СОДЕРЖАНИЕ

[СОДЕРЖАНИЕ 3](#_Toc452096730)

[ВВЕДЕНИЕ 4](#_Toc452096731)

[1. Выбор технологий для разработки эмулятора 9](#_Toc452096732)

[1.1 Аналогичные программные решения 9](#_Toc452096733)

[1.2 Выбор технологий реализации 10](#_Toc452096734)

[1.3 Постановка задачи 13](#_Toc452096735)

[1.4 Выбор фреймворка для разработки эмулятора 14](#_Toc452096736)

[2. Конструктивные особенности манипуляционных систем 17](#_Toc452096737)

[2.1 Классификация 17](#_Toc452096738)

[2.2 Кинематическая схема 18](#_Toc452096739)

[3. Разработка приложения эмулятора 22](#_Toc452096740)

[3.1 Разработка архитектуры эмулятора 22](#_Toc452096741)

[3.2 Описание режимов работы эмулятора 28](#_Toc452096742)

[4. Тестирование разработанных программных средств 30](#_Toc452096743)

[4.1 Тест-кейсы 34](#_Toc452096744)

[4.1.1 Тест-кейс: “Импорт модели” 34](#_Toc452096745)

[4.1.2 Тест-кейс: “Идентификация” 36](#_Toc452096746)

[4.1.3 Тест-кейс: “Прямая задача кинематики” 37](#_Toc452096747)

[4.1.4 Тест-кейс: “Обратная задача кинематики” 38](#_Toc452096748)

[4.2 Профилирование разработанных программных средств 40](#_Toc452096749)

[4.2.1 Анализ времени выполнения 42](#_Toc452096750)

[4.2.2 Анализ потребляемой памяти 43](#_Toc452096751)

[ЗАКЛЮЧЕНИЕ 45](#_Toc452096752)

[СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ 47](#_Toc452096753)

ВВЕДЕНИЕ

В связи с развитием технологий и высокими требованиями, предъявляемыми к точности выполняемых работ, современное производство сложно представить без автоматизации. Одним из главных средств автоматизации являются робототехнические системы [1]. Универсальность понятия “робот” может вводить неспециалистов в заблуждение, но когда упоминаются роботы на производстве, в основном говорят о робототехнических манипуляторах, управляемых посредством команд, отправляемых микропроцессорными устройствами [2]. На рисунке 1 представлена сравнительная характеристика роста численности поставок промышленных роботов на мировой рынок[3].

Рисунок 1 - Гистограмма динамики изменения объёма поставок тысяч промышленных роботов за период с 2002-го до 2015-го

На рисунке 1 видно, что после спада продаж в 2009-м году до 60-ти тысяч единиц, начиная с 2010-го года, к 2015-му прирост составил 300%. Согласно прогнозам с 2015-го года по 2018 уровень роста ежегодных продаж промышленных роботов составит около 15%. Наиболее востребованы промышленные роботы в автомобильной и электронной промышленностях. Самыми крупными производителями на рынке промышленной автоматизации считаются: FANUC (Япония), KUKA (Германия), ABB (Швеция, Швейцария), KAWASAKI (Япония) [4].

Использование манипуляторов позволяет увеличить производительность труда, качество продукции, ускорить перепрофилирование технологической линии [5, 6] на выпуск новой продукции [7, 8]. Манипулятор - составная система, подсистемы которой могут иметь как замкнутый, так и разомкнутый контур [1]. Замкнутые системы имеют обратную связь, и как следствие, сложнее в управлении и проектировании. Прогресс развития роботов, в частности манипуляторов, идёт в направлении совершенствования систем управления. Существуют промышленные роботы первого, второго и третьего поколений. Промышленные роботы первого поколения – это роботы с программным управлением, во многом позаимствованным от станков с числовым программным управлением (ЧПУ). Второе поколение - роботы с адаптивным управлением, снабжённые развитыми средствами очувствления, такие роботы значительно превосходят роботов первого поколения по функциональности и значительно дороже роботов первого поколения с программным управлением. Третье поколение - роботы с интеллектуальным управлением, в системах управления такими роботами влияние человека-оператора сводится к минимуму, в таких системах задание вводится в общей форме, а робот обладает возможностью принятия решений “по ситуации” [9].

На рисунке 2 представлена функциональная схема абстрактного манипулятора третьего поколения [1].

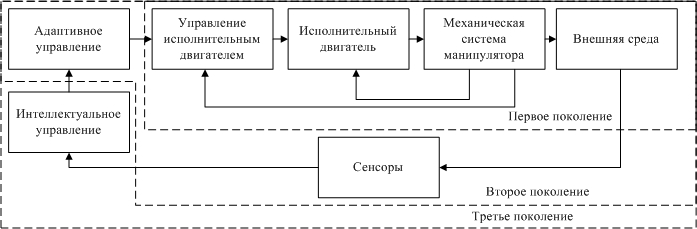


Рисунок 2 - Функциональная схема робота

В соответствии с рисунком 2 видно, что каждое следующее поколение - “надстройка” над предыдущим, соответственно базовые принципы и блоки одинаково важны для всех поколений промышленных роботов - манипуляторов, начиная от первого и заканчивая роботами третьего поколения.

Блок “Управление исполнительным двигателем” представляет собой составную часть системы программного управления. Системы программного управления разделяют на:

* системы дискретного управления - процесс управления приводами сводится к началу движения, движению с постоянной скоростью и окончанию движения путём торможения при достижении упора;
* системы дискретного позиционного управления - такие системы в отличие от систем циклического управления полагаются не на упоры, а на точность отработки приводами с обратной связью (исключение - разомкнутые системы на шаговых двигателях);
* системы непрерывного управления - когда движение по программной траектории осуществляется без остановок;
* системы управления по силе – такие системы управления, в которых поддерживается некоторый уровень воздействия, такие системы можно отнести как к системам программного, так и адаптивного управления, так как информация о силе воздействия - информация от датчиков о внешней среде.

Блок “Адаптивное управление” присущ роботам второго поколения, очувствленным, с системами адаптивного управления. Манипуляторы с системами адаптивного управления позволяют выполнять сложные операции, основанные на анализе информации, поступающей с блока “Сенсоры”, в соответствии с ней принимать решения о запуске той или иной “простейшей” системы управления, о которых говорилось выше в описании блока “Управление исполнительным двигателем”

Блок “Сенсоры” - это различные системы технического зрения (СТЗ). СТЗ выявляют различные свойства внешней среды, по этому признаку их разделяют на:

* системы определения геометрических свойств;
* системы выявляющие другие физические свойства (такие как плотность, вес и т.д.);
* системы выявляющие химические свойства.

Блоки “Управление исполнительным двигателем”, “Исполнительный двигатель”, “Механическая система манипулятора” и “обратная связь” зачастую в случае с манипуляторами представляет собой физически одно устройство точного позиционирования вала - сервопривод с присоединённым или встроенным в него редуктором.

Блок “Внешняя среда” представляет собой окружение рабочего пространства, условия работы манипулятора, манипулятор воздействует на внешнюю среду и посредствам блока “Сенсоры” получает обратную связь.

Манипулятор без управления не может осуществлять воздействие на объект управления. Манипуляционные системы вне зависимости от целей и форм воздействия призваны заменить человека, поэтому на всех без исключения уровнях управления происходит взаимодействие с оператором, которым является человек. По роду взаимодействия с оператором манипуляционные системы разделяют на автономные и эргатические (человеко-машинные) манипуляционные системы. В случае с автономными системами оператор программирует движения манипулятора, либо программирует функцию, описывающую это движение. Так в адаптивных системах работа оператора заканчивается до начала работы системы управления. Оператор должен предусмотреть всевозможные внешние воздействия, скорректировав систему управления, а так же установив всевозможные датчики, необходимые для правильного функционирования этой системы[2]. Управление манипулятором - комплекс задач, обобщённо включающий в себя несколько этапов:

* предварительный анализ рабочего процесса, подлежащего автоматизации;
* выбор существующего манипулятора или составление задания на разработку нового;
* адаптация манипулятора:
* разработка и модификация управляющих программ манипулятора в соответствии с требованиями, выявленными на этапе предварительного анализа.

