МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«Южно-Уральский государственный университет»
(национальный исследовательский университет)
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

РАБОТА ПРОВЕРЕНА

Рецензент
ведущий программист
ООО ВОРТЕКСКОД

_____ П.А. Михайлов

“___”_____ 2017 г.

ДОПУСТИТЬ К ЗАЩИТЕ

Заведующий кафедрой, д.ф.-м.н.,
профессор

_____ Л.Б. Соколинский

“___”_____ 2017 г.

РАЗРАБОТКА АДАПТИВНОГО БОТА ДЛЯ ИГРОВОГО ПРИЛОЖЕНИЯ В ЖАНРЕ СИМУЛЯТОР ВЫЖИВАНИЯ В ГОРОДСКОЙ СРЕДЕ

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
ЮУрГУ – 02.03.02.2017.115-056.ВКР

Научный руководитель
к.ф.-м.н., доцент кафедры СП
_____ В.А. Голодов

Автор работы,
студентка группы КЭ-401
_____ Г.А. Ядрышникова

Ученый секретарь
(нормоконтролер)

_____ О.Н. Иванова

“___”_____ 2017 г.

Челябинск-2017

ОГЛАВЛЕНИЕ

ГЛОССАРИЙ

Моб (mob, mobile, сокр. от англ. mobile object) – изначально данный термин был введен для обозначения самостоятельно перемещающихся объектов игры MUD1[3], впоследствии все чаще стал употребляться в значении монстров (агрессивных мобов) и вообще любых неигровых и игровых персонажей. В Minecraft моб – это живая, движущаяся игровая сущность (НПС) [8].

Бот (игровой) – программа, основанная на модуле искусственного интеллекта, имитирующая партнера в игре [2].

ВВЕДЕНИЕ

Актуальность

С момента изобретения компьютера людей волнует вопрос расширения области их применения для решения все более сложных проблем. Программисты пытаются автоматизировать задачи, считающиеся интеллектуальными [6], обучая программы имитировать когнитивные процессы, составляющие мыслительные способности человека, чтобы машины также могли познавать, обучаться, мыслить логически, анализировать, систематизировать, классифицировать информацию и т.д.

В сфере компьютерных игр искусственный интеллект (ИИ) определяет образ действий персонажей, управляемых компьютером [1]. Его задачей, в отличие от традиционного ИИ, является не максимально эффективное достижение поставленных (в данном случае, перед персонажами) целей, а обеспечение достоверного и интересного поведения персонажей в условии ограниченных вычислительных ресурсов. Поэтому при разработке применяются упрощения, эмуляции и обман игроков [12].

В основе механизма принятия решений могут лежать заданные шаблоны действий, конечные автоматы с набором состояний, деревья решений, планировщики [5] или даже нейронные сети [14]. Современные разработчики стараются делать ИИ адаптивным: заставлять анализировать предыдущий опыт взаимодействия с игроком и использовать его для определения следующих своих действий и предугадывания действий игрока.

Одной из главных проблем остается поиск оптимального уровня сложности стратегии персонажей. С одной стороны, если их поведение будет нелогичным или предсказуемым, игроку станет скучно, с другой – если противники будут действовать слишком эффективно, у игрока не останется шанса на победу. Поскольку взаимодействие с персонажами зачастую составляет большую часть геймплея, от этого фактора напрямую зависит качество игры.

Если рассмотреть наиболее популярные и покупаемые игры, можно

заметить, что большинство из них – многопользовательские: World of Warcraft, Counter-Strike, League of Legends, DOTA 2 [9]. Когда ход игры зависит от действий нескольких игроков, каждому отдельному участнику гарантируется множество уникальных игровых ситуаций, и это мотивирует возвращаться к ней снова и снова [4]. Адаптивные, разнообразно и непредсказуемо действующие персонажи тоже способны делать уникальным каждое прохождение.

Например, в ролевом шутере S.T.A.L.K.E.R используется сложная система симуляции жизни A-Life (Artificial Life), которая управляет населяющими игровой мир существами даже когда их не видит игрок [10]. Персонажи живут на протяжении всех уровней, выполняя генерируемые для них задания, и игрок сталкивается с последствиями их закадровых действий. Это стало одной из ключевых особенностей игры, поспособствовавшей ее популярности.

Таким образом, повышение качества ИИ в играх является актуальной проблемой.

Minecraft (от англ. mine «шахта», «добывать» и англ. craft «ремесло») – ставшая невероятно успешной инди-игра в жанре песочницы с открытым миром и элементами симулятора выживания, предоставляющая игрокам большую творческую свободу.

