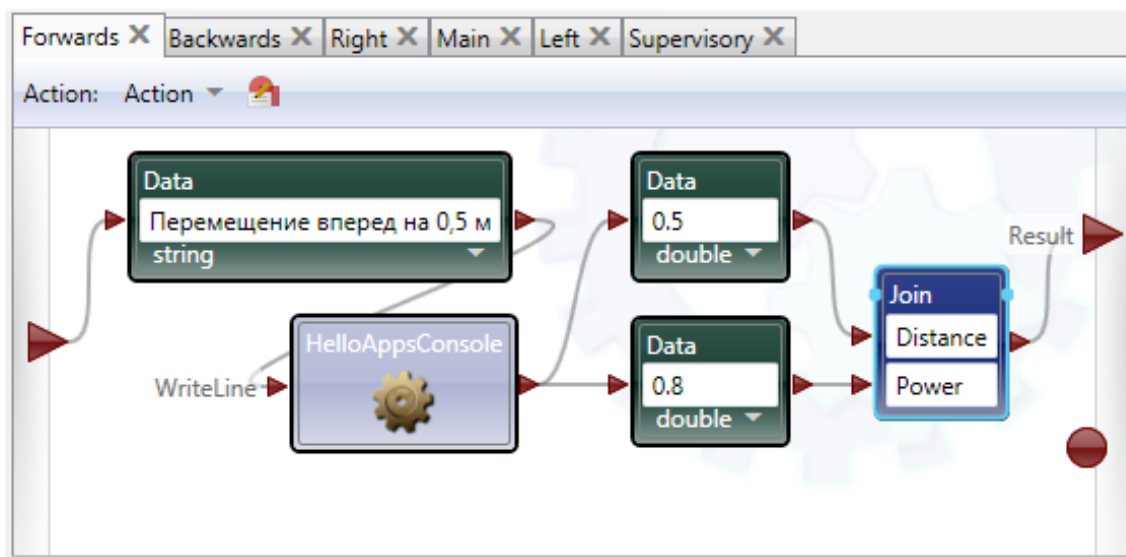


Рисунок 6.5 – Блок Activity «Forwards»

7. Вторая диаграмма Supervisory System (рис. 6.9) отвечает за систему оповещения оператора и является независимой, т. е. работает параллельно с диаграммой Main, ее удаление не повлияет на функционирование робота.

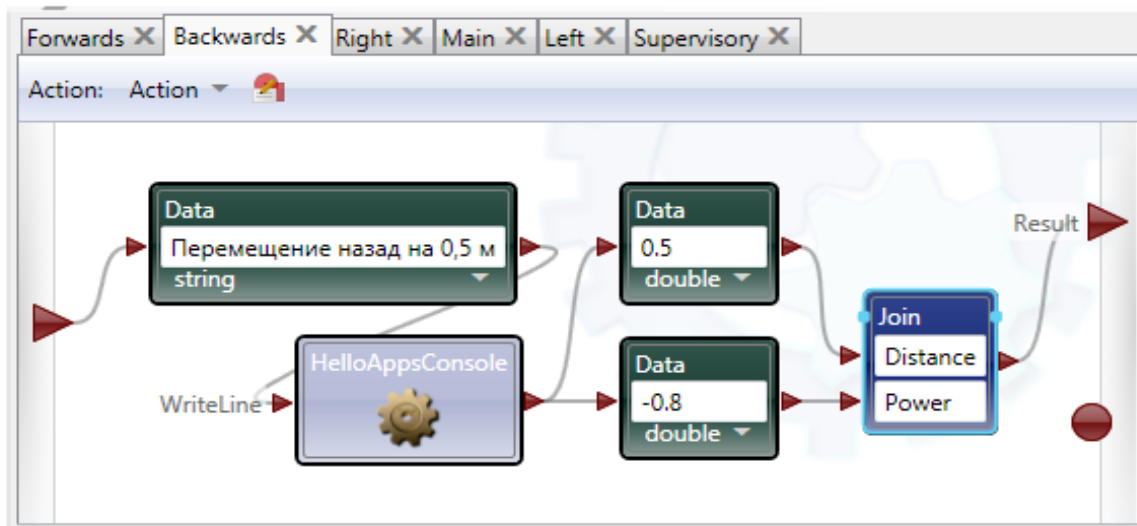


Рисунок 6.6 – Блок Activity «Backwards»

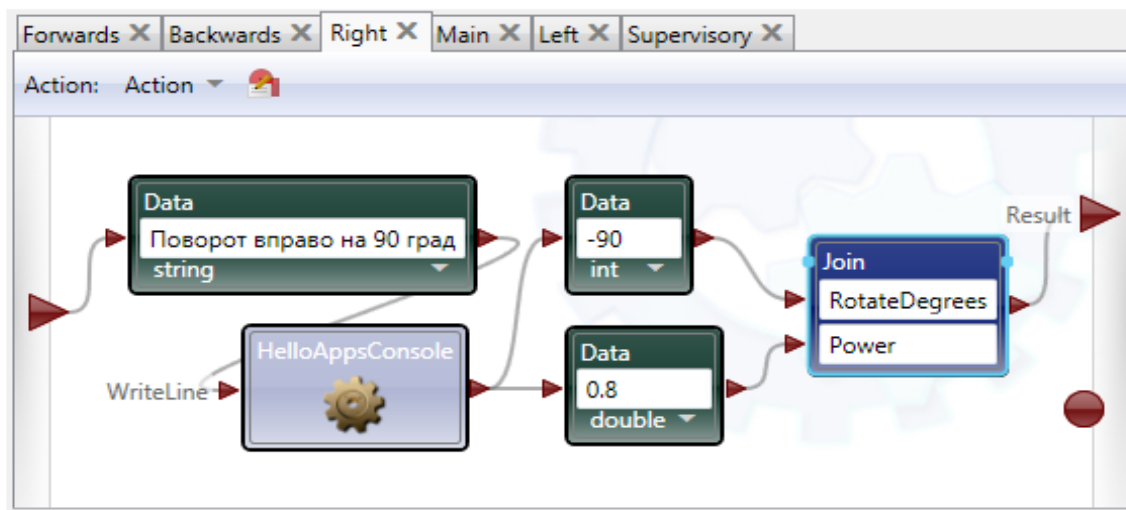


Рисунок 6.7 – Блок Activity «Right»

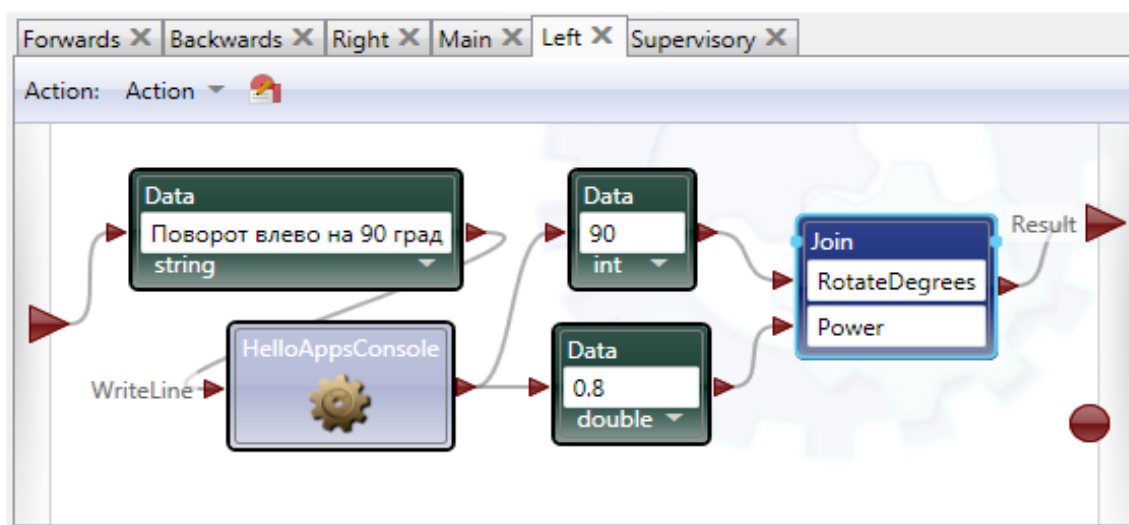


Рисунок 6.8 – Блок Activity «Left»

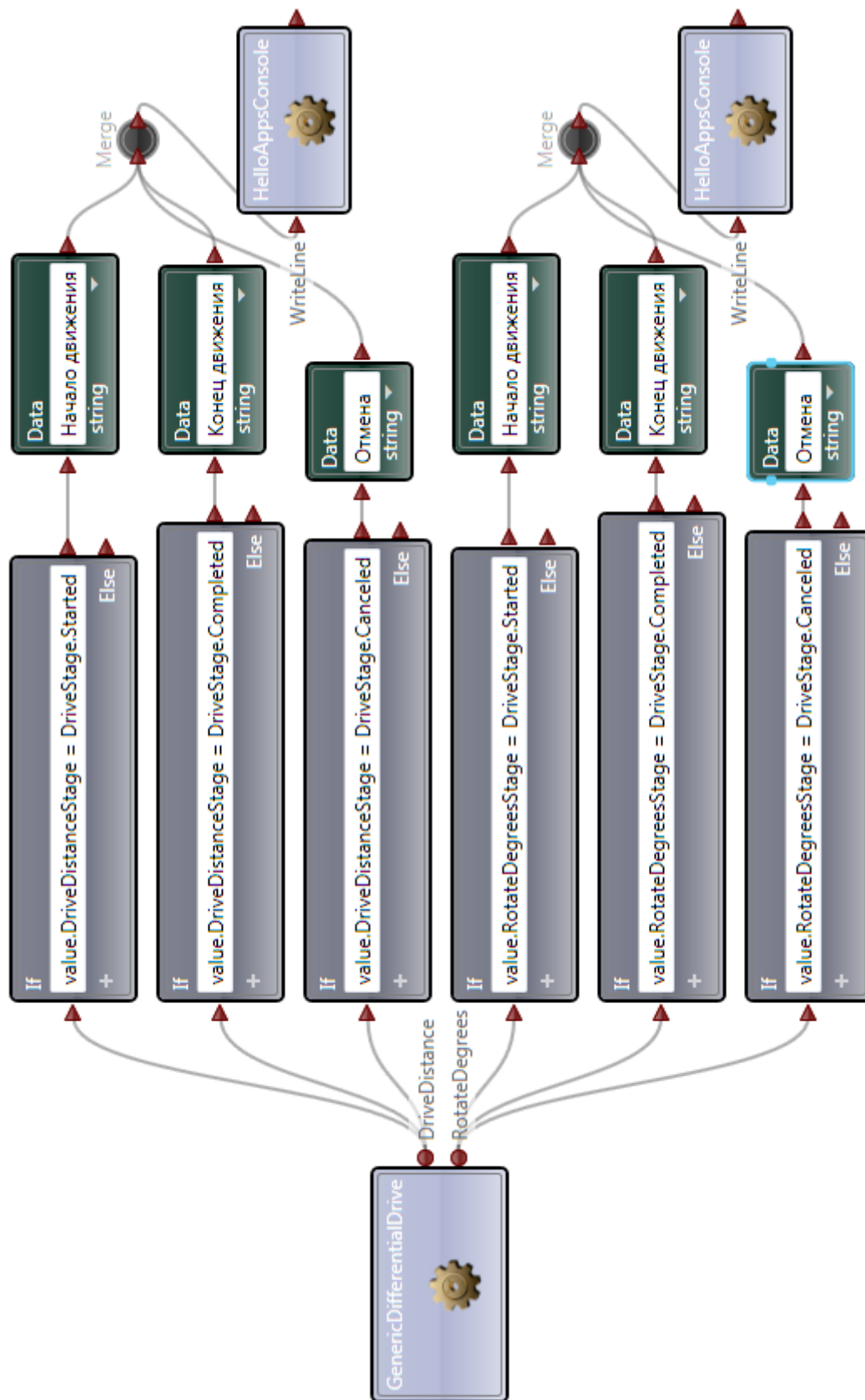


Рисунок 6.9 – Диаграмма Supervisory System

6.2.2. Методика организации супервизорного управления с помощью эмулятора пульта

1. Для организации супервизорного управления создайте диаграмму включающую сервис SimpleDashboard (рис. 6.10). Запустите программу на выполнение (рис. 6.11).

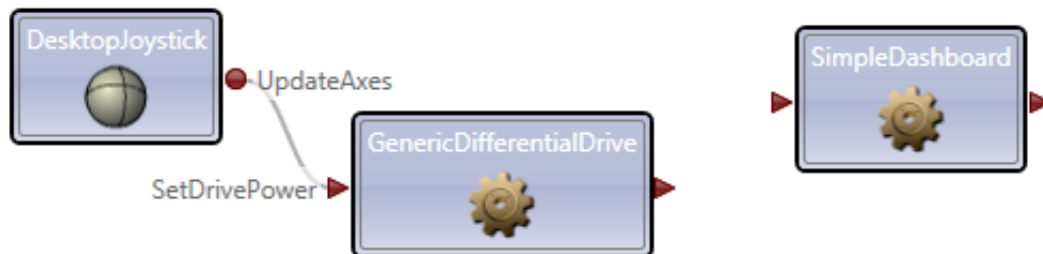


Рисунок 6.10 – Диаграмма супервизорного управления

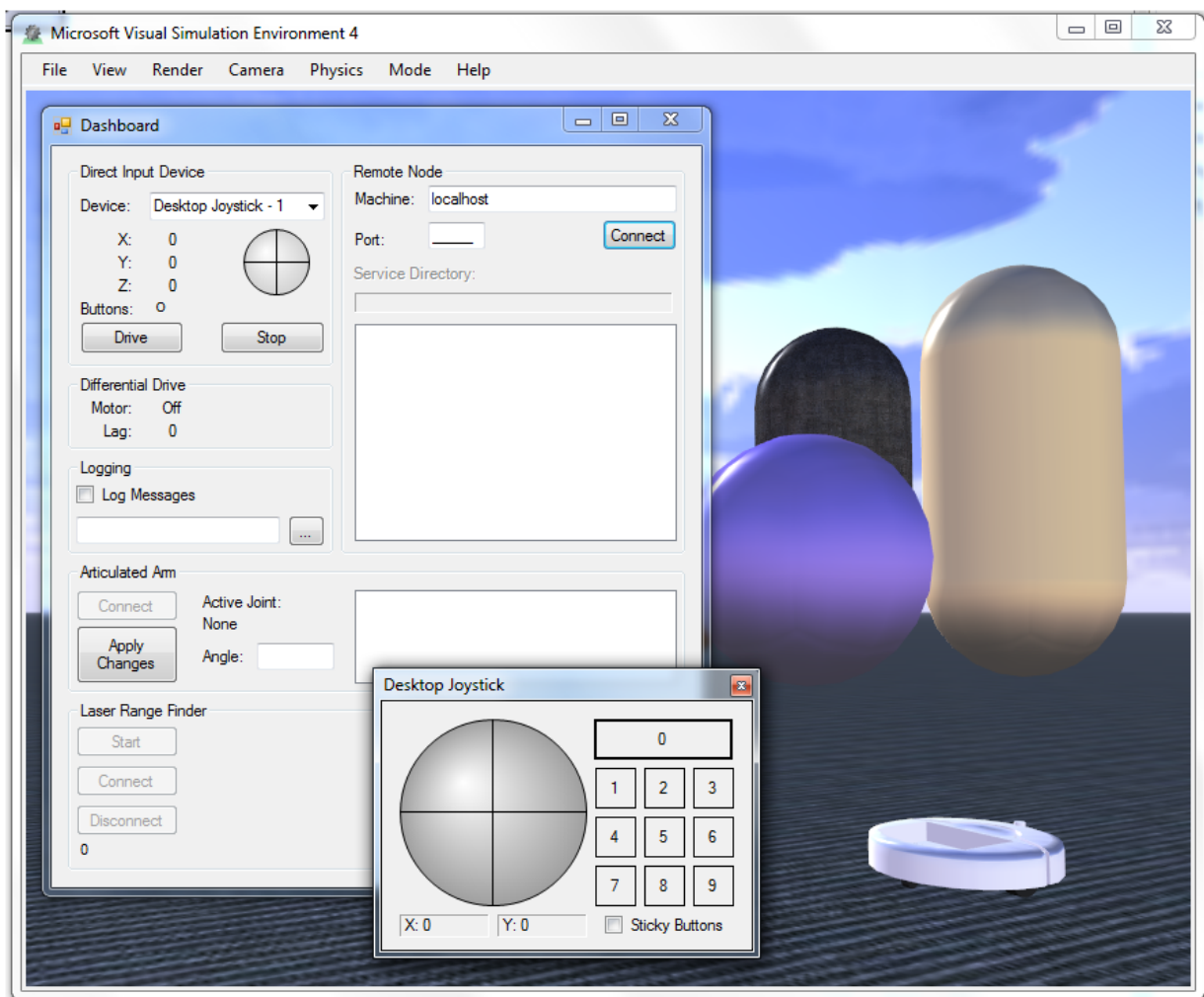


Рисунок 6.11 – Окно терминала и виртуального джойстика

2. В появившемся окне терминала нажмите кнопку Connect и из списка выберите RobotCreateMotorBase (в данном списке могут отображаться и другие устройства, например лазерный дальномер).

3. Активируйте привод нажатием на кнопку Drive. Управление роботом может осуществляться трекболом, встроенным в терминал или виртуальным джойстиком в окне сервиса Desktop Joystick.

4. Создайте новую программу как на рис. 6.12, реализуйте непрерывное управление по мощности. Выберите манифест для сервиса привода (рис. 6.13).

