Департамент образования города Москвы

Государственное бюджетное общеобразовательное учреждение города Москвы «Школа № 1505 «Преображенская»»

**Невероятная эволюция Изоморфов**

**(применение генетических алгоритмов в программировании)**

Выполнила:

Зеликова Вероника Ивановна, 10 класс, школа № 1505

Руководитель:

Ветюков Дмитрий Алексеевич, учитель физики, школа № 1505

**Москва, 2018-2019 год.**

Оглавление

[Введение 3](#_Toc5566776)

[Глава 1. Эволюционная теория и генетические алгоритмы 6](#_Toc5566777)

[1.1 Введение 6](#_Toc5566778)

[1.2 Когда нам нужны генетические алгоритмы 8](#_Toc5566779)

[1.3 Применение генетических алгоритмов в реальных задачах 8](#_Toc5566780)

[1.4 Примеры программ, наглядно демонстрирующих эволюцию 10](#_Toc5566781)

[1.5 Как работает эволюция 12](#_Toc5566782)

[1.6 Как это реализовано в генетических алгоритмах 13](#_Toc5566783)

[1.7 Наследственность и изменчивость 13](#_Toc5566784)

[1.8 Размножение 14](#_Toc5566785)

[1.9 Мутации 15](#_Toc5566786)

[1.10 Естественный отбор 16](#_Toc5566787)

[1.11 Слово о ландшафте приспособленности 19](#_Toc5566788)

[1.12 Ловушка локальных максимумов 20](#_Toc5566789)

[1.13 Ловушка номер 2 – гиперспециализация 21](#_Toc5566790)

[1.14 Поиск выхода из ловушек 21](#_Toc5566791)

[Глава 2. Программная часть 24](#_Toc5566792)

[2.1 Структура программы 24](#_Toc5566793)

[2.2 Работа программы 24](#_Toc5566794)

[2.3 Дополнительные функции программы 27](#_Toc5566795)

[Глава 3. Исследования 30](#_Toc5566796)

[3.1 Существующие настройки программы 30](#_Toc5566797)

[3.2 Функции, помогавшие в исследовании 32](#_Toc5566798)

[3.3 Графики различных измерений 33](#_Toc5566799)

[Заключение 39](#_Toc5566800)

[Список литературы 40](#_Toc5566801)

# Введение

Мир вокруг нас очень сложен, он значительно сложнее, чем в школьных учебниках и, тем более, в школьных задачниках. Реальная жизнь, наука и техника, зачастую ставят перед людьми проблемы, у которых нет или еще нет единственного арифметически верного решения. Тем не менее, такие проблемы, вопросы и задачи надо как-то решать, от этого может зависеть благополучие, развитие, да и само выживание человечества.

На сложные жизненные задачи нам не всегда нужен идеальный ответ прямо сейчас. Часто достаточно получить хоть какой-нибудь вариант, применить его на практике, а потом потихоньку улучшать решение, стремясь к идеалу. Так происходит уже на протяжении тысяч лет развития цивилизации. Результат можно наблюдать, просто оглядевшись по сторонам, и сравнив окружающие предметы с теми, что были доступны людям десять, сто или тысячу лет назад.

Самое интересное, что такой способ поступательного развития придумали не люди, сама природа создала его и использовала миллиарды лет, а люди только подсмотрели, назвали «эволюцией» и стали использовать. Сначала они сами просто биологически эволюционировали, подобно остальным живым существам, подстраиваясь под меняющиеся условия внешней среды. Потом научились создавать простейшие предметы и орудия вроде каменного рубила или палки-копалки, постепенно улучшая технологию, обучаясь у предков, исправляя их ошибки и передавая новые знания потомкам. Позже люди научились направлять эволюцию других живых существ, пусть и неосознанно, подчиняя ее своим нуждам, получая новые породы домашних животных и сорта культурных растений, так появилась селекция.

В середине 19-го века Чарльз Дарвин, изучая опыт селекционеров и сравнивая его со своими наблюдениями за живой природой во время кругосветного путешествия, смог установить и теоретически обосновать движущую силу эволюции, которой оказался «естественный отбор». Примерно в то же время Грегор Мендель, скрещивая разные сорта гороха, обнаружил, что в основе наследования фенотипических признаков организмов, лежат дискретные носители информации, которые он назвал «гены». На основании своих наблюдений Мендель сформулировал три закона генетики, но научное сообщество в то время не оценило его открытия.

В начале 20-го века генетики из группы Томаса Моргана установили более точные механизмы наследования, попутно «переоткрыв» законы Менделя. Также было выяснено, что физическим носителем наследственной информации является молекула ДНК. В 1953 году Джеймс Уотсон и Френсис Крик исследовали структуру ДНК и выяснили, каким образом за счет комплементарности нуклеотидов происходит точное копирование, содержащейся в ней информации. Позже другими учеными механизмы копирования ДНК были изучены более подробно, выяснено, как в копиях могут вкрадываться ошибки, расшифрован генетический код и сделано много удивительных открытий, на основании которых родилась современная «синтетическая теория эволюции» (что означает синтез классических идей Дарвина с достижениями молекулярной генетики). Ученые во всем мире продолжают исследовать загадки ДНК по сей день.

С появлением мощных компьютеров, оказалось, что биологические процессы кодирования, копирования и мутирования информации можно перенести на самые разные сферы человеческой (и не только человеческой) деятельности, создавая саморазвивающиеся математические модели сложных систем. Эти модели живут, размножаются и развиваются в недрах компьютерных процессоров, а управляют их жизнедеятельностью алгоритмы, которые по аналогии с биологической эволюцией так и назвали «генетические алгоритмы».

**Цель:** создание программы на основе метода генетических алгоритмов, имитирующей процесс эволюции, ее исследование и оптимизация – нахождение самых оптимальных настроек, которые позволят за минимальное время получить лучший результат. Если конкретнее, то программе на вход дается заданная картинка (далее, оригинал), а также некая популяция, которая в ходе эволюции должна приблизиться к оригиналу.

**Актуальность:** программа создана для наглядной демонстрации работы эволюции. Также разработанный алгоритм может использоваться в других практических целях, например, в области дизайна, планирования и для составления расписаний.

**Этапы работы:**

1. Изучение материалов по теме «генетические алгоритмы»;
2. Создание собственного алгоритма;
3. Написание программы;
4. Добавление дополнительных функций для оптимизации работы;
5. Исследование результатов работы программы. Фиксирование определенных параметров, контролируемое изменение других и сравнение результатов. Построение графиков. Определение идеальных или близких к идеальным настроек;
6. Проведение аналогий по полученным данным с реальной эволюцией;
7. Запись поясняющего видеоролика;
8. Создание демо-версии мобильного приложения.

# Глава 1. Эволюционная теория и генетические алгоритмы

## 1.1 Введение

Мир вокруг нас очень сложен, он значительно сложнее, чем в школьных учебниках и, тем более, в школьных задачниках. Реальная жизнь, наука и техника, зачастую ставят перед людьми проблемы, у которых нет или еще нет единственного арифметически верного решения. Тем не менее, такие проблемы, вопросы и задачи надо как-то решать, от этого может зависеть благополучие, развитие, да и само выживание человечества.

На сложные жизненные задачи нам не всегда нужен идеальный ответ прямо сейчас. Часто достаточно получить хоть какой-нибудь вариант, применить его на практике, а потом потихоньку улучшать решение, стремясь к идеалу. Так происходит уже на протяжении тысяч лет развития цивилизации. Результат можно наблюдать, просто оглядевшись по сторонам, и сравнив окружающие предметы с теми, что были доступны людям десять, сто или тысячу лет назад.

Самое интересное, что такой способ поступательного развития придумали не люди, сама природа создала его и использовала миллиарды лет, а люди только подсмотрели, назвали «эволюцией» и стали использовать. Сначала они сами просто биологически эволюционировали, подобно остальным живым существам, подстраиваясь под меняющиеся условия внешней среды. Потом научились создавать простейшие предметы и орудия вроде каменного рубила или палки-копалки, постепенно улучшая технологию, обучаясь у предков, исправляя их ошибки и передавая новые знания потомкам. Позже люди научились направлять эволюцию других живых существ, пусть и неосознанно, подчиняя ее своим нуждам, получая новые породы домашних животных и сорта культурных растений, так появилась селекция.

