Министерство образования и науки Российской Федерации
федеральное государственное автономное образовательное учреждение
высшего образования
**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Школа информационных технологий и робототехники
Направление подготовки 09.03.01 «Информатика и вычислительная техника»
Отделение информационных технологий

БАКАЛАВРСКАЯ РАБОТА

Тема работы
Разработка окружения для тестирования и настройки игр с математической моделью

УДК 004.415.53:004.925.84:519.876

Студент

Группа	ФИО	Подпись	Дата
8В5Б	Ивченко Александр Юрьевич		

Руководитель

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент отделения ИТ	Ботыгин Игорь Александрович	к.т.н., доцент		

Со-руководитель (по разделу «Концепция стартап-проекта»)

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент ШИП	Пожарницкая Ольга Вячеславовна	к.э.н., доцент		

КОНСУЛЬТАНТЫ:

По разделу «Социальная ответственность»

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Ассистент ООД ШБИП	Мезенцева Ирина Леонидовна	ассистент		

Нормоконтроль

Должность	ФИО	Ученая степень, звание	Подпись	Дата

ДОПУСТИТЬ К ЗАЩИТЕ:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Руководитель ООП	Погребной Александр Владимирович	к.т.н.		

ЗАПЛАНИРОВАННЫЕ РЕЗУЛЬТАТЫ ОБУЧЕНИЯ ПО ПРОГРАММЕ

Код результатов	Результат обучения (выпускник должен быть готов)
Р1	Применять базовые и специальные естественнонаучные и математические знания в области информатики и вычислительной техники, достаточные для комплексной инженерной деятельности.
Р2	Применять базовые и специальные знания в области современных информационных технологий для решения инженерных задач.
Р3	Ставить и решать задачи комплексного анализа, связанные с созданием аппаратно-программных средств информационных и автоматизированных систем, с использованием базовых и специальных знаний, современных аналитических методов и моделей.
Р4	Разрабатывать программные и аппаратные средства (системы, устройства, блоки, программы, базы данных и т. п.) в соответствии с техническим заданием и с использованием средств автоматизации проектирования.
Р5	Проводить теоретические и экспериментальные исследования, включающие поиск и изучение необходимой научно-технической информации, математическое моделирование, проведение эксперимента, анализ и интерпретация полученных данных, в области создания аппаратных и программных средств информационных и автоматизированных систем.
Р6	Внедрять, эксплуатировать и обслуживать современные программно-аппаратные комплексы, обеспечивать их высокую эффективность, соблюдать правила охраны здоровья, безопасность труда, выполнять требования по защите окружающей среды.
	Универсальные компетенции
Р7	Использовать базовые и специальные знания в области проектного менеджмента для ведения комплексной инженерной деятельности.
Р8	Владеть иностранным языком на уровне, позволяющем работать в иноязычной среде, разрабатывать документацию, презентовать и защищать результаты комплексной инженерной деятельности.
Р9	Эффективно работать индивидуально и в качестве члена группы, состоящей из специалистов различных направлений и квалификаций, демонстрировать ответственность за результаты работы и готовность следовать корпоративной культуре организации.
Р10	Демонстрировать знания правовых, социальных, экономических и культурных аспектов комплексной инженерной деятельности.
Р11	Демонстрировать способность к самостоятельному обучению в течение всей жизни и непрерывному самосовершенствованию в инженерной профессии.

Министерство образования и науки Российской Федерации
федеральное государственное автономное образовательное учреждение
высшего образования
**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Школа информационных технологий и робототехники
Направление подготовки: 09.03.01 «Информатика и вычислительная техника»
Отделение информационных технологий

УТВЕРЖДАЮ:
Руководитель ООП
_____Погребной А.В.
(Подпись) (Дата) (Ф.И.О.)

ЗАДАНИЕ
на выполнение выпускной квалификационной работы

В форме:

Бакалаврской работы

Студенту:

Группа	**ФИО**
8В5Б	Ивченко Александру Юрьевичу

Тема работы:

Разработка окружения для тестирования и настройки игр с математической моделью	
Утверждена приказом директора (дата, номер)	15.02.19, №1217/с

Срок сдачи студентом выполненной работы:	

ТЕХНИЧЕСКОЕ ЗАДАНИЕ:

Исходные данные к работе	Техническое задание к реализации программного окружения для тестирования и настройки игр с математической моделью
Перечень подлежащих исследованию, проектированию и разработке вопросов	Изучение существующих методов для тестирования и настройки игр с математической моделью; Выбор решений для тестирования и настройки игр с математической моделью; Реализация решений для тестирования и настройки игр с математической моделью;

	Концепция стартап-проекта; Анализ вредных производственных факторов.
Перечень графического материала	Архитектура модуля для регрессионного тестирования. Бизнес-модель Остервальдера

Консультанты по разделам выпускной квалификационной работы	
Раздел	**Консультант**
Концепция стартап-проекта	Доцент ШИП Пожарницкая О. В.
Социальная ответственность	Ассистент ООД ШБИП Мезенцева И. Л.

Дата выдачи задания на выполнение выпускной квалификационной работы по линейному графику	

Задание выдал руководитель:

Должность	**ФИО**	**Ученая степень, звание**	**Подпись**	**Дата**
Доцент отделения ИТ	Ботыгин Игорь Александрович	к.т.н., доцент		

Задание принял к исполнению студент:

Группа	**ФИО**	**Подпись**	**Дата**
8В5Б	Ивченко Александр Юрьевич		

Министерство образования и науки Российской Федерации
федеральное государственное автономное образовательное учреждение высшего образования
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Школа информационных технологий и робототехники

Направление подготовки: 09.03.01 Информатика и вычислительная техника

Уровень образования: Бакалавриат

Отделение информационных технологий

Период выполнения: весенний семестр 2018/2019 учебного года

Форма представления работы:

Бакалаврская работа

КАЛЕНДАРНЫЙ РЕЙТИНГ-ПЛАН
выполнения выпускной квалификационной работы

Срок сдачи студентом выполненной работы:	

Дата контроля	Название раздела (модуля) / вид работы (исследования)	Максимальный балл раздела (модуля)
20.05.2019	Основная часть	80
22.05.2019	Концепция стартап-проекта	10
14.06.2019	Социальная ответственность	10

Составил преподаватель:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент отделения ИТ	Ботыгин Игорь Александрович	к.т.н., доцент		

СОГЛАСОВАНО:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Руководитель ООП	Погребной Александр Владимирович	к.т.н.		

РЕФЕРАТ

Пояснительная записка содержит 81 страница, 28 рисунков, 14 таблиц, 22 источника, 6 приложений.

Ключевые слова: Java, генетический алгоритм, регрессионное тестирование, программирование.

Объектом исследования является разработка игр с математической моделью.

Цель работы – разработка программного окружения для тестирования и настройки игр с математической моделью.

В результате исследования изучены существующие методы для тестирования и настройки игр с математической моделью и выдвинуты новые решения для выполнения этих задач.

Основные конструктивные, технологические и технико-эксплуатационные характеристики: использование языка программирования Java.

Степень внедрения: Используется в компанию ООО «Коннектив Геймс», г. Томск.

Область применения: разработка компьютерных игр.

Значимость работы заключается в проектировании программного обеспечения, которое частично автоматизирует и ускоряет процесс создания и тестирования новых игр с математической моделью.

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ, СОКРАЩЕНИЯ И НОРМАТИВНЫЕ ССЫЛКИ

ПЗУ – постоянное запоминающие устройства;

ПО – программное обеспечение;

ГПСЧ – генератора псевдослучайных чисел;

ПЭВМ – персональная электронно-вычислительная машина;

ПК – персональный компьютер.

Оглавление

ВВЕДЕНИЕ 10
1. ОБЗОР МЕТОДОВ ДЛЯ ТЕСТИРОВАНИЯ И НАСТРОЙКИ ИГР С МАТЕМАТИЧЕСКОЙ МОДЕЛЬЮ 14
1.1. Способы тестирования игр с математической моделью 14
1.2. Способы настройки игр с математической моделью 14
1.3. Новые решения для тестирования и настройки игр с математической моделью 16
1.4. Необходимый функционал для работы нового решения 17
2. РЕАЛИЗАЦИЯ РЕШЕНИЯ ДЛЯ НАСТРОЙКИ ИГР С МАТЕМАТИЧЕСКОЙ МОДЕЛЬЮ 18
2.1. Реализация необходимого функционала для работы нового решения 18
2.2. Реализация решения для настройки игр с математической моделью 20
3. РЕАЛИЗАЦИЯ РЕШЕНИЯ ДЛЯ ТЕСТИРОВАНИЯ ИГР С МАТЕМАТИЧЕСКОЙ МОДЕЛЬЮ 31
3.1. Реализация решения. 31
3.2. Реализация вспомогательных решений для нахождения минимального количества для итераций сбора статистики. 36
4. КОНЦЕПЦИЯ СТАРТАП-ПРОЕКТА 42
4.1. Введение в предметную область: 42
4.2. Проблема: 43
4.3. Описание продукта: 44
4.4. Потребительские качества: 44
4.5. Защита интеллектуальной собственности: 45
4.6. Объем и емкость рынка 45
4.7. Анализ современного состояния и перспектив развития отрасли 46
4.8. Планируемая стоимость продукта 46
4.9. Конкурентные преимущества создаваемого продукта, сравнение технико-экономических характеристик с отечественными и мировыми аналогами. 47
4.10. Описание целевых сегментов потребителей создаваемого продукта 49
4.11. Бизнес модель проекта. Производственный план. План продаж. 50
4.12. Стратегия продвижения продукта на рынок 52
5. СОЦИАЛЬНАЯ ОТВЕТСВЕННОСТЬ 54
5.1. Производственная безопасность 55
5.1.1. Отклонение показателей микроклимата 55

5.1.2. Превышение уровня шума 56
5.1.3. Отсутствие или недостаток естественного света и недостаточная освещенность рабочей зоны 58
5.1.4.Электромагнитное излучение 58
5.1.5. Электробезопасность..... 59
5.2. Экологическая безопасность 62
5.3. Безопасность в чрезвычайных ситуациях 63
5.4. Правовые и организационные вопросы обеспечения безопасности 63
ЗАКЛЮЧЕНИЕ ПО РАЗДЕЛУ 66
ЗАКЛЮЧЕНИЕ 67
СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 68
Приложение А 71
Приложение Б..... 73
Приложение В 74
Приложение Г 77
Приложение Д 79
Приложение Е..... 81

ВВЕДЕНИЕ

Компьютерная игра – компьютерная программа, служащая для организации игрового процесса, связи с партнёрами по игре или сама выступающая в качестве партнёра. К компьютерным играм также относят видеоигры и мобильные игры. Существуют попытки выделить компьютерные игры как отдельную область искусства, наряду с театром, кино и т. п. По некоторым компьютерным играм проводятся любительские и профессиональные соревнования. Такого рода соревнования называются киберспортом.

Большинство компьютерных игр включают в себя одну или несколько математических моделей.

В данной работе рассматривается разработка игр, в которых ярко выражена математическая составляющая и присутствует сильное влияние генератора псевдослучайных чисел (ГПСЧ) на результаты. Как правило, в таких играх пользователь ограничен всего несколькими возможными действиями, а управление производится за счет компьютерной мышки (или её аналогов). Далее в работе такие игры будут называться игры с математической моделью.

Каркасом игры будем называть совокупность всех событий и их причин без привязки к числам. Каркас игры задается правилами. Примером части каркаса может послужить гравитация в игре.

Числовой реализацией (или числовым решением) назовем одну конкретную реализацию каркаса игры. Игру с хорошим числовым решением называют сбалансированной. Для примера с гравитацией это может быть ускорение свободного падения или максимальная безопасная высота падения. Большинство игроков назовут игру сбалансированной, если безопасная высота падения будет больше высоты прыжка персонажа.

Математической моделью игры является совокупность уравнений, описывающих каркас игры. Например, для описанного выше случая можно

ввести уравнения зависимости скорости падения от времени или зависимости урона от скорости падения.

Состояние игры – набор динамических внутриигровых параметров, полностью характеризующих момент в игре. Для описанного выше случая – скорость падения, ускорение и высота игрока в некоторый момент времени.

Стратегией игры будем называть совокупность решений и условий их применения, которая максимизирует результаты пользователя в игре. Для случая с гравитацией – не спрыгивать с большой высоты.

Игровое событие – явление в игре, изменяющее её состояние.

Действие игрока – игровое событие, инициируемое игроком.

Статистика игры – численное представление происшедших игровых событий за определенный момент времени (или за определенное количество действий игрока). Для примера с гравитацией статистическими сведениями может являться количество прыжков с разных высот за один час.

На рисунке 1.1 представлена схема, связывающая игровые события, состояние игры, каркас игры, ГПСЧ и числовую реализацию. Каркас игры, опираясь на ГПСЧ и числовую реализацию, применяет Игровое событие 1 к Состоянию 1 и сохраняет статистику, в результате чего рождается Состояние 2 и либо Действие игрока 1, в этом случае система ожидает решения игрока, либо Игровое событие 2, которое вместе с Состоянием 2 снова подается на вход системы.

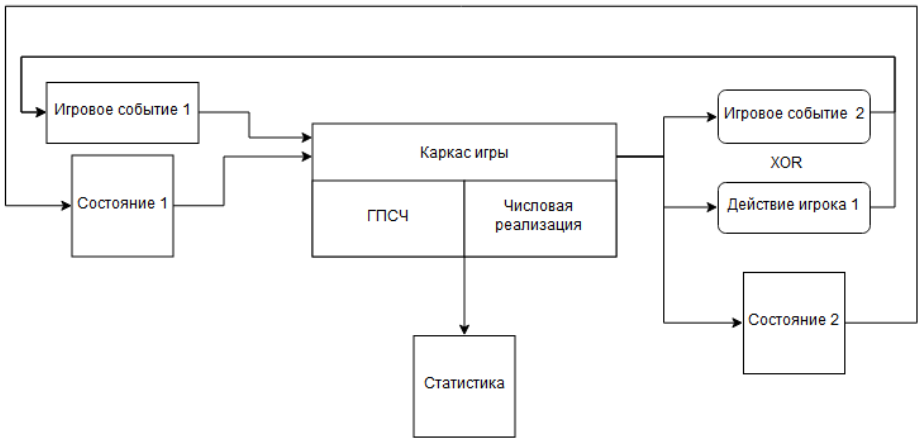

Рисунок 1.1 – Схема связи сущностей

Одной из проблем создания игр с математической моделью является недостаточная оптимизация процесса разработки ПО.

В ходе разработки игр с математической моделью перед разработчиком стоят следующие задачи:

1. Придумать каркас игры.
2. Придумать требования для числовой реализации.
3. Найти математическую модель для каркаса
4. Найти числовую реализацию игры, удовлетворяющую требованиям, при помощи математической модели.
5. Реализовать игру.
   5.1. Реализовать каркас
   5.2. Подставить значения в каркас из найденной числовой реализации
6. Реализовать тесты игры.

Основными проблемами, возникающими при выполнении этих задач, являются:

- Нахождения математической модели игры является нетривиальной задачей даже для простых каркасов. Также сложность нахождения модели увеличивает стоимость даже небольшого изменения каркаса игры;
- Сложность нахождения числовой реализации игры, удовлетворяющей требованиям, связана с тем, что математическая модель игры представлена множеством уравнений и сама числовая реализация, как правило, состоит из множества значений;
- В процессе активной разработки происходят изменения и рефакторинг программного кода. Таким образом, возникает потребность в регрессионном тестировании. Произвести подобное тестирование привычными методами весьма сложно из-за большого количества ситуаций, требующих проверки.

Целью работы является разработка окружения для тестирования и настройки игр с математической моделью.

Задачи:

1. Изучение существующих методов для тестирования и настройки игр с математической моделью. Выбор решений для тестирования и настройки игр с математической моделью;
2. Реализация решений для настройки игр с математической моделью;
3. Реализация решений для тестирования игр с математической моделью.

1. ОБЗОР МЕТОДОВ ДЛЯ ТЕСТИРОВАНИЯ И НАСТРОЙКИ ИГР С МАТЕМАТИЧЕСКОЙ МОДЕЛЬЮ

1.1. Способы тестирования игр с математической моделью

Модульное тестирование, или юнит-тестирование (англ. unit testing) – процесс в программировании, позволяющий проверить на корректность отдельные модули исходного кода программы. Идея состоит в том, чтобы писать тесты для каждой нетривиальной функции или метода. Это позволяет достаточно быстро проверить, не привело ли очередное изменение кода к регрессии, то есть к появлению ошибок в уже оттестированных местах программы, а также облегчает обнаружение и устранение таких ошибок.