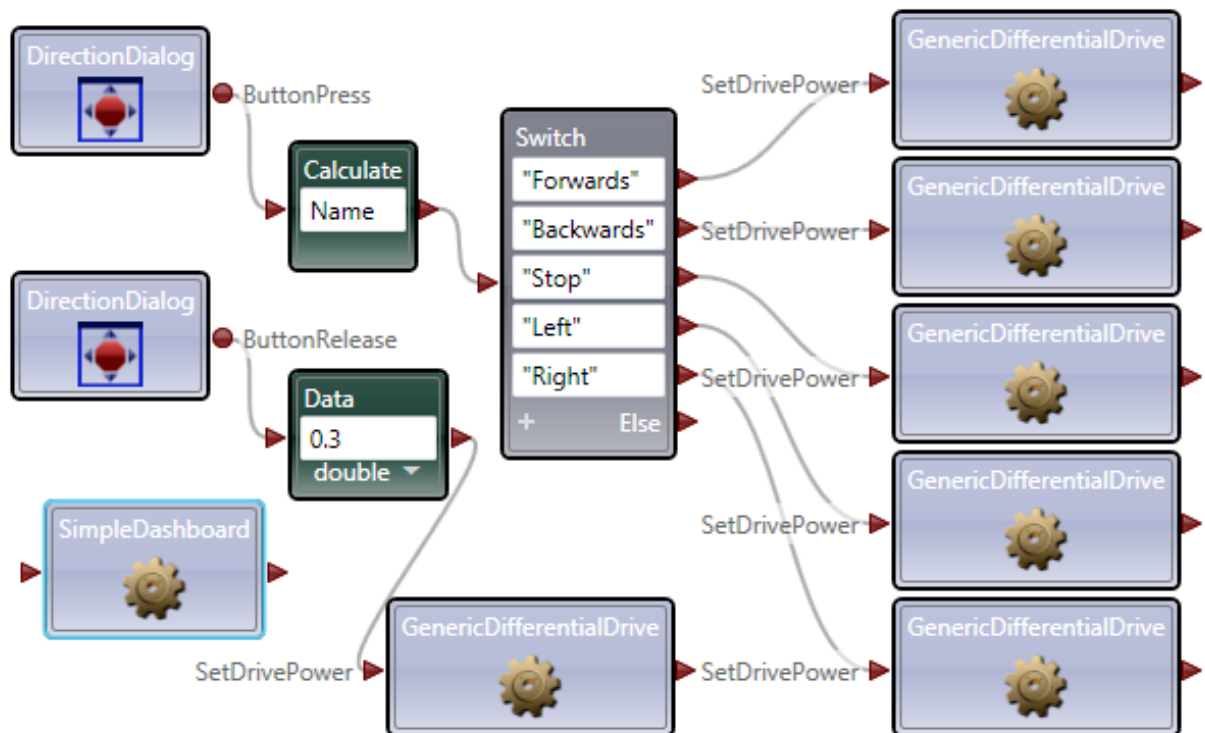


Рисунок 6.12 – Программа непрерывного управления по мощности

5. Запустите программу на выполнение. В появившемся окне терминала установите соединение с роботом, если необходимо используйте трекбол, предварительно активировав приводы. Подключите лазерный дальномер. В окне терминала будут отображаться контуры фигур, от которых отразился луч лазера (рис. 6.14).

6. Создайте новую программу (рис. 6.15), которая моделирует работу промышленного манипулятора KUKA (рис. 6.16).

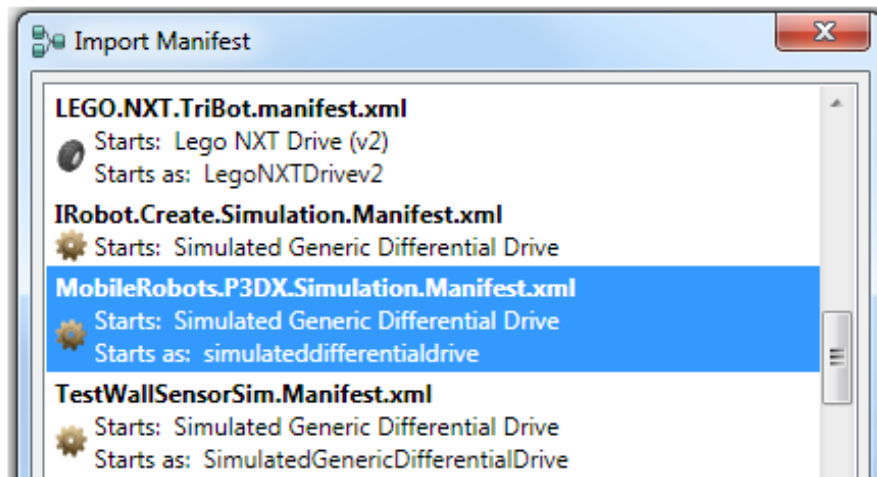


Рисунок 6.13 – Выбор манифеста для сервиса привода

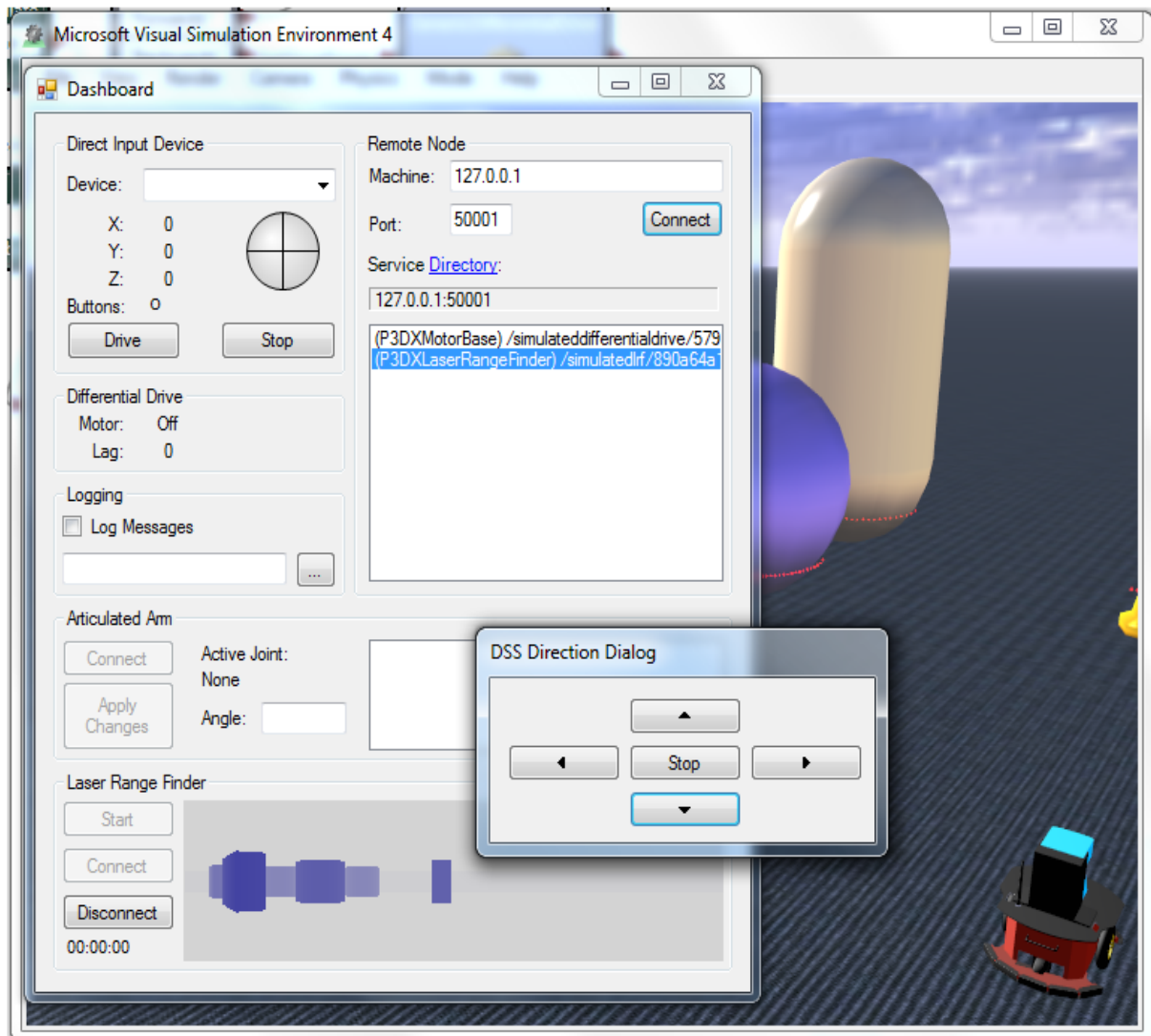


Рисунок 6.14 – Показание лазерного дальномера

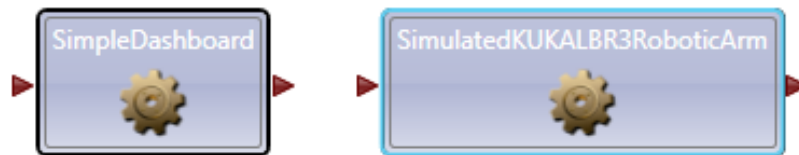


Рисунок 6.15 – Программа для управления манипулятором



Рисунок 6.16 – Внешний вид промышленного манипулятора KUKA (по материалам сайта <http://www.eurobots.net/>)

7. Выберите соответствующий манифест для сервиса манипулятора (рис. 6.17).

8. Запустите программу на выполнение. Для взаимодействия с манипулятором, в окне терминала выбирается необходимое звено (например, Joint1), задается угол поворота в окне Angle (например, -90). После нажатия

кнопки «Apply Changes», выбранное звено повернется на 90 градусов (рис. 6.18).

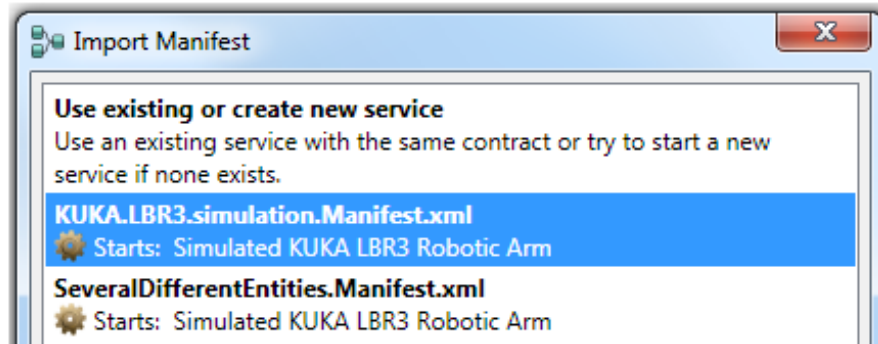


Рисунок 6.17 – Выбор манифеста

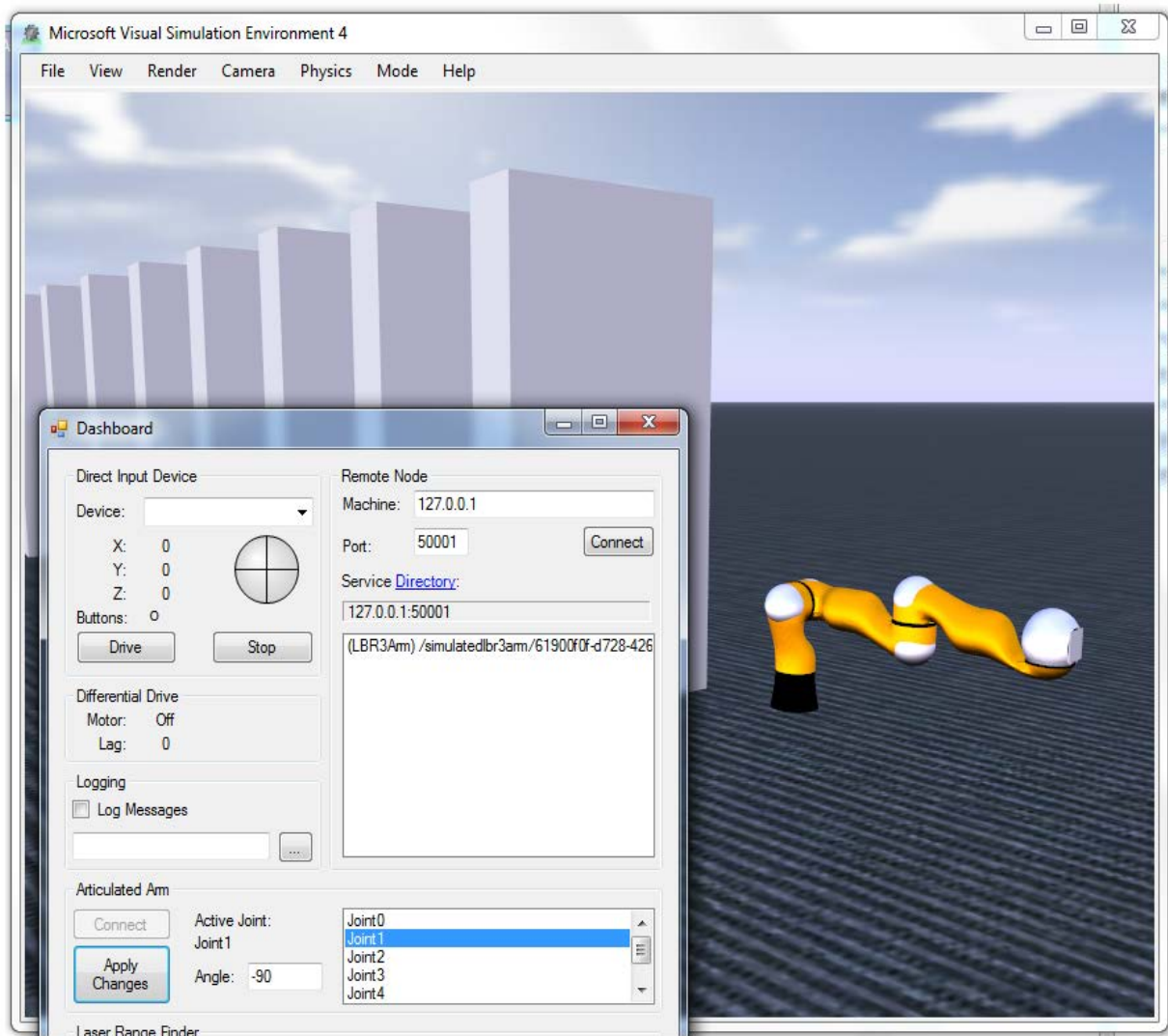


Рисунок 6.18 – Результат работы программы моделирования манипулятора

6.3. Лабораторная работа 6. Моделирование супервизорного и интерактивного управления

6.3.1. Цель и содержание

Целью лабораторной работы является приобретение студентами практических навыков организации супервизорного, интерактивного управления автономным подвижным объектом и манипулятором с использованием сервисов и с помощью эмулятора пульта.

Содержание

1. Организация супервизорного, интерактивного управления с использованием сервисов.
2. Организация супервизорного, интерактивного управления с помощью эмулятора пульта.

6.3.2. Аппаратура и материалы

Для изучения материалов этой лабораторной работы необходимо:

1. Компьютер, удовлетворяющий минимальным системным требованиям для установки Microsoft Robotics Developer Studio R4.
2. Установленная платформа RDS R4 с SPLX Launcher tool и SPL Service Pack.

6.3.3. Порядок выполнения работы

1. Изучение теоретического материала и методических рекомендаций.
2. Выполнение задания, предлагаемого в лабораторной работе.

Задание:

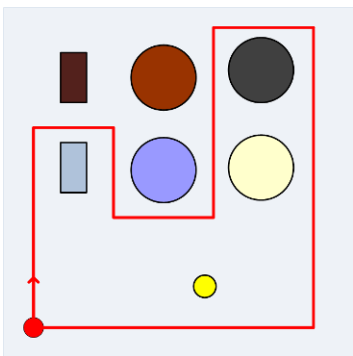
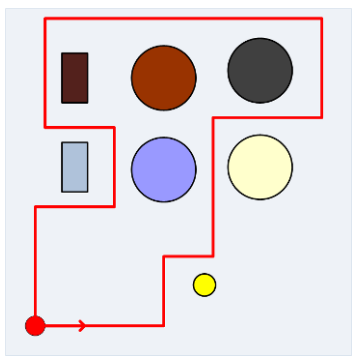
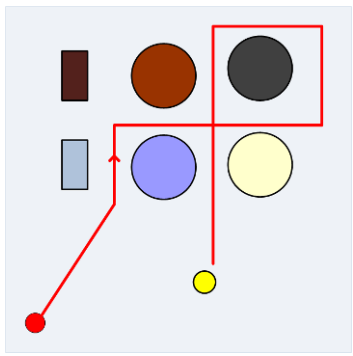
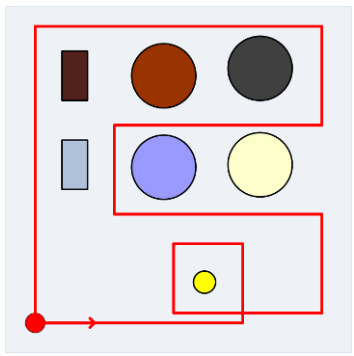
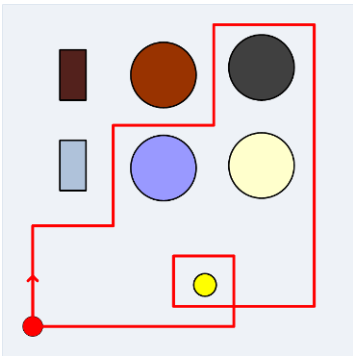
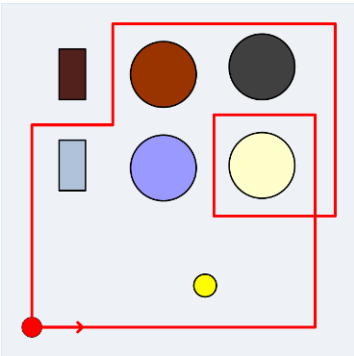
1. Организуйте управление мобильным роботом согласно методике 6.2.1. Скорректируйте параметры элементарных действий и прокомментируйте изменения.
2. Организуйте управление мобильным роботом согласно методике 6.2.2. Скорректируйте параметры работы сервиса DesktopJoystick и прокомментируйте изменения.