Для решения задач используют различные программные средства, такие как математические пакеты (Matlab, Mapple, WolframMathematica и др.), системы имитационного, аналитического моделирования. Также существует целый класс специализированных программных средств, позволяющих производить моделирование работы манипулятора в условиях приближённых к условиям рабочего процесса, осуществлять анализ правильности выбора манипулятора, произвести адаптацию и разработку или модификацию управляющих программ - симуляторов. Они позволяют моделировать внешние воздействия, кинематику, динамику, воздействия на объект управления без необходимости создавать опытный образец и без доступа к физическому устройству [9]. Симуляторы разделяют по языку программирования разработки самой программы-симулятора, возможности взаимодействия с другими программами (например, математическими пакетами), наличию или отсутствию эмулируемых средств очувствления, средствам визуализации, типу лицензии на использование и распространение [10], и возможностями симуляции различных семейств роботов[11]:

* UGV (наземные мобильные роботы);
* UAV (воздушные роботы);
* AUV (подводные роботы);
* роботы манипуляторы;
* роботизированные руки;
* гуманоидные роботы.

Обилие возможностей разработки реализации подобных программных средств является причиной анализа технологий, используемых в аналогичных программных решениях.

1. Выбор технологий для разработки эмулятора

# 

1.1 Аналогичные программные решения

Сейчас существует большое количество программных средств для симуляции [11] работы роботов, наиболее популярные из них это: V-REP, Webots, MORSE, Gazebo. Все перечисленные программные средства симуляции - кроссплатформенные, имеют возможности симуляции физических свойств объектов, систем очувствления роботов и средства визуализации. За исключением MORSE, данные системы имеют встроенные средства для 3D моделирования, MORSE и Gazebo распространяются под лицензией open source. Программирование систем осуществляется на языках программирования Python, LUA, C++. Все перечисленные программные средства имеют возможность эмуляции роботов разных поколений и семейств, широкие возможности интеграции и взаимодействия с другими программными средствами, но требуют установки на персональный компьютер, что не всегда оправдано и обладают избыточной функциональностью применимо к симуляции робота манипулятора.

На основе субъективного анализа существующих решений были выдвинуты следующие взаимоопыляемые критерии выбора технологий для разработки:

1. переносимость приложения: готовое приложение должно работать без необходимости установки дополнительного программного обеспечения;
2. кроссплатформенность;
3. технологии разработки должны быть доступны для свободного использования в рамках выполнения поставленных задач в той мере, в которой необходимо для выполнения поставленных задач, это должно гарантировать стоимость разработки, ограничивающуюся временными затратами программиста;
4. готовое приложение должно обладать современными возможностями визуализации (динамическое освещение, сглаживание, отражения и т.д.).

1.2 Выбор технологий реализации

В соответствии с пунктом (а) для обеспечения переносимости, ЯП на котором написано приложение должен поддерживать работу без необходимости установки средств разработки, компилятора и других программных средств. Условие кроссплатформенности, пункт (б) также зависит от ЯП. Для обеспечения выполнения пункта (в) необходимо обратить внимание на лицензию распространения ПО, которое планируется использовать при разработке приложения. Исходя из пункта (г), технологии визуализации, используемые при разработке должны быть в стадии развития или иметь активную поддержку в долгосрочной перспективе, насколько это возможно прогнозировать. Виду высокой значимости ЯП необходимо рассмотреть возможные варианты. Обратив внимание в первую очередь на ЯП, используемые в аналогичных программных решениях, описанных в разделе 1.1 исходя из предположения, что эти ЯП по определению в той или иной мере подходят для создания аналогов имеющихся программных решений со схожим функционалом.

Python, используемый в качестве основного в MORSE, считающийся языком с удобным, относительно простым синтаксисом [12], в связи с этим получил широкое распространение как инструмент быстрого прототипирования алгоритмов и программ. Благодаря открытому исходному коду имеет множетсво специализированных реализаций и расширений языка[12, 13]:

* RPython – позволяет использовать Python как промежуточный язык разработки, выполняя его компиляцию в другие языки, такие как C++, Java и др.;
* NumPy – мощное средство выполнения задач, связанных с научной деятельностью, предоставляет возможности, схожие с аналогичными в коммерческих математических пакетах, таких как Mathlab;
* Btythoh – интерпритатор Python версии три и выше в javascript спецификации ESMASCRIPT5.

C++ - основной язык разработки симулятора Gazebo. C++ является самым производительным из этой тройки языков, используемых в аналогичных программных решениях. Этот ЯП как правило используется для написания высоконагруженных приложений и системного программирования, где нужна предельная скорость выполнения программы. Программные языки семейства C++ и семейства C, идейным продолжением которого является C++, позволяют использовать некоторые низкоуровневые возможности, такие как работа напрямую с памятью компьютера.

Расширение возможностей программировании на C++ связано с использованием библиотек, так при помощи библиотеки Wt появляется возможность разработки современных, интерактивных веб-приложений [14]. Удобство использования данной библиотеки в абстракции от средств отображения HTML страниц. Также одним из самых мощных инструментов решения аналогичных задач является использование стека, состоящего из самого C++, компилятора и ещё одного компилятора. С++ код компилируется в низкоуровневый машинный код (LLVM), затем этот код компилируется при помощи компилятора Emscripten (asm.js) в javascript [15]. Такой подход не только позволяет получать высокопроизводительные javascript приложения, но и осуществлять комбинирование вручную написанного javascript кода со скомпилированным на основе LLVM кода.

Последний ЯП из тройки это Lua, используемый в качестве основного языка программирования в симуляторе V-Rep. Lua позволяет не придерживаться определённой парадигмы программирования. Это язык с динамической типизацией, нашедший широкое применение в качестве встраиваемого ЯП для создания программных средств. Lua имеет библиотеки для организации взаимодействия посредствам API как с C++, так и Python [16,17]

Все представленные ЯП удовлетворяют пункту (в) и не требуют покупки лицензии для их использования в разработке или для последующей дистрибуции конечному пользователю. В настоящее время существуют по крайней мере три средства, имеющих возможность визуализации на современном уровне, который требуется согласно пункту (г), это:

* DirectX;
* OpenGL;
* Vulkan.

Из представленных выше средств подходят только OpenGL, и Vulcan, т.к. у DirextX нет поддержки кроссплатформенности.

Проанализировав технологии, используемые в аналогичных программных решениях, а также современных тенденций переноса десктопных приложений в веб пространство для разработки своего приложения решено использовать следующие технологические средства:

* веб-браузер;
* javascript как язык программирования;
* WebGL(надстройка над OpenGL) как средство визуализации трёхмерной графики.

Выбор веб-браузера в качестве целевой платформы связан в первую очередь с его наличием во всех популярных операционных системах, имеющих графический интерфейс, присутствует веб-браузер. К тому же программный код, выполняемый в браузере является кроссплатформенным, что позволяет вести разработку, не заботясь о переносимости приложения.

Языки C++ и Python, имея средства для работы с web-страницами, описанные выше, не были выбраны по причине усложнения отладки работающего приложения за счёт промежуточного звена в виде компилятора. Средства C++ и Python для работы с web больше подходят для разработки web библиотек, фреймворков а также портирования десктопных приложений в веб.

Из недостатков выбранных средств стоит отметить связанные с выбором целевой платформы повышенные требования к аппаратной части в сравнении с десктопными приложениями, написанными, например, на C++ и использующие DirectX3D для визуализации. Но данные недостатки с течением времени будут минимизироваться за счёт развития перспективного направления разработки на javascript с применением относительно новой технологии WebGL [18] .

1.3 Постановка задачи

Разрабатываемое приложение является эмулятором, отличающимся от симулятора возможностью взаимодействовать с реальным устройством манипулятора в режиме реального времени и передавать команды на осуществление управляющих действий. Таким образом, приложение включает в себя не только функциональность программы-симулятора, но и обладает расширенными возможностями в части взаимодействия с физическими системами. На структурной схеме (рисунок 3) представлены базовые компоненты [19] приложения-эмулятора. Рассмотрим подробнее каждый из них.