В рамках геймплея игроку предоставляется трехмерный процедурно-генерируемый мир из кубических блоков, который можно свободно перестраивать, создавая из блоков сложные сооружения. Перед игроком не ставится однозначной цели, но ему предлагается большое количество возможностей и занятий: можно исследовать мир, создавать разнообразные сооружения и предметы, сражаться с различными противниками. Также внутри игры есть ресурсы, позволяющие имитировать устройства реальной электроники и булеву логику [13].

Также существует широкий спектр инструментов для создания модификаций игры для всех операционных систем (Windows, OS X, Linux,

Android, IOS, Raspberry Pi и Windows Phone). Поскольку сама игра написана на Java, в качестве языка программирования практически всегда используется он. Для написания непосредственно кода подходят все универсальные платформы и среды разработки Java (Eclipse, NetBeans, JDeveloper и т.д). При разработке можно использовать API интерфейсы (самый распространенный на данный момент – Minecraft Forge [15]), или с помощью декомпиляторов, таких как Minecraft Coder Pack, работать напрямую с исходным кодом. Для создания трехмерных моделей, помимо стандартных 3D-редакторов (Blender, BDcraft Cubik), можно использовать редакторы, написанные специально для Minecraft (MC Model Maker, MrCrayfish's Model Creator).

Модификации могут как добавлять новую, так и изменять оригинальную механику Minecraft. Работать можно со всеми ключевыми составляющими игры: измерениями (игровыми мирами), графикой, блоками, предметами и пр. В том числе, в сообществе моддеров актуален вопрос усовершенствования ИИ, управляющего мобами.

Мир Minecraft населен дружелюбными, нейтральными и враждебными мобами. ИИ представлен в виде множества классов, описывающих различные действия (бродить, атаковать, плыть, гореть), которые назначаются мобам с определенным приоритетом. Помимо обычных задач, определяющих базовое поведение, можно приписывать целевые задачи, влияющие на направление передвижения моба по миру [7]. Инструменты позволяют создавать свои классы действий, изменять существующих и добавлять новых персонажей с разнообразными поведенческими паттернами.

Таким образом разработка адаптивного бота для игры Minecraft является актуальной задачей.

Цель и задачи

Целью работы является разработка адаптивного враждебного бота для игры Minecraft.

Для достижения поставленной цели необходимо решить следующие

задачи:

1) ознакомиться с Minecraft Forge API;
2) разработать модель поведения бота;
3) адаптировать модель поведения бота;
4) реализовать приложение в соответствии с разработанной моделью;
5) протестировать адаптивные возможности бота.

Структура и объем работы

Работа состоит из введения, пяти разделов, заключения, библиографии и приложения. Объем работы составляет 24 страницы, объем библиографии составляет 15 источников, объем приложения – 1 страница.

Содержание работы

В первом разделе описывается принцип работы ИИ в Minecraft, частью которого будет являться разрабатываемый бот.

Во втором разделе содержится проектирование модели поведения бота: рассматриваются доступные входные данные, выделяются состояния и параметры бота, описываются модели передвижения и атаки.

В третьем разделе рассматриваются способы адаптации модели бота к внешним условиям, описывается выбранная модель оптимизации параметров.

Четвертый раздел содержит описание реализации модели бота.

Пятый раздел содержит некоторые результаты тестирования адаптивности бота.

В заключении приведены результаты работы.

Приложение содержит схему полной модели поведения бота.

1. ОПИСАНИЕ ВСТРОЕННОГО ИИ MINECRAFT

Функционал любого персонажа в Minecraft представлен в виде классов EntityAI<ClassName>, расширяющих класс EntityAIBase и реализующих состояния персонажа: простейшие законченные действия с его стороны (плыть, ломать дверь) и в его сторону (получить ранение, контролироваться игроком) [11]. Для того чтобы похожие действия не запускались одновременно и не конфликтовали, используется битовая маска, устанавливаемая в конструкторе класса с помощью метода setMutexBits(). Остальные обязательные методы класса описаны в таблице 1.

Табл. 1. Методы класса EntityAIBase

shouldExecute()	возвращает истину, если действие должно быть запущено
continueExecuting()	возвращает истину, если выполнение должно продолжаться
isInterruptible()	возвращает истину, если действие может быть прервано
startExecuting()	инициализация
resetTask()	сбросить действие
updateTask()	обновить действие

Действия назначаются в два независимых потока задач, способных выполняться параллельно: один поток (<EntityName>.tasks) отвечает за перемещение, второй (<EntityName>.targetTasks) – за цели перемещения (например, поиск ближайшей цели для атаки). Каждой задаче назначается приоритет выполнения. Каждый такт процессора ИИ выполняет следующую последовательность действий для каждого списка отдельно [7]:

1) начиная с задачи приоритета 0, проверяет, не заблокирована ли она задачами, уже выполняемыми в данный момент;
2) если нет, выполняется метод shouldExecute();
3) если shouldExecute() возвращает истину, выполняется метод

startExecute();

4) повторяет п. 2 и 3 для остальных задач списка в порядке возрастания номера приоритета;

5) затем для каждой активной задачи вызывется метод continueExecuting(), если он возвращает истину, задача может быть обновлена, если ложь – задача дожна быть возвращена в начальное состояние.