Цель модульного тестирования – изолировать отдельные части программы и показать, что по отдельности эти части работоспособны.

Этот тип тестирования обычно выполняется программистами.

Главным преимуществом такого подхода является проверка программного кода на корректность.

Недостатком такого подхода является то, что в играх с математической моделью существует множество зависимостей между блоками, покрытие которых тестами требует больших трудозатрат сотрудника.

1.2. Способы настройки игр с математической моделью

Настройки игры с математической моделью сводится к задаче нахождения числового решения для заданных правил, при котором статистика игры удовлетворяла бы требования.

Нахождение числовой реализации, удовлетворяющей требованиям, возможно следующими способами:

- Ручное изменение числовой реализации игры с последующим сбором статистики. Такой метод требует постоянного присутствия человека, который будет анализировать статистику и изменять числовую реализацию. Как показал эмпирический опыт, достижение нужной статистики иногда требует неочевидных изменений в числовой реализации из-за каскадных

зависимостей параметров. Такой режим работы уменьшает производительность труда работника, так как сбор статистики происходит за несколько минут, и за это время разработчик не может полностью переключиться на другую задачу. В итоге такой метод достаточно долог, ведет к уменьшению производительности труда у сотрудника, который совершает ручной перебор числовых реализаций;

- Нахождение математической модели игры, которая позволит подсчитать числено статистику игры. Такой метод, в зависимости от игры, выдвигает к разработчику дополнительные требования: как минимум это знание теории вероятности и математической статистики, для более сложных моделей потребуются знания числовых рядов (это являлось одной из причин создания комплекса). Такой метод крайне чувствителен к смене каркаса игры – при изменении каркаса математическая модель теряет актуальность. Также как и в прошлом методе, разработчик меняет числовую реализацию игры и анализирует вычисленную статистику. В конечном итоге такой подход требует неизменяемости правил, ставит дополнительные требования к сотруднику, который находит математическую модель игры;

- Также можно отметить метод ручного изменения числовой реализации по результатам тестирования или игрового опыта пользователей. Данный метод широко и успешно применятся в играх с небольшим количеством зависимостей между параметрами (к примеру, в играх с оружием это изменение его урона). Для некоторых игр возможен только такой метод настройки.

1.3. Новые решения для тестирования и настройки игр с математической моделью

Решение состоит из двух блоков: блок нахождения числовой реализации и блок регрессионного тестирования;

Блок нахождения числовой реализации представляет собой модуль, работающий над реализованным каркасом игры. Модуль посредством генетического алгоритма находит числовые реализации каркаса, удовлетворяющие требованиям. Модуль предоставляет функционал:

- генерация первичной числовой реализации в соответствии с требованиями;
- скрещивание числовых реализаций для получения новых;
- отсеивание реализаций, статистика которых не удовлетворяет требованиям.

Преимуществами такого подхода являются: автономность (не требуется постоянного присутствия разработчика), независимость от правил игры. Недостатками являются: дополнительные требования к организации статистики, возможность установить недостижимые требования для данной математической модели. Также следует отметить долгий поиск числовой реализации, однако комплекс может работать по ночам, что уменьшает значение данного фактора

Блок тестирования работает над статистикой игры. Модуль предоставляет функционал:

- сбора и сохранения в ПЗУ статистики игры.
- чтения из ПЗУ ранее собранной статистики игры.
- сравнение собранной и ранее сохраненной статистик.

Так как в рассматриваемых играх присутствует влияние ГПСЧ, то можно воспользоваться возможностью определять последовательности случайных чисел при помощи семени ГПСЧ. Преимуществами такого подхода являются: покрытие большого количества ситуаций, минимальные

трудозатраты сотрудника. Недостатком является отсутствие проверки правильности модулей игры. Также следует отметить достаточно большой объем времени, который необходим для выполнения теста, так как для получения среза статистики ее требуется собрать, однако тесты применяются единожды при сборке проекта и влияют не на производительность проекта, а лишь на процесс сборки, которых не является частым.

1.4. Необходимый функционал для работы нового решения

Для работы нового решения требуется унифицированный сбор статистик у разных игр. На момент постановки задачи существует стенд для сбора статистики и его наследники для каждой игры, переопределяющие поведение игрока в зависимости от игры. Необходимо инкапсулировать поведение при помощи шаблона проектирования «Стратегия»[1].

2. РЕАЛИЗАЦИЯ РЕШЕНИЯ ДЛЯ НАСТРОЙКИ ИГР С МАТЕМАТИЧЕСКОЙ МОДЕЛЬЮ

2.1. Реализация необходимого функционала для работы нового решения

На момент постановки задачи программное обеспечение для сбора статистики представляет собой два блока с высокой связанностью:

- стенд – внешний модуль, отвечающий за запуск сбора статистики с различными конфигурациями. В текущей реализации представлен классом StatisticStand и его наследником CgStand;
- игрок – внутренний модуль, через который происходит создание игры и изменение ее поведения и в котором задается поведение при различных внутриигровых ситуациях (Классы AbstractStandPlayer и StandPlayer после предварительного рефакторинга).

После исследования существующих стендов и «игроков» для сбора статистики (30 классов) было выявлено, что их существование необходимо для задания специфического поведения игры с последующим сбором дополнительной статистики, в то время как сбор обычной статистики может быть выделен в общую логику в общем «игроке».

Все игры, стенды и «игроки» которых требовали изменений, уже имплементировали нужные интерфейсы, поэтому было решено реализовать класс, который бы в зависимости от возможных действий, запрошенных через интерфейс «String get Action()» игры, вызывал бы методы работы с игрой [2].

В результате был создан класс PlayStrategy (программная реализация представлена в Приложении А), которому блок унифицированного «игрока» делегирует поведение при различных игровых ситуациях и который содержит:

- объекты: игры, параметров игры, статистики и словаря (HashMap), содержащего сущности из названия действия и объекта класса Action (программная реализация представлена в Приложении Б);
- методы, часть которых помещается в объекты класса Action, для работы с объектом игры. Методы оперируют состояниями игры, которые посылались бы игроку.

Класс Action содержит:

- условие, при котором действие целесообразно осуществить игроку;
- приоритет, при помощи которого определяется очередность целесообразных действий;
- ссылку на метод из класса PlayStrategy для работы с игрой.

Процесс выбора действия представлен на рисунке 2.1. Конечная архитектура представлена на рисунке 2.2.

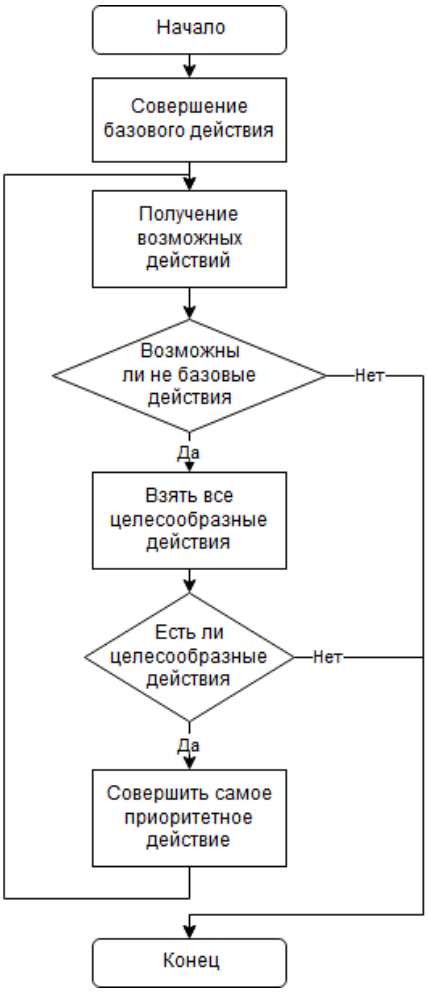

Рисунок 2.1 – Процесс выбора действия

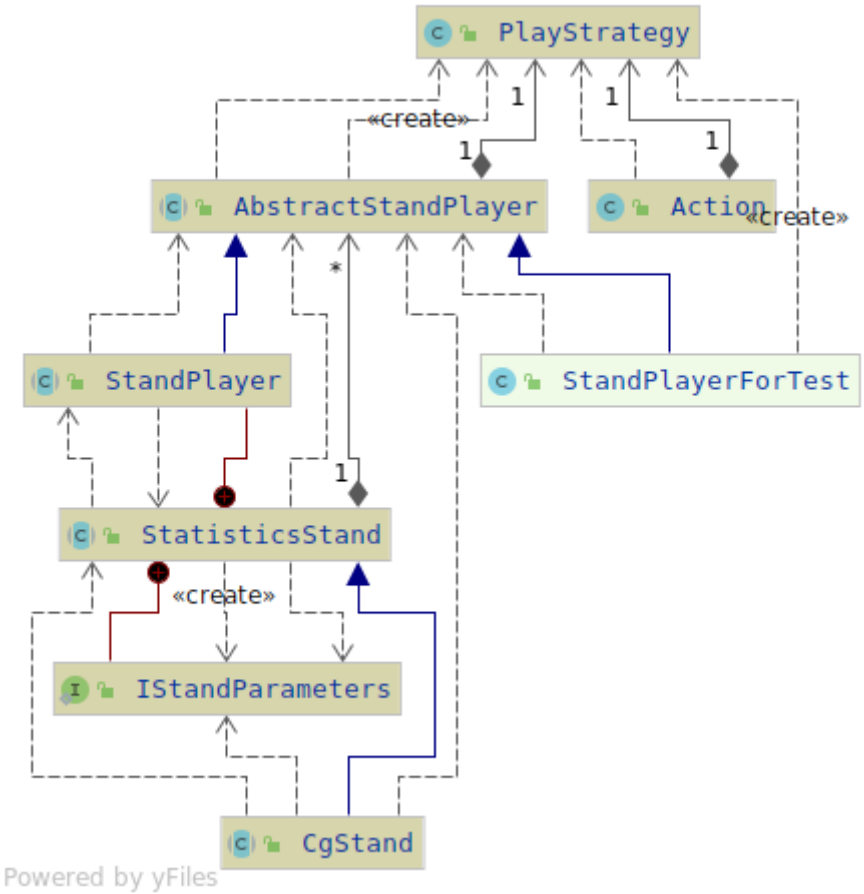

Рисунок 2.2 – Итоговая архитектура стендов

2.2. Реализация решения для настройки игр с математической моделью

Генетические алгоритмы – адаптивные методы поиска, которые используются для решения задач функциональной оптимизации. Они основаны на механизмах и моделях эволюции и генетических процессов биологических алгоритмов [3].

Хромосома – одно решение задачи[3].

Популяция – набор решений задачи. В начале алгоритма случайным образом генерируется набор решений (начальная популяция). Эти решения будут становиться лучше (эволюционировать) в процессе работы алгоритма до тех пор, пока не удовлетворят условиям задачи [3].

В общем случае популяция формируется случайным образом, но в реализации была предусмотрена возможность включения в набор «константной» хромосомы.

Функция пригодности (фитнесс-функция) – функция определяющая качество особей популяции [3]. В моем случае это будет имитация игры на сгенерированном числовом решении с последующим вычислением длины многомерного вектора, составленного из отклонения экспериментально полученных значений от необходимых значений статистики.

Генетические операторы – определённые правила, по которым изменяются особи в следующей популяции [3]. Среди них выделяют операторы скрещивания и мутации.

Селекция – это выбор тех особей, которые будут участвовать в создании потомков для следующей популяции, т.е. для очередного поколения. Такой выбор производится согласно принципу естественного отбора, по которому наибольшие шансы на участие в создании новых особей имеют особи с наибольшими значениями функции приспособленности [3].

Колесо рулетки – метод селекции, в котором родительские особи выбираются пропорционально значениям их функций приспособленности: каждой хромосоме сопоставлен сектор колеса рулетки, величина которого устанавливается пропорциональной значению функции приспособленности данной хромосомы. Таким образом, чем больше значение функции приспособленности, тем больше сектор на колесе рулетки и соответственно больше вероятность стать родительской особью (рис 2.3) [4].Программная реализация колеса рулетки представлена на рисунке 2.4

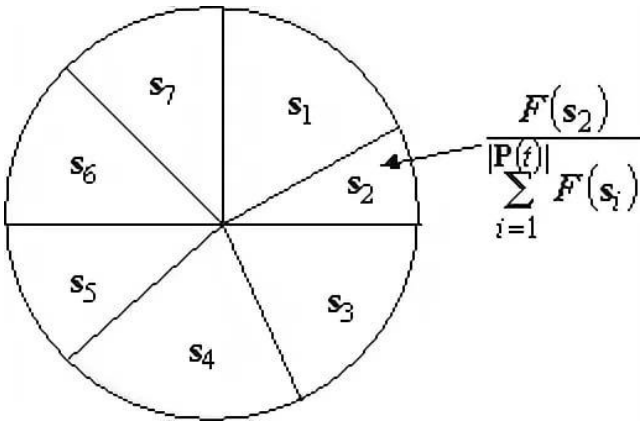

Рисунок 2.3 – Пример колеса рулетки.

```
static <STATS extends IGeneratorStats> List<Individual<STATS>> rouletteSelection(
        List<Individual<STATS>> individuals) {
    double[] fitnesses = individuals.stream()
            .mapToDouble(chromosome -> 1 - chromosome.fitness)
            .toArray();
    double sum = ArrayUtils.sum(fitnesses);

    for (int child = 0; child < fitnesses.length; child++) {
        fitnesses[child] = fitnesses[child] / sum;
    }
    List<Individual<STATS>> individualParentList = new ArrayList<>();
    individualParentList.add(individuals.get(selectParent(fitnesses)));
    individualParentList.add(individuals.get(selectParent(fitnesses)));
    return individualParentList;
}
```

Рисунок 2.4 – Листинг колеса рулетки

Турнирная селекция – метод, в котором все особи популяции разбиваются на подгруппы с последующим выбором в каждой из них особи с наилучшей приспособленностью[5]. Турнирная селекция продемонстрирована на рисунке 2.5, листинг ее программной реализации – на рисунке 2.6.

Рисунок 2.5 – Турнирная селекция

```
static <STATS extends IGeneratorStats> List<Individual<STATS>> tournamentSelection(
        List<Individual<STATS>> population,
        int nOfGroups) {
    int groupSize = population.size() / nOfGroups;
    Collections.shuffle(population);
    ArrayList<Individual<STATS>> individualsList = new ArrayList<>();
    for (int i = 0; i <= population.size() - groupSize; i += groupSize) {
        individualsList.add(population.subList(i, i + groupSize).stream()
                .min(Comparator.naturalOrder())
                .get());
    }
    return individualsList;
}
```

Рисунок 2.6 – Листинг турнирной селекции

Для выбора тех особей, которые будут участвовать в создании потомков для следующей популяции, было решено использовать колесо рулетки.

Для скрещивания особей было решено использовать многоточечное скрещивание. Механизм такого скрещивания представлен на рисунке 2.7. Самые лучшие результаты проявлялись при выборе количества точек для скрещивания случайным образом от 1 до 3.

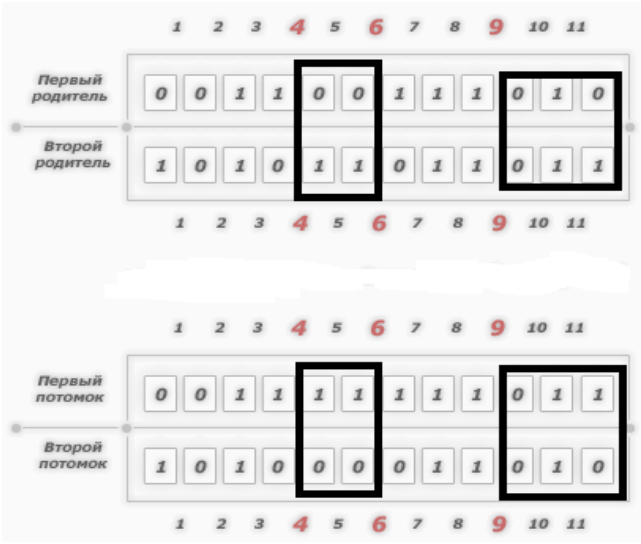

Рисунок 2.7 – Демонстрация скрещивания [4]

Для мутации особей была реализована точечная мутация, которая осуществляется в пределах одного гена, при этом потомок в общем случае содержит генотип родителя со слегка искажённой информацией. На рисунке 2.8 показан принцип мутации. Вероятность мутации индивидуальна для каждой игры.

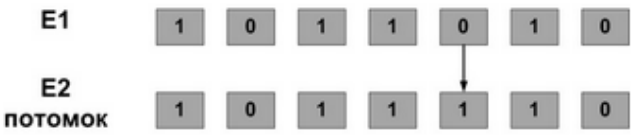

Рисунок 2.8 – Демонстрация мутации[4]

Для прореживания популяции было решено использовать турнирную селекцию.

