МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»
Кафедра математической кибернетики и компьютерных наук

МОДЕЛИРОВАНИЕ И ВИЗУАЛИЗАЦИЯ ГЕНЕТИЧЕСКИХ АЛГОРИТМОВ В СРЕДЕ UNITY

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 4 курса 451 группы
направления 09.03.04 — Программная инженерия
факультета КНиИТ
Михайлина Дмитрия Константиновича

Научный руководитель
к. ф.-м. н., доцент _____ А. С. Иванов

Заведующий кафедрой
к. ф.-м. н., доцент _____ С. В. Миронов

Саратов 2025

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В последние десятилетия развитие вычислительных методов и искусственного интеллекта привело к появлению эффективных алгоритмов, способных решать сложные задачи оптимизации и адаптации. Одним из таких методов являются генетические алгоритмы — вычислительные модели, основанные на принципах естественного отбора и наследования в биологии.

Целью данной работы является исследование и визуализация процесса эволюции виртуальных существ, обученных с использованием генетических алгоритмов. В качестве модельного существа был выбран многоногий агент с задачей — научиться эффективно передвигаться в трёхмерной физической среде.

Для достижения поставленной цели были сформулированы следующие ключевые задачи:

- разработать физическую модель виртуального существа, взаимодействующую с окружающей средой;
- реализовать систему управления на основе нейронной сети с параметрами, которые настраиваются генетическим алгоритмом;
- внедрить алгоритм для эволюции популяции существ с операторами отбора, скрещивания и мутации;
- определить функцию приспособленности, отражающую успешность выполнения задачи передвижения;
- провести эксперименты для анализа эволюционного обучения;
- визуализировать процесс эволюции и результаты с использованием Unity;
- проанализировать полученные данные и сделать выводы о применимости генетических алгоритмов для адаптивного управления.

Структура и объём работы. Для решения поставленных задач выполнена выпускная квалификационная работа, включающая в себя введение, 2 основные главы, заключение, список использованных источников из 20 наименований и 4 приложения. Работа изложена на 49 страницах. Первая глава имеет название «Теоретические основы и инструментарий» и содержит обзор генетических алгоритмов и инструментов разработки (Unity, C#). Вторая глава имеет название «Практическая реализация и экспериментальное исследование эволюции существа»: она содержит описание физической модели, архитектуры системы и результаты экспериментов. Работа заканчивается заключением, списком использованных источников, а также приложениями А-Г.

1 Теоретические основы и инструментарий

1.1 Генетические алгоритмы как метод эволюционных вычислений

Эволюционные вычисления представляют собой направление информатики, основанное на принципах биологической эволюции. Генетические алгоритмы (ГА), теоретический фундамент которых заложил Джон Холланд в работе «Адаптация в естественных и искусственных системах» (1975), являются эвристическим методом поиска и оптимизации, использующим механизмы естественной эволюции.

ГА работают с популяцией особей, где каждая особь представляет закодированное решение задачи. Функционирование основано на трех биологических принципах: приспособленности и отборе (селекции лучших решений через функцию приспособленности), наследовании и скрещивании (создание потомков путем обмена генетическим материалом), изменчивости и мутации (внесение случайных изменений для поддержания разнообразия).

Ключевые компоненты ГА включают генотип (внутреннее представление решения) и фенотип (внешнее проявление после декодирования). Методы кодирования варьируются от бинарного до вещественного, где параметры представляются действительными числами.

Классический ГА работает итеративно через поколения: создание начальной популяции, оценка приспособленности, отбор родителей (рулеточный, турнирный, ранжированный отбор), применение операторов скрещивания и мутации, формирование нового поколения. Процесс продолжается до выполнения критерия остановки.

Эффективность ГА определяется настройкой параметров: размером популяции (обычно 50-500 особей), вероятностями скрещивания (0.6-0.95) и мутации (0.001-0.05). Ключевая проблема — преждевременная сходимость к локальному оптимуму при потере генетического разнообразия.

Нейроэволюция объединяет ГА с искусственными нейронными сетями для автоматической оптимизации весовых коэффициентов и/или архитектуры сети. Этот подход эффективен в задачах обучения с подкреплением, где агент вырабатывает стратегию поведения через взаимодействие со средой.

ГА применяются в оптимизации функций, машинном обучении, планировании, робототехнике, биоинформатике и игровой индустрии. Преимущества включают глобальный поиск, устойчивость к шуму, параллелизм и отсут-

ствие необходимости в градиентной информации. Ограничения: вычислительная сложность, возможная преждевременная сходимость, сложность проектирования функции приспособленности и стохастичность результатов.

