МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НОВОСИБИРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ» (НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ, НГУ)

Факультет информационных технологий

Кафедра Систем Информатики

Направление подготовки: 230100 ИНФОРМАТИКА И ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ БАКАЛАВРСКАЯ РАБОТА

Система 3D визуализации для программного комплекса «Гаплоидный эволюционный конструктор»

Чеканцева Антона Дмитриевича

«К защите допущена»

Заведующий кафедрой,

д.ф.-м.н., профессор

Лаврентьев М.М./……..

(фамилия , И., О.) / (подпись, МП)

«……»……2014г.

Научный руководитель

с.н.с, ИЦиГ СО РАН,

к.б.н.

Лашин С.А./……..

(фамилия , И., О.) / (подпись, МП)

«……»……2014г.

Дата защиты: «……»……2014г.

Автор Чеканцев А.Д./……..

(фамилия , И., О.) / (подпись)

Новосибирск, 2014г.

МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НОВОСИБИРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ» (НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ, НГУ)

Кафедра систем информатики

УТВЕРЖДАЮ
Зав. Кафедрой: Лаврентьев М. М.
(фамилия, И., О.)
……
(подпись, дата)

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ БАКАЛАВРА

Студенту Чеканцеву Антону Дмитриевичу

Направление подготовки 230100.62 ИНФОРМАТИКА И ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА

ФАКУЛЬТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Тема: Система 3D визуализации для программного комплекса «Гаплоидный эволюционный конструктор»

Исходные данные (или цель работы): Целью данной работы является создание графического интерфейса с элементами управления процессом моделирования, визуализации различных параметров для программного комплекса «Гаплоидный эволюционный конструктор-3D версия» (ГЭК-3D).

Структурные части работы: Работа включает в себя обзор предметной области, обзор применяемых технологий и описание разработанного интерфейса.

Содержание

ВВЕДЕНИЕ

Настоящая работа выполнена в Институте цитологии и генетики (ИЦиГ) СО РАН. Она является частью проекта по разработке программного комплекса «Гаплоидный эволюционный конструктор» [1], предназначенного для моделирования эволюции и функционирования сообществ микробных популяций.

Исследование биологической эволюции является одной из актуальных задач современной науки. Однако тот факт, что большинство эволюционных процессов протекают на продолжительных временных интервалах, серьезно затрудняет работу. Поэтому математическое и компьютерное моделирование становится важным инструментом исследования эволюции.

Существует несколько способов, позволяющих моделировать эволюционные процессы популяций. Все возникающие при этом математические модели биологических систем, условно можно разделить на регрессионные, качественные и имитационные [2].

Регрессионная модель объединяет широкий класс универсальных функций, которые описывают некоторую закономерность. Для построения модели используются в основном измеряемые данные, а не конкретные свойства исследуемого объекта. По большей части это математические формулы, описывающие связь различных характеристик системы и не затрагивающие биологическую составляющую модели [3].

Качественная (базовая) модель – простая модель, которая легко поддается аналитическому исследованию. На основании изучения свойств и реакций такой простой модели, можно изучать и объяснять явления, относящиеся уже к более сложным и реальным системам [3].

Имитационная модель — это компьютерная программа, которая описывает структуру и воспроизводит поведение сложной реальной системы взаимодействующих элементов во времени, позволяет получать подробную статистику о различных аспектах функционирования системы в зависимости от входных данных. Суть имитационного моделирования заключается в исследовании сложной математической модели с помощью вычислительных экспериментов и обработки результатов этих экспериментов. При этом, как правило, максимально используется вся имеющаяся информация об объекте моделирования, как количественная, так и качественная [3].

Несмотря на разнообразие живых систем, при построении математических моделей следует учитывать тот факт, что все они обладают некоторыми специфическими свойствами.

Биологические системы являются сложными, пространственно-распределенными иерархическими структурами, состоящие из многих компонентов, каждый из которых

обладает некоторыми индивидуальными свойствами. При моделировании таких систем возможно два подхода.

Первый подход – агрегированный. В соответствии с этим подходом выделяются основные характеристики системы (например, общая численность видов) и рассматриваются различные свойства поведения этих величин во времени (устойчивость стационарного состояния, наличие колебаний, существование пространственной неоднородности). Такой подход является наиболее распространённым и свойственен динамической теории развития популяций.

Другой подход – подробное рассмотрение элементов системы и их взаимодействий, то есть рассмотренное выше имитационное моделирование. Имитационная модель не допускает аналитического исследования, но ее параметры имеют ясный физический и биологический смысл, и при хорошей экспериментальной изученности фрагментов системы она может дать количественный прогноз ее поведения при различных внешних воздействиях.

Существует множество различных типов биологических систем, основные особенности которых следует учитывать при построении модели. Существенную роль при моделировании играет описание размножающихся систем. Это важнейшее свойство живых систем определяет их способность перерабатывать неорганическое и органическое вещество для биосинтеза биологических веществ, клеток, организмов. Важную роль в развитии сложных пространственно-временных взаимосвязей играют процессы взаимодействия компонентов (биохимические реакции), процессы переноса, как хаотического (диффузия), так и связанного с направлением внешних сил (гравитация, электромагнитные поля), трофические взаимодействия, активное перемещение клеток, влияние протока и адаптивные функции живых организмов.