Рисунок 3 - Структурная схема базовых компонентов эмулятора робота-манипулятора

Компонент “Пользовательский графический интерфейс” на рисунке 3 позволяет пользователю вмешиваться в процесс работы системы без необходимости написания программного кода: задавать необходимые параметры, начальные условия, требуемую конфигурацию системы.

Компонент “Графическая визуализация” нужен для визуального контроля и анализа результатов проводимых экспериментов, трансляции работы реального манипулятора, поведение которого эмулируется. А также в качестве инструмента для проведения презентаций в ходе разработки манипулятора. Поэтому на компонент “Графическая визуализация” накладывается требование соответствия актуальным технологиям по отображению трехмерной графики (3D-графики) о которых говорилось ранее.

Компонент “Вычисления” служит для обслуживания математических расчетов, связанных с работой компонентов “Графическая визуализация” и “Управление виртуальным манипулятором”.

В компоненте “Управление виртуальным манипулятором” реализуется математический аппарат манипулятора, включая: реализацию кинематических схем, алгоритмов кинематики, уравнений динамики и прочее. Тип руки манипулятора и вид его зоны обслуживания определяет выбор подходящей для него системы координат [1].

Для сокращения времени на разработку приложения используются средства быстрой разработки приложений – фреймворки [20]. Выбор фреймворка определяет поставленная перед разработчиком задача и вытекающие из нее требования и ограничения. Помимо теоретических требований, диктуемых характером поставленной задачи, существенными являются и сугубо практические: выбранный для разработки язык программирования, платформа, под которую могут быть разработаны приложения на основе выбираемого фреймворка, а также лицензия на использование программного обеспечения. Важной частью работы с эмулятором является создание 3D-модели эмулируемого манипулятора, а значит необходима распространённая и бесплатная система 3D моделирования и простая методика импорта готовой модели в эмулятор.

1.4 Выбор фреймворка для разработки эмулятора

Исходя из выбранных технологий разработки эмулятора рассматривались следующие фреймворки Unity3D [21], Blend4Web[22], Treejs[23], Babylon.js[24]. Для разработки был выбран фреймворк Blend4Web, обладающий рядом преимуществ:

* полная совместимость со свободным редактором 3D-графики Blender3D [21,25];
* активное развитие фреймворка;
* частый выпуск новых версий;
* русскоязычная документация и поддержка;
* динамический язык программирования JavaScript с низким порогом вхождения [26];
* встроенные возможности вычисления различных математических данных, связанных с 3D-графикой (используется библиотека gl-matrix2);
* лицензия GPL-лицензия, позволяющая использовать, модифицировать и распространять программный продукт бесплатно [27].

Эмулятор манипулятора программно выполнен в виде модулей движка blend4web, который входит в состав фреймворка Blend4Web. Движок blend4web выполняет функции графической визуализации и вычислений данных (см. рисунок 3). Это позволяет сконцентрироваться на разработке основных частей и использовать современные средства визуализации.

Модель виртуальной сцены импортируется в форматах json и bin, используя подобный подход можно создать код для типовых решений и загружать их в приложение по мере необходимости. При таком подходе программист с любым уровнем подготовки может написать свою реализацию математического аппарата для описания работы конкретного компонента манипулятора.

В ходе работы эмулятор робота-манипулятора должен эмулировать трёхмерные модели манипулятора, позволять выполнить эмуляцию воздействия систем управления на виртуальный манипулятор, иметь возможность взаимодействия с реальным манипулятором - объектом моделирования.

Эмулятор должен предоставлять возможность взаимодействия с ним без необходимости вникать в его внутреннее устройство, без правки исходного кода самого манипулятора и/или его компонентов, на уровне достаточном для решения поставленных задач.

Программные средства эмуляции в своей архитектуре тесно связаны с объектом эмуляции, поэтому отдельное внимание при разработке таких программных средств уделяется описанию, подробное описание объектов эмуляции позволяет выявить дополнительные требования к архитектуре.

2. Конструктивные особенности манипуляционных систем

2.1 Классификация

Существует много конфигураций механических структур промышленных манипуляторов, их разделяют по грузоподъёмности, количеству манипуляторов (рабочих органов), по топологии кинематической цепи, по типу приводов, характеру выполняемых операций и по степени специализации [28,29]. По грузоподъёмности манипуляторы делятся на сверхлёгкие, лёгкие, средние, тяжёлые и сверхтяжёлые - до 1, до 10, до 100, до 1000 и свыше 1000 кг соответственно. По количеству рабочих органов манипуляторы можно условно разделить на 2 группы - манипуляторы с одним рабочим органом и манипуляторы с несколькими рабочими органами. Манипуляторы с одним рабочим органом встречаются чаще по сравнению с манипуляторами с двумя рабочими органами. В манипуляторах с двумя рабочими органами, как правило, второй рабочий орган выполняет вспомогательную функцию, например сталкивание готовой детали за пределы рабочего пространства манипулятора после выполнения всех поставленных задач. Топология кинематической цепи манипулятора в соответствии с количеством рабочих органов может быть простой разомкнутой и ветвящейся, существует также замкнутая кинематическая цепь, но применительно к манипуляторам такие цепи рассматриваются редко. По типу привода манипуляторы бывают электрическими, гидравлическими, пневматическими, пневмо-гидравлическими и гибридными, что означает, что в конструкции одного манипулятора могут использоваться разные привода из соображений массо-габаритных и экономических показателей.

По характеру выполняемых операций манипуляторы бывают технологические, вспомогательные и универсальные. Производственные манипуляторы выполняют операции, непосредственно связанные с производством промежуточного или конечного продукта на некотором этапе производства. Вспомогательные манипуляторы используют для вспомогательных операций, обслуживания производственных манипуляторов и основного технологического процесса в целом, для осуществления логистических операций, в частности учёт и транспортировка. Универсальные манипуляторы могут использоваться как производственные и вспомогательные.

Классификация позволяет сделать первые выводы о конструктивных особенностях, но для эмуляции этого недостаточно, необходимо знать кинематическую схему манипулятора.

2.2 Кинематическая схема

В кинематической схеме манипулятора, как правило, используются два типа сочленений - это телескопический шарнир и вращательный шарнир, эти сочленения образуют кинематические пары пятого класса - вращательная и поступательная. На рисунке 4 представлены условные отображения поступательной и вращательной кинематической пар.

|  |  |
| --- | --- |
| rotate.jpg  а) | translational.jpg  б) |

Рисунок 4 - Условные отображения кинематических пар: a - вращательная кинематическая пара; б - поступательная кинематическая пара

Кинематические схемы манипуляторов могут содержать как кинематические пары исключительно одного типа, так и комбинировать их. Зачастую кинематические схемы строятся по подобию проверенных и хорошо себя зарекомендовавших конструкций манипуляторов, например Puma и Unimate. Рассмотрим их кинематические схемы.

|  |  |
| --- | --- |
| kinematicscheme-pm01.jpg | kinematicscheme-unimate.jpg |
| а) | б) |
| Рисунок 5 - Кинематическая схема: а - Puma-560; б – Unimate | |

Как видно из рисунка 5(а) кинематическая схема манипулятора Puma представляет из себя сочетание вращательных кинематических пар. А схема манипулятора Unimate, изображённая на рисунке 5(б), содержит как вращательные, так и поступательные кинематические пары. Как можно заметить из иллюстраций на рисунке 5 схваты двух разных манипуляторов с разными кинематическими схемами имеют похожую кинематическую схему, такое сходство не является случайным, подобная кинематическая схема схвата позволяет выполнять вращения объекта манипулятора во всех плоскостях, что делает манипулятор более универсальным. Число степеней подвижности (далее степеней свободы) v как одна из характеристик манипулятора считается по формуле [5]:

|  |  |
| --- | --- |
| , | (1) |

где *n* - число подвижных звеньев, а *pi* - число кинематических пар *i*-го класса.

Формула (1) является справедливой как для Puma, так и для манипулятора Unimate, когда все кинематические пары манипулятора относятся к пятому классу. В таком случае количество степеней свободы равно числу кинематических пар, формула (1) приводится к следующей формуле:

|  |  |
| --- | --- |
| , | (2) |

В соответствии с формулой (2), манипуляторы Puma и Unimate имеют 5 и 3 степеней свободы соответственно.

