**ФГБОУВО «Пермский государственный национальный исследовательский университет»**

**Физический факультет**

**Кафедра компьютерных систем и телекоммуникаций**

Специальность «Радиофизика и электроника»

Специализация «Телекоммуникационные системы и

информационные технологии»

Дипломная работа

«Моделирование эволюции игровых объектов с помощью генетического алгоритма при разработке игрового искусственного интеллекта, способного адаптироваться к актуальной игровой ситуации»

Работу выполнил

**студент**

**Баландин Артём Дмитриевич**

***Подпись\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

Научный руководитель:

**Кандидат технических наук,**

**доцент кафедры КСиТ,**

**Скляренко М.С**

***Подпись\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

Допустить к защите

Зав. кафедрой, профессор, доктор

физико-математических наук

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

М.А.Марценюк

« » июня 2017 г.

**Оглавление**

[1. Аннотация 4](#_Toc12436138)

[2. Введение 5](#_Toc12436139)

[3. Постановка задачи 6](#_Toc12436140)

[4. Обзор существующих решений рассматриваемой задачи или ее модификаций 7](#_Toc12436141)

[4.1 Обзор BoxCar2D 7](#_Toc12436142)

[4.2 Обзор Genetic Cars 8](#_Toc12436143)

[5. Исследование и построение решения задачи 11](#_Toc12436144)

[6. Описание практической части 12](#_Toc12436145)

[6.1 Разработка генетического алгоритма 12](#_Toc12436146)

[6.1.1 Изучение принципов построения генетического алгоритма 12](#_Toc12436147)

[6.1.2 Изучение оптимизации математических задач с помощью генетического алгоритма 14](#_Toc12436148)

[6.1.3 Адаптация генетического алгоритма для нахождения экстремума функции и тестирование полученного алгоритма на известных функциях. 15](#_Toc12436149)

[6.2 Разработка игры на игровом движке 18](#_Toc12436150)

[6.2.1 Выбор подходящего игрового движка 18](#_Toc12436151)

[6.2.2 Создание префаба модели-машинки 19](#_Toc12436152)

[6.2.3 Создание префаба дороги 20](#_Toc12436153)

[6.2.4 Создание трассы 21](#_Toc12436154)

[6.2.5 Разработка контроллера модели-машинки со свойствами модели 21](#_Toc12436155)

[6.2.6 Разработка контроллера движения модели-машинки 22](#_Toc12436156)

[6.2.7 Разработка контроллера генерации объектов 22](#_Toc12436157)

[6.2.8 Тестирование игры 22](#_Toc12436158)

[7. Заключение. Полученные результаты 23](#_Toc12436159)

[8. Список цитируемой литературы 24](#_Toc12436160)

# Аннотация

# Введение

# Постановка задачи

Основной задачей является разработать приложение, которое наглядно продемонстрирует эволюцию игровых объектов к меняющейся среде, на основе генетических алгоритмов с использованием игрового движка UNITY.

# Обзор существующих решений рассматриваемой задачи или ее модификаций

На данный момент существуют несколько решений, которые представляют из себя игры, написанные на языке Java Script.

## Обзор BoxCar2D

BoxCar2D представляет из себя игру, использующую Flash и открытый игровой физический движок Box2D. Игра использует генетический алгоритм для создания модели машинки и последующего развития этой модели.