После скрещивания и мутации происходит верификация потомка, на соответствие установленным ограничениям.

Совокупный алгоритм, состоящий из выше рассмотренных функций, представлен на рисунке 2.9 в виде схемы, а его программная реализация представлена в Приложении В.

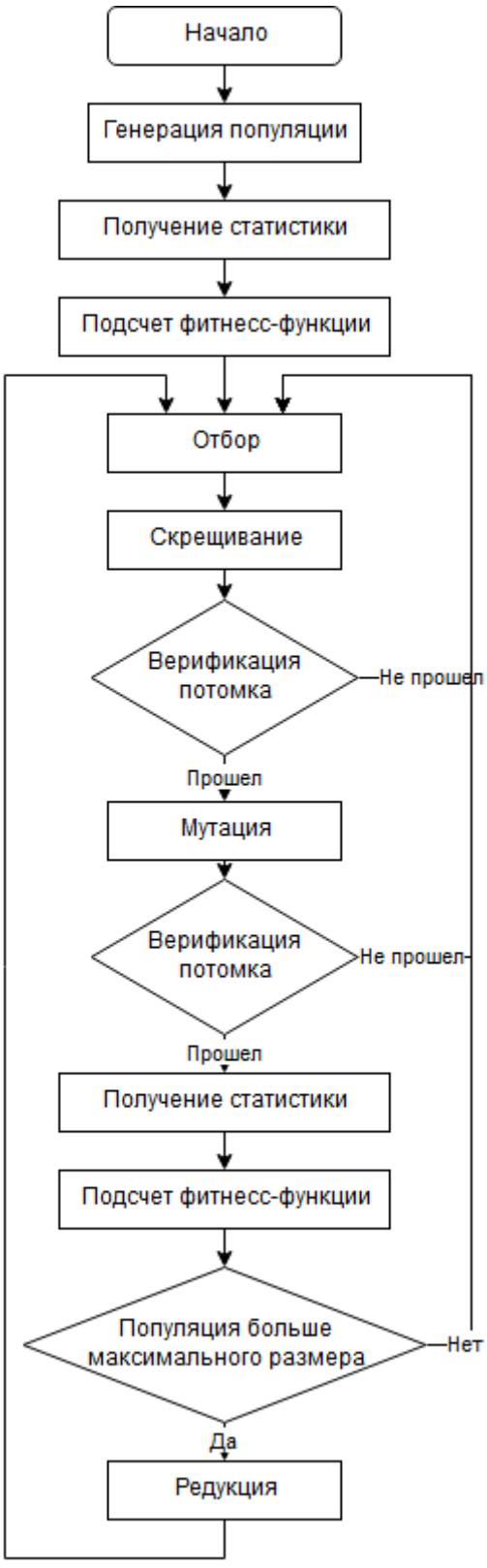

Рисунок 2.9 – Схема генетического алгоритма

Процесс нахождения статистики через имитацию игры занимает подавляющий объем в суммарном времени выполнения цикла, поэтому было решено использовать элементы многопоточного программирования [6]. Реализация алгоритма для параллельного выполнения представлена на рисунке 2.10. Реализованная архитектура для распараллеливания нахождения статистик позволяет построить распределенную систему на несколько компьютеров, связанных через интернет.

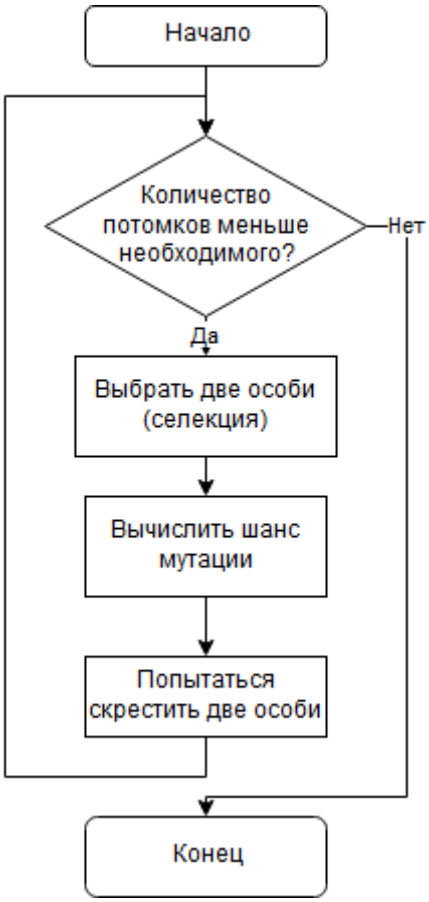

Рисунок 2.10 – Реализация многопоточности

Другим решением по оптимизации алгоритма является проверка показателей статистики, в процессе имитации игры, на попадание в необходимый интервал. Интервал формируется из необходимого значения и лучшего найденного значения. Интервал зависит от дисперсии величины, которая уменьшается с увеличением числа итераций. Для вычисления интервала необходимы коэффициенты распределения показателя статистики. Коэффициенты распределения показателя находятся следующим образом:

1. Нахождение показателя статистики при большом количестве итераций имитации игрового процесса.

2. Нахождение распределения показателя статистики при меньшем количестве итераций.

3. Построение столбчатой диаграммы для нахождения промежутка, в который может попасть значения при данном количестве итераций. На рисунке 2.11 представлено распределение параметра при 1000 измерений по 10000 итераций, при 50 млн. итераций параметр равен 0,96.

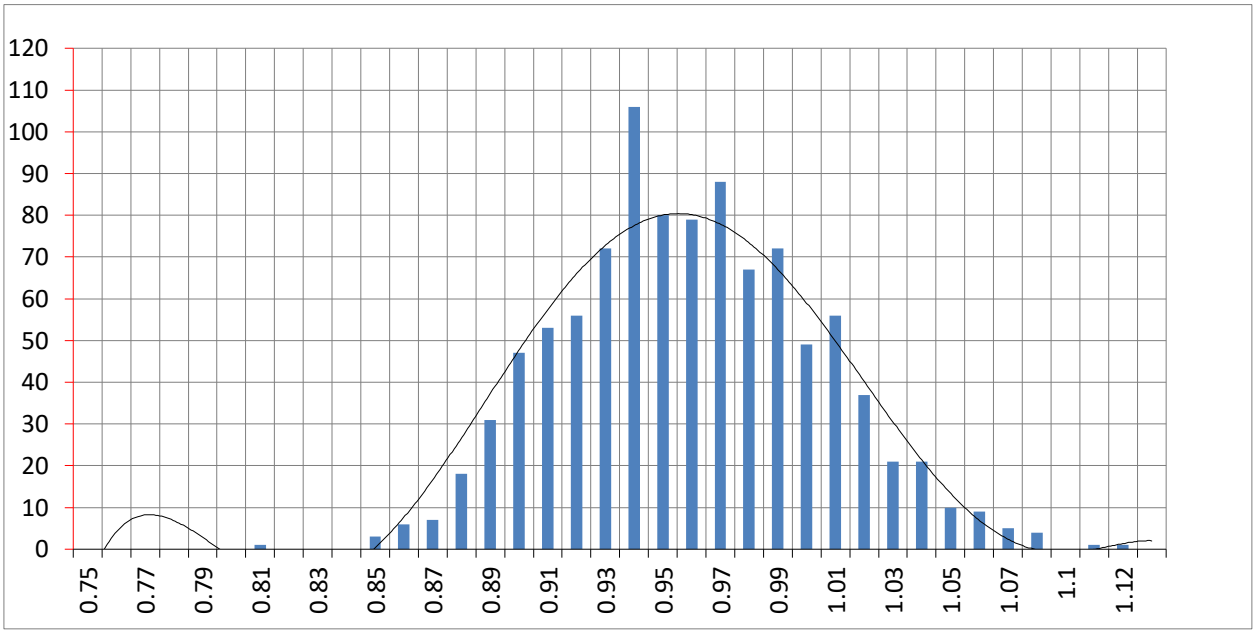

Рисунок 2.11 – Диаграмма распределения

4. Нахождение коэффициентов отклонения крайних точек от показателя, найденного в пункте 1.

Найденные коэффициенты позволяют сказать, попадает ли новое значение в промежуток между необходимым значением и лучшим значением при выбранном количестве итераций игры. Пример на рисунке 2.12.

- необходимая величина равна 0,96 (красный штрих);
- лучшее значение равно 1,02 (зеленый штрих) при n количестве итераций, и при данном количестве итераций истинное значение не может быть больше 1,03;

• новое значение равно 1,04 (оранжевый маркер) при k<n итераций, и существует вероятность того, что реальное значение находится в диапазоне 1,01 – 1,07 (выделен голубым цветом).

В этом случае существует вероятность того, что новое значение ближе к необходимому значению, чем ранее найденное, соответственно, следует продолжать имитацию игры. В противном случае происходит генерация ошибки, которая отменяет сбор статистики и запускает процесс скрещивания (рисунок 2.10) заново.

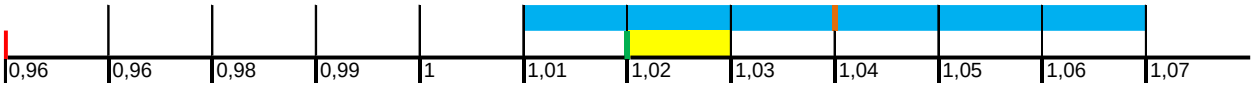

Рисунок 2.12 – Пример интервалов

Для мониторинга процесса была написана система логирования.

Каждую итерацию (итерация состоит из n количества скрещиваний) система выводит информацию:

• номер итерации;

• лучшее значение фитнес-функции;

• время в миллисекундах, затраченное на итерацию;

• время в секундах с момента запуска алгоритма.

В зависимости от настройки система может выводить:

• информацию о популяции: количество особей, значения фитнесс-функций особей, часть статистики особей;

• статистику новой лучшей числовой реализации;

• новую лучшую числовую реализацию.

Примеры работы алгоритма представлены на рисунках 2.13 – 2.15

```
successful crossing:  parent #176 0.3881;  parent #25 0.3463; tries=1;
     child #236 fitness=0.3057, mutations: 20;
     child #238 fitness=0.3503, mutations: 20;
```

Рисунок 2.13 – Пример скрещивания

```
INDIVIDUAL, population size=66
index:      114     140     257     274     276     278     279     283     288     297     302     312     313
fitness:    0.268   0.216   0.268   0.263   0.196   0.267   0.117   0.06    0.259   0.169   0.265   0.16    0.225
```

Рисунок 2.14 – Информация о популяции

```
Iteration=9    The best fitness=0.04877079435871641    Time spent iterating=(msec)47164    Total time spent(sec)=526
```

Рисунок 2.15 – Информация об итерации

Конечная архитектура представлена на рисунке 2.16. Класс Individual, содержит в себе набор хромосом (класс Chromosome), статистику и значение функции пригодности. Каждая хромосома определяет некоторый атомарный параметр игры. Процесс скрещивания двух объектов класса Individual представлен на рисунке 2.17. Буква Р и число – номер родительской особи, буква Х и число – номер хромосомы, буква П и число номер особи потомка (пример: Р2Х2 – родительская особь №2, вторая хромосома). Программная реализация скрещивания представлена в приложениях Г и Д.

Для подсчета функции пригодности и взаимодействия со статистикой игры, которая имплементирует интерфейс IGeneratorStats, используется класс StatComparator. Генетический алгоритм (рисунок 2.9) реализован в классе Generator. Результат выполнения представлен на рисунке 2.18.

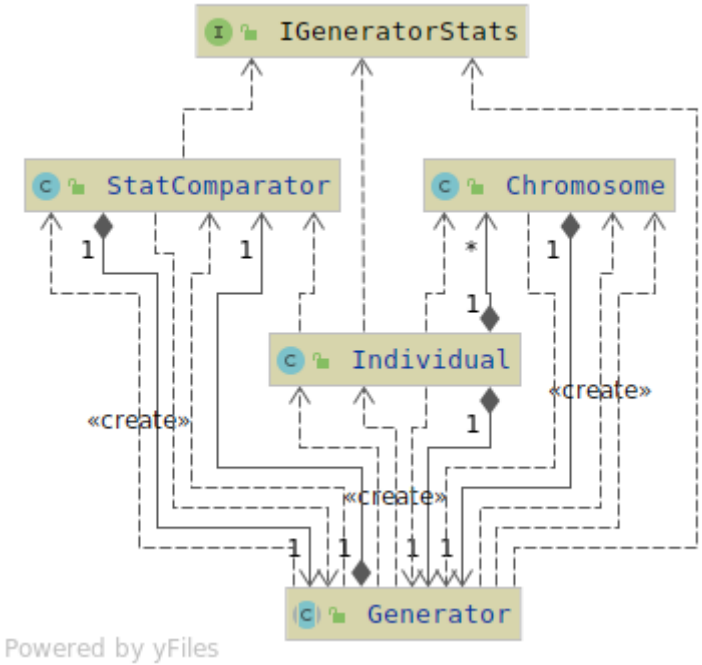

Рисунок 2.16 – Архитектура генетического алгоритма

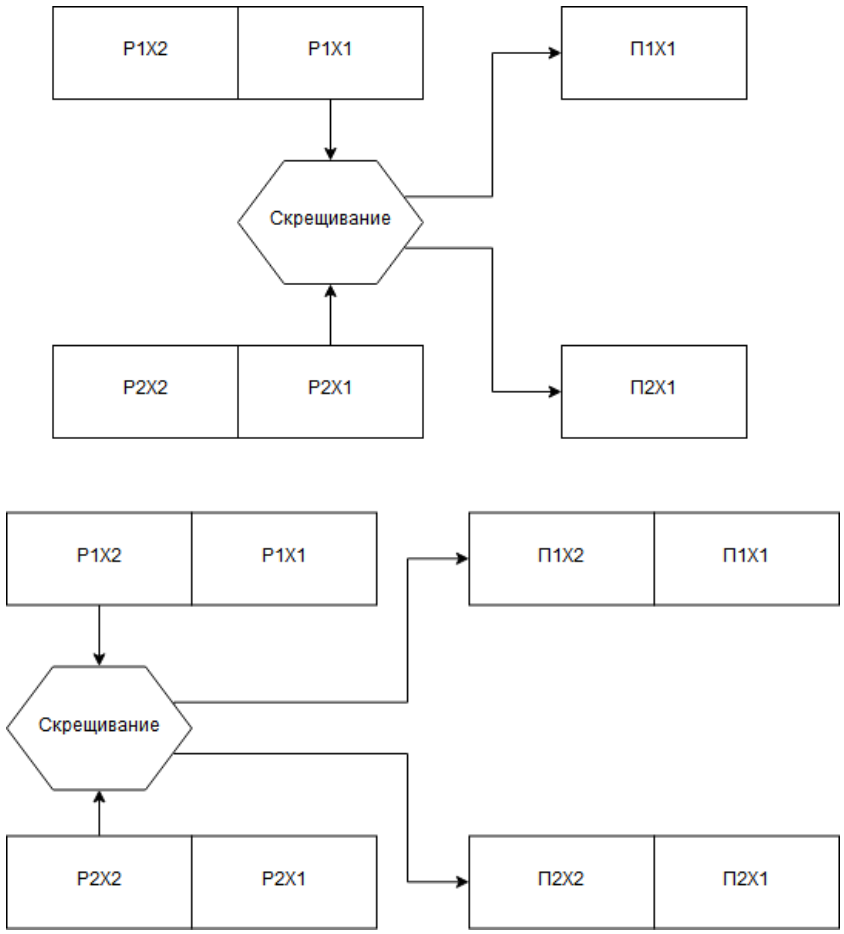

Рисунок 2.17 – Процесс скрещивания

```
INDIVIDUAL SOLUTIONS
{4, 3, 7, 1, 2, },
{0, 0, 3, 3, 5, 1, 4, },
{4, 2, 5, 0, 0, 0, 8, 0, 7, 0, 0, },
{1, 7, 7, 1, 1, 4, 4, 4, 5, 7, 8, 2, },
{3, 3, 7, 4, 3, 4, 3, 5, 4, 9, 6, 7, 2, 6, },
{4, 3, 2, 2, 1, 7, 7, 1, 1, },
{0, 1, 0, 3, 5, 6, 4, 2, 4, 6, 1, 1, },
{7, 0, 0, 6, 6, 5, 8, 6, },
{1, 9, 7, 1, 7, 4, 4, 4, 5, 7, },
{3, 7, 6, 4, 3, 4, 3, 5, 4, },
{4, 3, 7, 1, 4, 1, 7, 3, },
{0, 0, 3, 3, 5, 9, 4, 2, 4, 6, },
{1, 5, 4, 0, 5, 4, 0, 0, 7, },
{1, 9, 3, 1, 2, 7, 4, },
{3, 3, 1, 4, 3, 4, 3, 5, 4, 9, },
************************************************
```

Рисунок 2.18 – Вывод одного найденного решения

3. РЕАЛИЗАЦИЯ РЕШЕНИЯ ДЛЯ ТЕСТИРОВАНИЯ ИГР С МАТЕМАТИЧЕСКОЙ МОДЕЛЬЮ

3.1. Реализация решения.