В середине 19-го века Чарльз Дарвин, изучая опыт селекционеров и сравнивая его со своими наблюдениями за живой природой во время кругосветного путешествия, смог установить и теоретически обосновать движущую силу эволюции, которой оказался «естественный отбор». Примерно в то же время Грегор Мендель, скрещивая разные сорта гороха, обнаружил, что в основе наследования фенотипических признаков организмов, лежат дискретные носители информации, которые он назвал «гены». На основании своих наблюдений Мендель сформулировал три закона генетики, но научное сообщество в то время не оценило его открытия.

В начале 20-го века генетики из группы Томаса Моргана установили более точные механизмы наследования, попутно «переоткрыв» законы Менделя. Также было выяснено, что физическим носителем наследственной информации является молекула ДНК. В 1953 году Джеймс Уотсон и Френсис Крик исследовали структуру ДНК и выяснили, каким образом за счет комплементарности нуклеотидов происходит точное копирование, содержащейся в ней информации. Позже другими учеными механизмы копирования ДНК были изучены более подробно, выяснено, как в копиях могут вкрадываться ошибки, расшифрован генетический код и сделано много удивительных открытий, на основании которых родилась современная «синтетическая теория эволюции» (что означает синтез классических идей Дарвина с достижениями молекулярной генетики). Ученые во всем мире продолжают исследовать загадки ДНК по сей день.

С появлением мощных компьютеров, оказалось, что биологические процессы кодирования, копирования и мутирования информации можно перенести на самые разные сферы человеческой (и не только человеческой) деятельности, создавая саморазвивающиеся математические модели сложных систем. Эти модели живут, размножаются и развиваются в недрах компьютерных процессоров, а управляют их жизнедеятельностью алгоритмы, которые по аналогии с биологической эволюцией так и назвали «генетические алгоритмы».

## 1.2 Когда нам нужны генетические алгоритмы

Как уже было сказано ранее, существует множество реальных задач, обладающих следующими неприятными для нас свойствами:

1. Они не имеют известного строгого аналитического решения.
2. Возможно, у нас даже нет хорошей теории, описывающей те процессы, что происходят в рассматриваемой модели, или же эти процессы математически слишком сложны.
3. Мы не имеем понятия, каким именно должен быть идеальный результат.

С другой стороны, у рассматриваемых задач могут быть и удобные свойства:

1. Мы можем создать некие математические объекты, моделирующие хоть какой-нибудь результат решения
2. У этих моделей есть ряд параметров, меняя которые можно получить разные варианты решения.
3. У нас есть способ на практике сравнить два разных варианта и выбрать, какой из них лучше.

В этом месте мы можем запустить процесс эволюции и после нескольких итераций начать получать все более и более лучшие варианты, приближаясь к идеалу (возможно, никогда его не достигая).

## 1.3 Применение генетических алгоритмов в реальных задачах

Из приведенного в предыдущей главе списка условий легко определить какие математические модели хорошо подходят для того, чтобы натравить на них генетический алгоритм.

Очевидным объектом для генетических экспериментов являются нейронные сети. Несмотря на десятилетия исследований и огромный прогресс в результатах, у нас до сих пор нет строгой теории, как должна выглядеть топология нейронной сети под каждую конкретную задачу. До сих пор нейронные сети во многом строятся на глаз, а миллиарды нейронных связей заполняются случайными значениями весов. Просто идеальный кандидат для генетического алгоритма.

Вторым очевидным поставщиком задач для генетических алгоритмов является биохимия. В частности, исследования новых лекарств. Подобрать длинную органическую молекулу А такую, чтобы она хорошо взаимодействовала с молекулой Б – это тот самый вопрос, над которым бьется биологическая эволюция уже не первый миллиард лет. Правда, теперь нам не обязательно синтезировать обе молекулы в пробирке. Современная наука вполне способна запустить любую химическую реакцию на компьютере.

Не отстает и теория игр. На бумаге, где все игроки действуют четко и рационально, теоретические построения дают отличные результаты, а в реальности случайные решения не оставляют от теоретических предсказаний камня на камне. Но ведь случайность – это плоть и кровь генетических алгоритмов.

Другой важной областью применения стало составление расписаний. Это только на первый взгляд звучит просто. На самом деле задача о составлении расписаний относится к классу до сих пор теоретически не решенных задач. Мы умеем получать идеальные расписания только прямым перебором всех вариантов. Так что, при увеличении входных данных сложность задачи расчет экспоненциально, а в некоторых случаях и факториально. И если для составления школьного расписания достаточно пары завучей с карандашами и тетрадкой, то для крупного университета уже требуется компьютер и специальные программы, а для международного аэропорта или железнодорожной сети страны необходимы усилия самых лучших суперкомпьютеров. Преимущество генетического алгоритма тут в том, что на каждой итерации он дает все более лучший вариант расписания и в любой момент можно прекратить его работу, посчитав, что предложенный вариант уже достаточно хорош.

Ну и напоследок поговорим о применении генетических алгоритмов, про которое часто забывают упомянуть в специализированных статьях. Это популяризация эволюционной теории среди школьников и детей любого возраста. Сухие строчки из учебника биологии часто проходят мимо ушей учащихся, а факты, доказывающие существования и силу эволюции не впечатляют плохо подготовленный мозг. Дети не могут увидеть и пощупать эволюцию. В результате в них на всю жизнь закладывается отношение «Верю/не верю Дарвину» не только для теории эволюции, но и для всей науки. А сама эволюция лежит в сознании где-то между Человеком-Пауком и Черепашками ниндзя. Чем потом пользуются всевозможные шарлатаны, пропагандирующие веру в «несократимую сложность» и разумный дизайн.

А ведь достаточно всего несколько минут понаблюдать, как на экране компьютера или смартфона под действием генетического алгоритма из полного хаоса рождается «несократимая сложность», чтобы рассказ учителя о клювах галапагосских вьюрков или, или генах, влияющих на цвет гороха, лег на значительно более подготовленную к пониманию почву.

## 1.4 Примеры программ, наглядно демонстрирующих эволюцию

Одним из первых о написании такой программы, не делающей ничего полезного, а лишь демонстрирующую сам факт эволюции, еще в прошлом веке задумался знаменитый популяризатор эволюционной теории Ричард Докинз. Его первый вариант генетического алгоритма под названием «WEASEL PROGRAM» был очень примитивен. Он всего лишь превращал случайный набор букв в цитату из Гамлета: «METHINKS IT IS LIKE A WEASEL». Зато алгоритм шустро работал на тогдашних медленных компьютерах и позволял добиться результата всего за несколько секунд и пару десятков поколений. В своих книгах «Слепой Часовщик» и «Самое грандиозное шоу на Земле» Докинз сам разбирает недостатки такой программы - буквы в ней мутируют независимо и никак не связаны друг с другом, тогда как в реальности изменение работы одного гена приводит к целому каскаду изменений в результатах работы других генов. Это явление называется эпистаз. В книгах Докинз приводит и другие более продвинутые варианты своих эволюционных программ.

Несколько лет назад в интернете появился сайт boxcar2d.com, на котором случайно сгенерированные машинки, состоящие из нескольких треугольников с приделанными к ним кружками колес, пытались пройти ухабистую трассу. Сначала нелепые уродцы с колесами на крыше не могут проехать и метра, но потом мутации и естественный отбор потихоньку превращают их настоящие гоночные болиды или мощные внедорожники. Это очень реалистичная симуляция, так как не существует наперед заданной идеальной машинки, к которой нужно стремиться. Фактором отбора являются случайно сгенерированная трасса и законы физики, а результат непредсказуем. Впрочем, математическая модель, положенная в основу машинок, довольно проста и даже случайный перебор часто давал очень хорошие экземпляры уже на первом поколении. Что совсем не впечатляет адептов «несократимой сложности».

Ну а третьим примером здесь будет программа, которая легла в основу данной дипломной работы. Наша программа под названием «Эволюция Изоморфов» позволяет путем естественного отбора превратить случайный набор цветных пятен разной формы в любую заданную картинку. Причем картинка получается векторной, визуально слабо отличающейся от растрового оригинала. Так как полупрозрачные пятнышки многократно перекрывают друг друга, то изменение любого из них сразу влияет на цвет всех выше- и нижележащих. А сложность результата такова, что не хватит всего времени существования Вселенной, чтобы, набрасывая пятна случайного цвета и размера, получить портрет Моны Лизы или фото своей кошки, только что сделанное на мобильник.