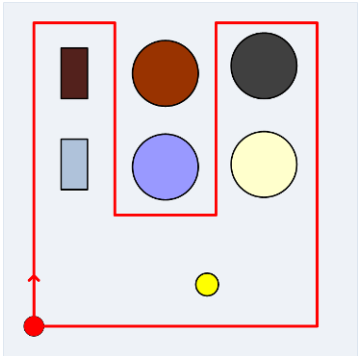
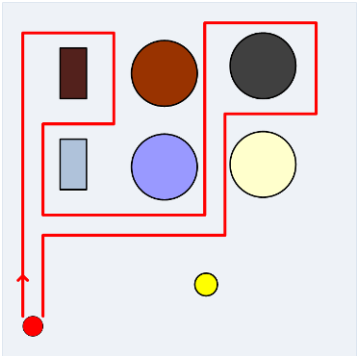
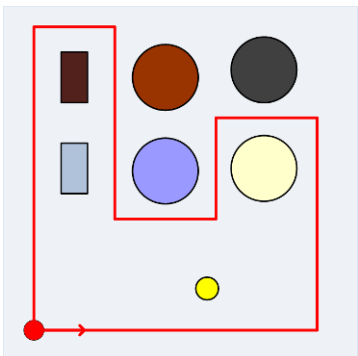
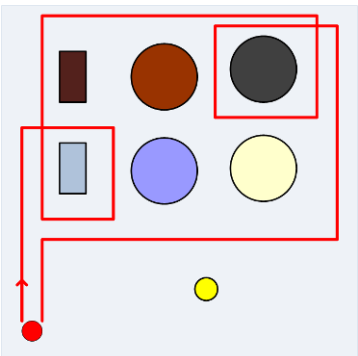
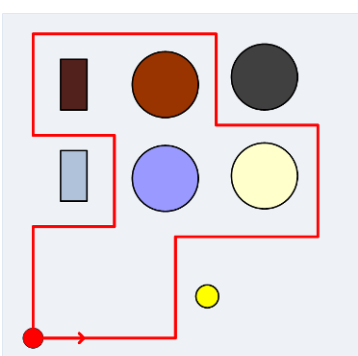
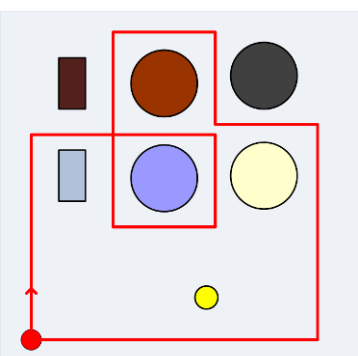
3. Организуйте непрерывное управление по мощности мобильным роботом согласно методике 6.2.2. Скорректируйте параметры элементарных действий и прокомментируйте изменения.

4. Организуйте управление манипулятором согласно методике 6.2.2. Скорректируйте параметры элементарных действий и прокомментируйте изменения.

5. Запрограммируйте движение мобильного робота по заданной траектории в существующей среде симуляции согласно варианту задания (табл. 6.1). Направление движения отмечено стрелкой.

Таблица 6.1. Варианты заданий для самостоятельного выполнения

№ варианта	Траектория движения	№ варианта	Траектория движения
1		2	
3		4	
5		6	

№ варианта	Траектория движения	№ варианта	Траектория движения
7		8	
9		10	
11		12	

6.4. Контрольные вопросы

1. Назовите принципиальные отличия организации супервизорного и интерактивного способов управления
2. Охарактеризуйте сервисы, используемые для реализации интерактивного управления.
3. Охарактеризуйте сервисы, используемые для реализации системы целеуказания.

4. Прокомментируйте особенности использования трекбола в сервисе SimpleDashboard.
5. Прокомментируйте особенности использования системы целеуказания в сервисе SimpleDashboard.
6. Назовите количество звеньев реализованных в модели манипулятора KUKA.

ГЛАВА 7. СОЗДАНИЕ МОДЕЛИ ВИРТУАЛЬНОЙ СРЕДЫ

Трёхмерная графика (3D Graphics) – раздел компьютерной графики, совокупность приемов и инструментов (как программных, так и аппаратных), предназначенных для изображения объёмных объектов [7]. Трёхмерное изображение на плоскости отличается от двумерного тем, что включает построение геометрической проекции трёхмерной модели **сцены** на плоскость (например, экран компьютера) с помощью специализированных программ.

7.1. Краткие теоретические сведения

Среда симуляции Visual Simulation Environment – это ключевая часть Microsoft Robotics Developer Studio R4, которая использует 3D графику, чтобы сформировать изображение виртуального мира и физический движок (механизм) для аппроксимации взаимодействия между объектами внутри физического мира.

Модель в среде VSE содержит один или несколько объектов (сущностей), отображаемых внутри кадра. Каждый кадр модели формируется динамически, и не всегда известно заранее, какие силы будут действовать на объект внутри каждого кадра. Благодаря этому, можно наблюдать, как робот будет взаимодействовать с окружающим миром.

Microsoft RDS предлагает несколько способов создания моделей и последующей работы с ними:

1. Использование редактора VSE.
2. Добавление объектов в модель путем создания проекта службы DSS.

Все компоненты в Microsoft Robotics Developer Studio представляют собой независимо исполняемые сервисы и выступают как основополагающий элемент RDS.

Microsoft Visual Studio – линейка продуктов компании Майкрософт, включающих интегрированную среду разработки программного обеспечения и ряд других инструментальных средств [10]. Данная платформа позволяет

разрабатывать консольные приложения и приложения с графическим интерфейсом, а также веб-сайты, веб-приложения, веб-службы в родном и управляемом кодах для всех платформ, поддерживаемых Microsoft Windows, Windows Mobile, Windows CE, .NET Framework, Xbox, Windows Phone .NET Compact Framework и Microsoft Silverlight [7].

Visual Studio включает в себя редактор исходного кода с поддержкой технологии IntelliSense и возможностью простейшего рефакторинга кода. Встроенный отладчик может работать как отладчик уровня исходного кода, так и как отладчик машинного уровня. Остальные встраиваемые инструменты включают в себя редактор форм для упрощения создания графического интерфейса приложения, веб-редактор, дизайнер классов и дизайнер схемы базы данных. Visual Studio позволяет создавать и подключать сторонние дополнения (плагины) для расширения функциональности.

Облегченная версия Microsoft Visual Studio – Visual Studio Express распространяется бесплатно. Она включает в себя небольшой набор инструментов, в отличие от полных версий: отсутствует дизайнер классов и многие другие инструменты, а также поддержка плагинов и удалённых баз данных в дизайнерах данных. Компиляторы в 64-битный код недоступны в Express редакциях до версий 2012 года (хотя компилятор бесплатно распространяется с Windows SDK и его можно использовать, компилировать автоматически из IDE нельзя).

На настоящий момент существуют следующие Express редакции:

- Visual Basic Express;
- Visual C++ Express;
- Visual C# Express;
- Visual Web Developer Express.

Среда RDS предоставляет шаблон Visual Studio, который можно использовать для создания новой службы DSS [1]. Создадим собственный сервис. Рассмотрим, как добавить к среде симуляции небо, землю и несколько других объектов симуляции.

7.2. Методические рекомендации

7.2.1. Методика создания сервиса в Visual Studio 2010

1. Создайте новый проект в среде Visual Studio 2010.

2. После установки RDS 4, в Visual Studio 2010 появился новый раздел «Microsoft Robotics», в котором есть шаблон «DSS Service», предназначенный для создания сервисов (рис. 7.1). В данном окне задайте имя сервиса (например, RTS_Lab7).

3. В окне конфигурирования нового DSS-сервиса введите IP хоста в Organization domain, а так же снимите галочки с Use subscription manager и Add document comments (рис. 7.2).

4. Перейдите на вкладку Partners. Здесь задаются дополнительные DSS-сервисы, связанные с нашим новым. Каждый раз при запуске нашего сервиса будут запускаться и его партнеры.

5. Введите в поле Service «Simulation Engine». Добавьте дополнительные сервисы, нажав кнопки «Add as partner» и OK (рис. 7.3).

Созданный шаблон DSS-сервиса содержит следующие файлы, представленные на рис. 7.4:

- **AssemblyInfo.cs** – этот файл определяет сборку как DSS-службу и содержит описание собираемой dll;
- **RTS_Lab7.cs** – основной исходный файл сервиса, содержит его реализацию (исполнение);
- **RTS_Lab7.manifest.xml** – это манифест, который будет использоваться для запуска сервиса RTS_Lab7. Среда Visual Studio запустит этот манифест при отладке сервиса (клавиша F5);
- **RTS_Lab7Types.cs** – этот файл содержит классы (или типы) используемые для взаимодействия со службой.

Рассмотрим подробнее два файла исходного кода шаблона DSS-сервиса, которые генерируются при создании проекта: RTSLab1.cs и RTSLab1Types.cs.

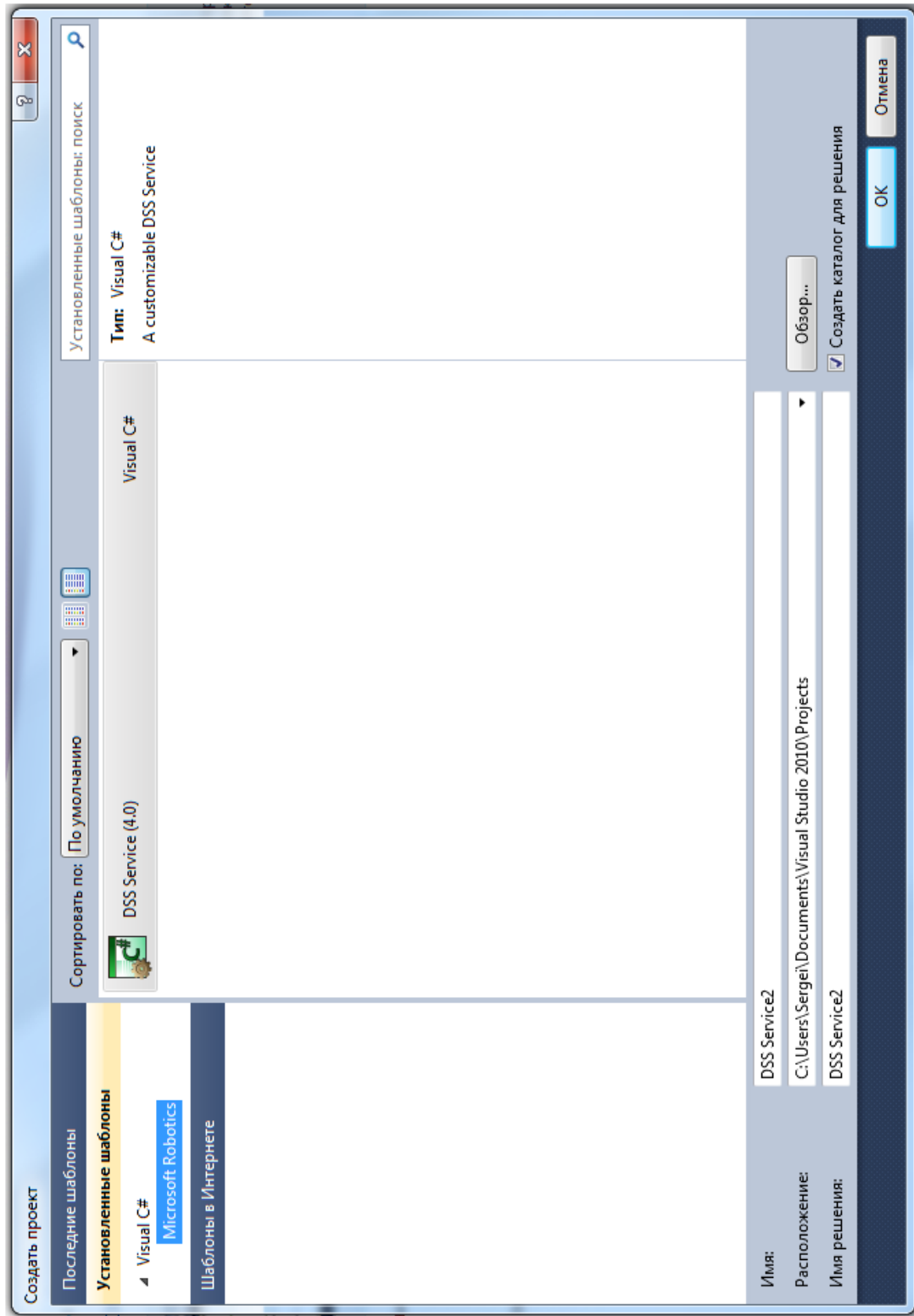


Рисунок 7.1 – Использование шаблона для создания сервиса в Visual Studio

The screenshot shows the 'New DSS Service' dialog box with the 'Service' tab selected. The 'Name' field contains 'RTS_Lab7' and the 'Organization domain (for contract identifier)' field contains '127.0.0.1'. The 'Namespace' field also contains 'RTS_Lab7'. The 'Year' is set to '2013' and the 'Month' is set to '12'. There are two unchecked checkboxes: 'Use subscription manager' and 'Add documentation comments'. At the bottom right are 'OK' and 'Cancel' buttons.

New DSS Service

Service Partners Alternate Contracts

Name: RTS_Lab7 Organization domain (for contract identifier): 127.0.0.1

Namespace: RTS_Lab7 Year: 2013 Month: 12

☐ Use subscription manager

☐ Add documentation comments

OK Cancel

Рисунок 7.2 – Окно конфигурирования нового DSS сервиса

The screenshot shows the 'New DSS Service' dialog box with the 'Partners' tab selected. On the left, a list of services includes 'Simulation Engine', 'Simulation Empty Project', and 'Simulation Engine' (which is highlighted). On the right, the 'Partner' dropdown is set to 'SimulationEngine (Simulation Engine)'. The 'Name' field contains 'SimulationEngine' and the 'Creation policy' dropdown is set to 'UseExistingOrCreate'. There is an unchecked checkbox for 'Add notification port' and a 'Delete this partner' button. At the bottom left is an 'Add as partner' button. At the bottom right are 'OK' and 'Cancel' buttons.

New DSS Service

Service Partners Alternate Contracts

Services: Simulation Engine Simulation Empty Project Simulation Engine

Partner: SimulationEngine (Simulation Engine)

Name: SimulationEngine

Creation policy: UseExistingOrCreate

☐ Add notification port

Delete this partner

Add as partner

OK Cancel

Рисунок 7.3 – Окно определения дополнительных сервисов

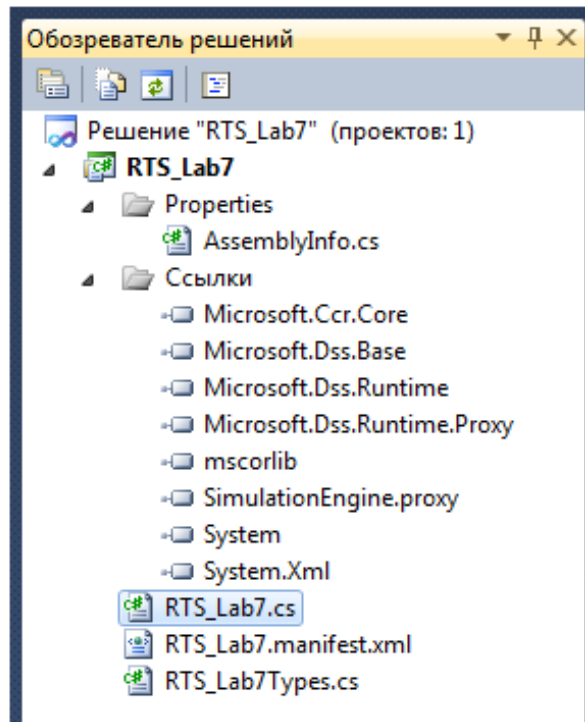


Рисунок 7.4 – Состав шаблона DSS сервиса

Файл типов – RTS_Lab7Types.cs

Начинается файл со строк:

```
namespace RTS_Lab7
{
    /// <summary>
    /// RTS_Lab7 contract class
    /// </summary>
    public sealed class Contract
    {
        /// <summary>
        /// DSS contract identifier for RTS_Lab7
        /// </summary>
        [DataMember]
        public const string Identifier =
http://127.0.0.1/2013/12/RTS\_Lab7.html";
    }
}
```

Пространство имен основано на имени службы (в нашем случае RTS_Lab7). Класс Contract является неотъемлемой частью контракта, и он должен содержать строковую переменную Identifier. Структура DSS не позволяет использовать обычное наследование, оно может осуществляться только через контракты.