Решение должно предоставлять функционал:

- сбора и сохранения в ПЗУ статистики игры;
- чтения из ПЗУ ранее собранной статистики игры;
- сравнение собранной и ранее сохраненной статистик.

Сравнение собранной и ранее сохраненной статистик представленных в текстовом (String) формате можно произвести средствами фреймворка для тестирования JUnit, который в дополнительном окне выделит расхождения.

Чтобы собрать статистику, для каждой игры необходимо создать объект унифицированного «игрока», разработка которого описана в пункте 1.5.1.Для этих целей был создан класс StandPlayerFabric[1], который также содержит в себе параметры для создания «игрока»:

- массив с количеством итераций игры для сбора срезов статистики;
- название игры; семя ГПСЧ;
- ряд служебных объектов для конфигурации игры.

Для каждой игры создается статический экземпляр StandPlayerFabric. Процесс сбора и сохранения статистики представлен на рисунке 3.1, более подробный процесс сбора срезов статистики представлен на рисунке 3.2.Листинг представлен на рисунке 3.3

Рисунок 3.1 – Процесс сбора и сохранения срезов статистики.

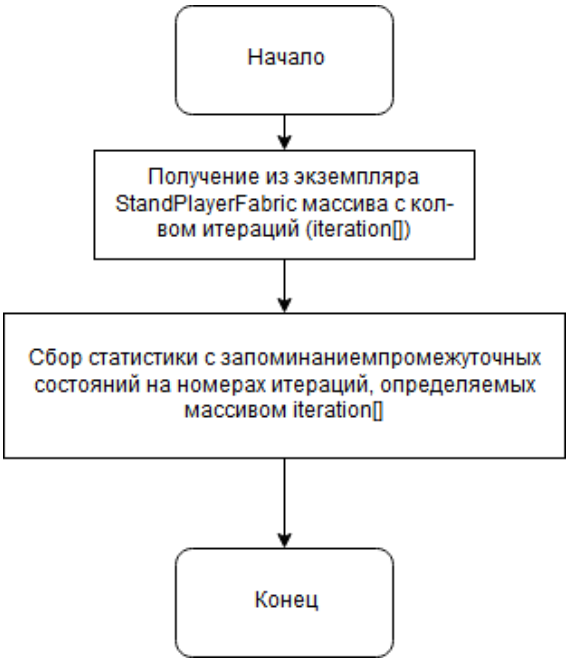

Рисунок 3.2 – Процесс сбора срезов статистики

```
private static void saveStats(StandPlayerFactory standPlayerFactory)
        throws IOException, CommonException {
    String gameName = standPlayerFactory.getGameName();
    int lines = standPlayerFactory.getLines();
    String fileName = formFileName(gameName, lines);
    List<String> stats = new ArrayList<>();
    long[] statCuts = standPlayerFactory.getEssays();
    AbstractStandRun player = standPlayerFactory.createStandRun();
    stats.add(collectStatsForTest(standPlayerFactory.getSeed(), player));
    for (int i = 1; i < statCuts.length; i++) {
        stats.add(continueCollectStats(player, statCuts[i] - statCuts[i - 1]));
    }
    File file = new File(
            "test-resources/com/intrice/games/" +
                    standPlayerFactory.getGamePackage() + "/" +
                    fileName + ".stats");
    file.getParentFile().mkdirs();
    file.createNewFile();
    try (FileOutputStream outputStream = new FileOutputStream(file);
         ZipOutputStream zos = new ZipOutputStream(outputStream);
         ByteArrayOutputStream bos = new ByteArrayOutputStream();
         ObjectOutputStream ous = new ObjectOutputStream(bos)) {
        ZipEntry entry = new ZipEntry(fileName + ".stats");
        zos.putNextEntry(entry);
        ous.writeObject(stats);
        ous.flush();
        zos.write(bos.toByteArray());
        zos.closeEntry();
    }
```

Рисунок 3.3 – Листинг сбора и сохранения в ПЗУ статистики.

Организация сохранения срезов статистики представлена следующим образом: строковое представление срезов статистики помещается в список, который впоследствии сериализуется с помощью класса ObjectOutputStream и сжимается с помощью класса ZipOutputStream и алгоритма сжатия deflate [2]. Сжатие позволяет уменьшить размер файла в 10 раз.

Для написания тестов использовался фреймворк JUnit. В теле теста игры происходит вызов метода для получения нужных срезов статистики из ПЗУ и вызов метода, который в процессе сбора статистики сравнивает средствами JUnitэталонный срез и новый срез. Таким образом, в случае несовпадения срезов происходит выброс исключения, и следующие срезы не собираются. Листинг чтения эталонной статистики из ПЗУ представлен на рисунке 3.4. Листинг программной реализации сравнения статистика представлен на рисунке 3.5. Схема прохождения теста представлена на

рисунке 3.7,. Реализованная архитектура позволяет достаточно коротко объявлять и определять тесты для игр (рис.3.8). Успешное прохождение тридцати реализованных тестов представлена на рисунке 3.6.

```
protected List<String> readStats(String fileName) throws ClassNotFoundException, IOException {
    String path = fileName + ".stats";
    try (ZipInputStream zis = new ZipInputStream(getClass().getResourceAsStream(path))) {
        zis.getNextEntry();
        ByteArrayOutputStream baos = new ByteArrayOutputStream();
        int value = zis.read();
        while (value != -1) {
            baos.write(value);
            value = zis.read();
        }
        try (ByteArrayInputStream bais = new ByteArrayInputStream(baos.toByteArray());
             ObjectInputStream ois = new ObjectInputStream(bais)) {
            return Unchecked.readObject(ois);
        }
    }
}
```

Рисунок 3.4 – Листинг чтения эталонной статистики из ПЗУ

```
protected void runTest(List<String> referenceStats, StandPlayerFactory factory)
        throws CommonException {
    AbstractStandRun standRun = factory.createStandRun();
    String statString = StatCreator.collectStatsForTest(factory.getSeed(), standRun);
    Assert.assertEquals(referenceStats.get(0), statString);
    long[] essays = factory.getEssays();
    for (int i = 1; i < essays.length; i++) {
        statString = StatCreator.continueCollectStats(standRun, essays[i] - essays[i - 1]);
        Assert.assertEquals(String.valueOf(i), referenceStats.get(i), statString);
    }
}
```

Рисунок 3.5 – Листинг сравнения статистик

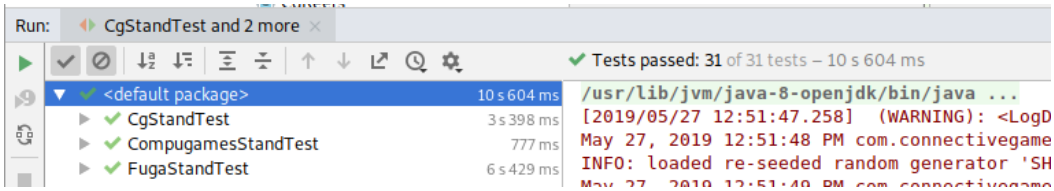

Рисунок 3.6 – Результат прохождения тестов

Рисунок 3.7 – Процесс прохождения теста

```
private static final StandPlayerFabric CORSAIRS_TREASURE_2_FABRIC_1 =
        new CgStandPlayerFabric<>(
                new long[]{1000, 2000, 10000},
                 lines: 1,
                 randomSeed: -1977279069,
                CORSAIRS_TREASURE_2,
                CorsairsTreasure2Game.class,
                CorsairsTreasure2Stats::new,
                 percent: 0.96);

public void testCorsairsTreasure2MinLines() throws IOException, CommonException, ClassNotFoundException {
    int lines = 1;
    String fileName = CORSAIRS_TREASURE_2 + "_" + lines;
    List<String> referenceStats = readStats(fileName);
    runTest(referenceStats, CORSAIRS_TREASURE_2_FABRIC_1);
}
```

Рисунок 3.8 – Пример листинга теста игры

3.2. Реализация вспомогательных решений для нахождения минимального количества для итераций сбора статистики.

Для нахождения минимального количества итераций, необходим критерий для оценки достаточности собранной статистики. Критерием оценки было решено сделать количество нулевых значений в объекте статистики – чем меньше нулевых значений, тем лучше.

Было решено реализовать метод для конвертации объекта статистики в массив типа long. Для реализации метода были изучены классы статистики и установлено, что типами полей являются: массивы объектов, массивы примитивов, коллекции (списки и словари), примитивы, обертки примитивов.

Первым шагом разработки алгоритма являлась реализация метода с использованием Java Reflection API для получения всех полей статистики. Для получения полей возможного суперкласса в метод использовался рекурсивный вызов.

Вторым шагом стало построение разветвленного алгоритма действий в зависимости от типа поля:

- примитивы помещаются в массив long;
- из словаря берутся все значения, к каждому из которых применяется данный алгоритм;
- к каждому элементу списков или массивов объектов применяется данный алгоритм;
- примитивные массивы и массивы оберток примитивов помещаются в массив long.

После преобразования объекта статистики в массив типа long можно оценить полноту статистики: чем меньше нулей, тем более полная статистика.

Метод для нахождения минимального количества итераций реализован в классе StatCreator. Он требует следующих аргументов для запуска:

- объект типа StandPlayerFabric для динамического создания модуля сбора статистики;
- шаг – количество итераций, через которое будет анализироваться полнота статистики;
- максимальное количество итераций;
- зерно ГПСЧ, при котором будет первый запуск;
- количество попыток – сколько зерен будет перебрано;
- минимальное количество нулей – необходимо для передачи информации о прошлом зерне при рекурсивном вызове.

Схема алгоритма предоставлена на рисунке 3.9, программная реализация представлена в Приложении Е.

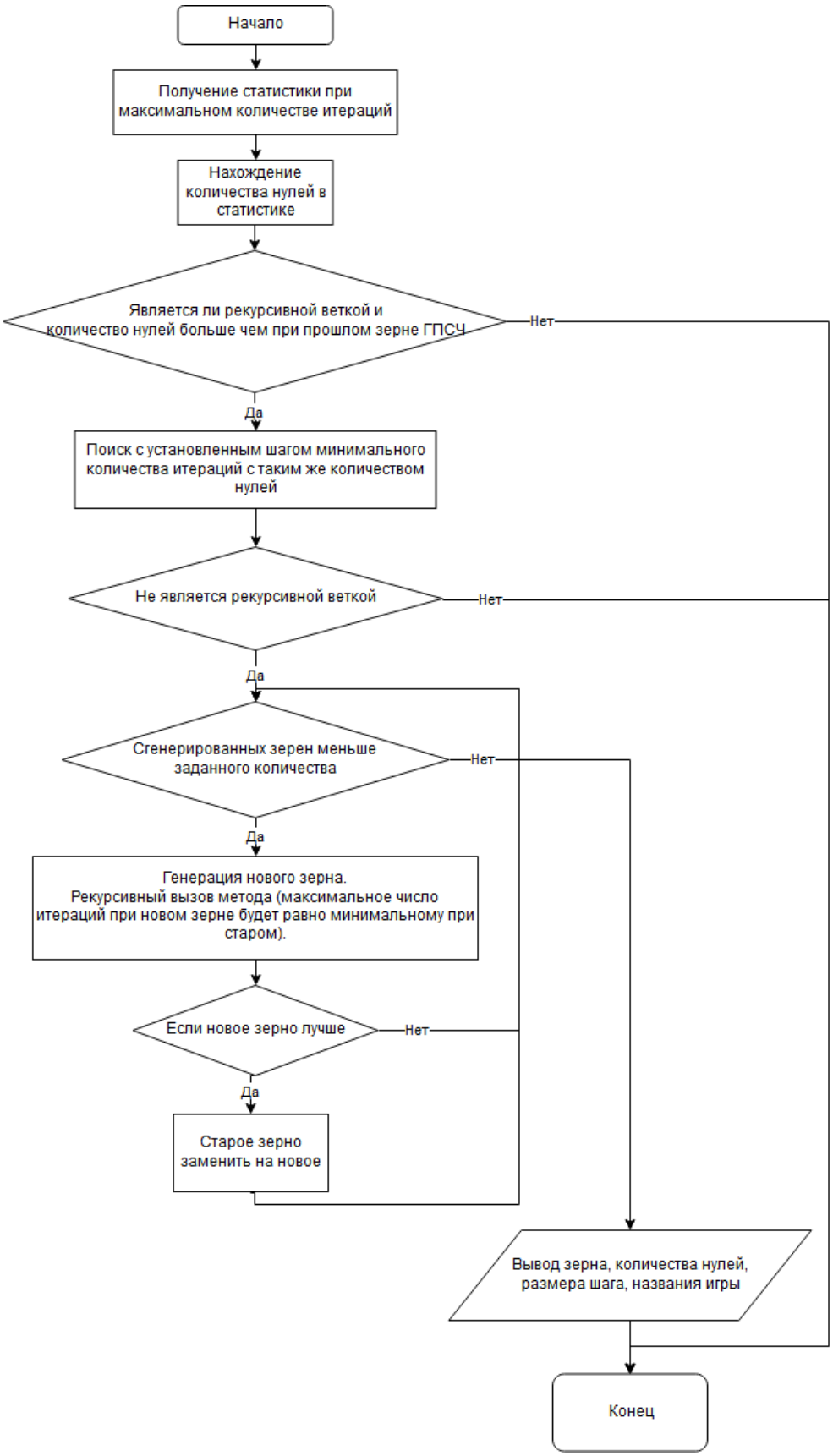

Рисунок 3.9 – Алгоритм поиска минимального количества итераций

На рисунке 3.10 представлена архитектура модуля для регрессионного тестирования.

