

**«Санкт-Петербургский государственный электротехнический университет
«ЛЭТИ» им. В.И. Ульянова (Ленина)»
(СПбГЭТУ «ЛЭТИ»)**

Направление	09.03.04 Программная инженерия
Профиль	Разработка программно-информационных систем
Факультет	КТИ
Кафедра	МО ЭВМ

К защите допустить

Зав. кафедрой

Кринкин К.В.

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
БАКАЛАВРА**

**ТЕМА: РАЗРАБОТКА ПРОГРАММЫ ОПРЕДЕЛЕНИЯ
МЕСТОНАХОЖДЕНИЯ ИГРОКОВ В ВИРТУАЛЬНОМ ФУТБОЛЕ**

Студент		<hr/>	Петруненко Д.А.
		<i>подпись</i>	
Руководитель	к.т.н., доцент	<hr/>	Беляев С.А.
		<i>подпись</i>	
Консультанты	к.э.н., доцент	<hr/>	Косухина М.А.
		<i>подпись</i>	
	к.т.н.	<hr/>	Заславский М.М.
		<i>подпись</i>	

Санкт-Петербург

2021

ЗАДАНИЕ

Зав. кафедрой МО ЭВМ

Кринкин К.В.

« » 2021 г.

Група 7304

Тема работы: Разработка программы определения местонахождения

Место выполнения ВКР: Санкт-Петербургский государственный

Исходные данные (технические требования):

Разработка программы определения местонахождения игроков в

Содержание ВКР:

Обзор и сравнение аналогов, Описание задачи и подходов к решению,

Перечень отчетных материалов: пояснительная записка, иллюстративный

Дополнительные разделы: Разработка и стандартизация программных

Дата представления ВКР к защите

«22» ИЮНЯ 2021 Г.

Петруненко Д.А.

К.Т.Н., доцент

2

КАЛЕНДАРНЫЙ ПЛАН ВЫПОЛНЕНИЯ ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ

Утверждаю
Зав. кафедрой МО ЭВМ
_____ Кринкин К.В.
«__» _____ 2021 г.

Студент Петруненко Д.А.

Группа 7304

Тема работы: Разработка программы определения местонахождения игроков
в виртуальном футболе

№ п/п	Наименование работ	Срок выполнения
1	Обзор литературы по теме работы	22.04 – 29.04
2	Обзор и сравнение аналогов	29.04 – 06.05
3	Описание задачи и подходов к решению	06.05 – 13.05
4	Реализация	13.05 – 20.05
5	Разработка и стандартизация программных средств	20.05 – 22.05
6	Оформление пояснительной записки	22.05 – 25.05
7	Оформление иллюстративного материала	25.05 – 27.05
8	Предзащита	03.06

Студент _____ Петруненко Д.А.

Руководитель к.т.н., доцент _____ Беляев С.А.

РЕФЕРАТ

Пояснительная записка 68 с., 15 рис., 5 табл., 20 источн., 1 прил.

ИНТЕЛЛЕКТУАЛЬНЫЕ АГЕНТЫ, ВИРТУАЛЬНЫЙ ФУТБОЛ, МУЛЬТИАГЕНТНЫЕ СИСТЕМЫ, ПОЗИЦИОНИРОВАНИЕ В УСЛОВИЯХ НЕОПРЕДЕЛЁННОСТИ, ФИЛЬТР КАЛМАНА, ИНЕРЦИАЛЬНАЯ НАВИГАЦИЯ

Объектом исследования являются интеллектуальные агенты в среде проведения международных соревнований по виртуальному футболу.

Цель работы – разработка программы, которая выполняет определение местонахождения агента и осуществляет предсказание координат для игроков недавно исчезнувших из поля зрения.

Выбор подходящего метода для определения местоположения игроков.

Задачи для достижения цели:

- Выбор подходящего метода для определения местоположения игроков.
- Проектирование математической модели.
- Проектирование архитектуры.
- Реализация алгоритма определения местонахождения объектов и прогнозирования местоположения для недавно исчезнувших из поля зрения игроков.
- Проведение экспериментов для определения точности полученного решения.
- Тестирование.

В результате решения поставленных задач был разработан алгоритм определения местонахождения объектов и прогнозирования местоположения для недавно исчезнувших из поля зрения игроков.

Предложенное решение было апробировано в статье Petrunenko D. A., Belyaev S. A. «Determining the Location of Players in Virtual Soccer», журнал «Software Journal: Theory and Applications», в печати.

ABSTRACT

The paper describes the relevance of expanding the information used in predicting the actions of objects in the virtual football environment. The problem of using information about recently disappeared objects from the field of view to assess the situation on the field and make decisions is considered in the paper.

An analysis of available solutions is carried out, based on the results of which a decision was made to begin development. The architecture of the developed system is given, the basic technical solutions are described. The interface of the developed solution is described. The testing of the program is described in the context of development.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	8
1. ОБЗОР ПРЕДМЕТНОЙ ОБЛАСТИ	11
1.1. Введение	11
1.2. Используемые в работе термины	12
1.3. Формальная запись решаемой проблемы	13
1.4. Сравнение существующих подходов к решению проблемы	13
1.4.1. Способы решения задачи	13
1.4.2. Сравнение аналогов по заданным критериям	21
1.5. Выводы	22
2. ОПИСАНИЕ ЗАДАЧИ И ПОДХОДОВ К РЕШЕНИЮ	24
2.1. Введение	24
2.2. Математическая модель решения	24
2.3. Архитектура программы	28
2.4. Предполагаемое решение	30
2.5. Алгоритм инерциальной навигации в условиях недостаточности флагов.....	33
2.6. Алгоритм прогнозирования местоположения исчезнувших из поля зрения объектов.....	34
2.7. Общая структура.....	35
2.7.1. Диаграмма классов.....	35
2.7.2. Хранение предыдущих состояний и прогнозирование местоположения	37
2.7.3. Обработка входных и вычисленных данных, определение местоположения видимых объектов	38
2.7.4. Вспомогательные функции	39
2.7.5. Построение статистики.....	40
2.8. Выводы	40
3. ПРОГРАММНАЯ РЕАЛИЗАЦИЯ	42
3.1. Введение	42
3.2. Используемые технологии.....	42
3.3. Описание реализованного решения.....	42
3.3.1. Функции преобразования входной информации	43
3.3.2. Функции определения местоположения игроков на игровом поле	

3.3.3. Функции прогнозирования местоположения для недавно исчезнувших из поля зрения объектов.....	44
3.3.4. Функции статистических подсчётов	45
3.4. Тестирование.....	45
3.5. Условия эксперимента	46
3.6. Результаты эксперимента	46
3.7. Выводы	51
4. РАЗРАБОТКА И СТАНДАРТИЗАЦИЯ ПРОГРАММНЫХ СРЕДСТВ	52
4.1. Введение	52
4.2. Планирование работ проекта с использованием диаграммы Ганта....	52
4.2.1. Состав частных работ и последовательности их выполнения	52
4.2.2. Определение исполнителей и соисполнителей.....	53
4.2.3. Определение календарных сроков реализации каждой из работ и выполнения всего проекта в целом	54
4.2.4. Диаграмма Ганта	56
4.3. Определение затрат на выполнение и внедрение проекта ПС	58
4.3.1. Расчёт полных затрат в день на работу специалистов	58
4.3.2. Расчет цены предлагаемого продукта	60
4.4. Определение классификационного кода разрабатываемого ПС	61
4.4.1. Определение кода в соответствии с действующим классификатором ОКПД 2.....	61
4.4.2. Определение кода по классификатору ОКП	61
4.5. Выводы	61
ЗАКЛЮЧЕНИЕ	63
СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ	66
ПРИЛОЖЕНИЕ А	69

ВВЕДЕНИЕ

В настоящее время искусственный интеллект интенсивно развивается. Одни платформы предназначены для научных исследований в области искусственного интеллекта (например, Gym OpenAI, GVGAI), другие для проведения соревнований (например, RoboCup 2D Soccer Simulation League Champion [1], RoboCup Rescue Simulation [2]). Моделирование в виртуальном футболе используется для исследования, разработки и сравнения мультиагентных систем, для упрощения данного процесса возможно использовать Soccer Simulation [3], который предоставляет соответствующие инструменты. Среда виртуального футбола очень динамична и может хорошо симулировать условия реального мира, в частности предполагается полная автономность программ управления игроками и предоставление игрокам визуальной и аудиоинформации с предопределённой погрешностью и ограничениями на расстояние. Для управления и позиционирования игроков используются различные методы: с использованием деревьев решений, которые на основании текущего состояния (узла дерева) и входных данных формируют варианты дальнейших действий [4, 5], с использованием случайных конечных множеств [4, 6], с использованием метода Монте-Карло [4, 7], нечётких автоматов [8], вероятностных автоматов [9], свёрточных нейронных сетей [1, 4, 10].

Принятие решений может осуществляться на основе только текущего состояния мира или нескольких предыдущих состояний. При этом агент, учитывающий предыдущие состояния, будет действовать эффективнее, а поэтому выигрывать у игрока, принимающего решение только с учётом одного состояния. Планирование действий с учетом истории изменения ситуации является важным вопросом при создании интеллектуальных агентов. Планирование действий включает в себя прогнозирование изменения ситуации на поле с учетом действий, которые могут быть выполнены в сложившейся ситуации. Прогноз может осуществляться на основе как поступающих данных и меняться вместе с обновленной

информацией на каждом такте [7], так и на основе сформированной модели с постепенным её уточнением [11]. В обоих случаях в [7] и [11] учитывается информация о видимых объектах, но при этом для объектов, которые были недавно в поле зрения, а затем исчезли из него, поведение неизвестно. Учёт их поведения существенно обогатит модель и улучшит точность прогноза развития ситуации на поле.

Следует отметить, что в условиях виртуального футбола неизвестны и координаты самого игрока – их нужно вычислять на основе видимых флагов, размещённых вокруг и на игровом поле. Данная задача не всегда решается однозначно, т.к. информация программе управления предоставляется с некоторой погрешностью [12].

Цель данной работы: разработка программы, которая выполняет определение местонахождения агента и осуществляет предсказание координат для игроков недавно исчезнувших из поля зрения.

Для достижения цели был сформирован список задач:

- Выбор подходящего метода для определения местоположения игроков.
- Проектирование математической модели.
- Проектирование архитектуры.
- Реализация алгоритма определения местонахождения объектов и прогнозирования местоположения для недавно исчезнувших из поля зрения игроков.
- Проведение экспериментов для определения точности полученного решения.
- Тестирование.

Объектом исследования являются интеллектуальные агенты в среде проведения международных соревнований по виртуальному футболу.

Предметом исследования является прогнозирование перемещения невидимых интеллектуальных агентов.

Практическая ценность работы: Улучшение точности прогноза развития ситуации на поле за счёт учёта поведения недавно исчезнувших из виду игроков.

Данная работа состоит из 4 глав. В первой главе представлен сформированный список критериев к алгоритмам определения местоположения динамических объектов, прогнозирования и принятия решения игроками в среде виртуального футбола, а также выполнен обзор существующих решений и сравнение их по данному списку. На основе выполненного сравнения был выбран один из алгоритмов определения местонахождения объектов, а также сделан вывод о необходимости разработки алгоритма определения местоположения в условиях недостаточности информации и прогнозирования местонахождения объектов, которые недавно исчезли из поля зрения. Вторая глава содержит описание проектирования разрабатываемой программы, включая архитектуру и основные технические решения. В третьей главе приведено описание возможностей реализованной программы, а также описание выполненного тестирования. Последняя глава посвящена разработке и стандартизации программных средств.

1. ОБЗОР ПРЕДМЕТНОЙ ОБЛАСТИ

1.1. Введение

Рассмотрим задачу определения местоположения динамических объектов, прогнозирования и принятие решения игроками в среде виртуального футбола Soccer Simulation [3].

Представленные алгоритмы и методы можно сравнить по следующим критериям:

- объём информации, сохраняемой в памяти алгоритма;
- цель алгоритма;
- тип алгоритма;
- погрешность вычисления местоположения (в виртуальном футболе измеряется в метрах).

Объём информации, сохраняемой в памяти алгоритма, подразумевает сравнение по признаку: вычисление на основе данных одного или нескольких тактов изменения окружающего мира. Данный объём влияет на количество факторов используемых при планировании действий.

Использование только текущего состояния мира – использование информации только о текущем такте работы в вычислениях.

Учёт каждого нового состояния мира – использование каждого нового такта работы для вычислений и корректировка полученной информации за счёт информации о предыдущих тактах работы, начиная с первого.

Тип алгоритма определяет основной принцип работы, от которого зависит скорость и точность.

Цель алгоритма – различные алгоритмы используются для разных целей. Поэтому определение основной цели, а как следствие и выходных данных, важно для корректного встраивания в архитектуру решения задачи.

Прогноз действий и принятие решений – на основе выходных данных алгоритма выполняется принятие решений о дальнейших действиях игрока.

Определение местоположения – на основе выходных данных возможно определение координат текущего игрока, а также определение местоположения других объектов относительно него.

Погрешность вычисления местоположения при расчёте.

1.2. Используемые в работе термины

В работе используются следующие термины и определения:

- **Виртуальный футбол** – симулятор реального футбольного турнира, компьютерное моделирование матчей при помощи искусственного интеллекта.
- **Мультиагентные системы** – сложная система, образованная функционированием несколькими взаимодействующими интеллектуальными агентами.
- **Интеллектуальный агент** – программа, самостоятельно выполняющая задание, указанное пользователем компьютера, в течение длительных промежутков времени.
- **Местоположение** – координаты и ориентация интеллектуального агента на поле во время игры в среде виртуального футбола.
- **Недавно исчезнувший объект** – один из динамических объектов поля, который был виден в ближайшие несколько предыдущих состояний, а затем исчез из поля зрения.
- **Динамический объект** – объект на игровом поле среды виртуального футбола, которые может менять своё местоположение.
- **Прогнозирование для недавно исчезнувшего объекта** – предположение о возможном местоположении динамического объекта, который недавно исчез из поля зрения игрока, на игровом поле среды виртуального футбола.
- **Инерциальная навигация** – это метод определения координат, скорости и угловой ориентации объекта на основе измерения и интегрирования его ускорения.

1.3. Формальная запись решаемой проблемы

Симуляция в виртуальном футболе динамична. Для успешного принятия решения в такой среде информации о текущем состоянии мира недостаточно, необходимо также планирование действий. При этом эффективность планирования действий зависит от полноты информации о текущем состоянии мира. Определение своего местонахождения и местонахождения других динамических объектов на поле для интеллектуального агента важно. Существующие решения используют информацию о видимых объектах, но при этом поведение невидимых объектов для них неизвестно. Учёт их поведения дополнит модель представления о мире и улучшит точность прогноза развития ситуации на поле. Поэтому решением данной проблемы является создание программы, которая позволяет определять местоположение игроков, запоминать предыдущие состояния и осуществлять прогноз действий на основе текущего и хранящихся в памяти состояний.