На рисунке 1 данный алгоритм представлен в виде схемы.

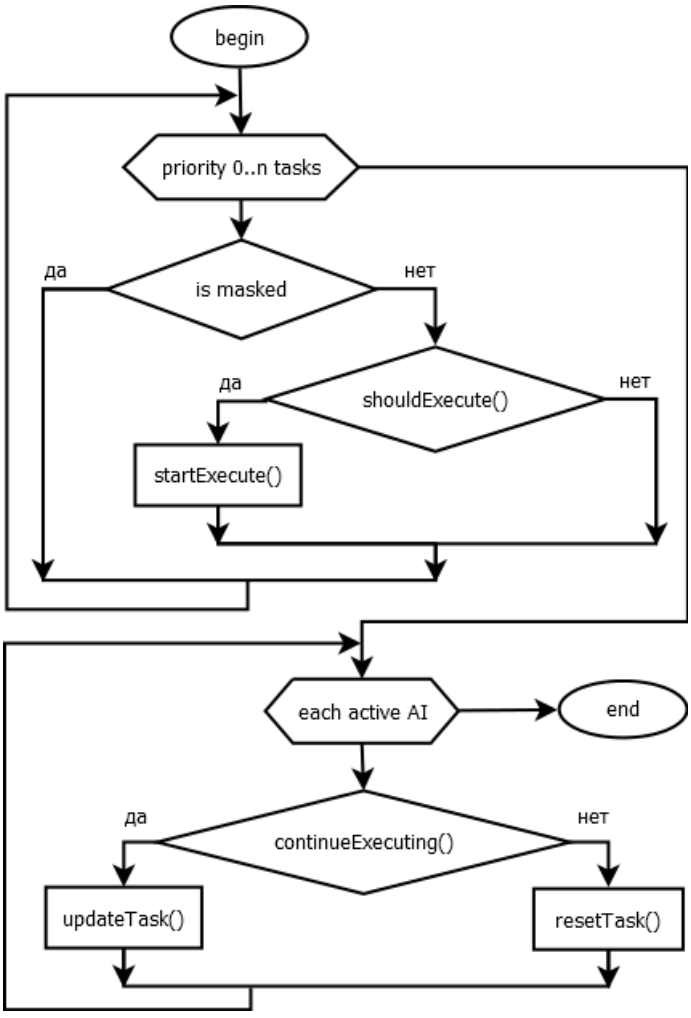

Рис. 1. Алгоритм встроенного ИИ

2. РАЗРАБОТКА МОДЕЛИ ПОВЕДЕНИЯ БОТА

Цель разрабатываемого бота – убить игрока как можно большее количество раз (способность игрового персонажа возрождаться не ограничена), оставаясь в живых. Группы ботов генерируются и бесцельно бродят в определенных локациях. Если бот обнаружит неподалеку игрока, он станет преследовать и атаковать его, пока не убьет. Боты могут атаковать только ночью, стараются избегать прямого взгляда игрока и не образовывать скоплений.

2.1. Входные данные

Классу сущностей оперирующих ИИ в Minecraft доступна информация о мире, представленная в таблице 2.

Табл. 2. Доступные данные о мире

<table>
<tr><th>Мир</th><th>Живые сущности</th></tr>
<tr><td rowspan="3">– список игровых сущностей (доступен фильтр по типу, местоположению и пр., поиск по ID / имени)
– информация о блоках, составляющих мир (расположение, состав, освещенность и пр.)
– список деревень
– локация
– время
– правила игры</td><td>– координаты
– размер
– издаваемые звуки
– направление взгляда
– крадется / бежит; скорость
– атакует / под атакой
– здоровье
– список задач</td></tr>
<tr><td>Игрок</td></tr>
<tr><td>– спит / бодрствует
– инвентарь
– возможности
– опыт (уровень)</td></tr>
</table>

В качестве входных будут использоваться следующие данные:

1) карта;
2) время суток;
3) живые сущности в поле зрения;
4) направление взгляда игрока;
5) атакует / находится ли под атакой живая сущность сейчас.

2.2. Состояния

Выделены следующие возможные состояния бота:

1) стоять;
2) блуждать;
3) искать неосвещенное место;
4) преследовать игрока незаметно;
5) атаковать игрока:

a) приближаться к игроку;

b) ударить игрока;

6) прятаться от игрока.