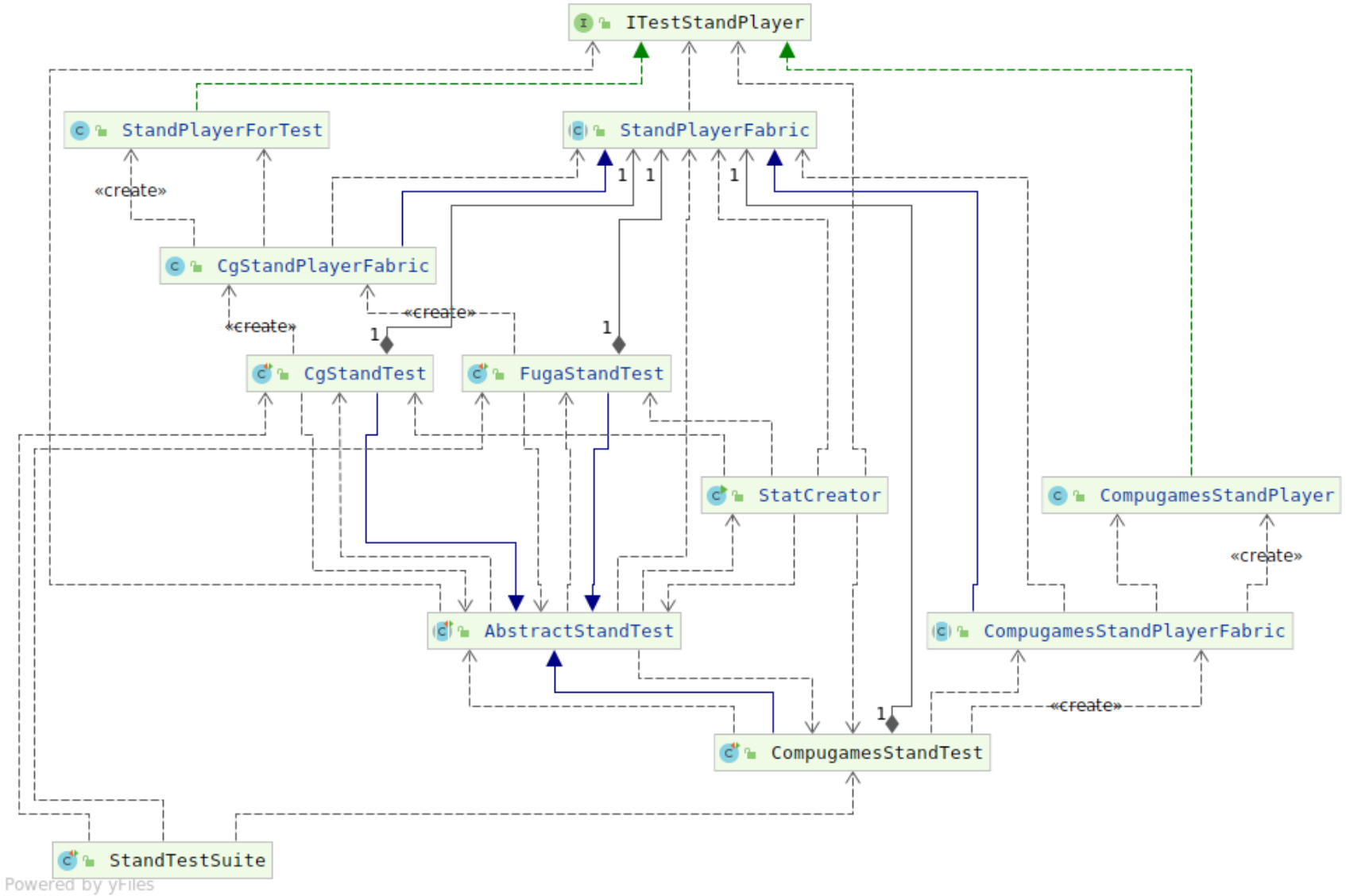

Рисунок 3.10 – Архитектура модуля для регрессионного тестирования

ЗАДАНИЕ ДЛЯ РАЗДЕЛА
«КОНЦЕПЦИЯ СТАРТАП-ПРОЕКТА»

Студенту:

Группа	ФИО
8В5Б	**Ивченко Александру Юрьевичу**

Школа	**инженерного предпринимательства**	**Направление**	09.03.01 «Информатика и вычислительная техника»
Уровень образования	Бакалавриат		

Перечень вопросов, подлежащих разработке:	
Проблема конечного потребителя, которую решает продукт, который создается в результате выполнения НИОКР (функциональное назначение, основные потребительские качества)	Недостаточная оптимизация процесса разработки игры.
Способы защиты интеллектуальной собственности	Алгоритмы, входящие в состав ПО являются коммерческой тайной.
Объем и емкость рынка	При цене 270 тыс. руб. за ПО максимальная емкость и объем рынка составит около 22,5 млн. руб. без учета возможных контрактов на сопровождение продукта.
Современное состояние и перспективы отрасли, к которой принадлежит представленный в ВКР продукт	Согласно отчетам рынок производства игр является быстрорастущим.
Себестоимость продукта	Себестоимость составит порядка 230482 рублей за электронную копию ПО
Конкурентные преимущества создаваемого продукта	Меньшие затраты времени на настройку и тестирования игр с математической моделью.
Сравнение технико-экономических характеристик продукта с отечественными и мировыми аналогами	Прямых аналогов не было найдено. Существующие методы для настройки и тестирования игр с математической моделью требуют больших трудозатрат, также могут выдвигаться дополнительные профессиональные требования к разработчику
Целевые сегменты потребителей создаваемого продукта	Целевой рынок состоит из разработчиков онлайн слотов. Согласно открытым источникам это порядка 83 компаний.
Бизнес-модель проекта	Бизнес-модель проекта представлена в виде бизнес-модели Остервальдера
Производственный план	Поиск клиента будет происходить с помощью прямого контакта с потенциальными заказчиками. Подстройка продукта под клиента

	подразумевает как ответвление от основного ПО, так и изменение базового функционала.
План продаж	Планируется выйти на продажу двух копий в месяц и удерживать это значение до полного покрытия рынка.
Перечень графического материала:	
При необходимости представить эскизные графические материалы(например, бизнес-модель)	Бизнес-модель Остервальдера

Дата выдачи задания для раздела по линейному графику	

Задание выдал консультант по разделу «Концепция стартап-проекта» (со-руководитель ВКР:

Должность	**ФИО**	**Ученая степень, звание**	**Подпись**	**Дата**
Доцент ШИП	Пожарницкая О. В.	к.э.н., доцент		

Задание принял к исполнению студент:

Группа	**ФИО**	**Подпись**	**Дата**
8В5Б	Ивченко А.Ю.		

4. КОНЦЕПЦИЯ СТАРТАП-ПРОЕКТА

4.1. Введение в предметную область:

Компьютерная игра – компьютерная программа, служащая для организации игрового процесса, связи с партнёрами по игре, или сама выступающая в качестве партнёра. К компьютерным играм также относят видеоигры и мобильные игры. Существуют попытки выделить компьютерные игры как отдельную область искусства, наряду с театром, кино и т. п. По некоторым компьютерным играм проводятся любительские и профессиональные соревнования. Такого рода соревнования называются киберспортом.

Большинство компьютерных игр включают в себя одну или несколько математических моделей.

В данной работе рассматривается разработка игр, в которых ярко выражена математическая составляющая и присутствует сильное влияние генератора псевдослучайных чисел (ГПСЧ) на результаты. Как правило, в таких играх пользователь ограничен всего несколькими возможными действиями, а управление производится за счет компьютерной мышки (или её аналогов). Далее в работе такие игры будут называться игры с математической моделью.

Каркасом игры будем называть совокупность всех событий и их причин без привязки к числам. Каркас игры задается правилами. Примером части каркаса может послужить гравитация в игре.

Числовой реализацией (или числовым решением) назовем одну конкретную реализацию каркаса игры. Игру с хорошим числовым решением называют сбалансированной. Для примера с гравитацией это может быть ускорение свободного падения или максимальная безопасная высота падения. Большинство игроков назовут игру сбалансированной, если безопасная высота падения будет больше высоты прыжка персонажа.

Математической моделью игры является совокупность уравнений, описывающих каркас игры. Например, для описанного выше случая можно ввести уравнения зависимости скорости падения от времени и зависимости урона от высоты падения.

4.2. Проблема:

Одной из проблем создания игр с математической моделью является недостаточная оптимизация процесса разработки.

В ходе разработки игр с математической моделью перед разработчиком стоят следующие задачи:

1. Придумать каркас игры.
2. Придумать требования для числовой реализации.
3. Найти математическую модель для каркаса
4. Найти числовую реализацию игры, удовлетворяющую требованиям, при помощи математической модели.
5. Реализовать игру.
   5.1. Реализовать каркас
   5.2. Подставить значения в каркас из найденной числовой реализации
6. Реализовать тесты игры.

Основными проблемами среди данных задач являются:

- Нахождения математической модели игры является нетривиальной задачей даже для простых каркасов. Также сложность нахождения модели увеличивает стоимость даже небольшого изменения каркаса игры.
- Сложность нахождения числовой реализации игры, удовлетворяющей требованиям, связана с тем, что математическая модель игры представлена множеством уравнений и сама числовая реализация, как правило, состоит из множества значений.

Как правило, игра частично использует уже готовую логику, и необходимо покрывать тестами только уникальные части игры. В процессе активной разработки возникает потребность в рефакторинге программного

кода. Таким образом, возникает потребность в регрессионном тестировании, произвести которое средствами обычного юнит-тестирования весьма сложно.

4.3. Описание продукта:

Конечный продукт состоит из двух блоков: блок нахождения числовой реализации и блок регрессионного тестирования.

Блок нахождения числовой реализации представляет собой модуль, работающий над реализованным каркасом игры. Модуль посредством генетического алгоритма находит числовые реализации каркаса, удовлетворяющие требованиям. Модуль предоставляет функционал:

- генерация первичной числовой реализации в соответствии с требованиями.
- скрещивание числовых реализаций для получения новых
- отсеивание реализаций, статистика которых не удовлетворяет требованиям

Для работы модуля требуется блок для сбора статистики игры в соответствии с оптимальной стратегией игры.

Блок регрессионного тестирования представляет собой модуль, работающий над блоком для сбора статистики игры. Модуль предоставляет функционал:

- сбора и сохранения в ПЗУ статистики игры.
- чтения из ПЗУ ранее собранной статистики игры.
- сравнение собранной и ранее сохраненной статистик.

4.4. Потребительские качества:

Основными потребительскими качествами является:

- универсальность – ПО можно применять к различным типам игр с математической моделью.
- легкость использования – для применения ПО не требуется создавать объемную программную инфраструктуру или вносить существенные изменения в уже существующую архитектуру.

- скорость – использованные алгоритмы распараллелены, для увеличения скорости работы на мощных ЭВМ. Блоки поддерживают функционал прерывания работы в случае, если результаты на данном этапе работы не приемлемы.

4.5. Защита интеллектуальной собственности:

Объектом авторского права является ПО для разработки и тестирования игр с математической моделью. Алгоритмы, входящие в состав ПО являются коммерческой тайной. ПО, выполняющее аналогичные функции, среди запатентованных изобретений не было найдено.

4.6. Объем и емкость рынка

Целевой сегментированный рынок (SAM) состоит из разработчиков онлайн слотов. Согласно открытым источникам это порядка 83 компаний [7], которых может заинтересовать разработанное ПО.

Из полученного практического опыта следует что, разработанное ПО является:

- уникальным (подобные решения не использовались в 3 из 3 компаний)
- необходимым для всех компаний на рынке (подобные решения потребовалось в 1 из 1 компаний).

Так как аналогов разработки и конкурентов не было найдено, то можно считать SAM и SOM одинаковыми величинами.

Так как ПО является необходимым для всех компаний на рынке, то емкость и объем рынка – одинаковые величины.

При цене 270 тыс. руб. за ПО максимальная емкость и объем рынка составит около 22,5 млн. руб. без учета возможных контрактов на сопровождение продукта.

4.7. Анализ современного состояния и перспектив развития отрасли

Согласно отчету «GlobalSlotMachineMarket 2018-2022» от TechNavio рынок онлайн слотов вырастет на 15,58% за период 2018 – 2022 [8].

Глобальный онлайн-рынок азартных игр в настоящее время приносит доход около $37 млрд в год, а около 85 стран по всему миру решили узаконить азартные игры в Интернете, в соответствии с Американской Игровой Ассоциацией.

По данным Европейской комиссии, доход от азартных онлайн-игр составляет около $15 млрд в год, и он растет быстрее, чем в остальном мире. [9].

Крупные компании на рынке слотов нацелены на 50-100 новых игровых автоматов в год. От трети до половины из них имеют новую оригинальную математику или изменения ранее успешных математических моделей [10].

4.8. Планируемая стоимость продукта

Предварительная стоимость ПО, при условии продажи двух копий в месяц на протяжении трех лет, состоит из следующих компонентов:

Аренда офисного помещения – 15000 рублей / месяц.

Зарплата аналитика – 70000 рублей / месяц.

Зарплата двух программистов – 200000 рублей / месяц.

Зарплата тестировщика – 70000 рублей / месяц.

Рабочие места – 250000 рублей.

Затраты на реализацию – 400000 рублей

Дополнительные расходы – 30000 рублей / месяц.

Таким образом, себестоимость копии будет равна:

(48(15000+70000+200000+70000+30000)+250000+400000)/83= =230482*

Планируемая стоимость продукта составит 270 тыс. руб.

4.9. Конкурентные преимущества создаваемого продукта, сравнение технико-экономических характеристик с отечественными и мировыми аналогами.

Так как нет возможности найти информацию о возможных аналогичных программных комплексах, то будет произведено сравнение с известными из практического опыта решениями. Краткое сравнение представлено в таблицах 1 и 2.

Таблица 1. Сравнение методов нахождения числовой реализации

Нахождение числовой реализации, удовлетворяющей требованиям.		
Решение	Преимущества	Недостатки
Существующие решения		
Ручное изменение числовой реализации игры с последующим сбором статистики.	Не требуется предварительных мероприятий	Времязатратно, уменьшение производительности труда у сотрудника
Нахождение математической модели игры, позволяющую подсчитать числено статистику игры	Относительно быстро	Требуется неизменяемость правил, дополнительные требования к сотруднику, ручной перебор числовых реализаций, требуются предварительные мероприятия
Предлагаемое решение		
Эволюционное программное изменение числовой реализации игры с последующим сбором статистики.	Не требует постоянного присутствия разработчика	Некоторые требования к организации сбора статистики, возможность установить недостижимые требования для данной математической модели.

Таблица 2. Сравнение методов регрессивного тестирования

Регрессионное тестирование.		
Решение	Преимущества	Недостатки
Существующие решения		
Написание тест кейсов.	Эквивалентно модульному тестированию	Большие трудозатраты для покрытия большого количества ситуаций.
Предлагаемое решение		
Генерации тест кейсов в виде срезов статистики.	Минимальные временные затраты сотрудников, большее покрытие в сравнении с классической реализаций регрессионных тестов	Большее время выполнения тестов

Нахождение числовой реализации, удовлетворяющей требованиям, возможно следующими способами:

- Ручное изменение числовой реализации игры с последующим сбором статистики. Такой метод требует постоянного присутствия человека, который будет анализировать статистику и изменять числовую реализацию. Такой режим работы уменьшает производительность труда работника, так как сбор статистики происходит в фоновом режиме, как правило, за несколько минут, и за это время разработчик не может полностью переключиться на другую задачу. В итоге: долго, уменьшение производительности труда у сотрудника, ручной перебор числовых реализаций.

- Нахождение математической модели игры, которая позволит подсчитать числено статистику игры. Такой метод, в зависимости от игры, выдвигает к разработчику дополнительные требования: как минимум это знание теории вероятности и математической статистики, для более сложных моделей потребуются знания числовых рядов (это являлось одной из причин создания комплекса). Такой метод крайне чувствителен к смене каркаса игры – при изменении каркаса математическая модель теряет актуальность. Также, как и в методе с ручным изменением числовой реализации игры с последующим сбором статистики, разработчику придется менять числовую реализацию и смотреть на вычисленную статистику. В итоге: неизменяемость правил, дополнительные требования к сотруднику, ручной перебор числовых реализаций.

- Использование программного комплекса (описание блока, отвечающего за нахождение числовой реализации, представлено выше). В итоге: долго (однако комплекс может работать по ночам), некоторые требования к организации сбора статистики, возможность установить недостижимые требования для данной математической модели.

Регрессионное тестирование:

- Для регрессионного тестирования используются тест кейсы, написанные на ранних стадиях разработки и тестирования. Это дает

гарантию того, что изменения в новой версии приложения не повредили уже существующую функциональность. Рекомендуется проводить автоматизацию регрессионных тестов, для ускорения последующего процесса тестирования и обнаружения дефектов на ранних стадиях разработки программного обеспечения. В итоге: появление новых механик увеличивает количество тестов, большие трудозатраты сотрудника для покрытия большого количества ситуаций, вместе с регрессионным тестированием производится и модульное тестирование, что является преимуществом.

- Регрессионное тестирование, представленное в комплексе, предоставляет возможность генерации тест кейсов в виде срезов статистики с последующим их использованием в процедуре тестирования. В итоге: минимальные временные затраты сотрудников, большее покрытие в сравнении с классической реализаций регрессионных тестов, большее время выполнения тестов (для получения среза статистики ее требуется собрать, что занимает время), отсутствие проверки правильности модулей игры.

4.10. Описание целевых сегментов потребителей создаваемого продукта

Полный объем целевого рынка (TAM) состоит из множества разработчиков игр с математической моделью. В него входят как разработчики инди-проектов, так и крупные компании.

Целевой сегментированный рынок состоит из разработчиков онлайн слотов. Согласно открытым источникам это порядка 83 компаний [7], которых может заинтересовать разработанное ПО.

Так как аналогов разработки и конкурентов не было найдено, то можно считать SAM и SOM одинаковыми величинами.

4.11. Бизнес модель проекта. Производственный план. План продаж.

Бизнес модель проекта можно представить в виде бизнес модели Остервальдера (таблица 3):

Таблица 3. Бизнес модель Остервалльдера

<table>
<tr><td rowspan="2">Потенциально: маркетинговые агентства.</td><td>Предоставление инструментального ПО для разработки и тестирования игр с мат. моделью</td><td rowspan="2">Новое решение в сфере частичной автоматизации разработки игр с математической моделью.</td><td>Особая персональная поддержка</td><td rowspan="2">Разработчики онлайн слотов</td></tr>
<tr><td>Персонал, ПО, аппаратное обеспечение, интеллектуальная собственность</td><td>Электронные копии</td></tr>
<tr><td colspan="3">Амортизация, заработная плата, аренда помещений, командировки</td><td colspan="2">Разовая сделка с возможностью платной поддержки.</td></tr>
</table>

Производственный план для одного клиента представлен в таблице 4. Возможно параллельное выполнение работ для нескольких клиентов по мере роста штата персонала.

