МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«Тольяттинский государственный университет»

Институт математики, физики и информационных технологий
(наименование института полностью)

Кафедра «Прикладная математика и информатика»
(наименование)

02.03.03 «Математическое обеспечение и администрирование информационных систем»
(код и наименование направления подготовки, специальности)

Технология программирования
(направленность (профиль)/специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

На тему: «Разработка программного обеспечения для 3D визуализации работы генетического алгоритма»

Студент М.В. Буреев
(И.О. Фамилия) (личная подпись)

Руководитель к.ф.-м.н., О.В. Лелонд
(ученая степень, звание, И.О. Фамилия)

Консультант М.В. Дайнеко
(ученая степень, звание, И.О. Фамилия)

Тольятти 2020

АННОТАЦИЯ

Темой бакалаврской работы является «Разработка программного обеспечения для 3D визуализации работы генетического алгоритма».

В рамках выполнения бакалаврской работы было разработано программное обеспечения для 3D визуализации процесса поиска глобальных экстремумов функции с помощью генетического алгоритма. ПО генерирует 3D сцену, содержащую в себе поверхность исследуемой функции и решения, найденные на текущей итерации генетическим алгоритмом. При этом обеспечено интерактивное управления обзором на 3D сцену, что повышает наглядность при наблюдении за процессом поиска экстремумов функции. Также обеспечена возможность задания произвольных параметров генетического алгоритма.

Структура бакалаврской работы представлена введением, тремя разделами, заключением, списком литературы.

Во введении описывается актуальность проводимого исследования, дается краткая характеристика проделанной работы.

В первом разделе проводится анализ перспектив развития направления по визуализации алгоритмов.

Во втором разделе описываются принципы работы генетических алгоритмов.

В третьем разделе приведено описание разработанного программного обеспечения для визуализации генетического алгоритма.

В заключении представлены выводы по проделанной работе: генетические алгоритмы требуют большого изучения, определены их параметры, разработано ПО для 3D визуализации, и определены перспективы.

В работе использовано 38 рисунков, 11 формул, список литературы содержит 20 литературных источников. Общий объем выпускной квалификационной работы составляет 52 страницы.

ABSTRACT

The given graduation work is devoted to software development for 3D visualization of genetic algorithm.

In the present research, the practical aspects of the genetic algorithm operation process visualization are investigated.

The graduation work consists of an introduction, three chapters including 38 figures, 11 formulae, conclusions and a list of 20 references.

As part of the research, there was developed some software for 3D visualization of the process of searching for global function extrema using the genetic algorithm. The software generates a 3D scene containing the surface of the investigated function and solutions found on the current iteration by the genetic algorithm. At the same time, an interactive control of the view on the 3D scene is provided, which increases the visibility when observing the process of searching for function extrema. It is also possible to set arbitrary parameters of the genetic algorithm.

The introduction describes the relevance of the conducted research and presents a brief description of the work done.

The first chapter analyzes the prospects for the development of the algorithm's visualization.

The second chapter reveals the principles of the genetic algorithm's operation.

The third chapter is dedicated to describing the developed software for visualization of the genetic algorithms.

Overall, the results suggest that the genetic algorithms need to be practically studied. It is concluded that the main parameters of the genetic algorithms are defined, the corresponding software for 3D visualization is developed, and the prospects for further development are determined.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В связи с эпидемией коронавирусной инфекции COVID-19 Всемирной организацией здравоохранения была объявлена чрезвычайная ситуация международного значения в области здравоохранения. Обучение практически во всех высших учебных заведений было переведено на дистанционный формат. В связи с этим актуальной задачей является обеспечение визуализации учебного материала, предоставляемого студентам, на высоком уровне.

Одновременно с этим в теории оптимизации существуют такие алгоритмы, принцип действия которых проще понять путем рассмотрения примеров вычислений. К таким алгоритмам относятся генетические алгоритмы.

Поэтому целью данной работы является повышение наглядности обучающих материалов по теории оптимизации за счет 3D визуализации работы генетического алгоритма.

Для достижения данной цели в исследовании решались следующие задачи: анализ перспектив визуализации алгоритмов, анализ принципов работы генетического алгоритма, разработка программного обеспечения для 3D визуализации генетических алгоритмов.

В результате выполнения бакалаврской работы на языке программирования Scala было разработано программное обеспечения для 3D визуализации процесса поиска глобальных экстремумов функции с помощью генетического алгоритма. Программное обеспечение генерирует 3D сцену, содержащую в себе поверхность исследуемой функции и решения, найденные на текущей итерации генетическим алгоритмом. При этом обеспечено интерактивное управления обзором на 3D сцену, что повышает наглядность при наблюдении за процессом поиска экстремумов функции.

При необходимости возможна интеграция в программное обеспечение поддержки шлемов VR (виртуальной реальности).

1 АНАЛИЗ ПЕРСПЕКТИВ ВИЗУАЛИЗАЦИИ АЛГОРИТМОВ

Возможность получения качественного образования продолжает оставаться одной из наиболее важных жизненных ценностей граждан Российской Федерации. Стратегической целью в области образования во всем мире сегодня является повышения доступности качественного образования. Одним из вариантов совершенствования доступности образование является внедрение дистанционного обучения. Примером такого внедрения является проект «Росдистант», разработанный в рамках Федеральной инновационной площадки Министерства науки и высшего образования Российской Федерации.

Возникшая в 2019-2020 годах эпидемия коронавирусной инфекции COVID-19 также явилась мощным стимулом для развития дистанционного образования. Так, например, перевод студентов очной формы обучения Тольяттинского государственного университета на дистанционную форму обучения – явился одной из действенных мер по сдерживанию распространения COVID-19 без остановки учебного процесса.

Однако дистанционное образование невозможно без создания современных и полных учебных материалов. Наглядность учебных материалов является одной из важнейших характеристик для студента, которая влияет на удобство восприятия излагаемых тем. Необходимо учитывать, что дистанционное обучение предполагает возможность обучаться в любое время дня или ночи, даже когда невозможно удаленное общение с преподавателем в режиме реального времени. По этой причине обеспечение наглядности учебных материалов является одной из ключевых задач обеспечения дистанционного образования.

Предполагается, что современные IT-технологии могут значительно увеличить наглядность учебных материалов. Если раньше для изучения математики использовался статичный текст, формулы и графики, то теперь при обучении стало возможным использование анимированных графиков,

интерактивных расчетов и т.д. Кроме того, наличие интерактивных элементов увеличивает интерес к изучаемому предмету.

В теории оптимизации существует множество алгоритмов поиска экстремумов функций. Ускорить процесс понимания принципов работы алгоритмов можно путем разбора примеров по их практическому применению. Но в случае изучения генетического алгоритма это сделать затруднительно по следующим причинам.

Во-первых, генетический алгоритм является мета-эвристическим алгоритмом. Алгоритм состоит из фиксированной последовательности шагов, но каждый шаг может выполняться с использованием различных эвристик. Таким образом, эвристический алгоритм можно конфигурировать, комбинируя различные наборы операторов выбора родителей, скрещивания, мутации и отбора особей в новую популяцию. Более подробно математический аппарат генетического алгоритма рассматривается во второй главе [1-5].

Во-вторых, работа генетического алгоритма зависит от таких параметров, как количество особей в популяции, количество пар генерируемых для скрещивания на каждой итерации генетического алгоритма, количество процентов от лучших (более приспособленных) особей, участвующих в скрещивании, количество процентов от лучших особей переходящих в новую популяцию, вероятность мутации особи в процентах и т.д. Таким образом, учитывая предыдущий пункт, становиться ясно, что количество реализаций генетических алгоритмов с конкретным набором параметров очень велико. Причем выбор конкретной конфигурации для успешного решения частной задачи оптимизации является исследовательской задачей [6-10].

В-третьих, генетический алгоритм является итерационным стохастическим алгоритмом. Это означает, что заранее никогда не известно, какое количество итераций потребуется для решения той или иной задачи.

Эти проблемы можно решить, разработав интерактивное приложение, которое визуализирует процесс работы генетического алгоритма.

На основе вышесказанного можно заключить, что актуальной задачей является разработка программного обеспечения, позволяющего визуализировать процесс работы генетического алгоритма при решении задач оптимизации и демонстрировать влияние его параметров на результат и скорость поиска решений [11-15].

Цель работы ― повышение наглядности обучающих материалов по теории оптимизации за счет 3D визуализации работы генетического алгоритма.

Поставленная цель достигается путем последовательного решения следующих задач:

1. Анализ перспективности выполнение исследований по визуализации алгоритмов.

2. Анализ математического аппарата генетических алгоритмов и определение параметров, влияющих на результат их работы. А также выбор тестовых функций для визуализации работы генетических алгоритмов.

3. Разработка программного обеспечения для 3D визуализации работы генетического алгоритма с заданными пользователем параметрами.

2 ПРИНЦИПЫ РАБОТЫ ГЕНЕТИЧЕСКИХ АЛГОРИТМОВ

2.1 Описание генетического алгоритма

Генетический алгоритм относится к итерационным стохастическим алгоритмам и позволяет решать задачи оптимизации в условиях различных ограничений.

Генетический алгоритм относится к классу мета эвристических алгоритмов. Это означает, что алгоритм состоит из множества операторов, конфигурируя которые можно управлять процессом поиска решений. Схема работы генетического алгоритма показана на рисунке 2.1.

Основная идея работы алгоритма базируется на теории естественного отбора Дарвина. На первом этапе работы алгоритма производится генерирование решений задачи оптимизации случайным образом до тех пор, пока не будет заполнено множество решений заданного размера. На терминологии генетического алгоритма каждое решение задачи называется особью, а множество решений на текущей итерации алгоритма – популяцией. При этом значения целевой функции данных решений называются приспособленностью особи.

Пред началом выполнения каждой следующей итерации генетического алгоритма осуществляется проверка – «достигнут ли остановки?». Если в течение последних нескольких итераций не удалось улучшить показатель приспособленности лучшего решения, то генетический алгоритм завершает свою работу. В противном случае выполняется следующая итерация генетического алгоритма.

Внутри итерации над множеством решений проводится фиксированный набор действий, направленный на формирование новых решений (более приспособленных особей). Для этого на основе лучших решений случайным образом генерируются пары. Эти пары в терминологии генетических алгоритмов называются родителями.

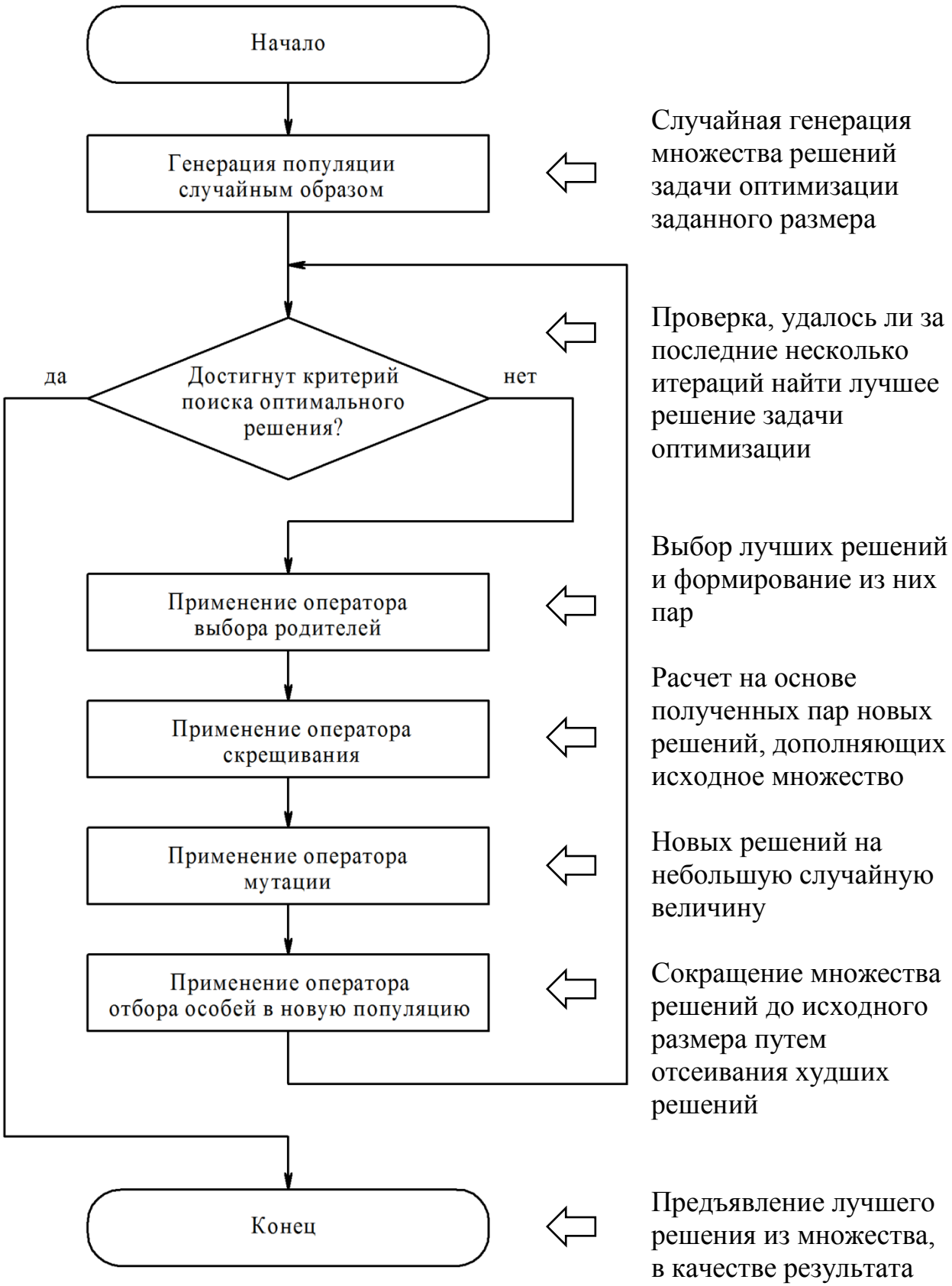

Рисунок 2.1 – Схема работы генетического алгоритма

Внутри пары решений производится комбинирование независимых переменных (скрещивание родителей) таким образом, чтобы получилось новое решение задачи оптимизации, включающее в себя фрагменты данных обоих родителей. Т.к. родители выбираются из лучших особей, то ожидается, что новое решение также будет не хуже (с точки зрения приспособленности) своих родителей. Решения, которые получаются в результате комбинирования пары других решений, называются потомками, а сама процедура комбинирования называется скрещиванием.