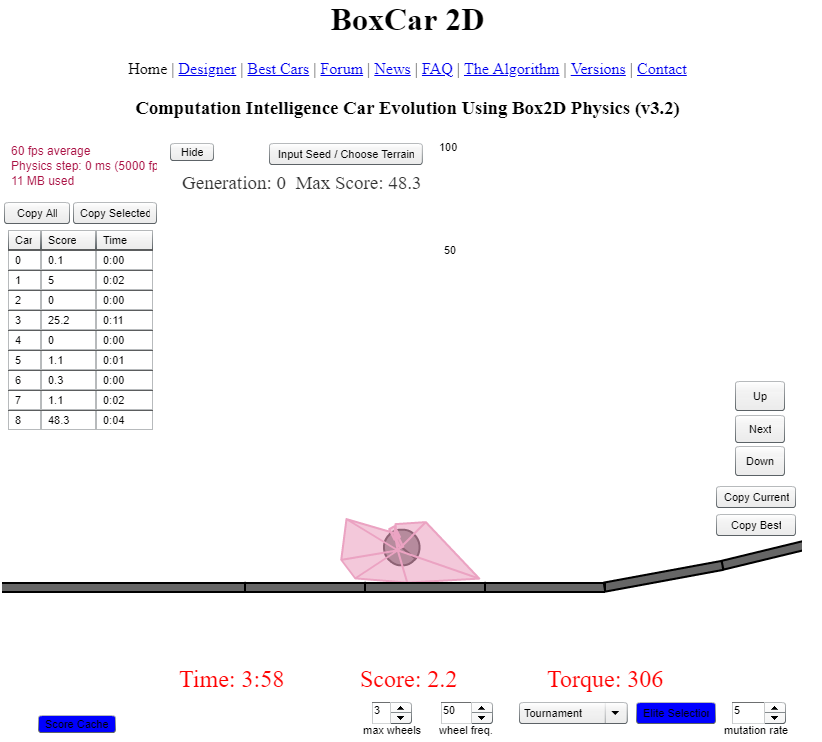
Человеку даётся возможность выбрать:

* максимальное количество колёс у машинки (max wheels);
* скорость, с которой будут крутиться колёса (wheel freq.);
* тип отбора (Tournament, Roulette Wheel);
* отбор «элитных» родителей (Elite Selection);
* степень мутации (mutation rate).

Также интерфейс отображает следующие элементы:

* таймер жизни особи (Time);
* счёт, который зависит от расстояния, которое прошла машинка (Score);
* крутящий момент (Torque);
* таблица с поколениями машинки, её счётом и временем жизни.

Кнопки управления отвечают за выбор особи, переход к следующему поколению, копированию лучшей особи или копированию текущей особи.