Биологические объекты имеют сложную многоуровневую систему регуляции. На уровне органа, организма, популяции живая система также является гетерогенной, и это ее основополагающее свойство также необходимо учитывать при создании математической модели.

На сегодняшний день существуют несколько программных средств, для моделирования микробных сообществ. Программа AgentCell [4] была разработана на основе модели агентов, для изучения взаимосвязей между стохастическими межклеточными процессами и поведения отдельных клеток. Эта программа делает акцент на моделировании хемотаксиса – одного из важнейших клеточных процессов, определяющего функционирование как отдельных микробных клеток, так и микробных сообществ. В этой модели, каждая бактерия — это агент, оснащенный собственной сетью

хемотаксиса и жгутиками. Плавающие клетки могут свободно перемещаться в трехмерном пространстве.

Программа-симулятор BacSim [5] была разработана на основе индивидуально ориентированной модели, для моделирования роста и поведения бактерии. Потенциал данного подхода заключается в изучении связи свойств микроскопических сущностей (клеток) со свойствами макроскопических, сложных систем, таких как биопленки.

Программа Aevol [6] является платформой, которая позволяет моделировать развитие популяций в различных условиях и изучать механизмы, ответственные за структурирование генома. Данная платформа поставляется с набором инструментов, позволяющих анализировать и измерять характеристики организмов на протяжении эволюции.

Однако существенный недостаток упомянутых выше программных комплексов заключается в том, что каждый из них делает упор лишь на часть свойств и взаимодействий микробных сообществ, фигурирующих в эволюционных процессах. Например, программа Aevol акцентирует внимание на описании процессов структурирования генома и не рассматривает процессы взаимодействия популяций на экологическом уровне. Другие программы – наоборот, детально рассматривают популяционный и экологический уровни, не описывая генетические процессы. При этом общая картина функционирования сообществ, максимально приближенная к реальности, остается неисследованной.

Таким образом, для качественного моделирования эволюции микробных сообществ необходимо программное средство, позволяющее строить и исследовать модели сообществ высокой численности со сложной структурой. В ИЦиГ СО РАН был создан программный комплекс «Гаплоидный эволюционный конструктор» (далее в тексте – ГЭК) с учётом данных требований. Он позволяет моделировать функционирование и эволюцию сообществ микробов с учётом влияния среды, трофических взаимоотношений между популяциями и видообразования.

Биологические объекты представляют собой сложные, иерархически организованные системы, поэтому возникают значительные трудности при попытке их визуализации. Все получаемые результаты должны быть представлены пользователю в информативном и понятном виде. А интерфейс должен предоставлять простые и удобные элементы управления, с возможностью конфигурации по желанию пользователя.

В результате данной работы была разработана система управления и визуализации ГЭК с учетом нового пространственного распределения, обладающая вышеописанными характеристиками и удовлетворяющая поставленным требованиям.

ОСНОВНАЯ ЧАСТЬ

1. Постановка задачи

Целью данной работы является создание графического интерфейса с элементами управления процессом моделирования, визуализации различных параметров в виде 3D объектов, графиков, гистограмм и графов для программного комплекса «Гаплоидный эволюционный конструктор-3D версия» (ГЭК-3D). Все основные объекты и процессы ГЭК, требующие визуализации, показаны на рисунке 1.

Для достижения цели были поставлены и решены следующие задачи:

- Освоена технология OpenGL
- Реализована визуализация в виде 3D объектов
- Разработан и реализован графический интерфейс

Работа была разбита на следующие этапы:

- Анализ данных, подлежащих визуализации
- Изучение технологии OpenGL
- Создание 3D модели на основе изученных материалов
- Объединение GUI с ядром ГЭК
- Создание дополнительных виджетов