1.4. Сравнение существующих подходов к решению проблемы

1.4.1. Способы решения задачи

Платформа виртуального футбола обеспечивает зашумление данных, поступающих игрокам от визуального сенсора, что накладывает ограничения на алгоритмы, применимые для решения задачи. Могут применяться следующие алгоритмы:

1. Навигация по ближайшему флагу и дальней линии.
2. Навигация по двум ближайшим флагам и дальней линии.
3. Навигация с использованием фильтра Калмана.
4. Навигация с использованием фильтра частиц.
5. Метод Монте-Карло и агрегации данных (MCSDA).

6. Метод RFS (случайных конечных множеств).

7. Концепция построения интеллектуальных агентов реального времени на основе модели опережающего итеративного планирования.

Рассмотрим алгоритмы подробнее.

- Навигация по ближайшему флагу и дальней линии.

Вычисления осуществляются на основе тригонометрических формул – выбирается ближайшая к агенту статическая точка на поле (центр поля, ворота команд, 4 области перед воротами, справа и слева от них, область непосредственно перед воротами) и дальняя видимая линия границы поля (см. рис. 1). С помощью вычисленного угла по формуле (1):

$$\beta = -\sin(\alpha)(90 - |\alpha|), \quad (1)$$

где α – угол под которым видна линия и доступных абсолютных координат флага агент рассчитывает свои реальные координаты [13] с использованием (2):

$$(p_x, p_y) = (f_x, f_y) - \pi(fr, f\phi + (90 + \beta)), \quad (2)$$

где $f_x, f_y, fr, f\phi$ – координаты x, y флага, а также расстояние до него и угол, под которым флаг виден соответственно, π – функция преобразования полярных координат в декартовы. Преобразование осуществляется в соответствии с (3):

$$\pi(r, \phi) = (r\cos(\phi), r\sin(\phi)) \quad (3)$$

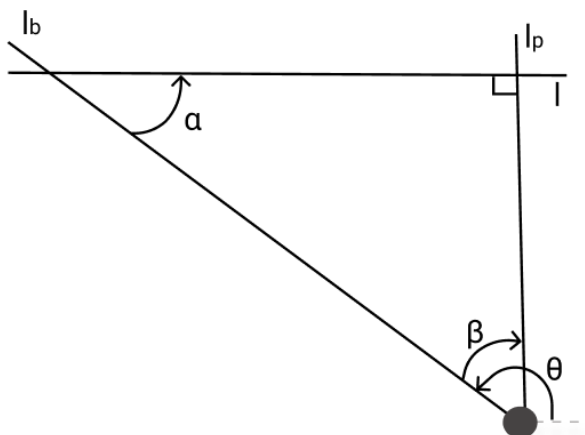


Рисунок 1. Иллюстрация к «Навигация по ближайшему флагу и дальней линии»

- Навигация по двум ближайшим флагам и дальней линии.

Метод проводит вычисления на основе тригонометрических формул с использованием двух ближайших к агенту статических точек на поле (центр поля, ворота команд, 4 области перед воротами, справа и слева от них, область непосредственно перед воротами) и дальней линии. Расстояние до флага и его координаты задают круг возможных позиций, а пересечение двух окружностей определяет положение игрока (см. рис. 2). Расстояние между флагами (f, g) определяется по формуле (4):

$$d = \sqrt{(gx - fx)^2 + (gy - fy)^2} \quad (4)$$

Вычисление абсолютных координат игрока [13] производится в соответствии с формулами (5), (6), (7):

$$(px, py) = (px' - h \sin(\alpha), py' + h \cos(\alpha)) \quad (5)$$

$$(px', py') = (fx + a \cos(\alpha), fy + a \sin(\alpha)) \quad (6)$$

$$\sin(\alpha) = \frac{\Delta y}{d}, \cos(\alpha) = \frac{\Delta x}{d} \quad (7)$$

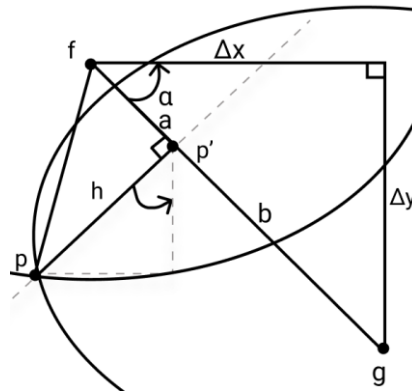


Рисунок 2 Иллюстрация к «Навигация по двум ближайшим флагам и дальней линии»

- Навигация с использованием фильтра частиц.

Фильтр частиц является методом определения абсолютных координат игрока, в соответствии с которым для оценки координат создается множество гипотез об их текущих значениях. Алгоритм определения абсолютных координат на основе фильтра частиц включает следующие шаги:

Шаг 1. Инициализация – получение информации о первом шаге работы, при этом генерация гипотез происходит случайным образом.

Шаг 2. Прогноз – предположение о местонахождении игрока на основе информации полученной от сервера.

Шаг 3. Коррекция – вычисление весовых коэффициентов, при этом перед данным вычислением выполняется отсеивание частиц за счёт вычисления верхней и нижней границей гипотез (частицы, которые не вошли в диапазон, удаляются) и ресэмплинг – удаление гипотез с малым весом и дублирование гипотез с большим весом.

Шаг 4. Оценка состояния – вычисление абсолютных координат, как взвешенной суммы состояний всех частиц [13, 14].

- Навигация с использованием фильтра Калмана.

Метод позволяет получать оценку вектора состояния объекта (в данном случае – координат игрока) на основе серии зашумленных измерений. Решение осуществляется в несколько шагов:

Шаг 1. Разбор и анализ информации, получаемой от визуального сенсора (информации от сервера). При наличии сохранённых данных этап разбора возможно пропустить и сосредоточится на анализе.

Шаг 2. Циклическая обработка всевозможных пар – перебор всех пар видимых флагов и вычисления абсолютных координат по этим флагам.

Шаг 3. Вычисление дисперсии ошибки сенсора – основной этап, на котором определяется качество работы данного метода. Вычисление производится в соответствии с формулой (8):

$$\sigma_{i+1}^2 = \frac{(r_{\max} - r_{\min})^2}{12} \quad (8)$$

где r_{\max} и r_{\min} – максимально и минимально возможные значения расстояния; σ^2 – дисперсия ошибки.

Шаг 4. Обновление значения коэффициента усиления Калмана (K) с учетом полученной дисперсии. Значение коэффициента должно обеспечивать максимальную близость вычисляемых оптимальных значений

абсолютных координат к их истинным значениям, вычисления проводятся с использованием формулы (9):

$$K = \frac{\sigma_i^2}{\sigma_i^2 + \sigma_{i+1}^2} \quad (9)$$

Шаг 5. Коррекция с помощью коэффициента Калмана оценочного значения абсолютных координат агента в данной итерации. Коррекция осуществляется на основе формул (10) и (11):

$$x_{i+1} = x_i * K + (1 - K) * (x_i + \Delta x) \quad (10)$$

$$y_{i+1} = y_i * K + (1 - K) * (y_i + \Delta y) \quad (11)$$

$\Delta x, \Delta y$ – предполагаемое изменение координат.

Таким образом вычисляются реальные координаты игрока [13-15].

- Метод Монте-Карло и агрегации данных (MCSDA).

Метод основывается на комбинации поиска Монте-Карло и агрегации данных (MCSDA) [7], чтобы адаптировать действия агента к стратегиям игры команды противника. Используя простое представление предметной области, алгоритм контролируемо обучается на начальном наборе данных, состоящем из нескольких имитаций реальных игр по аналогии с [16]. Метод позволяет управлять навигацией и помогает в принятии решений на поле. Алгоритм принимает в качестве входных данных набор пар состояние-действие. Затем происходит обучение классификатора $\hat{\pi}$ и на каждой итерации алгоритм расширяет свой набор данных $\sim \pi$ путем генерации состояния s_t на каждом временном шаге, при этом ожидаемое значение максимизируется функцией $V_p(s_t, a)$. После всего цикла агрегированный набор данных используется для обучения нового классификатора $\sim \pi$. Псевдокод алгоритма представлен на рис. 3.

begin

Тренировки классификатора $P(\hat{\pi})$ на экспертном наборе данных De .

$P1 \leftarrow \text{тренировкаКлассификатора}()$.

Инициализация $D \leftarrow De$.

for $i = 1$ **to** N **do**

инициализация $s_0 \leftarrow \text{init}(D$.

```

for  $t = 1$  to  $T$  do
    Получение состояний  $st$  от предыдущего состояния классификатора  $P_{i-1}(st-1)$ .
     $A \leftarrow$  выбор возможных действий из  $st$  (Если необходимо).
    foreach  $a \in A$  do
        Выполнение  $K$  симуляций методом Monte Carlo длины  $K$  для оценки состояний
         $Vp(st, a)$ .
    end
     $at \leftarrow \arg \max_a Vp(st, a)$ .
     $D \leftarrow D \cup \{st, at\}$ .
End
    Тренировки классификатора  $P_t(\tilde{\pi}_i)$  на наборе данных  $D$ .
end
return массив  $P_t$ 
end

```

Рисунок 3 Псевдокод алгоритма «MCSDA»

- Метод RFS (случайных конечных множеств)

Метод обеспечивает построение карт перемещений роботов с использованием случайных конечных множеств [6]. Он применяется к задаче оценки положения товарищей по команде и соперников в Лиге SPL и состоит из двух шагов:

Шаг 1. Предсказание – создание гипотез о появлении новых препятствий для следующего такта работы.

Шаг 2. Обновление – вычисление вероятности обнаружения новых препятствий в заданной точке, после чего выполняются операции обрезки и слияния элементов (Гауссианы, которые определяются достаточно близко, через порог расстояния Махаланобиса, объединяются в один).

Псевдокод алгоритма представлен на рис. 4.

```

// Этап прогнозирования
for  $i = 1$  to  $J_k-1$  do
    Вычисление новых ковариаций  $Ko_i \leftarrow$  Ковариация( $Ko_{Pi-1}, Pi-1 + Q, w_i$ )
end for
    Прогнозирование новой карты препятствий  $M_k \leftarrow \text{generateNewGaussian}(Z_{k-1}, X_{k-1})$ 

```

```

//Этап обновления(уточнения)
for  $i = 1$  to  $J_k|k-1$  do
    Вычисление вероятности препятствия  $P_D^{(i)}$ 
    обновление  $w_i$  с учётом новой вероятности  $P_i$  и предыдущих состояний
end for
 $N = 1$ 
for each  $z$  in  $Z_k$ 
    for  $i = 1$  to  $J_k|k-1$  do
        Вычисление  $H, S_i$  and  $K_i$ 
        Вычисление  $KoN+i$  используя предыдущее состояние Байесовского фильтра
        Получение вероятности  $P_{N+I}$  из предыдущего состояния
        Вычисление коэффициента коррекции  $t_i$ 
    end for
    for  $i = 1$  to  $J_k|k-1$  do
        Вычисление новых состояний для карты препятствий  $w_{N+i}$ , используя  $t_i$ 
    end for
    Обновление  $N$ 
end for
 $J_k = N$ 
    Уточнение спрогнозированной карты препятствий  $v_k$ , используя  $Ko_i, P_i, w_i$ 
    обрезка( $M$ )

```

Рисунок 4 Псевдокод алгоритма RFS

С помощью данного алгоритма вычисляются карты препятствий в видимом пространстве, чтобы получить положение объекта на поле необходимо оценить вес каждого элемента. Если эти значения превышают заданный порог, то положение препятствия задаётся средним вектором и добавляется к вектору, представляющему текущую карту (см. рис. 5).

```

 $M_k = []$ 
for  $i = 1$  to  $J_k$  do
    if  $w_k > thrld$  then
         $M_k = [M_k K_{ok}]$ 
    end if
end for

```

Рисунок 5 Получение местоположения после использования алгоритма
«RFS»

- Концепция построения интеллектуальных агентов реального времени.

Для принятия решений игроками используется концепция построения модели опережающего итеративного планирования [10]. Шаги являются обобщёнными, так как из-за экспоненциального роста деревьев возможных событий, точное прогнозирование невозможно. Цикл вычислений включает следующие шаги:

Шаг 1. Получение сенсорной информации от подсистемы восприятия.

Шаг 2. Оценка ситуации.

Шаг 3. Прогнозирование ситуации.

Шаг 4. Планирование действий.

Шаг 5. Выдача исполнительной подсистеме команд на выполнение действий. Псевдокод алгоритма представлен на рис. 6.

```
AFR = AFR0 ; PH = PH0 ; Actcur = Act0 ; {Actnext} = {Actnext}0 ;
```

```
While true do
```

```
    PCP ← выполнитьВосприятие(AFR);
```

```
    St ← оценитьСитуацию(PH, PCP, AFR);
```

```
    if ({CEP} = ∅)
```

```
        if (Actcur не завершено))
```

```
            продолжитьВыполнение(Actcur);
```

```
            AFR ← планироватьМентальнуюДеятельность(St, Actnext);
```

```
            St+n ← прогнозироватьСитуацию(St, n, AFR);
```

```
            {Actnext} ← планироватьВнешниеДействия(St+n, Actnext, AFR);
```

```
            AFR ← планироватьМентальнуюДеятельность(St, {Actnext});
```

```
        else
```

```
            AFR ← планироватьМентальнуюДеятельность(St);
```

```
            Actcur ← выбратьВнешнееДействие(Actnext, AFR);
```

```
            Actnext ← формироватьМножествоОбобщенныхВыходныхДанных(AFR);
```

```
    else
```

```
        Td ← определитьЗапасВремени({CEP}, St);
```

обдумать Реакцию На ПКС; end while
--

Рис.6 Псевдокод метода «Концепция построения интеллектуальных агентов реального времени на основе модели опережающего итеративного планирования»

1.4.2. Сравнение аналогов по заданным критериям

Сравнение аналогов по заданным критериям представлено в табл. 1.

Таблица 1. Сравнения аналогов

Критерии	Объём информации, сохраняемой в памяти	Цель алгоритма	Тип алгоритма	Погрешность вычисления местоположения (метры)
Концепция построения интеллектуальных агентов реального времени на основе модели опережающего итеративного планирования	Учёт каждого нового состояния мира	Прогноз действий и принятие решений	Не вероятностный	Не вычисляет местоположение
Навигация по ближайшему флагу и дальней линии	Использование только текущего состояния мира	Определение местоположения	Не вероятностный	0.25
Навигация по двум	Использование только	Определение	Не вероятностный	1.02

ближайшим флагам и дальней линии	текущего состояния мира	местополож ения	ый	
Навигация с использование м фильтра Калмана	Учёт каждого нового состояния мира	Определени е местополож ения	Вероятностн ый	0.29
Навигация с использование м фильтра частиц	Учёт каждого нового состояния мира	Определени е местополож ения	Вероятностн ый	0.10
Метод MCSDA	Учёт каждого нового состояния мира	Прогноз действий и принятие решений	Не вероятностн ый	Не вычисляет местоположен ие
Метод RFS	Учёт каждого нового состояния мира	Определени е местополож ения	Вероятностн ый	0.20

1.5. Выводы

Методы, использующие только текущий такт, работы работают быстрее, чем методы, использующие предыдущие состояния мира, но при этом имеют меньшие возможности по предсказанию будущих состояний. Концепция построения интеллектуальных агентов реального времени на основе модели опережающего итеративного планирования учитывает информацию, начиная с первого шага работы, при этом возможно планирование действий, а на основе этого и предсказание действий агентов, но алгоритм не осуществляет вычисление координат. Навигация с использованием фильтра частиц [13], показывает самую высокую точность

вычислений среди всех представленных алгоритмов, но при этом на вычисление тратится наибольшее время. Методы навигации с использованием фильтра Калмана, MCSDA, RFS соответствуют всем заданным параметрам, но их применение зависит от задачи и доступных данных.