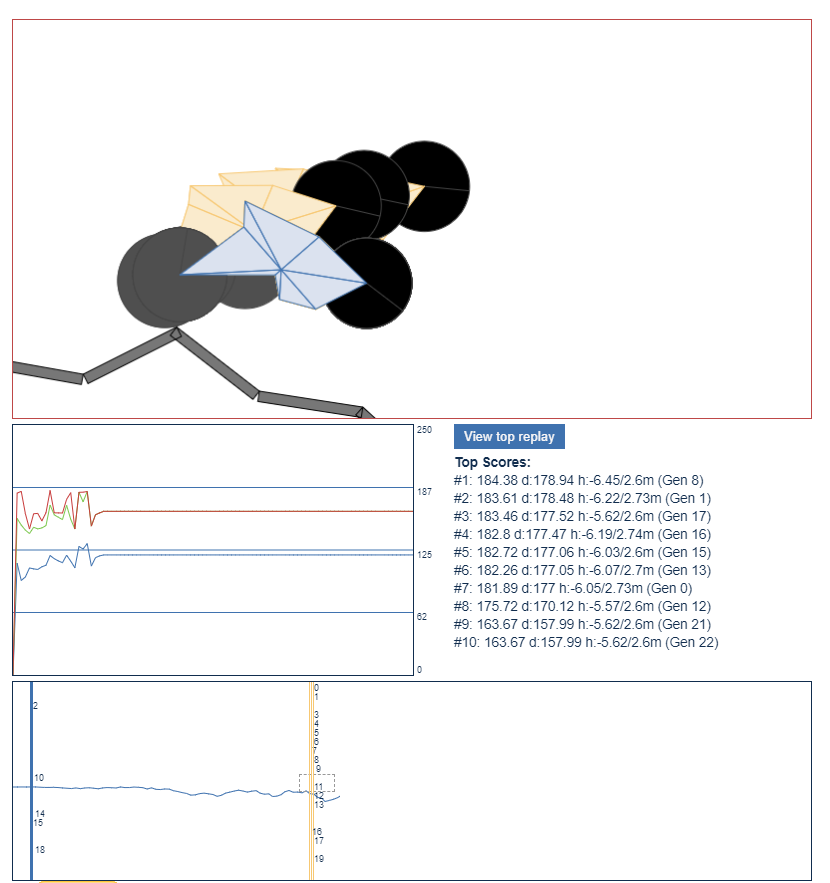
## Обзор Genetic Cars

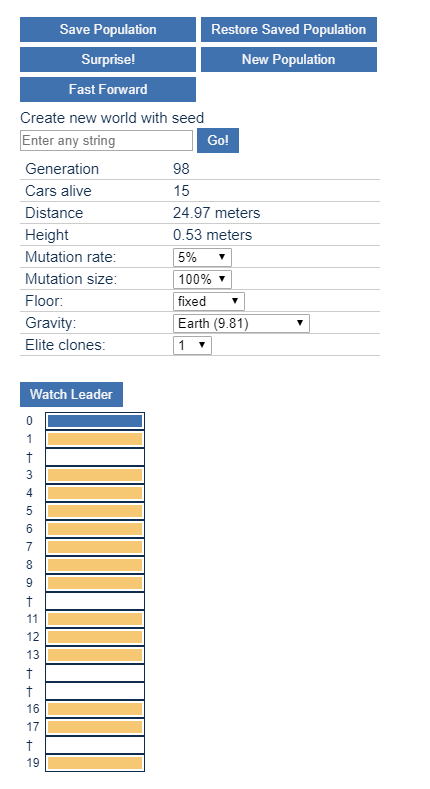
Genetic Cars игра написана с помощью HTML5 и JavaScript, использует физический движок box2d. Genetic Cars наследник BoxCar2D, но имеет ряд отличий:

* можно наблюдать за всем поколением машинок, а не только за одной особью в поколении;
* более детальная информация о поколении:
  + номер поколения;
  + число живых машинок;
  + дистанция, которую проехал «лидер» поколения;
  + графики передвижения по трассе с местонахождением каждой машинки.
* добавлена возможность сделать изменяющуюся трассу, выбор гравитации.

Осталась возможность выбора «элитных» родителей, вероятность мутации.

Кнопки управления позволяют сохранять поколения, восстановление сохраненного поколения, создание нового поколения, увеличение скорости работы алгоритма.





# Исследование и построение решения задачи

Для решения задачи необходимо разбить её на подзадачи:

1. Разработка генетического алгоритма
   1. Изучение принципов построения генетического алгоритма;
   2. Изучение оптимизации математических задач с помощью генетического алгоритма;
   3. Адаптация генетического алгоритма для нахождения экстремума функции и тестирование полученного алгоритма на известных функциях.
2. Разработка игры на игровом движке
   1. Выбор подходящего игрового движка;
   2. Создание префаба модели-машинки;
   3. Создание префаба дороги;
   4. Создание трассы
   5. Разработка контроллера модели-машинки со свойствами модели;
   6. Разработка контроллера движения модели-машинки;
   7. Разработка контроллера генерации объектов;
   8. Тестирование игры.

# Описание практической части

## Разработка генетического алгоритма

### Изучение принципов построения генетического алгоритма

Генетический алгоритм (ГА) был предложен Джоном Холландом в 1975 году и является первым из эволюционных алгоритмов. Особи в генетическом алгоритме представляются в виде строк фиксированной длины.

