Создание системы динамического взаимодействия с объектами дополненной реальности

Калинин Даниил Евгеньевич, Радькин Кирилл Алексеевич

МАОУ «Лицей №97 г. Челябинска», 10м1 класс

Научный руководитель:

Саканов Дамир Муратович

педагог дополнительного образования

МАОУ «Лицей №97 г. Челябинска»

Работа к защите допущена \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ // Саканов Дамир Муратович

Оглавление

[Введение 4](#_Toc8617221)

[Глава 1. Изучение литературы 6](#_Toc8617222)

[1.1. Разработка в Processing 6](#_Toc8617223)

[1.2. Разработка в Unity-3D 9](#_Toc8617224)

[1.3. Изучение способов отслеживания перемещения 11](#_Toc8617225)

[1.3.1. Акустические методы 12](#_Toc8617226)

[1.3.2. Радиочастотные методы 12](#_Toc8617227)

[1.3.3. Магнитные методы 12](#_Toc8617228)

[1.3.4. Оптические методы 13](#_Toc8617229)

[1.3.5. Инерциальный трекинг 14](#_Toc8617230)

[1.3.6. Выбор способа 14](#_Toc8617231)

[Глава 2. Разработка принципа работы модели 16](#_Toc8617232)

[2.1. Общее представление о работе модели 16](#_Toc8617233)

[2.2. Выбор оборудования 17](#_Toc8617234)

[2.2.1. Микроконтроллер Arduino 17](#_Toc8617235)

[2.2.2. Модуль трекинга 18](#_Toc8617236)

[2.3. Выбор способа передачи данных 19](#_Toc8617237)

[Глава 3. Математическая часть 21](#_Toc8617238)

[3.1. Накопление ошибки в расчетах 21](#_Toc8617239)

[3.2. Результат работы с датчиком 22](#_Toc8617240)

[Глава 4. Написание программного обеспечения 23](#_Toc8617241)

[4.1. Программирование на Unity и рендеринг модели на экране 23](#_Toc8617242)

[4.1.1. Динамическое позиционирование модели на изображении−метке 23](#_Toc8617243)

[4.1.2. Перемещение объекта посредством управления виртуальным джойстиком 25](#_Toc8617244)

[4.1.3. Объединение 27](#_Toc8617245)

[4.2. Реализация общения микроконтроллера и Unity 27](#_Toc8617246)

[4.2.1. Настройка последовательного порта 27](#_Toc8617247)

[4.2.2. Подключение ком-порта к основному проекту. 28](#_Toc8617248)

[4.3. Совместная работа двух программ 31](#_Toc8617249)

[Заключение 33](#_Toc8617250)

[Список литературы 34](#_Toc8617251)

[Приложение 35](#_Toc8617252)

# Введение

Прогресс человечества растет с невероятной скоростью. Прямо сейчас по всему миру происходит столько же научных открытий, сколько произошло за весь XV век. Ежеминутно создаются новые системы, призванные упростить повседневную жизнь человека. В качестве примера, приведем несколько таких систем. IOT[[1]](#footnote-1)-устройства, расположенные как в доме, так и в промышленном производстве, могут послужить хорошим примером, поскольку получили широкое распространение относительно недавно. Такие устройства позволяют осуществлять автоматический сбор и анализ данных, например, IOT-холодильник способен автоматически заказывать в интернет-магазине, а IOT-склад способен автоматически вести учет товара. Еще одним примером может стать технология компьютерного зрения и машинного обучения в целом. Данная система хоть и была разработана довольно давно, в жизнь обычного человека она вошла так же недавно, как и предыдущая. Данная технология используется для анализа больших баз данных и выведения закономерностей. Подборка фильмов, из учета ваших интересов, распознавание номера автомобильного знака по фото, сделанным камерой контроля скорости, распознавание лица для разблокировки смартфона – это алгоритмы машинного обучения. И последняя в списке примеров, но далеко не последняя по значимости технология – технология AR[[2]](#footnote-2)– дополненной реальности. Технология представляет дополнение нашей с вами реальности (отсюда и название) различными объектами из виртуального мира. От технологий, приведенных выше, эта имеет одно отличие: эта технология почти не используется в нашей повседневной жизни. Эта особенность связана, в первую, по нашему мнению, очередь с тем, что не создано по-настоящему удобной и универсальной системы взаимодействия с AR, а без такой системы, область применения данной технологии урезается мобильными играми и разными развлекательными приложениями. Отсюда вытекает цель нашей работы: разработать и реализовать универсальную интуитивно-понятную систему управления AR-объектами. Актуальность работы заключается в том, что создание такой системы позволит повсеместно распространить AR, а также значительно расширить область применения технологии, кроме того, далее будет рассмотрено, что подобных работ в свободном доступе найдено не было, а, значит, данную работу можно считать уникальной в своем роде. По окончании разработки проекта, все материалы будут выложены в open-source-источники[[3]](#footnote-3) для того, чтобы каждый желающий мог собрать данное устройство, и технология дополненной реальности получила максимально большое распространение.

# Изучение литературы

Перед началом работы необходимо было изучить множество различной литературы. Поскольку наш проект представляет собой систему взаимодействия дополненной реальности и физических объектов, первое, чему было уделено время, – выбор языка программирования и среды разработки для отрисовки объектов дополненной реальности. После тщательного изучения различных IDE[[4]](#footnote-4) и языков программирования, среди которых были, в том числе: С++, Kotlin, Java, C# и Processing, – были отобраны два варианта: Среда Unity-3D или язык программирования Processing. Чтобы использовать в своем проекте наиболее удобный вариант, было решено опробовать обе среды.

## Разработка в Processing

Processing – это подъязык программирования, основанный на java с простым и понятным синтаксисом. Он дает возможность быстро и легко создавать мультимедиа приложения[[5]](#footnote-5). Данный язык используется в основном дизайнерами и художниками. Ниже приведены примеры арт-объектов, созданных, с помощью processing.