Целью данной работы является вычисление местоположений объектов, поэтому алгоритм MCSDA будет исключен из рассмотрения, т.к. в нём принятие решений и навигация осуществляются на основе уже вычисленных местоположений объектов. Метод RFS будет исключен из дальнейшего рассмотрения, так как в нём получение местоположения осуществляется на основе карты препятствий, а построение данной карты для каждого исчезнувшего из рассмотрения игрока – крайне ресурсоёмкая задача.

Построим новый алгоритм определения координат игрока в том числе с учётом прогнозирования местоположения игроков, которые были недавно в поле зрения, а затем исчезли из него, на основе фильтра Калмана.

2. ОПИСАНИЕ ЗАДАЧИ И ПОДХОДОВ К РЕШЕНИЮ

2.1. Введение

В ходе обзора и сравнения аналогов был выявлен подходящий метод определения местоположения объекта в условиях, когда флагов недостаточно флагов для работы алгоритма. Так как определение местоположения и прогнозирование координат игроков должно обогатить общую картину мира, то решение должно быть модульным для возможности взаимодействия с другими решениями. Для выполнения поставленных целей необходимо решить следующие задачи:

- Определение местонахождения игрока на поле
- Определение местоположения в условиях недостаточности информации, полученной из текущего состояния (Видимых флагов меньше двух)
- Сохранение информации о предыдущих состояниях
- Прогнозирование местоположения для недавно исчезнувших из виду объектов.

Проектирование выполнялось на основе поставленных задач, для этого была разработана математическая модель будущего решения, построена архитектура будущего решения. Алгоритмы и устройство программы описано в диаграммах деятельности и классов.

2.2. Математическая модель решения

Для формализации сформулируем задачу в виде математической модели, представленной формулой (12):

$$M = (I, P, C, F, O), \quad (12)$$

где I – входные данные, поступающие от сервера. Входные данные приложены в виде формулы (13). В этих данных содержится информация о видимых статических (флаги, линии, ворота) и динамических объектах (мяч, другие игроки) на поле.

$$I = \{<f, l, g, b, a>\}, \quad (13)$$

где f – множество полученных видимых флагов, l – множество полученных видимых линий, g – множество видимых флагов ворот, b – множество положений мяча, состоящее из одного элемента, a – множество видимых агентов.

Функция обработки входных данных и вычисления координат агента, координат видимых объектов для агента представлена формулой (14).

$$P:I \rightarrow C \quad (14)$$

Вычисленные координаты для динамических объектов данного такта игры представлены формулой (15), где k – множество координат (x, y) , для каждого динамического объекта.

$$C = P(I) = \{ \langle k \rangle \} \quad (15)$$

Функция анализа полученных данных (см. формулу 16), которая выполняет прогнозирование, используя данные о текущем состоянии мира и предыдущих вычисленных состояний.

$$F:C \rightarrow O \quad (16)$$

Выходные данные содержат информация о местоположении видимых объектов для текущего такта и прогноз действий для объектов, недавно исчезнувших из поля зрения программы (см. формулу 17).

$$O = F(C) = \{ \langle k, p \rangle \} \quad (17)$$

k – вычисленные координаты для динамических объектов текущего такта игры, p – прогноз действий для исчезнувших из виду объектов, т.е. предположение о дальнейшем перемещении данных объектов.

Пример входных данных для платформы виртуального футбола Soccer Simulation [3, 4]:

Флаги(f) – опорные объекты, размещаемые на поле для вычисления координат:

```
{
'f b 1 20 dist': 43.8,
'f b 1 20 angle': 10,
...}
```

Линии (l) – опорные объекты, размещаемые вокруг поля для вычисления координат:

```
{  
'l b dist': 43.8,  
'f b l 20 angle': -41,  
...}
```

Ворота (g) – опорные объекты, размещаемые на поле для идентификации различных частей ворот и вычисления координат:

```
{  
'g r dist': 100,  
'g r angle': -44,  
...}
```

Мяч (b) – динамический объект, перемещающийся по полю. На основе получаемой информации возможно вычисление местоположения мяча:

```
{  
'b dist': 33.1,  
'b angle': 2,  
...}
```

Видимые игроки (a) – динамические объекты, перемещающиеся по полю. На основе получаемой информации возможно вычисление местоположения других игроков:

```
{  
'p HELIOS2017 2 dist': 33.1,  
'p HELIOS2017 2 angle': -7,  
...}
```

Пример внутреннего представления генерируемых данных на основе полученной от сервера информации приведён на рис. 7.

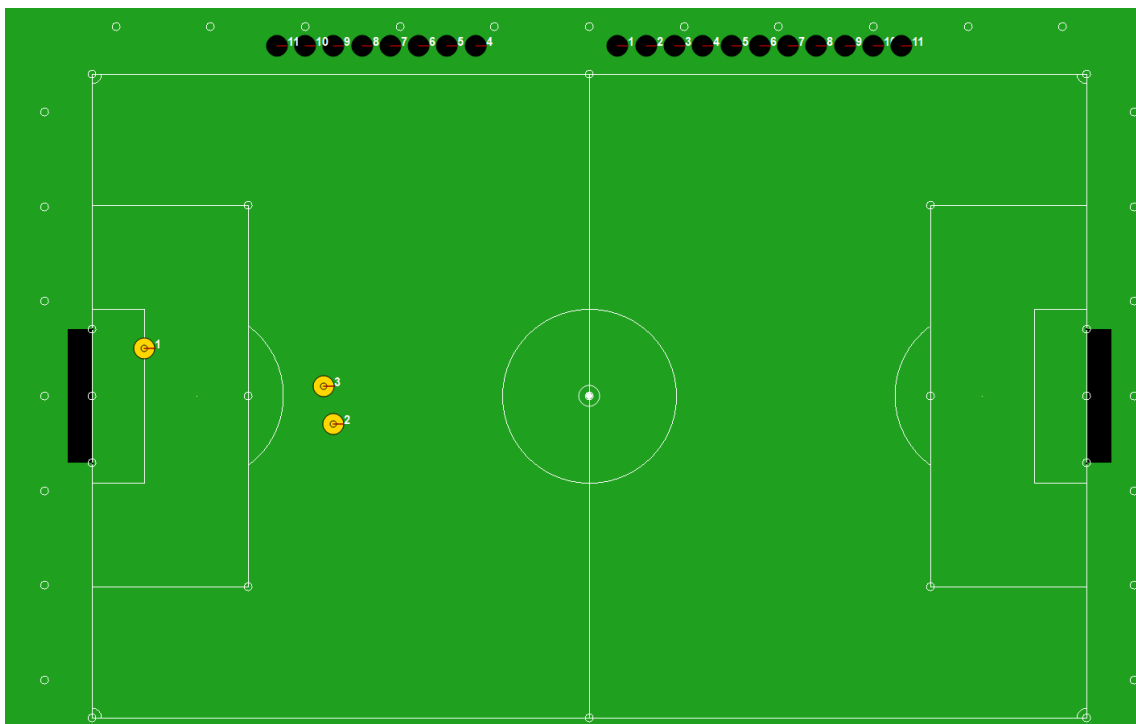


Рисунок 7. Иллюстрация к примеру генерируемых данных на основе полученной от сервера информации

Информация о игроке под номером 1, где x и y – вычисленные координаты, $absX$ и $absY$ – реальные координаты игрока, $angle$ – ориентация игрока на поле, $speedX$ и $speedY$ – скорость игрока по соответствующим координатам (k – вычисленные координат для динамических объектов текущего такта игры):

```
x = -46.77
y = -5.88
absX = -47.49
absY = -5.12
angle = 2.28(radian)
speedX = 0.72
speedY = 0.44
```

Информация о видимых игроках 2 и 3, где `viewPlayer` – массив имён видимых игроков, `mapPlayer` – содержит поимённый массив видимых игроков с вычисленным местоположением, при этом x , y – вычисленные координаты игрока, $angle$ – ориентация игрока на поле (k – вычисленные координат для динамических объектов текущего такта игры):

```
otherPlayers.viewPlayer = ['b dist', 'p HELIOS2017 2', 'p
HELIOS2017 3']
otherPlayers.mapPlayer = {
  'p HELIOS2017 2': {
```

```

        x: -27.53,
        y: 3.18,
        angle: -48
    },
    'p HELIOS2017 3': {
        x: -28.51,
        y: -1.77,
        angle: -60
    }
}

```

Прогноз действий для пропавших из виду объектов (в данном случае игрок 4), где x и y – спрогнозированные координаты, $beforeX$ и $beforeY$ – вычисленные координаты игрока на предыдущем шаге, $angle$ – ориентация игрока на поле, $predictTick$ – номер предсказанного такта (p – прогноз действий для исчезнувших из виду объектов):

```

[ {
    'p HELIOS2017 4': {
        x: -28.03,
        y: 8.02,
        beforeX: -27.53,
        beforeY: 7.18,
        angle: -51,
        predictTick: 1
    }
} ]

```

2.3. Архитектура программы

Структура вычисления местоположения видимых объектов и построения прогноза для исчезнувших из вида объектов включает в себя следующую последовательность действий:

- обработка информации о текущем такте работы для получения координат видимых динамических объектов;
- взаимодействие с хранилищем состояний для использования информации о предыдущих тактах работы;
- прогнозирование перемещения для недавно исчезнувших из поля зрения динамических объектов;
- сохранение информации о текущем состоянии.

Данная последовательность действий может рассматриваться как базовый набор компонентов архитектуры (см. рис. 8). При этом компоненты включают в себя специфические модули:

– Компонент обработки информации, поступившей от сенсора, выполняет получение и обработку информации, вычисление координат для всех видимых объектов с использованием фильтра Калмана, а также уточнение местоположения агента, при недостаточности информации от сервера.

– Компонент взаимодействия с хранилищем информации о предыдущих состояниях выполняет определение недавно скрывшихся из виду объектов и уточнение информации о действиях агентов, которые исчезли из виду.

– Компонент прогнозирования перемещения осуществляет анализ скрывшихся из вида объектов, длительности их пребывания в поле зрения текущего объекта, выполнение прогнозирования для исчезнувших из виду объектов.

– Компонент сохранения информации о текущем такте работы выполняет сохранение данных о текущем такте игры, контроль и удаление информации о тактах работы, которые более не входят в фиксированный промежуток времени.

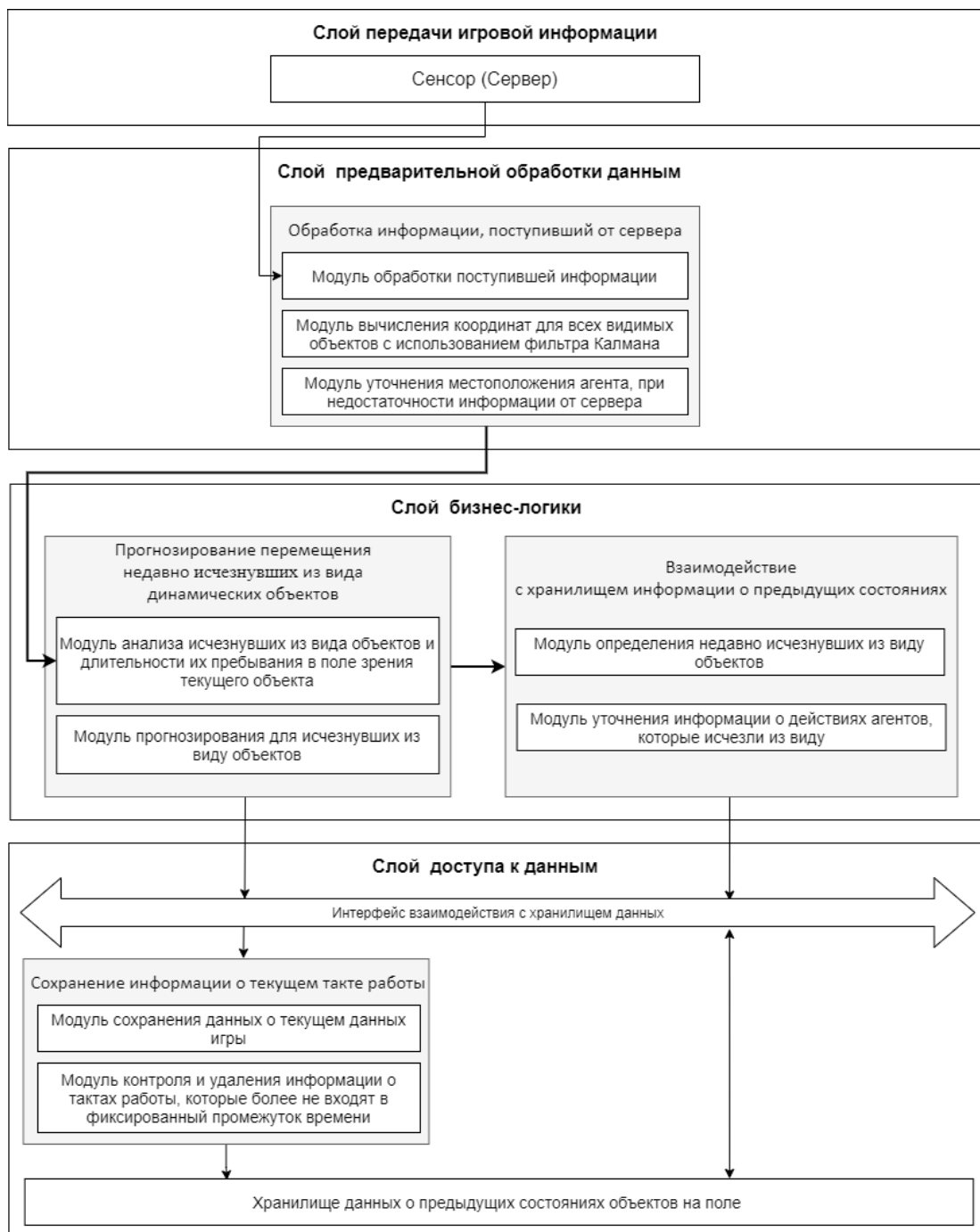


Рисунок 8. Архитектура приложения

2.4. Предполагаемое решение

На основе модели программы управления реализован алгоритм определения координат объектов на поле с использованием фильтра Калмана и построения прогноза для недавно скрывшихся из вида объектов.

Шаг 1. Разбор и анализ информации, получаемой от сервера.