Контракт (Contract identifier) – уникальный идентификатор, позволяющий однозначно отличать один сервис от другого. Он определяет, каким должно быть состояние сервиса, и какие операции можно с ним произвести. Контракт похож на **интерфейс** (interface) из объектно-ориентированного программирования. Концепция контракта проста: определяется состояние сервиса, набор типов сообщений и операционный порт. Контракты могут использоваться для различных моделей роботов, так как имеют общий интерфейс. Это позволяет различным приложениям работать с большим набором роботов. Приложению не известно, какому роботу передается сообщение, все они выглядят одинаково с точки зрения API.

Далее идет состояние службы с атрибутом [DataContract()]. Он необходим для того, чтобы класс был скопирован в Проху DLL при компиляции.

```
[DataContract]
public class RTS_Lab7State
{
}
```

Следующий фрагмент кода описывает основной порт операций (PortSet) с обязательным атрибутом [ServicePort].

```
[ServicePort]
public class RTS_Lab7Operations : PortSet<DsspDefaultLookup,
DsspDefaultDrop, Get, Subscribe>
{
}
```

Оканчивается файл определением типов запросов операций.

```
public class Get : Get<GetRequestType, PortSet<RTS_Lab7State, Fault>>
{
    public Get()
    {
    }

    public Get(GetRequestType body)
        : base(body)
    {
    }

    public Get(GetRequestType body, PortSet<RTS_Lab7State, Fault>
responsePort)
        : base(body, responsePort)
    {
    }
}
```

Стоит обратить внимание, что Get использует два обобщенных типа: GetrequestType для типов запросов и PortSet<ServiceExState, Fault> для ответов.

Основной файл реализации службы – RTS_Lab7.cs

Основной файл реализации службы содержит инициализацию и операции службы (или поведения).

```
[Contract(Contract.Identifier)]
[DisplayName("RTS_Lab7")]
[Description("RTS_Lab7 service (no description provided)")]
```

Назначение всех трех атрибутов понятно из названия: отображаемое имя (DisplayName), описание (Description), контракт (Contract).

Каждая служба должна обладать экземпляром своего состояния (даже если оно пустое) и основным портом операций.

```
[ServiceState]
RTS_Lab7State _state = new RTS_Lab7State();
/// <summary>
/// Main service port
/// </summary>
[ServicePort("/RTS_Lab7", AllowMultipleInstances = true)]
RTS_Lab7Operations _mainPort = new RTS_Lab7Operations();
```

Рекомендуется использовать общепринятые имена для этих переменных: _state, _mainPort.

Далее идет связывание нашего сервиса. Эту операцию можно задавать и вручную, так что добавить другие сервисы в уже работающий проект не составит труда.

```
[Partner("SimulationEngine", Contract = engine.Contract.Identifier,
CreationPolicy = PartnerCreationPolicy.UseExistingOrCreate)]
engine.SimulationEnginePort _simulationEnginePort = new
engine.SimulationEnginePort();
```

Затем идут конструктор класса и метод Start, который инициализирует службу при его запуске.

```
Public RTS_Lab7Service(DsspServiceCreationPort creationPort)
: base(creationPort)
{
}
```

```
protected override void Start()
{
    base.Start();
}
```

7.2.2. Методика создания среды симуляции

Для создания среды симуляции необходимо изменить созданный по предыдущей методике шаблон, добавив в него библиотеки [28, 36]:

- **Microsoft.Xna.Framework** – набор инструментов с управляемой средой времени выполнения (.NET) облегчающий разработку и управление компьютерными играми; данный фреймворк входит в состав интегрированной среды разработки XNA Game Studio;

- **PhysicsEngine** – предоставляет доступ к программе моделирования физических взаимодействий AGEIA;

- **RoboticsCommon** – предоставляет доступ к пространству имен PhysicalModel, которое используется для задания физических параметров роботов;

- **SimulationCommon** – предоставляет доступ к определению типов, используемых при работе с моделями и программами обработки физических взаимодействий;

- **SimulationEngine** – предоставляет доступ к обработчику модели;

- **SimulationEngine.proxy** – представляет собой прокси для обработчика модели, который используется при загрузке обработчика модели в качестве партнера.

1. Используйте пункт Добавить ссылку (Add Reference) (рис. 7.5).

2. На вкладке .NET найдите библиотеку Microsoft.Xna.Framework (рис. 7.6).

3. На вкладке Обзор (Browse) найдите остальные библиотеки в папке C:\Users\%username%\Microsoft Robotics Dev Studio 4\bin. В случае отсутствия, какой либо библиотеки необходимо установить соответствующие приложения (Microsoft XNA, Microsoft RDS, SPL).

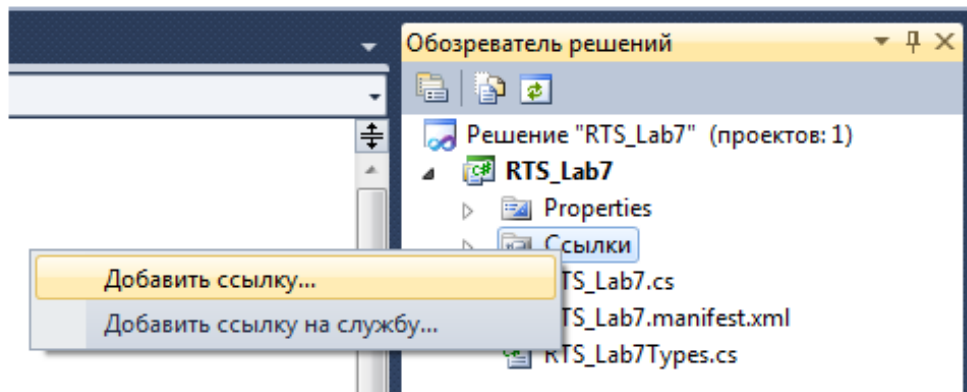


Рисунок 7.5 – Окно добавления новых DLL-библиотек

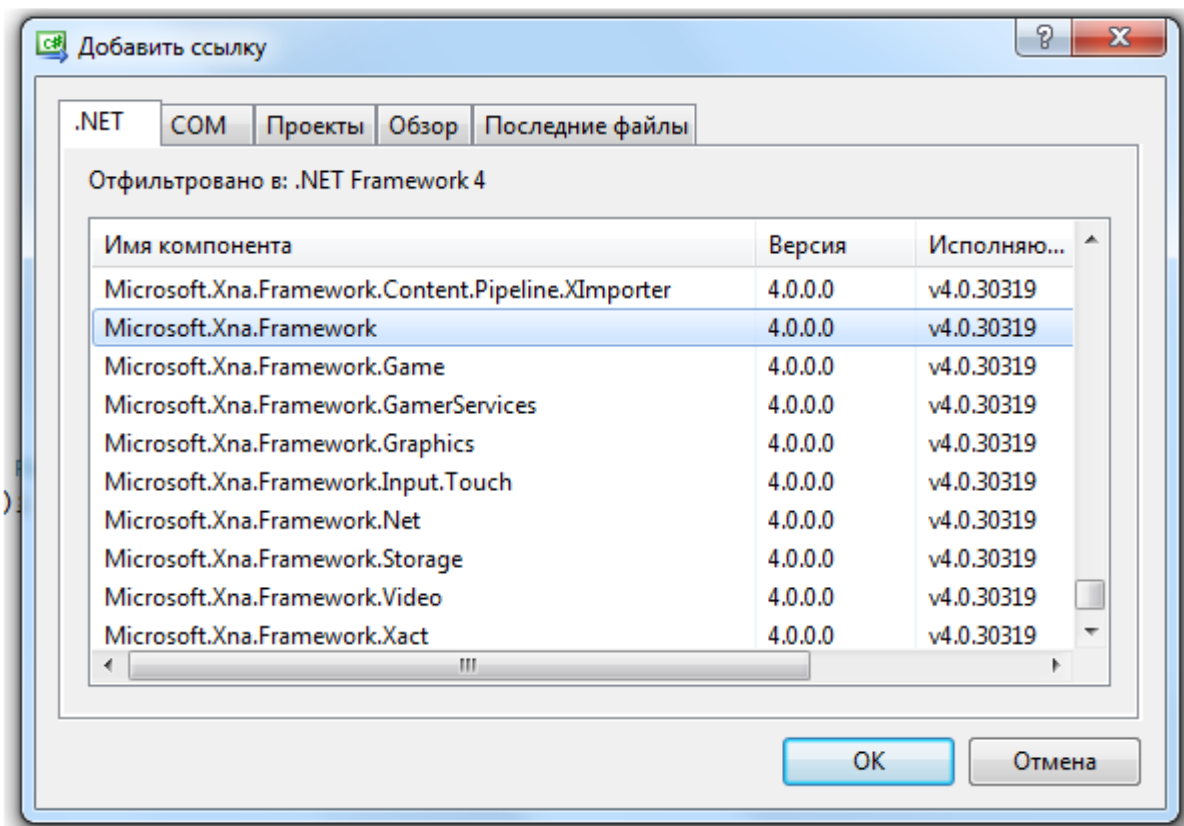


Рисунок 7.6 – Окно добавления библиотеки Microsoft.Xna.Framework

4. После подключения библиотек, требуется описать их в заголовке. Для этого добавьте:

```
#region Additional Namespaces
using xna = Microsoft.Xna.Framework;
using Microsoft.Robotics.Simulation;
using Microsoft.Robotics.Simulation.Physics;
using Microsoft.Robotics.Simulation.Engine;
using Microsoft.Robotics.PhysicalModel;
#endregion
```

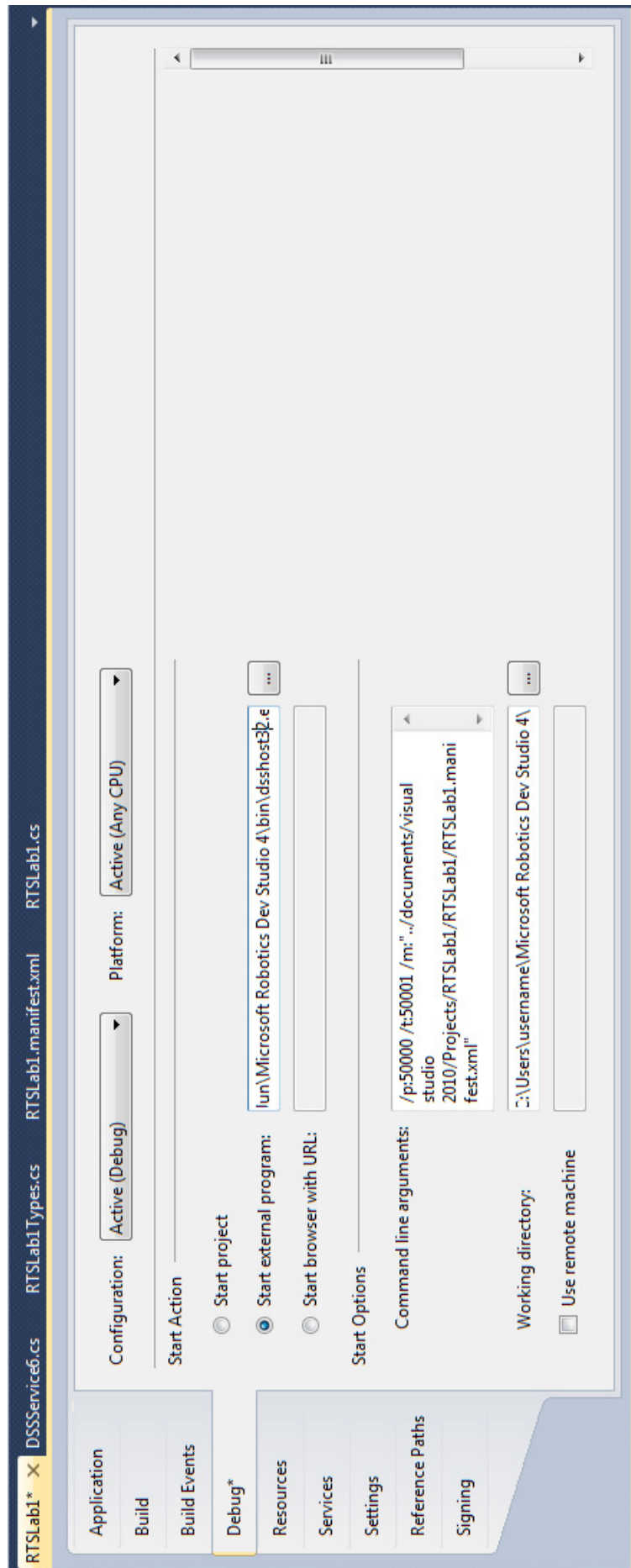


Рисунок 7.7 – При работе в win64 следует использовать dsshost32

6. Запустите на выполнение проект. Должно появиться абсолютно черное окно симуляции.

При работе в x64 битной системе и наличии Visual Studio Professional, следует сменить dsshost.exe на dsshost32.exe. Для этого надо зайти в настройки проекта, выбрать вкладку Debug и изменить значение в поле Start external program (рис. 7.7).

7. Создайте декорации, которые будут заполнять пространство симулируемой среды. В функцию Start, после вызова `base.Start();` добавьте новую функцию.

```
protected override void Start()
{
    base.Start();
    Populate();
}
```

8. Запишите реализацию функции ниже:

```
private void Populate()
{
    // Установим начальную камеру
    CameraView view = new CameraView();
    view.EyePosition = new Vector3(-6.78f, 14.27f, 6.79f);
    view.LookAtPoint = new Vector3(-5.8f, 12.15f, 6.8f);
    SimulationEngine.GlobalInstancePort.Update(view);

    // Еще одна камера сверху
    CameraEntity fromAbove = new CameraEntity();
    fromAbove.State.Name = "FromAbove";
    fromAbove.Location = new xna.Vector3(-2.12f, 9.27f, -0.45f);
    fromAbove.LookAt = new xna.Vector3(-2.1f, -1.78f, -1.45f);
    fromAbove.IsRealTimeCamera = false;
    SimulationEngine.GlobalInstancePort.Insert(fromAbove);

    // Добавляем небо
    SkyDomeEntity sky = new SkyDomeEntity("skydome.dds",
"sky_diff.dds");
    SimulationEngine.GlobalInstancePort.Insert(sky);

    // Добавляем солнце
    LightSourceEntity sun = new LightSourceEntity();
    sun.State.Name = "Sun";
    sun.Type = LightSourceEntityType.Directional;
    sun.Color = new Vector4(1, 1, 1, 1);
    sun.Direction = new Vector3(-0.47f, -0.8f, -0.36f);
    SimulationEngine.GlobalInstancePort.Insert(sun);
}
```

```

        HeightFieldShapeProperties hf = new
HeightFieldShapeProperties("height field",
        64, // количество строк
        10, // дистанция в метрах между строками
        64, // количество колонок
        10, // дистанция в метрах между колонками
        1, // коэффициент масштабирования
        -1000); // "глубина" пола. Должна быть равна большому
отрицательному значению

        // Создаем массив из элементов пола
        hf.HeightSamples = new HeightFieldSample[hf.RowCount *
hf.ColumnCount];
        for (int i = 0; i < hf.RowCount * hf.ColumnCount; i++)
        {
            hf.HeightSamples[i] = new HeightFieldSample();
            hf.HeightSamples[i].Height = (short)(Math.Sin(i * 0.01));
        }

        // Создаем материал для всего массива, его так же можно
задавать и для каждого элемента
        hf.Material = new MaterialProperties("ground", 0.8f, 0.5f,
0.8f);

        // Добавляем пол в сцену и задаем текстуру
        SimulationEngine.GlobalInstancePort.Insert(new
HeightFieldEntity(hf, "WoodFloor.dds"));

        // Рисуем стены
        Wall[] Walls = new Wall[]
        {
            new Wall("n1", new Vector3(2f, 0, 0), new Vector3(4f,
0.8f, 0.1f), Quaternion.FromAxisAngle(0, 1, 0, (float)(Math.PI / 2))),
            new Wall("n2", new Vector3(0f, 0, -2.05f), new
Vector3(4f, 0.8f, 0.1f), new Quaternion(0,0,0,1)),
            new Wall("n3", new Vector3(0f, 0, 2), new Vector3(4f,
0.8f, 0.1f), new Quaternion(0,0,0,1)),
            new Wall("c1", new Vector3(0f, 0, 3.0f), new Vector3(4f,
0.8f, 0.1f), new Quaternion(0,0,0,1)),
            new Wall("c2", new Vector3(2f, 0, 5f), new Vector3(4f,
0.8f, 0.1f), Quaternion.FromAxisAngle(0, 1, 0, (float)(Math.PI / 2))),
            new Wall("c3", new Vector3(-2f, 0, 5f), new Vector3(4f,
0.8f, 0.1f), Quaternion.FromAxisAngle(0, 1, 0, (float)(Math.PI / 2))),
            new Wall("f1", new Vector3(1.947f, 0f, 9.228f), new
Vector3(2f, 0.8f, 0.1f), Quaternion.FromAxisAngle(0, 1, 0,
(float)(Math.PI / 2))),
            new Wall("f2", new Vector3(0.326f, 0f, 9.221f), new
Vector3(2f, 0.8f, 0.1f), Quaternion.FromAxisAngle(0, 1, 0,
(float)(Math.PI / 2))),

```

```

        new Wall("f3", new Vector3(-0.045f, -0.07f, 8.195f), new
Vector3(4f, 0.8f, 0.1f), new Quaternion(0,0,0,1)),
        new Wall("u1", new Vector3(0.04f, 0f, 11.108f), new
Vector3(4f, 0.8f, 0.1f), Quaternion.FromAxisAngle(0, 0, 0, 1)),
        new Wall("u2", new Vector3(-1.959f, 0f, 13.064f), new
Vector3(4f, 0.8f, 0.1f), Quaternion.FromAxisAngle(0, 1, 0,
(float)(Math.PI / 2))),
        new Wall("u3", new Vector3(0.04f, 0f, 15.108f), new
Vector3(4f, 0.8f, 0.1f), Quaternion.FromAxisAngle(0, 0, 0, 1)),

};

foreach (Wall w in Walls)
{
    SingleShapeEntity wall =
        new SingleShapeEntity(
            new BoxShape(
                new BoxShapeProperties(
                    0, //у стен отсутствует масса (не
интерактивные объекты)
                    new Pose(),
                    w.Dimensions)), //размеры
                    w.Position);
    wall.State.Pose.Orientation = w.Orientation;
    wall.State.Name = w.Name;
    wall.State.Assets.DefaultTexture = "MayangConcrP.dds";
    SimulationEngine.GlobalInstancePort.Insert(wall);
}
}
}
public struct Wall
{
    public string Name;
    public Vector3 Position;
    public Vector3 Dimensions;
    public Quaternion Orientation;
    public Wall(string name, Vector3 position, Vector3 dimensions,
Quaternion orientation)
    {
        Name = name;
        Position = position;
        Dimensions = dimensions;
        Orientation = orientation;
    }
}
}

```

9. Запустите проект. На экране вы увидите надпись «псфу» собранную из стен (рис. 7.8).