В робототехнике для представления данных, описывающих кинематические соотношения манипулятора, принято использовать представление Денавита-Хартенберга (ДХ-представление) [5], оно позволяет составить систему координат для каждого звена манипулятора так, чтобы пользуясь математическим аппаратом однородных матриц преобразования, можно было решать прямую и обратную задачи кинематики. Аппарат однородных матриц преобразований может быть оптимизирован за счёт применение кватернионов, бикватернионов и различных подходов к выполнению матричных вычислений. В таблице 1 представлены характеристики звеньев манипулятора в соответствии с ДХ-представлением:

Таблица 1 - ДХ-представлние манипулятора

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Звенья | α | φ | l, мм | d, мм | Диапазон изменений q |
| Звено 0 | α0 | φ0 | l0 | d | от q0 до q0′ включ. |
| Звено 1 | α1 | φ1 | l1 | d1 | от q1 до q1′ включ. |
| … | … | … | … | … | … |
| Звено N | αn | φn | ln | dn | от qn до qn′ включ. |

В таблице 1 буквенные обозначения заменяются числами, полученными при построении системы координат манипулятора согласно ДХ-представлению: α - угол поворота системы координат *i*-го звена манипулятора относительно i-1 звена; φ - угол скрутки, *l* - длина звена манипулятора; *d* - расстояние от начала системы координат *i*-го звена манипулятора до *i*+1 звена; *q* - обобщённая координата, для поступательной кинематической пары это длина звена, а для вращательной – угол поворота. Вектор из чисел α,φ,l,d однозначно описывает положение каждого звена относительно предыдущего. Диапазон измерений *q* обуславливается предельно допустимыми углами поворотов или изменениями длины звеньев, заложенных при проектировании манипулятора.

Перед тем, как приступить непосредственно к разработке эмулятора и его тестированию в отношении некоторой, конкретной модели манипуляционной системы, описываемой и классифицируемой на основании вышеизложенных понятий, представлений и формул, необходимо создать детализированную структуру приложения, такая структура называется архитектурой.

3. Разработка приложения эмулятора

3.1 Разработка архитектуры эмулятора

Рассмотрим архитектуру приложения эмулятора манипулятора, представленную в виде структурной схемы на рисунке 6.



Рисунок 6 - Структурная схема архитектуры приложения

Структурная схема даёт лишь общее понимание работы приложения и не является стандартом, таковым является UML, в частности UML2, являющийся новейшим на текущий момент. Разработка эмулятора ведётся, придерживаясь парадигм объектно-ориентированного программирования (ООП), диаграммы классов являются программным представлением структурных единиц схемы. Язык объектно-ориентированного моделирования UML возник из методов моделирования Object-Modeling Technique и Booch как инициатива Гради Буча и Джеймса Рамбо в 1994-ом году. К 1997-му году распространение UML достигло впечатляющих масштабов, к разработке новых версий спецификации языка в рамках консорциума UML Partners подключились ведущие комании, такие как: Digital Equipment Corporation, Hewlett-Packard, i-Logix, IntelliCorp, IBM, ICON Computing, MCI Systemhouse, Microsoft, Oracle Corporation, Rational Software, Texas Instruments, а также Unisys. В том же году сотрудничество дало плоды в виде версии UML 1.0 и UML 1.1 [30]. Возвращаясь к стандартизации, основополагающим в выборе инструмента создания диаграмм является то, что UML версий 1.4.2 и 2.4.1 является действующим международным стандартом [31] ISO/IEC 19501 и ISO/IEC 19505-1 ,19505-2 соответственно. Но стоит отметить, что при проектировании диаграмм были допущены упрощения в плане отсутствия описаний типов свойств классов, что связано с неявной динамической типизацией ЯП(языка программирования), за счёт этого было сэкономлено время построения диаграмм и при этом не потеряна их функциональная нагрузка. Построение диаграмм UML были выполнены в программе UMLet [32].

Модуль “Вычисления” содержит модули “FABRIK”, “Прямая кинематика” и “Angles”. В модуле “FABRIK” реализуется алгоритм обратной кинематики “F.A.B.R.I.K.”, являющийся итерационным алгоритмом, который приобрел популярность за счёт простоты и низкой стоимости вычислений, также в результате работы алгоритма F.A.B.R.I.K. звенья принимают относительно естественные положения[33,34]. В модуле “Angles” вычисляются углы поворота сервоприводов, необходимые для достижения звеньями манипулятора заданного положения. На рисунке 7 приведены диаграммы классов соответствующих модулей.

|  |  |
| --- | --- |
| a) | в) |
| б) |

Рисунок 7 - Диаграммы классов: а - Angles; б - FABRIK; в – forwardKinematicProblemSolver

Метод calculateAngles() класса Angles принимает массив точек и вычисляет углы поворота. Метод algorithm() реализует алгоритм F.A.B.R.I.K., метод distBeetweenPoint() является вспомогательным для алгоритма и его использование извне напрямую не предусмотрено. Метод calcPositions() реализует алгоритм решения прямой задачи кинематики при помощи инструмента матриц преобразований, составленных на основе ДХ-представления, соответственно он получает параметры ДХ-представления, а также необходимые углы поворота. На выходе могут быть получены координаты в явном виде или матрица однородных преобразований для каждой входной точки (сочленения).

Модуль “Manipulator” состоит из подмодулей “Arm”, “Nodes”, “Fingers”, “TargetPoint”, представляющих математическое описание звеньев, шарниров, схвата, целевой точки схвата эмулятора соответственно. Каждый из них находит отражение в 3D-представлении манипулятора в эмуляторе.

|  |  |
| --- | --- |
| а) | б) |
| Рисунок 8 - Иллюстрация классов Manipulator и взаимодействий: а - Модуль Manipulator и подмодули Finger, Nodes, TargetPoint, Arm, а также модуль CraneGUI; б - Класс Particle и его потомки Arm, TargetPointm Finger, Nodes | |

При разработке используется ООП, поэтому придерживаясь его принципов и используя прототипное наследование, классы “Arm”, “Nodes”, “Fingers”, “TargetPoint” являются потомками класса “Particles”, этот приём проиллюстрирован на рисунке 8 (б).

Модуль “Blend4Web” содержит модуль “Обработка 3D-графики” и “Модуль вычислений”. Модуль “Обработка 3D-графики” служит для взаимодействия с объектом canvas html-страницы посредствам WebGL. В “Модуле вычислений” реализованы математические функции для работы с векторами, матрицами и кватернионами. На Рисунке 9 представлена диаграмма программной реализации фреймворка Blend4Web. Но так как фреймворк представляет из себя большую программу с богатой структурой, содержащей множество классов и обилие методов этих классов, методы и классы непосредственно не используемые в эмуляторе не представлены на диаграммах.

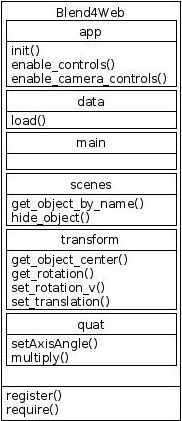


Рисунок 9 – структура движка Blend4Web

На диаграмме на рисунке 9 представлены следующие методы модулей Blend4Web, наиболее значимые в контексте частоты вызовов это: init() – являясь методом модуля «app» предоставляет средства инициализации движка, принимая в качестве параметров id html элемента canvas для отрисовки графики, позволяет на основе полученных параметров конфигурировать поведение движка, например, выводить отладочные выходные данные, такие как: частота кадров в секунду(fps); сообщения об ошибках; информация о количестве шейдеров и другую техническую информацию, которая может быть полезна как разработчикам движка, так и приложений на его основе; методы enable\_controls() и enable\_camera\_controls() позволяют использовать встроенные средства интерактивного взаимодействия с холстом (область html5 тега canvas); метод load() модуля data принимает в качестве аргумента путь к json файлу, а также дополнительные параметры, влияющие на поведение метода, например вызов callback по завершению загрузки экспортированного файла или включения ожидания загрузки всех ресурсов импортируемой сцены; модуль main является своего рода низкоуровневым, потому как его методы имеют больший приоритет нежели методы других модулей движка blend4web. Например метод init() модуля app основан на init() модуля main. Методы модуля scene используются для взаимодействия с объектами импортированной 3D сцены, позволяют искать, скрывать объекты. Методы модуля transform позволяют выполнять манипуляции с 3D объектами сцены, выполнять перемещение, повороты, а также получать информацию об объекте, например геометрический центр объекта можно узнать передав в качестве аргумента в метод get\_object\_center() 3D объект сцены. Методы моделя quat позволяют выполнять арифметические операции с кватернионами; глобальные методы require() и refister() позволяют модулям взаимодействовать между собой.