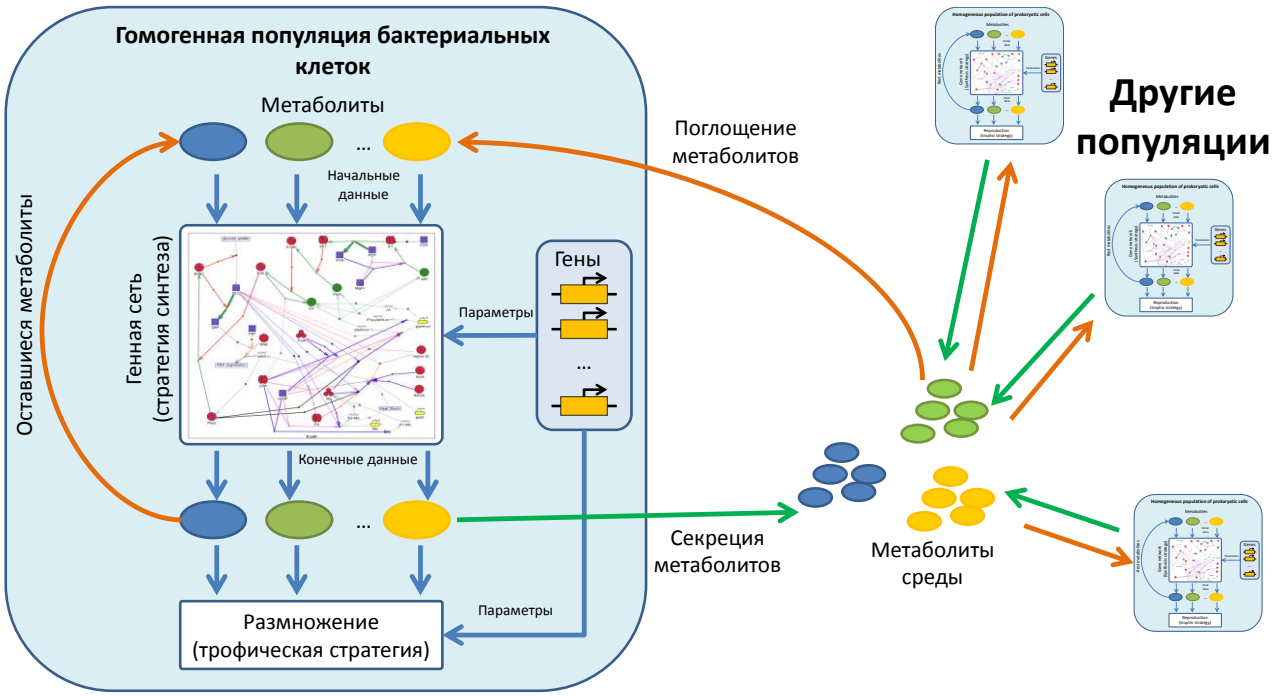

Рис.1 Схема объектов ГЭК.

Функционирование ГЭК

ГЭК позволяет моделировать функционирование и эволюцию микробных сообществ.

Вначале была разработана система, описывающая некоторый объем, содержащий несколько популяций микробных сообществ и специфические субстраты (вещества, обеспечивающие жизнедеятельность популяции и которые производятся самими клетками). Такие субстраты, производимые одной популяцией, могут оказывать ингибирующее или активирующее воздействие на другие популяции.

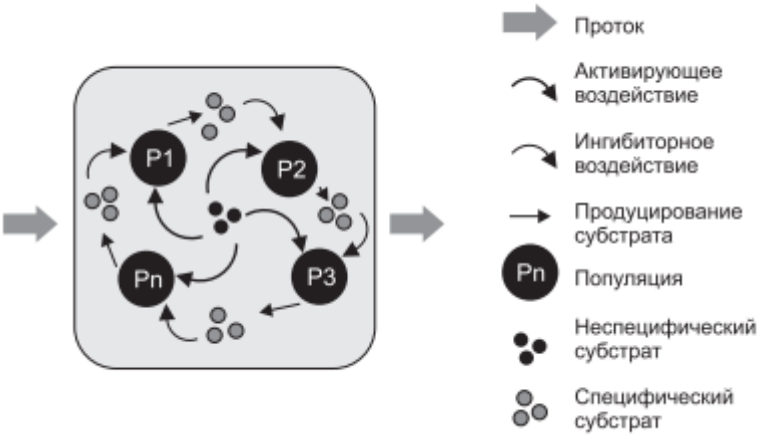

Рис.2 Схема основных процессов взаимодействий (0D система) [7].

Существует некоторый проток, который приносит неспецифические субстраты (вещества, которые не производятся клетками) в рассматриваемый объем. Все это так называемая 0D система (рис. 2).

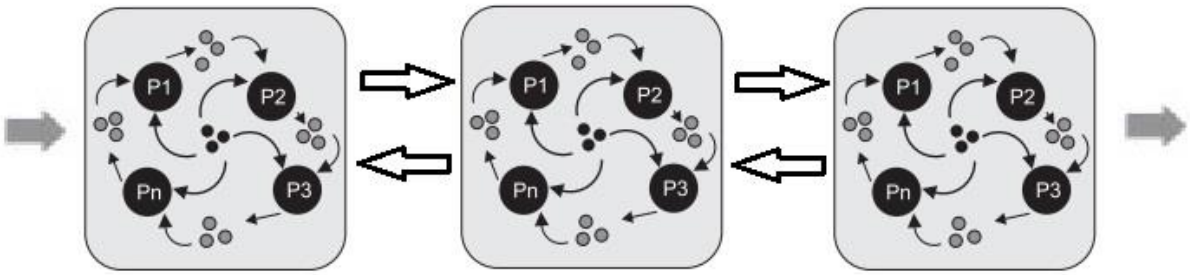

Рис.3 Схема пространственной организации в ГЭК (1D случай).

Затем происходило расширение сначала до 1D (рис.3), когда уже несколько 0D ячеек могли взаимодействовать друг с другом, в пределах одного измерения, а затем 2D (рис.4) и 3D (рис.5) соответственно. Взаимодействия между популяциями происходят посредством цепей питания.