Рис. 1. Мультимедийный арт-объект, созданный в processing.



Рис. 2. Мультимедийная динамическая скульптура, созданная в processing.

Разработчики processing создали интерактивный курс, позволяющий изучить основы и синтаксис языка людям, не работавшим ранее с программированием вообще[[6]](#footnote-6). Этот видеокурс удобен тем, что для написания программ, которые используются в качестве примеров, не нужно ничего скачивать. При просмотре курса человеку будет доступно окно ввода кода и вывода результата, код в которых будет помещаться автоматически, в зависимости от того примера, который рассматривается в данный момент в видео, но обучающийся сможет изменять его и наблюдать за результатом.



Рис. 3. Страница видеокурса Processing

Таким образом, видно, что Processing имеет большие возможности в реализации графических приложений, а также прост в обучении. К сожалению, Processing – это только подъязык, иными словами, – обертка для Java, специализирующийся на графике, то есть его возможности в других областях довольно узки. В этом мы убедились на собственном опыте, попытавшись написать AR-приложение со следующим принципом работы: программа получает изображение с веб-камеры, и если на этом изображении присутствует некий маркер – специальное изображение, которое будет распознавать программа, например, купюра определенного достоинства, или уникальный рисунок (вообще, AR-маркером может служить любое изображение, главное – обучить программу отличать его от остального фона) – то программа к изображению с веб-камеры добавляет некую 3D-модель. После позиционирования модель на маркере, она должна следовать за маркером, как если бы была настоящей: при повороте маркера – поворачиваться, при перемещении маркера – перемещаться вслед за ним. В следствии узконаправленности Processing, часть с отрисовкой и перемещением 3D-модели была написана нами без особых трудностей, а часть с добавлением изображения с веб-камеры, распознаванием маркера и вообще внедрением AR вызвала такое множество проблем, что мы решили перейти к Unity-3D, опробовать разработку там, и только в случае неудачи вернуться к Processing, либо при необходимости внедрить часть кода на Processing в наш проект, но, в любом случае, отказаться от него, как от основного средства разработки.

## Разработка в Unity-3D

Следующим нашим шагом был переход к Unity-3D. Unity представляет из себя не конкретный язык программирования, а систему, объединяющую среду программирование скриптов[[7]](#footnote-7), инструменты для дизайна приложения, добавления анимации, и т.д. Данная IDE пользуется успехом среди разработчиков игр и мобильных приложений за простоту использования, многофункциональность и широкую область применения.

|  |  |
| --- | --- |
|  |  |
| Рис. 4.  Иерархия обьектов внутри проекта в Unity | Рис. 5.  Иерархия файлов, доступных для использования |

У системы Unity-3D нет официальных систематизированных обучающих курсов, как у Processing, но, благодаря популярности этой среды разработки, существует множество неофициальных уроков о работе в данной среде. Воспользовавшись одним из таких уроков, мы написали приложение, принцип работы которого изложен в предыдущем параграфе. Вкратце, приложение позиционирует 3D-модель на специальном маркере, а также отслеживает положение маркера и перемещает модель вслед за ним. Подробнее об этом будет сказано далее.



Рис. 6. Функионал Unity на примере нашего приложения.

Стоит упомянуть, что разработка скриптов в Unity осуществляется на языке С# в среде разработки Visual Studio, что несколько упрощает нашу задачу, ведь, как язык, так и среда, довольно популярны в среде разработчиков, поэтому ответы на возникшие вопросы мы наверняка сможем найти в интернете.

## Изучение способов отслеживания перемещения

Отслеживание перемещения (positional tracking) объектов в пространстве – достаточно сложная задача, и, чтобы делать это с достаточно большой точностью, необходимо реализовать сочетание аппаратных средств и программного обеспечения.

Методы решения такой задачи можно поделить на несколько групп:

1. Акустические
2. Радиочастотные
3. Магнитные
4. Оптические
5. Инерциальные
6. Гибридные

Рассмотрим базовые принципы, на которых построены вышеперечисленные методы.

### Акустические методы

Акустические приборы слежения используют высокочастотные волны для определения положения объекта в пространстве. А именно, они замеряют время пролета волны от объекта к одному из датчиков системы, которое пропорционально расстоянию до этого датчика. Измерив эти расстояния, можно однозначно определить положения объекта относительно датчиков.

Акустические трекеры, как правило, имеют низкую скорость обновления, вызванную относительно низкой скоростью звука в воздухе.

### Радиочастотные методы

Методов, основанных на радиоволнах множество, и по принципам работы они схожи с акустическими, только работают с другими типами волн. Также существует система устройств, измеряющие время пролета волны от объекта до датчика. Данный способ, подобно предыдущему, является недостаточно точным (точность порядка сантиметров, что является некомфортным для восприятия человеческим глазом).

### Магнитные методы

Магнитный трекинг[[8]](#footnote-8) основан на измерении интенсивности магнитного поля в различных направлениях. Как правило, в таких системах существует генератор переменного или постоянного магнитного поля.

Точность такого метода может быть достаточно высока в контролируемых условиях, однако в обычных помещениях точность может нарушаться из-за различных проводящих поверхностей или других электронных устройств, например, из-за металлических стержней в блоках железобетона, из которого строятся некоторые здания.

### Оптические методы

Оптические методы представляют собой комбинацию из алгоритмов машинного обучения (в частности, алгоритмы сверточных нейронных сетей, или, как их еще называют, компьютерного зрения) и различных отслеживающих устройств, в роли которых могут выступать различные камеры: видимого или инфракрасного диапазона, стерео-камеры[[9]](#footnote-9) и камеры глубины.