Таблица 4. Производственный план для одного клиента

Производимые работы	Календарные недели							
	1	2	3	4	5	6	7	8
Нахождение клиента								
Подстройка продукта под клиента								
Мероприятия по предоставлению продукта клиенту.								

Поиск клиента будет происходить с помощью прямого контакта с потенциальными заказчиками.

Подстройка продукта под клиента подразумевает как ответвление от основного ПО, так и изменение базового функционала.

Мероприятия по предоставлению продукта клиенту могут включать в себя: консультацию персонала заказчика, развертку программного продукта на машинах заказчика и т.д.

Производственный план на год представлен в таблице 5

Таблица 5. Производственный план на год

Клиент	Количество месяцев с начала деятельности											
	1	2	3	4	5	6	7	8	9	10	11	12
1												
2												
3												
4												
5												
6												
7												
8												
9												
10												
11												
12												

Производственный план на четыре года представлен на рисунке 4.1.

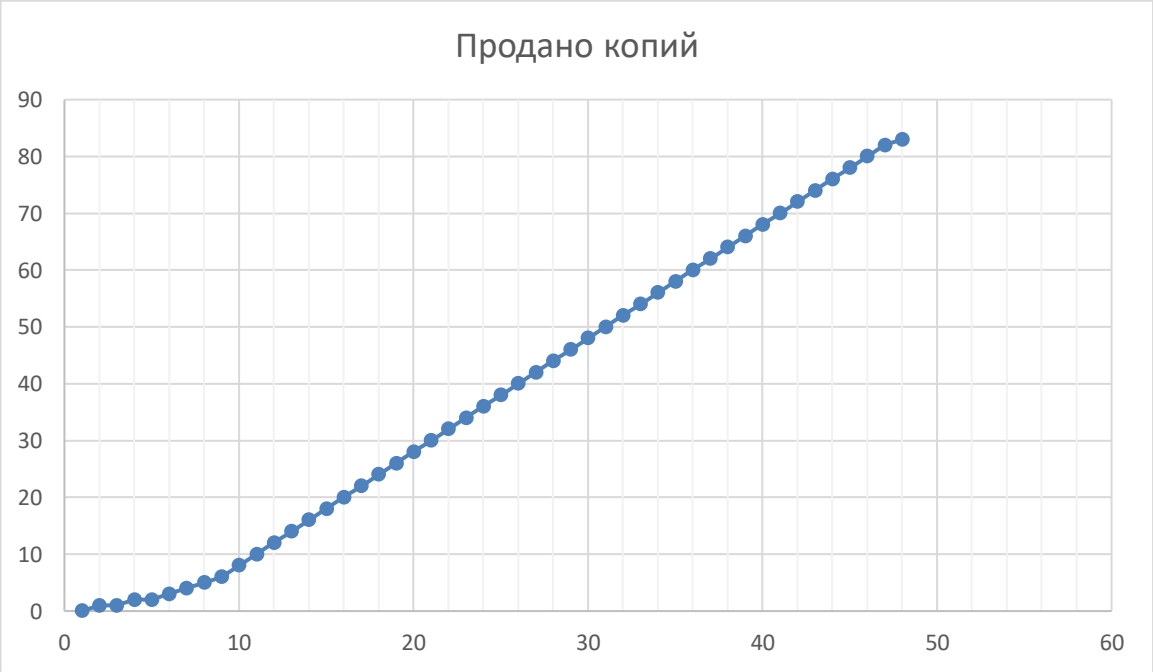

Рисунок 4.1. Производственный план.

Предполагаемый план продаж представлен в таблице 6. Планируется выйти на продажу двух копий в месяц и удерживать это значение до полного покрытия рынка.

Таблица 6. План продаж на год.

Месяц	Продажи
1	0
2	1
3	0
4	1
5	0
6	1
7	1
8	1
9	1
10	2
11	2
12	2

4.12. Стратегия продвижения продукта на рынок

Так как целевой сегментированный рынок состоит из 83 компаний, планируется использование стратегии навязывания потенциальному потребителю программного обеспечения за счет целенаправленного воздействия с использованием современных средств коммуникаций (электронная почта, онлайн конференции). Возможно делегирование обязанностей по продвижению маркетинговому агентству.

ЗАДАНИЕ ДЛЯ РАЗДЕЛА
«СОЦИАЛЬНАЯ ОТВЕТСТВЕННОСТЬ»

Группа	ФИО
8В5Б	Ивченко Александру Юрьевичу

Школа	ИШИТР	Отделение	ИТ
Уровень образования	Бакалавриат	**Направление/специальность**	09.03.01 «Информатика и вычислительная техника»

Тема ВКР:

Разработка окружения для тестирования и настройки игр с математической моделью.	
Исходные данные к разделу «Социальная ответственность»:	
1. Характеристика объекта исследования (вещество, материал, прибор, алгоритм, методика, рабочая зона) и области его применения	Разработка приложения на рабочем месте при помощи персонального компьютера, монитора, клавиатуры и компьютерной мыши. Область применения конечного продукта – разработка игр.
Перечень вопросов, подлежащих исследованию, проектированию и разработке:	
1.Производственная безопасность – Отклонение показателей микроклимата – Превышение уровня шума – Отсутствие или недостаток естественного света – Недостаточная освещенность рабочей зоны – Повышенное значение напряжения в электрической цепи, замыкание которой может произойти через тело человека	Анализ вредных факторов: − Параметры микроклимата (СанПиН 2.2.2/2.4.1340-03). − Освещенность рабочего места (СанПиН 2.2.1/2.1.1.1278-03). − Уровень шума (СанПиН 2.2.4.3359-16). − Влияние электромагнитного поля (СанПиН 2.2.2/2.4.1340–03) Анализ опасных факторов: − Опасность поражения электрическим током (ГОСТ Р 50571.17-2000). − Поражение статическим электричеством (СанПиН 2.2.2/2.4.1340-03). − Короткие замыкания (ГОСТ Р 50571.5-94).
2. Экологическая безопасность:	Прямого влияния на экологическую обстановку не оказывается
3. Безопасность в чрезвычайных ситуациях:	Чрезвычайная ситуация техногенного характера для рассматриваемого случая – пожар.
4. Правовые и организационные вопросы обеспечения безопасности: − Специальные правовые нормы трудового законодательства; − Организационные мероприятия при компоновке рабочей зоны.	− Организация рабочего места (ГОСТ 12.2.032-78). − Общие требования безопасности к рабочим местам (ГОСТ 12.2.061-81).

Дата выдачи задания для раздела по линейному графику	

Задание выдал консультант:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
ассистент ООТД	Мезенцева И.Л.	ассистент		

Задание принял к исполнению студент:

Группа	ФИО	Подпись	Дата
8В5Б	Ивченко Александр Юрьевич		

5. СОЦИАЛЬНАЯ ОТВЕТСВЕННОСТЬ

В любой научно-исследовательской и проектной деятельности немаловажную роль занимает такая область как безопасность труда и окружающей среды.

В понятие «социальная ответственность» входит следующее: состояние рабочего места, помещения, режим трудовой деятельности и обеспечение мероприятий по защите трудящихся в моменты чрезвычайных ситуаций регламентируются в соответствии с международным стандартом ICCSR26000:2011 «Социальная ответственность организации» [11]. Целью данного стандарта является принятие проектных решений, исключающих несчастные случаи на производстве и снижение негативных воздействий на окружающую среду.

Согласно данному стандарту такое понятие, как «социальная ответственность», означает ответственность организации за воздействие решений, которые были предложены, на общество и окружающую среду.

Раздел, посвященный социальной ответственности организации, включает в себя следующие составляющие: техногенная безопасность, региональная безопасность, организационные мероприятия обеспечения безопасности, особенности законодательного регулирования проектных решений и безопасность в чрезвычайных ситуациях.

Научно-исследовательский проект представляет собой разработку программного обеспечения для тестирования и настройки игр с математической моделью и предполагает большой объем работы с ПК, поэтому важным критерием безопасности является организация рабочего места и режима трудовой деятельности. К опасным факторам труда разработчика-программиста относятся: недостаточная освещенность рабочей зоны, отклонение параметров микроклимата в помещении и уровень шума, а к вредным факторам: излучение электромагнитных полей, электробезопасность и пожарная безопасность[12].

5.1. Производственная безопасность

Опасные и вредные факторы[11], возникающие при разработке и эксплуатации проектируемого решения, перечислены в таблице 7

Таблица 7 – Вредные и опасные факторы труда

Факторы	Нормативные документы
1.Отклонение показателей микроклимата	СанПиН 2.2.4.548 – 96
2. Превышение уровня шума	СанПиН 2.2.4.3359-16
3.Отсутствие или недостаток естественного света	СанПиН 2.2.1 / 2.1.11278-03
4.Недостаточная освещенность рабочей зоны	
5.Повышенное значение напряжения в электрической цепи, замыкание которой может произойти через тело человека	ГОСТ Р 50571.17-2000 ГОСТ 12.4.124-83 ГОСТ Р 50571.5-94

5.1.1. Отклонение показателей микроклимата

Компьютеры могут привести к увеличению температуры и снижению относительной влажности в помещении. Микроклимат не вызывает повреждений или нарушений состояния здоровья, но могут приводить к возникновению общих и локальных ощущений теплового дискомфорта, напряжению механизмов терморегуляции, ухудшению самочувствия и понижению работоспособности. В СанПиН 2.2.4.548 – 96 [13] установлены величины параметров микроклимата, создающие комфортные условия.

Работа программиста относится к легкой категории 1Б [13]. В таблицах 8 и 9 представлены данные показатели для теплого периода года (плюс 10 °С и выше) и для холодного периода года [13].

Таблица 8 – Оптимальные величины показателей микроклимата

Период года	Температура воздуха, °С	Температура поверхностей, °С	Относительная влажность воздуха, %	Скорость движения воздуха, м/с
Холодный	21-23	20-24	40-60	0,1
Теплый	22-24	21-25		0,1

Таблица 9. – Допустимые величины показателей микроклимата

Период года	Температура воздуха, °С	Температура поверхностей, °С	Относительная влажность воздуха, %	Скорость движения воздуха, м/с
Холодный	19-24	18-25	15-75	0,1-0,2
Теплый	20-28	19-29		0,1-0,3

Если температура воздуха отличается от нормальной, то время пребывания в таком помещении должно быть ограничено в зависимости от категории тяжести работ. Температура в рассматриваемом помещении в холодное время года может опускаться до 19-21 °С, а в теплое время года подниматься до 25-28 °С. Данные показатели соответствуют допустимым значениям температуры.

Таблица 10 – Рекомендуемое время работы при температуре воздуха ниже допустимых величин (СанПиН 2.2.4.548 – 96) [13]

Температура воздуха, °С	Время пребывания, не более, ч
17	6
18	7

Таблица 11. – Рекомендуемое время работы при температуре воздуха выше допустимых величин (СанПиН 2.2.4.548 – 96) [13]

Температура воздуха, °С	Время пребывания, не более, ч
30,0	5
29,5	5,5
29,0	6

К мероприятиям по оздоровлению воздушной среды в производственном помещении относятся правильная организация вентиляции и кондиционирования воздуха, отопление помещений. В рассматриваемой аудитории вентиляция осуществляется естественным и механическим путём. В зимнее время в помещении предусматривается система отопления. Это обеспечивает нормальное состояние здоровья работников в аудитории.

5.1.2. Превышение уровня шума

Люди, которым приходится работать в условиях длительного шума, обычно имеют головные боли, раздражительность, сталкиваются со

снижением памяти, повышенной утомляемостью, также у многих понижен аппетит, есть боли в ушах и т. д. Перечисленные факты снижают производительность, работоспособность человека, а также качество труда [14].

Шумовой фон помещения создают десять одновременно работающих компьютеров. Также возникает шум, исходящий от принтера или телефонных аппаратов. Также источником шума является система вентиляции или шумы, поступающие извне помещения.

Во избежание негативных последствий от производственного шума, его необходимо регулировать в соответствие с нормами, которые указаны в ГОСТ 12.1.003-83«ССБТ. Общие требования безопасности» [15].

Допустимые уровни звука и звукового давления для рабочего места разработчика-программиста согласно пункту 2 выше указанного ГОСТ 12.1.003-83 представлены в таблице 12.

Таблица 12 – Предельно допустимые уровни звука (ГОСТ 12.1.003-83 [15])

Вид трудовой деятельности/ Частоты	Уровни звука и звукового давления, дБ, в октавных полосах со среднегеометрическими частотами, Гц								
	31,5	63	125	250	500	1000	2000	4000	8000
Научная деятельность, проектирование, программирование, Рабочие места проектно-конструкторских бюро, программистов вычислительных машин и т.д.	86	71	61	54	49	45	42	40	38

Допустимый уровень звукового давления колеблется от 38 дБ до 86 дБ при частоте от 8000 Гц до 31,5 Гц, соответственно.

Для уменьшения воздействий шума можно использовать следующие методы, согласно СП51.13330.2011[16]:

- экранирование рабочих мест, то есть установка перегородок между рабочими местами;
- установка оборудования, производящего минимальный шум.

Для снижения уровня шума, производимого персональными компьютерами, рекомендуется регулярно проводить их техническое обслуживание: чистка от пыли, замена смазывающих веществ; также применяются звукопоглощающие материалы.

5.1.3. Отсутствие или недостаток естественного света и недостаточная освещенность рабочей зоны

Рабочее (общее) освещение – это основное освещение, обеспечивающее нормальные условия для нахождения человека в помещении. Под нормальными понимаются условия жизнедеятельности человека, при которых он не напрягает зрение, чтобы выполнить любое действие, для которого данное помещение предназначено. [17].

Освещение в недостаточной степени может привести к напряжению зрения, ослаблению внимания и наступлению преждевременной утомленности. Ослепление, резь в глазах и раздражение могут быть вызваны чрезмерно ярким освещением. Свет на месте труда может создать сильные тени или отблески, а также дезориентировать работающего. Основным документом, регламентирующим нормы освещенности, является СП 52.13330.2016 Естественное и искусственное освещение. [17].

Основным показателем качества освещения является освещенность *E*– поверхностная плотность светового потока. По характеристике зрительной работы труд программиста относится к разряду III подразряду Г (высокой точности), т.е. наименьший размер объекта различения от 0,3 до 0,5 мм (точка) [3]. Это значит, что нормативное значение освещенности рабочего места должно быть 200 лк [17].

5.1.4.Электромагнитное излучение

Электромагнитное излучение, распространяющееся в пространстве возмущение электрических и магнитных полей [20]. Источниками электромагнитного излучения в данном исследовании являются мониторы и системный блок. В таблице 13 приведены нормы уровня ЭМП (СанПиН

2.2.2/2.4.1340–03[19]) ЭМП низкой частоты могут легко проникать вглубь тела и вызывать циркуляцию токов внутри него. При определенной величине они могут стимулировать активность нервной системы и мускулатуры и оказывать влияния на другие биологические процессы.

Таблица 13 – Допустимые уровни ЭМП, создаваемых ПК

Наименование параметров		ВДУ ЭМП
Напряженность электрического поля	в диапазоне частот 5 Гц - 2 кГц	25 В/м
	в диапазоне частот 2 кГц - 400 кГц	2,5 В/м
Плотность магнитного потока	в диапазоне частот 5 Гц - 2 кГц	250 нТл
	в диапазоне частот 2 кГц - 400 кГц	25 нТл
Электростатический потенциал экрана видеомонитора		500 В

Для того чтобы снизить воздействие излучения, рекомендуют применять такие мониторы, у которых уровень излучения понижен, а также установить защитные экраны и соблюдать режимы труда и отдыха.

5.1.5. Электробезопасность

Электробезопасность – система организационных и технических мероприятий, и средств, обеспечивающих защиту людей от вредного и опасного для жизни воздействия электрического тока, электрической дуги, электромагнитного поля и статического электричества.

Согласно СанПиН 2.2.2/2.4.1340-03 «Гигиенические требования к персональным электронно-вычислительным машинам и организации работы»[19], рабочее место должно находиться в безопасной зоне, которая не характеризуется наличием таких условий, как:

- повышенная влажность (относительная влажность воздуха длительно превышает 75%);
- высокая температура (более 35°С);
- наличие токопроводящей пыли и полов;
- возможность одновременного соприкосновения к имеющим соединения с землей металлическим элементам и металлическим корпусам электрооборудования.

Электрические установки, к которым относится ПК, представляют для человека большую потенциальную опасность, так как в процессе эксплуатации или проведения профилактических работ человек может коснуться комплектующих компьютера, находящихся под напряжением.

Специфическая опасность – корпуса ПК и прочего оборудования, оказавшегося под напряжением в результате повреждения или пробоя изоляции, не подают каких-либо сигналов, которые предупреждают человека об опасности. Причинами электропоражений являются: провода с поврежденной изоляцией, розетки сети без предохранительных кожухов.