В таблице 3 представлены более подробные описания.

Табл. 3. Состояния бота

Состояние		Описание
Стоять		стоит на твердом блоке
Блуждать		двигается по случайной траектории, избегая других ботов
Искать неосвещенное место		ищет твердый блок с терпимым уровнем освещенности
Перемещаться незаметно для игрока	Преследовать	выдерживает заданную дистанцию между собой и игроком, стараясь находиться как можно дальше от линии взгляда
	Прятаться	
Атаковать игрока	Приближаться к игроку	приближается к игроку, стараясь распределяться равномерно с другими ботами и избегать прямого взгляда игрока
	Ударить игрока	наносит игроку урон

2.3. Параметры

Для характеристики состояния бота выделены параметры, приведенные в таблице 4. Стандартные параметры – это встроенные параметры всех живых сущностей, оперирующих ИИ в Minecraft, специальные параметры – индивидуальные параметры бота. Константные параметры представляют

неизменяемые признаки бота, модифицируемые параметры влияют на поведение бота и могут быть изменены в ходе игры, служебные параметры хранят вспомогательную информацию для разработчика.

С помощью угла атаки обозначается сектор, попадания в который бот должен избегать. Сектор ограничен двумя векторами, отложенными от точки расположения игрового персонажа и отклоняющимися от направления его взгляда на угол атаки влево и вправо.

Табл. 4. Параметры бота

Стандартные	Здоровье	
	Вероятность не быть отброшенным при атаке	
	Скорость	
	Обзор	
	Урон, причиняемый атакой	
Специальные	Константные	Терпимый уровень освещенности
		Расстояние, с которого возможно ударить игрока
		Перерыв между ударами
		Шанс повторной генерации поблизости после смерти
		Скорость восстановления здоровья ночью
	Модифицируемые	Расстояние, на котором бот преследует / прячется от игрока
		Угол атаки (угол к направлению взгляда игрока, на который бот не может приближаться при атаке)
		Дистанция обхода (расстояние, на котором бот держится, обходя игрока)
		Критический уровень здоровья
Специальные	Служебные	Состояние (для коммуникации с другими ботами)
		Частота обновления модифицируемых параметров

2.4. Модель передвижения

Когда игрок вне поля зрения, бот блуждает ночью и пережидает день в неосвещенном месте. Поле зрения – это пространство вокруг бота (круг с заданным радиусом), попадая в который игрок будет замечен. Когда бот замечает игрока, он начинает преследовать его днем и атаковать ночью. Если в ходе боя уровень здоровья падает ниже критического, бот прячется от игрока, пока здоровье не восстановится. Схема модели передвижения приведена на рисунке 2.