Тем не менее, генетический алгоритм уже через несколько минут работы производит результат, похожий на некую магию, а за часы и дни он способен создавать картины, почти не отличимые от оригиналов. Но, прежде чем раскрыть тайну волшебства генетического алгоритма, надо немножко поговорить о том, как работает эволюция.

## 1.5 Как работает эволюция

Для того чтобы система смогла начать эволюционировать, она должна соответствовать 4 простым условиям:

1. **Изменчивость**. Система должна состоять из элементов, отличающихся друг от друга некоторыми свойствами.
2. **Наследственность**. Входящие в систему элементы, должны хранить в себе информацию, однозначно определяющую данные свойства. Идентичная информация должна порождать идентичные элементы.
3. **Размножение и мутации**. Элементы системы должны уметь создавать себе подобных на основе хранящейся в них информации, при этом информация не всегда копируется точно, что иногда приводит к появлению наследников, со свойствами, отличными от свойств родителя (это называется мутацией).
4. **Естественный отбор**. От свойств элемента тем или иным образом должно зависеть количество потомков, которых он может оставить.

В результате, элементов системы, обладающих лучшими с точки зрения критериев отбора свойствами, с каждым поколением будет становиться все больше. Они будут оставлять еще больше потомков, обладающих благоприятными свойствами. Потомки, получившие неблагоприятные мутации, будут отсеиваться отбором, а удачные мутанты наоборот заполнят всю популяцию, вытеснив своих менее приспособленных собратьев. И так будет повторяться по кругу много раз, улучшая среднее качество элементов в системе.

## 1.6 Как это реализовано в генетических алгоритмах

По аналогии с биологической эволюцией, эволюционирующие компьютерные системы называют популяциями, а входящие в их состав элементы – организмами или особями. Даже если в роли организма выступает набор весов для нейронной сети, коэффициенты системы линейных уравнений, алгоритм действий для разделения N ценных предметов между M пиратами, последовательность инструкций для игрового бота, или совокупность графических примитивов для рисования векторного изображения. Все это математические организмы.

Чтобы запустить процесс естественного отбора, компьютерные организмы и популяции должны удовлетворять тем же условиям, что и их биологические собраться, но с некоторыми нюансами, продиктованными их цифровой природой.

## 1.7 Наследственность и изменчивость

Если организм представляет собой математическую модель с изменяемыми параметрами, то в качестве наследственной информации удобно хранить эти параметры просто как массив чисел. Тогда каждое число будет геном. Удобно заранее подготовить математическую модель так, чтобы все параметры были нормализованы и изменялись в диапазоне, например, от 0 до 1. Это упростит дальнейшие манипуляции с геномом.

При необходимости можно объединить числовые параметры в логические группы и назвать генами уже их. Гены также можно группировать в более крупные объединения, называя их по биологической аналогии хромосомами. В большинстве случаев программисты не усложняют модель и создают столько одну хромосому, содержащую весь геном организма.

Все организмы популяции хранятся в виде таких числовых массивов и при необходимости разворачиваются в соответствующие объекты (в смысле объектно-ориентированного программирования), с которыми можно производить предусмотренные моделью манипуляции. Так набор чисел от нуля до единицы превращается в набор нейронов нейросети с соответствующими весами. А нейросеть уже потом может заниматься различением изображений котиков от песиков, играть в шахматы или управлять финансовыми потоками гигантской транс-корпорации.

## 1.8 Размножение

Размножение числовых массивов – очень простое действие, состоящие просто из копирования данных из старого массива в новый. Такое размножение называется клонированием или бесполым размножением.

Возможна также реализация аналога полового размножения в биологии. В этом случае часть наследственной информации нужно взять у одного родителя, а оставшуюся часть – у другого. На практике это реализуется поочередным копированием в новый массив соответствующих данных то от одного родителя, то от второго. Как выбирается, какие именно данные будут скопированы от каждого родителя и будет ли этих родителей только двое, зависит уже от конкретной реализации генетического алгоритма. Самый простой способ – скопировать элементы массива с нулевого до N-ного от первого родителя, а с N-ного по последний – от второго. N при этом может стоять как ровно посередине хромосомы, так и выбираться случайным образом.

В эволюционном программировании обычно не различают организмы по полам, и они соответствуют биологическим гермафродитам. Хотя, для некоторых задач такое разделение на условно «мужской» и условно «женский» пол с разными свойствами могло бы быть интересным. Также программисты редко пытаются создать цифровой аналог двойного набора хромосом (при котором каждый ген организма имеет копию от отца и от матери). Это связано прежде всего с необходимостью в дальнейшем определять, какой из генов является доминантым и влияет на фенотипические признаки организма, а какой – рецессивным и просто наследуется половиной потомков. Но если математическая модель позволяет провести такое различие, то запрограммировать пололвое размножение более близкое к биологическому с гаплоидными гаметами и диплоидными зиготами не составляет труда.

1

2

3

4

5

6

7

1

2

3

4

5

6

7

4

5

6

7

1

2

3

1

2

3

4

5

6

7

N = 3

## 1.9 Мутации

Независимо от способа размножения с потомками могут произойти случайные мутации. На практике обычно это выглядит так: с некоторым установленным заранее шансом выбираются те гены, которые будут подвергнуты мутационному воздействию. Этот шанс называется шансом мутации. После чего значения чисел, хранящиеся в выбранных генах меняются на случайную величину в заранее определенном диапазоне значений. Этот диапазон называется максимальной величиной мутации или максимальной дисперсией.

Шанс мутации и максимальная дисперсия - это ключевые параметры, радикальным образом влияющие на скорость эволюции и получившийся результат. Для каждой популяции и математической модели организмов они должны подбираться индивидуально.

Помимо мутации генов возможны мутации, изменяющие число и взаимное расположение уже существующих генов в хромосоме. Это аналог хромосомных мутаций в биологии. При таких мутациях уже существующие гены могут быть продублированы, перемещены или скопированы в другое место хромосомы или на другую хромосому. Часть генов может быть заменена другими или полностью вырезана из генома. В биологии это называется: дупликация, транслокация и делеция генов. Не для всех математических моделей возможны хромосомные мутации, иногда для работоспособности алгоритма число и взаимное расположение генов должны оставаться строго неизменными.

1

2

3

4

5

6

7

4

1

2

3

5

6

7

4

1

2

3

5

6

7

Дупликликация

Транслокация

Делеция

В некоторых реализациях генетических алгоритмов мутациям подвергаются только потомки в момент размножения, а в других и «взрослые» особи тоже.

## 1.10 Естественный отбор

Так как компьютерные мощности всегда ограничены, то цифровая популяция обычно имеет фиксированную численность. Действие естественного отбора на такую популяцию идет двумя путями:

1. Выяснение, кто из текущих организмов будет размножаться и сколько раз
2. Определение, какие организмы останутся в популяции на следующем поколении.

Первое соответствует половому отбору, когда красивые, умные и сильные легко находят себе партнеров (или достаточное количество ресурсов при бесполом размножении), а неудачники остаются бобылями до конца жизни. Для определения количества потомков часто используют виртуальную рулетку с неравными секторами. Самый большой сектор на рулетке соответствует лучшему организму, сектор поменьше – второму номеру, еще поменьше – третьему и т.д. Вращение такой рулетки определяет пары для размножения, подобно игре в «бутылочку». Но иногда поступают проще: самый лучший организм просто скрещивается с несколькими партнерами похуже, такой способ называется элитной селекцией.

лучший

2-й

3-й

4-й

5-й

Второе моделирует давление на популяцию хищника, который съедает слабых и беспомощных, давая шанс выжить и размножиться только наиболее приспособленным. Здесь тоже возможны варианты. В одном из них мы просто отбираем N самых лучших особей и пропускаем их дальше, а остальные считаются съеденными. Это жесткий отсекающий отбор, в результате которого популяция может быстро забиться идентичными клонами самого лучшего организма.

Отсекающий отбор

Турнир

В другом варианте можно устроить своеобразный турнир, где организмы делятся на случайные пары и соревнуются друг с другом за право выхода в следующий круг, пока у нас опять не останется N особей. Такой подход позволяет выжить чуть более слабым, но перспективным организмам и повышает изменчивость в популяции.

Часто делают так, что в следующее поколение переходят только родившиеся на прошлом кругу потомки, а их предки все поголовно «умирают», или от них оставляют только парочку чемпионов прошлого сезона. Возможны и более щадящие условия отбора. При реализации собственных генетических алгоритмов можно пробовать разные варианты.