В оптических методах трекинга выделяют два подхода:

* *Outside-in[[10]](#footnote-10)* – в качестве точки отсчета присутствует внешняя камера, определяющая положение объекта по определенным маякам[[11]](#footnote-11), расположенным на нем самом.
* *Inside-out[[12]](#footnote-12)* – обратный метод, точкой отсчета является сам объект, на котором закреплен определенный оптический сенсор, а маяки расположено около отслеживаемого тела.



Рис. 7.Наглядная демонстрация обоих методов трекинга

### Инерциальный трекинг

Современные инерциальные измерительные системы позволяют отслеживать ориентацию и положение в пространстве с большой точностью и минимальными задержками с помощью таких датчиков, как гироскоп и акселерометр.

Акселерометр – прибор, измеряющий проекцию разности истинного ускорения объекта и гравитационного ускорения по каждой из трех осей. Многие акселерометры оснащены электронной составляющей, способной передавать их показания на различные контроллеры.

Гироскоп – устройство, способное реагировать на изменение углов ориентации тела, на котором оно установлено, относительно инерциальной системы отсчета. Термин впервые был введен Ж. Фуко в своем докладе в 1852 году во Французской академии наук.

### Выбор способа

Рассмотрев таким образом, возможные способы отслеживания перемещения, мы пришли к выводу, что для нас наиболее удобен способ инерциального трекинга. Объясним свою позицию. Во-первых, этот способ является достаточно точным, его точность сравнима с точностью оптического метода, который использует большинство корпораций, разрабатывающих VR-шлемы, а именно: HTC, Valve, Oculus VR, Sony PlayStation, Microsoft и другие. Во-вторых, он имеет достаточно высокую частоту опроса[[13]](#footnote-13). В-третьих, это одно из самых дешевых решений в современных реалиях. Существуют специальные модули, которые содержат в себе все датчики, необходимые для инерциального трекинга, один из таких модулей был использован в нашей работе.

# Разработка принципа работы модели

## Общее представление о работе модели

После изучения литературы мы занялись разработкой устройства управления AR-объектами. Для того, чтобы иметь представление о внешнем виде, а также функциях устройства, мы определили для себя критерии, которым это устройство должно соответствовать, а также функции, которое оно должно выполнять. Приведем их ниже.

Итак, устройство должно:

1. Отслеживать свое местоположение.
2. Передавать данные на ЭВМ, а также получать их оттуда.
3. Быть легким.
4. Быть мобильным.
5. Простым в использовании (после настройки и отладки).
6. Unisize[[14]](#footnote-14). То есть, одно и то же устройство могут использовать разные люди, вне зависимости от их размера.

Учитывая данные требования, мы пришли к выводу, что лучше всего будет создать устройство в виде перчатки, надеваемой на руку и содержащей на себе всю необходимую аппаратуру. Таким образом, надевая перчатку на руку, подключая ее к ЭВМ, человек, двигая рукой сможет управлять некоторым AR-объектом.

## Выбор оборудования

Как было сказано выше, для реализации нашей работы нами был выбран способ инерциального трекинга. Соответственно, было необходимо подобрать оборудование, которое бы позволило нам все реализовать.

### Микроконтроллер Arduino

Arduino – торговая марка аппаратно-программных средств для создания различных систем автоматики и робототехники. В частности, Arduino выпускает одноименные микроконтроллеры, линеек которых существует великое множество, приведем самые популярные из них:

1. Uno – классический вариант, один из самых дешевых, но обладающий достаточной производительностью
2. Mega – вариант, обладающий большей, чем Uno мощностью, но и уступающий ему в цене и по размерам
3. Due – самый мощный контроллер, однако обладающий отличной от остальных архитектурой, соответственно не приспособлен для использования некоторых библиотек и работы в связке с другими контроллерами
4. Nano – миниатюрная плата размером около 5-ти сантиметров, используется для проектов, которым не нужны большие мощности

Изначально мы не знали, насколько большими окажутся затраты вычислительных ресурсов у контроллера, поэтому мы решили использовать Arduino Mega, достаточно мощную, имеющуюся у нас на руках, плату, к работе с которой мы привыкли и, пожалуй, все «подводные камни» нам уже известны.



Рис. 8. Arduino Mega

### Модуль трекинга

В качестве модуля трекинга, т.е. датчика для отслеживания перемещений, мы решили использовать плату MPU-9250. Этот модуль содержит в себе такие датчики, как гироскоп, акселерометр и магнитометр.

Магнитометр – прибор для измерения характеристик магнитного поля и магнитных свойств материалов, например, направления поля, градиента поля[[15]](#footnote-15), магнитной индукции, магнитного потока и т.д.



Рис. 9. Модуль MPU-9250

Магнитометры применяются в таких областях, как:

1. Геология
2. Археология
3. Навигация
4. Военная разведка
5. Биология и медицина
6. Сейсмология

Используя данные с этого модуля, мы можем отслеживать перемещение и вращение объекта в пространстве с учетом некоторой погрешности, от которой можно избавиться с помощью программы-фильтра. Вычисление расстояние, с помощью данного модуля будет рассмотрено ниже.

## Выбор способа передачи данных

Очевидно, что, имея внешнее устройство, определяющее свое положение, нам необходимо передавать эти данные на некоторую ЭВМ, где будут производится вычисления. Для осуществления такой операции необходимо определить два параметра: способ и протокол передачи данных.

Основываясь как на изученных источниках информации, так и на личном опыте работы с некоторыми способами передачи данных, мы пришли к выводу, что наиболее простым в реализации будет последовательный com-порт[[16]](#footnote-16).

Последовательный порт – способ проводной передачи данных, при котором данные записываются в пакеты, установленного размера, отправляются на другое устройство, где пакеты распаковываются и данные восстанавливаются в единое целое.