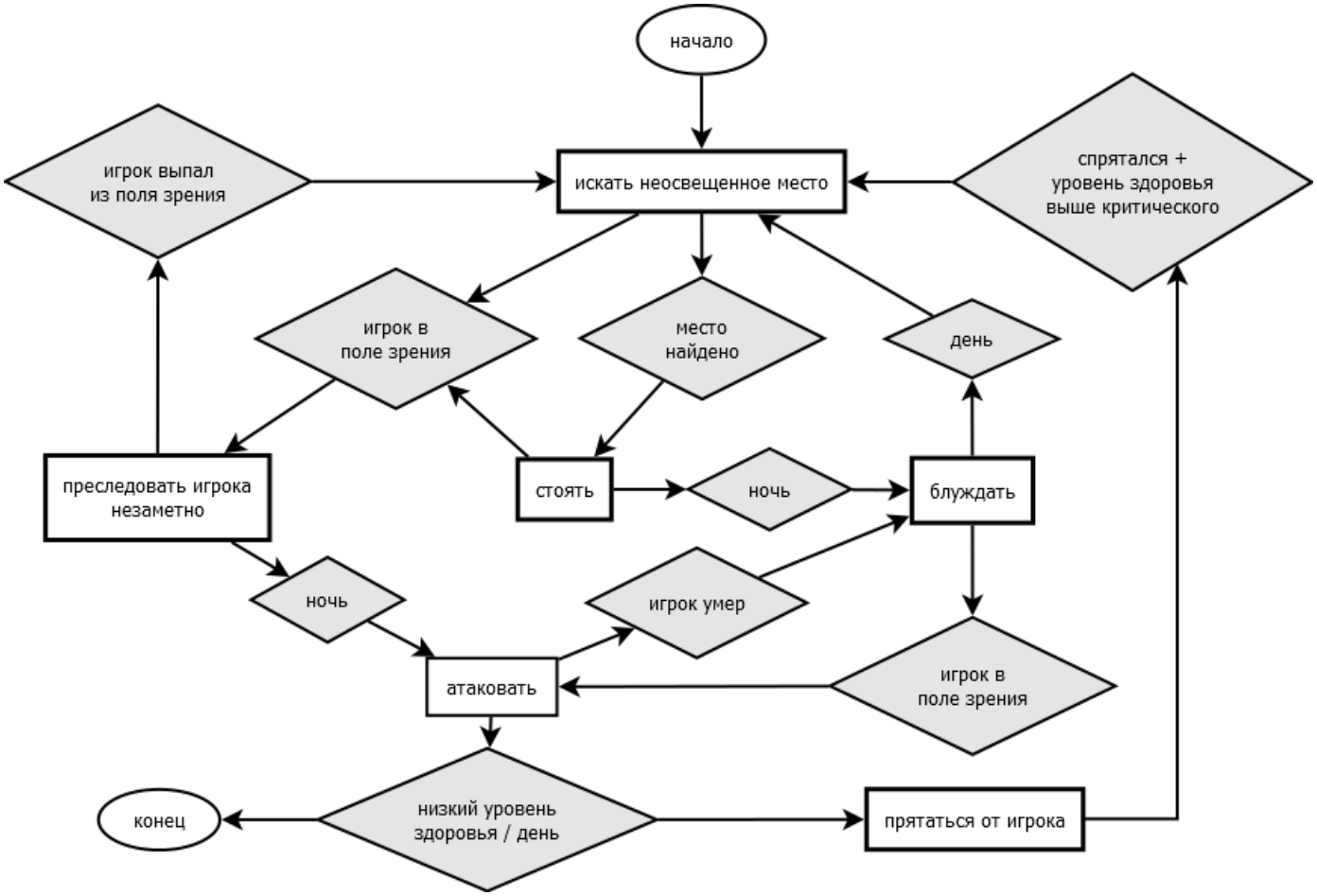

Рис. 2. Схема модели поведения бота

2.5. Модель атаки

Бот приближается к игроку, ударяет его, выжидает заданное время на заданной дистанции от игрока и повторяет сначала (схема приведена на рисунке 3).

В целом модель поведения бота можно представить в виде следующей схемы, приведенной на рисунке 1 приложения.

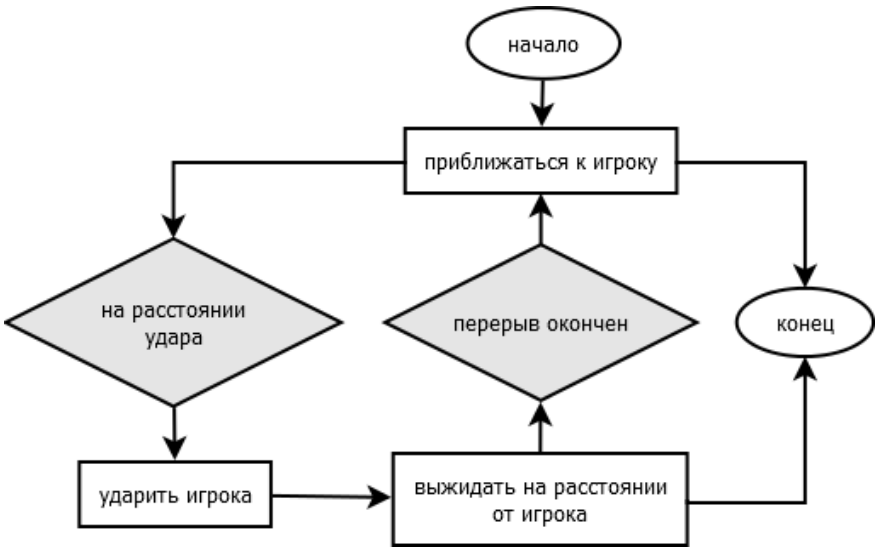

Рис. 3. Схема модели атаки

Начальное состояние в полной схеме – это момент генерации бота в точке появления, конечное состояние – смерть моба. При падении уровня здоровья ниже критического (что возможно только во время боя с игроком), бот прячется от игрока, если не погибает.

3. АДАПТАЦИЯ МОДЕЛИ ПОВЕДЕНИЯ БОТА

Чтобы обеспечить возможность адаптации бота к изменяющимся внешним условиям, модуль ИИ должен анализировать опыт бота, и на основе этой статистики изменять его поведение. Если представить модуль ИИ в виде иерархического списка задач, как в Minecraft, можно модифицировать приоритеты этих задач, изменяя алгоритм поведения, и модифицируемые параметры бота, изменяя характер выполнения конкретных задач.

Разработанную модель поведения бота можно применить как конечный алгоритм поведения, а вот подбор модифицируемых параметров вручную будет неэффективным решением. Поэтому данные параметры будут адаптивными, т.е. их значение будет оптимизироваться в ходе игрового процесса.