Для того чтобы результат скрещивания не был жестко привязан к паре исходных решений в алгоритм вводится процедура случайной мутации потомков. При этом в процентах задается как вероятность мутации потомка, так и сила мутации (сила изменения независимых переменных решения).

Рассмотрим подробнее каждый шаг генетического алгоритма.

Особь O_i включает в себя значения n независимых переменных. Количество n зависит от целевой функции:

$$O = (x_1, x_2, ..., x_n) \tag{2.1}$$

В одной популяции содержится заданное N количество особей, тогда популяцию Pop в зависимости от номера итерации t можно представить как:

$$Pop(t) = (O_1, O_2, ..., O_N) \tag{2.2}$$

Для генерации случайным образом первоначальной популяции необходимо выполнить действия представленные на (2.3):

$$\begin{gathered} i = 1...N \\ \begin{cases} O_i = (x_1, ..., x_n) \\ x_1 = random(\min(x_1), \max(x_1)) \\ x_2 = random(\min(x_2), \max(x_2)) \\ ... \\ x_n = random(\min(x_n), \max(x_n)) \end{cases}, \end{gathered} \tag{2.3}$$

где random (*A*, *B*) – функция, генерирующая случайное число в диапазоне от *A* до *B*; min(*x*) – функция, возвращающая минимальное значение области определения переменной *x*; max(*x*) – функция, возвращающая максимальное значение области определения переменной *x*.

Критерий остановки сформулируем так – выполнять поиск решения поиск до тех пор, пока все решения текущей популяции не локализуются в одной окрестности:

$$\frac{\sum_{i=1}^{N} f(O_i)}{N} \le T, \tag{2.4}$$

где *f* () – целевая функция, O_i – *i*-тая особь текущей популяции, *N* – размер популяции, *T* – параметр, задающий размер окрестности.

Выбор родителей для скрещивания будем осуществлять случайным образом:

$$\begin{cases} Pf = random(O_1,...,O_N) \\ Ps = random(O_1,...,O_N) \end{cases}, \tag{2.5}$$

где *Pf* и *Ps* – родители, $O_1,...,O_N$ - особи текущей популяции.

Расчёт потомков на основе родителей будем осуществлять с использованием множителя рекомбинации α, который выбирается случайным образом на участке от [0 – *d*; 1 + *d*] для каждого потомка. Затем каждый *i*-й ген особи-потомка *D* рассчитывается на основе генов родителей, так как это показано:

$$\begin{gathered} i = 1...n \\ \begin{cases} \alpha = random(-0,25;1,25), \\ D_i = Pf_i + \alpha \cdot (Ps_i - Pf_i) \end{cases} \end{gathered} \tag{2.6}$$

где *d* – заданный коэффициент рекомбинации. Под геном в генетических алгоритмах понимается независимая переменная (первый ген особи – значение первой независимой переменной и т.д.)

Мутация генов потомка задается так:

$$\begin{aligned} & j = 1...n \\ & \delta_j \leftarrow \text{random}([0.5 \cdot \min(x_j)], [0.5 \cdot \max(x_j)]) \\ & \alpha_j \leftarrow \text{random}(0,1) \\ & D_j = \begin{cases} D_j - \delta \ \ if \ \ \alpha_j \le 0.5 \\ D_j + \delta \ \ if \ \ \alpha_j > 0.5 \end{cases} \end{aligned}, \qquad (2.7)$$

где $\min(x_j)$ и $\max(x_j)$ - минимальное и максимальное значения области определения параметра x_j.

В процессе скрещивания размер популяции становиться больше первоначально заданного размера *N*. Текущую популяцию сортируют в порядке снижения приспособленности. В новую популяцию *Pop_new* попадают только *K* лучших особей, а остальные *N* – *K* особи генерируются случайным образом:

$$\begin{cases} i = 1...K \\ Pop\_new_i = Pop\_sort_i \end{cases}, \qquad (2.8)$$

где *N* – размер популяции, *Pop_sort* – отсортированная текущая популяция размером > N, *Pop_new* – новая популяция размером *N*, *i* – индекс особи.

Основными параметрами генетического алгоритма, необходимые для учета при программном моделировании его работы являются:

- Количество особей в популяции.
- Количество пар, генерируемых для скрещивания на каждой итерации генетического алгоритма.
- Количество процентов от лучших (более приспособленных) особей, участвующих в скрещивании.

- Количество процентов от лучших особей, переходящих в новую популяцию (остальные свободные места в популяции займут особи, сгенерированные случайным образом).
- Значение параметра‘d’ – коэффициент расчета параметров особи после скрещивания.
- Вероятность мутации особи в процентах.
- Сила мутации особи в процентах.

2.2 Выбор тестовых функций для визуализации

Теперь, когда было определены основные параметры генетического алгоритма, необходимые для учета при его программном моделировании, нужно выбрать тестовые функции, которые будут использоваться при визуализации. Для этого предложено использовать 3 самых известных тестовых функций:

1) Функция Растригина с двумя переменными (2.9):

$$f(x_1, x_2) = 20 + (x_1)^2 - 10 \cdot \cos(2\pi \cdot x_1) + (x_2)^2 - 10 \cdot \cos(2\pi \cdot x_2) \quad (2.9)$$

График функции, поострённый в Mathcad’e, представлен на рисунке 2.2.

2) Функция Швефеля с двумя переменными (2.10):

$$f(x_1, x_2) = 418{,}9829 \cdot 2 + (-x_1 \cdot \sin(\sqrt{|x_1|})) + (-x_2 \cdot \sin(\sqrt{|x_2|})) \quad (2.10)$$

График функции, поострённый в Mathcad’e, представлен на рисунке 2.3.

3) Функция Грайвенка с двумя переменными (2.11):

$$f(x_1, x_2) = 1 + \frac{1}{4000} x_1^2 + \frac{1}{4000} x_2^2 - \cos(x_1)\cos\left(\frac{1}{2} x_2 \sqrt{2}\right) \quad (2.11)$$

График функции, поострённый в Mathcad’e, представлен на рисунке 2.4.

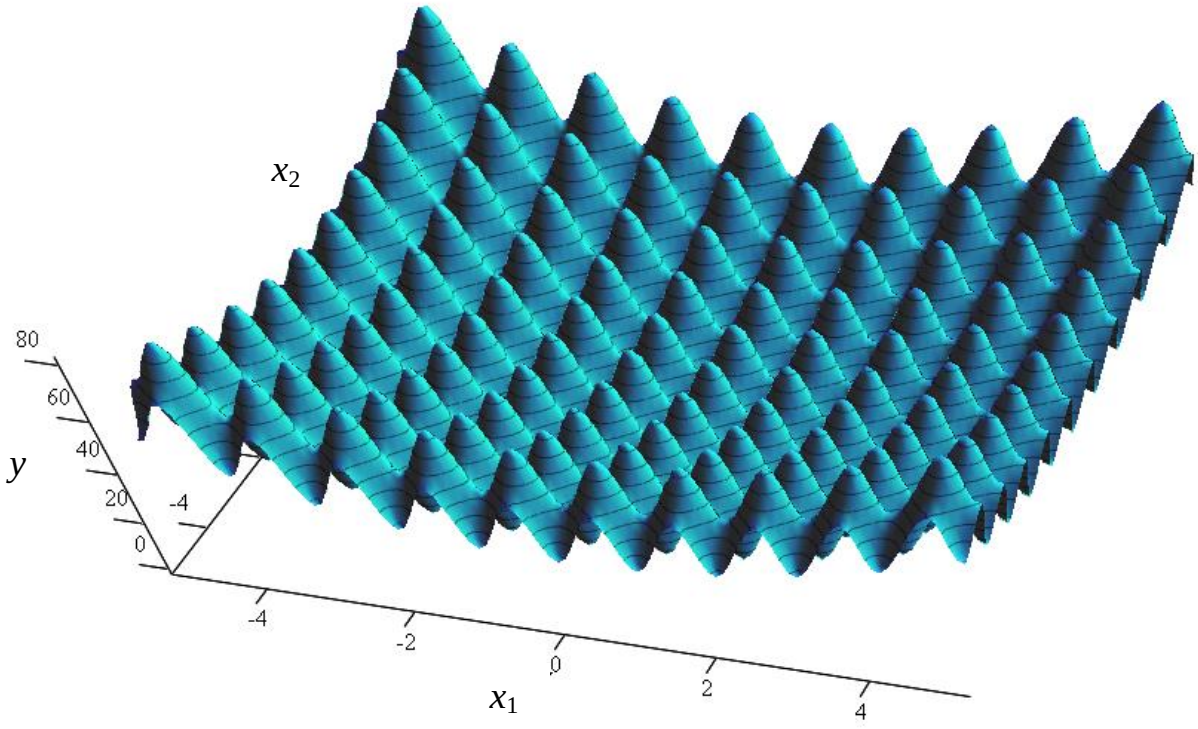

Рисунок 2.2 – 3D график функции Растригина

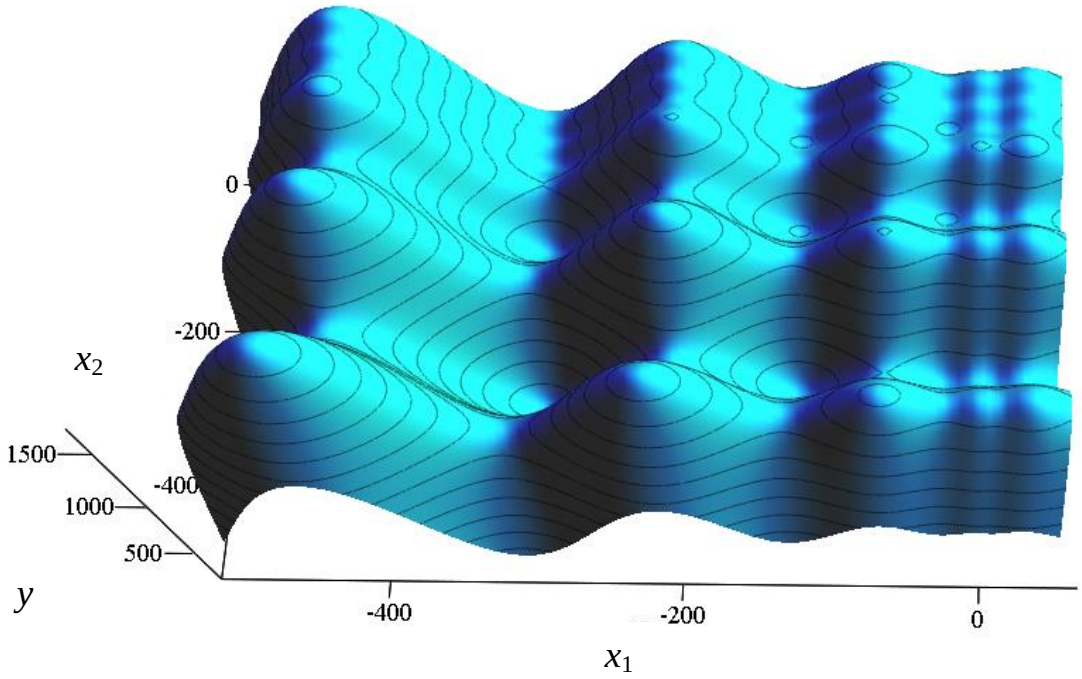

Рисунок 2.3 – 3D график функции Швефеля

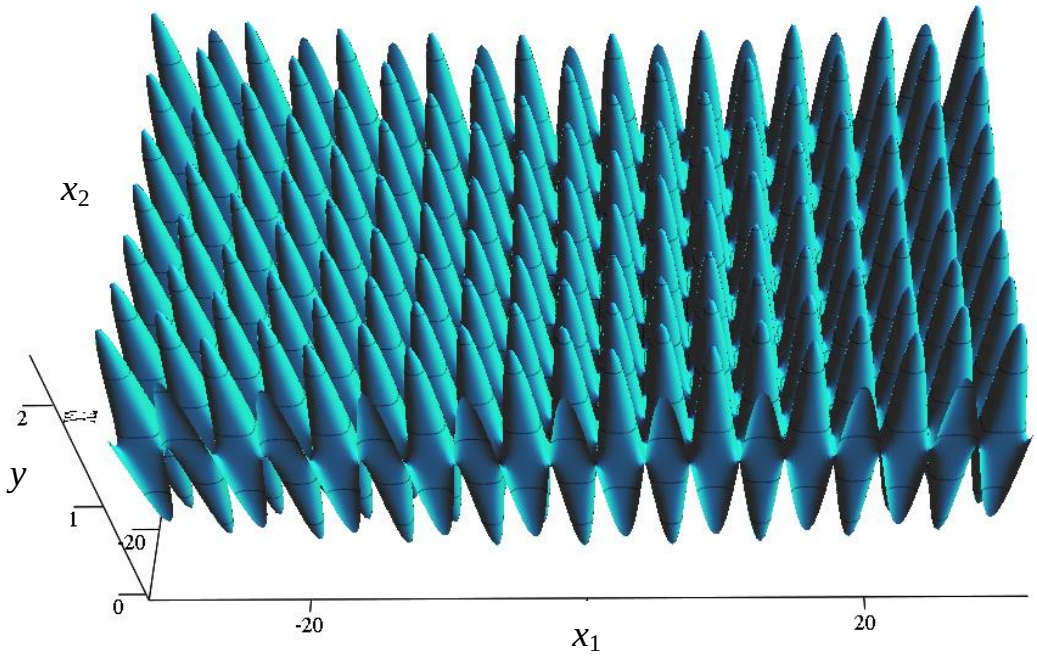

Рисунок 2.4 – 3D график функции Растригина

Результатами данного раздела являются:

- параметры генетического алгоритма, необходимые для учета при его программной реализации;
- набор тестовых функций, которые будут использованы при визуализации работы алгоритма.