Объясним свой выбор. Во-первых, для реализации общения по ком-порту не нужно никакого дополнительного (и, конечно, дорогостоящего) оборудования. Это существенный плюс, поскольку одна из наших целей – сделать проект общедоступны, т.е. таким, чтобы любой желающий, используя наше программное обеспечение и минимальный перечень оборудования, смог собрать такое устройство у себя дома. Во-вторых, из нашего опыта работы с Bluetooth мы знаем, что при беспроводной передачи данных возникает множество проблем. Например, пакеты с информацией необходимо шифровать, а также разрабатывать фильтры для обработки не до конца переданных пакетов, или для защиты от дублирования информации, что довольно часто возникает, в силу сложности системы. В-третьих, как программная, так и техническая реализация ком-порта намного проще, что также важно для обеспечения возможности повторения проекта.

Стоит отметить, что у способа есть недостатки, но несущественные. Главный недостаток в том, что способ проводной, а устройство должно обладать достаточной мобильностью. Тем не менее, провод всего один, и подобрать его можно достаточно длинным. По отношению ко всем плюсам, недостаток действительно несущественный, поэтому ком-порт нам подходит.

# Математическая часть

Для того, чтобы динамически определять местоположение объекта, нужно динамически вычислять и передавать положение микроконтроллера. Модуль MPU-9250 передает следующие показания по всем трем осям: линейное ускорение, угловую скорость и модуль вектора индукции магнитного поля.

Сразу условимся, что в данной работе мы будем определять относительное местоположение, то есть местоположение относительно предыдущего, поскольку этих данных нам хватит для перемещения объекта.

## Накопление ошибки в расчетах

Итак, первое, что мы сделали – взяли двойной интеграл от значения ускорения по времени. И тут же возникло несколько проблем. Во-первых, как мы поняли экспериментально, у датчика присутствует такой эффект, как дрейф нуля. Суть данного эффекта заключается в том, что из-за некоторых конструкционных и архитектурных особенностей датчика, он передает небольшие показания, порядка 10-20 условных единиц (единица измерения зависит от того, на что мы смотрим: ускорение, угловая скорость или модуль вектора магнитной индукции), даже когда не двигается, находится в клетке Фарадея (клетка Фарадея обладает эффектом исключения электромагнитного поля из внутренней своей части), и вообще в вакууме. Поэтому при взятии интеграла у нас накапливается ошибка, которая в конечном итоге дает довольно сильное отличие от действительного значения перемещения, ведь мы перемещаем датчик всего на несколько сантиметров. Следующая проблема оказалась еще интереснее. Заключается она в следующем. Оси, по которым производит измерение параметров датчик, привязаны к его корпусу, а оси, относительно которых мы хотим получать показания – абсолютны и неподвижны. Таким образом, мы должны при каждом повороте датчика, брать интеграл от угловой скорости по времени, находить угол поворота каждой оси датчика, относительно абсолютных осей координат, перемножать все показания ускорения на косинус полученных углов соответственно. «Ладно» – решили мы и реализовали и такую систему расчета. Как мы и ожидали: она не работала (работала со значительной ошибкой порядка 15-20 сантиметров). Дело оказалось в том, что при повороте датчика направление вектора гравитационного ускорения Земли меняется относительно его внутренней системы координат. Меняется направление вектора – меняется компонента этого вектора в показаниях датчика по каждой из осей, так что и эту компоненту нужно умножать на косинус по всем трем осям, и только потом получать ускорения, по которым можно брать двойной интеграл. В любом случае, при интегрировании накапливается значительная ошибка, которая сильно мешает вычислениям.

При поиске решения в интернете, в одном из электронных ресурсов, нами был найден специальный фильтр, вычисляющий погрешность показаний модуля при его наклоне, а также ошибку при интегрировании, но использующий для этих целей магнитометр. К сожалению, этот фильтр оказался чересчур сложным, и мы не смогли переписать его для конкретно нашего датчика.

## Результат работы с датчиком

В конечном счете, мы написали программу, которая вычисляет местоположение датчика с погрешностью около 40%. Погрешность немаленькая, и, более того, неприемлемая, но, учитывая, что кроме вычитания константы из значения датчика при каждом интегрировании, мы ничего не придумали в силу нехватки матаппарата, то на такой погрешности было решено остановиться.

# Написание программного обеспечения

## Программирование на Unity и рендеринг модели на экране

### Динамическое позиционирование модели на изображении−метке

Как было сказано в введении, цель нашего проекта – создать удобную систему взаимодействия с AR-объектами. Соответственно, первой нашей задачей было научиться генерировать AR-объекты на экране компьютера.

Unity – сложная, многогранная и постоянно развивающаяся среда разработки и, разумеется, среди asset-ов[[17]](#footnote-17) к Unity, нашлось то, что нам необходимо.

.

Рис 10. Интерфейс разработки на Unity

Asset, который мы использовали для нашего проекта, называется Vuforia. С помощью нескольких простых шагов он позволяет создать необходимую нам программу, а именно, алгоритм проецирования выбранной пользователем модели на специальные маркеры, в качестве которых может выступать любое изображение. Чем больше на выбранном изображении различных уникальных элементов, расположенных в разных его частях, тем эффективнее оно будет работать в качестве маркера. При работе с Vuforia, необходимо создавать базы данных, в которых, кроме самой 3D-модели, должно находиться наше изображение-маркер. Vuforia автоматически оценивает эффективность вашего изображения по пятибалльной шкале и показывает вам местонахождение тех уникальных мест по которым будет ориентироваться программа.



Рис. 11. Функционал Vuforia на примере 50-рублевой купюры.

Как мы видим на рисунке выше, изображение 50-рублевой купюра имеет рейтинг 4/5 и достаточно большое количество уникальных точек, которые отмечены желтыми крестиками, что, впрочем, логично, ведь это необходимо для того, чтобы отличить настоящие купюры от поддельных.