Шаг 2. Если видимых флагов больше двух, то переходим к шагу 3, иначе у модуля взаимодействия с хранилищем информации выполняется

запрос двух последних состояний (тактов работы) и на основе координат, направления движения и скорости выполняется вычисление нового местоположения. Если шаг повторяется в течение нескольких тактов игры, то необходимо выполнять развороты головы или всего тела для попадания в поле зрения не менее двух флагов, чтобы уточнить своё местоположение.

Шаг 3. Вычисление координат осуществляется с использованием алгоритма фильтра Калмана [13]. Коррекция осуществляется по формулам (18) и (19):

$$x_{i+1} = x_i * K + (1 - K) * (x_i + \cos(\alpha) * \text{speedX}) \quad (18)$$

$$y_{i+1} = y_i * K + (1 - K) * (y_i + \sin(\alpha) * \text{speedY}), \quad (19)$$

где α – направление тела игрока (в радианах),

speedX, speedY – скорости перемещения по координате x и y соответственно.

Шаг 4. После определения координат текущего игрока выполняется вычисление координат видимых игроков алгоритмом определения координат по трём флагам.

Шаг 5. Затем выполняется анализ исчезнувших из вида игроков и длительности их пребывания в поле зрения текущего игрока. Если игрок был в поле зрения два и более такта, то включается в множество, для которого выполняется прогнозирование

Шаг 6. После получения массива данных анализа выполняется прогнозирование новых координат на основе последних двух состояний, из которых определяется последнее известное местоположение, направление движения и скорости объекта.

Примеры расчётов:

1. После преобразования входных данных информация о видимых флагов представляет собой массив объектов вида:

```
[{  
  name: 'f b l 20 dist'  
  dist: 43.8,  
  angle: 10,  
},
```

...]

2. Если видимых флагов меньше, чем два, то на основании двух последних состояния выполняется вычисление координат:

```
firstCoordVal.x = -9.01, firstCoordVal.y = 28.27
secondCoordVal.x = -8.42, secondCoordVal.y = 28.61
radian = 3.80
speedX = 0.59
speedY = -0.34
averageX = firstCoordVal.x + speedX * cos(radian) = -9.47
averageY = firstCoordVal.y + speedY * sin(radian) = 28.48
```

3. Если видимых флагов больше двух. На основе полученных данных о флагах вычисляются координаты по двум флагам.

```
[{
name: 'f b l 20 dist'
dist: 43.8,
angle: 10,
}, ...]
```

Циклически перебираются всевозможные пары, затем считается среднее для соответствующих координат:

```
averageX: -46.71, averageY: -6.64
```

4. Вычисление дисперсии ошибки

```
distanceMax = 111.0, distanceMix = 12.6
variance = ((distanceMax - distanceMix) ** 2) / 12 = ((111.0 -
12.6)^2)/12 = 806.88
```

5. Обновляется коэффициент Калмана:

```
varianceLast = 805.24
kalman = (varianceLast) / (varianceLast + variance) =
805.24/(805.24+806.88) = 0.4994
```

6. Выполняется коррекция с помощью коэффициента Калмана:

```
speedX = 0.72, speedY = 0.44, radian = 2.29, coordLast.x = -
46.36, coordLast.y = -5.45
x = averageX * kalman + (1 - kalman) * (coordLast.x + speedX *
cos(radian)) = -46.70 * 0.4994 + (1 - 0.4994) * (-46.35 + 0.72 * 2.29)
= -46.77
y = averageY * kalman + (1 - kalman) * (coordLast.y + speedY
*sin(radian)) =
-6.64 * 0.4994 + (1 - 0.4994) * (-5.45 + 0.44 * 2.29) = -5.87
```

7. Далее выполняется определение координат видимых игроков по трём флагам, один флаг всегда заменяется местоположением текущего игрока. Пример данных:

```
'p HELIOS2017 2 dist': {
x: -27.53,
```



```

        y: 3.18,
        angle: -48
    }

```

8. Выполняется анализ исчезнувших из вида игроков. После выполнения получается список исчезнувших из виду объектов:

```
['p HELIOS2017 7 dist', 'p HELIOS2017 4 dist', 'p Oxsy 1 dist']

```

9. Прогнозирование осуществляется после получения двух последних состояния для исчезнувших объектов:

```

radian = 53.32
metPos[length-2].x = -25.63, metPos[length-2].y = -21.49
metPos[length-1].x = -27.34, metPos[length-1].y = -21.78

```

На основе этих данных получаем скорость объекта:

```
speedX: 1.71, speedY: 0.29

```

Затем выполняется прогнозирование:

```

predictX = metPos[length-1].x + speedX * cos(radian) = -27.34 +
1.71 * cos(53.32) = -29.05
predictY = metPos[length-1].y + speedY * sin(radian) = -21.78
+0.29 * sin(53.32) = -21.75

```

2.5. Алгоритм инерциальной навигации в условиях недостаточности флагов

На основе математической модели, архитектуры и предполагаемого решения была построена диаграмма деятельности для алгоритма определения местоположения в условиях недостаточности флагов, которая представлена на рисунке 9.

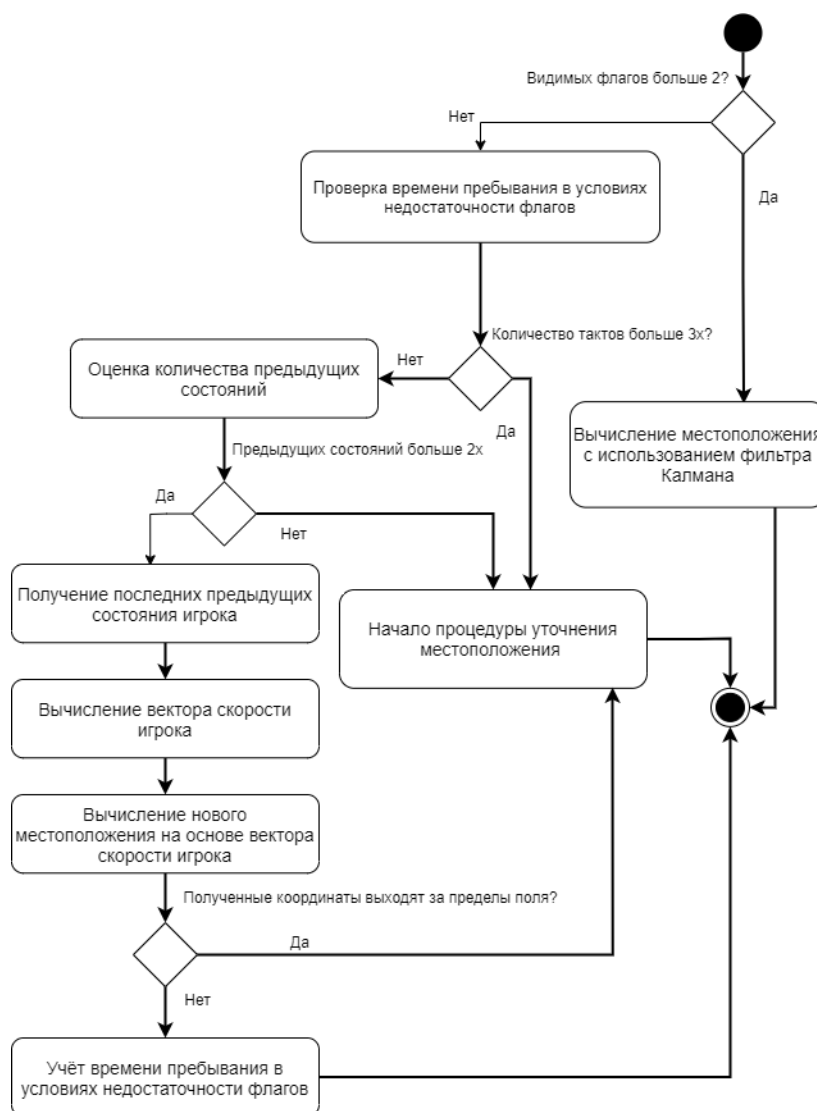


Рисунок 9. Диаграмма деятельности для алгоритма инерциальной навигации

2.6. Алгоритм прогнозирования местоположения исчезнувших из поля зрения объектов

На основе математической модели, архитектуры и предполагаемого решения была построена диаграмма деятельности для алгоритма прогнозирования местоположения исчезнувших из поля зрения объектов, которая представлена на рисунке 10.

Прогнозирование состоит из двух этапов:

- Определение списка исчезнувших объектов.
- Прогнозирование местоположения исчезнувших объектов.

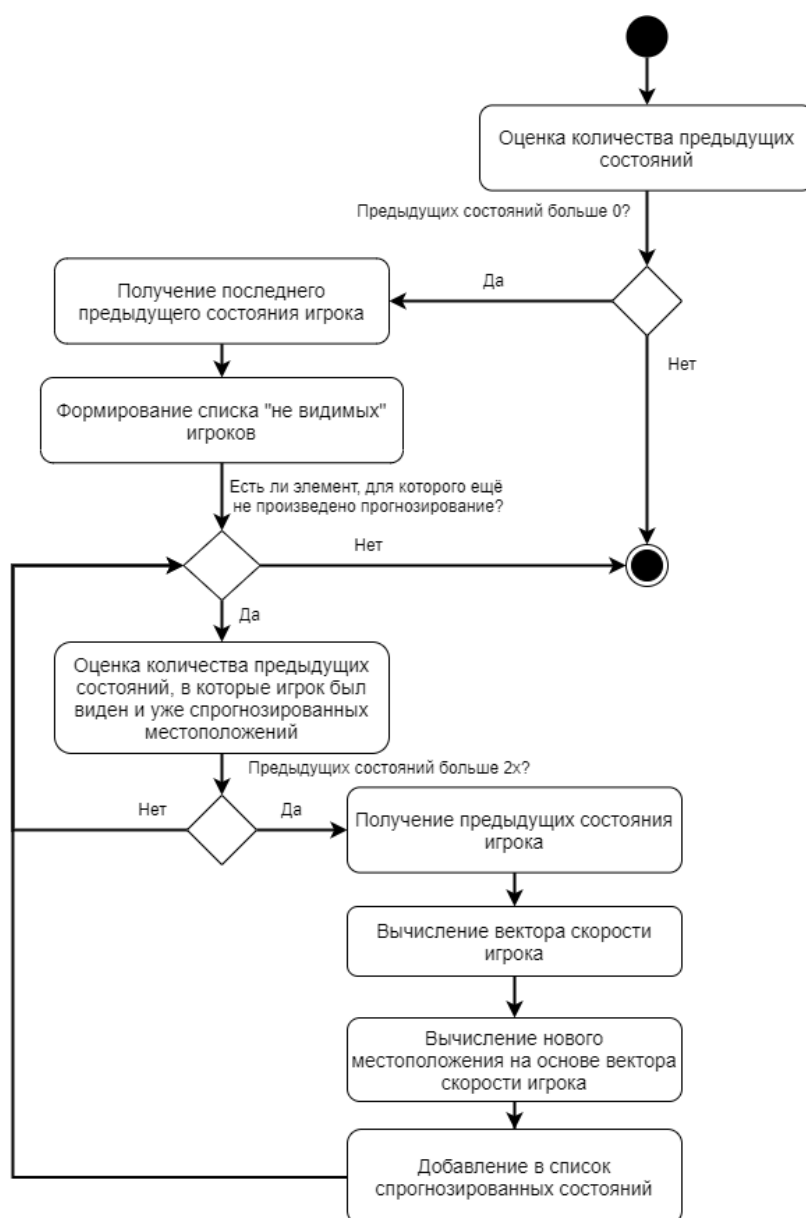


Рисунок 10. Диаграмма деятельности для алгоритма прогнозирования местоположения исчезнувших из поля зрения объектов

2.7. Общая структура

2.7.1. Диаграмма классов

В результате анализа предметной области, построения математической модели, архитектуры, предполагаемого решения и построенных диаграмм деятельности была построена диаграмма классов программы, которые представляют элементы программы. Диаграмма представлена на рисунке 11. Подробное описание методов и свойств каждого класса приведено в приложении А.

Построение диаграммы осуществлялось с использованием шаблонов проектирования:

- Шаблон проектирования Одиночка(Singleton) используется в создании хранилища информации о предыдущем состоянии мира для каждого отдельного игрока.
- Шаблон проектирования Строитель(Builder) используется для класса вычисления координат и статистики.
- Шаблон проектирования Адаптер(Adapter) используется для преобразования информации, поступающей от сервера(сенсора).

Диаграмма классов представлена на рисунке 10 и условно разделена на 4 части:

- Хранение предыдущих состояний и прогнозирование местоположения.
- Обработка входных и вычисленных данных, определение местоположения видимых объектов.
- Вспомогательные функции.
- Построение статистики.

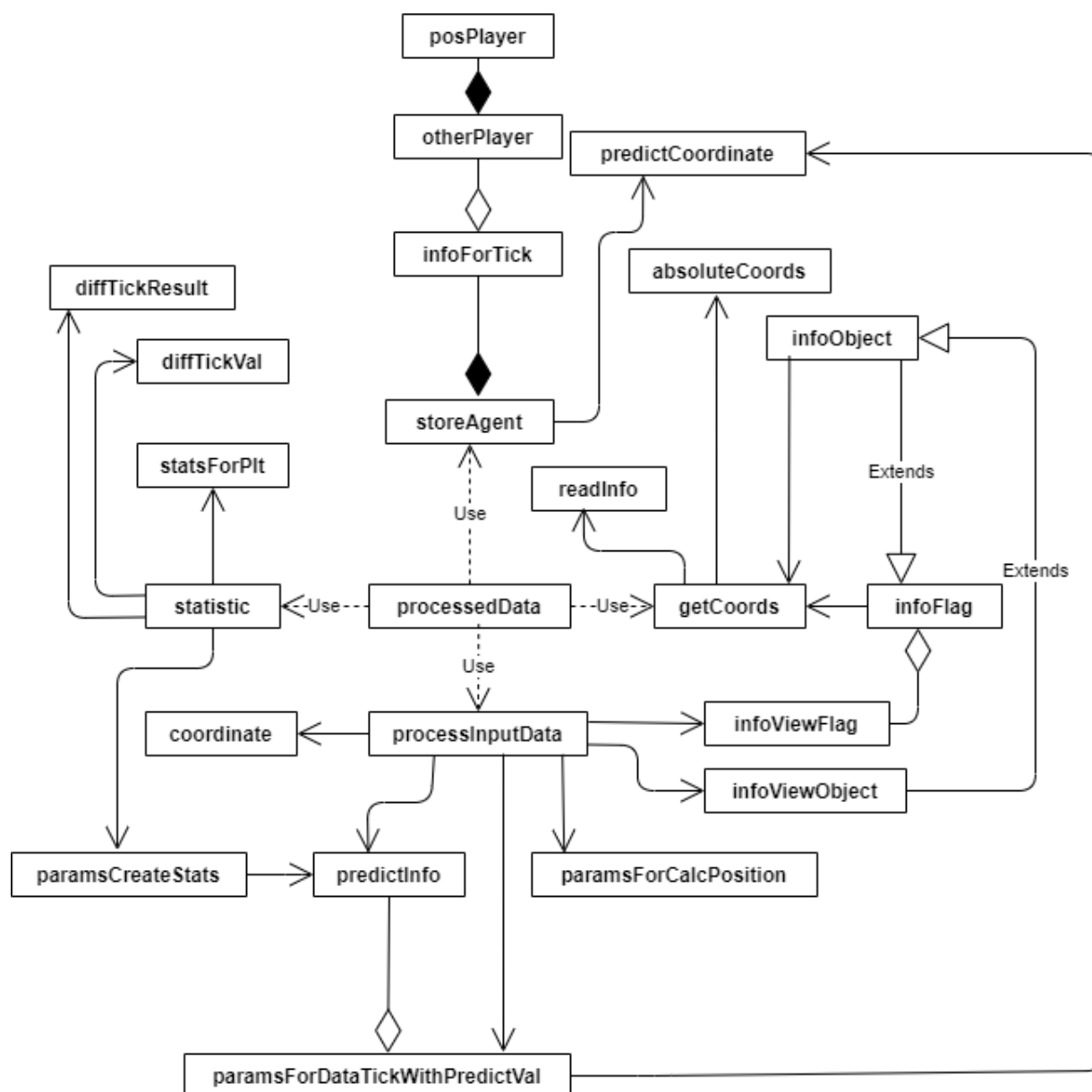


Рисунок 11. Диаграмма классов

2.7.2. Хранение предыдущих состояний и прогнозирование местоположения

Данная часть содержит в себе классы, предназначенные для хранения предыдущих вычисленных состояний и на их основе выполняющие прогнозирование.