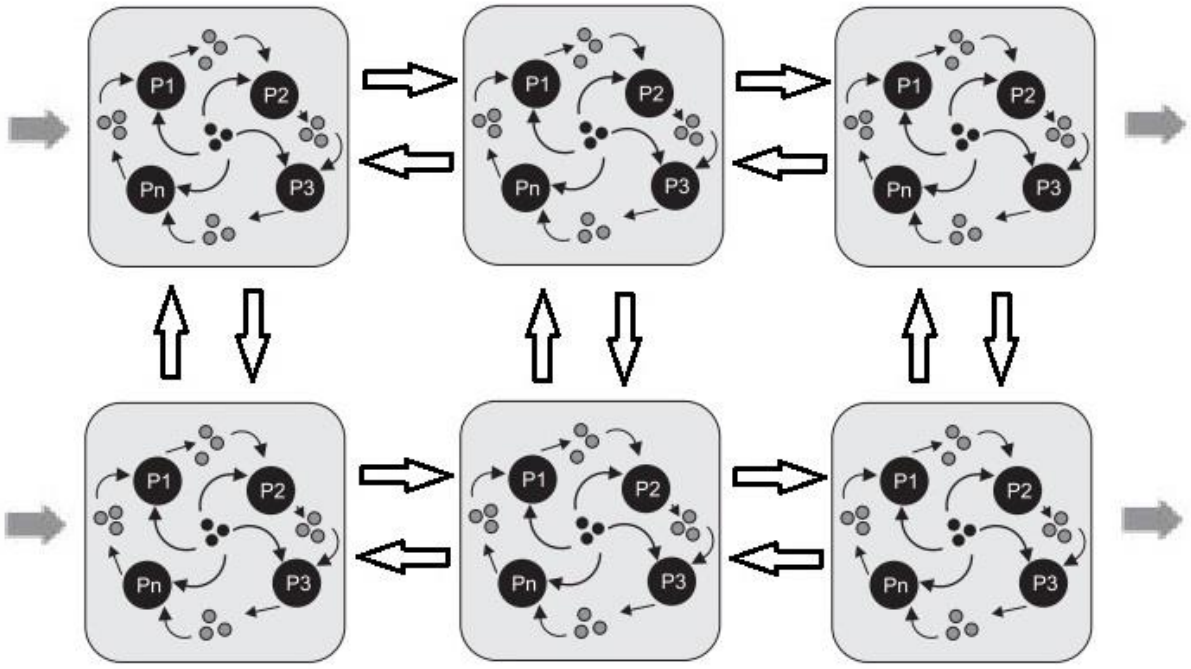

Рис.4 Схема пространственной организации в ГЭК (2D случай).

Специфические субстраты, производимые одной популяцией, могут потребляться другими популяциями. Кроме того, все популяции могут потреблять субстраты, приносимые в объем протоком (неспецифические субстраты).

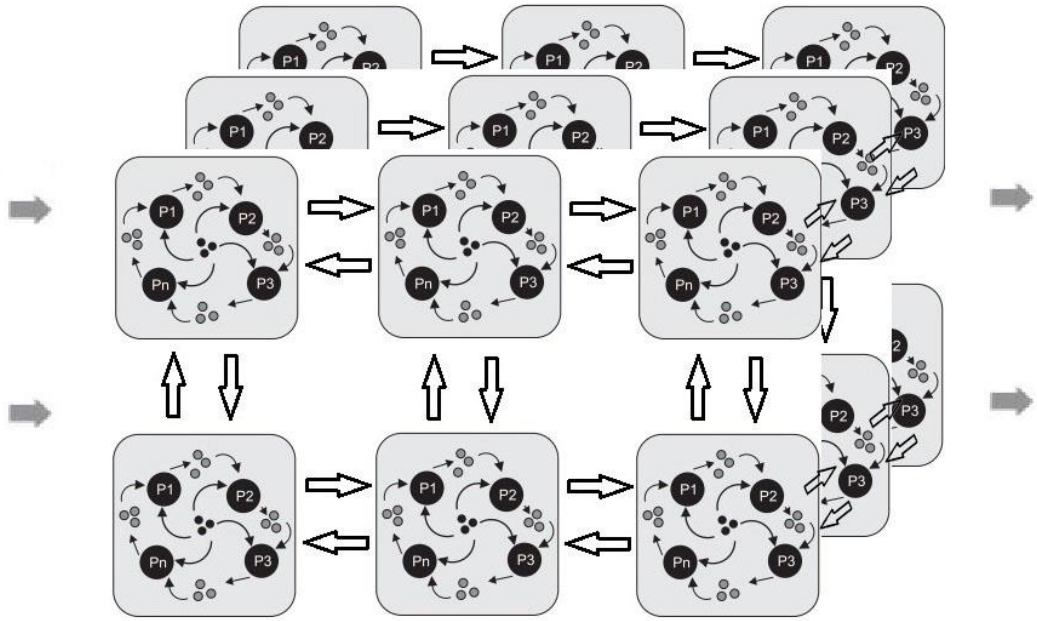

Рис.5 Схема пространственной организации в ГЭК (3D случай).

Схема работы ГЭК выглядит следующим образом (рис.6):