В любом случае, чтобы определить лучших, нам нужно уметь сортировать организмы по определенному рейтингу приспособленности, или хотя бы сравнивать, два организма между собой, выявляя победителя. Этим сравнением занимается алгоритм, который в специальной литературе называют «фитнесс функция» или «функция приспособленности». Вид фитнесс функции радикально зависит от той математической модели, что лежит в основе организмов, в каждом случае она своя.

Например, нейронные сети, учащиеся различать на картинках котиков, можно прогнать через тестовый набор картинок и вычислить для них процент правильных ответов. Это и будет фитнесс-функцией. Игрового бота можно отправить в бой с другим ботом и посмотреть, кто выйдет победителем или замерить для них время прохождения игрового уровня и набранные очки. Для лекарственной молекулы можно посчитать скорость и продолжительность взаимодействия с молекулой-мишенью. Биржевых анализаторов можно отправить играть на виртуальную биржу виртуальными деньгами и выбрать самого богатого.

А если организмы, как в случае нашей программы, представляют собой векторные изображения, то эти изображения нужно как минимум нарисовать, а потом еще и сравнить специальным алгоритмом, определяющим сходство тестируемой картинки с оригиналом.

Из сказанного очевидно, что именно работа фитнесс функции тратит в генетических программах львиную долю времени и потребляет огромные машинные ресурсы.

## 1.11 Слово о ландшафте приспособленности

Процесс эволюционного развития популяции можно свести к поиску максимума некой сложной математической функции. Функции, являющейся математическим воплощением результатов работы, рассмотренной выше фитнесс функции, и зависящей от всех параметров, заложенных в генах организмов. Нам совершенно необязательно знать, как именно выглядит такая функция (если бы мы знали, нам может быть и не понадобились бы генетические алгоритмы), главное понимать, какими свойствами она может обладать.

Приспособленность

Изменчивость

Сложная поверхность, порождаемая попыткой визуализировать график такой функции, называется «ландшафтом приспособленности». Потому в научно-популярных статьях поиск максимума такой функции сравнивают с путешествием по холмистой местности.

Изначально популяция сидит в долине (там, где приспособленность организмов низка), разные особи находятся на разной высоте, те, кто расположен выше, получают эволюционное преимущество и оставляют больше потомков, а нижние постепенно вымирают. В результате через некоторое время вся популяция оказывается на вершине ближайшего холма. Ричард Докинз даже метко назвал такой процесс «восхождением на пик Невероятности», имея в виду, что вероятность оказаться на вершине холма в результате случайных блужданий крайне низка.

Ура! Эволюция свершилась, достигнут идеал и порожден венец творения!

Но достигнут ли?..

## 1.12 Ловушка локальных максимумов

Нет, идеал не достигнут. Ведь рядом за соседней долиной вполне может быть холм повыше, а за ним еще повыше, неизвестно, существует ли вообще самый большой на свете холм. И главное, существует ли такой путь на этот самый высокий холм, при котором особям не приходится спускаться в долину? Ответ на этот вопрос разный для разных систем и моделей. И именно от этого ответа зависит, является ли для данной модели генетический алгоритм эффективным способом решения задач или не является.

Попадание в локальный максимум на практике можно проиллюстрировать следующим образом. Допустим, в результате естественного отбора у нас уже получился хороший очень приспособленный организм, небольшая мутация в любом из генов которого вредна и уменьшает приспособленность, а значит будет отсеяна отбором. Повышение приспособленности можно добиться только одновременной сонаправленной мутаций нескольких генов, вероятность чего крайне низка. В результате через некоторое время вся популяция заполняется идентичными копиями этого самого лучшего организма, после чего теряет эволюционные перспективы

Именно регулярные попадания популяцией в локальные максимумы являются одной из серьезных слабостей генетических алгоритмов, с которой разработчику алгоритма постоянно приходится бороться.

## 1.13 Ловушка номер 2 – гиперспециализация

Если проблемой нейронных сетей является переобучаемость, то ее сестрой-близнецом у генетических алгоритмов становится гиперспециализация. В прошлом динозавры были очень хорошими специалистами и мгновенно вымерли, не сумев приспособиться к изменившимся условиям среды. Так что гиперспециализация с точки зрения эволюции - это плохо.

Гиперспециализация наступает, когда фитнесс-функция и ряд других параметров популяции, образующих вместе аналог экологической ниши для организмов, остаются неизменными на протяжении многих тысяч поколений. В этом случае отбор идет уже не в сторону решения нашей задачи, а в сторону приспособления к конкретной фитнесс-функции. А так как фитнесс-функция – это всего лишь упрощенная модель реальной задачи, то попытка приспособиться к ее мелким деталям только уводят нас в сторону, отвлекая от главного.

Например, в задаче рисования картинок векторными примитивами мы не можем постоянно использовать для сравнения изображения большого размера, так как это требует неоправданно больших вычислительных ресурсов. В дело поначалу идут картинки с длиной стороны в 8, 16, 32, 64 пикселя. Попытка генетического алгоритма точно повторить каждый пиксель на такой картинке приводит не только к неприятной «лесенке» на контрастных границах, но и к появлению странных ярких полос толщиной меньше пикселя, но прекрасно заметных глазом.

## 1.14 Поиск выхода из ловушек

Попробуем рассмотреть несколько способов обхода ловушки локальных максимумов и гиперспециализации, применимых в разных ситуациях:

**«Мутабор»!** Именно этим волшебным словом халиф из сказки превращался в аиста и обратно, значительно облегчая себе жизнь. То есть просто попробуем время от времени повышать шанс и величину мутаций, что позволяет согнать популяцию с насиженного холма. И заставить поискать холм поинтереснее. Но такими методами можно загнать популяцию и в еще более глубокую яму.

**«Мелкими группами по миллион человек»**. Если наша популяция достаточно велика и разнообразна, возможно, нам повезет и ее части полезут на разные холмы, и какой-то из них окажется тем заветным самым высоким в мире холмом. А может быть и не повезет, тогда мы просто зря потратим вычислительные мощности.

**«А мы пойдем на север!»**. Также можно попробовать разбить большую популяцию на много маленьких подпопуляций, проживающих в разных условиях, и время от времени обменивающихся мигрантами и новыми интересными генами.

**«Добрым словом и пистолетом»**. Естественный отбор слишком суров и не дает особям популяции спуститься с гор вниз? Чуть ослабляем давление, смотрим как популяция расползается в разные стороны и ждем, что кто-нибудь успеет перебежать долину до того, как его съедят. Возможно, ждать придется долго.

**«Такой фитнесс нам не нужен»!** Если мы застряли на пути к идеалу, возможно, стоит сменить фитнесс-тренера, ну или хотя бы фитнесс-функцию. Небольшие изменения в параметрах алгоритма естественного отбора способны творить чудеса, превращая брюки в элегантные шорты, а вершину холма в середину склона.

**«Варп-скачок»**. Не забываем, что ландшафт приспособленности в виде холмов и долин – это всего лишь красивая метафора. На самом деле форма этой поверхности значительно сложнее, а живет она в многомерном пространстве. И мерность этого пространства зависит от числа изменяемых параметров системы, а значит от числа генов. Просто меняя количество генов в организме, можно добиться того, что счастливчик как в фантастической книжке уйдет в подпространство (или в надпространство) и там преодолеет досадный провал одним прыжком. Да, мы говорим про те самые хромосомные мутации, положительное влияние которых на скорость эволюции будет рассмотрено дальше в исследовательской части работы.

# **Глава 2. Программная часть**

## 2.1 Структура программы

Для начала, в программу загружается картинка *оригинал*, к которой впоследствии *популяция*, находящаяся внутри программы, будет стремиться. Но что такое популяция? Популяция – это набор организмов, которые как раз и участвуют в размножении, естественном отборе и остальных процессах эволюции. Каждый *организм* (или по-другому *организмус* или *Изоморф*, в тексте эти слова будут использоваться равноценно, как синонимы) состоит из *генов* – простейших фигур (эллипсов, прямоугольников и треугольников), которые обладают рядом свойств: тип фигуры (собственно, форма), цвет + прозрачность, положение, размер, позиция относительно других генов (выше или ниже). Совокупность генов создает картинку, которая соревнуется с оригиналом в похожести.