3 ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ДЛЯ 3D ВИЗУАЛИЗАЦИИ

3.1 Функциональная особенность программной реализации

В ходе выполнения исследований в рамках бакалаврской работы было разработано программное обеспечения для 3D визуализации процесса поиска глобальных экстремумов функции с помощью генетического алгоритма.

Проект создавался с помощью интегрированной среды разработки IntelliJ IDEA. Функциональные особенности разработанного программного обеспечения следующие:

1) Наличие функционала для построения 3D сцены, содержащей в себе поверхность исследуемой функции и решения, найденные на текущей итерации генетическим алгоритмом. Решения отображаются на поверхности в виде сфер (точек).

2) Наличие возможности интерактивного управления камерой, обеспечивающей обзор 3D сцены.

3) Наличие функционала программного моделирования работы генетического алгоритма с разными параметрами. Обеспечена возможность задания таких параметров, как размер популяции на каждой итерации, количество пар генерируемых для скрещивания, процент лучших особей, участвующих в скрещивании, процент лучших особей попадающих в популяцию перед переходом на следующую итерацию (остальные места в популяции занимают особи, сгенерированные случайным образом), вероятность мутации особи после скрещивание и сила мутации.

4) Наличие возможности выбора одной из трех тестовых функций: функция Растригина от двух переменных, функция Швефеля от двух переменных, функция Грайвенка от двух переменных.

5) Наличие графического интерфейса.

3.2 Технические особенности программной реализации

Проект написан на языке программирования Scala, следовательно он, как и все Java-подобные языки, компилируется не напрямую в байт-код, а сначала в промежуточную стадию – в файлы, понятные для JVM (Java Virtual Machine).

Необходимые настройки компьютера и проекта перед сборкой и компиляцией проекта на локальной машине:

1) Версия JRE. Java Runtime Environment — минимальная реализация виртуальной машины, необходимая для исполнения Java-приложений, без компилятора и других средств разработки. Состоит из виртуальной машины — Java Virtual Machine и библиотеки Java-классов. Минимальная версия JRE, необходимая для запуска приложения – 1.8.0. В версиях ниже 1.8.0, отсутствуют открытые API, без которых работа проекта невозможна.

2) Версия Scala SDK. Software Development Kit — набор средств разработки, который позволяет специалистам по программному обеспечению создавать приложения для определённого пакета программ, программного обеспечения базовых средств разработки, аппаратной платформы, компьютерной системы, игровых консолей, операционных систем и прочих платформ. Необходимая версия Skala SDK – 2.12.8. В более поздних версиях упразднен некоторый функционал, на который опирается проект, что сделает невозможным сборку и запуск проекта.

3) Версия SBT. Scala Build Tool — система автоматической сборки для проектов, написанных на языках Scala и Java. SBT построена на принципах Apache Ant и Apache Maven, но предоставляет DSL (domain-specific language) на языке Scala вместо традиционной XML-образной формы представления конфигурации проекта. Необходимая версия SBT – 1.2.8.

4) Версия плагина Scala в среде разработки JetBrains. Плагин – независимо компилируемый программный модуль, динамически подключаемый к основной программе и предназначенный для расширения

и/или использования её возможностей. Версия плагина для разработки – 2019.3.26

3.2 Структура проекта

Проект состоит из нескольких основных директорий, таких как: gui, lib, project, src. Данные директории хранят в себе весь исходный материал и код проекта до компиляции.

Общая структура проекта представлена на рисунке 3.1:

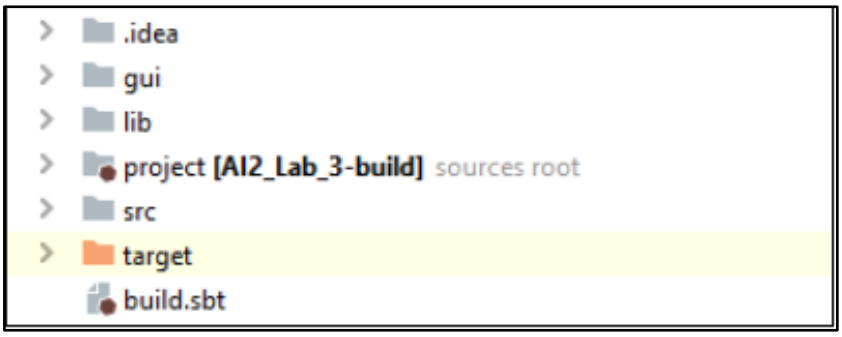

Рисунок 3.1 – Общая структура проекта

Директория gui представлена на рисунке 3.2. В данной директории содержатся картинки в формате png. С помощью данных картинок происходит отображение объектов в пользовательском интерфейсе программы.

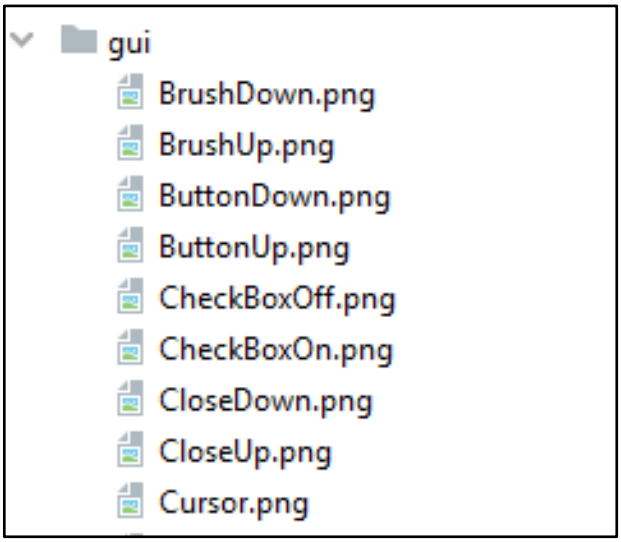

Рисунок 3.2 – Частичное содержание директории gui

Директория lib представлена на рисунке 3.3. Это стандартная директория любого проекта, не использующего сборщик проектов по типу Maven или Gradle, суть которых – хранение всех библиотек в своих централизованных репозиториях, а не в проектной директории и сборка проекта с помощью этих, выкаченных из репозиториев, библиотек.

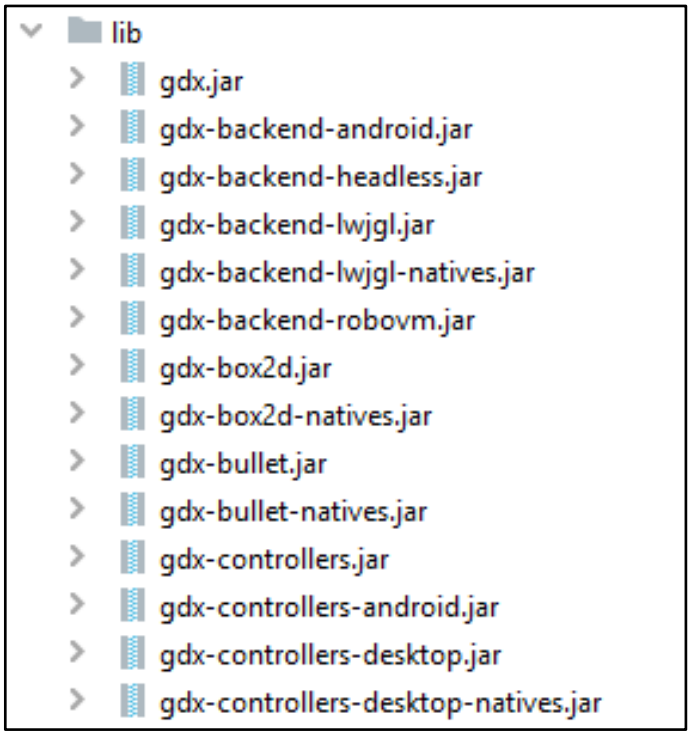

Рисунок 3.3 – Частичное содержание директории lib

Директория project предназначена для хранения служебной информации на этапе компиляции, свойств компилятора (версии, контрольных сумм, кэшей и т.д.) (рисунок 3.4).

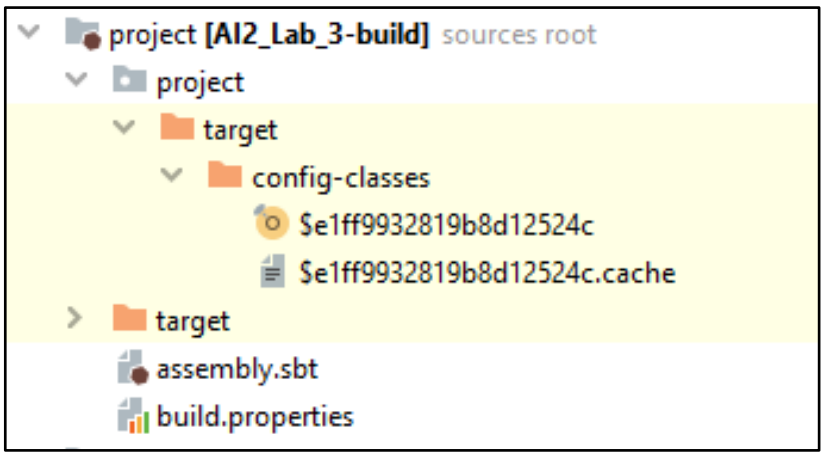

Рисунок 3.4 – Содержание директории project

Директория src – главная директория разработчика (рисунок 3.5). В ней содержатся все необходимые пакеты программы с классами, тестами и ресурсами. В данном проекте отладка программы производилась вручную, без написания Unit-тестов.

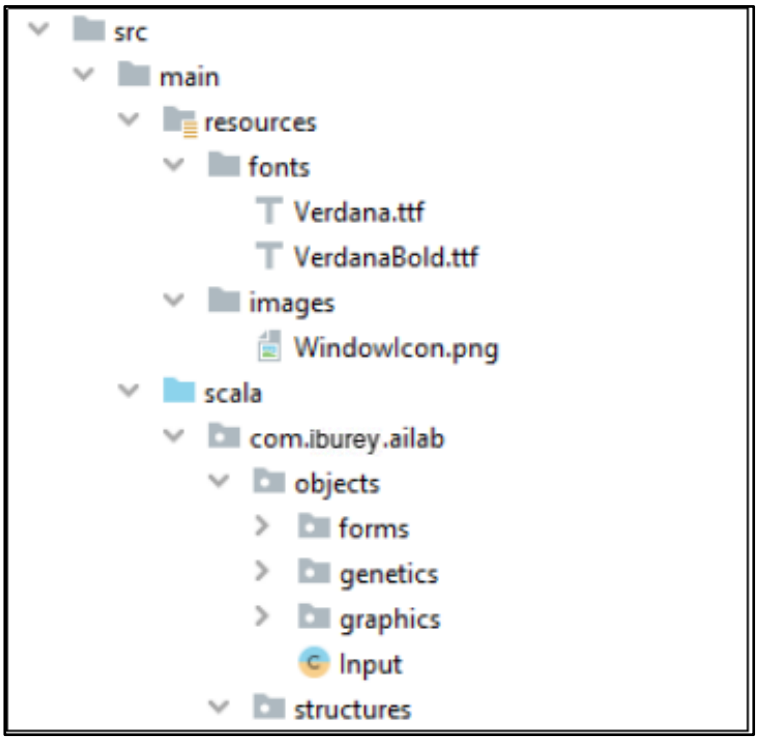

Рисунок 3.5 – Частичное содержание директории src

Во внутреннем пакете ресурсов хранится изображение иконки и 2 шрифта, используемых в пользовательском интерфейсе.

В главной папке проекта наравне с остальными директориями находится файл build.sbt (рисунок 3.6). Он необходим для сборщика проекта и содержит служебную информацию по проекту, такую как название программы, версию программы и версию Scala SDK.

```
1  name := "AI2_Lab_3"
2
3  version := "0.2"
4
5  scalaVersion := "2.12.8"
```

Рисунок 3.6 – Содержание служебного файла build.sbt

3.3 Классовая наполненность проекта

Все классы проекта находятся в директории src/main/scala и распределены по пакетам: objects, structures, utils (рисунок 3.7).

А также класс MainApplication, где происходит основная работа инициализации всех объектов, и объект Main, который является точкой входа в программу. В проекте каждый пакет агрегирует классы по определенным критериям (рисунок 3.8):

- structures – хранит в себе класс функции, отвечающий за построение и расчет графиков распределения;
- execute – содержит классы, отвечающие за выполнение команд пользователя;
- utils – предоставляет классы, содержащие в себе полезные методы, такие как математические, графические, по работе с файлами, и т.д.;
- base и forms – содержат элементы логики и отображения формы управления из графического интерфейса;

- genetics – хранит объекты и классы, отвечающие за элементарные сущности логики программы (генетические единицы);
- graphics – содержит класс камеры, на сцене пользовательского интерфейса.