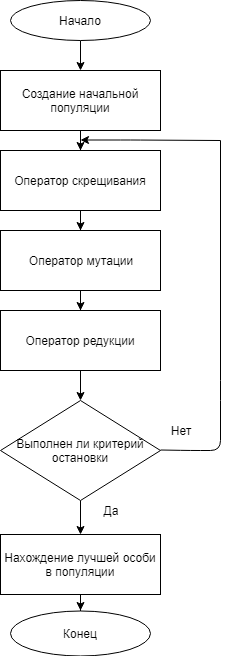
В теории генетических алгоритмов применяется следующая терминология:

* Ген (свойство) – атомарный элемент хромосомы. Ген может быть битом, числом или неким другим объектом.
* Аллель – значение конкретного гена.
* Локус – положение конкретного гена в хромосоме.
* Хромосома (цепочка) – упорядоченная последовательность генов.
* Генотип (код) – упорядоченная последовательность хромосом.
* Особь (индивидуум) – конкретный экземпляр генотипа.
* Фенотип – аргумент (набор аргументов) целевой функции, соответствующий генотипу (т.е. интерпретация генотипа с точки зрения решаемой задачи).
* Популяция – совокупность особей, используемых в итерации.
* Размер популяции – количество особей в популяции.

Генетический алгоритм содержит 4 главных операции:

* Инициализация – создание случайной начальной популяции. Создание особи может быть с помощью заранее придуманного алгоритма или просто случайным образом.
* Отбор (оператор редукции) – выбор определенного количество особей с помощью методов отбора. Популярные методы отбора: метод рулетки, ранговый метод, элитизм и турнирный метод.
* Кроссовер (оператор скрещивания) – создание новой особи на основе параметров отобранных особей. Существует одноточечный, двухточечный и однородный кроссовер.
* Мутации (оператор мутации) – замена случайным значением гена случайно выбранной особи.

Последовательность работы генетического алгоритма.



Критерием остановки могут быть следующие события:

* сформировано заданной число поколений;
* популяция достигла заданного качества
* улучшение особей происходит очень медленно, достигнут уровень сходимости;
* закончилось время, отведенное на эволюцию.

### Изучение оптимизации математических задач с помощью генетического алгоритма

Задачей оптимизации в математике называется задача нахождения экстремума целевой функции (минимума или максимума) в некоторой области конечного пространства, ограниченной набором линейных/нелинейных равенств/неравенств или их комбинации.

Обычно ставится цель найти наилучшую структуру или значение параметров объектов. Такая задача называется оптимизационной. Если оптимизация связана с нахождением оптимальных значений параметров, то такая оптимизация называется параметрической оптимизацией, Если оптимизация связана с нахождением оптимальной структуры, то называется структурной оптимизацией.

Математическая задача оптимизации формулируется таким образом, что среди элементов *x*, образующих множество *X*, нужно найти такой элемент *x\**, с помощью которого получим минимальное значение *f(x\*)* заданной функции *f(x).*

Чтобы корректно поставить задачу оптимизации, необходимо задать

* допустимое множество X;
* целевую функцию;
* критерий поиска.

Сравним методы нахождения экстремума функции с одной переменной. Задана целевая функция: . Требуется найти максимальное и минимальное значение целевой функции в интервале .

Рассмотрим решение с помощью классической теории оптимизации. Классическая теория оптимизации базируется на аппарате дифференциального исчисления (математического анализа). Алгоритм выглядит следующим образом:

1. Найти первую производную:
2. Найти корни квадратного уравнения:
3. Найти вторую производную:
4. Определить тип экстремума в точках
5. Найти значения функции в точках экстремума
6. Найти значения функции на границах интервала a и b
7. Определить максимальные и минимальные значения целевой функции*.*

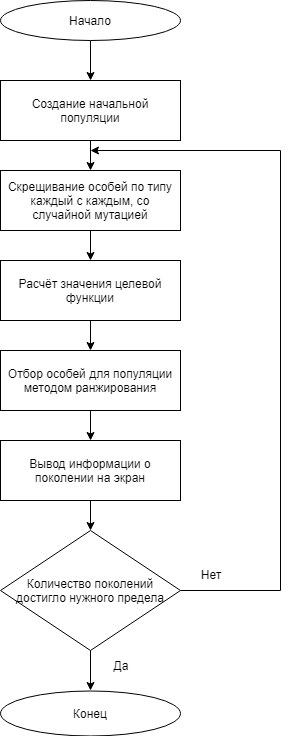
Решение с помощью генетического алгоритма:

1. Выбираются параметры для генетического алгоритма:
   1. Количество особей в популяции;
   2. Число скрещиваний;
   3. Число мутаций на поколение;
2. Создается начальная популяция со случайным набором генов;
3. Высчитывается значение целевой функции для каждой особи;
4. Отбираются наилучшие особи, удовлетворяющие условию;
5. Происходит скрещивание и мутация;
6. Следующая итерация, на которой происходят шаги 3-5;
7. Алгоритм завершает работу при достижении заданного условия.

### Адаптация генетического алгоритма для нахождения экстремума функции и тестирование полученного алгоритма на известных функциях.

В ходе выполнения выпускной квалификационной работы в качестве примера работы приложения на генетических алгоритмах было создано приложение, которое находит минимум функции.

Алгоритм выполнения приложения представлен на рис.



Параметры генетического алгоритма:

1. Размер популяции равен 10
2. Количество поколений 5000
3. Вероятность мутации 30%
4. Величина мутации 50%
5. Отбор методом ранжирования
6. Кроссовер - каждый с каждым

Тестирование было проведено на функции . При нахождении минимума функции с помощью классической теории оптимизации был получен результат

При нахождении минимума с помощью генетического алгоритма получены следующие результаты:

* Лучший:
* Худший: 0

Также были рассмотрены другие функции

* Функция Розенброка (Многомерное обобщение):
* Функция Бута (Двумерный случай):
* Функция для одной переменной:

Были получены следующие результаты:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Аналитическое решение | Генетический алгоритм  (Лучший) | Генетический алгоритм  (Худший) |
| Функция Розенброка |  | при n=15 | при n=15 |
| Функция Бута |  |  |  |
| Функция для одной переменной |  |  |  |

По полученным данным можно сказать, что генетический алгоритм хорошо срабатывает для функций с одной и двумя неизвестными в функции. Отклонение от аналитического решения составляет 1%-2%. Для увеличения точности в многомерном случае необходимо доработать алгоритм.

## Разработка игры на игровом движке

### Выбор подходящего игрового движка

На данный момент в мире много игровых движков для создания 2D и 3D игр. Мной было рассмотрено два популярных игровых движка:

* UNITY
* Unreal Engine.

Задачи, которые решаются игровым движком:

* симуляция физики, модуль столкновений (collisions)
* искусственный интеллект в игре
* реализация 3d или 2d графики и анимации
* звук в игре
* сетевое взаимодействие клиентов и серверов игры
* локализация
* выполнение скриптов (в том числе сюжетных)
* управление памятью, и т.д.

Игровые движки Unity и Unreal Engine 4 поддерживают все задачи представленные выше, но у каждого игрового движка есть свои плюсы и минусы.

Из ярких плюсов Unreal Engine – это наличие системы визуального программирования Blueprint, который позволяет создать работающую игру без особого знания программирования. Блоки Blueprint расширяются языком C++, но и реализованной логики хватит для создания простой игры.

Из минусов можно выделить:

* тяжеловесный клиент среды, не все компьютеры смогут потянуть его
* отсутствием готовых ассетов для разработки, необходим начальный навык создания спрайтов.

Из ярких плюсов Unity – это наличие бесплатных ассетов для создания игр, подробного руководства по созданию игры, использование языка программирования C#, создание с помощью одного набора игр для множества платформ. Относительно легкий клиент среды разработки не требующий мощных вычислительных возможностей компьютера.