Модуль “GUI” (Graphical User Interface) выполняет функции обработки данных модуля “Манипулятор” для вывода их в графическом интерфейсе и для передачи данных из графического интерфейса в модуль “Манипулятор”.

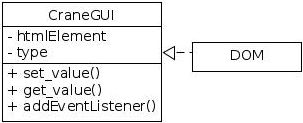


Рисунок 10 - диаграмма класса CraneGUI

GUI в веб приложениях как правило представлен в виде HTML, CSS кода ввиду его простоты и малогопотребления ресурсов в сравнении с WebGL графикой. Document Object Model (DOM) предоставляет методы для работы с ним, а клиентское приложение, используя данные методы, изменяет объекты DOM (HTML). Используя шаблоны проектирования, следует применять метод addEventListener[30], предоставляемый DOM, что и сделано в классе “CraneGUI”. Примеры графического интерфейса программы представлены на Рисунке 11

|  |  |
| --- | --- |
| mainMenu | |
| a) | |
| roboArmMenu | D:\alexey\Загрузки\11 |
| б) | в) |
| Рисунок 11 - примеры GUI: а - главное меню; б - подменю RoboArm1; в - подменю настройки привязки программной логики к импортируемой 3D модели | |

Изображённые на рисунке 11 элементы интерфейса, а также другие элементы интерфейса манипулятора осуществляют задачу взаимодействия с манипулятором без изменения программного кода эмулятора.

Исходя из задач, в соответствии с архитектурой был разработан эмулятор, обеспечивающий несколько режимов работы приложения. Описание режимов работы приложения является важной частью исследования в рамках дипломной работы, и может быть отнесено к архитектуре.

3.2 Описание режимов работы эмулятора

Работа приложения начинается с создания экземпляра класса движка Blender4Web. Далее происходит создание экземпляров классов остальных модулей и запуск управляющей программы. Возможны несколько вариантов управляющей программы:

* демонстрационная управляющая программа;
* поворот звеньев на заданный фиксированный угол;
* перестроение звеньев манипулятора для достижения заранее заданной точки.

Демонстрационная управляющая программа служит для демонстрации работы эмулятора без необходимости вмешательства пользователя. В процессе работы она итеративно задает точку на основе заранее определенного массива конечных точек и выполняет на каждой итерации управляющую программу “Перестроение звеньев манипулятора для достижения заранее заданной точки”.

Управляющая программа “Поворот звеньев на заданный фиксированный угол” получает экземпляры класса Arm и углы поворота. Далее она передает полученные объекты и углы в модуль “Обработка 3D-графики”. Модуль “Обработка 3D-графики” передает свойства экземпляров класса Arm, необходимые для ориентации в пространстве, в модуль вычислений. Модуль вычисления возвращает новые свойства экземпляров класса Arm, а модуль “Обработка 3D-графики” перерисовывает кадр.

Управляющая программа “Перестроение звеньев манипулятора для достижения заранее заданной конечной точки” получает экземпляр класса TargetPoint и изменяет его свойства. Программа получает остальные экземпляры класса Arm и передает свойства экземпляров классов TargetPoint и Arm в модуль “FABRIK”. Модуль “FABRIK” передает новые свойства экземпляров класса в модуль Angles. Angles возвращает новые свойства.

Согласно принятому циклу разработки после разработки архитектуры эмулятора необходимо выполнить тестирование программных средств.

4. Тестирование разработанных программных средств

Для тестирование программных средств разрабатываемого эмулятора разработаны тест-комплекты. Тест-комплекты представляют собой объединение тест-кейсов по признаку одинакового объекта тестирования. Тест-кейс это описание функции некоторого функционала относительно частного случая, который является вход/выходными данными функции. В простейшем случае тест-кейс содержит только ожидаемый результат и идентификатор (ID). Тест-кейсы, которые используются в рамках разработки программных средств эмулятора, имеют структуру из трёх элементов помимо ожидаемого результата: шаги - это описание того, что необходимо выполнить; исполнение шагов - описание самого действия; ожидаемый результат – результат , который должен быть получен при выполнении действий описанных шагов; фактический результат - это получившийся результат. Результат выполнения тест-кейса – это сравнение ожидаемого результата с фактическим.

Для составления тест-комплекта тест-кейсов необходимо определить основные, базовые функции, входящие в функционал тестируемых программных средств. В случае с эмулятором это: импорт модели; работа с данными модели; визуализация. Входные данные для всех тест-кейсов основаны на модели и описании некоторого манипулятора, для этого используем конкретный манипулятор, физическая конструкция которого была реализована в рамках дипломной работы проекта по созданию и программированию манипуляторов. Вид этого манипулятора представлен на рисунке 12:

|  |
| --- |
| D:\alexey\Загрузки\manipulator.jpg |
| Рисунок 12 - Фотография манипулятора |

Чтобы использовать данные манипулятора, фотография которого представлена на рисунке 12, его необходимо идентифицировать. Для этого необходимо выполнить классификацию, нарисовать кинематическую схему и составить ДХ-представление данного манипулятора. Классификация манипулятора представленная в таблице 2 составлена согласно классификации, введённой ранее в разделе 2.1

Таблица 2 – идентификация манипулятора согласно введённой классификации

|  |  |  |  |
| --- | --- | --- | --- |
| грузоподъёмность | сверхлёгкий | тип приводов | электрический |
| количество рабочих органов | с одним рабочим органом | характер выполняемых операций | вспомогательный |
| топология кинематической цепи | простая разомкнутая | степень специализации | универсальный |

Кинематическая схема манипулятора представлена на рисунке 13

|  |
| --- |
| D:\alexey\Dropbox\Учёба\Спииран\Манипулятор\3Demulator\kinematicscheme1.jpg |
| Рисунок 13 - кинематическая схема манипулятора |

На рисунке 13 видно, что все кинематические пары в исследуемом манипуляторе - вращательные, следовательно вектор обобщённых координат составляет вектор углов. Также на рисунке 13 видно, что угол скрутки для всех звеньев будет равен нулю для любой конфигурации манипулятора. Исходя из анализа кинематической схемы на рисунке 13 было составлено ДХ-представление манипулятора. В таблице 3 опущены постоянно нулевые переменные, такие как *d* и φ, диапазоны изменений угла поворота взяты в соответствии со спецификацией используемых в манипуляторе сервоприводов (RDS3115 [35]).

Таблица 3 - ДХ-представлние манипулятора

|  |  |  |  |
| --- | --- | --- | --- |
| Звенья | α | l | q |
| Звено 0 | 0° | 0 | от плюс 90 до минус 90° включ. |
| Звенья | α | l | q |
| Звено 1 | 0° | 1 | от плюс 90 до минус 90° включ. |
| Звено 2 | 0° | 1 | от плюс 90 до минус 90° включ. |
| Звено 3 | 0° | 1 | от плюс 90 до минус 90° включ. |

Исходя из описания манипулятора, в пакете Blender3D была составлена упрощённая 3D модель изображённая на рисунке 14.

|  |  |  |
| --- | --- | --- |
| D:\alexey\Dropbox\manipulatorPerspectiveView.png | D:\alexey\Dropbox\manipulatorFrontView.png | D:\alexey\Dropbox\manipulatorUpView.png |
| а) | б) | в) |
| Рисунок 14 - 3D модель манипулятора: а - перспектива; в - вид спереди;  б - вид сверху | | |

Модель на рисунке 14 является универсальной для ряда манипуляторов, а именно её можно использовать для любого манипулятора схожей с описываемым классификацией, потому как параметры ДХ-представления в таблице 3 задаются программно и могут быть изменены, не прибегая к помощи редактора 3D графики.