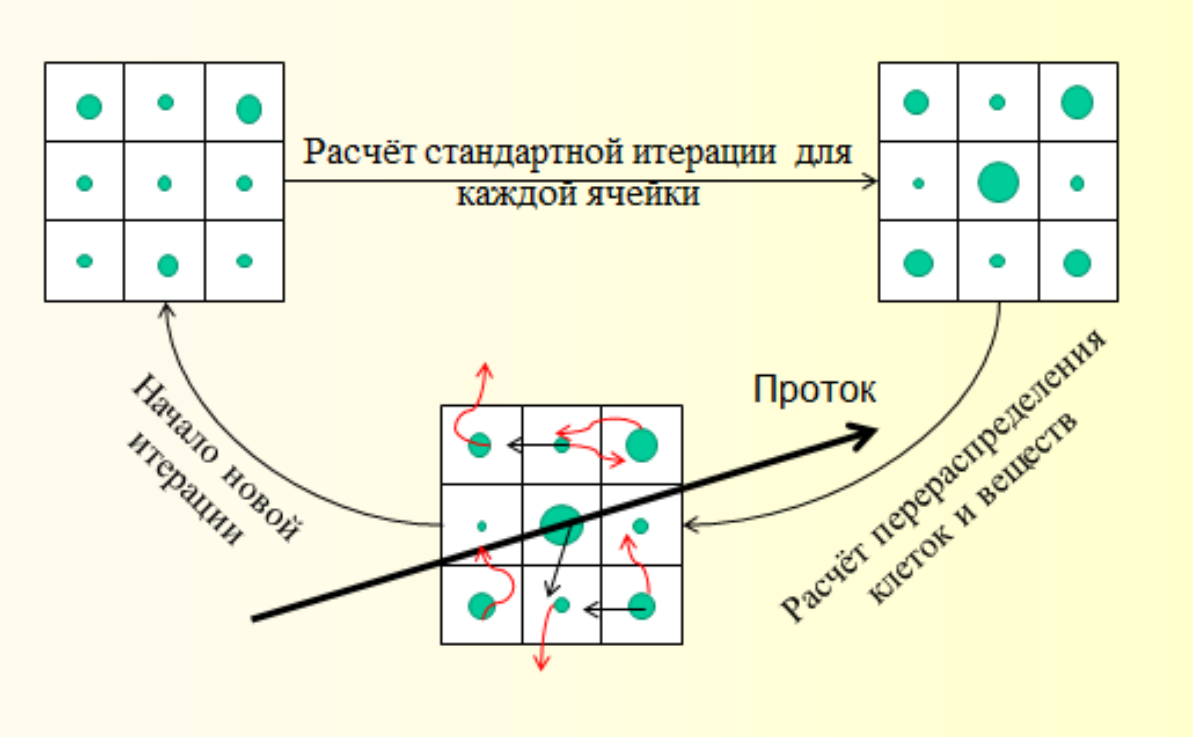

Рис.6 Схема работы ГЭК.

1) Задано начальное распределение популяций и веществ в объеме.
2) Для каждой ячейки происходит независимый расчет стандартной итерации.
3) Расчет перераспределения клеток и веществ

Расчет стандартной итерации для каждой ячейки (рис.7):

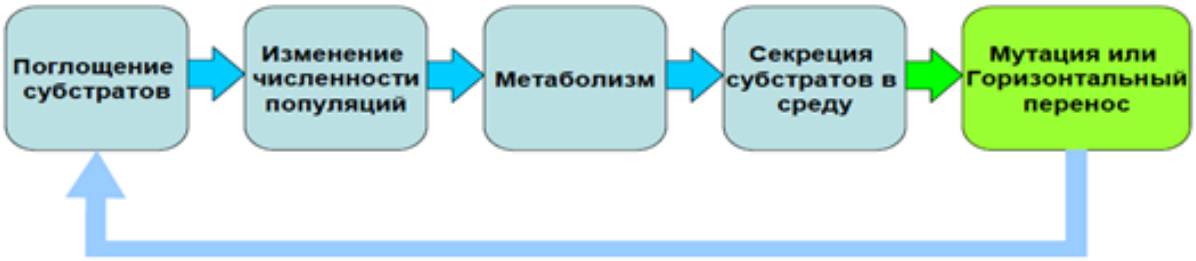

Рис.7 Стандартная итерация.

Расчет перераспределения клеток и веществ:

- Проток
- Диффузия
- Активное перемещение клеток

Ранее был разработан графический интерфейс для 2D версии ГЭК, на языке Java. Однако для существующей 3D версии потребовался концептуально новый интерфейс, с поддержкой всех существующих настроек и возможностью визуализации рассматриваемой 3D модели в наглядном виде.

При разработке новой системы визуализации были учтены и исправлены недостатки предыдущей версии, интерфейс приобрел больше возможностей для пользовательской настройки, исчезла необходимость в обеспечении дополнительного взаимодействия Java интерфейса с ядром ГЭК (C++). Был значительно расширен функционал в плане визуализации различных пространственно распределенных систем (0D – 3D), в отличие от предыдущей версии интерфейса, где была возможность просмотра только одной 0D системы.

2. Требования к системе

Для достижения поставленных задач были сформулированы следующие требования:

1) Система должна обладать возможностью визуализации рассматриваемой пространственной модели, где каждая ячейка отображается в виде прозрачного куба, внутри которого располагается сфера, представляющая концентрацию популяций, находящихся в данной ячейке. Чем больше концентрация, тем больше диаметр сферы. Также необходима возможность просмотра анимированного процесса изменения концентраций популяций на протяжении заданного интервала. При рассмотрении большого объема, необходимо иметь возможность выбрать требуемый интервал из рассматриваемого объема, для удобства наблюдения. Изображаемую модель необходимо масштабировать и вращать.
2) Возможность загружать новые модели с заданными параметрами, указанные в специальном текстовом файле (скрипты).
3) Возможность графического представления динамики роста популяций и концентрации субстратов, как по всему объему, так и в конкретной ячейке.
4) Система должна визуализировать граф трофических отношений между популяциями, в котором направляющими стрелками указаны потребляемые и производимые субстраты, а также потребление неспецифических субстратов. Отображение для всего объема и для конкретной ячейки. Возможность перемещения каждого элемента и масштабирование.
5) Система должна поддерживать визуализацию распределения аллельных частот генов в популяциях в виде гистограмм и изменение частот в популяции на заданном интервале времени. Должна быть предусмотрена возможность просмотра частот для всего объема, а также для конкретной популяции и конкретного гена.
6) Система должна обладать гибким и дружественным интерфейсом, позволяющим с легкостью получить наглядную информацию любого уровня, для любого элемента рассматриваемой модели. Все объекты должны быть расположены на специальных виджетах, которые пользователь сможет перемещать, масштабировать, закрывать, сворачивать во вкладки и выносить в отдельные окна.