Для защиты от поражения электрическим током все токоведущие части должны быть защищены от случайных прикосновений кожухами, корпус устройства должен быть заземлен. Заземление выполняется изолированным медным проводом сечением 1.5 мм^2, который присоединяется к общей шине заземления с общим сечением 48м^2 при помощи сварки. Общая шина присоединяется к заземлению, сопротивление которого не должно превышать 4 Ом.

Согласно ГОСТ Р 50571.5-94, питание устройства в помещении, в котором выполнялась работа, осуществляется от силового щита через автоматический предохранитель, который срабатывает при коротком замыкании нагрузки. Для снижения величин возникающих разрядов применяются покрытия из антистатического материала.

Контроль уровней электрического поля осуществляется по значению напряженности электрического поля. Контроль уровней магнитного поля осуществляется по значению напряженности магнитного поля или по значению магнитной индукции.

К мероприятиям по предотвращению возможности поражения электрическим током относятся:

● запрет на работы на задней панели ПЭВМ при включенном сетевом напряжении;

● мероприятия по сертификации ПЭВМ (ПК) и аттестации рабочих мест;

● применение экранов и фильтров;

● организационно-технические мероприятия;

● применение средств индивидуальной защиты путем экранирования пользователя ПК целиком или отдельных зон его тела.

Токи статического электричества, наведенные в процессе работы компьютера на корпусах, могут приводить к разрядам при прикосновении к этим элементам. Такие разряды не опасны для человека, но могут привести к выходу из строя компьютера. Для снижения величин токов статического электричества используются нейтрализаторы, местное и общее увлажнение воздуха, использование покрытия полов с антистатической пропиткой.

Программист работает с электроприборами: компьютером (монитор, системный блок, компьютерная мышь и клавиатура). В данном случае существует опасность электропоражения:

● при непосредственном прикосновении к токоведущим частям во время ремонта ПК;

● при прикосновении к нетоковедущим частям, оказавшимся под напряжением (в случае нарушения изоляции токоведущих частей ПК);

● при соприкосновении с полом, стенами, оказавшимися под напряжением;

● имеется опасность короткого замыкания в высоковольтных блоках: блоке питания и блоке дисплейной развёртки.

Рабочее место программиста должно быть оборудовано таким образом, чтобы исключить взаимное соприкосновение кабелей и шнуров питания соседних компьютеров.

К организационно-техническим мероприятиям относится первичный инструктаж по технике безопасности. Первичный инструктаж по технике безопасности является обязательным условием для допуска к работе в данном помещении.

5.2. Экологическая безопасность

Современные компьютеры непосредственно практически не оказывают никакого влияния на окружающую среду, так как электромагнитные излучения, производимые техникой ничтожно малы, вибрации и шумы так же практически неощутимы. С выполнением данной работы, могут быть связаны негативно влияющие на природу факторы, сопутствующие эксплуатации ПК. Многое сырье, используемое в сборке компьютеров, является токсичным. Следовательно, когда техника выходит из строя, возникает потребность в переработке отходов. Однако многие сегодня пренебрегают этим, поэтому отходы в виде неисправной техники не исчезают, превращаясь в свалки, негативно влияя на гидросферу и литосферу, или перерабатываются, что так же приносит вред.

Помимо техники, в рабочем процессе используются другие, сопутствующие рабочему процессу материалы, которые так же при неправильной утилизации наносят вред окружающей среде. Это могут быть батарейки, люминесцентные лампы и т.д.

Также необходимо рационально использовать природные ресурсы. Большое потребление бумаги ведет к вырубке лесов. Чрезмерное потребление электроэнергии ведет к увеличению выброса парниковых газов, что влияет на изменение климатических условий. Поэтому необходимо предпринимать различные меры, для того, чтобы как можно больше сократить негативное влияние на окружающую среду. В качестве таких мер, можно рассматривать:

- использование экономного режима работы электротехники;
- использовать энергосберегающие лампы для освещения помещений;
- устанавливать режим освещения, соответствующий времени года;
- правильно утилизировать отходы (компьютерный лом, газоразрядные лампы, бумага и др.);

- применять расходные материалы с высоким коэффициентом использования и возможностью их полной или частичной регенерации;
- использовать малотоксичные материалы при производстве компьютерной техники.

Если придерживаться мер снижения негативного влияния на окружающую среду, можно значительно снизить ущерб, наносимый природе в процессе эксплуатации и утилизации компьютеров и сопутствующей рабочему процессу технике.

5.3. Безопасность в чрезвычайных ситуациях

Одними из наиболее вероятных и разрушительных видов чрезвычайных ситуаций являются пожар или взрыв на рабочем месте.

Всякий работник при обнаружении пожара должен (О противопожарном режиме [21]):

- незамедлительно сообщить об этом в пожарную охрану;
- принять меры по эвакуации людей, каких-либо материальных ценностей согласно плану эвакуации;
- отключить электроэнергию, приступить к тушению пожара первичными средствами пожаротушения. Если потушить пожар самостоятельно не получается, то немедленно покинуть помещение;

При возникновении пожара должна сработать система пожаротушения, передав на пункт пожарной станции сигнал о ЧС. В случае если система не сработала, то необходимо самостоятельно произвести вызов пожарной службы по телефону 101, сообщить точный адрес места возникновения ЧС и ожидать приезда специалистов.

5.4. Правовые и организационные вопросы обеспечения безопасности

Большое значение для профилактики статических физических перегрузок имеет правильная организация рабочего места человека, работающего с ПЭВМ. Рабочее место должно быть организовано в

соответствии с требованиями стандартов, технических условий и (или) методических указаний по безопасности труда. Оно должно удовлетворять следующим требованиям:

- обеспечивать возможность удобного выполнения работ;
- учитывать физическую тяжесть работ;
- учитывать размеры рабочей зоны и необходимость передвижения в ней работающего;
- учитывать технологические особенности процесса выполнения работ.

Невыполнение требований к расположению и компоновке рабочего места может привести к получению производственной травмы или развития профессионального заболевания. Рабочее место программиста должно соответствовать требованиям СанПин 2.2.2/2.4.1340-03 [19] (таблица 14).

Таблица 14 – Требование к рабочему месту программиста.

Наименование показателя	Значение
Высота рабочей поверхности стола	Должна регулироваться. 680 -800 мм;725 мм при отсутствии регуляции
Пространство для ног: высота ширина глубина на уровне колен глубина на уровне вытянутых ног	 не менее 600 мм; не менее 500 мм; не менее 450 мм; не менее 650 мм
Ширина и глубина поверхности сиденья	не менее 400 мм;
Регулировка высоты поверхности сиденья	400 -550мм;
Угол наклона сидения: вперед назад	 до 15 град; до 5 град;
Опорная поверхность спинки: высота ширина	 300 +-20 мм; не менее 380 мм;
Угол наклона спинки в вертикальной плоскости в пределах	+-30 градусов;
Регулировка расстояния спинки от переднего края сиденья	260 -400 мм;
Стационарные или съемные подлокотники: длина ширина	 не менее 250 мм; 50 -70 мм;
Регулировка: подлокотников по высоте над сиденьем внутреннего расстояния между подлокотниками	 230 +-30 мм; 350 -500 мм;
Подставка для ног: ширина глубина высота угол наклона опорной поверхности подставки Поверхность подставки должна быть рифленой и иметь по переднему краю бортик высотой 10 мм.	 не менее 300 мм; не менее 400 мм; до 150 мм; до 20°;
Расположение экрана монитора	600 - 700 мм, но не ближе 500 мм с учетом размеров алфавитно-цифровых знаков и символов.
Расположение клавиатуры	100 - 300 мм от края, обращенного к пользователю или на специальной, регулируемой по высоте рабочей поверхности, отделенной от стола.

В ТОИ Р-45-084-01 «Типовая инструкция по охране труда при работе на персональном компьютере» (нормы для творческой работы) и СанПиН 2.2.2/2.4.1340-03 «Гигиенические требования к персональным электронно-

вычислительным машинам и организации работы»[19] содержатся пункты по организации работы за ПЭВМ:

- Продолжительность рабочей недели не должна превышать 40 часов в неделю;
- Рабочие места с компьютерами должны размещаться таким образом, чтобы расстояние от экрана одного видеомонитора до тыла другого было не менее 2,0 м, а расстояние между боковыми поверхностями видеомониторов - не менее 1,2 м
- Продолжительность непрерывной работы за компьютером без регламентированного перерыва не должна превышать 1 час;
- Рекомендуется делать перерывы в работе за ПК продолжительностью 10-15 минут через каждые 45-60 минут работы;
- Во время регламентированных перерывов целесообразно выполнять комплексы упражнений и осуществлять проветривание помещения;
- Не рекомендуется работать за компьютером более 6 часов за смену.

ЗАКЛЮЧЕНИЕ ПО РАЗДЕЛУ

Используя все приведенные правила и нормы, касающиеся работы с ПЭВМ, исследование и дальнейшая эксплуатация разрабатываемого модуля не приведет к ухудшению здоровья работника и окружающей его среды.

ЗАКЛЮЧЕНИЕ

Практическим результатом работы стало улучшение навыков программирования, знакомство с необходимыми для Java-разработчика технологиями. Было реализовано программное обеспечение для разработки и тестирования игр с математической моделью, которое получило применение в компании.

В результате было реализовано окружение для тестирования и настройки игр с математической моделью. Данное окружение состоит из двух модулей: модуль для тестирования и модуль для настройки. Существует потенциал для дальнейшей модернизации модуля для настройки математической модели. Дальнейшая модернизация тестового модуля не кажется возможной, так как модуль достиг предела развития.

При проектировании данного окружения был обоснован выбор проектного решения поставленной задачи.

При разработке данного окружения были углублены знания языка программирования Java, улучшились навыки многопоточного программирования, получен опыт в создании архитектуры программного обеспечения, получен опыт коммерческой разработки.

Разработки получила применение в компании ООО «Коннектив Геймс» в сфере разработки игр.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1) Э. Гамма Приёмы объектно-ориентированного проектирования. Паттерны проектирования / Э. Гамма, Р. Хелм, Р. Джонсон, Дж. Влиссидес. – СПб: Питер, 2001. — 368 с.: ил. (Серия «Библиотека программиста»)

2) Философия Java / Брюс Эккель. – 4-ое полное – Санкт-Петербург: Питер, 2017. – 1168 с.

3) Генетические алгоритмы или как учебник по биологии может помочь в функциональной оптимизации // LAZYSMART[Электронный ресурс]. – URL: http://lazysmart.ru/iskusstvenny-j-intellekst/geneticheskie-algoritmy-ili-kak-uchebn/ (Дата обращения 27.05.2018).

4) Классический генетический алгоритм. Часть IV. Скрещивание, мутация, создание популяции // aiportal[Электронный ресурс]. – URL: http://www.aiportal.ru/articles/genetic-algorithms/classic-alg-part4.html (Дата обращения 27.05.2018).

5) Лекция 1: Введение. Основы генетических алгоритмов// Интуит [Электронный ресурс]. – URL: https://www.intuit.ru/studies/courses/14227/1284/lecture/24168? (Дата обращения 27.05.2018).

6) Многопоточность в Java // habr [Электронный ресурс]. – URL: https://habr.com/post/164487/ (Дата обращения 27.05.2018).

7) CasinoSoftware // iGamingSuplier [Электронный ресурс]. – URL: http://www.igamingsuppliers.com/suppliers/online-gaming/games-software/casino-software/ (Дата обращения 27.05.2018).

8) GLOBAL SLOT MACHINE MARKET2018-2022 // Marketreserch.com [Электронный ресурс]. – URL: https://www.marketresearch.com/infiniti-research-limited-v2680/global-slot-machine-11912012/ (Дата обращения 27.05.2018).

9) Онлайн-казино набирают популярность// byyd [Электронный ресурс]. – URL: https://habr.com/ru/company/byyd/blog/302880/ (Дата обращения 27.05.2018).

10) Скрытая сложность видеоигр слотов // Timothy Ryan [Электронный ресурс]. – URL: https:/habr.com/ru/post/449006/(Дата обращения 27.05.2018).

11) Международный стандарт ICCSR26000:2011 «Социальная ответственность организации»

12) ГОСТ 12.0.003-2015ССБТ. Опасные и вредные производственные факторы. Классификация. - М.: Издательство стандартов, 2001. – 4 с.

13) СанПиН 2.2.4.548 – 96. Гигиенические требования к микроклимату производственных помещений. – М.: Информационно-издательский центр Минздрава России, 1997. – 20 с.

14) Борьба с шумом на производстве: Справочник / Е.Я. Юдин, Л.А. Борисов; Под общ. ред. Е.Я. Юдина – М.: Машиностроение, 1985. – 400с.

15) ГОСТ 12.1.003-83. ССБТ. Общие требования безопасности. – М.: Издательство стандартов, 2002. – 13 с.

16) СП 51.13330.2011 Защита от шума. Актуализированная редакция СНиП 23-03-2003 (с Изменением N 1) [Электронный ресурс] / Электронный фонд правовой и нормативно-технической документации. Режим доступа: http://docs.cntd.ru/document/1200084097(07.06.2019).

17) СП 52.13330.2016 Естественное и искусственное освещение. Актуализированная редакция СНиП 23-05-95* [Электронный ресурс] / Электронный фонд правовой и нормативно-технической документации. Режим доступа: http://docs.cntd.ru/document/456054197 (07.06.2019).

18) ГОСТ 6825-91. Лампы люминесцентные трубчатые для общего освещения. – М.: Издательство стандартов, 1992. – 242 с.

19) СанПиН 2.2.2/2.4.1340-03. Гигиенические требования к персональным электронно-вычислительным машинам и организации

работы.– М.: Информационно-издательский центр Минздрава России, 2003. – 54 с.;

20) Безопасность жизнедеятельности. /Под ред. Н.А. Белова - М.: Знание, 2000 - 364с.

21) О противопожарном режиме (с изменениями на 7 марта 2019 года) [Электронный ресурс] / Правительство российской федерации, постановление от 25 апреля 2012 года N 390 «О противопожарном режиме» /Электронный фонд правовой и нормативно-технической документации. Режим доступа: http://docs.cntd.ru/document/456054197 (07.06.2019).