Теперь при достаточном объёме информации о манипуляторе и наличии его 3D модели можно выполнить тестирование при помощи тест-кейсов.

4.1 Тест-кейсы

Всего для тестирования составлено N тест-кейсов, тест-кейсы имеют заранее определённые пункты, обязательные к заполнению, о них говорилось ранее.

4.1.1 Тест-кейс: “Импорт модели”

ID: 1

Шаги:

* открыть Blender3D;
* установить blend4web плагин версии 16.02 для Blender3D версии не ниже 2.77;
* открыть файл “manipulator.blend”, содержащий модель манипулятора;
* выбрать в качестве движка рендеринга Blend4Web;
* выполнить экспорт модели в формат Blend4Web(.json);
* запустить эмулятор манипулятора, открыв index.html в веб-браузере;
* выбрать появившемся меню пункт “Новый манипулятор”;
* в меню загрузки данных модели выбрать .json и .bin файлы, полученные в результате экспорта;
* нажать “Далее”;
* в выпадающем меню выбрать 3D объекты;
* нажать “Готово”.

Исполнение шагов:

* в редакторе 3D графики Blender3D выбран движок рендеринга Blend4Web;
* выполнен экспорт модели в формат Blend4Web(.json);
* загружена Web страница с эмулятором;
* в меню выбран пункт “Новый манипулятор”;
* загружены manipulator.json и manipulator.bin;
* в качестве сочленений из выпадающего списка выбраны 3D объекты Node\_0, Node\_1, Node\_2, Node\_3, Node\_4, а в качестве TargetPoint выбран 3D объект TargetPoint;
* выполнен запуск эмулятора.

Ожидаемый результат: в окне веб-браузера выполнится рендеринг 3D модели манипулятора, созданной в редакторе Blender3D.

Фактический результат: выполнен рендеринг 3D модели, представленный на рисунке 15

|  |
| --- |
| initial.jpg |
| Рисунок 15 - Визуализация 3D модели манипулятора в эмуляторе |

Вывод: фактический результат соответствует ожидаемому, значит эмулятор прошёл тест-кейс “Импорт модели” успешно.

4.1.2 Тест-кейс: “Идентификация”

Тест-кейс “Идентификция” выполняется с загруженной в эмулятор 3D моделью манипулятора.

ID: 2

Шаги:

* вызвать информационную панель манипулятора, через кнопку обозначенную текстом “Манипулятор” и изображением знака вопроса;
* выбрать в списке “Идентификация”;

Исполнение шагов:

* вызвана информационная панель;
* выбран пункт “Идентификация”.

Ожидаемый результат: в результате на экран должна быть выведена матрица ДХ-представления манипулятора, в виде таблицы, идентичной таблице 3

Фактический результат:

Таблица 4 - ДХ-представление манипулятора, полученное при помощи эмулятора

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Звенья | α | φ | l | d | Диапазон изменений q |
| Звено 0 | 0° | 0 | 1 | 0 | - |
| Звено 1 | 0° | 0 | 1 | 0 | - |
| Звено 2 | 0° | 0 | 1 | 0 | - |
| Звено 3 | 0° | 0 | 1 | 0 | - |

Сравнение: фактический результат не соответствует ожидаемому, при помощи эмулятора не получилось составить полное ДХ-представление манипулятора. Это связано с тем, что 3D модель не содержит информации об ограничениях объекта моделирования. Эмулятор прошёл тест-кейс “Идентификация” идентификация не полностью. Прохождение тест-кейса считается успешным только в том случае, когда ожидаемый результат совпадает с фактическим, значит тест-кейс “Идентификация” не успешен.

4.1.3 Тест-кейс: “Прямая задача кинематики”

ID: 3

Шаги:

* выбрать пункт с текстом “Прямая задача манипулятора”;
* указать XYZ углы Эйлера для каждого звена;
* нажать кнопку “Решить”;

Исполнение шагов;

* выбран пункт “Прямая задача манипулятора”;
* для каждого звена в соответствии с таблицей 5 задать углы поворота;
* нажать кнопку “Решить”.

Таблица 5 – входные данные, задание углов для решения прямой задачи кинематики манипулятора

|  |  |  |  |
| --- | --- | --- | --- |
| Звенья | Углы поворота Эйлера XYZ | | |
| X | Y | Z |
| Звено 0 | 1° | 18° | 4° |
| Звено 1 | -4° | 5° | 20° |
| Звено 2 | 1° | 28.612° | 3.779° |
| Звено 3 | -73.908° | -29.816° | -13.656° |

Ожидаемый результат:

Таблица 6 - координаты точек, соответствующие положению сочленений манипулятора

|  |  |  |  |
| --- | --- | --- | --- |
| Сочленения | x | y | z |
| Сочленение 0 | 0 | 0 | 0 |
| Сочленение 1 | -0.04112 | -0.03051 | 0.99869 |
| Сочленение 2 | -0.3389 | 0.08059 | 1.94684 |
| Сочленение 3 | -0.83648 | -0.93051 | 2.12018 |
| Сочленение 4 | -0.93039 | 0.65117 | 3.28095 |

Фактический результат: идентичен значениям из таблицы 6

Сравнение: ожидаемый результат совпал с фактическим, тест-кейс пройден успешно.

4.1.4 Тест-кейс: “Обратная задача кинематики”

ID: 4

Шаги:

* выбрать пункт с текстом “Обратная задача манипулятора”;
* указать координаты TargetPoint в предложенном поле ввода для координат x,y и z;
* нажать кнопку “Решить”;
* получить решение.

Исполнение шагов

* выбран пункт “Обратная задача манипулятора”;
* для TargetPoint указан вектор [1,1,-2], для x,y и z координат соответственно;
* нажата кнопка “Решить”;
* получено решение.

Ожидаемый результат:

* новые координаты сочленений соответствующие координатам в таблице 7;
* углы поворота в градусах для каждого звена равны углам в таблице 8;
* отображение 3D модели манипулятора в соответствии с новыми данными.

Таблица 7 - координаты точек, соответствующие положению сочленений манипулятора, полученные при помощи эмулятора

|  |  |  |  |
| --- | --- | --- | --- |
| Сочленения | x | y | z |
| Сочленение 0 | 0 | 0 | 0 |
| Сочленение 1 | -0,0411167554557 | -0,0305058732628 | 0,9986885786056519 |
| Сочленение 2 | -0,338895946741 | 0,08059246093034744 | 0,9986885786056519 |
| Сочленения | x | y | z |
| Сочленение 3 | -0,836477816104 | 0,9305116534233093 | 2,1201846599578857 |
| Сочленение 4 | -1 | 2 | 2 |

Таблица 8 - углы поворота звеньев, вычисленные в соответствии с новыми координатами из таблицы 7

|  |  |  |  |
| --- | --- | --- | --- |
| Звенья | Углы поворота Эйлера XYZ | | |
| X | Y | Z |
| Звено 0 | - | - | - |
| Звено 1 | 1,646° | -0,017° | -1,602° |

Продолжение таблицы 8

|  |  |  |  |
| --- | --- | --- | --- |
| Звенья | Углы поворота Эйлера XYZ | | |
| X | Y | X |
| Звено 2 | -6,546° | -0,296° | -17,177° |
| Звено 3 | -79,939° | 28,612° | 3,779° |

Фактический результат: фактический результат аналогичен таблице 7 и таблице 8.

Сравнение: фактический результат совпадает с ожидаемым, следовательно тест-кейс успешен.

Помимо тест-кейсов, тестирующих функционал необходимо проводить тестирование программной реализации. Тестирование программной реализации на предмет производительности в целом или классов, а также отдельных функций называют профилированием, профилирование позволяет выявить “слабые” участки кода, получив данные для дальнейшего анализа. Под профилем понимается определённый набор выходных данных по состоянию системы на момент проведения измерений. Профили позволяют выявить динамику изменения профилируемой системы относительно замеряемых параметров.