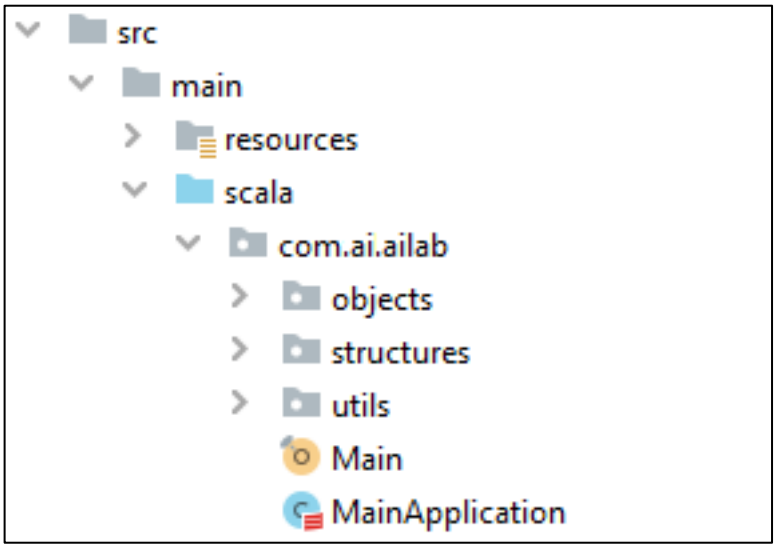

Рисунок 3.7 – Структура пакетов проекта

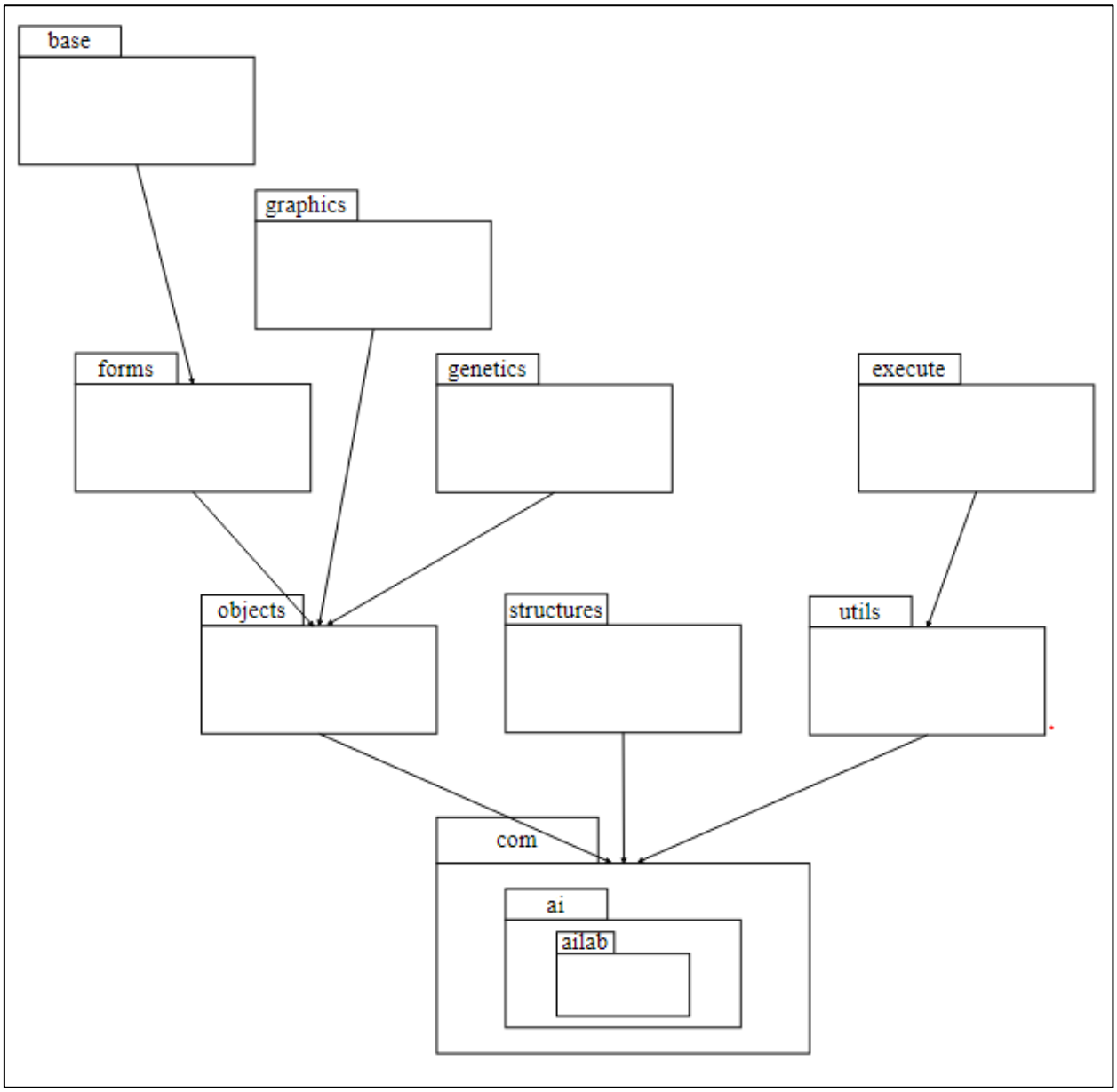

Рисунок 3.8 – UML-диаграмма пакетов

Классы связаны друг с другом посредством создания объектов классов через конструкцию new, а объекты через непосредственный вызов объекта. Наследований, имплементаций и зависимостей в проекте нет, за исключением пакета objects/forms.

Ниже представлена UML-диаграмма классов проекта (рисунок 3.9)

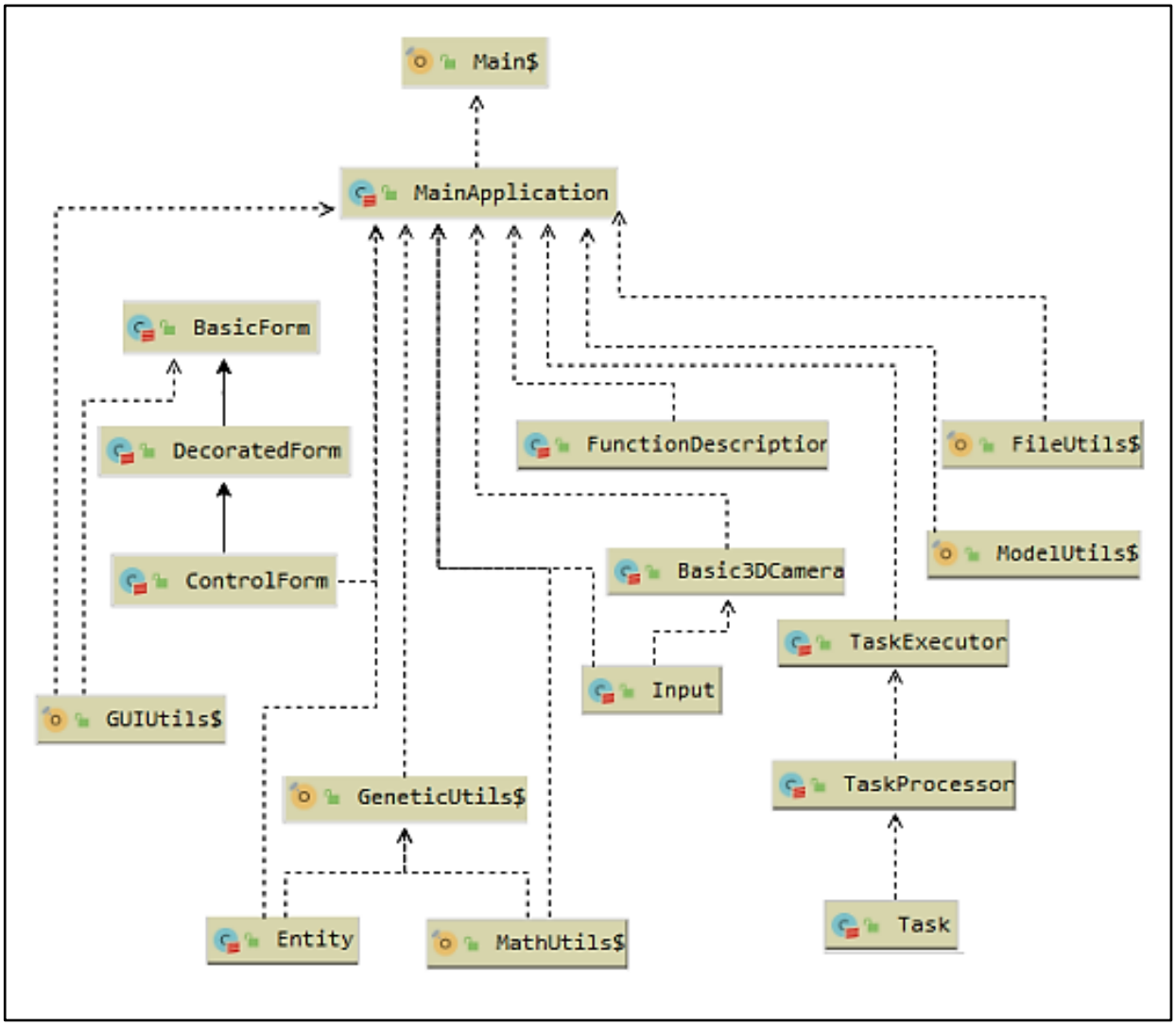

Рисунок 3.9 – Диаграмма связи классов проекта

Большинство связанных классов находится в одном и том же пакете, что свидетельствует о правильной организации проекта.

Как видно из диаграммы – основная работа программы по инициализации и настройке классов происходит в классе MainApplication, который, в свою очередь, вызывается из объекта Main.

Далее представлен разбор каждого класса в отдельности с ветвей древовидной схемы до корня.

Объект FileUtils находится в пакете utils. Данный объект предназначен для работы с файлами: чтение из файла, запись в файл, загрузке текстур из

файла, загрузки шрифтов из файла и многие другие перегруженные методы, принимающие на вход различные параметры.

Объект FileUtils используется только в классе MainApplication, для загрузки всех изображений из директории gui в хранилище текстур и загрузки всех шрифтов в хранилище текстур.

Диаграмма методов объекта представлена на рисунке 3.10.

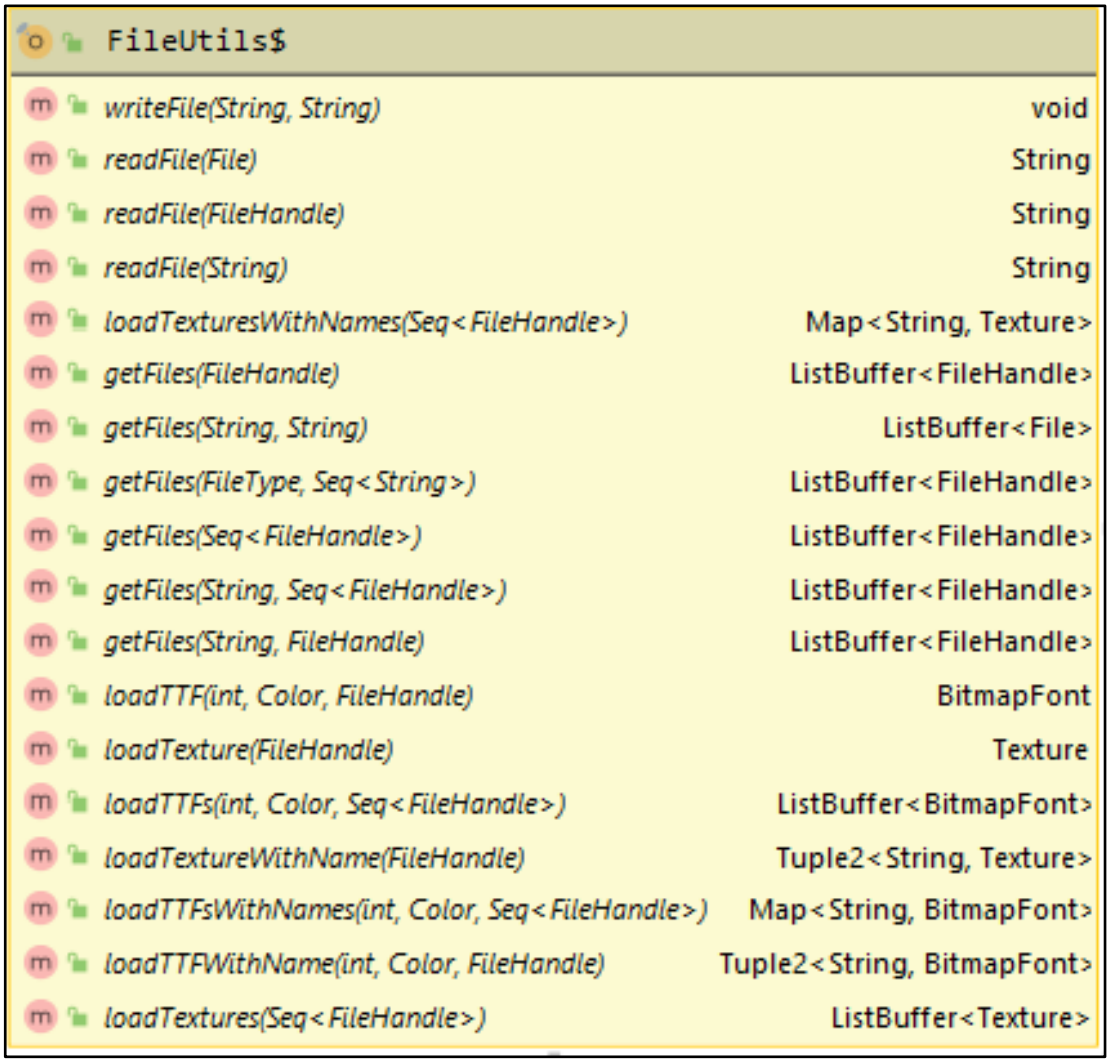

Рисунок 3.10 – Объект FileUtils

Так как данный класс является объектом, то его вызов происходит без конструкции new, а вызывается напрямую, как статичный класс. Пример методов для загрузки файлов (рисунок 3.11):

```
def getFiles(directories: FileHandle*): ListBuffer[FileHandle] = {
  val result = new ListBuffer[FileHandle]
  directories.foreach(directory => {
    result ++= getFiles(directory)
  })
  result
}

def getFiles(text: String, directory: FileHandle): ListBuffer[FileHandle] = {
  val result = new ListBuffer[FileHandle]
  if (directory.isDirectory) {
    val items = directory.list()
    if (items != null) {
      for (item <- items) {
        if (item.isDirectory) {
          result ++= getFiles(text, item)
        } else {
          if (item.name().contains(text))
            result += item
        }
      }
    }
  }
  result
}
```

Рисунок 3.11 – Методы объекта FileUtils

На рисунке 3.11 представлен листинг перегрузки метода по загрузке файлов с директории. Первый метод рекурсивно проходит по всем директориям переданной директории и загружает все файлы в хранилище. Второй метод производит рекурсивный поиск файла по имени внутри переданной директории, возвращая лист файлов.