1.2 Инструментарий: Среда Unity для разработки и симуляции

Для практической реализации поставленных задач была выбрана среда разработки Unity благодаря ее возможностям для создания интерактивных 3D-приложений, мощному физическому движку и системе скриптования на языке C#.

1.2.1 Обзор среды Unity

Unity представляет собой кроссплатформенный игровой движок и интегрированную среду разработки для создания видеоигр, интерактивных симуляций и 3D-приложений. Ключевые особенности Unity для данной работы включают:

Встроенный физический движок PhysX обеспечивает реалистичную симуляцию физических взаимодействий объектов, включая гравитацию, столкновения, силы и моменты инерции, что критически важно для моделирования движения виртуальных существ.

Компонентно-ориентированная архитектура позволяет строить игровые объекты путем добавления различных компонентов, отвечающих за физические свойства (`Rigidbody`, `Collider`), визуальное представление или логику поведения.

Визуальный редактор сцены предоставляет интуитивный интерфейс для создания и компоновки 3D-сцен, настройки свойств объектов и их взаимосвязей.

Поддержка скриптов на C# позволяет реализовывать сложные алгоритмы, включая генетические алгоритмы и управление нейронными сетями.

Система префабов обеспечивает создание шаблонов объектов для многократного использования, что особенно удобно для создания популяций существ в контексте генетических алгоритмов.

1.2.2 Язык C# и скриптинг в Unity

Язык C# является основным для написания скриптов в Unity. Скрипты наследуются от базового класса `MonoBehaviour` и определяют поведение игровых

объектов через встроенные методы жизненного цикла:

— `Awake()` — вызывается при загрузке экземпляра скрипта
— `Start()` — вызывается перед первым обновлением кадра
— `Update()` — вызывается каждый кадр для общей логики
— `FixedUpdate()` — вызывается через фиксированные промежутки для физической логики

1.2.3 Компоненты для физической симуляции

Для симуляции физического поведения существ используются ключевые компоненты Unity:

Rigidbody придает объекту физические свойства (масса, инерция) и позволяет управлять движением через применение сил и моментов.

Collider определяет физическую форму объекта для обнаружения столкновений. Основные типы: `BoxCollider` (параллелепипед), `SphereCollider` (сфера), `CapsuleCollider` (капсула), `MeshCollider` (повторяет форму объекта).

Physic Material определяет свойства поверхностей при столкновении, такие как трение и упругость.

Joints создают сочленения между частями существа:

— `Hinge Joint` — шарнирное соединение с вращением вокруг одной оси, настраиваемыми пределами, пружиной и мотором
— `Fixed Joint` — жесткая фиксация двух объектов

Система шаблонов обеспечивает эффективное управление популяцией существ: быстрое создание множества экземпляров, централизованное изменение шаблонов и упрощенную организацию проекта. Управляющий скрипт координирует инициализацию популяции, запуск итераций генетического алгоритма, оценку приспособленности и создание новых поколений.

2 Практическая реализация и экспериментальное исследование эволюции существа

Практическая часть работы посвящена разработке системы эволюционного обучения виртуальных существ с использованием генетических алгоритмов. Основным объектом исследования является модель многоногого существа в трехмерной среде Unity, способного к автономному передвижению через настройку параметров управляющей нейронной сети.

2.1 Конструкция физической модели существа

2.1.1 Базовые компоненты

Разработано четыре базовых типа префабов для конструирования существа:

SpiderBody — тело существа на основе модифицированной сферы, сплющенной по вертикальной оси. Включает компоненты `Rigidbody` (масса 7 единиц) и `BoxCollider` с физическим материалом среднего трения.

SpiderLegSegment — сегменты конечностей в виде цилиндров с `Rigidbody` (масса 1.0 единица). Соединяются с суставными элементами через `Fixed Joint`.

SpiderConnect — суставные элементы в виде сфер с `Rigidbody` (масса 0.5 единиц) и `HingeJoint` для обеспечения подвижности. Управление движением осуществляется через параметры встроенной пружины.

SpiderFoot — стопы конечностей в виде плоских параллелепипедов с `Rigidbody` (масса 0.2 единицы) и `BoxCollider` с максимальными значениями трения для надежного сцепления с поверхностью.

2.1.2 Сборка существа

Из базовых элементов сконструирована модель конечности, состоящая из четырех активных шарнирных сочленений и трех основных сегментов. Каждая нога обеспечивает комбинированные горизонтальные и вертикальные движения. Первая версия существа (V1) включает восемь конечностей, прикрепленных к центральному телу.