4.2 Профилирование разработанных программных средств

Для профилирования использовались “Инструменты разработчика FireFox”, имеющие обширные возможности для выполнения профилирования с последующей визуализацией и экспортом полученных данных для анализа, а также инструмент быстрого сравнения двух получившихся профайлов. Инструменты профилирования как правило входят в инструменты разработки, предоставляемые браузерами. Аналоги инструментов разработчика FireFox других, популярных браузерах [36]:

* Internet explorer (IE) – «Средства разработчика» ;
* Google Chrome (Chrome) – «Инструменты разработчика Chrome»;
* Opera – «Средства для разработчика», представлены на рисунке 18.

Выбор “Инструментов разработчика FireFox” как инструмента профилирования субъективен, основываясь на предпочтениях в дополнительном функционале визуализации, таком как отображение вызовов функций Waterfall(“Водопад”), представленном на рисунке 19.

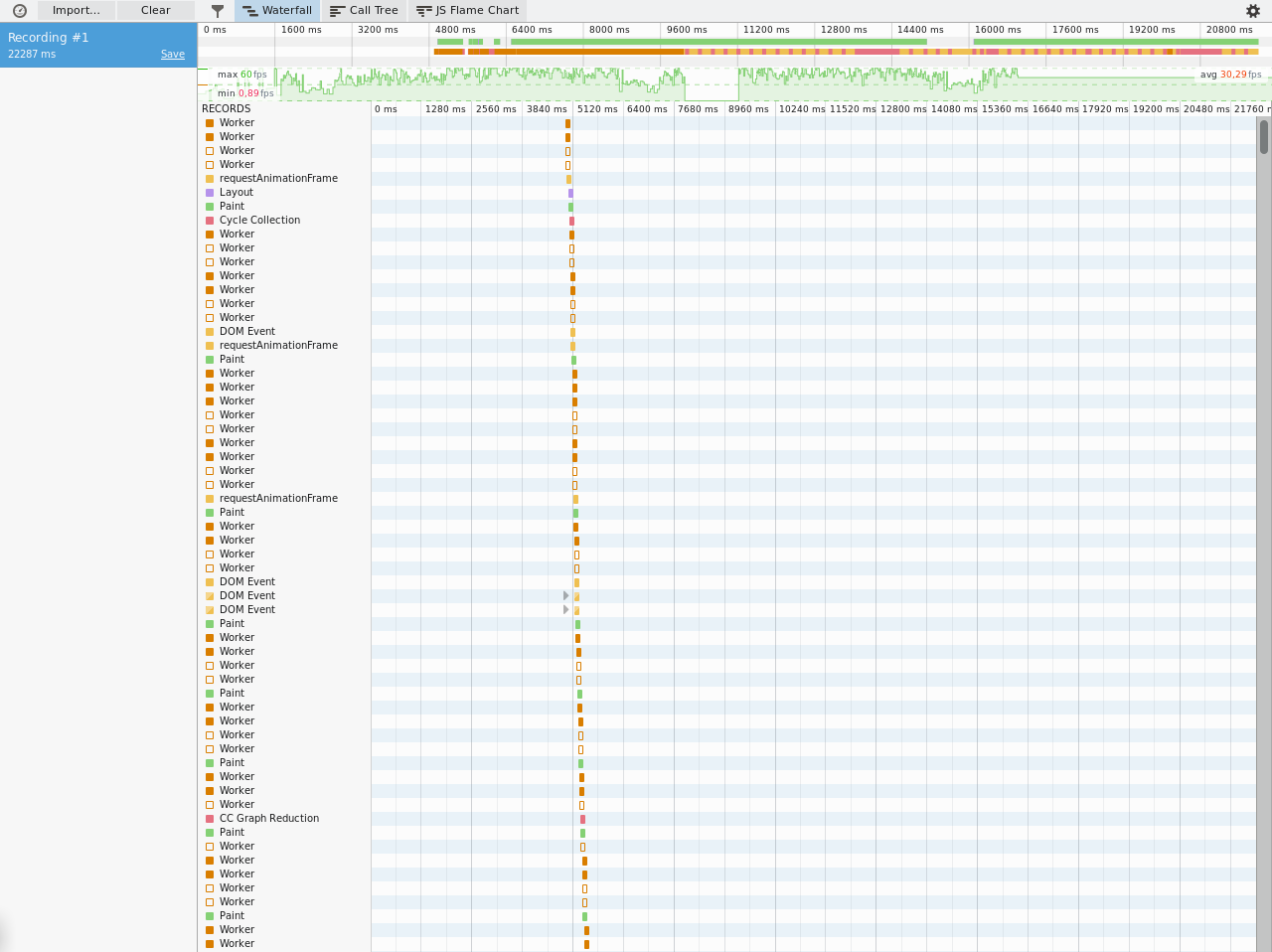


Рисунок 19 - средство визуализации инструментов для разработчика FireFox “Водопад”

Отображение “Водопад” позволяет наблюдать порядок запуска, а также выполнять быстрый поиск вызовов, снижающих быстродействие.

4.2.1 Анализ времени выполнения

В соответствии с тремя режимами работы управляющей программы эмулятора, о которых говорилось в разделе 3.2, для каждого из режимов работы выполнено профилирование, в результате которого создаётся три профиля:

* Время выполнение демонстрация;
* Время выполнения прямая задача кинематики;
* Время выполнения обратная задача кинематики

В профиле содержится информация об исследуемых параметрах всех скриптов выполняемых на странице. Также профиль содержит информацию о времени выполнения скриптов, к которым программный код эмулятора имеет только косвенное отношение, например внутренние скрипты браузера для парсинга и рендеринга HTML кода. В таблице 9 приведены данные пяти профилей, четыре из которых получены измерением времени от начала загрузки HTML страницы до полного выполнения управляющей программы, а также эталонное измерение, в качестве которого выступают данные профиля полученные загрузкой страницы без вызова управляющей программы эмулятора манипулятора.

Таблица 9 - профилирование времени выполнения управляющих программ манипулятора

|  |  |
| --- | --- |
| Профиль | Данные измерения времени, мс |
| Время выполнение демонстрация | 12462 |
| Время выполнения прямая задача кинематики | 77,62 |
| Время выполнения обратная задача кинематики | 80,67 |
| Эталонное измерение | 74,47 |

Относительно долгое выполнение управляющей программы связано с реализацией демонстрации, динамическим решением обратной задачи кинематики на каждой итерации. Исходя из таблицы 9, путём вычитания измеренного времени из эталонного замера времени на решения прямой и обратной задачи кинематики можно выявить, какое количество времени тратится на каждую из задач кинематики, а именно 3,15 и 6,26 мс соответственно.

В качестве предварительных рекомендаций на основе профилирования при следующей итерации разработки необходимо отказаться от динамической демонстрации в пользу анимации, что должно дать прирост в скорости порядка 10000%, предположение основывается на эталонном измерении.

4.2.2 Анализ потребляемой памяти

Аналогично пункту 4.2.1 измерение потребляемой памяти будет производится для четырех случаев с соответствующим созданием четырёх профилей:

* Потребляемая память - демонстрация;
* Потребляемая память - прямая задача кинематики;
* Потребляемая память - обратная задача кинематики;
* Эталонное измерение

Результаты измерений представлены в таблице 10.

Таблица 10 - профилирование потребляемой памяти во время выполнения управляющих программ манипулятора

|  |  |
| --- | --- |
| Профиль | Данные измерения времени, мс |
| Потребляемая память - демонстрация | 62.11 |
| Потребляемая память - прямая задача кинематики | 61.9 |

Продолжение таблица 10

|  |  |
| --- | --- |
| Профиль | Данные измерения времени, мс |
| Потребляемая память - обратная задача кинематики | 62 |
| Эталонное измерение | 61.14 |

Исходя из данных в таблице 8 разница в потреблении памяти во время выполнения различных управляющих программ эмулятора различается не сильно, значит утечек памяти в процессе выполнения управляющих программ не происходит.

# 

ЗАКЛЮЧЕНИЕ

В бакалаврской работе дано описание предметной области, а также описание объекта предметной области и его классификация в рамках общепринятых понятий и определений. В результате выполнения бакалаврской работы был проведён анализ современных средств, аналогичных по функционалу с разрабатываемым в рамках данной работы. Был выведен ряд требований, предъявляемых к программным средствам эмуляции. С использованием современного подхода к разработке программного обеспечения на базе готовых решений был выбран фреймворк для делегирования некоторых задач, непосредственно не связанных с эмуляцией манипулятора на основе фреймворка.