Оптимизация модифицируемых параметров

Для части модифицируемых параметров очевидно, какие события нужно учитывать для вычисления их оптимальных значений. Эти параметры представлены в таблице 5.

Табл. 5. Модифицируемые параметры скрытности

Состояние	Событие	Параметр	Действие
преследовать	преследуемый игрок атаковал днем	дистанция преследования	увеличить
	преследуемый игрок выпал из поля зрения		уменьшить
прятаться	бот убит в процессе восстановления здоровья	критическое здоровье	увеличить
	бот восстановил здоровье после падения ниже критического уровня		уменьшить

Для параметров атаки нельзя провести однозначной связи «событие – изменение параметра». Но можно однозначно сказать, что чем больший

урон нанес бот игроку и чем меньший урон нанес боту игрок, тем эффективнее атака бота. Если начислять некоторое количество очков за нанесение урона игроку и отнимать за урон от игрока (таблица 6), можно накапливать статистику и сравнивать эффективность бота с разными значениями параметров атаки.

Табл. 6. Модифицируемые параметры атаки

Состояние	Вознаграждение		Параметры
	+	-	
атаковать	удар по игроку убийство игрока	атака игроком убийство игроком	угол атаки
			дистанция обхода

Более подробное описание алгоритма может выглядеть следующим образом:

1) параметр p инициализируется случайным значением, задается шаг модификации параметра $\Delta p > 0$;

2) в течение некого промежутка игрового времени бот подсчитывает количество отслеживаемых событий и переводит его в баллы вознаграждения;

3) запоминается текущее значение p и набранные баллы;

4) если текущее значение p – инициализационное, Δp умножается на единицу со случайным знаком, переход в п. 6;

5) если в текущем промежутке набрано меньше баллов, чем в предыдущем, Δp умножатся на -1;

6) $p = p + \Delta p$;

7) повторять с п. 2.

Поскольку параметров два, они оптимизируются поочередно. В качестве конечного условия выбирается переход параметра в некоторое значение второй раз (то есть, ситуация, когда текущее значение p лучше, чем $p \pm \Delta p$). Когда оканчивается оптимизация одного из параметров, начинается оптимизация второго и т.д.

4. РАЗРАБОТКА ПРИЛОЖЕНИЯ

Для того чтобы воспользоваться возможностями встроенного ИИ, разрабатываемый бот должен расширять (extends) класс EntityCreature. Состояния, выделенные в модели поведения бота, будут представлены в виде EntityAI-классов и добавлены в поток задач tasks (таблица 7).

Табл. 7. Список задач tasks

Состояние	ИИ-класс	Приоритет
Прятаться от игрока	Hide	0
Преследовать игрока незаметно	Spy	1
Атаковать игрока	AttackAsBot	1
Искать неосвещенное место	WanderSearchShadow	2
Блуждать	Wander	2
Стоять	Stand	3

Функциональность классов описана в таблице 8.

Табл. 8. Классы ИИ бота

ИИ-класс	Условия выполнения	Описание
Hide	– игрок в поле зрения – уровень здоровья ниже критического	перемещается по случайной траектории в направлении от игрока
Spy	– игрок в поле зрения – сейчас день	перемещается по случайной траектории, придерживаясь заданной дистанции с игроком и направляясь ему за спину
AttackAsBot	– игрок в поле зрения – сейчас ночь	находясь по направлению взгляда игрока, пытается его обойти, иначе приближается к игроку напрямую; наносит удар, оказавшись на достаточно близком расстоянии

Окончание табл. 8

ИИ-класс	Условия выполнения	Описание
WanderSearch Shadow	– сейчас день – уровень освещенности ниже терпимого	перемещается по случайной траектории, выбирая для каждого следующего шага точку, освещенную не ярче своего текущего местоположения
Wander	сейчас ночь	перемещается по случайной траектории
Stand	сейчас день	стоит на месте

Также сформирован поток targetTasks (таблица 9).

Табл. 9. Список задач targetTasks

Описание	ИИ-класс	Приоритет
Найти игрока в зоне видимости	NearestAttackableTarget	0
Подвергнуться атаке	HurtByTarget	1

Представленные в таблице классы стандартные. Класс NearestAttackableTarget получает на вход класс сущности и отвечает за обнаружение сущностей этого класса поблизости. С помощью класса HurtByTarget можно назначать действия в случае нанесения урона заданным классом сущностей. Для данного бота выбрано действие: позвать на помощь сущностей своего класса, если они есть поблизости.

Адаптация параметров реализована в статическом методе главного класса бота в соответствии с алгоритмом, разработанным в предыдущем разделе. Сбор статистических данных организован с помощью обработчика событий, который сохраняет данные в статические переменные класса и периодически вызывает метод обновления адаптируемых параметров.