Для того чтобы сравнить картинки, в программе существует еще один объект – *Критикус*. Его задача – выбрать двух Изоморфовов из популяции, сравнить каждый с оригиналом, по итогам чего проставить каждой значение *difference* (то есть разница), и у какой оно будет больше, та «съедается», выкидывается из программы. Difference высчитывается так: каждому пикселю картинки сопоставляется пиксель образца, высчитывается квадрат разности по трем каналом RGB и сумма всех этих квадратов разностей и есть искомое значение.

## 2.2 Работа программы

Итак, пройдем по работе программы от начала и до конца. Первым делом мы инициализируем интерфейс (загружаем оригинальную картинку), а также создаем все объекты, которые будут работать далее: Критикус, популяция и каждый организм в этой популяции. После этого начинается естественный отбор – в цикле каждое поколение популяция проходит через стадию взросления, размножения, естественного отбора и некоторые другие (случайная смерть организмуса и смерть от старости), после чего выбирается лучший Изоморф и выводится на экран. Теперь подробнее про каждую стадию.

* **Взросление** или функция growUp().

Тут все просто – у каждого Изоморфаа есть свой возраст (количество поколений, которое он уже живет), и в момент вызова функции он увеличивается на 1.

* **Размножение** или функция reproduction().

Для начала определимся, что типов размножения есть два – половое и бесполое (почкование или клонирование). В программе реализовано оба способа, выбрать, какой будет использоваться, можно в settingах (настройках).

* + **Клонирование**. Программа проходит по всей популяции и для каждого Изоморфа производит следующие действия: создает некоторое количество организмусов (в settingах это numberOfChildren), являющихся точной копией родителя, производит над ними мутации (про это подробнее будет рассказано позднее) и добавляет к массиву Изоморфов. По итогу мы получаем популяцию, в которой часть организмусов – исходные родители, а остальные – их видоизмененные копии.
  + **Половое размножение**. Из всей популяции случайным образом выбирается два Изоморфа, один из которых условно нарекается мамой, а второй папой (условно, потому что на деле никаких гендерных отличий нет, и это разделение дает лишь возможность различать две особи между собой). После чего создается организмус child (ребенок), у которого первая половина\* генов копируется из первой половины генов мамы, а вторая из папы. Далее над ребенком, как и при почковании, проводятся мутации, и конечный результат записывается в конец массива Изоморфов. По итогу мы получаем популяцию, в которой есть некоторое количество n родителей (в settingах это число названо numberOfOrganizmusInPopulation, которое также является максимальным числом изморфов и используется в естественном отборе) + число детей равное n×numberOfChildren.   
    \*Деление идет не ровно пополам, а по определенной точке, остановка которой выбирается рандомно.

Теперь подробнее про **мутации**. Их тоже есть два типа, один из которых может использовать другой. Это мутация гена и мутация всего днк (совокупности генов, создающей организмус).

* **Мутация гена – э**то изменение его размера, цвета, положения, типа на некоторую величину (максимальная величина – maxDispersion) с некоторым шансом (chanceToMutateGene)
* **Хромосомная мутация** – это изменение количества генов путем удаления некоторых; добавления в конец мутированных копий имеющихся или рандомных; замены имеющихся на мутированные копии имеющихся или рандомные. Шанс такой мутации – chanceToMutateChromosome. Благодаря таким мутациям можно начинать эволюцию с одного гена в Изоморфе, это позволяет ускорить прохождение первой сотни поколений буквально до нескольких секунд.
* **Естественный отбор** или функция naturalSelection().

В цикле, пока размер популяции не станет снова равным numberOfOrganizmusInPopulation, Критикус сравнивает двух рандомных организмусов с оригиналом и выкидывает того, у которого difference больше. Оставшемуся же он записывает значение difference и свой ID, чтобы в будущем не тратить время на его просчитывание (это заметно ускоряет работу программы).

* **Смерть от старости.**

Когда Изоморф достигает определенного возраста, он умирает (удаляется из популяции)

* **Случайная смерть.**

С некоторым шансом Изоморф может просто умереть.

* **Выбор лучшего организмуса** или функция getTop()**.**

Лучшим становится тот организмус, у которого по прошествии всех остальных стадий оказался difference самым низким. Этот Изоморф и будет выведен далее на экран вместе со значением поколения и определенным difference. Но важный факт – этот difference будет отличаться от того, который существует в самом Изоморфе. Дело в том, что при увеличении размера Критикуса (о чем будет рассказано чуточку позднее), dfifference естественно тоже меняется, т.к. картинка, похожая на 8×8 пикселей внрезапно может стать совершенно не похожей, при ее же размере в 16×16 пикселей. Но чтобы не мучить зрителя этими скачками, на экран выводится значение difference для так называемого топКритикуса. Его размер = topCriticusImageSize (по умолчанию 512)

## 2.3 Дополнительные функции программы

По основным функциям все. Теперь пройдемся по некоторым дополнительным особенностям программы, которые помогают работать ей еще лучше и быстрее.

* **Увеличение размера Критикуса**

Когда Критикус сравнивает две картинки, то он сначала отрисовывает каждую на своем холсте внутри компьютера. Это занимает довольно много времени, так что рисовать картинку в ее оригинальном размере слишком затратно – все же Критикусу нужно нарисовать таких картинок больше, чем максимальное количество организмусов в популяции. Поэтому на первых этапах, когда картинку достаточно лишь приблизить к оригиналу, создать примерные очертания, размер Критикуса (criticusImageSize) можно сделать достаточно маленьким, например, 32. Это значит, что внутри себя он будет рисовать картинки размером 32×32 пикселя. Да, они будут без деталей, нечеткие, но это и не требуется на начальных этапах. Когда же для Критикуса размером 32 difference каждой картинки становится достаточно маленьким (становится меньше заданного значения criticusThreshold) – он удваивается, чтобы теперь Изоморфы повторяли более четкую картинку.

* **Изменение значения criticusThreshold** с ростом Критикуса

На ранних стадиях, когда размер Критикуса равен 8 или 16 (то есть он совершенно размыт), не хочется, чтобы программа пыталась повторить именно каждый пиксель этой в действительности не похожей на оригинал картинки. При этом это требуется на поздних стадиях, чтобы прорисовывать каждую деталь, каждый усик. Для этого было введено изменение criticusThreshold. Для Критикуса размером 8 он, допустим, равен 1000, до этого значения дойти очень просто, поэтому к моменту увеличения Критикуса, будут набросаны лишь основные цвета. Для следующего размера, criticusThreshold будет уже равен criticusThreshold÷criticusThresMultiplicator (стандартно делитель равен 1,5), то есть меньше, а значит до него дойти будет сложнее и программа добавит уже чуть больше детализации. К моменту, когда Критикус дойдет до 256, criticusThreshold будет уже около 150

* **Запись значения difference** каждому Изоморфу

Как уже было сказано, когда Критикус сравнивает картинку с оригиналом, проходит довольно много времени, поэтому делать это по несколько раз для одного и того же организмуса слишком расточительно. Поэтому после каждого сравнения полученное значение difference записывается в отведенное для этого поле в Изоморфе вместе со своим ID, чтобы в следующий раз при проверке этого же Изоморфа этим же Критикусом не считать второй раз. Для чего нужно передавать ID Критикуса? Так как Критикус может расти, то и difference, который он выдает, при проверке того же организмуса, изменится. Поэтому выросшему Критикусу присваивается новый ID, и, при проверке, казалось бы, известных организмусов, значение разности пересчитывается и записывается уже с новым ID.

* **Позиция** генов относительно друг друга

Часто бывает, что генов слишком много, и поэтому некоторые большие широкие и непрозрачные закрывают остальные, не позволяя тем повлиять на картинку и, быть может, улучшить ее, добавить ту саму ключевую деталь. Для того чтобы дать таким мелким генам возможность «проявить себя», им был добавлен параметр position, который может мутировать, благодаря чему гены с более высокой позицией окажутся выше. В ситуации, когда этого параметра нет, все гены рисуются по порядку их нахождения в массиве, и поэтому первые оказываются наглухо закрыты остальными.