Объект ModelUtils находится в пакете utils. Данный объект предназначен для создания трехмерных моделей в пользовательском интерфейсе накладывания на объекты текстур и создания текстур из собственной базы материалов из класса com.badlogic.gdx.graphics.g3d.Material.

Объект ModelUtils используется только в классе MainApplication для создания текстур и объектов пользовательского интерфейса.

Диаграмма методов объекта представлена на рисунке 3.12:

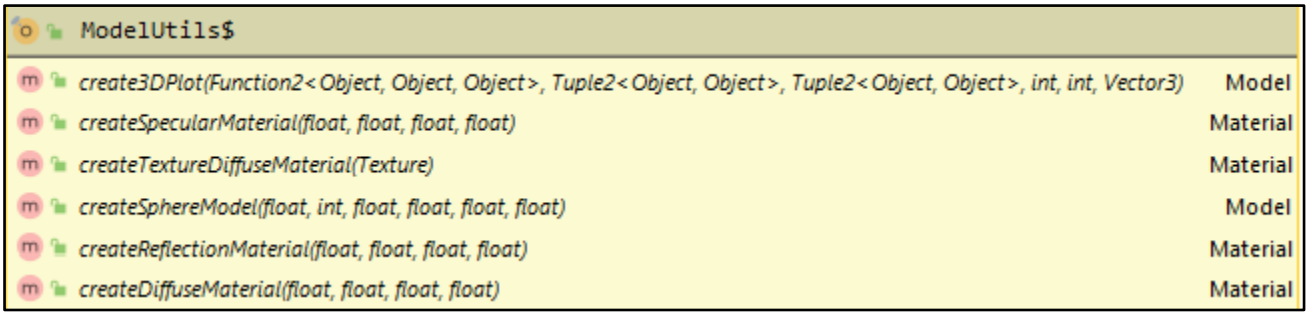

Рисунок 3.12 – Объект ModelUtils

Так как данный класс является объектом, то его вызов происходит без конструкции new, а вызывается напрямую, как статичный класс. Пример методов для создания материала и трехмерного графика (рисунок 3.13):

```
def createTextureDiffuseMaterial(texture: Texture): Material =
  new Material(TextureAttribute.createDiffuse(texture))

def create3DPlot(function: (Float, Float) => Float,
                 dx: (Float, Float), dy: (Float, Float),
                 xParts: Int, yParts: Int,
                 scale: Vector3 = new Vector3(1f, 1f, 1f)): Model = {
  val modelBuilder = new ModelBuilder

  val xPartSize = (dx._2 - dx._1) / xParts
  val yPartSize = (dy._2 - dy._1) / yParts

  modelBuilder.begin()
  for (i <- 0 until xParts) {
    modelBuilder.node()
    val meshBuilder = modelBuilder.part(
      i.toString,
      GL20.GL_TRIANGLES,
      VertexAttributes.Usage.Position | VertexAttributes.Usage.Normal,
      createDiffuseMaterial(0.58f, 0.57f, 0.5f)
```

Рисунок 3.13 – Методы объекта ModelUtils

На рисунке 3.13 представлен листинг методов по созданию рассеивающего материала, текстуры рассеивающего материала и трехмерного графика по функции двух параметров.

Класс FunctionDescription – единственный класс в пакете structures. Данный класс предназначен для расчета графика по функции 2-х параметров, получаемой в конструкторе, верхним и нижним границам каждой переменной и вектора масштаба.

FunctionDescription используется только в классе MainApplication, для создания трех трехмерных трафиков в графическом интерфейсе пользователя.

Данный класс имеет только конструктор, при этом не имеет методов и атрибутов. Вызов в классе MainApplication происходит с помощью конструкции new. Далее приведены примеры вызовов класса (рисунок 3.14):

```
new FunctionDescription((x, y) => {
  (20d + x*x-10d*cos(2d*PI*x) + y*y-10d*cos(2d*PI*y)).toFloat },
  (-10f, 10f), (-10f, 10f), new Vector3(30f, 3f, 30f)),

new FunctionDescription((x, y) => {
  (-x*sin(sqrt(abs(x))) -y*sin(sqrt(abs(y)))).toFloat },
  (-1000f, 1000f), (-1000f, 1000f), new Vector3(1f, 1f, 1f)),

new FunctionDescription((x, y) => {
  (1d + x/4000d + y/4000d - cos(x) * cos(y/sqrt(2d))).toFloat },
  (-30f, 30f), (-30f, 30f), new Vector3(10f, 10f, 10f))
```

Рисунок 3.14 – Вызовы класса MainApplication

Три функции используют конструкторы класса для инициализации трех различных функций поверхности.

Объект GUIUtils находится в пакете utils. GUIUtils – главный объект работы с пользователем. В задачи данного класса входит: создание кнопок, полос загрузки, текстовых полей, изображений, сцен, чек-боксов, надписей и т.д.

Объект GUIUtils используется в классе MainApplication, для создания объектов пользовательского интерфейса, а в классе BasicForm для создания стиля заднего фона.

Диаграмма методов объекта представлена на рисунке 3.15:

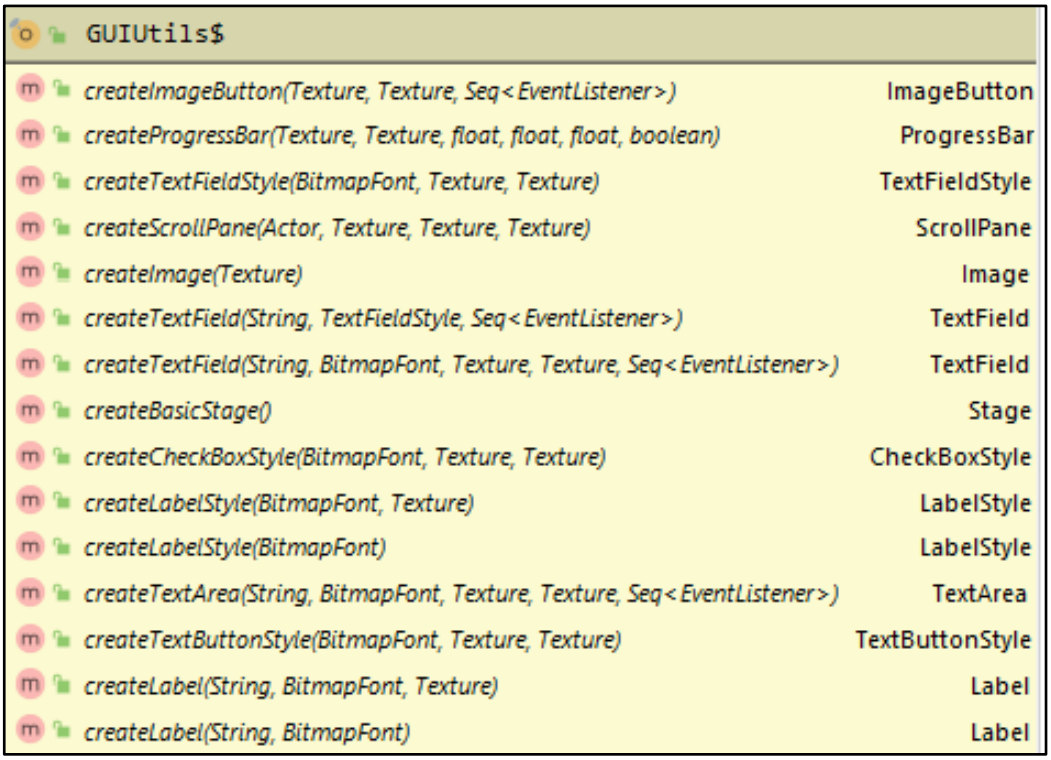

Рисунок 3.15 – Объект GUIUtils

Так как данный класс является объектом, то его вызов происходит без конструкции new, а вызывается напрямую, как статичный класс. Примеры методов представлены на рисунке 3.16:

```
def createTextArea(text: String,
                   font: BitmapFont,
                   background: Texture,
                   cursor: Texture,
                   events: EventListener*): TextArea = {
  val textFieldStyle = new TextFieldStyle()
  textFieldStyle.background = new TextureRegionDrawable(background)
  textFieldStyle.font = font
  textFieldStyle.fontColor = font.getColor
  textFieldStyle.cursor = new TextureRegionDrawable(cursor)
  val textArea = new TextArea(text, textFieldStyle)
  events.foreach(event => textArea.addListener(event))
  textArea
}
```

```
def createImageButton(up: Texture,
                      down: Texture,
                      events: EventListener*): ImageButton = {
  val style: ImageButtonStyle = new ImageButtonStyle()
  style.up = new TextureRegionDrawable(up)
  style.down = new TextureRegionDrawable(down)
  val button = new ImageButton(style)
  events.foreach(event => button.addListener(event))
  button
}
```

Рисунок 3.16 – Методы объекта GUIUtils

На рисунке 3.16 представлен листинг методов по созданию текстового поля и кнопки. Большое количество методов в объекте являются перегрузочными методами, которые принимают другие атрибуты на вход, но отдавая неизменяемые объекты на выходе.

Класс BasicForm находится в пакете objects\forms\base. Сутью класса является инициализация контейнера, который содержит все объекты графического интерфейса пользователя.

Данный класс является родительским классом для класса DecoratedForm.

BasicForm имеет конструктор, принимающий на вход начальные координаты, ширину, высоту и текстуру для заднего фона и метод container, который возвращает объект управляющего меню (рисунок 3.17).

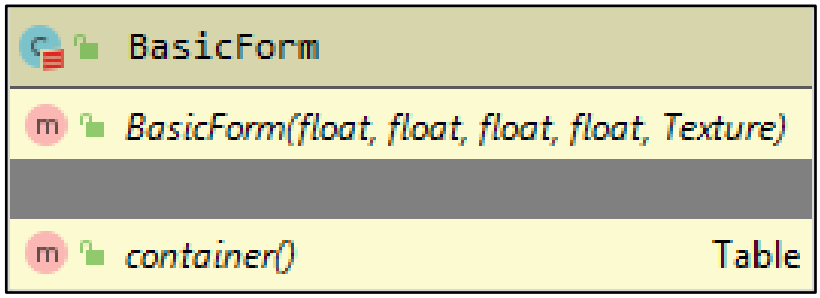

Рисунок 3.17 – Класс BasicForm

Так как данный класс нужен только как родительский, поэтому его вызов напрямую не производится. Ниже приведен листинг класса (рисунок 3.18):

```
class BasicForm (x: Float = 0f, y: Float = 0f,
                 width: Float = 800f, height: Float = 600f, background: Texture = null)
  // Create
  val container: Table = new Table().top().pad( pad = 5f)


  // Style
  if (background != null) container.setBackground(GUIUtils.toDrawable(background))


  // Position
  container.setSize(width, height)
  container.setPosition(x, y)
  container.addListener(new DragListener() {
    override def drag(event: InputEvent, x: Float, y: Float, pointer: Int): Unit = {
      container.moveBy(x - container.getWidth / 2, y - container.getHeight / 2)
    }
  })
}
```

Рисунок 3.18 – Листинг класса BasicForm

Из рисунка 3.18 видно, что все данные, принимаемые в конструкторе, присваиваются объекту container, а сам конструктор не имеет никакой логики.

Класс DecoratedForm находится в пакете objects\forms\base. Данный класс предназначен для надстройки над базовыми атрибутами формы из класса BasicForm.

Класс DecoratedForm является родительским классом для класса ControlForm. Данный класс имеет конструктор, принимающий на вход все координаты родительского класса, а также некоторые объекты прокрутки, надписей и заголовка, и методы, возвращающие объекты для меню управления (рисунок 3.19).

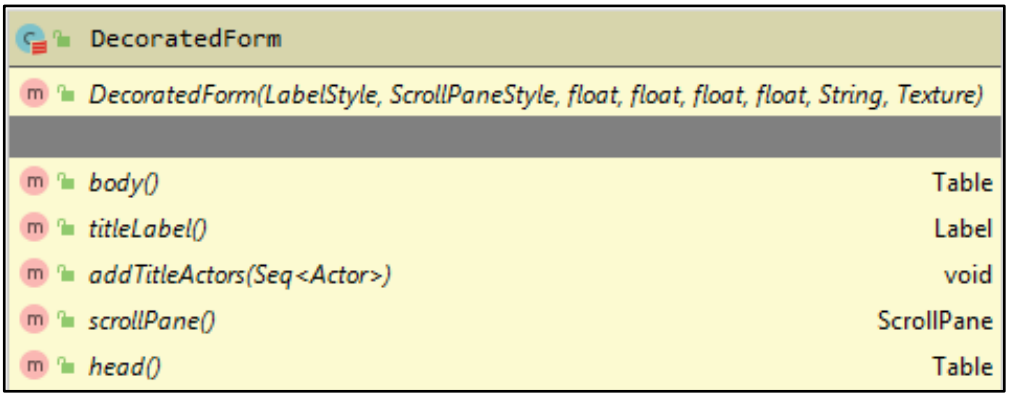

Рисунок 3.19 – Класс DecoratedForm

```
// Create
val head: Table = new Table().left().top()
val body: Table = new Table().top().pad( top = 0f,
   left = 5f,
   bottom = 5f,
   right = 5f)
val scrollPane: ScrollPane = new ScrollPane(body, scrollPaneStyle)
val titleLabel: Label = new Label(title, titleLabelStyle)


// Position
head.add(titleLabel).expandX().left()
container.add(head).expandX().fill().padBottom( padBottom = 5f)
container.row()
container.add(scrollPane).expand().fill()

def addTitleActors(actors: Actor*): Unit =
  actors.foreach(titleActor => head.add(titleActor).padLeft( padLeft = 5f))
```

Рисунок 3.20 – Листинг класса DecoratedForm

Данный класс используется только как надстройка над базовой формой. Ниже приведен листинг класса (рисунок 3.20).