5. ТЕСТИРОВАНИЕ

Для демонстрационного тестирования разработанного механизма оптимизации были выбраны боевые параметры бота: угол атаки и дистанция обхода – так как для оптимизации атаки можно собрать большое количество тестовых данных за короткое время.

В игровом пространстве была огорожена арена, на которой проводилась серия сражений игрока с ботами. В ходе каждого сражения игрок должен был убить 5 ботов, используя одно и то же оружие и начиная с одним и тем же уровнем здоровья. В случае смерти игрок возрождался на этой же арене с максимальным здоровьем и продолжал бой. По результатам сражения угол атаки и дистанция обхода модифицировались в соответствие с собранной статистикой.

В качестве шагов модификации для оптимизируемых параметров использовались значения, представленные в таблице 10.

Табл. 10. Шаги модификации параметров

Параметр	Шаг
Угол атаки	10°
Дистанция обхода	2

Баллы вознаграждения за действия в ходе сражения начислялись в соответствии с таблицей 11.

Табл. 11. Баллы вознаграждения

Событие	Баллы
Удар по игроку	2
Убийство игрока	5
Пропущенный удар	-1

Результаты тестирования представлены в таблицах 12 и 13.

Выявленные оптимальные значения для угла атаки и дистанции обхода первого игрока: (55; 12). Стартовые значения – случайные числа.

Табл. 12. Результаты тестирования для первого игрока

№	1	2	3	4	5
Нанесенных ударов	14	20	17	19	16
Убитых игроков	1	2	1	2	1
Полученных ударов	48	52	50	47	43
Вознаграждение	-15	-2	-11	1	-6
Угол атаки	45	55	65	55	55
Дистанция обхода	10	10	10	12	14

Табл. 13. Результаты тестирования для второго игрока

№	1	2	3	4	5	6
Нанесенных ударов	18	24	23	3	15	8
Убитых игроков	2	2	2	0	1	1
Полученных ударов	74	83	79	88	75	70
Вознаграждение	-28	-25	-23	-82	-40	-49
Угол атаки	55	65	75	85	75	75
Дистанция обхода	12	12	12	12	14	10

В качестве стартовых значений для следующего игрока использовались выявленные оптимальные значения первого. В итоге для второго игрока оптимальными значениями угла атаки и дистанции обхода оказались: (75; 12).

ЗАКЛЮЧЕНИЕ

Целью работы являлась разработка адаптивного бота для игры Minecraft. В рамках работы были решены все поставленные задачи и достигнуты следующие результаты:

1) изучен программный интерфейс Minecraft Forge;
2) разработана адаптивная модель поведения бота;
3) реализовано приложение в соответствии с разработанной моделью;
4) протестированы адаптивные возможности бота.

ЛИТЕРАТУРА

1. Artificial Intelligence for Computer Games. [Электронный ресурс] URL: https://www.hindawi.com/journals/ijcgt/2009/251652/ (дата обращения: 10.04.2017).

2. Adobbati R. Gamebots: A 3D Virtual World Test Bed For Multi Agent Research. R. Adobatti, G. Kaminka, A. Marshall, et al. // Journal of Multimedia, 2007. – Vol. 2. – P. 48–53.

3. Bartle R. Designing Virtual Worlds. – USA: New Riders, 2003. – 768 p.

4. Marchand A. 7 Multiplayer Features and Game Success in Kowert R., Quandt T. New Perspectives on the Social Aspects of Digital Gaming: Multiplayer 2. – USA: Routledge, 2017. – 210 p.

5. Mark D. Behavioral Mathematics for Game AI. – USA: Cengage Learning PTR, 2009. – 480 p.

6. McCarthy J. What is artificial intelligence? [Электронный ресурс] URL: http://www-formal.stanford.edu/jmc/whatisai/whatisai.html (дата обращения: 26.04.2017).

7. Minecraft Modding: Custom Entity AI. [Электронный ресурс] URL: http://jabelarminecraft.blogspot.ru/p/minecraft-forge-1721710-custom-entity-ai.html (дата обращения: 16.02.2017).

8. Mob. [Электронный ресурс] URL: http://minecraft.gamepedia.com/Mob (дата обращения: 17.05.2017).

9. Most played PC games. [Электронный ресурс] URL: https://www.statista.com/statistics/251222/most-played-pc-games/ (дата обращения: 26.04.2017).

10. S.T.A.L.K.E.R.: Чистое небо – интервью о проблемах выживания искусственного интеллекта в Чернобыльской Зоне. [Электронный ресурс] URL: http://www.gametech.ru/articles/85/ (дата обращения: 10.04.2017).