Вникнем в логику приложения чуть глубже. Вначале программа получает изображение с веб-камеры на устройстве и просто выводит его на экран. При появлении в области видимости камеры AR-маркера, программа определяет перспективное искажение изображения, на основе чего определяет и ориентацию маркера в пространстве, и затем позиционирует 3D модель на маркер. Далее алгоритм продолжает отслеживать перспективное искажение маркера и передвигать объект в соответствии с положением маркера, т.е. динамически позиционирует объект на AR-метке.

### Перемещение объекта посредством управления виртуальным джойстиком

После разработки динамического позиционирования нужно было дать объекту перемещаться вне зависимости от маркера, т.к. система управления AR-объектами подразумевает взаимодействие с ними с помощью некого устройства, а не посредством передвижения маркера. Для начала, было решено перемещать объект с помощью виртуального джойстика.

Для реализации задуманного мы написали скрипт, который добавили в проект динамического позиционирования. Общая логика скрипта такова. Существуют две встроенные функции: одна из них вызывается только при запуске скрипта, а вторая вызывается при каждом обновлении кадра видеоряда программы. В первой производится настройка программы, углубляться в которую нет смысла в тексте данной работы.[[18]](#footnote-18) Вторая функция считывает отклонение джойстика, записывает пару координат джойстика в трехмерный вектор (трехмерный вектор (x, y, z) был выбран потому, что тогда мы можем перемещать объект во всех плоскостях, записывая в соответствующие позиции вектора координаты отклонения джойстика), после чего умножает вектор на некоторый коэффициент скорости и перемещает объект в соответствии с вектором. Поясним. Например, мы отклонили джойстик от начального положения (координаты (0, 0)) в положение с координатами (x', y'), а затем заполнили вектор перемещения следующим образом: (x', 0, y') – тогда отклонение джойстика вверх повлечет перемещение объекта в направлении от нас, а перемещение вниз повлечет перемещение объекта «к нам». Отклонение джойстика вправо и влево будут перемещать объект в соответствующем направлении. Запись данных в следующей форме: (x', y', 0) – повлечет за собой перемещение объекта в направлении «вверх-вниз» при отклонении джойстика в соответствующем направлении, и перемещение объекта «вправо-влево», как в предыдущей форме записи, при отклонении джойстика «вправо-влево». Таким образом у нас была возможность определять плоскость перемещения объекта.

В итоге, было решено воспользоваться первым вариантом записи – (x', 0, y') – поскольку он являлся интуитивно-понятным при управлении объектом с помощью джойстика. Здесь стоит отметить, что пока что мы можем управлять объектом только в одной плоскости, поскольку джойстик двигается только в двух осях. При управлении непосредственно с помощью устройства, способного перемещаться во всех трех осях, объект, конечно, будет передвигаться во всех возможных плоскостях.

### Объединение

Unity (с англ.) – объединение, объединять. Это слово очень точно отображает суть Unity3D – объединять различные языки программирования, скрипты и т.д., поэтому последним нашим шагом в программировании на Unity стало объединение двух вышеописанных кодов, а именно: рендеринг объекта на определенной метке и движение этого объекта с помощью данных, которые передает Arduino (например, данных о нажатии на кнопки матричной клавиатуры или данных с акселерометра). Получение этих данных, а также их передача описаны ниже.

## Реализация общения микроконтроллера и Unity

### Настройка последовательного порта

Следующим этапом разработки стала настройка ком-порта. Как и в случае с реализацией перемещения модели, мы разбили задачу на несколько простых этапов:

1. В отдельном Unity-проекте настроить общение микроконтроллера Arduino и объектов Unity.
   1. Написать тестовое программное обеспечение на микроконтроллер, которое будет отправлять данные, а также C#-скрипт на Unity, который будет эти данные читать.
2. Протестировать работу приложение, определить оптимальные настройки протокола общения.
3. Используя полученные знания, реализовать односторонний обмен данными между микроконтроллером и Unityв основном проекте.

Следуя данным шагам, мы определили оптимальную задержку чтения (время, между двумя прочтениями данных из порта, работает как фильтр от ошибок формирования пакета, таких, как недоотправка пакета, или дублирование данных, которые хоть и случаются значительно реже в проводном ком-порте, но все же имеют место быть, если не использовать задержку), оптимальную скорость передачи данных, а также решили несколько небольших проблем с кодировками текстовых данных (данные, даже числа, через порт отправляются как текстовая информация), о которых не видим смысла упоминать здесь. Таким образом, мы получили тестовый проект, в котором, изменяя сопротивление потенциометра, подключенного к микроконтроллеру, мы двигали куб в Unity-проекте по принципу: если сопротивление меньше некоторого значения, куб движется вправо, если больше – влево. В основном проекте потенциометр заменит устройство, отслеживающее свое местоположение, а движение куба (или другого объекта) будет осуществляться по более сложным принципам. Пусть данный тестовый проект и выглядит достаточно просто, его реализация дала нам понимание основ работы с последовательным портом на языке C#, что достаточно важно, ведь данный язык является для нас новым и малоизученным. Кроме того, мы получили готовые функции, которые можно просто вставить в наш основной проект.

### Подключение ком-порта к основному проекту.

Одной из самых сложных задач оказалось именно подключение через ком-порт, поскольку не существует практически никаких гайдов[[19]](#footnote-19) по работе с ком-портом на С#. Официальная документация, в которой содержится информация о том, зачем нужны те или иные функции, представляет из себя всего одну страницу с кратким описанием функций, из которой, очевидно, мало всего можно понять. Поэтому мы действовали практически вслепую, довольствуясь несколькими англоязычных видео пятилетней давности.