* Возможность **«отключения»** генов

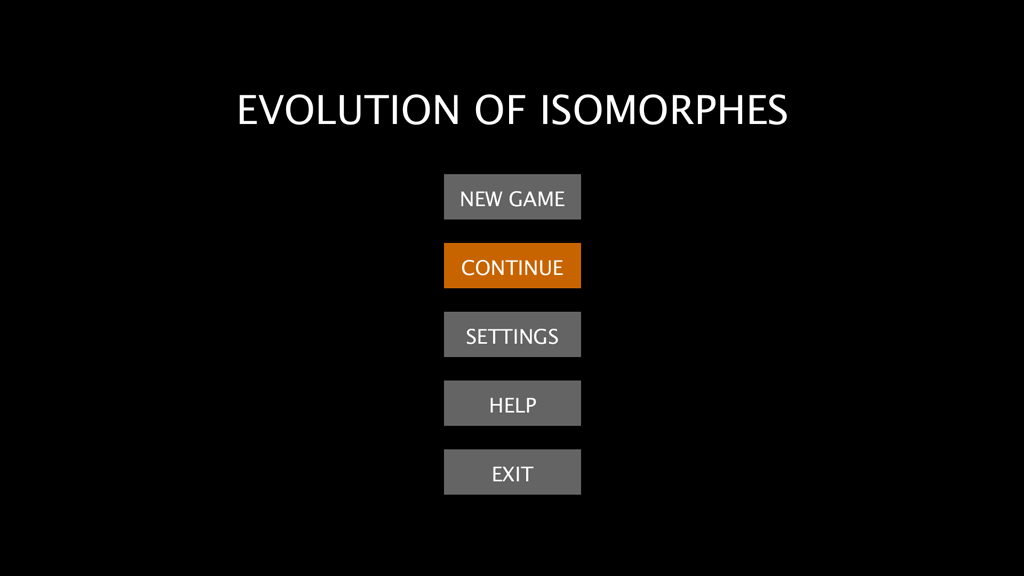
Иногда бывает, что в ходе естественного отбора, выигрывает организм, который вроде как лучше остальных, однако некоторые его гены абсолютно не вписываются. При этом если их мутировать, то лучше не станет, поэтому они остаются на своих местах и портят картину. Так что для того, чтобы безболезненно от них избавиться, был придуман еще один параметр – visible, если его значение true, то его видно, а если false, то он «отключен» и его не видно. При этом он так же может мутировать, и если в ходе эволюции он включится обратно (за это отвечает функция мутации гена), то внезапно может встать именно в то место, куда нам нужно.

## 2.4 Дружественный интерфейс

После разработки самой программы, когда все исследования (см. 3 главу)

были проведены, было принято решение создать удобный и понятный интерфейс, чтобы превратить программу в полноценное приложение. Для этих целей для начала были созданы с нуля класс кнопок и «ползунков», так как идеально подходящих под наши цели библиотек найдено не было. А теперь разберем по окнам, как выглядит интерфейс программы.

* **Игровое меню**

Итак, после запуска игры, меню – первое, что видит пользователь. Оно и верно, ведь именно оттуда можно попасть на все оставшиеся интересующие нас вкладки. Меню встречает красочной надписью «Evolution of Isomorphes», что является названием моей программы, а также обилием кнопок.

* **New Game (Новая игра)**

Самая главная кнопка. Позволяет создать новую популяцию и запустить эволюционный процесс. Нажав на нее, вы продвинетесь дальше на окошко с выбором целевой картинки, к которой будет стремиться популяция, и уже будете в шаге от запуска новой эволюции, где все уже в руках естественного отбора…

* **Continue (Продолжить)**

Возвращает игрока из меню к текущей эволюционирующей популяции. Или же подгружает сохраненную популяцию с диска, если эволюция еще не начата.

* **Settings (Настройки)**

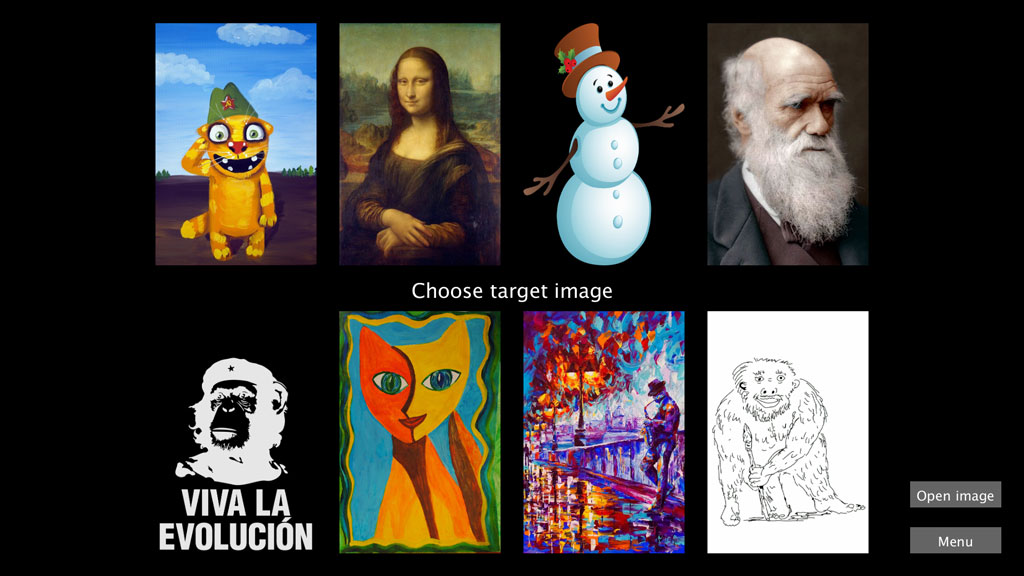
Вторая самая главная кнопка. В программе существует очень много различных настроек, которые радикально меняют скорость и направление эволюционного процесса. Параметры можно менять и даже в ходе эволюции, хотя в этом случае не все из них окажут влияние на идущий естественный отбор.

* **Help (Помощь)**

Эта кнопка переносит на сайт (html страничку - http://www.veronika.zelikov.name/evolution/help.html) в которой подробнейшим образом рассказано, как пользоваться программой, и обо всех аспектах игры. Информация оттуда частично повторяет информацию диплома, однако и несколько подробнее рассказывает, например, о том, как определенные настройки влияют на эволюционный процесс.

* **Exit (Выход)**

Кнопка, ответственная за выход из программы

* **Выбор целевой картинки**

Нажав пункт меню **«New Game»,** вы попадаете на следующий экран, где вам предстоит сделать самый главный выбор – определиться, к какой картинке будет стремиться естественный отбор.

На выбор предлагается 8 картин разных жанров и различной художественной ценности. Та картина, которая будет выбрана кликом мыши, станет «оригиналом»

Картины отличаются не только цветом, но и сложностью для генетического алгоритма. Некоторые эволюция сможет воспроизвести легко, быстро и очень подробно. Для других же потребуется подобрать более удобные настройки. Смотрите, пробуйте, экспериментируйте.

Кнопка **«Open Image»** позволяет открыть собственную картинку с диска.

Нажав на кнопку **«Menu»**, можно вернуться в главное меню игры.

* **Основное окошко игры**



Основное пространство окна занимают два изображения. Левое - это та картинка, которая была выбрана на предыдущем шаге. Именно с ним будет сверяться Критикус, чтобы оценить, кто из двух Изоморфов ему не нравится, и кого из них сегодня съесть.

Правое изображение - это портрет лучшего на данный момент Изоморфа в популяции. В каждом поколении лидер меняется, так что статичный портрет превращается в настоящее эволюционное кино-полотно, на котором из хаоса случайно движущихся пятен проявится узнаваемая картина.

**Статистика**

Справа от изображений размещается текстовая статистика, которая показывает текущее состояние популяции, Критикуса и лучшего Изоморфа.

* **Generation** - количество поколений Изоморфов, сменившихся с момента старта.
* **Difference** - численно выраженное различие между оригинальной картинкой и лучшим Изоморфом в популяции (с точки зрения Критикуса с идеальным зрением). Чем это различие ниже, тем больше Изоморф похож на любимую картину Критикуса. При значении 0 картинки становятся идентичными.
* **Current Dif.** - тот же самый Difference, но уже для реального подслеповатого Критикуса. Через слэш указан порог различия, после достижения которого Критикус улучшит свое зрение (увеличит свой размер).
* **Criticus view size** - размер картинки в пикселях, которую на самом деле видит Критикус. Именно этот размер увеличивается при достижении порогового значения из предыдущего пункта.
* **DNA** - размер генома лучшего Изоморфа. Каждый ген в геноме отвечает за рисование 1 пятнышка на правой картинке.
* **Generation time** - время, за которое происходит обсчет 1 поколения Изоморфов. С увеличением зоркости Критикуса и размера генома это время будет заметно расти, а программа работать все медленнее.
* **Total time** - общее время работы эволюционного алгоритма. При правильных настройках результат заметен уже через несколько минут работы, но для получения картинки очень большой точности могут потребоваться часы и даже дни.