11. Sams Teach Yourself Mod Development for Minecraft. – USA: Pearson Education, Inc., 2015 – 448 p.

12. Scott B. The Illusion of Intelligence in Rabin S. AI Game Programming Wisdom. – USA: Charles River Media, 2002. – 704 p.

13. The Minecraft Generation. [Электронный ресурс] URL: https://www.nytimes.com/2016/04/17/magazine/the-minecraft-generation.html?_r=0 (дата обращения: 10.04.2017).

14. The past, present and future of artificial neural networks in digital games. [Электронный ресурс] URL: https://www.researchgate.net/publication/229034750 (дата обращения: 26.04.2017).

15. Документация по Minecraft Forge. [Электронный ресурс] URL: https://mcforge.readthedocs.io (дата обращения: 17.05.2017).

ПРИЛОЖЕНИЕ

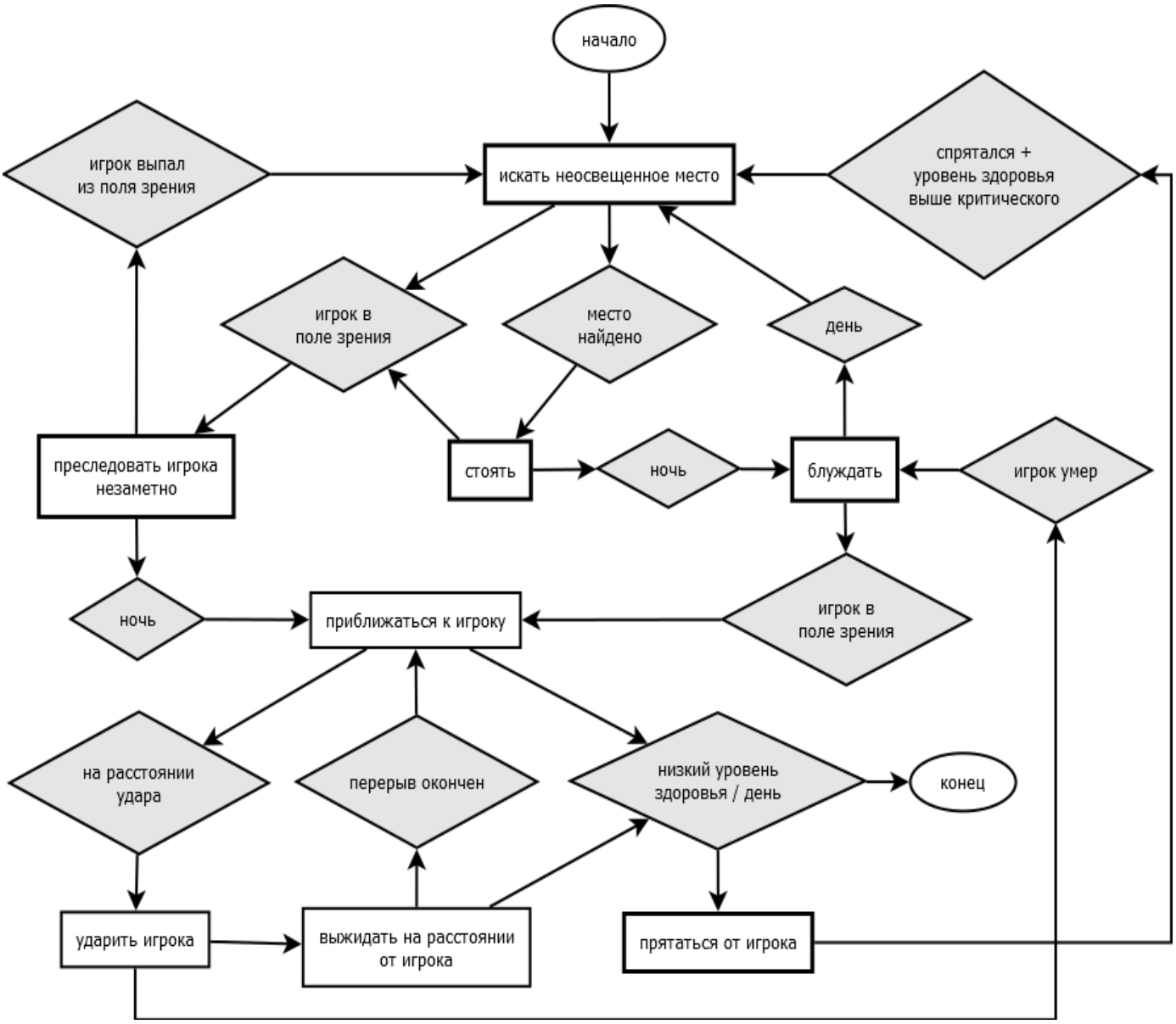

Рис. 1. Схема полной модели поведения бота