Расскажем о нескольких проблемах, возникших в ходе связки ком-порта, чтобы не быть голословными. Первое, что необходимо сделать для работы с ком-портом, – это подключить библиотеку[[20]](#footnote-20) System.IO.Ports. Это, казалось бы, простое действие было достаточно сложно совершить, т.к. при подключении библиотеки, Unity не распознавал ее. Все выглядело так, будто бы ее нет в фреймворке[[21]](#footnote-21). Признаться, мы не на шутку запаниковали, когда действительно не нашли этой библиотеки в офлайн-документации Visual Studio. Решение оказалось близко. Дело, что Unity обладает несколькими фреймворками и версиями ядра (машины, которая исполняет написанный алгоритм) для С#. Версия фреймворка (которая, по «счастливому» стечению обстоятельств и стояла у нас) не имела нужной нам библиотеки. Сменив ядро и фреймворк, мы решили проблему.

Следующим шагом нам необходимо создать и настроить объект (упрощенно – переменную) ком-порта (Рис 12).

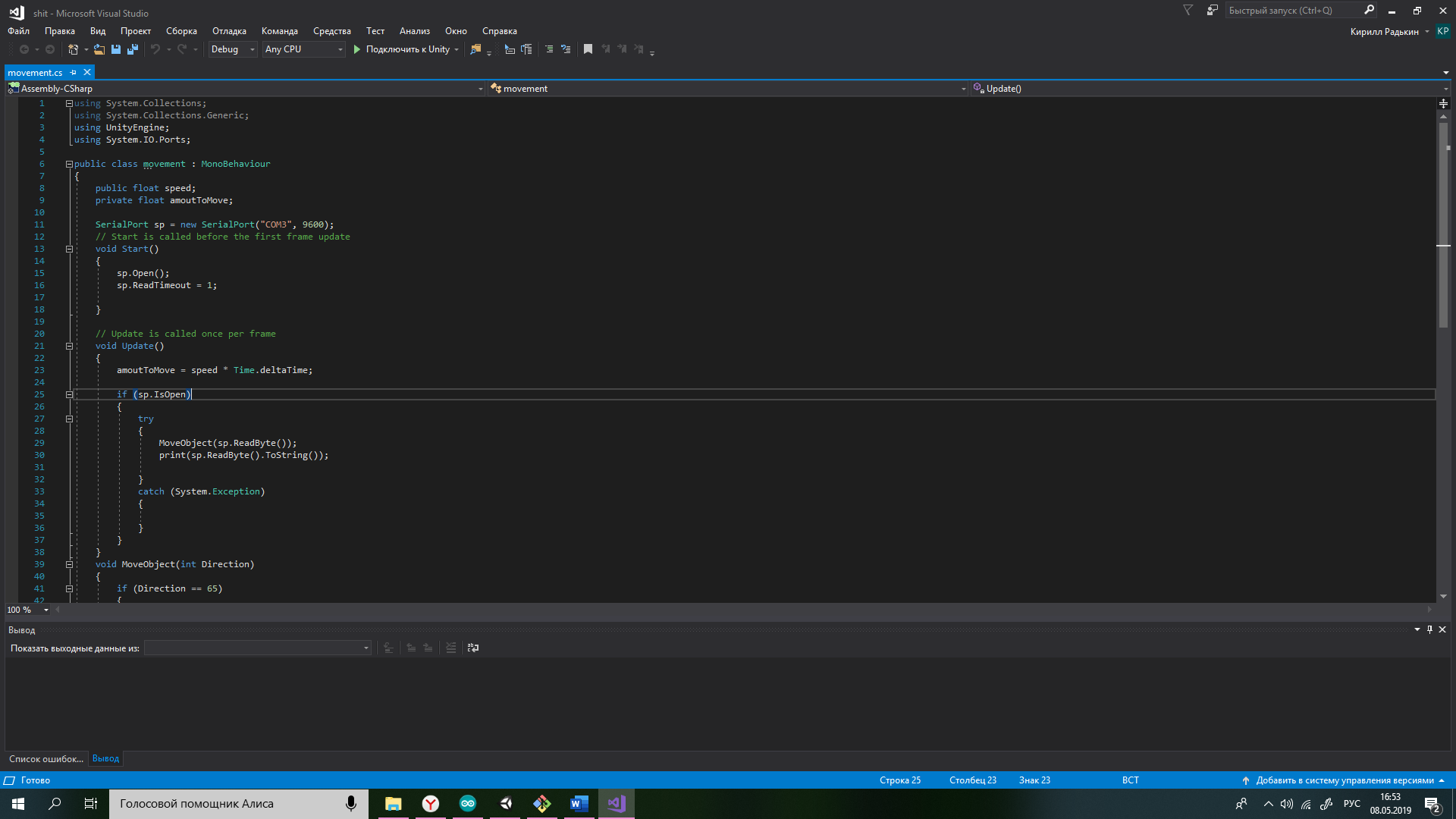


Рис 12. Объявление порта.

Параметры, указанные в скобках, характеризуют скорость передачи данных, которая должна быть одинаковой на обоих устройствах: передатчике и приемнике, – и номер ком-порта, через который происходит обмен. Данные параметры, как говорилось выше, были подобраны нами экспериментально, в тестовом проекте.

Далее нужно было организовать обработку исключений[[22]](#footnote-22). В нашем случае – это проверка на то, есть ли в данный момент какие-то данные для приема. На многих современных языках программирования обработка исключений реализуется с помощью конструкции try-catch. Подробнее можно увидеть на рисунке ниже.