**Пошаговое рисование**

Нажатие кнопки **«Step by step drawing»** переводит игру в режим пошаговой отрисовки. В этом режиме лучший организм популяции появляется на экране не весь целиком, а постепенно. Рисование начинается с пятна, закодированного первым геном, поверх него рисуется второе, потом третье и так далее до самого последнего пятна и последнего гена.

После рисования последнего гена игра вновь переходит в стандартный режим. Повторное нажатие на кнопку также возвращает игру в стандартный режим.

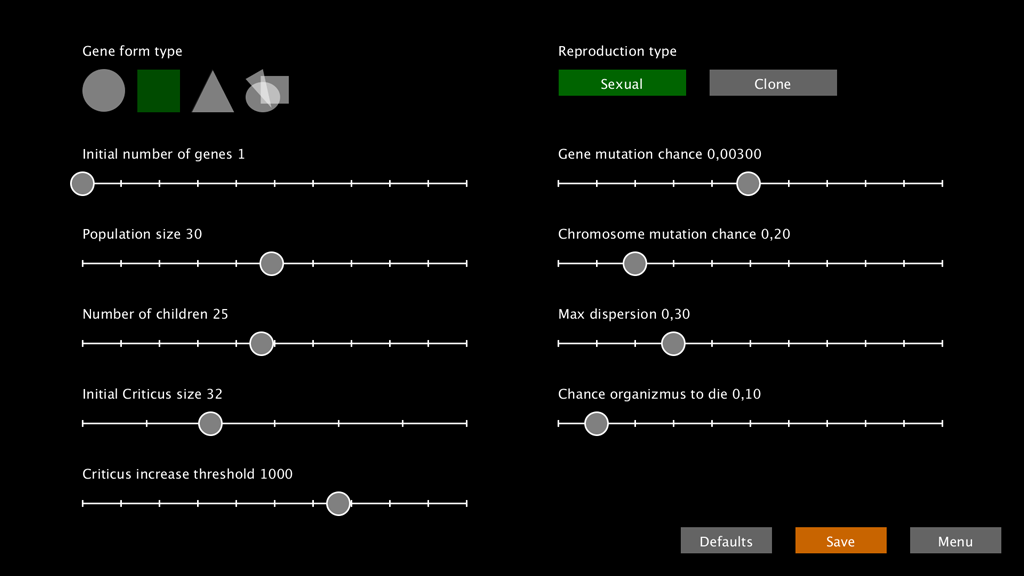
**Обзор эволюционной истории**

Следующая кнопка **«Review evolution»** позволяет ещё раз посмотреть, как проходила эволюция. Нажатие на кнопку переводит игру в режим исторического обзора и показывает по очереди всех лучших Изоморфов от первого до последнего поколения.

После рисования последнего поколения игра вновь переходит в стандартный режим. Повторное нажатие на кнопку также возвращает игру в стандартный режим.

Кнопка выхода в главное меню с одноимённым именем **«Menu»**, соответственно, позволяет выйти в главное меню где можно запустить новую игру или поменять настройки.

Для возврата к текущему эволюционному процессу можно использовать кнопку главного меню **«Continue»**, подсвеченную оранжевым цветом.

* **Settings – настройки**

У программы существует множество настроек, которые влияют на эволюционный процесс. Меняя их до начала эволюции, или же, что еще интереснее, в самом процессе, можно получить абсолютно различные результаты, как по времени, так и по количеству прошедших поколений. Для каждой картинки свой оптимальный набор параметров, подбираемый исключительно экспериментально.

Выставив определенные значения, пользователь может сохранить их кнопкой **«Save»** (если она горит оранжевым, значит, были произведены какие-то изменения).

Если прийти к оптимальным настройкам не получилось, эволюция не хочет идти дальше, процесс стоит на месте, то по кнопке **«Default»** можно всегда вернуться к стандартным настройкам, рекомендованным разработчиком для большинства ситуаций.

Подробнее про то, на что влияет каждый параметр, можно прочесть в Help или же в 3 главе данного диплома.

## 2.5 Мобильное приложение

Когда с интерфейсом для программы было покончено, было принято решение создать из программы мобильное приложение и выложить на Google Play. Благо, в Processingе предусмотрена функция конвертирования всего кода в программу для Android, поэтому, воспользовавшись этим, и изменив лишь пару функций, которые не поддерживаются на мобильных устройствах, приложение успешно попало и на мобильные платформы.

К сожалению, пришлось убрать кнопку, которая позволяет выбрать собственную картинку с диска, а также убрать некоторый функционал кнопки «Save» в настройках. Однако это не сильно повлияло на программу, и теперь вы можете скачать ее для своего телефона и протестировать все функции. Для этого всего лишь нужно иметь телефон с операционной системой Android, а также ссылку на приложение в Google Play - https://play.google.com/store/apps/details?id=evolution.of.isomorphes

А это ссылка на мой GitLab с исходниками кода моей программы и историей его создания: <https://gitlab.com/VenerinnaWolf/evolution-of-isomorphes>

# Глава 3. Исследования

## 3.1 Существующие настройки программы

Все необходимые функции написаны, естественный отбор работает как часы, результат стабильно радует похожестью на оригинал, но все же есть сомнения, а быть может, с другими настройками все будет работать еще быстрее?

Для начала, давайте определимся, какие вообще настройки существуют у нашей программы, и какие из них интересно поменять, чтобы посмотреть, как изменится результат.

|  |  |  |  |
| --- | --- | --- | --- |
| ***Название переменной*** | ***Русское описание*** | ***Стандартное значение*** | ***Примечание*** |
| formTypeSettings | Форма генерируемых генов | RECTANGLE | RECTANGLE – прямоугольники, ELLIPSE – эллипсы, TRIANGLE – треугольники, RANDOM – любые |
| reproductionType | Тип размножения | CLONE | CLONE – бесполое, SEXUAL – половое |
| formTypeChanceToMutate | Шанс мутации формы гена | 0,2 | Участвует только когда гены типа RANDOM |
| visiabilityChanceToMutate | Шанс мутации «видимости» гена | 0,2 |  |
| numberOfOrganizmus-InPopulation | Количество Изоморфовов в популяции на момент начала нового поколения | 100 |  |
| numberOfChildren | Количество детей у каждого Изоморфа | 5 |  |
| numberOfGenes-InOrganizmus | Количество генов в Изоморфе | 300 |  |
| chanceToMutateGene | Шанс мутации гена | 0,001 |  |
| chanceToMutate-Chromosome | Шанс мутации хромосомы | 0,1 |  |
| maxDispersion | Максимальная величина, на которую может мутировать ген | 0,2 |  |
| chanceToDie | Шанс случайной смерти Изоморфа | 0,1 |  |
| maxAge | Максимальный возраст Изоморфа | 5 |  |
| criticusImageSize | Изначальный размер Критикуса | 8 |  |
| topCriticusImageSize | Размер топКритикуса | 256 |  |
| criticusThreshold | Порог difference, после которого Критикус меняет свой размер | 1000 |  |
| criticusThresMultiplicator | Значение, на которое делится значение criticusThreshold при увеличении Критикуса | 1,5 |  |

Все эти параметры можно менять, получая разные результаты, сравнивать их, находить какие лучше, какие быстрее по времени, а какие достигают лучшего результата за наименьшее число поколений. Однако всевозможных комбинаций неизмеримо много, чтобы умесить все в один диплом, поэтому в данной работе я провела лишь исследования на самых важных, по моему мнению, параметрах. Стандартные настойки я оставляла везде одинаковыми, меняя лишь значение какой-то одной переменной и записывала значение difference на протяжении 10000 поколений, для того, чтобы потом построить график и сравнить, какое же значение самое оптимальное (доходит до наименьшего difference при одинаковом generation)

## 3.2 Функции, помогавшие в исследовании

Перед тем как перейти непосредственно к конкретным измерениям, стоит сказать пару слов о способе, которым записывались данные. Для того чтобы не отслеживать каждое поколение и не переносить данные вручную, была создана функция, которая каждое поколение записывала в массив значение difference (то, которое выводится на экран, т.е. для топКритикуса), а каждые 500 поколений создавала файл, в который записывалось все содержимое этого массива вместе с начальными настройками.