7.3. Лабораторная работа 7. Основы работы в среде симуляции Visual Simulation Environment

7.3.1. Цель и содержание

Целью лабораторной работы является приобретение студентами практических навыков работы в среде симуляции VSE Microsoft RDS с использованием Microsoft Visual Studio.

Содержание

1. Создание нового сервиса на языке C# в Visual Studio 2010.
2. Связывание созданного сервиса с сервисом движка среды симуляции.
3. Создание среды симуляции.

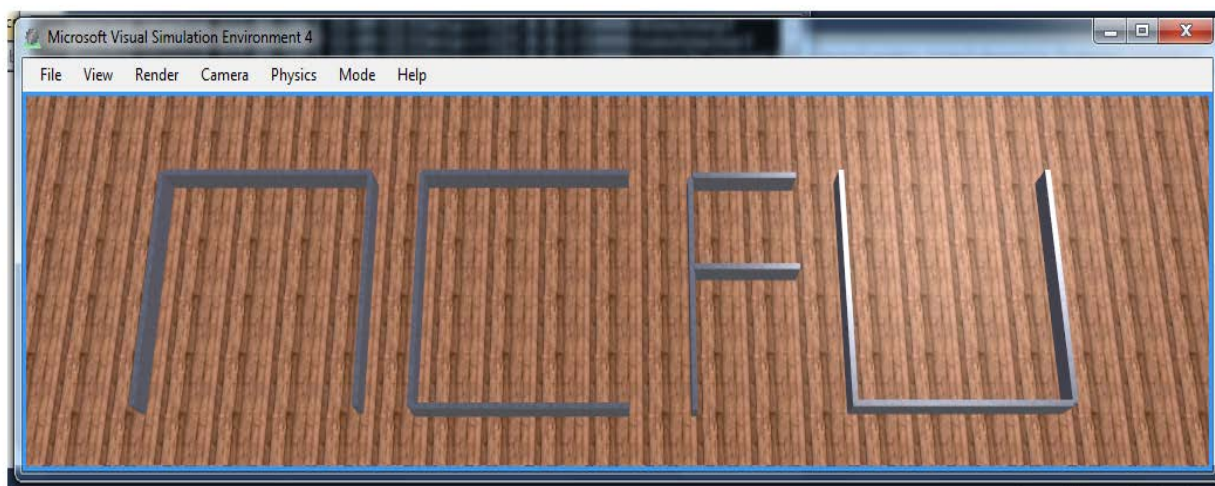


Рисунок 7.8 – Результат работы программы

7.3.2. Аппаратура и материалы

Для изучения материалов этой лабораторной работы необходимо:

1. Компьютер, удовлетворяющий минимальным системным требованиям для установки Microsoft Robotics Developer Studio R4.
2. Установленная платформа RDS R4.
3. Установленная среда XNA Game Studio 4.
4. Установленная интегрированная среда разработки программного обеспечения Visual Studio 2010 Express.

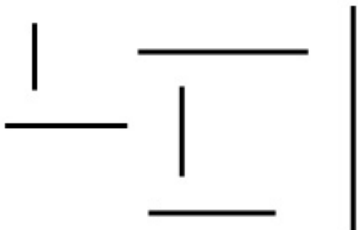
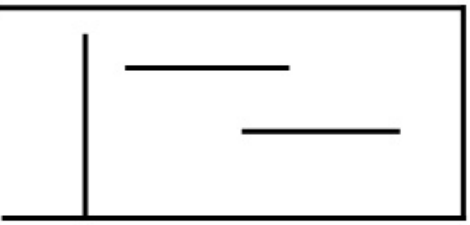
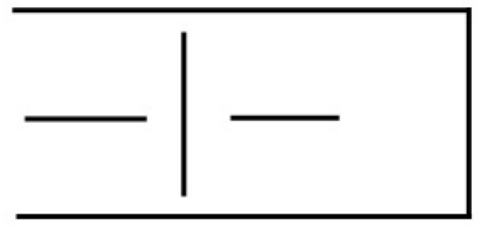
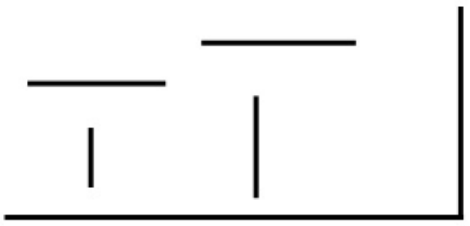
7.3.3. Порядок выполнения работы

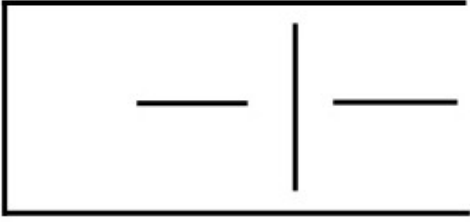
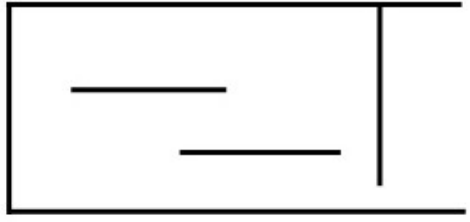
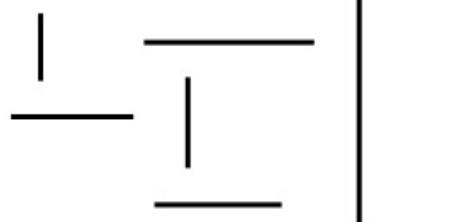
1. Изучение теоретического материала и методических рекомендаций.
2. Выполнение задания, предлагаемого в лабораторной работе.

Задание:

1. Создайте новый сервис в среде Visual Studio 2010 Express согласно методике 7.2.1. Назовите его согласно вашей фамилии (например, ivanov.cs).
2. Создайте модель виртуальной среды согласно методике 7.2.2. Измените положение стен в сцене согласно варианту задания (табл. 7.1).

Таблица 7.1. Варианты заданий для самостоятельного выполнения

№ вар.	Положение стен в сцене	№ вар.	Положение стен в сцене
1.		2.	
3.		4.	
5.		6.	
7.		8.	

№ вар.	Положение стен в сцене	№ вар.	Положение стен в сцене
9.		10.	
11.		12.	

7.4. Контрольные вопросы

1. Перечислите и охарактеризуйте тип файлов, входящих в шаблон нового DSS-сервиса.
2. Какие библиотеки требуется добавить в шаблон нового DSS-сервиса для работы со средой симуляции?
3. Для чего предназначена среда XNA Game Studio?
4. Прокомментируйте особенности добавления объекта в сцену.
5. Как задаются параметры стены в сцене?

ГЛАВА 8. УПРАВЛЕНИЕ СРЕДОЙ ВИРТУАЛИЗАЦИИ

Моделирование технических систем представляет собой сложную и объемную задачу. Во многих случаях требуется принимать решения в течение заданного короткого промежутка времени (режим реального времени). Новые подходы к управлению могут быть получены с использованием методов технологии «виртуальная реальность».

3D моделирование – это процесс создания трехмерной модели объекта. Задача 3D моделирования – разработать визуальный объемный образ желаемого объекта. С помощью трехмерной графики можно и создать точную копию конкретного предмета, и разработать новое, даже нереальное представление до сего момента, не существовавшего объекта [7].

Виртуальная реальность – созданный техническими средствами мир, передаваемый человеку через его ощущения: зрение, слух, обоняние, осязание и другие [7]. Виртуальная реальность имитирует как воздействие, так и реакции на воздействие.

8.1. Краткие теоретические сведения

Для получения трёхмерного изображения на плоскости требуется выполнить следующие шаги (рис. 8.1) [7]:

- моделирование – создание трёхмерной математической модели сцены и объектов в ней;
- текстурирование – назначение поверхностям моделей растровых или процедурных текстур, а также настройка свойств материалов (прозрачность, отражения, шероховатость и пр.);
- освещение – установка и настройка источников света;
- динамическая симуляция – автоматический расчёт взаимодействия частиц, твёрдых/мягких тел и пр. с моделируемыми силами гравитации, ветра, выталкивания и др., а также друг с другом;

- анимация – придание движения объектам;
- рендеринг (визуализация) – построение проекции в соответствии с выбранной физической моделью;
- отображение полученного изображения на устройстве вывода.

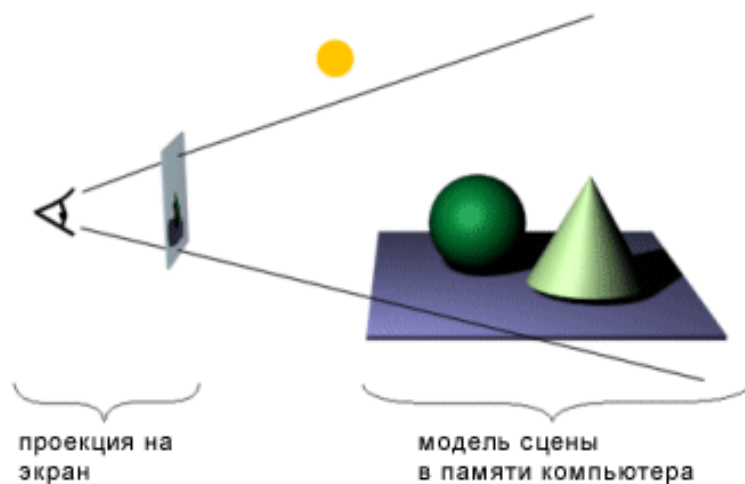


Рисунок 8.1 – Схема проецирования сцены на экран компьютера

Моделирование сцены (виртуального пространства моделирования) включает в себя несколько категорий объектов [7]:

- геометрия – построенная с помощью различных техник модель (например, здание);
- материалы – информация о визуальных свойствах модели (например, цвет стен и отражающая/преломляющая способность окон);
- источники света – настройки направления, мощности, спектра освещения;
- виртуальные камеры – выбор точки и угла построения проекции;
- силы и воздействия – настройки динамических искажений объектов;
- дополнительные эффекты – объекты, имитирующие атмосферные явления (свет в тумане, облака, пламя и пр.).

Задача трёхмерного моделирования – описать эти объекты и разместить их в сцене с помощью геометрических преобразований в соответствии с требованиями к будущему изображению.

Текстурирование подразумевает проецирование растровых или процедурных текстур на поверхности трехмерного объекта в соответствии с картой UV-координат, где каждой вершине объекта ставится в соответствие определенная координата на двумерном пространстве текстуры [7]. Как правило, многофункциональные редакторы UV-координат входят в состав универсальных пакетов трехмерной графики.

Освещение заключается в создании, направлении и настройке виртуальных источников света [7]. При этом, в виртуальном мире источники света могут иметь негативную интенсивность, отбирая свет из зоны своего «отрицательного освещения». Как правило, пакеты 3D графики предоставляют следующие типы источников освещения:

- omni light (point light) – всенаправленный;
- spot light – конический (прожектор), источник расходящихся лучей;
- directional light – источник параллельных лучей;
- area light (plane light) – световой портал, излучающий свет из плоскости;
- photometric – источники света, моделируемые по параметрам яркости свечения в физически измеримых единицах, с заданной температурой накала.

Существуют также другие типы источников света, отличающиеся по своему функциональному назначению в разных программах трехмерной графики и визуализации.

Одно из главных призваний трехмерной графики – придание движения (**анимация**) трехмерной модели, либо имитация движения среди трехмерных объектов [7]. Универсальные пакеты трехмерной графики обладают весьма богатыми возможностями по созданию анимации. Существуют также узкоспециализированные программы, созданные сугубо для анимации и обладающие очень ограниченным набором инструментов моделирования:

На этапе **рендеринга** математическая (векторная) пространственная модель превращается в плоскую (растровую) картинку [7]. Если требуется создать фильм, то рендерится последовательность таких картинок – кадров.

Как структура данных, изображение на экране представлено матрицей точек, где каждая точка определена, по крайней мере, тремя числами: интенсивностью красного, синего и зелёного цвета. Таким образом, рендеринг преобразует трёхмерную векторную структуру данных в плоскую матрицу пикселей. Этот шаг часто требует очень сложных вычислений, особенно если требуется создать иллюзию реальности.

Вследствие большого объёма однотипных вычислений рендеринг можно разбивать на потоки (распараллеливать). Поэтому для рендеринга весьма актуально использование многопроцессорных систем.

Среда симуляции Visual Simulation Environment RDS включает в себя две программы: графический и физический движок [1].

Основной задачей **графического движка** является визуализация (рендеринг) двухмерной или трехмерной компьютерной графики. Графический движок работает в режиме реального времени.

Физический движок производит симуляцию физических законов реального мира в виртуальном мире с той или иной степенью точности. Он позволяет создать виртуальное пространство, в которое можно добавить виртуальные статические и динамические объекты и указать законы взаимодействия тел и среды. Расчет взаимодействия тел выполняется самим движком. Рассчитывая взаимодействие тел между собой и средой, физический движок приближает физическую модель получаемой системы к реальной, передавая уточненные геометрические данные графическому движку. В состав RDS входит физический движок AGEIA PhysX Engine.

Объекты в симуляторе могут создавать иерархию, реализуя отношение предок/потомок. Например, манипулятор и сенсор являются дочерними объектами робота.

После запуска симулятора в нем отображается мир с позиции главной камеры, которая соответствует позиции глаза. Для изменения направления главной камеры нажмите левую кнопку мыши и переместите указатель в желаемом направлении.

Для **управления камерой** используются следующие клавиши клавиатуры [1]:

W – движение вперед;

S – движение назад;

A – передвижение влево;

D – передвижение вправо;

Q – перемещение вверх;

E – перемещение вниз.

Можно использовать сочетания этих клавиш. Удержание нажатой клавиши Shift одновременно с одной из клавиш управления камерой, ускоряет движение камеры в 10 раз.

Для управления симулятором доступны следующие команды (клавиши):

F2 – изменяет режим рендеринга;

F3 – включает/выключает физический движок;

F5 – переключение между режимами Run и Edit;

F8 – переключение активной камеры.

В нижней части окна симулятора расположена строка состояния, которая вызывается в пункте меню View → Status Bar. В строке состояния отображается позиция активной камеры, представленная тремя координатами X, Y и Z, где оси OX и OZ представляют направления, параллельные плоскости земли, ось OY располагается перпендикулярно земле.

В данном случае используется правая координатная система. Это означает, что направление оси OX указывает направо, оси OY – вверх, ось OZ указывает направление за пределы экрана. При этом направление глаза соответствует отрицательному направлению оси OZ, т. е. вглубь экрана.

Симулятор может работать в следующих двух режимах (пункт меню Mode):

1. Edit – в данном режиме возможно изменение сцены, параметров объектов, находящихся на сцене, добавление новых объектов в сцену.

2. Run – в данном режиме запускается процесс симуляции.

Симулятор поддерживает следующие режимы отображения сцены (пункт меню Render) [1]:

1. Visual – полностью отображаются все объекты сцены.
2. WireFrame – отображаются только линии треугольников, из которых построены объекты сцены.
3. Physics – отображаются фигуры, из которых построены объекты и с которыми работает физический движок. Сложные объекты иногда представляются в виде простых объектов, таких как кубы и сферы. Если некоторые объекты не сталкиваются или передвигаются случайным образом, нужно переключиться в указанный режим и проанализировать фигуры, составляющие объекты.
4. Combined – объединяет режимы Visual и Physics (рис.8.2), при этом параллелепипед, размещенный внутри объекта, показывает центр масс; если цвет параллелепипеда красный, то объект управляется вручную (т. е. является кинематическим), если – белый, то объект управляется физическим симулятором;
5. No Rendering – объекты сцены не отображаются.