3. Аналитический обзор использованных технологий и методов

3.1 Среда разработки

В качестве среды разработки была выбрана кроссплатформенная свободная IDE Qt Creator для разработки на языке C++ [8]. Интегрированный GUI компоновщик и редактор форм для C++ проектов, позволяет быстро проектировать и строить виджеты и диалоги используя экранные формы на основе тех же виджетов что будут использованы в разрабатываемом приложении.

Еще одной ключевой особенностью является механизм сигналов и слотов, и вероятно, это та часть, которая выгодно отличает Qt от других фреймворков. Сигнал вырабатывается, когда происходит определенное событие. Слот — это функция, которая вызывается в ответ на определенный сигнал. Виджеты Qt имеют много предопределенных сигналов и слотов, но всегда можно создать свои сигналы и слоты. Механизм сигналов и слотов безопасен. Сигнатура сигнала должна совпадать с сигнатурой слота-получателя. (Фактически слот может иметь более короткую сигнатуру чем сигнал который он получает, так как он может игнорировать дополнительные аргументы). Так как сигнатуры сравнимы, компилятор может помочь нам обнаружить несовпадение типов. Сигналы и слоты слабо связаны. Класс, который вырабатывает сигнал, не знает и не заботится о том, какие слоты его получат. Механизм сигналов и слотов Qt гарантирует, что если подключить сигнал к слоту, слот будет вызван с параметрами сигнала в нужное время. Сигналы и слоты могут принимать любое число аргументов любого типа. Все это составляет мощный механизм создания объектов [9]. А, благодаря функциональным формам, можно быстро удостовериться, что все работает так, как и задумано.

3.2 Технологии создания 3D объектов

Для создания 3D объектов была выбрана технология OpenGL ES 2.0. Это открытая графическая библиотека, спецификация, определяющая независимый от языка программирования платформонезависимый программный интерфейс для написания приложений, использующих двумерную и трёхмерную компьютерную графику. Включает в себя множество функций для рисования сложных моделей из простых примитивов. ES 2.0 – это подмножество графического интерфейса OpenGL, разработанная специально для встраиваемых систем и поддерживаемая средой разработки Qt Creator версии 5.0 и выше.

В ядре OpenGL все расчеты осуществляются непосредственно на графическом процессоре, используя аппаратное ускорение. Видеокарта серьезно облегчает работу центрального процессора, поскольку она может произвести значительную часть вычислений, связанных с обработкой изображений, до тех пор по они не будут

представлены на мониторе. OpenGL предоставляет множество возможностей для хранения изображений, данных и различной информации в оптимизированном формате. Эти данные затем будут непосредственно обработаны на графическом процессоре. Следовательно, поддержка OpenGL зависит от аппаратного обеспечения. Однако на сегодняшний день большинство видеокарт поддерживают данную технологию. [10 - 11]

Структура OpenGL очень проста и основана на 3 компонентах:

- **Примитивы:**

  Точка в пространстве, с координатами (x, y, z)

  Прямая в пространстве, состоящая из двух точек

  Треугольник в пространстве, состоящий из трех точек

- **Буферы:**

  Буфер – это временное оптимизированное хранилище. В OpenGL существуют три вида буферов.

  Кадровый буфер. Когда итоговое изображение сформировано, его можно отправить сразу на экран, либо в кадровый буфер. Этот буфер хранит информацию о 3D объектах, их положение, видимые части, пересечения. Вся эта информация о пикселях хранится в буфере, в виде двоичного массива. Кадровый буфер является коллекцией буферов визуализации.

  Буфер визуализации. Это временное хранилище для одного изображения. Существуют несколько видов буферов визуализации.

  Буфер визуализации цвета хранит итоговое цветное RGB изображение. Буфер визуализации глубины хранит координату глубины положения объектов. Буфер визуализации шаблона хранит видимую часть объекта. В буфере хранится черно-белое изображение.

  Буфер объектов. Это тоже временное хранилище, однако, информация может храниться в нем на протяжении всей работы программы. Эта информация бывает двух типов: структуры и индексы.

  Структура – это массив, который описывает 3D объект как массив вершин, массив координат или любой другой информации. А массив индексов используется для обозначения того, как будет построена модель на основании массива структур.

  Для примера, рассмотрим куб в пространстве. Он состоит из 8 вершин и 6 сторон. Так как OpenGL работает только с треугольниками, необходимо каждую сторону куба представить в виде двух треугольников, которые будут задаваться тремя вершинами каждый. Таким образом, 6 сторон преобразуются

в 12. Чтобы построить куб, необходимо создать массив с информацией о 8 вершинах. И вместо того чтобы перезаписывать эту информацию для каждой стороны, можно создать массив индексов, в котором каждая комбинация из трех элементов представляет треугольник. Таким образом, можно один раз записать информацию обо всех вершинах и затем многократно ее использовать. Большое преимущество буферов объектов заключается в том, что они оптимизированы для работы непосредственно на графическом процессоре и нет нужды хранить массив в приложении после создания объекта.