Полученные классы:

- storeAgent – основной класс хранения информации о предыдущих состояниях игроков. Содержит методы уточнения пропавших из виду

объектов, выполнения прогнозирования для скрывшихся из виду объектов, сохранением вычисленной информации о текущем состоянии.

- `infoForTick` – класс хранящий информацию о вычисленном местоположении для текущего игрока и видимых объектов.
- `otherPlayer` – вспомогательный класс, хранящий массив имён видимых игроков, а также их вычисленные координаты
- `posPlayer` – вспомогательный класс, хранящий информацию о местоположении объекта, включающий координаты и ориентацию.
- `predictCoordinate` – вспомогательный класс, хранящий информацию о предсказанном местоположении для объекта.

2.7.3. Обработка входных и вычисленных данных, определение местоположения видимых объектов

Данная часть содержит в себе классы, предназначенные для обработки полученных и обработанных данных от сервера, определения местоположения текущего игрока, местоположения видимых для него объектов в условиях, когда информации достаточно, и в условиях недостаточности флагов.

Полученные классы:

- `processInputData` – основной класс для обработки входных данных и определения местоположения игрока и видимых объектов. Вычисление и подготовка данных всех вычисленных данных для текущего состояния мира для использования в статистических расчётах.
- `predictInfo` – класс хранящий информации о предсказанном объекте, средней ошибкой вычисления, среднеквадратичной ошибкой вычисления, номере предсказываемого состояния.
- `paramsForDataTickWithPredictVal` – вспомогательный класс, содержит параметры для функции преобразовывающие данные для статистических вычислений.

- `paramsForCalcPosition` – вспомогательный класс, содержащий параметры для вычислений местоположения текущего состояния.
- `infoViewObject` – вспомогательный класс, наследующий поля от `infoObject`, используемый для хранения информации о видимых объектах для игрока в текущем состоянии.
- `infoObject` – вспомогательный класс, наследующий поля от `infoFlag`, используемый для хранения информации о видимых объектах игрока после обработки входной информации.
- `infoViewFlag` – вспомогательный класс, наследующий поля от `infoFlag`, используемый для хранения информации о видимых флагах для игрока в текущем состоянии.
- `infoFlag` – основной класс, хранящий информацию о видимом флаге, дистанции до него и угле, под которым флаг виден.
- `Coordinate` – вспомогательный класс, хранящий координаты объекта.

2.7.4. Вспомогательные функции

Данная часть содержит в себе вспомогательные классы, в которые вынесены не основные функции по получению или вычислению координат, преобразованию входящей информации с сервера.

Полученные классы:

- `getCoords` – основной класс, который содержит набор методов для вычисления координат с помощью различного количества исходных флагов, получению абсолютных координат игрока, преобразованию для вычисления координат объекта и обработки входных данных.
- `absoluteCoords` – вспомогательный класс хранящий информации о полученных абсолютных координатах объекта.
- `readInfo` – вспомогательный класс, считывающий информацию, поступающей от сервера.

2.7.5. Построение статистики

Данная часть содержит в себе классы, предназначенные для преобразования вычисленных данных в статистические для построение графиков, подсчёта средней и среднеквадратичной ошибки вычислений

Полученные классы:

- `statistic` – основной класс для преобразования вычисленной информации для статистического отображения.
- `diffTickVal` – вспомогательный класс, хранящий данные об разнице абсолютных и вычисленных координат.
- `statsForPlt` – вспомогательный класс, содержащий информацию в виде необходимом для построения графиков.
- `paramsCreateStats` – вспомогательный класс содержащий параметры для функции преобразования вычисленных данных.
- `diffTickResult` – вспомогательный класс хранения информации о вычисленной статистике ошибок за всю игру.

2.8. Выводы

Для решения поставленной задачи и выполнения всех критериев разработаны алгоритмы, позволяющие определять местоположение в условиях недостаточности флагов и прогнозировать действия для недавно исчезнувших из виду объектов. Данное решение позволит обогатить информацию о текущем состоянии для оценки ситуации на игровом поле и принятии решения.

Для реализации программы сначала было выполнено проектирование математической модели будущего решения.

Сформирована архитектура разрабатываемого решения, позволяющая реализовать необходимые для решения функции.

После чего были описаны основные технические решения в рамках реализуемой системы с помощью диаграмм в нотации UML и рисунков. В результате чего было построено 3 диаграмм: 1 диаграмма классов и 2 диаграмм деятельности.

Следующим шагом требуется описать реализованное решение и результаты его тестирования.

3. ПРОГРАММНАЯ РЕАЛИЗАЦИЯ

3.1. Введение

На основе результатов проектирования, представленных в разделе 2, была разработана программа, позволяющая решить поставленную задачу.

Следующий этап работы – описание реализованного решения, функциональных возможностей и проведённых экспериментов.

Также требуется описать тестирование программы в ходе её разработки.

3.2. Используемые технологии

При разработке программы использовался python3, а также библиотеки:

- Pandas
- NumPy
- SeaBorn
- collections
- unittest
- random

3.3. Описание реализованного решения

На основе результатов проектирования разработаны функции преобразования входной информации, алгоритмы определения местоположения игроков на игровом поле, прогнозирования местоположения для недавно исчезнувших из поля зрения объектов, функции статистических подсчётов.

Также после разработки всех функций программы было реализовано модульное и интеграционное тестирование.

3.3.1. Функции преобразования входной информации

Функции преобразования входной информации находятся в модуле обработки входных и вычисленных данных, определения местоположения видимых объектов. К данным функция относятся:

- `readFile` – функция считывания информации из файлов, в которые записаны все события проведённой игры. В данной функции используется функция из вспомогательного модуля: `Remove_Null_or_NAN_Columns` – метод удаления пустых ячеек в считанных данных.
- `createMapViewFlag` – функция создания карты видимых флагов для текущего игрока по каждому записанному такту игры. Для преобразования информации о текущем такте игры используется функция из вспомогательного модуля: `Find_All_Flags` – метод преобразование информации о видимых флагах текущего игрока в текущий момент времени.
- `createMapViewMove` – функция создания карты видимых объектов для текущего игрока по каждому записанному такту игры. Для преобразования информации о текущем такте игры используется функция из вспомогательного модуля: `Find_All_Object` – метод преобразование информации о видимых объектах для игрока в текущий момент времени.

3.3.2. Функции определения местоположения игроков на игровом поле

Функции определения местоположения игроков находятся в модуле обработки входных и вычисленных данных, определения местоположения видимых объектов. К данным функция относятся:

- `calcInfoForTick` – функция определения местоположения игрока, которая использует различные алгоритмы в зависимости от входных данных. При достаточности информации, т.е. флагов больше 2х, используется алгоритм определения местоположения с использованием фильтра Калмана, в котором также используется функция из вспомогательного модуля: `getAnswerForTwoFlags` – метод определения координат объекта на основе двух флагов. Если флагов меньше 2х, то используется алгоритм определения

местоположения в условиях недостаточности флагов, который производит вычисления на основе предыдущих состояний игрока.

- `calcPosOtherPl` – функция определения местоположения видимых динамических объектов. Функция вычисляет местоположение динамических объектов с использованием методов из вспомогательного модуля: `getAnswerForTwoFlags` – определяет координаты объекта на основе двух флагов, `getAnswerForThreeFlags` – определяет координаты объекта на основе трёх флагов, при этом одним за один из флагов берётся вычисленное местоположение текущего игрока.

3.3.3. Функции прогнозирования местоположения для недавно исчезнувших из поля зрения объектов

Функции прогнозирования местоположения для недавно исчезнувших из поля зрения объектов находятся в модуле хранения предыдущих состояний и прогнозирования местоположения. К данным функция относится:

- `removeList` – функция анализа исчезнувших из поля зрения объектов, которая сравнивает объекты текущего и предыдущего состояние мира, и на основе этого определяет скрывшиеся из виду объекты. Для получения предыдущего состояния мира используется функция из этого же модуля – `getLastItem`.

- `predictForDisappearedPlayer` – функция прогнозирования местоположения динамических объектов на основе, сформированного методом `removeList`, списка исчезнувших игроков и предыдущих состояний игрового мира. Вычисления производятся на основе метрик, которые получены на основе использования двух (вычисляется скорость объекта) или трёх предыдущих состояний мира (вычисляется ускорение).

- `savePredictCoords` – функция сохранения спрогнозированных местоположений динамических объектов.

3.3.4. Функции статистических подсчётов

Функции статистических подсчётов находятся в модуле статистики. К данным функция относятся:

- `calculateExpectationAndVariance` – функция сбора общей статистики по средней и среднеквадратической ошибке определения и прогнозирования местоположения динамических объектов.
- `createDataForPlt` – функция преобразования спрогнозированных данных для графического отображения.
- `calcMaxAndMidDistForInterval` – функция определения максимального и среднего удаления игроков от начальной точки за заданный интервал времени.

3.4. Тестирование

Тестирование программы выполнялось с помощью модульного тестирования и интеграционного тестирования. Написание тестов выполнялось с помощью стандартной библиотеки тестирования `unittest`.

Было реализовано 5 модульных тестов для функций:

- Вычисление координат на основе двух видимых флагов.
- Вычисление координат на основе трёх видимых флагов.
- Сохранение информации о местоположении видимых объектов.
- Сохранение информации о текущем состоянии мира.
- Получение списка исчезнувших из поля зрения объектов на предыдущем такте игры.

Было реализовано 2 интеграционных теста:

- Вычисление местоположения текущего игрока в зависимости от количества флагов видимых флагов и определение местоположения видимых объектов.
- Прогнозирование местоположения для недавно исчезнувших из поля зрения объектов.

Старт полной последовательности модульных тестов осуществляется запуском модуля testModule.

В результате покрытие тестами программы составляет 80%.

3.5. Условия эксперимента

Эксперименты проводились на основе каталога ранее записанных игр. Данные для экспериментов взяты в [17].

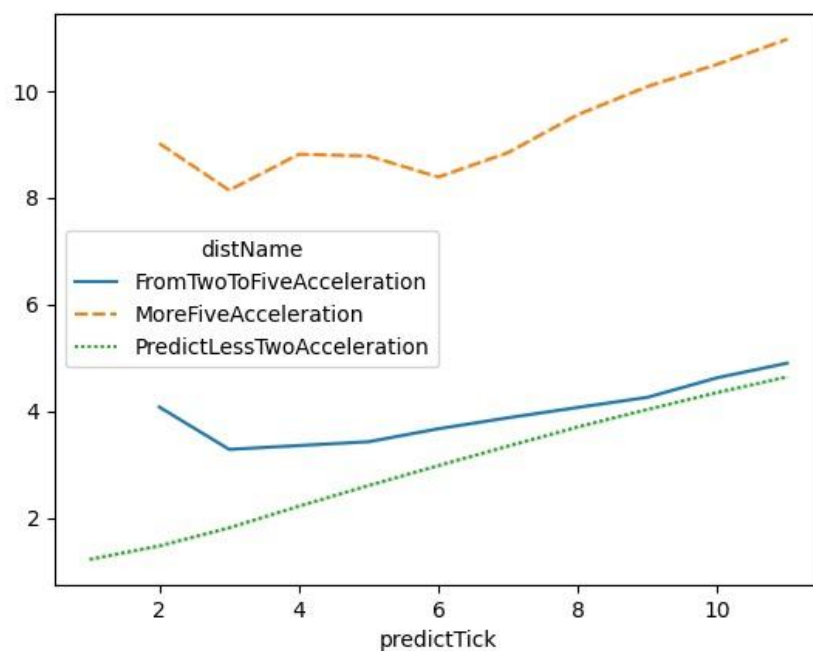
Для представления используется игра между командами helios2017 и oxy2017. Прогнозирование для недавно исчезнувших из виду объектов производится с использованием двух и трёх предыдущих состояний мира. Прогнозирование ведётся 10 тактов (состояний мира) после исчезновения игрока.

3.6. Результаты эксперимента

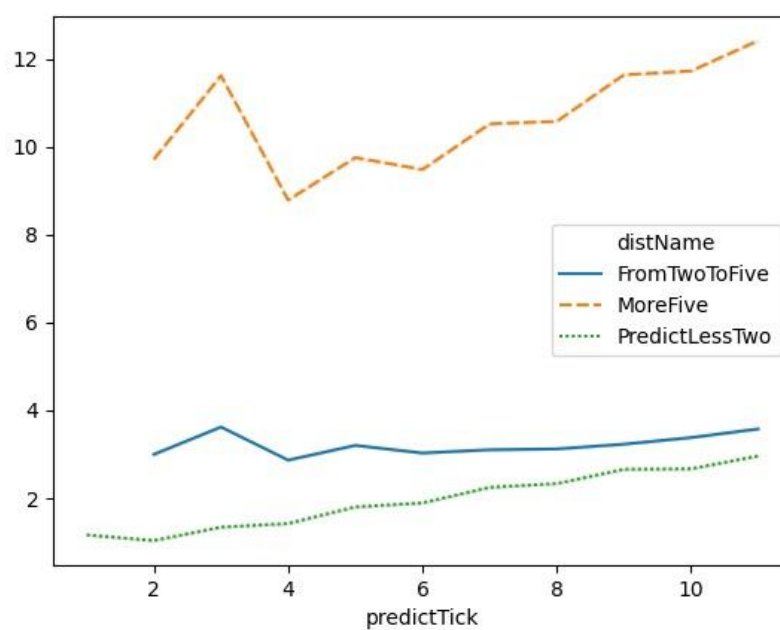
В ходе экспериментов выяснено, что предложенное решение лучше всего предсказывает местоположение для объектов, находящихся на расстоянии от двух до пяти метров от точки начала предсказания, при этом предсказания могут считаться действительными не более чем для десяти тактов игры (рис.12-15). На рисунках по оси ординат показывается ошибка прогнозирования новых координат в зависимости от реальных координат игрока (рисунки 12, 14 – средняя ошибка вычислений, рисунки 13, 15 – СКО ошибки вычисления), по оси абсцисс номера прогнозируемых тактов от момента исчезновения игрока из поля зрения. Данная точность наблюдается как при использовании трёх состояний (на графиках обозначается под буквой – б) при прогнозировании, так и для двух состояний (на графиках обозначается под буквой – а). При этом вычисления на основе трёх состояний формируют меньшее количество предсказанных состояний, но их точность выше, по сравнению с использованием двух состояний.