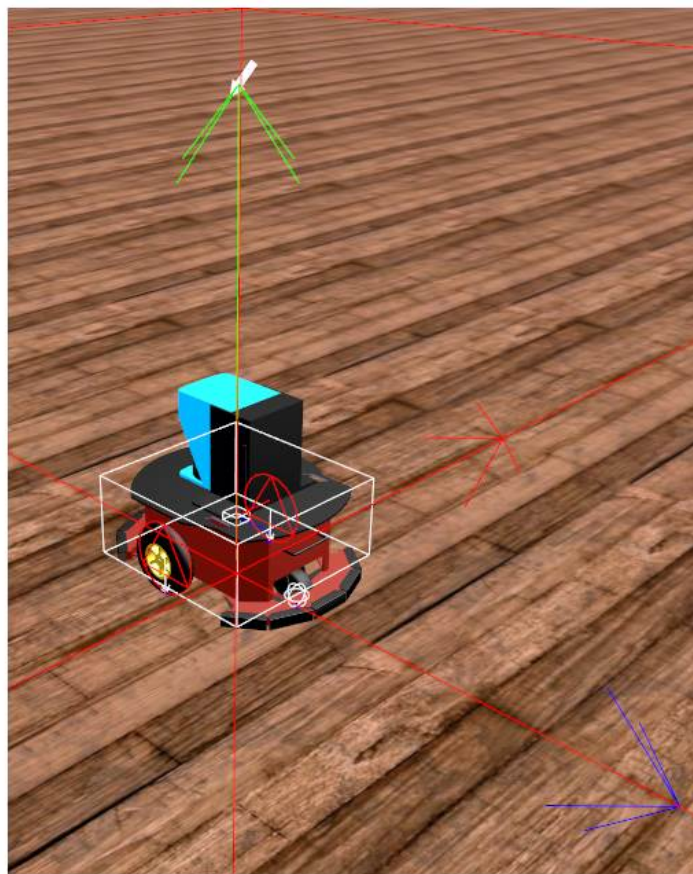


Рисунок 8.2 – Окно симулятора в режиме Combined

Рассмотрим особенности **редактирования и добавления объектов в сцену** в среде Microsoft Robotics Developer Studio R4.

Редактор симулятора открывается при выборе пункта **Mode → Edit** главного меню окна (рис. 8.3). После запуска редактора окно будет содержать отображение сцены, список объектов сцены и свойства выделенного объекта.

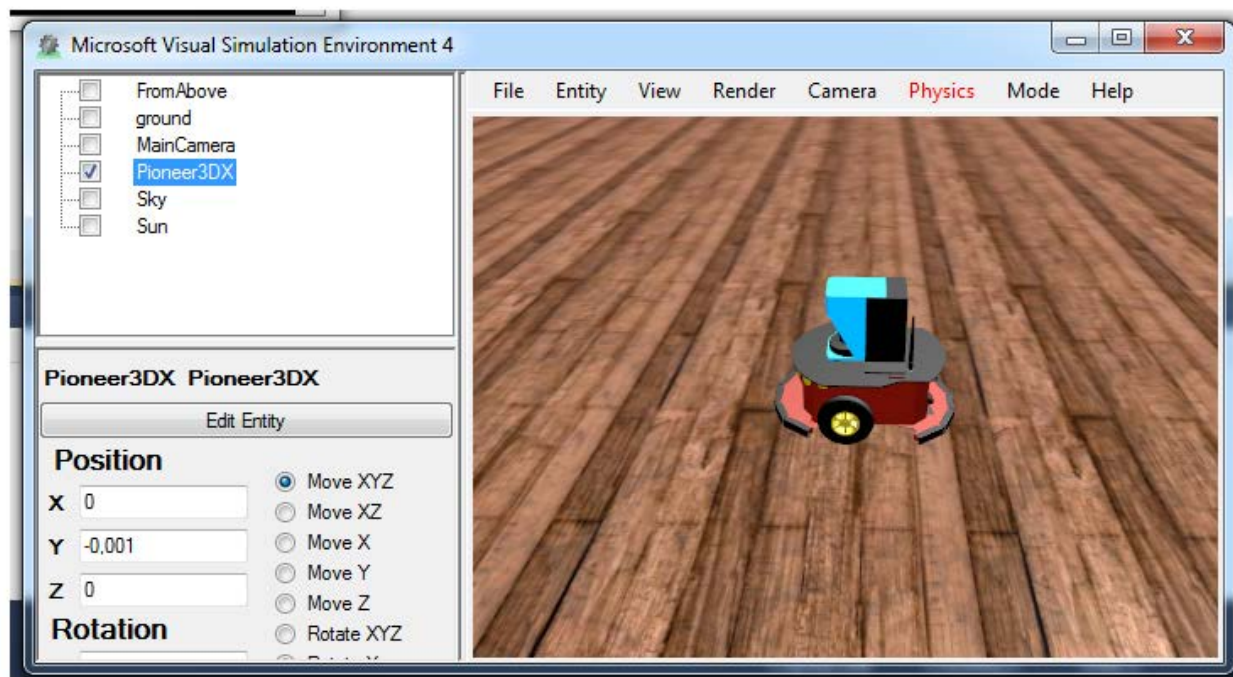


Рисунок 8.3 – Окно редактора симулятора

Объект можно отметить на сцене, если выделить его в списке объектов и нажать клавишу **Ctrl**. Если после выполнения данной операции камера не показывает объект, то можно использовать сочетания клавиш **Ctrl**, **Shift** и клавиш управления курсором для просмотра различных видов объекта:

- **Ctrl** + клавиша вверх: вид сверху;
- **Ctrl** + **Shift** + клавиша вверх: вид снизу;
- **Ctrl** + клавиша влево: вид справа;
- **Ctrl** + **Shift** + клавиша влево: вид слева;
- **Ctrl** + клавиша вправо: вид спереди;
- **Ctrl** + **Shift** + клавиша вправо: вид сзади.

Для перемещения выбранного объекта по сцене нужно выбрать направление движения в меню редактирования свойств (**Move XYZ**, **Move XZ**,

Move X, Move Y, Move Z) и при нажатой левой кнопке мыши переместить курсор в нужном направлении. В этом же меню можно указать точные координаты расположения объекта и организовать вращение объекта вдоль каждой из осей.

При вставке объекта в сцену можно указать предка данного объекта, которым может быть один из объектов сцены. В результате симулятор будет рассматривать эти два объекта как единое целое.

Рассмотрим некоторые свойства объекта доступные в окне свойств, которое открывается при нажатии на кнопку Edit Entity (рис. 8.4) [1].

1. **Position, Rotation** – определяют позицию и угол наклона объекта к каждой из осей координат.

2. **MeshScale/MeshTranslation/MeshRotation** – позволяют упростить подгонку формы (сетки) объекта к самому объекту. Это связано с тем, что разные средства создания трехмерных объектов могут использовать разные направления осей и масштаб.

3. **Meshes** – отображает информацию о сетках, загруженных для объекта. Сетка может быть создана из фигур, которые составляют объект, или загружена из .obj-файла. Также данное свойство позволяет изменять цвет, яркость и другие параметры объекта. Если были изменены параметры материала, их можно сохранить, используя пункт меню File → Save Material Changes.

4. Свойства, определяющие освещенность объекта:

– **Ambient** – определяет, как объект отражает свет, поступающий из окружающей среды, т. е. свет, отраженный от поверхности других объектов;

– **Diffuse** – определяет, как свет будет отражаться от объекта, если он будет освещен рассеянным светом; данная величина определяет максимальное значение света, которое будет отражено частью объекта, повернутой к источнику света; если некоторая часть объекта размещена перпендикулярно источнику света, то она ничего отражать не будет; диффузный свет позволяет видеть объекты трехмерными;

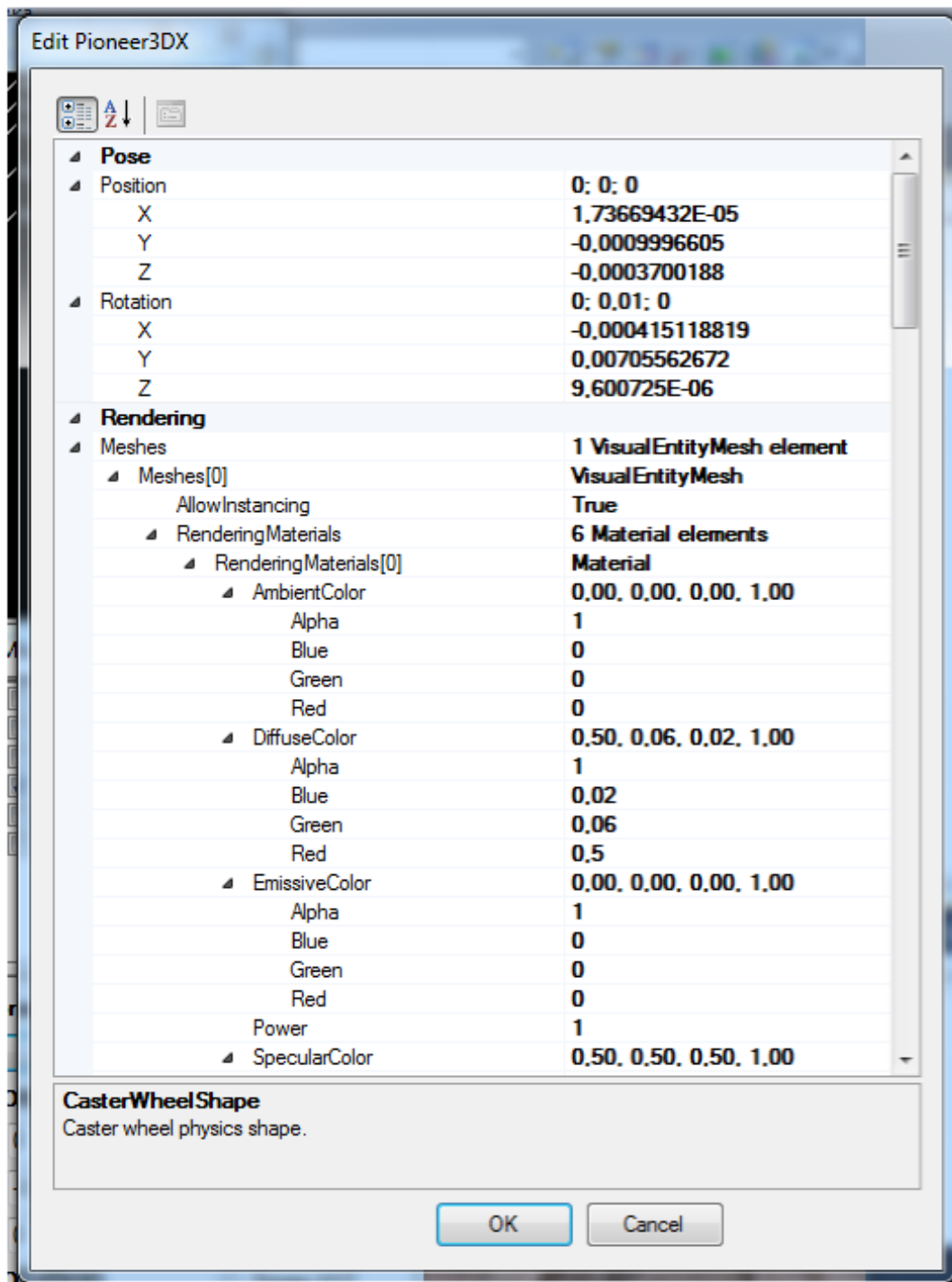


Рисунок 8.4 – Окно свойств объекта

- Specular – определяет, как много света будет отражаться от объекта при его освещении отраженным светом, обычно используемым для создания эффекта освещения солнцем; цвет отраженного света эквивалентен цвету света, а цвет диффузного света – цвету объекта;
- Power – определяет плотность отраженного пятна, большее значение параметра соответствует меньшему отражению.

5. В разделе **Прочее** (Misc) можно найти:

5.1. **Flags** – определяет поведение объекта. Если флаги не определены, предполагается, что объект – динамический:

- **Kinematic** – кинематический объект;
- **IgnoreGravity** – на объект не имеет влияние гравитация;
- **DisableRotationX**, **DisableRotationY**, **DisableRotationZ** – запрещает вращение объекта вдоль указанных осей;
- **DisableCollisions** – объект не взаимодействует с другими объектами сцены.

5.2. **Name** – имя объекта (должно быть уникальным). При изменении имени некоторые сервисы не смогут работать с данным объектом.

Каждый объект имеет состояние, представляющее собой набор параметров. Окно изменения состояния объекта открывается в режиме **Mode** → **Edit** следующим образом:

1. Выберите объект в списке.
 2. В окне свойств объекта нажмите кнопку «Edit Entity».
 3. В открывшемся окне выделите строку «Entity State» и нажмите кнопку (...).
- Рассмотрим некоторые параметры состояния объекта (рис. 8.5) [1].

1. Раздел **Graphics Assets**.

1.1. **Default Texture** – если с объектом не связана сетка, определяющая ее форму, то она генерируется из физических фигур, составляющих объект; в этом случае указанное свойство определяет текстуру, которая накладывается на сгенерированную сетку; текстура должна быть двумерной и находиться в одном из следующих форматов: .dds, .bmp, .png, .jpg, .tif.

1.2. **Mesh** – файл, определяющий форму объекта; используемый формат файла – .obj или .bos.

2. Раздел **Motion**.

2.1. **Angular Velocity** – вектор угловой скорости перемещения объекта в каждом из направлений.

2.2. Velocity – вектор скорости перемещения объекта в каждом из направлений.

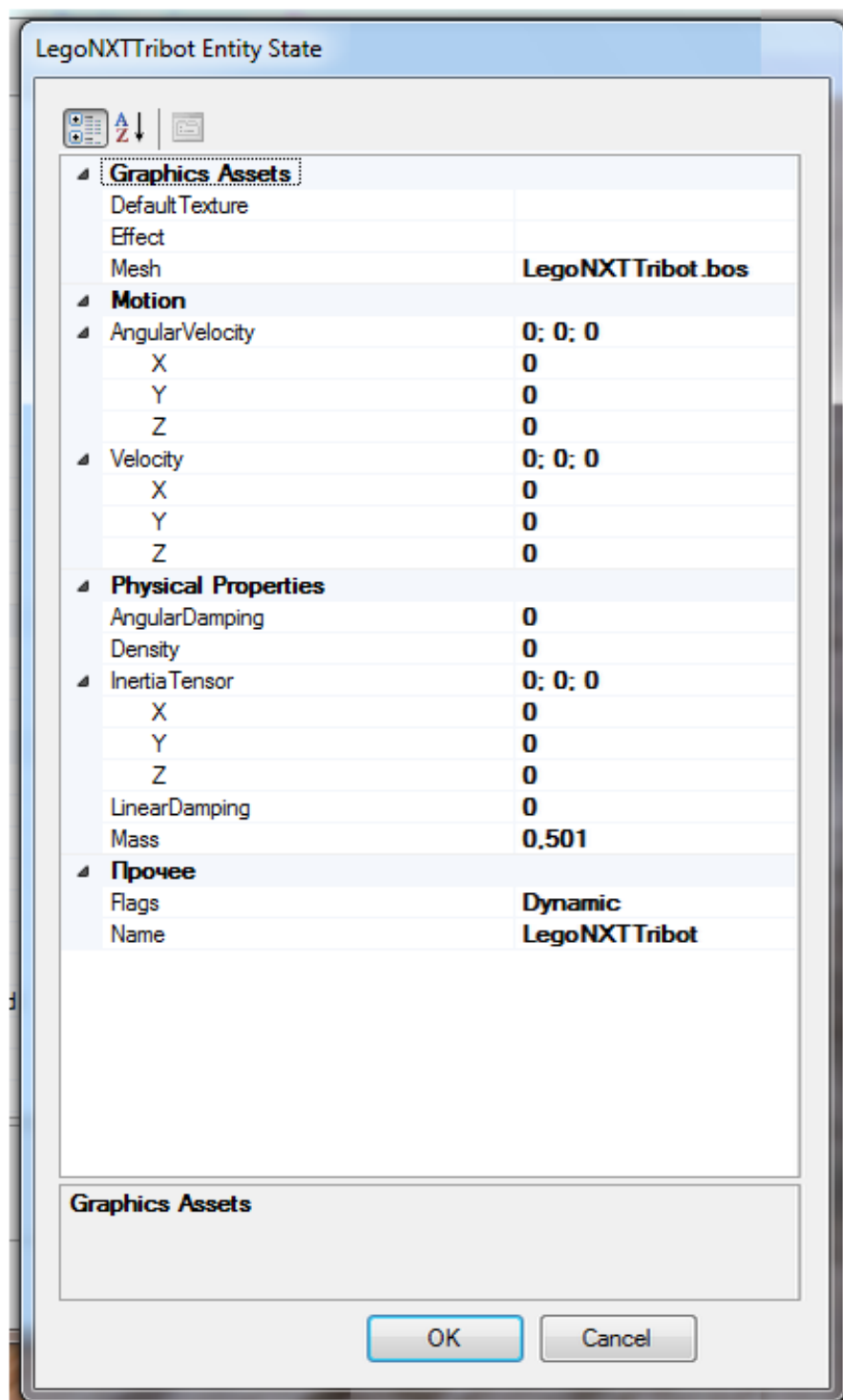


Рисунок 8.5 – Окно состояния объекта

3. Раздел **Physical Properties**.

3.1. Mass/Density – масса и плотность объекта. Если хотя бы одна из фигур объекта имеет массу и плотность, то вычисляется масса и плотность всего

объекта из суммы масс фигур, составляющих объект. Если сумма масс фигур равна нулю, симулятор использует величину Mass для определения массы объекта. Центр масс вычисляется так, как если бы плотность объекта была однородной. Указанные замечания также касаются и плотности объекта.

3.2. Inertia Tensor – тензор моментов инерции, определяет насколько сложно повернуть фигуру в различных направлениях. Например, цилиндр просто повернуть около его продольной оси, но сложно повернуть около двух других осей, вследствие распределения масс. Если вектор не определен, симулятор вычисляет его на основе расположения и относительной массы фигуры объекта.