- **Растеризация**

  Растеризация – это процесс, когда OpenGL собирает всю информацию о 3D объектах (все координаты, вершины и т.д.) для создания 2D изображения. Изображение подвергается некоторым изменениям и выводится на экран. При рендеринге изображения, можно выбрать, куда его помещать в первую очередь. Можно вывести его сразу на экран, либо в кадровый буфер и уже оттуда, с помощью существующего API на экран. Данная опция зависит от реализации.

Графический конвейер

Графический конвейер, реализуемый OpenGL, представляет собой следующую последовательность действий:

Основная программа заполняет память, находящуюся под управлением OpenGL, массивами вершин. Эти вершины проецируются на пространство экрана, собранные в треугольники и растеризованные во фрагменты, размером один пиксель. Наконец фрагментам присваиваются значения цветов и помещаются в кадровый буфер (см. рис.8). Современные графические процессоры получили больше гибкости за счет того, что переложили серьезную часть работы на специальные программы, называемые шейдерами [12].

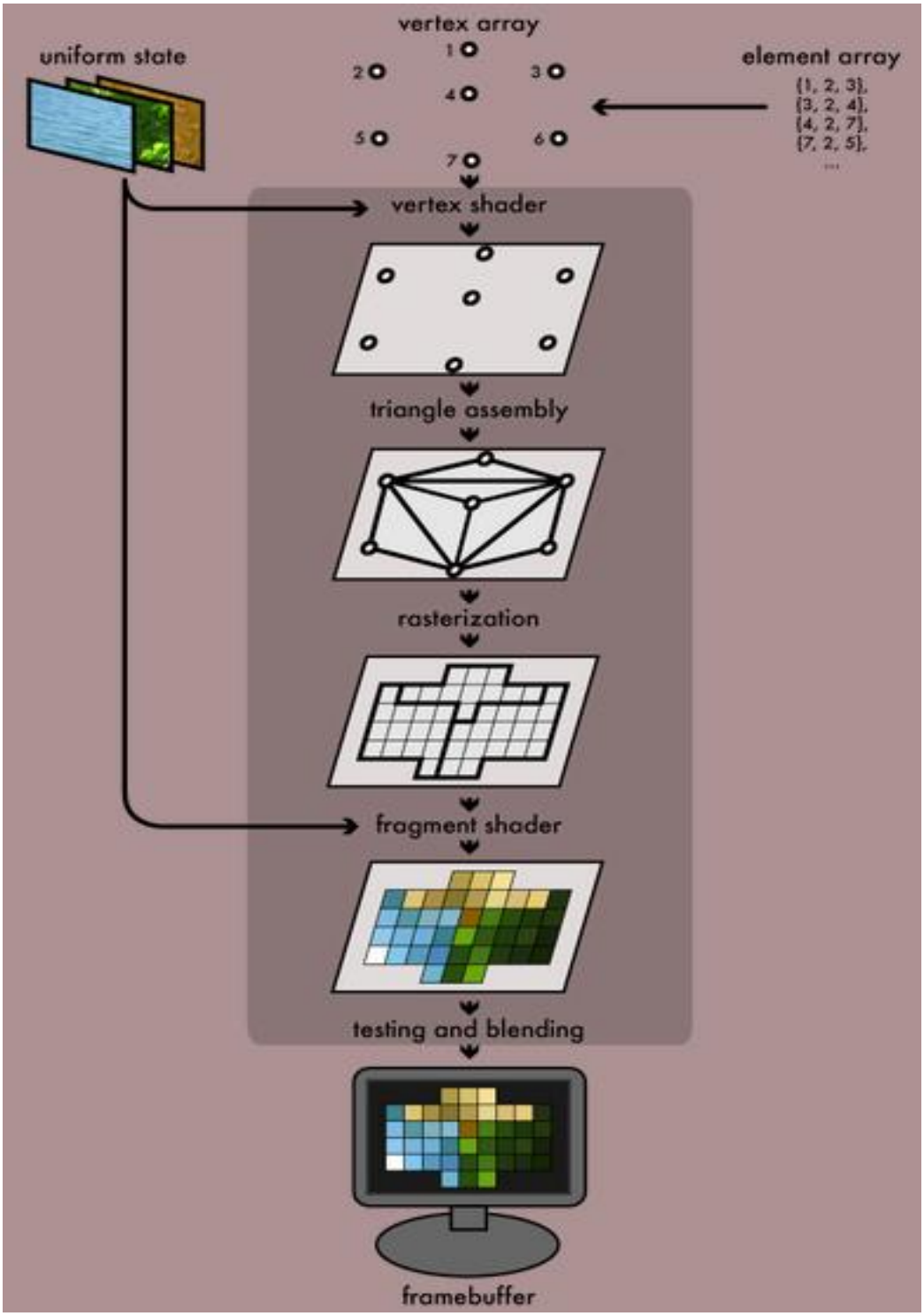

Рис.8 Этапы работы графического конвейера OpenGL.

Рассмотрим подробнее каждый этап (рис.8):

Массивы вершин и элементов.

Работа по выполнению рендеринга начинается на конвейере в наборе одного или нескольких буферов вершин, заполненных массивами атрибутов вершин. Эти атрибуты используются в качестве входных данных для шейдеров. Основные атрибуты вершин включают в себя позицию вершины в 3D пространстве и один или несколько наборов текстурных координат, которые отображают вершину в точку на одной или нескольких текстурах.