При сравнении с предыдущим классом видно, что внутри DecoratedForm создаются объекты прокрутки, надписей, тела и заголовка, которые в свою очередь заполняют объект container родительского класса.

Класс ControlForm находится в пакете objects\forms.

Предназначение класса – окончательная модификация объекта container, который передается в Main класс для отображения в пользовательском интерфейсе в виде панели управления программой.

Класс ControlForm является последним в цепочке наследования BasicForm -> DecoratedForm -> ControlForm и вызывается в классе MainApplication через конструкцию new.

Данный класс имеет конструктор, принимающий на вход все координаты родительского класса, а также методы для заполнения контейнера.

Диаграмма методов класса представлена на рисунке 3.21.

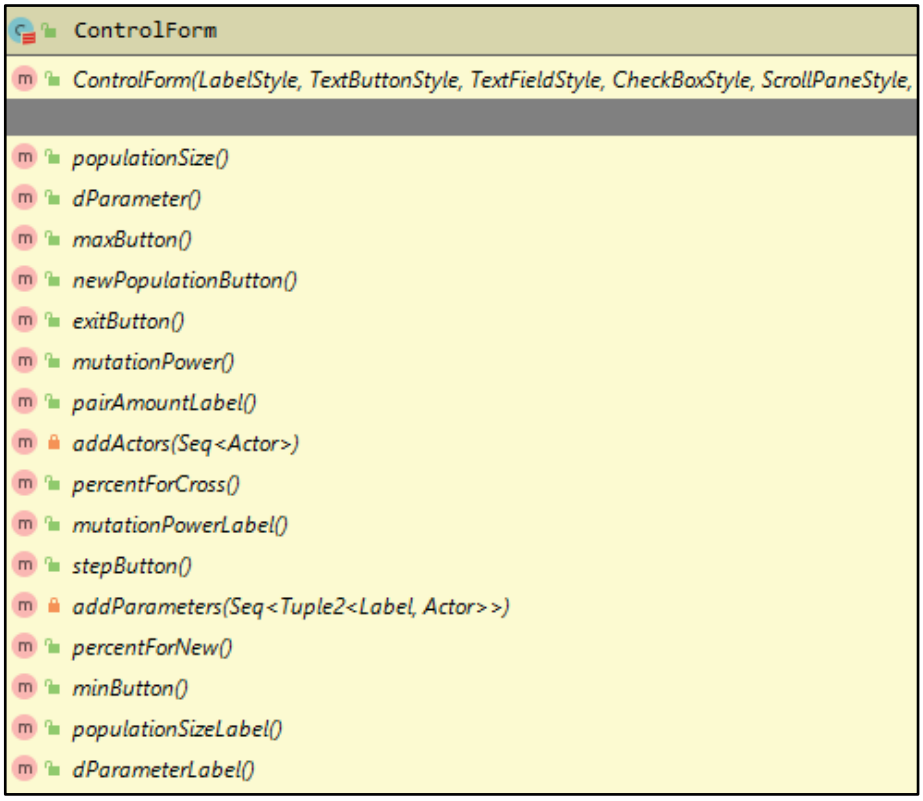

Рисунок 3.21 – Класс ControlForm

Ниже приведен частичный листинг класса (рисунок 3.22):

```
class ControlForm (labelStyle: LabelStyle,
                   textButtonStyle: TextButtonStyle,
                   textFieldStyle: TextFieldStyle,
                   checkBoxStyle: CheckBoxStyle,
                   scrollPaneStyle: ScrollPaneStyle,
                   x: Float = 0f, y: Float = 0f,
                   width: Float = 480f, height: Float = 720f,
                   background: Texture = null) extends
  DecoratedForm (labelStyle, scrollPaneStyle, x, y, width, height,
    title = "Control form", background) {
  // Create
  val populationSize = new TextField( text = "100", textFieldStyle)
  val pairAmount = new TextField( text = "25", textFieldStyle)
  val percentForCross = new TextField( text = "20", textFieldStyle)
  val percentForNew = new TextField( text = "20", textFieldStyle)
  val dParameter = new TextField( text = "0.25", textFieldStyle)
  val mutationChance = new TextField( text = "90", textFieldStyle)
  val mutationPower = new TextField( text = "10", textFieldStyle)
```

Рисунок 3.22 – Листинг класса ControlForm

Из листинга виден конечный результат кастомизации меню управления: тестовые поля, чек-боксы и т.д.

Класс Task находится в пакете utils\execute. Данный класс является оболочкой любого действия, произведенного пользователем.

Класс Task вызывается только в классе TaskProcessor, в котором происходит обработка действий пользователя.

Данный класс имеет список с упорядоченными по времени вызовами класса, что обеспечивает последовательное выполнение всех действий. Диаграмма методов класса представлена на рисунке 3.23.

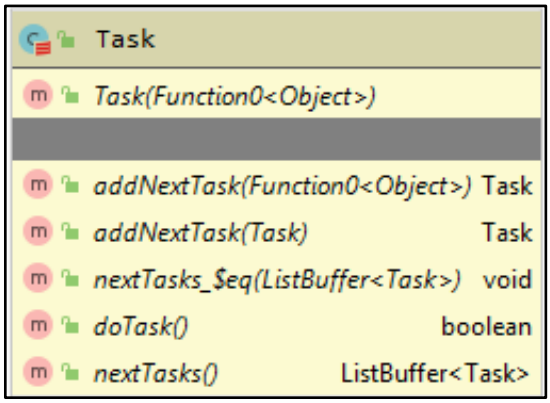

Рисунок 3.23 – Класс Task

Ниже приведен частичный листинг класса. В методах addTask видно, что все последующие задания используют список объекта. Задания добавляются в конец списка для сохранения необходимой последовательности (рисунок 3.24).

```
class Task (task: () => Boolean) {

  var nextTasks: ListBuffer[Task] = new ListBuffer[Task]

  def doTask(): Boolean = task()

  def addNextTask(nextTask: () => Boolean): Task = {
    val newTask = new Task(nextTask)
    nextTasks += newTask
    newTask
  }

  def addNextTask(nextTask: Task): Task = {
    nextTasks += nextTask
    nextTask
  }

}
```

Рисунок 3.24 – Листинг класса Task

Класс TaskProcessor находится в пакете utils\execute. Вызов этого класса происходит в моменты создания каких-либо объектов на этапе

подготовки сцены пользовательского интерфейса или выполнения заданий после получения данных из TaskExecutor.

Класс TaskProcessor вызывается только в классе MainApplication через конструкцию new. Данный класс имеет список с упорядоченными по времени вызовами класса, что обеспечивает последовательное выполнение всех действий. Диаграмма методов класса представлена на рисунке 3.25.

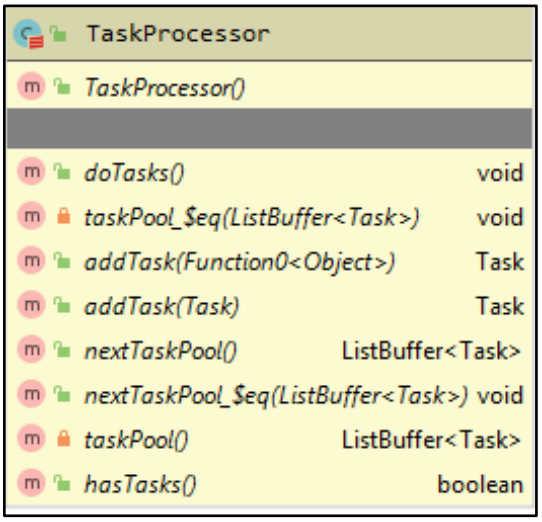

Рисунок 3.25 – Класс Task

Основное предназначение класса, это добавление заданий во временный лист хранилища, проверка хранилище на пустоту, и вызов выполнения задания.

Класс TaskExecutor находится в пакете utils\execute. Данный класс является своего рода слушателем всех действий из пользовательского интерфейса.

Так как TaskExecutor нужен для реакции на действия пользователя, то его вызов происходит не явно, используя пул потоков программы. Данный класс имеет список с объектами интерфейса Runnable, которые, используя преобразование, переходят в объект Task. Диаграмма методов класса представлена на рисунке 3.26.

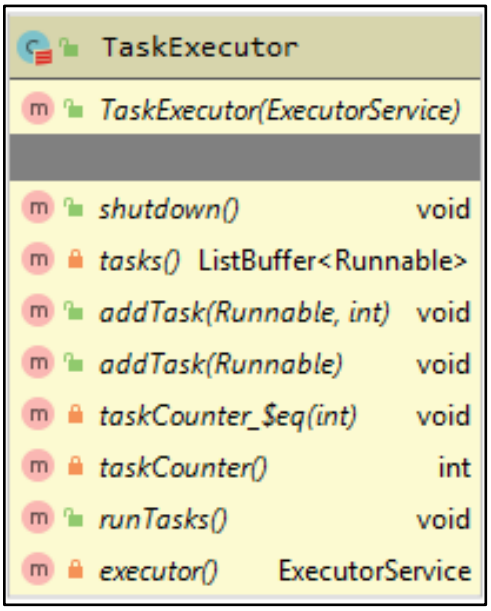

Рисунок 26 – Класс TaskExecutor

Для синхронизации листа заданий с потоками используется счетчик, благодаря которому разные потоки будут брать параллельно разные задачи.

Классы Input и Basic3DCamera находится в пакете objects\graphics. Класс Input предназначен для приема с устройств входа команд по управлению камерой, за которую отвечает класс Basic3DCamera.

Данные классы вызываются друг из друга, а также в MainApplication.

Все методы классов направлены на принятие управления с мыши и клавиатуры, с последующим перемещением камеры в обрисованной сцене. Скорость передвижения и поворота камеры задается в классе MainApplication.

Диаграммы методов классов Input и Basic3DCamera представлены на рисунке 3.27.

Клавиши управления камерой («W», «A», «S», «D», «Space», «Shift») заданы жестко в программном коде. Листинг кода, отвечающий за отлавливание нажатий клавиш, представлен на рисунке 3.28

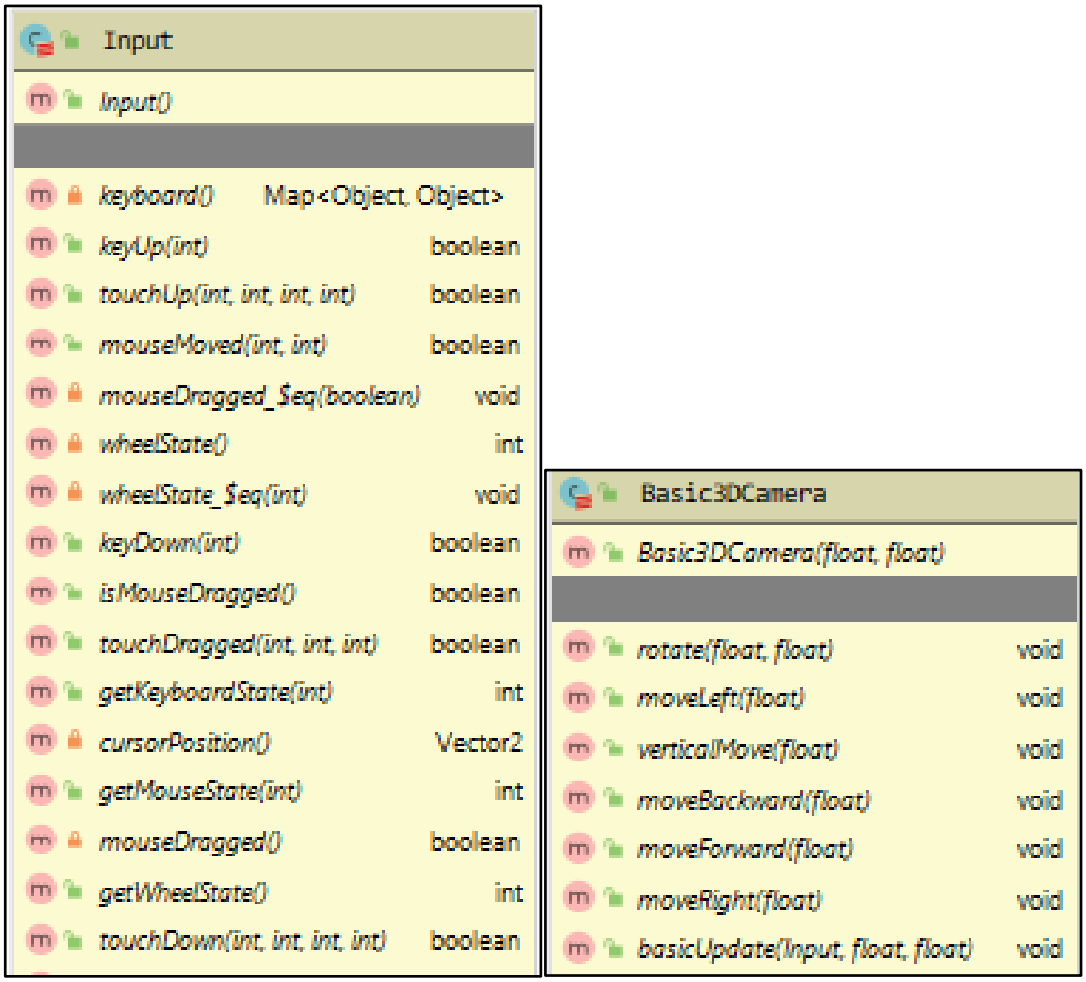

Рисунок 3.27 – Классы Input и Basic3DCamera

```
def verticalMove(speed: Float): Unit = {
  translate(0.0f, speed, 0.0f)
}

def basicUpdate(input: Input, sens: Float, moveSpeed: Float): Unit = {
  rotate(Gdx.input.getDeltaX / sens, Gdx.input.getDeltaY / sens)
  if (input.getKeyboardState(Keys.W) == KEY_REPEAT) moveForward(moveSpeed)
  if (input.getKeyboardState(Keys.A) == KEY_REPEAT) moveLeft(moveSpeed)
  if (input.getKeyboardState(Keys.S) == KEY_REPEAT) moveBackward(moveSpeed)
  if (input.getKeyboardState(Keys.D) == KEY_REPEAT) moveRight(moveSpeed)
  if (input.getKeyboardState(Keys.SPACE) == KEY_REPEAT) verticalMove(moveSpeed)
  if (input.getKeyboardState(Keys.SHIFT_LEFT) == KEY_REPEAT) verticalMove(-moveSpeed)
}
```

Рисунок 3.28 – Классы Input и Basic3DCamera

Класс Entity находится в пакете objects\genetics. Данный класс является обычным POJO классом, который хранит в себе Map с собственными

атрибутами. Ключом для Map может быть что угодно, так как класс написан с использованием generics.