Из минусов можно выделить:

* отсутствие широкого русскоязычного сообщества
* все руководства на английском языке
* большую часть логики необходимо реализовывать самому

Исходя из плюсов и минусов обоих игровых движков мною был выбран игровой движок Unity, из-за большого количества руководств и понятной логики разработки игры, а также наличия небольшого опыта работы с данной средой разработки.

### Создание префаба модели-машинки

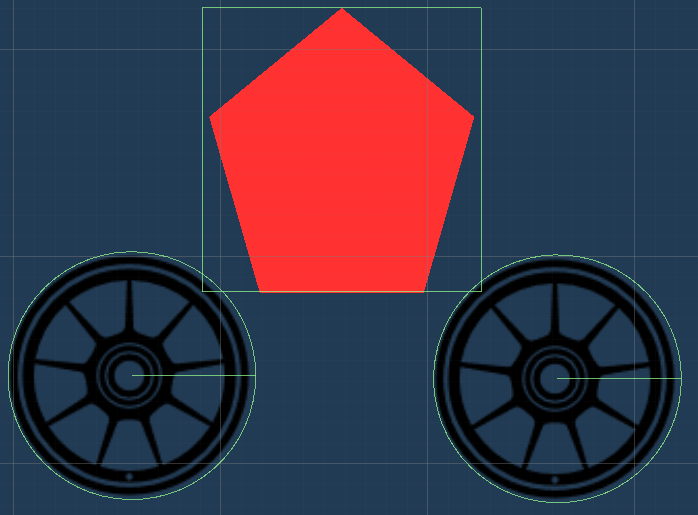
Префаб – это особый тип ассетов, который позволяет хранить весь игровой объект с его компонентами и параметрами. Префаб выступает в роли шаблона для экземпляра сцены. Внесения изменений в префаб влечёт моментальное изменение в экземпляр этого префаба.

Для создания префаба модели-машинки были использованы спрайты пятиугольника и спрайт колеса. Пятиугольник выступает в роли корпуса, а колёса в роли колёс.

Для придания физических свойств модели-машинке на компоненты префаба были установлены следующие компоненты Unity, отвечающие за физику объектов:

* Sprite Renderer – компонент, отвечающий за отрисовку спрайта на сцене. Данный компонент установлен на корпусе, колёсах машинки.
* Rigidbody 2D – компонент, который говорит движку, что тело должно обладать физикой. Данный компонент устанавливается на корпус и колёса машинки.
* Box Collider 2D – компонент, отвечающий за коллизию объектов квадратного или прямоугольного вида, то есть определения столкновений, прикосновений и тому подобного. Данный компонент установлен на корпус машинки.
* Circle Collider 2D - компонент, отвечающий за коллизию объектов имеющих круглый вид. Данный компонент был установлен на колёса машинки.
* Distance Joint 2D – компонент, отвечающий за соединение одного физического объекта с другим. Данный компонент был установлен на колёса для соединения с корпусом.

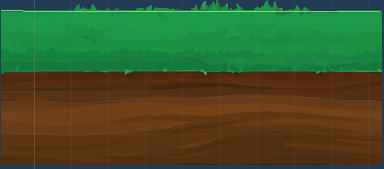
После объединения в один родительский объект получили префаб машинки рис.