2.2 Архитектура программных компонентов

2.2.1 Базовый класс BaseCreature

Разработан абстрактный класс `BaseCreature`, наследник `MonoBehaviour`, который определяет общую функциональность для всех типов существ. Класс

содержит:

- Ссылку на нейронную сеть (`brain`)
- Свойства приспособленности (`Fitness`) и активности (`IsAlive`)
- Массивы сенсорных входов и выходов сети
- Список частей тела с компонентами `Rigidbody`

Основной цикл обновления включает: сбор сенсорных данных, передачу в нейронную сеть, извлечение выходных сигналов, применение управляющих воздействий и расчет приспособленности.

2.2.2 Класс SpiderCreature

Класс `SpiderCreature` наследует от `BaseCreature` и реализует специфическую логику для многоногого существа:

Сенсорная система включает текущие углы суставов, информацию о контакте конечностей с землей, локальную скорость и высоту над стартовой позицией.

Управление суставами осуществляется через изменение целевого положения пружин `HingeJoint`. Выходные сигналы нейронной сети преобразуются в целевые углы для каждого сустава.

Функция приспособленности рассчитывается как пройденное расстояние вдоль оси X с применением штрафов за нестабильность ориентации и боковые отклонения от целевой траектории.

2.2.3 Управление эволюцией

Класс `EvolutionManager` координирует полный жизненный цикл популяции:

Параметры эволюции: размер популяции, время жизни поколения, количество элитных особей, вероятности скрещивания и мутации, сила мутации.

Эволюционный цикл: создание начальной популяции со случайными нейронными сетями, оценка приспособленности, применение генетических операторов (элитизм, усеченный отбор, равномерное скрещивание, адаптивная мутация), создание нового поколения.

Логирование результатов: автоматическая запись статистики каждого поколения в файл с временными метками для последующего анализа эффективности эволюционного процесса.

Система обеспечивает полную автоматизацию процесса эволюционного обучения с возможностью настройки всех ключевых параметров генетического алгоритма через интерфейс Unity.

2.2.4 Реализация нейронной сети и генетических операторов

Для управления поведением виртуальных существ разработан класс `NN`, реализующий многослойный перцептрон с фиксированной архитектурой из четырех слоев: входной, два скрытых и выходной. Каждый слой содержит одинаковое количество нейронов, определяемое параметром `width`.

Структура нейронной сети включает:

- `width` — общее количество нейронов в каждом слое
- `reserved` — количество нейронов для встроенных осцилляторов, генерирующих ритмические сигналы
- `neurons[,]` — двумерный массив активаций нейронов
- `bias[,]` — массив смещений для скрытых и выходного слоев
- `weights[,,]` — трехмерный массив весовых коэффициентов
- `octaves[]` — массив частот осцилляторов

Инициализация сети происходит в конструкторе `NN(int totalNetworkWidth,` со случайными значениями весов и смещений в диапазоне [-1, 1]. Частоты осцилляторов инициализируются в предопределенном диапазоне `[octavesMin, octavesMax`

Прямое распространение реализовано в методе `Update()`:

- Обновление осцилляторов через синусоидальные функции `Mathf.Sin(Time.time`
- Получение сенсорных входов от существа
- Расчет активаций по слоям через взвешенную сумму с добавлением смещений
- Применение функции активации `Mathf.Clamp(sum, -1f, 1f)`

Генетические операторы:

Копирование реализовано через конструктор `NN(NN source)` для механизма элитизма, обеспечивающего перенос лучших особей в следующее поколение.

Скрещивание выполняется конструктором `NN(NN parentA, NN parentB, floa` с использованием равномерного кроссовера. Для каждого параметра случайно выбирается родитель: с вероятностью `crossoverRate` значение наследуется от `parentA`, иначе — от `parentB`.

Мутация реализована в методе `Mutate(float chance, float rate)`:

— С вероятностью `chance` каждый ген подвергается мутации
— К текущему значению добавляется случайное число из диапазона `[-rate, rate]`
— Результат приводится к допустимому диапазону через `Mathf.Clamp`

Геном особи представлен всеми настраиваемыми параметрами нейронной сети: весами, смещениями и частотами осцилляторов, которые эволюционируют совместно для оптимизации поведения существа.

2.3 Экспериментальные исследования различных версий существа

Для проведения экспериментов была создана сцена с поверхностью для взаимодействия, стартовой линией и управляющим объектом `Main` с компонентом `Evolution Manager`. Это позволило настраивать гиперпараметры эволюции через интерфейс Unity и автоматизировать процесс эволюционного обучения.