3.3. Angular Damping/Liner Damping – данные параметры позволяют моделировать свойство трения. Большие значения данных параметров соответствуют более быстрому затуханию угловой или линейной скорости. При нулевом значении объект будет двигаться бесконечно до тех пор, пока не столкнется с другим объектом.

Сцена, находящаяся в симуляторе, может быть сохранена в XML-файл и затем загружена. При загрузке сцены все объекты получают то же местоположение и скорость, которые они имели при сохранении сцены. Для сохранения сцены используется пункт меню File → Save Scene, для загрузки File → Load Scene.

При сохранении сцены записываются два файла: первый содержит состояние сцены и имеет расширение .xml, второй содержит манифест для сцены и заканчивается на .manifest.xml. [1]

8.2. Методические рекомендации

8.2.1. Методика добавления объекта в сцену средствами Visual Studio

1. Создайте проект содержащий сцену по методике 7.2.2, при этом исключив стены из перечня декораций.

2. Для добавления в создающуюся сцену нового объекта в функции Populate() допишите следующий код:

```

// Добавляем объекты
    SingleShapeEntity dice = new SingleShapeEntity( // Создаем
новый предмет
        new BoxShape(new BoxShapeProperties( // Для физического
движка предмет примет форму коробки
            10, // Масса объекта
            new Pose(new Vector3(-1f,-1f,-1f)), // Внутреннее
положение объекта
            new Vector3(0.1f,0.1f,0.1f)           // Размеры
объекта
        )),
        new Vector3(0.1f,0.1f,2.6f) // Положение объекта в
сцене
    );
    dice.State.Assets.Mesh =
"C:\\Users\\Student\\Documents\\Visual Studio
2010\\Projects\\RTS_Lab8\\models\\Dice_high_poly.obj";
    //dice.MeshScale = new Vector3(0.01f, 0.01f, 0.01f); //
Размер объекта

    dice.State.Name = "Dice"; // Название объекта
    SimulationEngine.GlobalInstancePort.Insert(dice); //
Добавляем объект в сцену
    SingleShapeEntity chair = new SingleShapeEntity(
        new SphereShape(new SphereShapeProperties( // Форма шара
            10, // Масса
            new Pose(), // Положение
            1 // Радиус
        )),
        new Vector3(2.1f, 2.1f, 2.6f)
    );
    chair.State.Assets.Mesh =
"C:\\Users\\Student\\Documents\\Visual Studio
2010\\Projects\\RTS_Lab8\\models\\Chair.obj";
    chair.MeshScale = new Vector3(0.01f, 0.01f, 0.01f);
    chair.SphereShape.SphereState.Material = new
MaterialProperties("sphereMaterial", 0.5f, 0.4f, 0.5f);
    chair.State.Name = "Chair";
    SimulationEngine.GlobalInstancePort.Insert(chair);

```

3. Для корректного выполнения кода в соответствующий каталог поместите папку с моделями объектов.

4. Запустите проект. На экране вы увидите сцену с добавленными объектами (рис. 8.6).

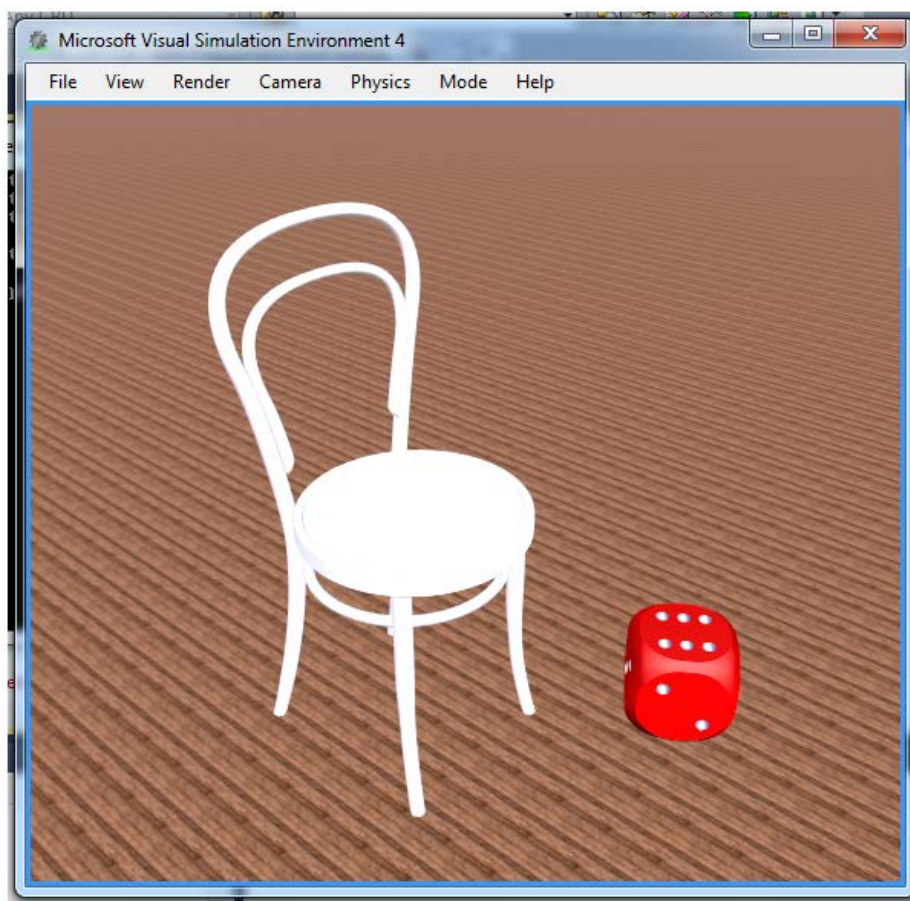


Рисунок 8.6 – Результат работы программы

8.2.2. Методика добавления объекта в сцену средствами Microsoft RDS

1. Для добавления нового объекта в созданную сцену кликните правой клавишей мыши в режиме Edit на любое место в сцене, и выберите из выпадающего подменю пункт «New Entity» (рис. 8.7). Для корректной работы программы требуется .NET 4.0, на версиях выше и ниже она может отрабатывать некорректно [1].

2. В открывшемся меню можно выбрать различные типы объектов. Для вставки препятствия выберите строку «MeshEntity» (рис. 8.8). Данный объект может иметь практически любой вид.

3. Выберите тип объекта «Convex», т. е. выпуклый (рис. 8.9).

4. Задайте свойства объекта, нажав на кнопку «Edit Entity». Задайте текстуру в строке «DefaultTexture». Выберите в качестве модели конус в строке «Mesh» (рис. 8.10). В итоге получим в исходной сцене конус (рис. 8.11).