Данный класс вызывается в классе MainApplication для создания списка сущностей на экране пользователя, а также в классе GeneticUtils. Класс Entity представлен на рисунке 3.29.

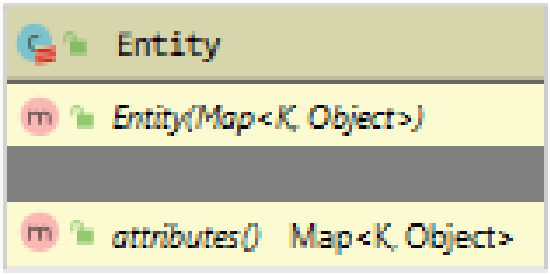

Рисунок 3.29 – Класс Entity

Данный класс не несет в себе никакой логики, а служит контейнером для хранения информации о своих атрибутах, предоставляя свою оболочку для инкапсуляции информации.

Для демонстрации работы программы представлена UML-диаграмма состояний объекта (рисунок 3.30). Так как все последующие итерации происходят по одному алгоритму, то алгоритм состояний объекта можно свернуть в алгоритм на одну итерацию.

В результате скрещивания сущностей создается новая сущность с высчитанными координатами 2-х родителей, а сущность родителей уничтожается. Сущности с худшими показателями так же подвергаются удалению.

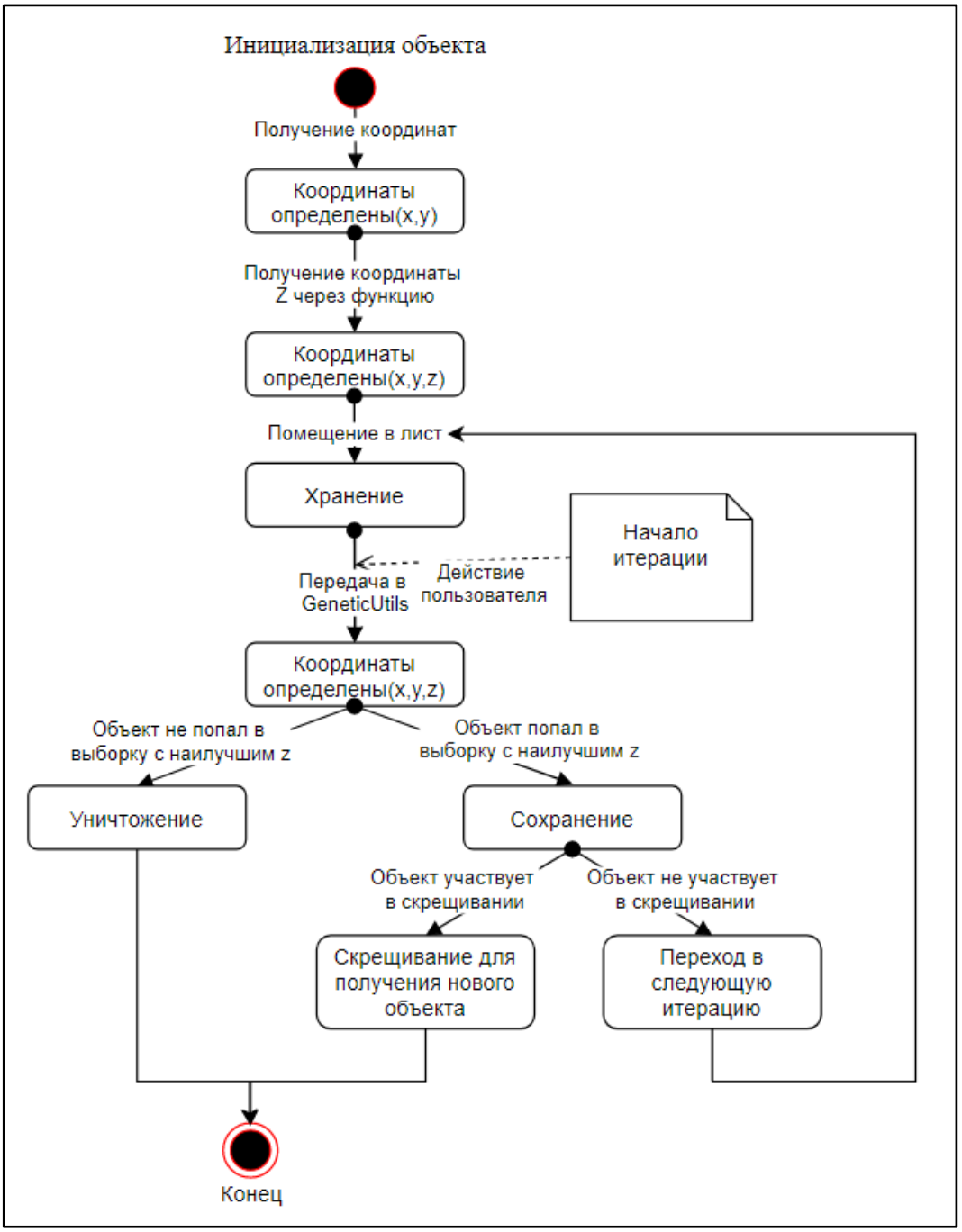

Рисунок 3.30 – UML-диаграмма состояний

MathUtils и GeneticUtils являются главными объектами, обеспечивающими логику приложения. Все вычисления, логика старта, остановки, приема всех входных атрибутов происходят в них (рисунок 3.31).

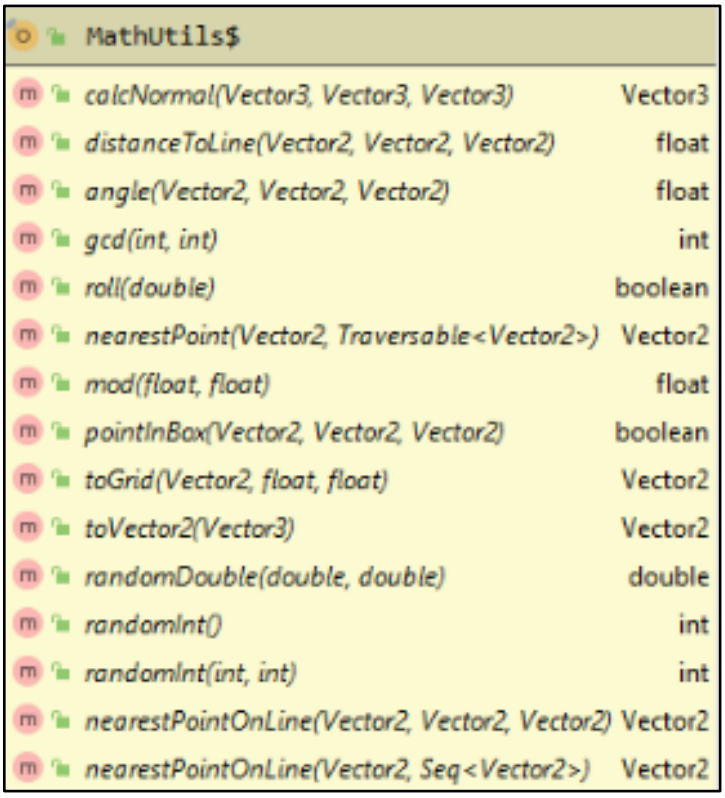

Рисунок 3.31 – Объект MathUtils

Объект MathUtils занимается всеми вычислениями: расчет нормали, расстояние до линии, ближайшие точки к линии, угол, определения, находится ли точка в указанных границах, вычисление сетки, модуля квадрата, случайных чисел и т.д.

Объект GeneticUtils использует все методы объекта MathUtils для реализации логики работы проекта. Для компоновки атрибутов по каким-либо объектам используется класс Entity. Объект GeneticUtils реализует всего один метод: nextGeneration, принимая на вход множество параметров для расчета следующего шага работы генетического алгоритма.

Объект Main сдержит метод main, который является точкой входа в программу (рисунок 3.32). В данном методе происходит конфигурация пользовательского интерфейса. Задаются параметры высоты и ширины экрана, режима отображения, названия проекта и установка иконки. После чего все параметры конфигурации передаются в класс MainApplication.

```
object Main {
  def main(args: Array[String]): Unit = {
    val config = new LwjglApplicationConfiguration()
    val screenSize = Toolkit.getDefaultToolkit.getScreenSize
    config.title = "AI2_Lab_3"
    config.width = screenSize.width
    config.height = screenSize.height
    config.fullscreen = true
    config.addIcon( path = "images/WindowIcon.png",Files.FileType.Internal)
    new LwjglApplication(new MainApplication(), config)
  }
}
```

Рисунок 3.32 – Объект Main

В классе MainApplication происходит основная работа, по инициализации, входным параметрам, настройке графики, сцены, отображения и т.д. Методы данного класса представлены на рисунке 3.33.

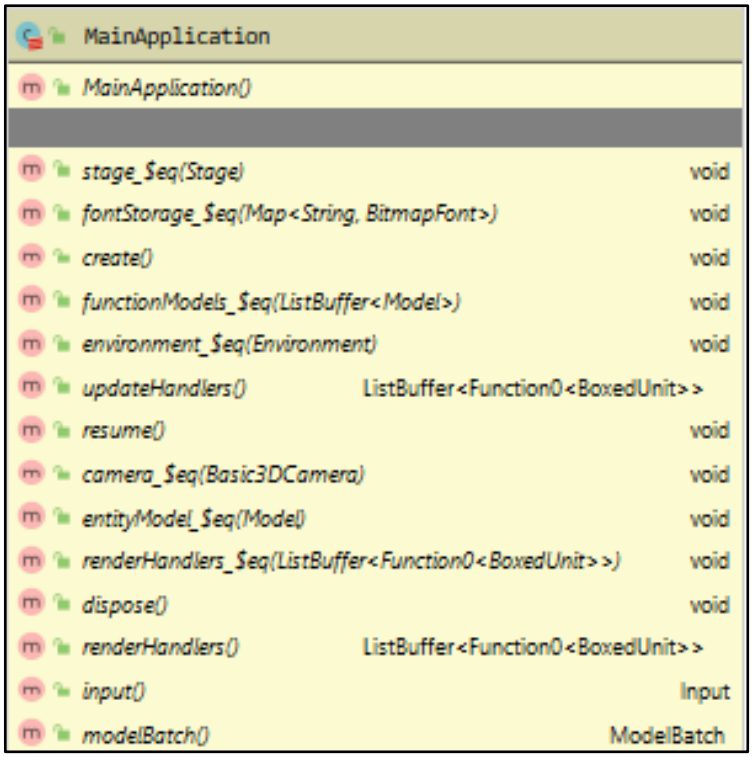

Рисунок 3.33 – Методы класса MainApplication

Вся логика инициализации подготовки к работе обернута в метод create. Работа с камерой, сценой и обновлением координат находится в

методе render. В методе dispose находится расстановка всех объектов по сцене, а в методе resize рассчитывается масштабное увеличение или уменьшение объектов по мере приближения или отдаления камеры.

3.4 Описание интерфейса пользователя

Для запуска программы необходимо открыть .jar файл с помощью JRE. Некоторое время будут производиться промежуточные расчеты, связанные с генерированием 3D сцены и построением поверхности тестовой функции, после чего на весь экран отобразить основное окно программы (рисунок 3.34).