### Создание префаба дороги

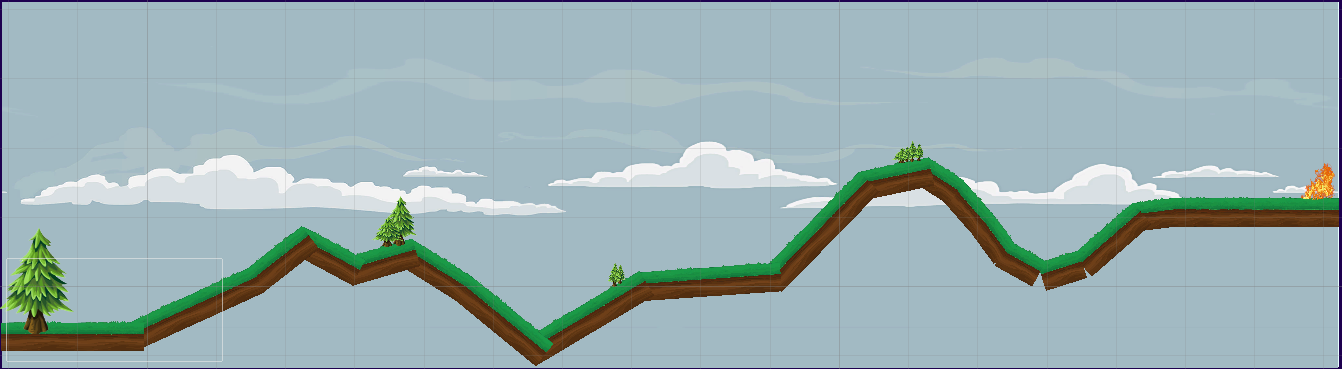
Для создания префаба дороги был использован спрайт травы с землёй и компонентом Edge Collider 2D, который отвечает за определение коллизий объектов, данный компонент подходит для создания коллайдера линии с редактированием формы.

Получившийся префаб представлен на рис.



### Создание трассы

Трасса была составлена из префабов дороги в ручную. Также на трассу были помещены точка генерации машинок и конечная точка, до которой машинки должны доехать. Трасса представлена на рис.



### Разработка контроллера модели-машинки со свойствами модели

Контроллер представляет из себя скрипт, написанный на языке программирования C#. Скрипт отвечает за расчёт значения fitness функции; ограничения скорости машинки; уничтожение машинки, если машинки «застряла» или истекло время на эволюцию.

Скрипт привязан к префабу модели-машинки, как следствие данный скрипт выполняется для каждого экземпляра этого префаба отдельно.

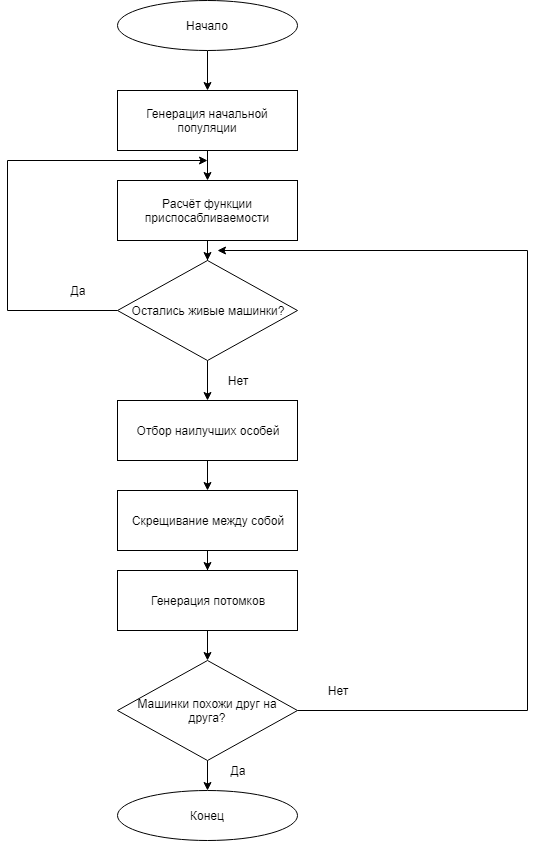
### Разработка контроллера движения модели-машинки

Скрипт отвечает за движение колёс машинки с помощью задания угловой скорости и силы.

Скрипт привязан к префабу колёс машинки, как следствие данный скрипт выполняется для каждого экземпляра этого префаба отдельно.

### Разработка контроллера генерации объектов

Скрипт отвечает за начальную генерацию объектов и дальнейшее их создание на основе генетического алгоритма. Алгоритм представлен на рис.



### Тестирование игры

# Заключение. Полученные результаты

# Список цитируемой литературы