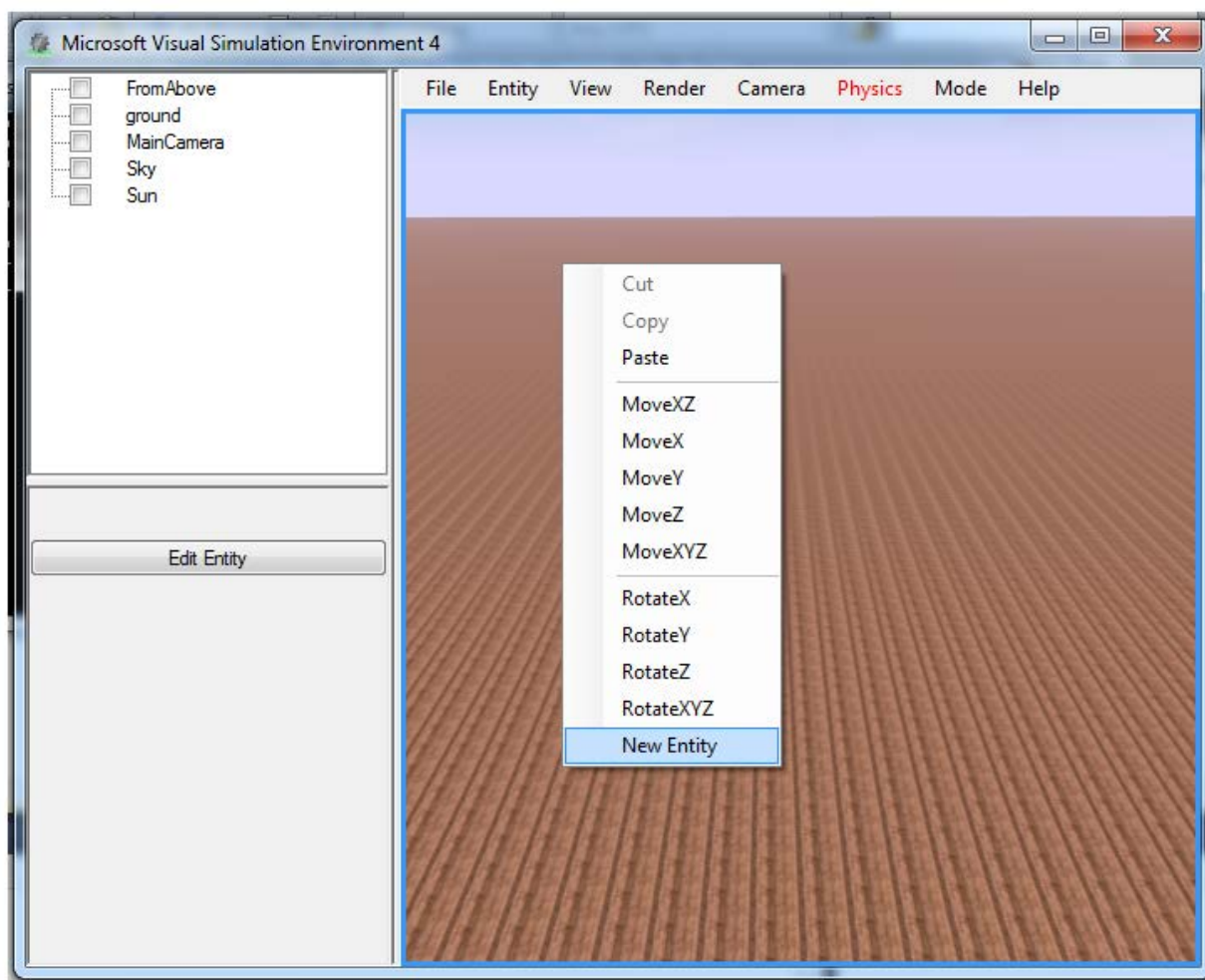


Рисунок 8.7 – Вставка нового объекта в сцену

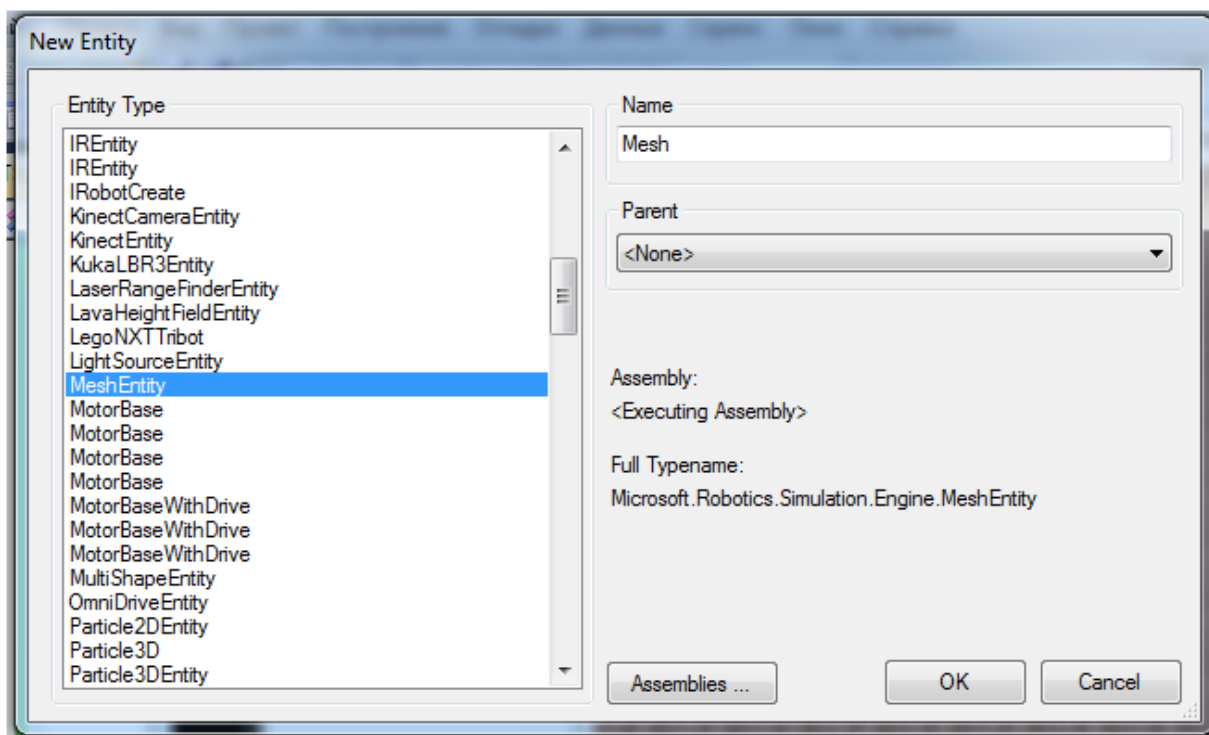


Рисунок 8.8 – Окно выбора нового объекта

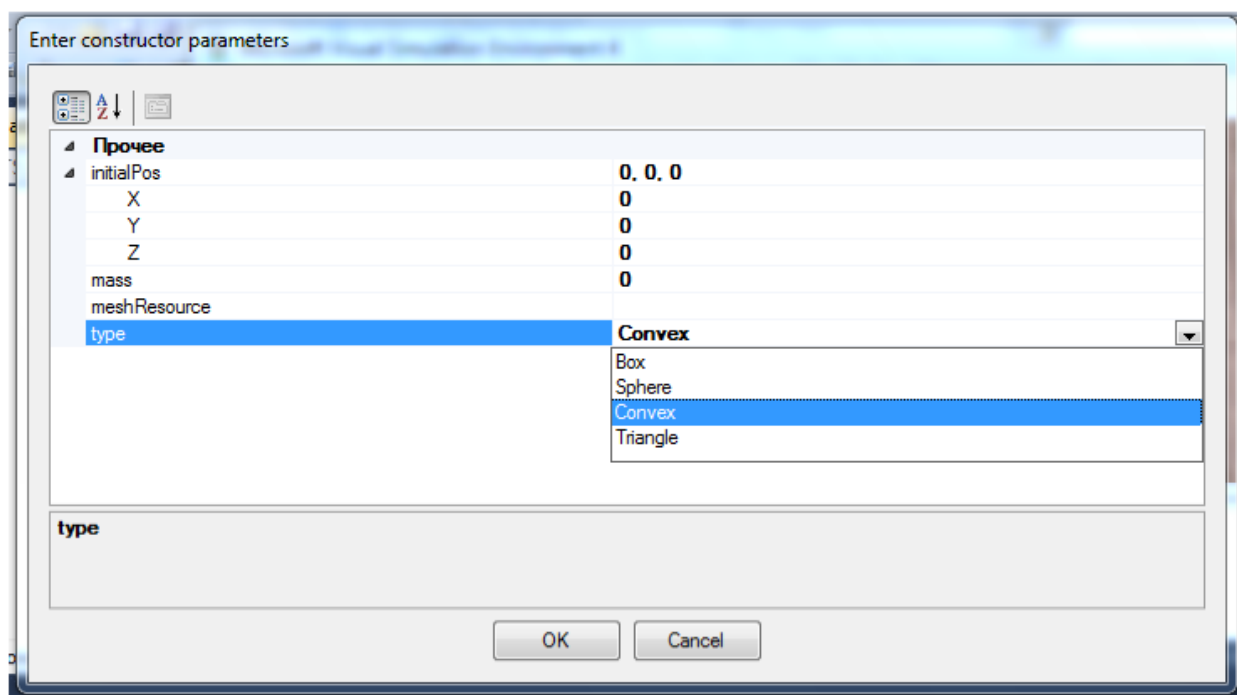


Рисунок 8.9 – Окно выбора параметров объекта MeshEntity

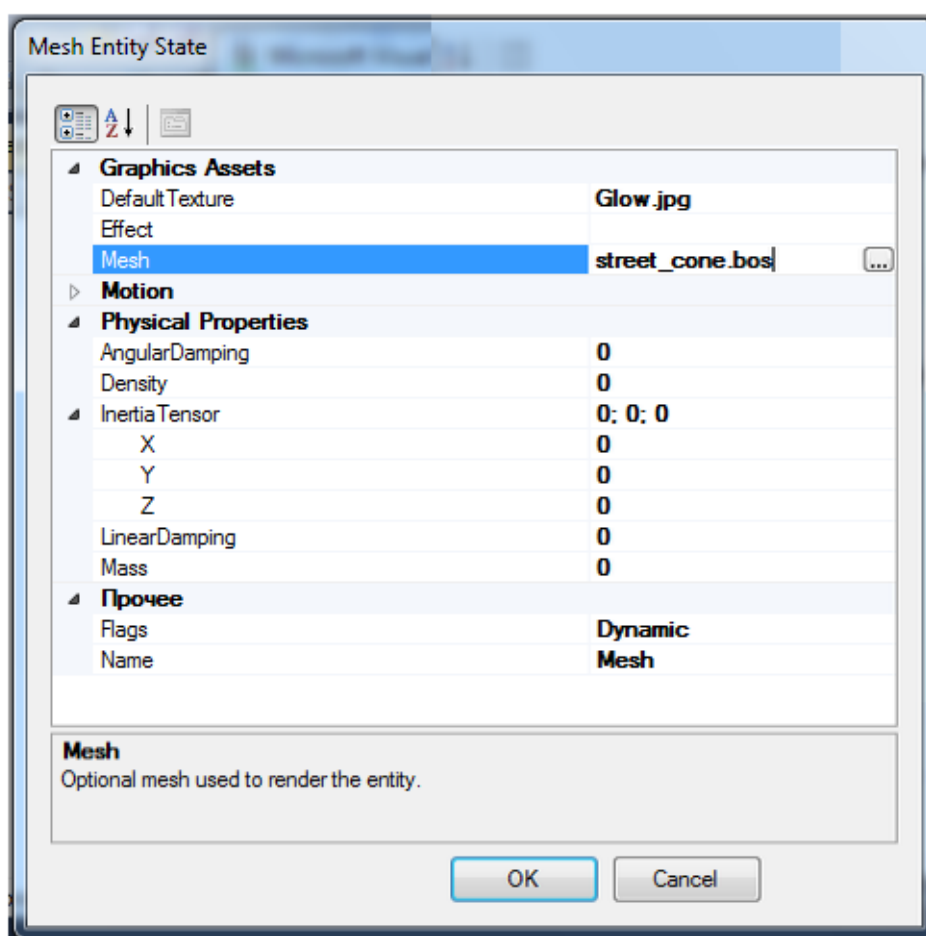


Рисунок 8.10 – Окно выбора состояния объекта MeshEntity

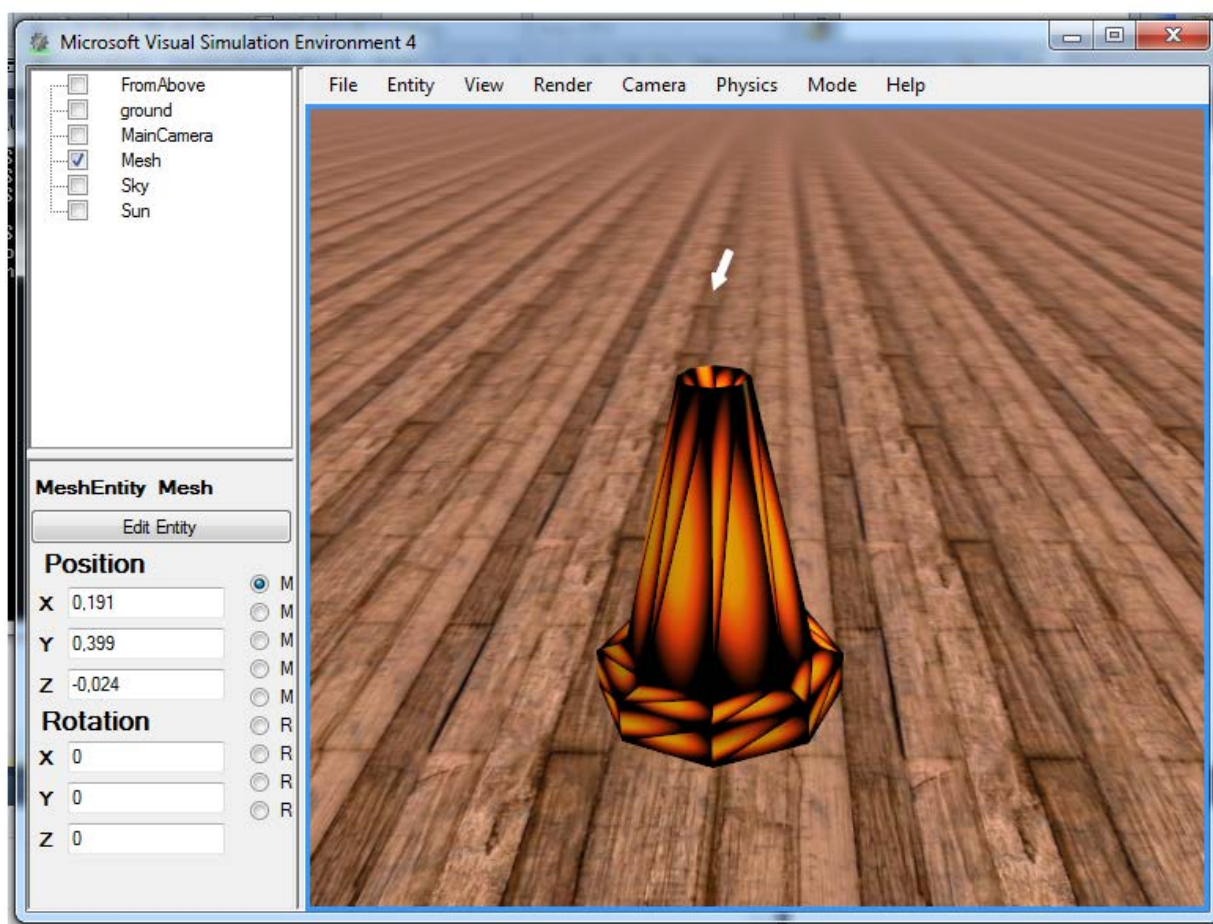


Рисунок 8.11 – Окно симулятора с добавленным объектом

8.2.3. Методика изменения положения объекта в сцене

1. Для изменения положения объекта в пространстве войдите в режим Edit и выберите объект, который необходимо переместить. Кликните правой клавишей мыши на объекте и в выпадающем меню выберите пункт «MoveXYZ» (рис. 8.12). Далее зажав клавишу Ctrl, можно перемещать объект.

2. Поднимите объект над поверхностью. Теперь если вы перейдете в режим Run, то объект будет висеть в воздухе. Данное состояние объясняется тем, что у объекта не заданы параметры массы и плотности.

3. Измените свойства объекта. Для этого перейдите в меню «Edit Entity» и выберите строку «Entity State» (рис. 8.13).

4. Задайте произвольные значения параметров Density и Mass (рис. 8.14). Теперь если перейти в режим Run, то объект упадет на поверхность. Таким образом, можно менять свойства объектов (форму, текстуру и т. д.).

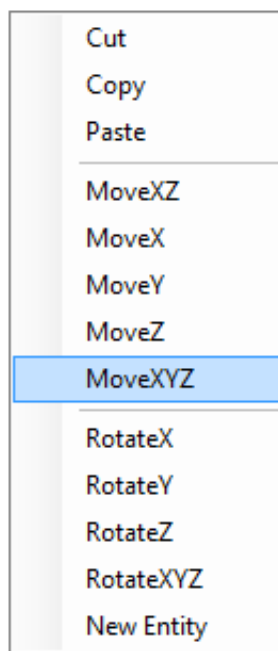


Рисунок 8.12 – Меню выбора трансформации над объектом

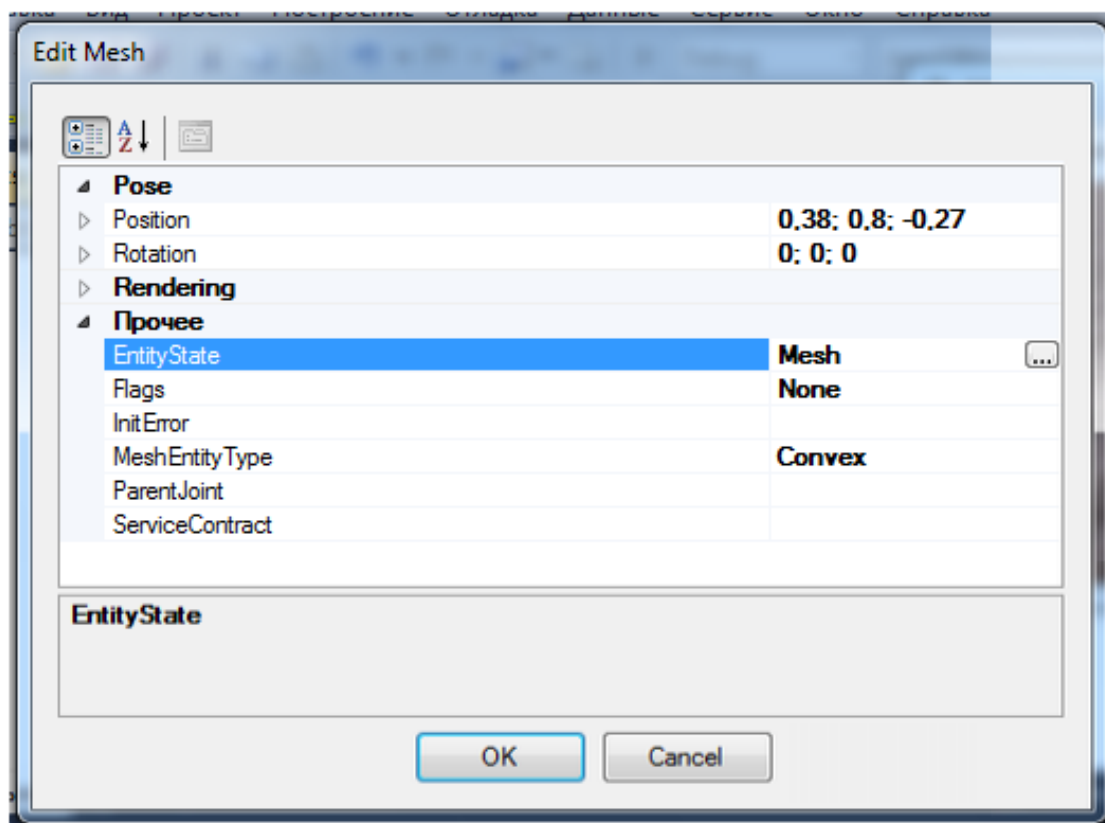


Рисунок 8.13 – Меню выбора свойств объекта

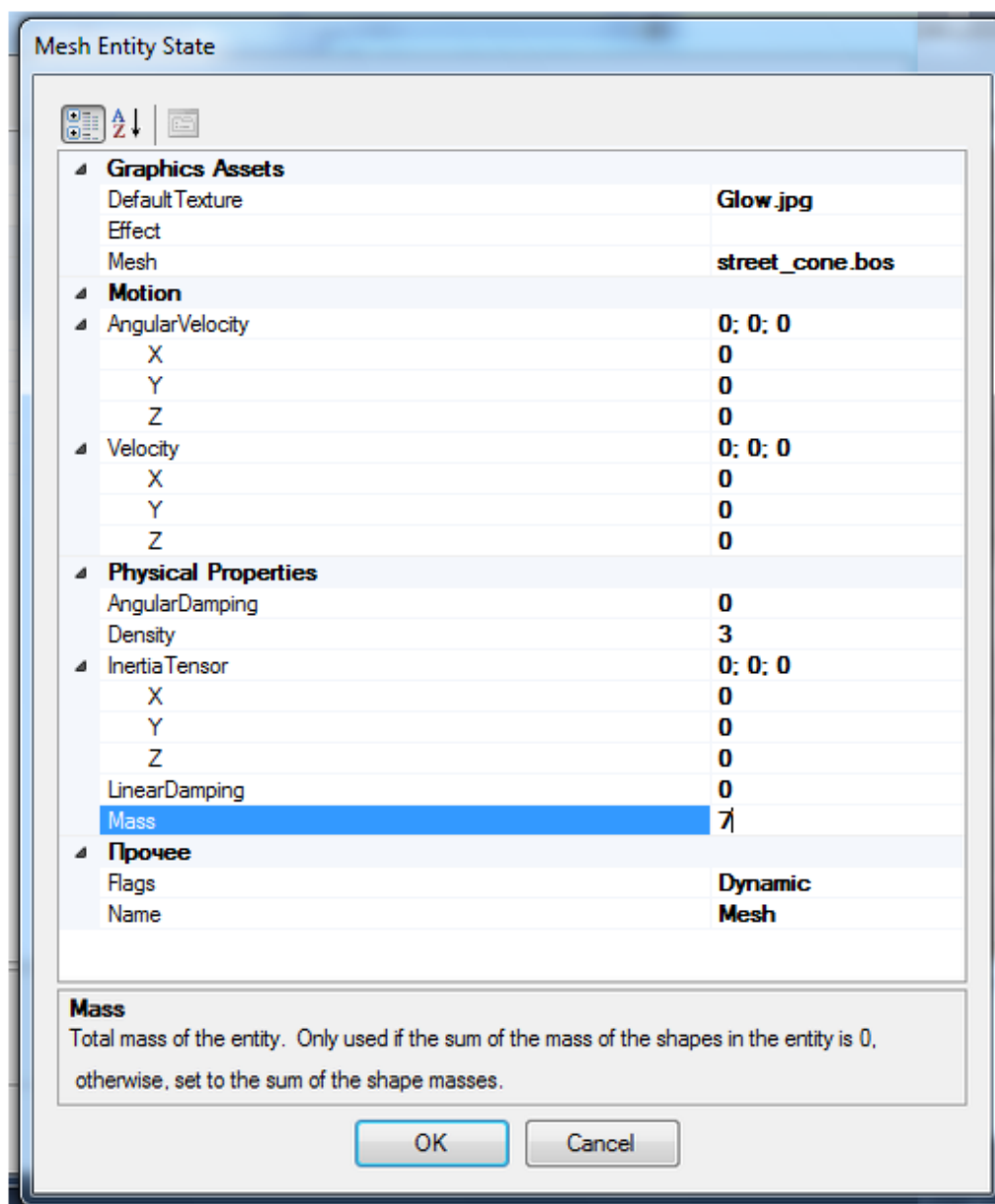


Рисунок 8.14 – Определение физических параметров Density и Mass

8.3. Лабораторная работа 8. Управление средой симуляции Visual Simulation Environment

8.3.1. Цель и содержание

Целью лабораторной работы является приобретение студентами практических навыков управления средой симуляции VSE Microsoft RDS с использованием Microsoft Visual Studio.

Содержание

1. Изучение процесса добавления объекта в сцену средствами Visual Studio.

2. Изучение процесса добавления объекта в сцену средствами RDS.
3. Изучение процесса изменения положения объекта в сцене.

8.3.2. Аппаратура и материалы

Для изучения материалов этой лабораторной работы необходимо:

1. Компьютер, удовлетворяющий минимальным системным требованиям для установки Microsoft Robotics Developer Studio R4.
2. Установленная платформа RDS R4.
3. Установленная среда XNA Game Studio 4.
4. Установленная интегрированная среда разработки программного обеспечения Visual Studio 2010 Express.

8.3.3. Порядок выполнения работы

1. Изучение теоретического материала и методических рекомендаций.
2. Выполнение задания, предлагаемого в лабораторной работе.

Задание:

1. Создайте сцену и добавьте два новых объекта средствами Visual Studio согласно методике 8.2.1 и с учетом задания в табл. 8.1. Убедитесь, что масштаб всех объектов соблюден, и они имеют свой вес (не висят в воздухе). Проект назовите согласно вашей фамилии (например, ivanov.cs).
2. Добавьте два новых объекта в созданную сцену средствами Microsoft RDS согласно методике 8.2.2 и с учетом задания в табл. 8.1.
3. Убедитесь, что колеса занимают соответствующие места у машины и робота, а ворота на футбольном поле расположены правильно. При необходимости воспользуйтесь методикой 8.2.3.
4. Задайте параметр перемещения для одного объекта согласно варианту задания (табл. 8.1).
5. Запустите симуляцию и прокомментируйте результат.
6. Сохраните созданную сцену.

Таблица 8.1. Варианты заданий для самостоятельного выполнения

№ п/п	Задание		
	средствами Visual Studio	средствами Microsoft RDS	Параметр перемещения объекта
1.	apple.obj, Book.obj	4x4Body.obj, 4 колеса (4x4 Wheel)	Angular Velocity (X)
2.	Table.obj, Book.obj	Corobot.obj, 4 колеса (CorobotWheel.obj)	Angular Velocity (Y)
3.	apple.obj, Table.obj	SoccerField.obj, 2-е ворота (BlueGoal.obj)	Angular Velocity (Z)
4.	Dice.obj, Book.obj	4x4Body.obj, 4 колеса (4x4 Wheel)	Velocity (X)
5.	apple.obj, Dice.obj	Corobot.obj, 4 колеса (CorobotWheel.obj)	Velocity (Y)
6.	vase.obj, Book.obj	SoccerField.obj, 2-е ворота (BlueGoal.obj)	Velocity (Z)
7.	apple.obj, vase.obj	4x4Body.obj, 4 колеса (4x4 Wheel)	Angular Velocity (X)
8.	vase.obj, table.obj	Corobot.obj, 4 колеса (CorobotWheel.obj)	Angular Velocity (Y)
9.	vase.obj, Dice.obj	SoccerField.obj, 2-е ворота (BlueGoal.obj)	Angular Velocity (Z)
10.	cup.obj, Book.obj	4x4Body.obj, 4 колеса (4x4 Wheel)	Velocity (X)
11.	apple.obj, cup.obj	Corobot.obj, 4 колеса (CorobotWheel.obj)	Velocity (Y)
12.	cup.obj, Table.obj	SoccerField.obj, 2-е ворота (BlueGoal.obj)	Velocity (Z)

8.4. Контрольные вопросы

1. Перечислите шаги, которые необходимо выполнить для получения трёхмерного изображения на плоскости.
2. Назовите основные функции выполняемые физическим и графическим движками.
3. В каких двух режимах может работать симулятор Microsoft RDS?
4. Как добавить объект в сцену средствами Visual Studio?
5. Как добавить объект в сцену средствами Microsoft RDS?

6. Какие режимы существуют в симуляторе Microsoft RDS для отображения сцены?
7. Назовите и охарактеризуйте файлы, которые генерируются при сохранении сцены?