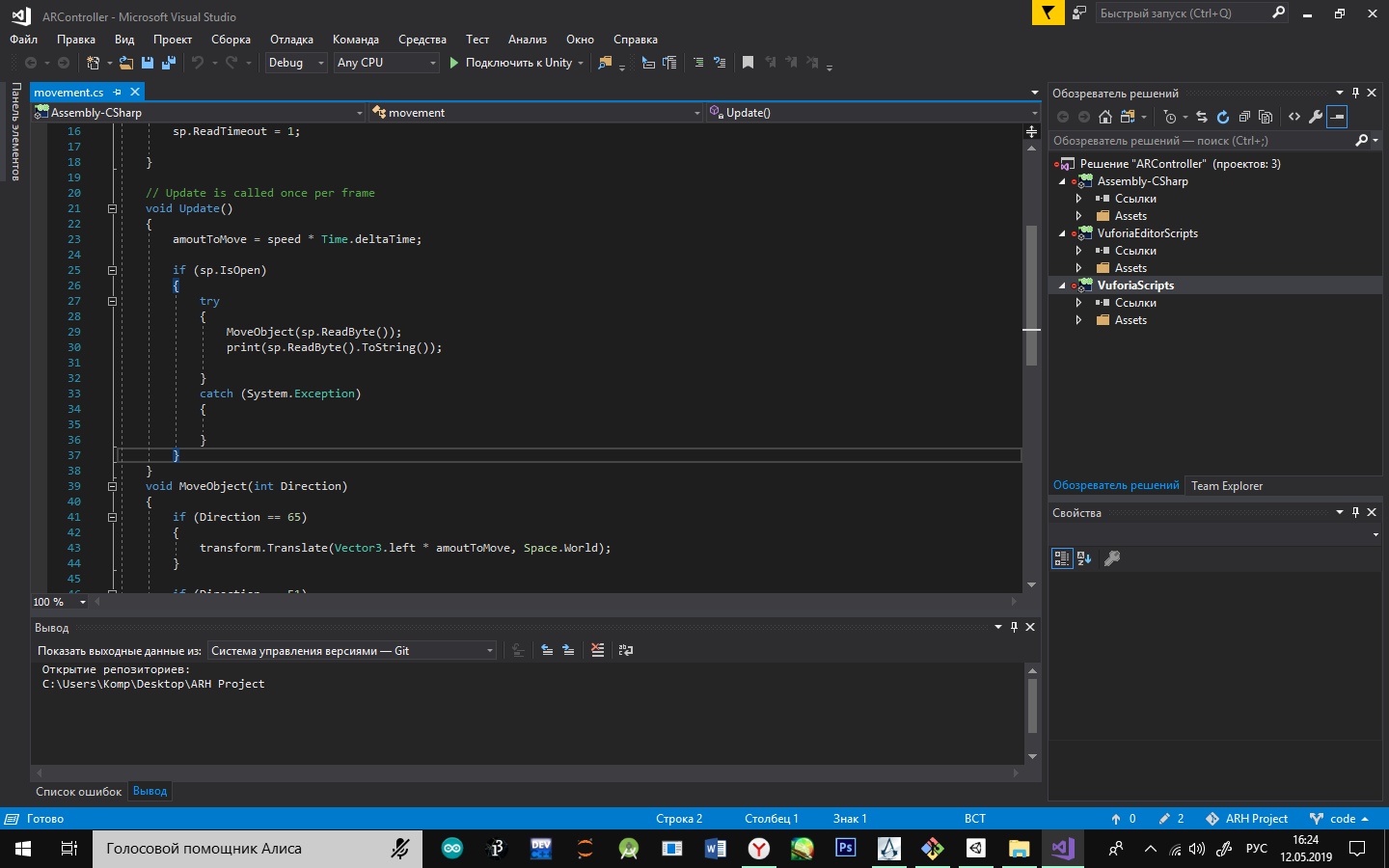


Рис 13. Обработка исключений

Обработка исключений необходима для того, чтобы наш код не пытался получить данные из ком-порта, когда их там нет, т.к. это может вызвать «IOException» (англ. Input-Output Exception) – ошибку ввода-вывода.

И последнее принятое нами решение – это поместить функции (выделено красным на рисунке выше), отвечающие за принятие данных из ком-порта, в конструкцию для обработки исключений. Далее остается лишь использовать принятые данные в своих целях.

## Совместная работа двух программ

Подведем итоги написания программного обеспечения и опишем логику работы всех компонентов вместе. Итак, при запуске программы, она выводит видеоряд, полученный с веб-камеры, после чего переходит в режим поиска AR-маркера. Найдя маркер, программа определяет его положение (угол наклона относительно камеры, угол поворота и т.д.), вычисляя перспективное искажение изображения, и располагает AR-объект соответствующим образом. При перемещении маркера программа будет динамически изменять положение объекта. Далее происходит опрос ком-порта на наличие данных в нем.

Сделаем отступление для рассмотрения работы алгоритма на микроконтроллере Arduino. Программа на микроконтроллере после запуска опрашивает акселерометр, магнитометр и гироскоп. На основании принятых данных производит расчеты, которые были рассмотрены в предыдущей главе, и получает свое относительное положение[[23]](#footnote-23). Затем микроконтроллер передает эти данные по ком-порту, а Unity их получает, и производит дальнейшие манипуляции, описанные ниже. Здесь стоит отметить, что передаваемые данные никак не шифруются по нескольким причинам. Во-первых, мы не сочли нужным шифровать данные, поскольку в передаваемых пакетах не содержится никакой важной или персональной информации, которой можно было бы воспользоваться в случае перехвата пакета. Во-вторых, расшифровка пакета будет занимать дополнительное время, а для нас было очень важным реализовать прием и отправку данных без задержек.

Вернемся к Unity. При появлении в порту данных, алгоритм читает их и переводит в свою кодировку (может так получится, что кодировки на микроконтроллере и на компьютере, где будет работать алгоритм, не совпадут). Затем он переводит полученные данные в систему координат Unity, создает трехмерный вектор, записывает в него новые координаты объекта и перемещает объект в заданную точку, но не мгновенно, а с некоторым параметром скорости, таким образом, что мы интерпретируем это, как привычное нам движение.