## 3.3 Графики различных измерений

Для измерения были выбраны параметры chanceToMutateGene, chanceToMutateChromosome, numberOfOrganizmusInPopulation и numberOfChildren; maxDispersion. По прогнозам, их изменение дало бы самую большую разницу в конечных результатах. В легенде указано значение изменяемого параметра и цвет линии, который соответствует полученному графику. По оси x расположен номер generation, по оси y значение difference, соответствующее ему. На каждом графике красная линия – это значение при стандартных настройках (с некоторыми исключениями для chanceToMutateGene\*).

* **chanceToMutateGene** – шанс мутации гена

Шанс мутации гена влияет на то, как часто будет мутировать организмус. Поэтому если он высок (1 или 0,1, то есть в каждом организме 100% или 10% генов точно будут мутировавшими), то из-за хаоса естественный отбор не справляется, и эволюция останавливается на определенном моменте. На графиках это хорошо видно.

При слишком низком шансе мутации (0,00001 или 0,0001 – при количестве генов 300 это значит, что на 100 Изоморфов будет ~ 3-30 мутировавших генов) эволюция протекает слишком медленно, что тоже не способствует положительному развитию. Хотя, есть вероятность, что в далеком будущем они все таким же темпом будут продолжать двигать эволюцию вперед, тогда как при более низких шансах она уже остановится, и в итоге получится более детализированная картинка. Но до этого придется ждать слишком долго.

Поэтому самыми оптимальными значениями оказались 0,01-0,001 (на каждого Изоморфа приходится по несколько мутаций). При этом для 0,01 на 10000-м поколении difference самый меньший (1246), однако можно заметить по изгибу линии, что скорость эволюции вскоре может пойти на спад. Тогда как для 0,001 она все еще достаточно велика, и быть может, в дальнейшем «перегонит» остальные. Так что выбор конечных настроек зависит от того, какого результата мы хотим добиться: лучше хорошо и быстро или еще лучше, но медленно.

\*На самом деле для проведения этого исследования были взяты не совсем стандартные настройки. Здесь были полностью отключены хромосомные мутации. Дело в том, что при хромосомных мутациях, добавляемые или заменяемые гены тоже часто мутируют, и это не контролируется параметром chanceToMutateGene. То есть, грубо говоря, изменение шанса мутации гена не дало бы такого разительного результата, потому что все бы сглаживала хромосомная мутация.

* **chanceToMutateChromosome** – шанс хромосомной мутации

Шанс хромосомной мутации определяет, как часто добавляются/заменяются/удаляются гены. Добавление хромосомной мутации вообще очень сильно улучшает работу программы, и чем больше ее влияние, чем «сходимость» - приближение к difference=0 быстрее. Что мы и видим на графиках: чем больше chanceToMutateChromosome, тем меньше difference для 10000-го поколения.

Дело в том, что программно ограничено, что при любом шансе хромосомной мутации, максимум может измениться лишь один ген в организме, поэтому нет ярко выраженного отрицательного эффекта от высоких значений, как при chanceToMutateGene.

В перспективе, можно некоторым образом изменить программу, чтобы при значениях больше 1, мутировало больше одного гена, и провести исследование для больших значений.

* **numberOfOrganizmusInPopulation** и **numberOfChildren** – количество Изоморфов в популяции и количество детей (жесткость естественного отбора)

Количество Изоморфов в популяции и их детей, а точнее их соотношение определяет, насколько жестким будет естественный отбор. Числа подобраны таким образом, чтобы *эффективный размер популяции (ЭРП)* в каждом случае был примерно равен 450. ЭРП - это количество организмусов, которое получается после стадии размножения (математически это numberOfOrganizmusInPopulation×(numberOfChildren+1)×(1- chanceToDie) и значит количество огранизмусов изначальное + количество их детей – количество организмусов, умершее от случайной смерти).

Однако, хоть ЭРП и одинаковый, естественный отбор оставляет в живых различное количество Изоморфов. Это количество всегда равно numberOfOrganizmusInPopulation, и поэтому чем оно больше, тем менее жесткий естественный отбор, и тем проще не самым хорошим организмусам выжить.

На графике все хорошо видно. При соотношении 400/1 (400 Изоморфов изначально и 1 ребенок от каждого), и при этом шансе случайной смерти равном 0,3, а не 0,1, как по стандарту, эволюция протекает медленнее всего. Это происходит, потому что нет такой жесткой конкуренции, чтобы выжить, и соответственно впоследствии размножаются не только лучшие из лучших, но и те, что похуже.

Напротив, самая большая скорость эволюции и соответственно наименьший difference на 10000-м поколении получается при соотношении 2/300! Это два родителя и 300 детей от каждого (напоминаем, размножение тут бесполое). При таком раскладе после каждого раза, когда Критикус проходится по популяции и съедает не годных ему Изоморфов, остаются лишь двое - самые лучшие - которые и передают будущим поколениям свои лучшие гены.

Здесь может крыться проблема отсутствия разнообразия, так как все Изоморфы являются детьми одного-двух практически одинаковых по генотипу. Однако практика показывает, что для этой программы эта проблема не играет роли и по итогу получается лучший результат.

Но возникает другая проблема – время. Да, с таким соотношением родителей/детей программа работает значительно дольше. Для сравнения, при соотношении 400/1, до 10000 поколений эволюция доходит за пару часов, тогда как при 2/300 за 2-3 дня! Это происходит из-за того, что в первом варианте в следующее поколение проходит много организмов с уже подсчитанным difference, а во втором случае таких особей лишь 2, а остальные 448 Критикусу приходится рисовать и обсчитывать раз за разом. На этапах, когда размер Критикуса уже превосходит 64, это занимает очень много времени, и каждое поколение длится по 5, 10, 20 секунд. Поэтому последний график не дочерчен до конца – работа программы была остановлена до того, как generation стало равняться 10000, ибо были опасения, что эволюция будет длиться слишком долго.

* **maxDispersion** – максимальная величина, на которую может мутировать ген

maxDispersion определяет, на какую максимальную величину относительно размера картинки может мутировать ген. То есть она определяет величину мутации. Это достаточно важный параметр, который в других алгоритмах может сильно повлиять на эволюцию. Однако, как видно из графика, в нашей реализации различие в difference не так уж велико. Самым лучшим значением оказывается стандартное 0,2.

# Заключение

По итогу работы была создана программа на основе генетических алгоритмов, постепенно создающая из хаотичного набора фигур картинку, максимально похожую на заданную изначально. По результатам исследований мною был определен оптимальный набор параметров, и именно они заложены в программу по умолчанию. Они позволяют уже через несколько минут работы программы из ничего получить Изоморфов правильной формы и цвета. Через час они с некоторого расстояния становятся неотличимы от оригинала. А несколько суток работы алгоритма и десятки тысяч поколений приводят к тому, что на картинке появляются уже мельчайшие детали, а в среднем каждый пиксель картинки отличается всего на 5-10 тоновых уровней.

По мере написания учебная программа вылилась в полноценный продукт с графическим интерфейсом и потенциалом к дальнейшему развитию. Мне даже удалось сделать мобильную версию. У нее немного урезана функциональность, зато каждый уже сейчас может скачать это приложение в Google Play - https://play.google.com/store/apps/details?id=evolution.of.isomorphes

# Список литературы

1. Онлайн курс по эволюции и генетическим алгоритмам [Электронный ресурс] – Режим доступа: <https://ulearn.me/Course/AIML/Evolyutsiya_7cdbc3b1-2131-4f1c-8435-033f574082e7>
2. Статья о моделировании эволюционного процесса на квантовых компьютерах [Электронный ресурс] – https://www.sciencealert.com/scientists-simulate-artificial-life-in-quantum-algorithm-for-first-time
3. Статья о довольно большом продвижении в разработке проекта моделирования искусственной жизни [Электронный ресурс] – <https://www.sciencealert.com/yeast-genome-synthesis-brings-scientists-another-step-closer-to-creating-artificial-life>
4. J. H. Holland. Adaptation in natural and artificial systems (. [В электронном виде можно прочитать здесь] - <http://bookre.org/reader?file=658089&pg=1>
5. Ричард Докинз. Слепой часовщик