2.3.1 Эксперимент с версией V1

Полная модель с восемью конечностями тестировалась с популяцией 40 особей в течение 1200 поколений. Результаты показали активную фазу эволюции в первые 200 поколений со стремительным ростом приспособленности. После 800-го поколения рост замедлился, достигнув локального минимума. **Максимальная дистанция составила 176 метров при средней по популяции 115 метров**.

Анализ поведения выявил проблему взаимного пересечения конечностей: особи передвигались полубоком, некоторые ноги скрещивались между собой, затрудняя эффективное передвижение. Это ограничивало потенциал развития популяции.

2.3.2 Эксперимент с версией V2

Упрощенная модель с четырьмя конечностями устранила проблему пересечений. Популяция увеличена до 60 особей. В отличие от V1, популяция показывала устойчивый рост на протяжении всей эволюции с периодическими плато. **Максимальная дистанция достигла 281 метра при средней 175 метров**.

Сформировался эффективный паттерн движения: существо использовало диагонально расположенные конечности как весла, в то время как остальные выполняли толкающие и подтягивающие движения. Отсутствие центральных конечностей исключило взаимные помехи при передвижении.

2.3.3 Эксперимент с версией V3

Модель с укороченными конечностями и ограниченными углами вращения суставов позволила увеличить популяцию до 126 особей. Упрощение конструкции обеспечило стабильную стойку без касания тела поверхности.

Максимальная дистанция составила 142 метра при средней 79 метров. Существа демонстрировали более четкий паттерн ходьбы с диагональным движением конечностей, но ограничения мобильности суставов быстро привели к достижению предела развития.

2.3.4 Сравнительный анализ и выводы

Сравнение результатов показало, что **наиболее эффективной оказалась версия V2**, демонстрирующая устойчивый рост приспособленности без достижения локальных оптимумов. В отличие от V1, она не имела проблем со скрещиванием конечностей, а в отличие от V3 — не была ограничена в мобильности суставов.

Версия V3 показала наиболее стабильные и предсказуемые результаты, но быстро достигла предела из-за конструктивных ограничений модели.

2.3.5 Исследование влияния параметров мутации

Дополнительные эксперименты с версией V3 исследовали влияние параметров мутации на эволюционный процесс:

Сильная мутация (вероятность 0.5, сила 1.0) препятствовала сохранению полезных признаков, разрушая устойчивые паттерны движения. Эксперимент остановлен на 600-м поколении из-за отсутствия прогресса.

Отсутствие мутации привело к быстрому достижению предела развития около 300-го поколения. Популяция развивалась только на основе стартовых генов, что ограничило потенциал роста.

Оптимальные параметры (вероятность мутации 0.03, сила 0.1) обеспечили баланс между сохранением полезных признаков и поддержанием генетического разнообразия, что позволило популяции продолжать медленное, но устойчивое развитие.

Исследование подтвердило критическую важность правильной настройки параметров генетического алгоритма для эффективного эволюционного обучения виртуальных агентов.

ЗАКЛЮЧЕНИЕ

Выполненная выпускная квалификационная работа была посвящена исследованию и практической реализации эволюционного обучения виртуальных существ на основе генетических алгоритмов с визуализацией в среде Unity. В ходе работы был осуществлён полный цикл проектирования, программной реализации, тестирования и анализа модели, демонстрирующей возможности генетических алгоритмов для решения задач адаптивного управления в симулированных физических средах.

К ключевым практическим достижениям работы относятся:

1. Разработка физической модели виртуального существа, основанной на простых геометрических примитивах и сочленениях, способной к взаимодействию с трёхмерной физической средой Unity.
2. Реализация системы управления на основе нейронной сети, параметры которой формируются генетическим алгоритмом, что обеспечивает адаптивное поведение и возможность эволюционного развития.
3. Внедрение классического генетического алгоритма с поддержкой методов отбора, скрещивания и мутации, а также гибкой настройки параметров эволюции.
4. Определение и внедрение функции приспособленности, адекватно оценивающей успешность выполнения целевой задачи (перемещения на максимальное расстояние).
5. Проведение серии вычислительных экспериментов, позволивших проанализировать динамику обучения и влияние различных параметров на эффективность эволюционного процесса, такие как мутация или конструкция существа.
6. Визуализация процесса эволюции и результатов обучения с использованием средств Unity, включая графическое отображение поведения существ и построение графиков изменения приспособленности по поколениям.

Поставленные цели и задачи были полностью достигнуты. Разработанная система может быть расширена для решения более сложных задач, интеграции с другими моделями искусственного интеллекта и дальнейшего совершенствования методов эволюционного обучения.