# Заключение

В заключение хотелось бы отметить, что мы добились всех поставленных целей. А именно: разработали и реализовали устройство для управления объектами дополненной реальности, разработали программное обеспечение установки, реализовали передачу информации между установкой и электронно-вычислительной машиной, на которой проходила обработка данных. Более того, мы добились главной нашей цели: выложили проект в открытый доступ, а также сделали его доступным по бесплатной некоммерческой лицензии Massachusetts Institute of Technology License. Весь программный код, данная работа, маркеры позиционирования объектов дополненной реальности, а также прочие файлы, непосредственно связанные с созданием проекта выложены на электронном ресурсе GitHub. Кроме вышеуказанного, мы планируем написать статью на электронном ресурсе HabraHabr с кратким содержанием работы, неформальным описанием процесса создания устройства и комментариями для тех, кто захочет данный проект повторить. Таким образом, мы сделали все возможное для того, чтобы проект повторил любой желающий, тем самым внесли свою лепту в развитие прогресса и распространения новых технологий.

# Список литературы

[В Интернете] // Википедия. - 24 Февраль 2019 г.. - 21 Апрель 2019 г.. - https://ru.wikipedia.org/wiki/Стереоскопический\_фотоаппарат.

**Hello, Processing** [В Интернете] / авт. Shiffman Daniel // Processing. - Ben Fry, Casey Reas, 2001 г.. - https://hello.processing.org/.

**Знакомство с Processing 1.0** [В Интернете] / авт. Frexin (@sindrom) // Habrahabr. - 27 Апрель 2009 г.. - https://habr.com/ru/post/58314/.

**Обзор методов и технологий отслеживания положения для виртуальной реальности** [В Интернете] / авт. Сайфуллин Дамир // habr.com. - 20 Сентябрь 2016 г.. - 20 Апрель 2019 г.. - https://habr.com/ru/post/397757/.

# Приложение

1. Программный код Unity, программное обеспечение микроконтроллера и прочие документы, непосредственно связанные с проектом: <https://github.com/DKay7/AR-Hand/tree/code>

  
Рис 1. QR-код для сканирования  
 с ссылкой на программный код работы

1. Internet of things (англ.) – Интернет вещей. [↑](#footnote-ref-1)
2. Augmented reality (англ.) – дополненная реальность. [↑](#footnote-ref-2)
3. Open-source-источники (англ.) – источники свободного доступа. [↑](#footnote-ref-3)
4. IDE – Integrated Development Environment (англ.) – интегрированная среда разработки. Набор средств (специализированные программы) для разработки программного обеспечения. [↑](#footnote-ref-4)
5. (Frexin, 2009) [↑](#footnote-ref-5)
6. (Shiffman, 2001) [↑](#footnote-ref-6)
7. Скрипт (от англ. Script – сценарий) – профессионализм, обозначающий исполняемый файл, содержащий набор инструкций, которым следует программа. [↑](#footnote-ref-7)
8. Трекинг (англ. Tracking – отслеживание) – определение местоположения [↑](#footnote-ref-8)
9. Стереокамеры (или стереоскопический фотоаппарат) – тип фотоаппарата с двумя или более объективами, которые позволяют фотоаппарату симулировать человеческое (бинокулярное) зрение. [↑](#footnote-ref-9)
10. Outside-in (англ.) – дословно переводится как «снаружи – внутри». Здесь и далее профессиональные термины будут использоваться на родном языке с приложением перевода. [↑](#footnote-ref-10)
11. Маяк (здесь и далее) – какие-то характерные отличительные знаки или точки, радиометки, и другие подобные устройства, с помощью которых определяется положение объекта. [↑](#footnote-ref-11)
12. Inside-out (англ.) – дословно переводится как «Внутри – снаружи». [↑](#footnote-ref-12)
13. Частота опроса – величина, показывающая, как часто датчик (или другое любое устройство) может передавать информацию на вычислительную машину. [↑](#footnote-ref-13)
14. Unisize (англ.) – универсальный размер (одежды), т.е подходящий всем людям, независимо от их размера одежды. [↑](#footnote-ref-14)
15. Речь идет о градиент-функции, векторе (матрице, в общем виде), заключающим в себе все частные производные данной функции, т.е. непосредственно показывающей направление к экстремумам функции. [↑](#footnote-ref-15)
16. Сom-порт (англ. Communication – коммуникационный) – профессиональное название последовательного порта. Далее – ком-порт. [↑](#footnote-ref-16)
17. Asset (англ.) – дословно – ценный вклад, достояние. В Unity – созданный кем-то (организацией или одним человеком) набор скриптов и моделей (т.е. готовый пакет) для выполнения определенной задачи [↑](#footnote-ref-17)
18. Код программы содержится на сайте GitHub по ссылке в приложении [↑](#footnote-ref-18)
19. Гайд (от англ. “guide”) – урок, курс, какой-то алгоритм, который помогает людям разобраться в той или иной проблеме. [↑](#footnote-ref-19)
20. Библиотека (проф.) – пакет функций, процедур, классов и т.д., которые не вшиты в программу изначально для экономии памяти. Программист сам выбирает, какие библиотеки ему подключить, в зависимости от того, что должна делать программа. [↑](#footnote-ref-20)
21. Фреймфорк (проф.) – совокупность библиотек, которыми располагает разработчик, т.е. тех, которые он может включить в свой проект. [↑](#footnote-ref-21)
22. Обработка исключений (проф.) – термин в программировании, который обозначает ту область кода, которая отвечает за обработку внештатных ситуаций, из-за которых код может просто зависнуть, а программа перестать работать. [↑](#footnote-ref-22)
23. То есть, насколько переместился микроконтроллер относительно своего предыдущего положения, и в каком направлении произошло это перемещение. [↑](#footnote-ref-23)