Ошибка прогнозирования координат растёт с увеличением продолжительности прогноза, что является следствием высокой динамичности игры. При этом для игроков,двигающихся с большой

скоростью, качество прогноза падает. Так для объектов, которые находятся дальше пяти метров от начала точки прогнозирования, расхождение в точности прогнозирования стремительно возрастает. Это связано с тем, что среднее значение перемещения игрока за 10 тактов игры варьируется от 1 до 4 м.

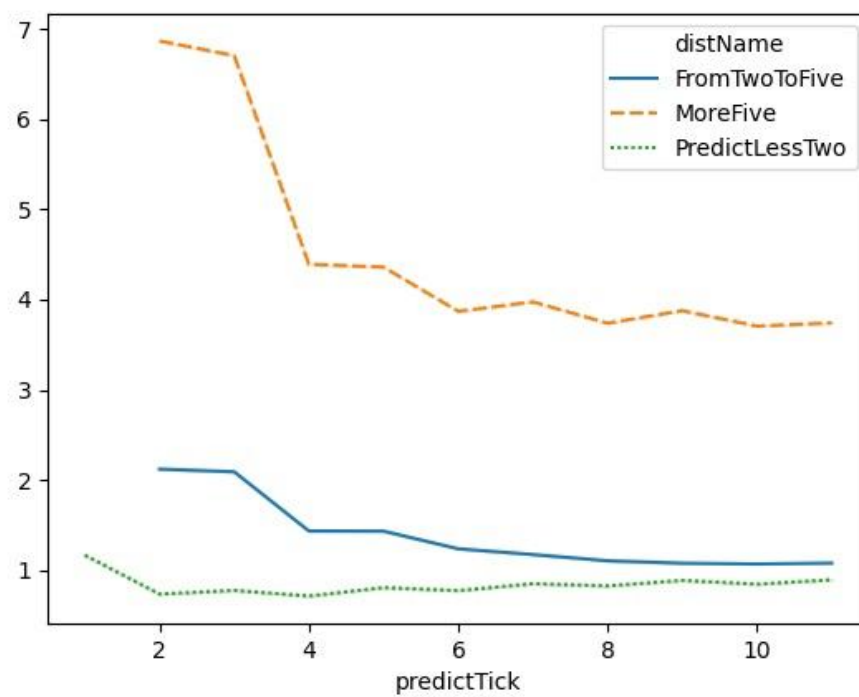


a

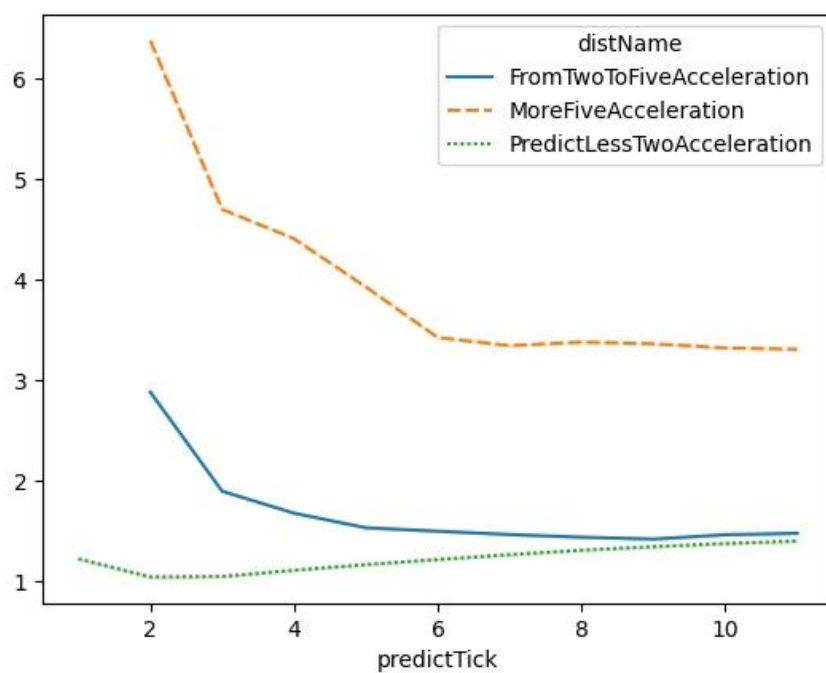


б

Рисунок 12. График разницы спрогнозированных и абсолютных координат для игроков, а – с использованием двух состояний, б – с использованием трёх состояний

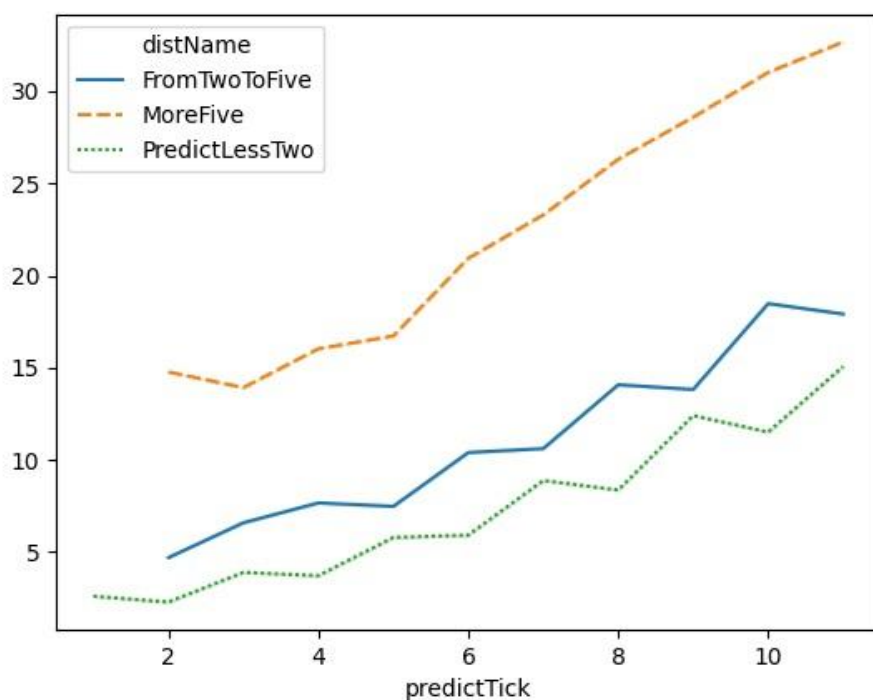


а

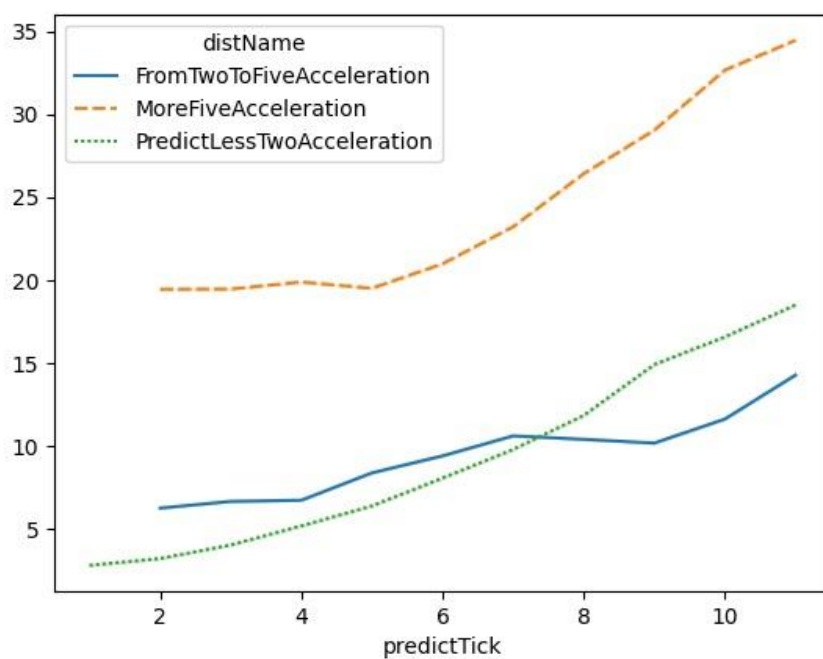


б

Рисунок 13. График среднеквадратичной ошибки спрогнозированных и абсолютных координат для игроков, а – с использованием двух состояний, б – с использованием трёх состояний

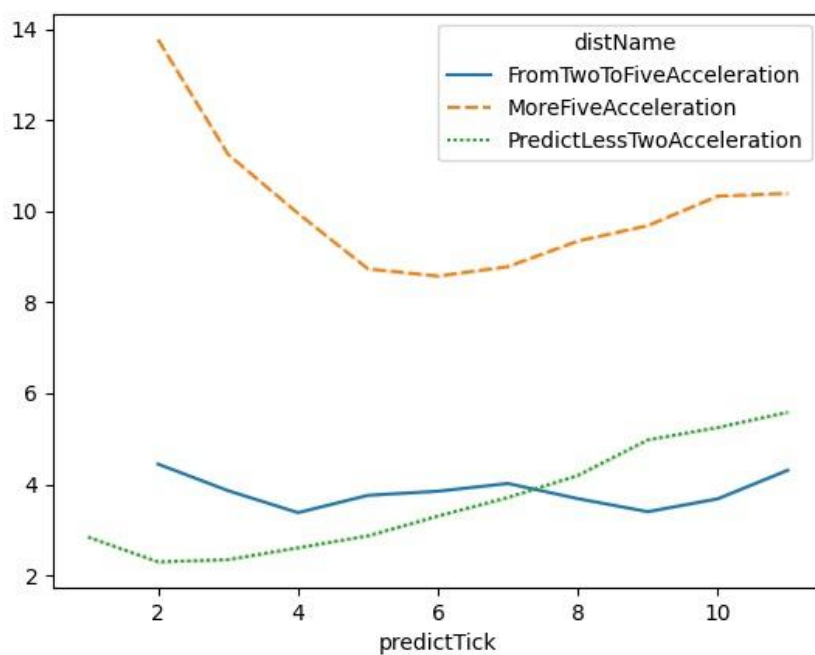


а

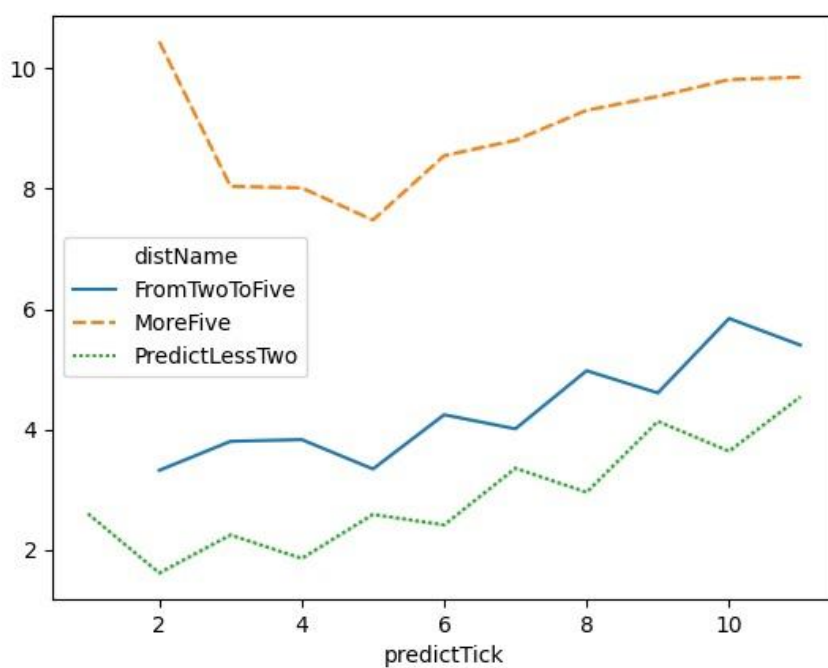


б

Рисунок 14. График разницы спрогнозированных и абсолютных координат для мяча, а – с использование двух состояний, б – с использованием трёх состояний



а



б

Рисунок 15. График среднеквадратичной ошибки спрогнозированных и абсолютных координат для мяча, а – с использование двух состояний, б – с использованием трёх состояний

3.7. Выводы

Была реализована программа, решающая заданные задачи.

Реализованное решение состоит из четырёх модулей, один из которых отвечает за преобразование информации, поступающей с сервера и вычисляющий местоположение объектов, другой отвечает за сохранение информации о текущем состоянии мира и выполняет прогнозирование для недавно исчезнувших из поля зрения объектов, а два других являются вспомогательными. Один из вспомогательных модулей хранит вспомогательные функции для работы программы, ещё один одержит функции для сбора статистических данных.

В ходе разработки проводилось тестирование системы, в результате чего покрытие программы тестами составляет 80.

Предложенное решение апробировано в рамках статьи [18].

4. РАЗРАБОТКА И СТАНДАРТИЗАЦИЯ ПРОГРАММНЫХ СРЕДСТВ

4.1. Введение

Разработка и стандартизации программных средств (ПС) рассматривает вопросы организации проектирования ПС различного назначения с использованием международных и отечественных стандартов, регулирующих основные процессы жизненного цикла ПС и определяющих требования к конечному продукту.

Для рассмотрения организации проектирования ПС необходимо рассмотреть следующие пункты:

- Планирование работ проекта с использованием диаграммы Ганта
- Определение затрат на выполнение и внедрение проекта ПС
- Определение классификационного кода разрабатываемого ПС

4.2. Планирование работ проекта с использованием диаграммы Ганта

4.2.1. Состав частных работ и последовательности их выполнения

Для реализации проекта необходимо выполнение:

1. Формулировка (формализация) перечня требований.
2. Поиск и сравнение аналогов.
3. Разработка математической модели.
4. Разработка архитектуры решения.
5. Реализация модуля определения местоположения игрока.
6. Реализация модуля сохранения предыдущих состояний.
7. Реализация алгоритма определения местоположения в условиях недостаточности ориентиров (флагов).
8. Реализация модуля прогнозирования местоположения для недавно исчезнувших из поля зрения объектов.
9. Проверка корректности работы разработанных модулей.
10. Сбор статистики о работе приложения.

11. Документирование выполненной работы.
12. Написание дополнительной главы.