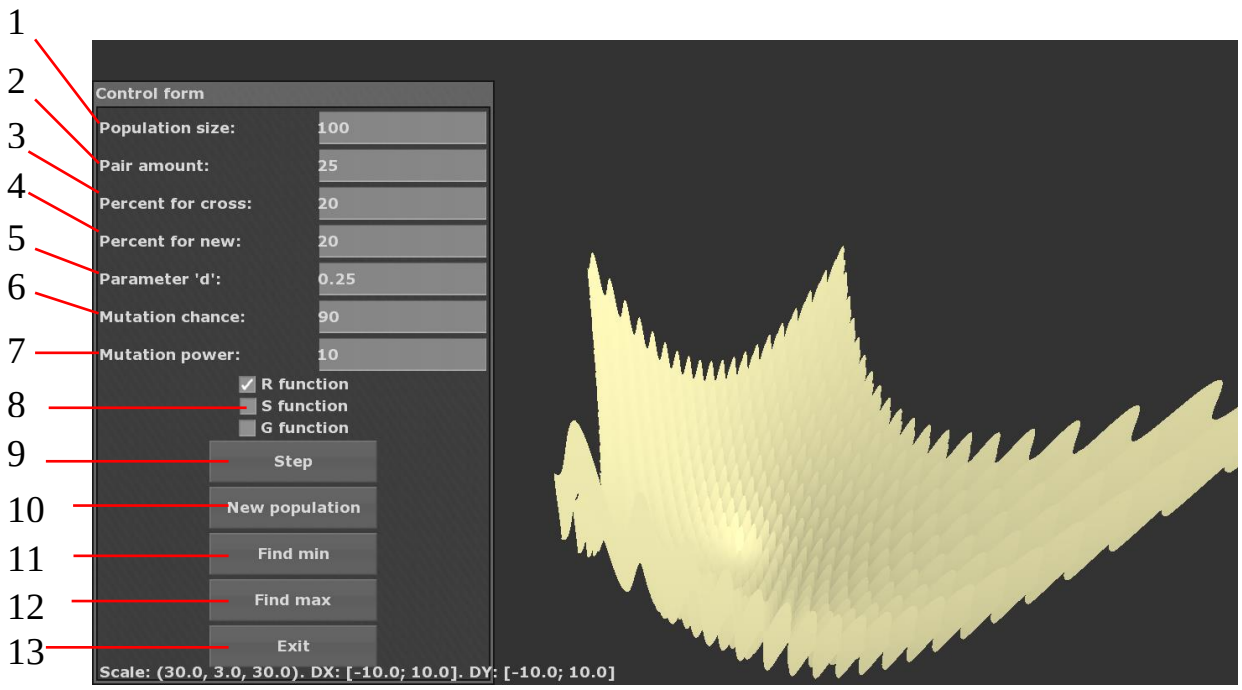

Рисунок 3.34 – Интерфейс программы

Теперь с помощью клавиши «q» можно скрыть/отобразить левую панель для задания параметров генетического алгоритма, Позиции на рисунке 3.34:

1) Количество особей в популяции.

2) Количество пар, генерируемых для скрещивания на каждой итерации генетического алгоритма.

3) Количество процентов от лучших (более приспособленных) особей, участвующих в скрещивании.

4) Количество процентов от лучших особей, переходящих в новую популяцию (остальные свободные места в популяции займут особи, сгенерированные случайным образом).

5) Значение параметра‘d’ – коэффициент расчета параметров особи после скрещивания.

6) Вероятность мутации особи в процентах.

7) Сила мутации особи в процентах.

8) Выбор одной из трех функции для тестирования генетического алгоритма.

9) Кнопка для перехода к просмотру результатов следующей итерации работы генетического алгоритма с заданными параметрами.

10) Кнопка для генерирования первоначальной популяции случайным образом.

11) Перемещение камеры к особи, находящейся ближе всего к глобальному минимуму.

12) Перемещение камеры к особи, находящейся ближе всего к глобальному максимуму.

13) Выход из программы.

В качестве выводов по данной главе можно отметить следующее. Разработанное программное обеспечение для 3D визуализации работы генетического алгоритма успешно справляется с поставленными задачами. Данное приложение может быть использовано для демонстрации принципов работы алгоритма в рамках дистанционного образования при изучении методов оптимизации.

ЗАКЛЮЧЕНИЕ

В заключение бакалаврской работы можно сделать следующие выводы:

1. Распространение вирусной инфекции COVID-19 будет стимулировать развитие дистанционного образования, по этой причине актуальным вопросом будет являться разработка средств визуализации изучаемого учебного материала.

2. Генетические алгоритмы является одной из тем теории оптимизации, понимание которой (из-за множества параметров алгоритма) требует рассмотрения большого количества практических примеров.

3. Основными параметрами генетического алгоритма, влияющими на результат решения задачи оптимизации, являются: размер популяции на каждой итерации; количество пар, генерируемых для скрещивания; процент лучших особей, участвующих в скрещивании; коэффициент расчета параметров особи после скрещивания; процент лучших особей попадающих в популяцию перед переходом на следующую итерацию (остальные места в популяции занимают особи, сгенерированные случайным образом); вероятность мутации особи после скрещивание и сила мутации.

4. На языке программирования Scala было разработано программное обеспечения для 3D-визуализации процесса поиска глобальных экстремумов функции с помощью генетического алгоритма. Программное обеспечение генерирует 3D сцену, содержащую в себе поверхность исследуемой функции и решения, найденные на текущей итерации генетическим алгоритмом. При этом обеспечено интерактивное управления обзором на 3D сцену, что повышает наглядность при наблюдении за процессом поиска экстремумов функции.

5. Дальнейшее развитие программного обеспечения возможно за счет внедрения поддержки шлемов виртуальной реальности VR.

Таким образом, все поставленные задачи выполнены, и цель работы достигнута.

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. Басинский, В.М. Алгоритм муравьиной колонии при решении задачи классификации и использование генетического алгоритма для подбора его параметров / В.М. Басинский, Ю.Г. Степин // Информационно-коммуникационные технологии: достижения, проблемы, инновации (ИКТ-2018) Электронный сборник статей I международной научно-практической конференции, посвященной 50-летию Полоцкого государственного университета. 2018. – Новополоцк: Издатель Учреждение образования «Полоцкий государственный университет» =Установа адукацыі "Полацкі дзяржаўны універсітэт", 2018. – С. 118-122. – Текст: непосредственный.

2. Бедин, Д.А. Использование генетического алгоритма для определения параметров многогипотезного алгоритма восстановления траектории воздушного судна / Д.А. Бедин, А.Г. Иванов // XXVI Санкт-Петербургская международная конференция по интегрированным навигационным системам Сборник материалов. 2019. – Санкт-Петербург: Издатель «Концерн "Центральный научно-исследовательский институт "Электроприбор", 2019. – С. 87-90. – Текст: непосредственный.

3. Ванюкова, Д.И. Специализированные методы визуализации 2d и 3d-изображений в бортовых картографических системах / Д.И. Ванюкова, А.А. Залучанов, К.В. Малынкин, П.А. Соколов // Навигация и управление движением. Санкт-Петербург, 12 марта-01 ноября 2013 г. – Санкт-Петербург: Издатель «Концерн " Центральный научно-исследовательский институт "Электроприбор", Санкт-Петербург, 2013. – Р. 166-171. – Текст: непосредственный.

4. Васильева, О.В. Алгоритм формирования эффективного расписания работы с грузами в морском порту с использованием технологии генетических алгоритмов / О.В. Васильева, В.И. Кияев // Сборник научных трудов школы-семинара по искусственному интеллекту Материалы школы-

семинара. 2018. – Тверь: Издатель Тверской государственный технический университет, 2018. – С. 4-13. – Текст: непосредственный.

5. Влацкая, Л.А. Разработка алгоритма оптимального размещения компенсирующих устройств в узлах электрической сети с применением генетического алгоритма / Л.А. Влацкая, Н.Г. Семенова // Состояние и перспективы развития электро- и теплотехнологии (бенардосовские чтения) Материалы Международной (XX Всероссийской) научно-технической конференции. 2019. – Иваново: Издатель Ивановский государственный энергетический университет имени В.И.Ленина", 2019. – С. 303-306. – Текст: непосредственный.

6. Гоголь, А.А. Особенности изучения основ работы с изображениями 3d в лаборатории кафедры телевидения и видеотехники / А.А. Гоголь, В.В. Дуклау, С.Э. Коганер, Т.Г. Смаглиенко, О.В. Скраинский // II Международная научно-техническая и научно-методическая конференция «Актуальные проблемы инфотелекоммуникаций в науке и образовании», Санкт-Петербург, 27-28 февраля 2013 г. – Санкт-Петербург: Издатель Санкт-петербургский государственный университет телекоммуникаций им. Проф. М.А. Бонч-Бруевича Санкт-Петербург, 2013. – Р. 280-285. – Текст: непосредственный.

7. Грачев, А.Н. Адаптивный алгоритм Калмановской фильтрации для трассового сопровождения целей с использованием быстрого генетического алгоритма / А.Н. Грачев, Х.А. Хусейн // XII Всероссийское совещание по проблемам управления ВСПУ-2014, Москва, 16-19 июля 2014 г. – Москва: Издатель Институт проблем управления им. В.А. Трапезникова РАН, Москва, 2014. – Р. 9092-9103. – Текст: непосредственный.

8. Левина, О.С. Метод обработки изображений карт глубин, полученных с помощью активных 3d-сканеров / О.С. Левина, В.В. Воронин, В.А. Франц, Р.А. Кожин, А.В. Фисунов // ИНЭРТ-2014, Ростов-на-Дону, 07-10 октября 2014 г. – Ростов-на-Дону: Издатель Донской государственный

технический университет, Ростов-на-Дону,2014. – P. 1341-1346. – Текст: непосредственный.

9. Федотов, Н.Г. Проблемы распознавания 3d изображений у машин и людей: сравнительная характеристика /Н.Г. Федотов, А.А. Сёмов, А.А. Курносов //Проблемы информатики в образовании, управлении, экономике и технике, Пенза, 13-14 ноября 2014 г. – Пенза : Издатель Автономная некоммерческая научно-образовательная организация «Приволжский Пом знаний», Пенза, 2014. – P. 185-193. – Текст: непосредственный.

10. Фурсов, В.А. Гибридный параллельный алгоритм сопоставления стерео-изображений в задаче реконструкции 3d-сцен / В.А. Фурсов, Е.В. Гошин, А.П. Котов // Научный сервис в сети интернет, Новороссийск, 22-27 сентября 2014 г. – Новороссийск : Издатель Московский государственный университет имени м.В. Ломоносова" издательский дом, Москва, 2014. – P. 61-65. – Текст: непосредственный.

11. Camino, R.V. Initialization in Genetic Algorithms for Constraint Satisfaction Problems / R.V. Camino, V. Ramiro, J. Puente // International Work-Conference on Artificial Neural Networks, Connectionist Models of Neurons, Learning Processes, and Artificial Intelligence: 6th International Work-Conference on Artificial and Natural Neural Networks, IWANN 2001 Granada, Spain, June 13–15, 2001 Proceedings, Part 1. – Granada: Springer-Verlag Berlin Heidelberg, 2001. – P. 693-700. – Text: direct.

12. Freisleben, B. Optimization of Genetic Algorithms by Genetic Algorithms / B. Freisleben, M. Härtfelder // Artificial Neural Nets and Genetic Algorithms: Proceedings of the International Conference in Innsbruck, Austria, 1993. – Innsbruck: Springer-Verlag/Wien, 1993. – P. 392-399. – Text: direct.

13. Mariano, V. Topology Optimization of bidimensional continuum structures by genetic algorithms and stress iso-lines / V. Mariano, M. Pascual // III European Conference on Computational Mechanics: Solids, Structures and Coupled Problems in Engineering: Book of Abstracts. – Lisbon: Springer Netherlands, 2006. – P. 442-442. – Text: direct.

14. Martyna, J. Application of genetic algorithms to computer assignment problem in distributed hard real-time systems / J. Martyna // International Conference on Computational Intelligence, Computational Intelligence Theory and Applications: International Conference, 5th Fuzzy Days Dortmund, Germany, April 28–30, 1997 Proceedings. – Dortmund: Springer-Verlag Berlin Heidelberg, 1997. – P. 564-564. – Text: direct.

15. Murata, T. Specification of Genetic Search Directions in Cellular Multi-objective Genetic Algorithms / T. Murata, H. Ishibuchi, M. Gen // International Conference on Evolutionary Multi-Criterion Optimization, Evolutionary Multi-Criterion Optimization: First International Conference, EMO 2001 Zurich, Switzerland, March 7–9, 2001 Proceedings. – Zurich: Springer-Verlag Berlin Heidelberg, 2001. – P. 82-95. – Text: direct.

16. Niwa, T. Analysis on the Island Model Parallel Genetic Algorithms for the Genetic Drifts / T. Niwa, M. Tanaka // Asia-Pacific Conference on Simulated Evolution and Learning, Simulated Evolution and Learning: Second Asia-Pacific Conference on Simulated Evolution and Learning, SEAL'98 Canberra, Australia, November 24–27, 1998 Selected Papers. – Canberra: Springer-Verlag Berlin Heidelberg, 1999. – P. 349-356. – Text: direct.

17. Vinvent, T. Applying Genetic Algorithms and Other Heuristic Methods to Handle PC Configuration Problems / T. Vinvent, K.T. Ma // International Conference on Computational Science, Computational Science - ICCS 2001: International Conference San Francisco, CA, USA, May 28—30, 2001 Proceedings, Part II. – San Francisco: Springer-Verlag Berlin Heidelberg, 2001. – P. 439-446. – Text: direct.

18. Wong, P.K. A technique for improving the convergence characteristic of genetic algorithms and its application to a genetic-based load flow algorithm / P.K Wong, Li An // Asia-Pacific Conference on Simulated Evolution and Learning, Simulated Evolution and Learning: First Asia-Pacific Conference, SEAL'96 Taejon, Korea, November 9–12, 1996 Seclected Papers. – Taejon: Springer-Verlag Berlin Heidelberg, 1997. – P. 167-176. – Text: direct.

19. Yu, T.L. Genetic Algorithm Design Inspired by Organizational Theory: Pilot Study of a Dependency Structure Matrix Driven Genetic Algorithm / T.L. Yu, D.E. Goldberg, A. Yassine, Y.P. Chen // Genetic and Evolutionary Computation Conference, Genetic and Evolutionary Computation — GECCO 2003: Genetic and Evolutionary Computation Conference Chicago, IL, USA, July 12–16, 2003 Proceedings, Part II. – Chicago: Springer-Verlag Berlin Heidelberg, 2003. – P. 1620-1621. – Text: direct.

20. Yusoff, M. Evaluation of Genetic Algorithm and Hybrid Genetic Algorithm-Hill Climbing with Elitist for Lecturer University Timetabling Problem / M.Yusoff, N. Roslan // International Conference on Swarm Intelligence, Advances in Swarm Intelligence: 10th International Conference, ICSI 2019, Chiang Mai, Thailand, July 26–30, 2019, Proceedings, Part I.– Chiang Mai : Springer Nature Switzerland AG, 2019 . – P. 363 - 373. – Text: direct.