22) Приказ Министерства Российской Федерации по связи и информатизации от 2 июля 2001 г. N 162 «ТОИ Р-45-084-01 Типовая инструкция по охране труда при работе на персональном компьютере»

Приложение А

Программная реализация класса PlayStrategy.

```
public class PlayStrategy<STATS extends IStats, GAME extends
AbstractCgGame<?, STATS, ?, ?>>
implements IPlayStrategy<GAME> {

private static final Map<String, Action<GameStateBean>>DEFAULT_ACTIONS = new
HashMap<>();
private static final Comparator<Action<GameStateBean>>PRIORITY_COMPARATOR =
            Comparator.<Action<GameStateBean>,
Integer>comparing(Action::getPriority).reversed();

static {
DEFAULT_ACTIONS.put(CgBonusSelectActionHandler.ACTION, new
Action<>(PlayStrategy::selectBonus, bean ->true, 1));
DEFAULT_ACTIONS.put(CgSelectModifierActionHandler.ACTION, new
Action<>(PlayStrategy::selectModifier, bean ->true, 1));
    }

private GAME game;
private STATS stats;
private final Map<String, Action<GameStateBean>>actions;

public PlayStrategy(STATS stats) {
this.stats = stats;
actions = new HashMap<>();
actions.putAll(DEFAULT_ACTIONS);
    }

public PlayStrategy(STATS stats, Map<String, Action<GameStateBean>>
extraAction) {
this(stats);
actions.putAll(extraAction);
    }

public void selectModifier() throws CommonException {
game.selectModifier(Rng.getInstance().nextInt(3), stats, (int[]) null);
    }

public void play(GAME game) throws CommonException {
        playMain(game);
        List<String> possibleActions = new ArrayList<>(game.getActions());
        possibleActions.removeIf(MAKE_BET::equals);
while (!possibleActions.isEmpty()) {
            List<Action<GameStateBean>> actions = new ArrayList<>();
for (String possibleAction : possibleActions) {
                Action<GameStateBean> Action =
this.actions.get(possibleAction);
if (Action != null && Action.checkCondition(game.getStateBean())) {
                    actions.add(Action);
                }
            }
            actions.sort(PRIORITY_COMPARATOR);
            actions.get(0).play(this);
            possibleActions = new ArrayList<>(game.getActions());
            possibleActions.removeIf(MAKE_BET::equals);
```

```
        }
    }

@Override
public void playMain(GAME game) throws CommonException {
this.game = game;
        game.play();
    }

@Override
public void selectBonus() throws CommonException {
        IBonusGameIndex stateBean = (IBonusGameIndex) game.getStateBean();
int[] possibleIndex = stateBean.getPossibleIndex();
if (possibleIndex.length == 0){
            possibleIndex = new int[]{0};
        }
game.selectBonusPrize(possibleIndex[Rng.getInstance().nextInt(possibleIndex.l
ength)], stats, (int[]) null);
    }
}
```

Приложение Б

Программная реализация класс Action.

```
public class Action <BEAN extends GameStateBean> {

private final LambdaExceptionUtil.ConsumerWithExceptions<PlayStrategy,
CommonException>action;
private final Function<BEAN, Boolean>condition;
private final int priority;

public Action(LambdaExceptionUtil.ConsumerWithExceptions<PlayStrategy,
CommonException> action,
                      Function<BEAN, Boolean> condition,
int priority) {
this.action = action;
this.condition = condition;
this.priority = priority;
    }

public LambdaExceptionUtil.ConsumerWithExceptions<PlayStrategy,
CommonException> getAction() {
return action;
    }

public Function<BEAN, Boolean> getCondition() {
return condition;
    }

public boolean checkCondition(BEAN gameBean) {
return condition.apply(gameBean);
    }

public void play(PlayStrategy playStrategy) throws CommonException {
action.accept(playStrategy);
    }

public int getPriority() {
return priority;
    }
}
```

Приложение В

Реализация генетического алгоритма

```
private List<Individual<STAT>> startGenAlgorithm(int iterations, int
firstGenerationSize, int nOfCrossing, int[][][] baseSolution, int
maxPopulation, double suitableLimit, double unsuitableLimit,int reductionSize,
int nOfChild, int solutionTypes)throws InterruptedException {
    List<Individual<STAT>> ancestorIndividuals = new ArrayList<>();
if (baseSolution != null) {
try {
            List<Chromosome> chromosomes = new ArrayList<>();
for (int id = 0; id < baseSolution.length; id++) {
                chromosomes.add(new
Chromosome(ArrayUtils.copy(baseSolution[id]),
this,
0,
                        id));
            }
            ancestorIndividuals.add(new Individual<>(chromosomes, this,
nOfIterationToGetStats));
        } catch (UnsuitableException e) {
            System.out.print("abnormal primary chromosome with base
solutions: ");
            e.print();
        }
    }
while (ancestorIndividuals.size() < firstGenerationSize) {
try {
            List<Chromosome> chromosomes = new ArrayList<>();
for (int solution = 0; solution < solutionTypes; solution++) {
int[][] solutions = generate(solution);
                chromosomes.add(new Chromosome(solutions,
this,
0,
                        solution));
            }
            Individual<STAT> individual = new Individual<>(chromosomes, this,
nOfIterationsToGetStats);
            ancestorIndividuals.add(individual);
            System.out.println("individual created");
        } catch (UnsuitableException e) {
            System.out.print("abnormal primary chromosome: ");
            e.print();
        }
    }
long totalTime = System.currentTimeMillis();
    List<Individual<STAT>> population = new ArrayList<>(ancestorIndividuals);
    Individual<STAT> bestIndividual = Individual.findBest(population);
for (int iter = 0; iter < iterations; iter++) {
long time = System.currentTimeMillis();

        List<Individual<STAT>> descendantIndividuals = new ArrayList<>();
        ExecutorService executorService =
Executors.newFixedThreadPool(Runtime.getRuntime().availableProcessors());
        CountDownLatch finishLatch = new CountDownLatch(nOfCrossing);
        Semaphore semaphore = new Semaphore(1);
        population.sort(Comparator.naturalOrder());
```

```
for (int i = 0; i < nOfCrossing; i++) {
            Individual<STAT> finalBestIndividual = bestIndividual;
            executorService.submit(() -> {
                List<Individual<STAT>> parents = null;
                List<Individual<STAT>> children = new ArrayList<>();
boolean isSuccesses = false;
int tries = 0;
while (!isSuccesses) {
                    parents = selection(population);
double mutateChance = calcMutationChance(Rng.getInstance().nextDouble());
                    children.addAll(crossing(parents.get(0),
                            parents.get(1),
                            mutateChance,
finalBestIndividual,
triesToCrossing));
if (children.size() >= nOfChild) {
                        isSuccesses = true;
                        children.sort(Comparator.naturalOrder());
                    }
                    tries++;
                }
if (containAny(PrintEnum.ALL_WITHOUT_FAILED, PrintEnum.SUCCESSFUL)) {
                    System.out.println("successful crossing: "+
parents.get(0).getParentInfo() +
                            parents.get(1).getParentInfo() +
"tries="+ tries + "; ");
for (Individual<STAT> child : children) {
                        System.out.println("\t"+ child.getChildInfo());
                    }
                }
try {
semaphore.acquire();
descendantIndividuals.addAll(children);
semaphore.release();
                } catch (InterruptedException e) {
                    System.out.println("Exception in semaphore");
                    e.printStackTrace();
                }
finishLatch.countDown();
            });
        }
        finishLatch.await();
        executorService.shutdown();
        executorService.awaitTermination(1, TimeUnit.DAYS);
        population.addAll(descendantIndividuals);
        Individual<STAT> previousBestIndividual = bestIndividual;
if (population.size() >= maxPopulation) {
            reduction(reductionSize, population);
        }
        bestIndividual = Individual.findBest(population);
        population.sort(Comparator.comparingInt(Individual::getId));

System.out.println("**********************************************************
*********");
        System.out.println("Iteration="+ iter + "    The best fitness="+
bestIndividual.getFitness() +
"    Time spent iterating=(msec)"+ (System.currentTimeMillis() - time) +
"    Total time spent(sec)="+ ((System.currentTimeMillis() - totalTime) /
```

```
1000));
if (containAny(PrintEnum.ALL_WITHOUT_FAILED, PrintEnum.POPULATION_INFO)) {

System.out.println("**********************************************************
*********");
            System.out.println("INDIVIDUALS, population size="+
population.size());
            System.out.print("index:  \t");
            population.forEach(individual ->
System.out.print(FormatTools.pad(individual.getId(), 6) + "\t"));
            System.out.println();
            System.out.print("fitness:\t");
            population.forEach(individual ->
System.out.print(FormatTools.pad(Doubles.round(individual.getFitness(), 3),
6) + "\t"));
            System.out.println();
        }
if (previousBestIndividual != bestIndividual) {
            System.out.println("NEW THE BEST INDIVIDUAL");
if (containAny(PrintEnum.ALL_WITHOUT_FAILED, PrintEnum.FITNESS)) {
                System.out.println("INDIVIDUAL FITNESS:");
                bestIndividual.printFitness(System.out, suitableLimit,
unsuitableLimit, statComparator);
            }
if (containAny(PrintEnum.ALL_WITHOUT_FAILED, PrintEnum.BEST_SOLUTION_STAT)) {

System.out.println("**********************************************************
*********");
                System.out.println("INDIVIDUAL STAT");
                bestIndividual.printStat(System.out);
            }
if (containAny(PrintEnum.ALL_WITHOUT_FAILED, PrintEnum.BEST_SOLUTION_SCAN)) {
                System.out.println("INDIVIDUAL SOLUTIONS");
                printSolutions(bestIndividual.get(), System.out);
System.out.println("**********************************************************
*********");
            }
        }
    }
return population;
}
```

Приложение Г

Метод для многоточечного скрещивания двух объектов класса Individual.

```
List<Individual<STATS>> multiPointCrossing(Individual<STATS> parent, double
mutateChance, int nOfPoints, Individual<STATS> bestIndividual,
int triesToCrossing) {
this.bestIndividual = bestIndividual;
    List<Individual<STATS>> newIndividuals = new ArrayList<>();
while (triesToCrossing >0) {
        List<Chromosome> child1 = new ArrayList<>();
        List<Chromosome> child2 = new ArrayList<>();
for (int Type = 0; Type <chromosomes.size(); Type++) {
            List<Chromosome> crossing =
chromosomes.get(Type).multiPointCrossing(parent.chromosomes.get(Type),
                    mutateChance,
                    nOfPoints,
                    getChromosomeInfo(Type),
                    parent.getChromosomeInfo(Type));
if (crossing.isEmpty()) {
                Type--;
            } else if (crossing.size() == 1) {
                child1.add(crossing.get(0));
                child2.add(crossing.get(0));
            } else {
                child1.add(crossing.get(0));
                child2.add(crossing.get(1));
            }
        }
int mutationSymbolNumber = child1.stream()
                .mapToInt(Chromosome::getMutationSymbolNumbers)
                .sum();

        createAndAddChromosomePack(newIndividuals, child1,
mutationSymbolNumber);

        mutationSymbolNumber = child2.stream()
                .mapToInt(Chromosome::getMutationSymbolNumbers)
                .sum();
        createAndAddChromosomePack(newIndividuals, child2,
mutationSymbolNumber);
        triesToCrossing--;
if (!newIndividuals.isEmpty()) {
break;
        }
    }

    exchange(newIndividuals, chromosomes, parent.chromosomes, 0);
if (chromosomes.size() >2) {
for (int Type = 1; Type <chromosomes.size(); Type++) {
            exchange(newIndividuals, chromosomes, parent.chromosomes, Type);
        }
    }
return newIndividuals;
}
```

```
private void createAndAddChromosomePack(List<Individual<STATS>>
newIndividuals, List<Chromosome> child, int mutationSymbolsNumber) {
try {
        newIndividuals.add(new Individual<>(child, this,
mutationSymbolsNumber));
    } catch (SolutionGenerator.UnsuitableException e) {
if (solutionGenerator.containAny(SolutionGenerator.PrintEnum.FAILED)) {
            e.print();
        }
    }
}

private void exchange(List<Individual<STATS>> newIndividual, List<Chromosome>
chromosomes1, List<Chromosome> chromosomes2, int Type) {
    exchange(chromosomes1, chromosomes2, Type, newIndividual);
    exchange(chromosomes2, chromosomes1, Type, newIndividual);
}

private void exchange(List<Chromosome> recepient, List<Chromosome> donor,
int Type, List<Individual<STATS>> packs) {
    List<Chromosome> changedRecepient = recepient.stream()
            .map(Chromosome::new)
            .collect(Collectors.toList());
    changedRecepient.set(Type, donor.get(Type));
    createAndAddChromosomePack(packs, changedRecepient, -1);
}
```

Приложение Д

Методы для многоточечного скрещивания двух объектов класса Chromosome:

```
List<Chromosome> multiPointCrossing(Chromosome parent, double mutateChance,
int nOfPoints, String... chromosomeInfo) {
List<int[][]> parents = Arrays.asList(this.solutions, parent.solutions);
int crossingNumber = Rng.getInstance().nextInt(this.solutions.length);
return formChildrenSolution(parents, crossingNumber, nOfPoints).stream()
            .map(childSolution -> createChromosome(childSolution,
mutateChance, chromosomeInfo))
            .filter(Objects::nonNull)
            .collect(Collectors.toList());
}
private List<int[][]> formChildrenSolution(List<int[][]> parentsSolutions,
int crossingSolutionNumber, int nOfPoints) {
    List<List<Integer>> parentList = parentsSolutions.stream()
            .map(solutions ->
Arrays.stream(solutions[crossingSolutionNumber])
                    .boxed()
                    .collect(Collectors.toList()))
            .collect(Collectors.toList());
    List<List<Integer>> points = new ArrayList<>();
for (List<Integer> aParentList : parentList) {
        points.add(createExtendedPointsSet(aParentList.size(), nOfPoints));
    }
    List<int[][]> childrenSolutions = parentsSolutions.stream()
        .map(ArrayUtils::copy)
        .peek(solutions -> solutions[crossingSolutionNumber] = new int[]{})
        .collect(Collectors.toList());
    List<int[][]> children = new ArrayList<>(); //из-за эксепции могут не все
элементы childrenSolutions изменится
for (int childIndex = 0; childIndex < childrenSolutions.size();childIndex++){
try {
int[][] childSolution = childrenSolutions.get(childIndex);
            childSolution[crossingSolutionNumber] =
formChild(parentList.get(childIndex % 2),parentList.get((childIndex + 1)% 2),
                    crossingSolutionNumber, points.get(childIndex % 2),
points.get((childIndex + 1) % 2));
            children.add(childSolution);
        } catch (SolutionGenerator.UnsuitableException e) {
if (solutionGenerator.containAny(SolutionGenerator.PrintEnum.FAILED)) {
                e.print();
            }
        }
    }
return children;
}
private List<Integer> createExtendedPointsSet(int parentLength, int
pointsNumber) {
    List<Integer> points = new ArrayList<>();
    points.add(0);
    points.add(parentLength);
for (int point = 0; point < pointsNumber; point++) {
int p;
do {
```

```
            p = Rng.getInstance().nextInt(parentLength);
        } while (points.contains(p));
        points.add(p);
    }
    points.sort(Comparator.naturalOrder());
return points;
}
private int[] formChild(List<Integer> parent1List, List<Integer> parent2List,
int crossingSolutionNumber, List<Integer> pointsFirstParent, List<Integer>
pointsSecondParent)
throws SolutionGenerator.UnsuitableException {
    List<Integer> childList = new ArrayList<>();
int startPosition = 0;
for (int point = 0; point < pointsFirstParent.size(); point++) {
if (point % 2 == 0) {
            childList.addAll(parent1List.subList(startPosition,
pointsFirstParent.get(point)));
            startPosition = pointsSecondParent.get(point);
        } else {
            childList.addAll(parent2List.subList(startPosition,
pointsSecondParent.get(point)));
            startPosition = pointsFirstParent.get(point);
        }
    }
if (childList.size()
>solutionGenerator.getSolutionMaxLength()[crossingSolutionNumber]) {
throw new SolutionGenerator.UnsuitableException("very long solution");
    }
return childList.stream().mapToInt(Integer::intValue).toArray();
}
private Chromosome createChromosome(int[][] childSolutions, double
mutateChance, String[] chromosomeInfo) {
int mutationSymbolNumbers = mutate(childSolutions, mutateChance);
try {
        checkSolution(childSolutions);
return new Chromosome(childSolutions, solutionGenerator,
mutationSymbolNumbers, id);
    } catch (SolutionGenerator.UnsuitableException e) {
if (solutionGenerator.containAny(SolutionGenerator.PrintEnum.FAILED)) {
            System.out.print("unsuccessful crossing: "+ chromosomeInfo[0] +
chromosomeInfo[1]);
            e.print();
        }
    }
return null;
}
```

Приложение Е

Метод поиска минимального числа итераций для сбора «адекватной» статистики.

```
private static Pair<Integer, Integer> findMinEssays(
        StandPlayerFabric standPlayerFabric,
        int step,
        long maxEssays,
        long seed,
        int tries,
        int minZero,
        PrintStream out)
throws Exception {
ITestStandPlayer player = standPlayerFabric.createStandPlayer(maxEssays);
collectStats(seed, player);
int zeroQuantity = sortAndFindFirstZero(player.convertStatsToLongArray());
if (tries == -1 && zeroQuantity > minZero) {
return new Pair<>(Integer.MAX_VALUE, minZero);
}
player = standPlayerFabric.createStandPlayer(step);
collectStats(seed, player);
int firstZero = sortAndFindFirstZero(player.convertStatsToLongArray());
int steps = 1;
while (zeroQuantity != firstZero) { //find min steps to achieve zeroQuantity
player.continueGame(step);
firstZero = sortAndFindFirstZero(player.convertStatsToLongArray());
steps++;
if (steps > maxEssays / step) {
throw new Exception(standPlayerFabric.getGameName() + " a lot of steps");
}
}
for (int i = 0; i < tries && steps >1; i++) {
SecureRandomFactory.getInstance().reset();
Rng.getInstance().reset(); //if don't reset, int newSeed =... is constant
int newSeed = Rng.getInstance().nextInt();
 Pair<Integer, Integer> result = findMinEssays(standPlayerFabric,step,
step * steps, newSeed, -1, zeroQuantity,out);
int steps0 = result.getFirst();
int zeros = result.getSecond();
if (steps0 < steps && zeros <= zeroQuantity) {
            steps = steps0;
            zeroQuantity = zeros;
            seed = newSeed;
        }
    }
if (tries != -1) {
        String gameName = FormatTools.pad(
formFileName(standPlayerFabric.getGameName(), standPlayerFabric.getLines()),
25);
        String stepsStr = FormatTools.pad("steps="+ steps, 15);
        String s = gameName + stepsStr + "\tzeros="+ firstZero + "\tstep
size="+ step + "\tseed="+ seed;
        out.println(s);
    }
return new Pair<>(steps, zeroQuantity);
}
```