Последовательность выполнения работ:

- 1 Формулировка (формализация) перечня требований.
- 1, 2 Поиск и сравнение аналогов.
- 2 Разработка математической модели.
2. Разработка архитектуры решения.
3. Реализация модуля определения местоположения игрока.
3. Реализация модуля сохранения предыдущих состояний.
4. Реализация алгоритма определения местоположения в условиях недостаточности ориентиров(флагов).
4. Реализация модуля прогнозирования местоположения для недавно исчезнувших из поля зрения объектов.
5. Проверка корректности работы разработанных модулей.
6. Сбор статистики о работе приложения.
- 6, 7. Документирование выполненной работы.
7. Написание дополнительной главы.

4.2.2. Определение исполнителей и соисполнителей

Исполнители и соискатели для каждой работы:

1. Формулировка (формализация) перечня требований, исполнитель – дипломник (Петруненко Д.А.), соисполнитель – дипломный руководитель дипломного проектирования(Беляев С.А.).
2. Поиск и сравнение аналогов, исполнитель – дипломник (Петруненко Д.А.).
3. Разработка математической модели, исполнитель – дипломник (Петруненко Д.А.).
4. Разработка архитектуры решения, исполнитель – дипломник (Петруненко Д.А.).

5. Реализация модуля определения местоположения игрока, исполнитель – дипломник (Петруненко Д.А.).

6. Реализация модуля сохранения предыдущих состояний, исполнитель – дипломник (Петруненко Д.А.).

7. Реализация алгоритма определения местоположения в условиях недостаточности ориентиров(флагов), исполнитель – дипломник (Петруненко Д.А.).

8. Реализация модуля прогнозирования местоположения для недавно исчезнувших из поля зрения объектов, исполнитель – дипломник (Петруненко Д.А.).

9. Проверка корректности работы разработанных модулей, исполнитель – дипломник (Петруненко Д.А.).

10. Сбор статистики о работе приложения, исполнитель – дипломник (Петруненко Д.А.).

11. Документирование выполненной работы, исполнитель – дипломник (Петруненко Д.А.), соисполнитель – дипломный руководитель дипломного проектирования(Беляев С.А.).

12. Написание дополнительной главы, исполнитель – дипломник (Петруненко Д.А.), соисполнитель – консультант по дополнительному разделу ВКР (Косухина М.А.).

4.2.3. Определение календарных сроков реализации каждой из работ и выполнения всего проекта в целом

Сроки выполнения работ:

1. Формулировка (формализация) перечня требований (Начало работ 22.04.2021 – окончание работ – 24.04.2021).

5. Поиск и сравнение аналогов (Начало работ 24.04.2021 – окончание работ – 29.05.2021).









6. Разработка математической модели (Начало работ 29.04.2021 – окончание работ – 01.05.2021).

7. Разработка архитектуры решения (Начало работ 01.05.2021 – окончание работ – 04.05.2021).
8. Реализация модуля определения местоположения игрока (Начало работ 04.05.2021 – окончание работ – 07.05.2021).
9. Реализация модуля сохранения предыдущих состояний (Начало работ 07.05.2021 - окончание работ – 09.05.2021).
10. Реализация алгоритма определения местоположения в условиях недостаточности ориентиров(флагов) (Начало работ 09.05.2021 – окончание работ – 12.05.2021).
11. Реализация модуля прогнозирования местоположения для недавно исчезнувших из поля зрения объектов (Начало работ 09.05.2021 – окончание работ – 12.05.2021).
12. Проверка корректности работы разработанных модулей (Начало работ 12.05.2021 - окончание работ – 13.05.2021).
13. Сбор статистики о работе приложения (Начало работ 13.05.2021– окончание работ – 20.05.2021).
14. Документирование выполненной работы (Начало работ 20.05.2021 – окончание работ – 27.05.2021).
15. Написание дополнительной главы (Начало работ 20.05.2021 – окончание работ – 27.05.2021).

4.2.4 Диаграмма Ганта

Реализация диаграммы Ганта представлено в табл. 2.

Таблица 2 – Диаграмма Ганта

Работы	Начало	Конец	Временные периоды (1 недели)					Исполнители
			I	II	III	IV	V	
Формулировка перечня требований	22.04	24.04						дипломник: Петруненко Д.А., дипломный руководитель дипломного проектирования: Беляев С.А.
Поиск и сравнение аналогов	24.04	29.04						дипломник: Петруненко Д.А.
Разработка математической модели	29.04	01.05						дипломник: Петруненко Д.А.
Разработка архитектуры решения	01.05	04.05						дипломник: Петруненко Д.А.
Реализация модуля определения местоположения игрока	04.05	07.05						дипломник: Петруненко Д.А.
Реализация модуля сохранения предыдущих состояний	07.05	09.05						дипломник: Петруненко Д.А.
Реализация алгоритма определения местоположения в условиях недостаточности ориентиров	09.05	12.05						дипломник: Петруненко Д.А.
Реализация модуля прогнозирования местоположения для недавно исчезнувших из	09.05	12.05						дипломник: Петруненко Д.А.

поля зрения объектов								
Проверка корректности работы разработанных модулей	12.05	13.05						дипломник: Петруненко Д.А.
Сбор статистики о работе приложения	13.05	20.05						дипломник: Петруненко Д.А.
Документирование выполненной работы	20.05	27.05						дипломник: Петруненко Д.А., дипломный руководитель дипломного проектирования: Беляев С.А
Написание дополнительной главы	20.05	27.05						дипломник: Петруненко Д.А., консультант по дополнительному разделу ВКР: Косухина М.А.

4.3. Определение затрат на выполнение и внедрение проекта ПС

4.3.1. Расчёт полных затрат в день на работу специалистов

Для каждого расчёта НДС = 20%, Прибыль = 15%, Накладные расходы(Н) = 42%, Страховые выплаты(Ф) = 30,2%. Были рассчитаны полные затраты в день на работу проектировщика (см. табл. 3), дипломного руководителя (см. табл. 4), консультанта по дополнительному разделу ВКР (см. табл. 5).

Таблица 3. Расчёт полных затрат в день на работу проектировщика (дипломника)

Наименование статей	ед. измерения	нормативы /затраты	Примечание
Величина среднемесячной начисленной заработной платы специалиста (Нс1)	руб./месяц	30000,00	Оклад сотрудника Ззп
Среднемесячное количество рабочих дней (Тср)	дней./месяц	20,69	Среднее за 2019-2022 годы
Расчет ставки специалиста в день:			
Тарифная ставка дневная (Нс),	руб./день	1449,98	$Z_d = Z_{зп} / T_{ср}$
Страховые взносы 30,2% от суммы зарплаты работников,	руб./день	437,89	$C_{сд} = Z_d \times \Phi$
Оплата основных работников с со страховыми взносами (Zдс),	руб./день	1887,87	$Z_{дс} = Z_d + C_{сд} = Z_d \times (1 + \Phi)$
Накладные расходы (Снр)	руб./день	608,99	$C_{нр} = Z_d \times H$
Себестоимость одного человек/дня (Сч/д),	руб./день	2496,86	$C_{ч/д} = Z_{дс} + C_{нр} = Z_d \times (1 + \Phi + H)$
Дневная прибыль (Спрд),	руб./день	374,53	$C_{прд} = C_{ч/д} \times \Pi$
Ставка специалиста без учета НДС (Сдсс),	руб./день	2 871,39	$C_{дсс} = C_{ч/д} + C_{прд}$
Дневная сумма НДС (Сндс),	руб./день	574,28	$C_{ндс} = C_{дсс} \times \text{НДС}$
Ставка специалиста в день с учётом НДС (Сполн),	руб./день	3445,66	Сполн

Таблица 4. Расчёт полных затрат в день на работу дипломного руководителя

Наименование статей	ед. измерения	нормативы /затраты	Примечание
Величина среднемесячной начисленной заработной платы специалиста (Нс1)	руб./месяц	60000,00	Оклад сотрудника Ззп
Среднемесячное количество рабочих дней (Тср)	дней./месяц	20,69	Среднее за 2019-2022 годы
Расчет ставки специалиста в день:			
Тарифная ставка дневная (Нс),	руб./день	2899,95	$Z_d = Z_{зп} / T_{ср}$
Страховые взносы 30,2% от суммы зарплаты работников,	руб./день	875,79	$C_{сд} = Z_d \times \Phi$
Оплата основных работников с со страховыми взносами (Zдс),	руб./день	3775,74	$Z_{дс} = Z_d + C_{сд} = Z_d \times (1 + \Phi)$
Накладные расходы (Снр)	руб./день	1 217,98	$C_{нр} = Z_d \times H$
Себестоимость одного человек/дня (Сч/д),	руб./день	4993,72	$C_{ч/д} = Z_{дс} + C_{нр} = Z_d \times (1 + \Phi + H)$
Дневная прибыль (Спрд),	руб./день	749,06	$C_{прд} = C_{ч/д} \times \Pi$
Ставка специалиста без учета НДС (Сдсс),	руб./день	5 742,77	$C_{дсс} = C_{ч/д} + C_{прд}$
Дневная сумма НДС (Сндс),	руб./день	1 148,55	$C_{ндс} = C_{дсс} \times H_{ндс}$
Ставка специалиста в день с учётом НДС (Сполн),	руб./день	6891,33	Сполн

Таблица 5. Расчёт полных затрат в день на работу консультанта по дополнительному разделу ВКР

Наименование статей	ед. измерения	нормативы /затраты	Примечание
Величина среднемесячной начисленной заработной платы специалиста (Нс1)	руб./месяц	65000,00	Оклад сотрудника Ззп
Среднемесячное количество рабочих дней (Тср)	дней./месяц	20,69	Среднее за 2019-2022 годы
Расчет ставки специалиста в день:			

Тарифная ставка дневная (Нс),	руб./день	3141,61	$Z_d = Z_{zp}/T_{cp}$
Страховые взносы 30,2% от суммы зарплаты работников,	руб./день	948,77	$C_{сд} = Z_d \times \Phi$
Оплата основных работников с со страховыми взносами ($Z_{дс}$),	руб./день	4090,38	$Z_{дс} = Z_d + C_{дс} = Z_d \times (1 + \Phi)$
Накладные расходы ($C_{нр}$)	руб./день	1 319,48	$C_{нр} = Z_d \times H$
Себестоимость одного человек/дня ($C_{ч/д}$),	руб./день	5409,86	$C_{ч/д} = Z_{дс} + C_{нр} = Z_d \times (1 + \Phi + H)$
Дневная прибыль ($C_{прд}$),	руб./день	811,48	$C_{прд} = C_{ч/д} \times \Pi$
Ставка специалиста без учета НДС ($C_{дсс}$),	руб./день	6 221,34	$C_{дсс} = C_{ч/д} + C_{прд}$
Дневная сумма НДС ($C_{ндс}$),	руб./день	1 244,27	$C_{ндс} = C_{дсс} \times \text{НДС}$
Ставка специалиста в день с учётом НДС ($C_{полн}$),	руб./день	7465,61	$C_{полн}$

4.3.2. Расчет цены предлагаемого продукта

Время участия дипломника – 35 дней, коэффициент загрузки – 1.0. Время участия дипломного руководителя дипломного проектирования – 10 дней, Так как дипломный руководитель дипломного проектирования активно учувствовал во внесении правок в оформление документации по выполненной работе, то его коэффициент загрузки – 7%, что является максимальным значением из рекомендуемого диапазона 5–7%.

Время участия консультанта по дополнительному разделу ВКР – 4 дней. Так как консультант по дополнительному разделу ВКР активно учувствовал во внесении правок в оформление документации по выполненной работе, то его коэффициент загрузки взято среднее значение – 4% из рекомендуемого диапазона 3-5%.

Тогда цена проекта $C_{np} = 3445,66 \text{ (руб./день)} \times 35 \text{ (день)} \times 1.0 \text{ (\%/100)} + 6891,33 \text{ (руб./день)} \times 10 \text{ (день)} \times 0.07 \text{ (\%/100)} + 7465,61 \text{ (руб./день)} \times 4 \text{ (день)} \times 0.04 \text{ (\%/100)} = 126\,616,3 \text{ рублей.}$

С учётом того, что НДС = 20%, прибыль = 15%, $C_{\text{изгот}} = 5000$ рублей, то цена предполагаемого продукта $C_{\text{прогр}} = 126\,616,3$ (рублей) + 5000 (рублей) $\times (1 + 0,15 (\%/100)) \times (1 + 0,20 (\%/100)) = 133\,516,3$ рублей.

4.4. Определение классификационного кода разрабатываемого ПС

4.4.1. Определение кода в соответствии с действующим классификатором ОКПД 2

Код по классификации ОКПД 2 – 62.01.29.000 (Оригиналы программного обеспечения прочие). Определение выполнялось с помощью ресурса [19].

4.4.2. Определение кода по классификатору ОКП

Полученная программа относится к классу – программные средства и информационные продукты вычислительной техники, подклассу – Прикладные программные средства для научных исследований, группе – Программные средства для моделирования и исследования. Далее нет уточняющих кодов, поэтому код по классификатору ОКП – 503200. Определение выполнялось с помощью ресурса [20].

4.5. Выводы

Для планирования работ проекта с использованием диаграммы Ганта был описан состав частных работ и последовательности их выполнения, определены исполнители и соисполнители, определены календарные сроки реализации каждой из работ и выполнения всего проекта в целом. На основе этого составлена диаграмма Ганта, которая наглядно отражает выполнение работ во времени и показывает ответственных за каждую их работ.

Было выполнено определение затрат на выполнение и внедрение проекта ПС, где были подсчитаны затраты на работу каждого из специалистов, а также рассчитана цена предлагаемого продукта.

Также был определён классификационный код разрабатываемого ПС в соответствии с классификатором ОКП и действующим классификатором ОКПД 2.

ЗАКЛЮЧЕНИЕ

Для достижения поставленной цели были решены следующие задачи:

1) Выбор подходящего метода для определения местоположения игроков.

В качестве существующих решений были выбраны методы: навигация по ближайшему флагу и дальней линии, навигация по двум ближайшим флагам и дальней линии, навигация с использованием фильтра Калмана, навигация с использованием фильтра частиц, метод Монте-Карло и агрегации данных (MCSDA), метод RFS (случайных конечных множеств), концепция построения интеллектуальных агентов реального. Для каждого решения было приведено краткое описание. Для выбранных существующих решений было выполнено сравнение по заданным критериям. В результате сравнения был выбран метод навигации с использованием фильтра Калмана, как основной метод вычисления местоположения в условиях, когда информации достаточно для работы. Но для обогащения модели и улучшения точности принятия решения, было принято решение о разработки алгоритмов определения местоположения в условиях недостаточности флагов, а также прогнозировании координат для недавно исчезнувших из виду динамических объектов.

2) Проектирование математической модели

Для выполнения поставленных целей была разработана математическая модель решения и составлен пример данных, генерируемых данной моделью.

3) Проектирование архитектуры.

На основе поставленных целей и построенной математической модели были сформированы архитектура разрабатываемого решения.

Основные технические решения описаны с помощью диаграмм в нотации UML, всего было построено 3 диаграммы: 1 диаграмма классов, 2 диаграммы деятельности.

4) Реализация алгоритма определения местонахождения объектов и прогнозирования местоположения для недавно исчезнувших из поля зрения игроков.