В результате анализа объекта предметной области робота-манипулятора была разработана архитектура эмулятора и UML диаграммы, описывающие программную реализацию эмулятора. Также были выделены режимы работы, которые были реализованы в рамках созданной архитектуры. В ходе данной работы было выполнено тестирование разработанных программных средств с использованием тест-кейсов и профилирования. В результате выполнения тест-кейсов два из четырёх были выполнены успешно. Однако если проанализировать расхождения ожидаемого результата с фактическим можно сделать вывод, что изменения, необходимые для достижения идентичности - незначительны, а значит должны и будут выполнены при следующей итерации разработки. Профилирование показало, что большую часть времени выполнения программы составляет первичный и последующий рендеринг сцены. А потребление памяти в свою очередь незначительно меняется в зависимости от выбранного режима работы управляющей программы эмулятора, следовательно теоретически можно добиться увеличения скорости выполнения циклов работы эмулятора за счёт увеличения потребляемой памяти.

Исходя из результатов прохождения тест-кейсов и выполненного профилирования на следующей итерации разработки будет дополнена функциональность структурных единиц модулей, не связанных с движком. Также будет выполнена оптимизация демонстрационной управляющей программы в соответствии с результатами профилирования.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. *Климчик А. С., Гомолицкий Р. И., Фурман Ф. В., Семкин К. И.* Разработка управляющих программ промышленных роботов/ Белорусский государственный университет информатики и радиоэлектроники. Минск, 2008. 131 с.
2. *Зенкевич С. Л., Ющенко А. С.* Управление роботами. Основы управления манипуляционными роботами: учеб. для вузов*.* М.: Изд-во МГТУ им. Н.Э. Баумана, 2000. 400 с.
3. WorldRobotics URL: http://www.worldrobotics.org/index.php?id=home&news\_id=289 (дата обращения: 20.03.2016)
4. *Егоров В. Б., Голубков П. С. АНАЛИЗ РАЗВИТИЯ ПРОМЫШЛЕННОЙ РОБОТОТЕХНИКИ В МИРЕ //Автоматизация технологических бизнес-процессов. – 2015. – Т. 1. – №. 22.*
5. *Юревич Е.* Основы робототехники. 2-е изд. СПб: БХВ-Петербург, 2005. 420 с.
6. *Robinson, P.* Robotics Education and Training: A Strategy for Development// Industrial Robot. 1996. vol. 23(2). pp. 4-6.
7. Основы робототехники/ *H. B. Василенко К. Д., Никитин В. П., Пономарев, А. Ю. Смолин*. Томск: МГП «РАСКО», 1993. 480 с.
8. *Кондратьева М. Н.* Экономика предприятия: учеб. пособие для студентов высших учебных заведений. Ульяновск : УлГТУ, 2008. 241 с.
9. Омар М. и др. Анализ современного состояния развития интеллектуальных роботов //АвтоматизаRobotic Modelling and Simulation: Theory and Application, Robot Manipulators Trends and Development, Agustin Jimenez and Basil M Al Hadithi (Ed.)/ M.I. Jambak, H. Haron, H. Ibrahim, N. A. Hamid, ISBN: 978-953-307-073-5, InTech, DOI: 10.5772/9185, 2010, http://www.intechopen.com/books/robot-manipulators-trends-and-development/robotic-modelling-and-simulation-theory-and-application
10. G. Echeverria, N. Lassabe, A. Degroote, S. Lemaignan. Modular open robots simulation engine: MORSE // Robotics and Automation (ICRA), 2001 IEEE International Conference on, Shanghai, 2001. pp. 46-51.  
    ция и управление в технических системах. – 2015. – №. 4. – С. 48-54.
11. Robotics simulator [Электронный ресурс]. - https://en.wikipedia.org/wiki/Robotics\_simulator . - (дата обращения: 14.04.2016).
12. Lutz M. Python pocket reference. – " O'Reilly Media, Inc.", 2014.
13. Nelli F. Introduction to the Python’s World //Python Data Analytics. – Apress, 2015. – С. 13-34.
14. Dumon W., Deforche K. Wt: A web toolkit //DR DOBBS JOURNAL. – 2008. – Т. 33. – №. 3. – С. 55-59.
15. Zakai A. Emscripten: an LLVM-to-JavaScript compiler //Proceedings of the ACM international conference companion on Object oriented programming systems languages and applications companion. – ACM, 2011. – С. 301-312
16. Ierusalimschy R. Programming in lua. – Roberto Ierusalimschy, 2006.
17. Aroca R. V., Gardiman R. Q., Gonçalves L. M. G. Web-based robot programming environment and control architecture //Robotics Symposium and Latin American Robotics Symposium (SBR-LARS), 2012 Brazilian. – IEEE, 2012. – С. 27-32.
18. Мацуда К., Ли Р. WebGL: программирование трехмерной графики. – Litres, 2015.
19. Optimal Usage of Robot Manipulators, Robot Manipulators Trends and Development, Agustin Jimenez and Basil M Al Hadithi (Ed.)/ B. Kamrani, V. Berbyuk, D. Wäppling, etc., ISBN: 978-953-307-073-5, InTech, DOI: 10.5772/9198, 2010 <http://cdn.intechopen.com/pdfs-wm/10489.pdf>
20. A. Gachet Software Frameworks for Developing Decision Support Systems – A New Component in the Classification of DSS Development Tools// Journal of Decision Systems. 2003. vol. 12(3). pp. 271–281.
21. Creighton R. H. Unity 3D Game Development by Example: A Seat-of-Your-Pants Manual for Building Fun, Groovy Little Games Quickly. Packt Publishing Ltd, 2010.
22. Трехмерные решения для сайтов [Электронный ресурс]. - <https://www.blend4web.com/ru/> . - (дата обращения: 10.04.2016).
23. three.js - Javascript 3D library [Электронный ресурс]. - http://threejs.org/ . - (дата обращения: 01.04.2016).
24. BabylonJS - 3D engine based on WebGL/Web Audio and JavaScript [Электронный ресурс]. - http://www.babylonjs.com/ . - (дата обращения: 02.04.2016).
25. Brito A. Blender 3D 2.49 Architecture, Buildings, and Scenery. Packt Publishing Ltd, 2010
26. Д. Флэнаган JavaScript. Карманный справочник. Вильямс, 2015. 320 с.
27. GPL License [Электронный ресурс]. URL: http://www.gnu.org/licenses/ Режим доступа: свободный (дата обращения: 28.03.2016)
28. Грабченко А. И. и др. Введение в мехатронику. – 2014
29. Dobra A. General classification of robots. Size criteria //Robotics in Alpe-Adria-Danube Region (RAAD), 2014 23rd International Conference on. – IEEE, 2014. – С. 1-6.
30. Ларман К. Применение UML 2.0 и шаблонов проектирования //М.: Вильямс, 2006.–736 с. – 2007.
31. International Organization for Standardization. ISO/IEC 19501: information technology-open distributed processing-unified modeling language (UML) version 1.4. 2. – ISO, 2005.
32. Auer M., Tschurtschenthaler T., Biffl S. A flyweight uml modelling tool for software development in heterogeneous environments //Euromicro Conference, 2003. Proceedings. 29th. – IEEE, 2003. – С. 267-272.
33. Aristidou A., Chrysanthou Y., Lasenby J. Extending FABRIK with model constraints //Computer Animation and Virtual Worlds. – 2015.
34. Aristidou A, Lasenby J. FABRIK: A Fast, Iterativesolver for the Inverse Kinematics Problem// GraphicalModels. 2001. vol. 73(5). pp. 243–260.
35. Khow H. W. Design and Development of a Brain Computer Interface Controlled Robotic Arm : дис. – UTAR, 2015.
36. Desktop Browser Market Share [Электронный ресурс]. URL: http://www.netmarketshare.com/browser-market-share.aspx?qprid=0&qpcustomd=0&qpsp=2015&qpnp=2&qptimeframe=Y Режим доступа: свободный (дата обращения: 01.05.2016)