Реализованное решение состоит из четырёх модулей, один из которых отвечает за преобразование информации, поступающей с сервера, и определение своего местонахождения и местонахождения видимых динамических объектов, другой отвечает за сохранение информации о текущем состоянии мира и выполняет прогнозирование для недавно исчезнувших из поля зрения объектов, а два других являются вспомогательными. Один из вспомогательных модулей хранит вспомогательные функции для работы программы, ещё один одержит функции для сбора статистических данных.

5) Проведение экспериментов для определения точности полученного решения.

В ходе экспериментов выяснено, что предложенное решение лучше всего предсказывает местоположение для объектов, находящихся на расстоянии от двух до пяти метров от точки начала предсказания, при этом предсказания могут считаться действительными не более чем для десяти тактов игры. Данная точность наблюдается как при использовании трёх состояний при прогнозировании, так и для двух состояний. При этом вычисления на основе трёх состояний формируют меньшее количество предсказанных состояний, но их точность выше, по сравнению с использованием двух состояний

6) Тестирование.

Для тестирования программы были написано

- 5 модульных тестов. 2 теста покрывают вычисление координат по двум и трём видимым флагам на поле, 1 тестирует анализ исчезнувших игроков, 2 проверяют корректность сохранения информации о вычисленном состоянии.
- 2 интеграционных теста. 1 тест выполняет проверку корректности предсказания координат для исчезнувших из виду игроков, 1 тестирует

корректность определения местоположения игрока и видимых ему объектов на поле для текущего состояния.

В результате покрытие тестами программы составляет 80%.

Таким образом, система вычисления местоположения видимых объектов и построения прогноза для исчезнувших из вида объектов позволяет определить координаты текущего игрока с использованием фильтра Калмана при наличии двух и более видимых флагов, а при недостатке флагов определение выполняется на основе предыдущих состояний. Также позволяет определить местоположение других игроков, а для недавно исчезнувших из поля зрения объектов спрогнозировать их местоположение. Данная система может быть использована совместно с системами принятия решения. Следующий шаг – проверить работу данной системы во время соревнований совместно с системами принятия решений.

Предложенное решение было апробировано статье [18].

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Suzuki Y., Fukushima T., Thibout L., Nakashima T., Akiyama H. (2019) Game-Watching Should be More Entertaining: Real-Time Application of Field-Situation Prediction to a Soccer Monitor. In: Chalup S., Niemueller T., Suthakorn J., Williams MA. (eds) RoboCup 2019: Robot World Cup XXIII. RoboCup 2019. Lecture Notes in Computer Science, vol 11531. Springer, Cham. https://doi.org/10.1007/978-3-030-35699-6_35
2. Visser A., Nardin L.G., Castro S. (2019) Integrating the Latest Artificial Intelligence Algorithms into the RoboCup Rescue Simulation Framework. In: Holz D., Genter K., Saad M., von Stryk O. (eds) RoboCup 2018: Robot World Cup XXII. RoboCup 2018. Lecture Notes in Computer Science, vol 11374. Springer, Cham. https://doi.org/10.1007/978-3-030-27544-0_39.
3. Akiyama H., Nakashima T. (2014) HELIOS Base: An Open Source Package for the RoboCup Soccer 2D Simulation. In: Behnke S., Veloso M., Visser A., Xiong R. (eds) RoboCup 2013: Robot World Cup XVII. RoboCup 2013. Lecture Notes in Computer Science, vol 8371. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-662-44468-9_46.
4. S. A. Belyaev, "Mathematical Model of the Player Control in Soccer Simulation," 2021 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (ElConRus), St. Petersburg, Moscow, Russia, 2021, pp. 233-237, doi: 0.1109/ElConRus51938.2021.9396517.
5. Akiyama H., Nakashima T., Fukushima T., Zhong J., Suzuki Y., Ohori A. (2019) HELIOS2018: RoboCup 2018 Soccer Simulation 2D League Champion. In: Holz D., Genter K., Saad M., von Stryk O. (eds) RoboCup 2018: Robot World Cup XXII. RoboCup 2018. Lecture Notes in Computer Science, vol 11374. Springer, Cham. https://doi.org/10.1007/978-3-030-27544-0_37.
6. Cano P., Ruiz-del-Solar J. (2017) Robust Tracking of Multiple Soccer Robots Using Random Finite Sets. In: Behnke S., Sheh R., Sarel S., Lee D. (eds) RoboCup 2016: Robot World Cup XX. RoboCup 2016. Lecture Notes in Computer Science, vol 9776. Springer, Cham. https://doi.org/10.1007/978-3-319-68792-6_17.

7. Using Monte Carlo Search With Data Aggregation to Improve Robot Soccer Policies / Riccio, Francesco; Capobianco, Roberto; Nardi, Daniele. - 9776(2017), pp. 256-267. (Intervento presentato al convegno 20th Annual RoboCup International Symposium, 2016 tenutosi a Leipzig; Germany. ISBN 978-331968791-9 978-3-319-68792-6

8. E. V. Postnikov, S. A. Belyaev, A. V. Ekalo and A. A. Shkulev, "Application of Fuzzy State Machines to Control Players in Virtual Soccer Simulation," 2019 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus), Saint Petersburg and Moscow, Russia, 2019, pp. 291-294, doi: 10.1109/EIConRus.2019.8657109

9. Беляев С. А. Применение вероятностных и временных автоматов в программах управления многоагентных систем //Научные технологии в космических исследованиях Земли. – 2020. – Т. 12. – №. 3. – С. 47–53.

10. Pomas T., Nakashima T. (2019) Evaluation of Situations in RoboCup 2D Simulations Using Soccer Field Images. In: Holz D., Genter K., Saad M., von Stryk O. (eds) RoboCup 2018: Robot World Cup XXII. RoboCup 2018. Lecture Notes in Computer Science, vol 11374. Springer, Cham. https://doi.org/10.1007/978-3-030-27544-0_23.

11. Пантелеев М. Г. Концепция построения интеллектуальных агентов реального времени на основе модели опережающего итеративного планирования //Тринадцатая национальная конференция по искусственному интеллекту с международным участием КИИ-2012: Труды конференции. – 2012. – Т. 3 – С. 25-33.

12. Беляев С.А. Интеллектуальные системы. программирование игроков в виртуальном футболе: Лабораторный практикум. //Спб.: Издательство СПбГЭТУ «ЛЭТИ» – 2020. – 62 с.

13. Пантелеев М. Г., Салимов А. Ф. Анализ алгоритмов навигации интеллектуального агента в виртуальном футболе. //Известия СПбГЭТУ ЛЭТИ. – 2020. – Т. 1 – С. 60-70.

14. Дубровин Ф. С., Щербатюк А. Ф. Исследование некоторых алгоритмов однопаяковой мобильной навигации АНПА: результаты моделирования и морских испытаний. //Гироскопия и навигация. – 2015. – №. 4. – С. 160-172.

15. Кучерский Роман Владимирович, Манько Сергей Викторович Алгоритмы локальной навигации и картографии для бортовой системы управления автономного мобильного робота //Известия ЮФУ. Технические науки. – 2012. – Т. 3 – С. 13-22.

16. D. A. Chentsov and S. A. Belyaev, "Monte Carlo Tree Search Modification for Computer Games," 2020 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus), St. Petersburg and Moscow, Russia, 2020, pp. 252-255, doi: 10.1109/EIConRus49466.2020.9039281

17. [Электронный ресурс] // Исходные данные: RoboCupSimData Files Overview. URL: <http://oliver.obst.eu/data/RoboCupSimData/overview.html> (дата обращения: 10.05.2021).

18. Petrunenko D. A., Belyaev S. A. «Determining the Location of Players in Virtual Soccer», журнал «Software Journal: Theory and Applications», в печати.

19. [Электронный ресурс] //Общероссийский классификатор продукции по видам экономической деятельности. URL: <https://classifikators.ru/okpd> (дата обращения: 20.05.2021).

20. [Электронный ресурс] //Общероссийский классификатор продукции. URL: <https://classifikators.ru/okp> (дата обращения: 20.05.2021).

ПРИЛОЖЕНИЕ А

Тестовые сценарии для клиентской части

Таблица А.1 – Свойства класса «storeAgent»

Название свойства	Тип данных
storeCoord	infoForTick[]
removePlayer	{[key: string]: posPlayer[]}

Таблица А.2 – Методы класса «storeAgent»

Название метода	Тип входных данных	Тип выходных данных
addNewTickInfo	infoForTick	void
getLastItem	–	infoForTick
getItemAt	int	infoForTick
getLength	–	int
removeList	–	string[]
predictForDisappearedPlayer	string[]	predictCoordinate[]
savePredictCoords	predictCoordinate	void

Таблица А.3 – Свойства класса «predictCoordinate»

Название свойства	Тип данных
x	double
y	double
name	string
beforeX	double
beforeY	double
angle	int
predictTick	int

Таблица А.4 – Свойства класса «infoForTick»

Название свойства	Тип данных
x	double
y	double
angle	int
absX	double
absY	double

speedX	double
speedY	double
Players	otherPlayer

Таблица А.5 – Свойства класса «otherPlayer»

Название свойства	Тип данных
viewPlayer	string[]
mapPlayer	{[key: string]: posPlayer[]}

Таблица А.6 – Методы класса «otherPlayer»

Название метода	Тип входных данных	Тип выходных данных
addNewViewPlayer	string,posPlayer	void

Таблица А.7 – Свойства класса «posPlayer»

Название свойства	Тип данных
x	double
y	double
angle	int

Таблица А.8 – Методы класса «processInputData»

Название метода	Тип входных данных	Тип выходных данных
readFile	pandas.DataFrame,pandas.DataFram	readInfo
createMapViewFlag	{[key: String]:{[key: String]: infoViewFlag}}, String:[key: String]: pandas.DataFrame}}	{[key: String]:{[key: String]: infoViewFlag[]}}
createMapViewMove	{[key: String]:{[key: String]: infoViewFlag}},{[key: String]:{key: String}: pandas.DataFrame}}	{[key: String]:{[key: String]: infoViewObject[]}}

calcPosOtherPl	paramsForCalcPosition, {[key: String]: {[key: String]: infoViewObject[]}}, String, Int	otherPlayer
calcInfoForTick	paramsForCalcPosition, {[key: String]: {[key: String]: infoViewObject[]}}, String, Int, coordinate[]	paramsForCalcPosition
createDataTickWithPredictVal	paramsForDataTickWithPredictVal, String	predictInfo

Таблица А.9 – Свойства класса «coordinate»

Название свойства	Тип данных
x	double
y	double

Таблица А.10 – Свойства класса «predictInfo»

Название свойства	Тип данных
viewFrom	string
timeNow	int
nowX	double
nowY	double
nowAbsoluneX	double
nowAbsoluneY	double
predictX	double
predictY	double
predictAbsoluneX	double
predictAbsoluneY	double
nowDiffX	double
nowDiffY	double
predictDiffX	double
predictDiffY	double

predictVarianceX	double
predictVarianceX	double
predictTick	int

Таблица А.11 – Свойства класса «paramsForDataTickWithPredictVal»

Название свойства	Тип данных
listPredict	predictCoordinate
predictObj	predictInfo
angleOrientation	int
elems	infoViewObject

Таблица А.12 – Свойства класса «infoViewFlag»

Название свойства	Тип данных
time	int
flags	infoFlag

Таблица А.13 – Свойства класса «infoFlag»

Название свойства	Тип данных
column	string
dist	double
angle	int

Таблица А.14 – Свойства класса «infoViewObject»

Название свойства	Тип данных
time	int

Таблица А.15 – Свойства класса «infoObject»

Название свойства	Тип данных
plArr	infoFlag
ballArr	infoFlag

Таблица А.16 – Свойства класса «paramsForCalcPosition»

Название свойства	Тип данных
elems	infoViewObject
nowPIObj	storeAgent
angleOrientation	int
valueLackFlag	int

varianceArray	int[]
angleFlag	int
absoluteX	double
absoluteY	double
averageX	double
averageY	double
arrPlayer	otherPlayer
radian	double
speedX	double
speedY	double

Таблица А.17 – Методы класса «getCoords»

Название метода	Тип входных данных	Тип выходных данных
Remove_Null_or_NAN_Columns	pandas.DataFrame	pandas.DataFrame
Find_All_Flags	pandas.Series	infoFlag[]
Find_All_Object	pandas.Series	infoObject
getAnswerForThreeFlags	coordinate, coordinate, coordinate, Double, Double, Double	coordinate
getAnswerForTwoFlags	coordinate, coordinate, Double, Double	coordinate
coordsForSeemX	coordinate[], int[], Int, Int,Int	coordinate
coordsForSeemY	coordinate[], int[], Int, Int,Int	coordinate
checkAnswersForTwoFlags	coordinate[], coordinate[]	coordinate

getAbsolutedCoordinate	string, int, string, ing, boolean	absoluteCoords
------------------------	--------------------------------------	----------------

Таблица А.18 – Свойства класса «readInfo»

Название свойства	Тип данных
resFlags	{[key: String]:{[key: String]: pandas.DataFrame}}
resMov	{[key: String]:{[key: String]: pandas.DataFrame}}

Таблица А.18 – Свойства класса «absoluteCoords»

Название свойства	Тип данных
absoluteX	double
absoluteY	double
angleFlag	int
angleOrientation	int
nowPlayer	string

Таблица А.20 – Методы класса «statistic»

Название метода	Тип входных данных	Тип выходных данных
calculateExpectationAndVariance	pandas.DataFrame, coordinate[], string	pandas.DataFrame
createDataForPlt	pandas.DataFrame, pandas.DataFrame, pandas.DataFrame, string	infoFlag[]
calcSizeMid	diffTickVal[]	diffTickVal
calcEuclidDist	coordinate	double
calcMaxAndMidDistForInterval	coordinate[], int, string	diffTickResult
addDataForStatDist	paramsCreateStats, boolean	paramsCreateStats

Таблица А.21 – Свойства класса «paramsCreateStats»

Название свойства	Тип данных
predictObj	predictInfo

resultPredictMoreFiveBallDF	pandas.DataFrame
resultPredictFromTwoToFiveBallDF	pandas.DataFrame
resultPredictMoreFiveDF	pandas.DataFrame
resultPredictFromTwoToFiveDF	pandas.DataFrame
resultPredictLessTwoDF	pandas.DataFrame
resultPredictStatisticLessTwoDF	pandas.DataFrame
sresultPredictStatisticFromTwoToFiveDF	pandas.DataFrame
resultPredictStatisticMoreFiveDF	pandas.DataFrame

Таблица А.22 – Свойства класса «diffTickResult»

Название свойства	Тип данных
maxValue	double
maxValueX	double
maxValueY	double
midValue	double
midValueX	double
midValueY	double
typeCoord	double

Таблица А.23 – Свойства класса «statsForPlt»

Название свойства	Тип данных
DFAddOne	pandas.DataFrame
DFAddTwo	pandas.DataFrame

Таблица А.24 – Свойства класса «diffTickVal»

Название свойства	Тип данных
diffX	double
diffY	double
diff	double