Множество буферов вершин, поставляющих данные для рендеринга, в общем случае называются массивом вершин. Дополнительный массив элементов (массив индексов), позволяет определить, какие вершины отправятся на конвейер. Порядок индексов также определяет, как вершины будут собираться в треугольники.

Однородное (uniform) состояние и текстуры.

Работа по выполнению рендеринга имеет специальное однородное состояние, которое обеспечивает набор общих, доступных только для чтения значений на каждом этапе работы конвейера. Это позволяет шейдерным программам использовать параметры, которые не меняются от фрагмента к фрагменту. Такое состояние содержит текстуры, которые хранятся в виде многомерных массивов и которые могут обрабатываться шейдерами. Как следует из названия, текстуры в основном используются для отображения текстурных изображений на поверхность. Они также могут использоваться в качестве справочных таблиц для предварительных вычислений или в качестве наборов данных для различных видов эффектов.

Вершинные шейдеры.

Графический процессор считывает каждую выбранную вершину из массива вершин и пропускает ее через вершинный шейдер, программу, которая принимает набор вершинных атрибутов в качестве входных параметров и на выходе получает набор новых атрибутов, с различными значениями, которые подлежат растеризации. Как минимум, вершинный шейдер вычисляет спроецированное положение вершины на экране. Также шейдер может вычислить дополнительные атрибуты, такие как цвет или текстурные координаты.

Соединение в треугольники.

Затем графический процессор соединяет спроецированные вершины в треугольники. Делается это путем извлечения вершин в порядке, определенном массивом и

группировкой их в наборы из трех элементов. Вершины могут быть сгруппированы несколькими различными способами:

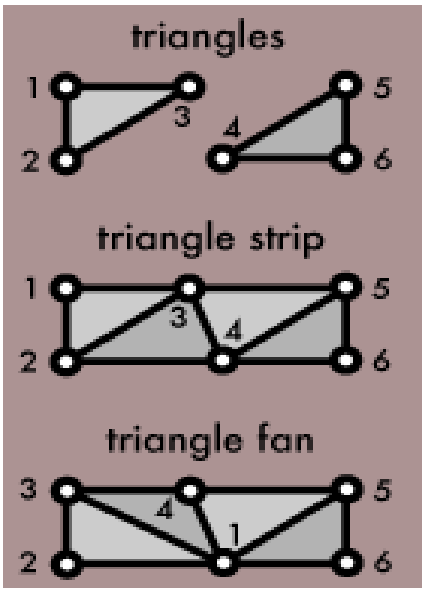

Рис.9 Соединение вершин в треугольники.

Объединение каждых трех вершин в независимый треугольник.

Создание полоски треугольников, где последние две вершины каждого треугольника используются в качестве первых двух вершин следующего треугольника.

Создание веера треугольников, соединяющий первый элемент, с каждой последующей парой элементов.

На рисунке 9 показано, как происходит соединение в различных случаях. Полосы и веера требуют только один новый индекс на каждый треугольник в массиве элементов, после начальных трех, что позволяет увеличить эффективность использования памяти в массиве элементов.

Растеризация.

Растеризатор принимает каждый треугольник, обрезает и отбрасывает части, которые находятся за пределами экрана, и разбивает оставшиеся видимые части на фрагменты, размером в пиксель. Если вершинный шейдер присваивает значение цвета каждой вершине, то растеризатор смешает эти цвета по всей пиксельной поверхности, как показано на рисунке 10.

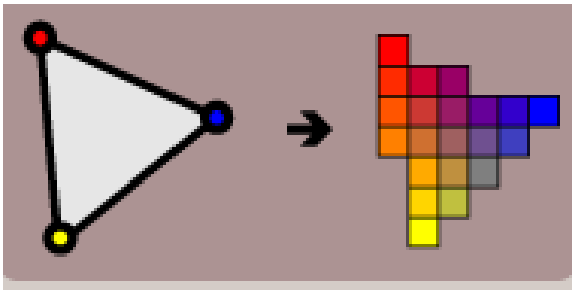

Рис.10 Смешивание цветов вершин по поверхности.

Фрагментные шейдеры.

Сформированные фрагменты затем проходят через еще одну программу, называемую фрагментным шейдером. Фрагментный шейдер принимает на вход данные, полученные вершинным шейдером и интерполированные растеризатором. На выходе получаются значения цвета и глубины, которые затем передаются в кадровый буфер. Основными операциями, выполняемыми фрагментным шейдером являются наложение текстур и освещение. Поскольку фрагментный шейдер выполняется независимо для каждого пикселя, он может выполнять самые сложные операции. Тем не менее, это самая требовательная к производительности часть конвейера.

Кадровые буферы, тестирование и смешивание.

Кадровый буфер является конечным пунктом для вывода результатов рендеринга. В дополнение к стандартному кадровому буферу, большинство современных реализаций OpenGL позволяют создавать объекты кадрового буфера, которые можно поместить в закадровые кадровые буферы или в текстуры. Эти текстуры можно использовать как входные данные для другого рендеринга. Сам по себе кадровый буфер больше чем простое 2D изображение. В дополнение к одному или нескольким цветовым буферам, кадровый буфер может иметь буфер глубины или шаблона, оба из которых опционально фильтруют фрагменты, до того, как они будут помещены в кадровый буфер.

Тестирование глубины отбрасывает те фрагменты, которые располагаются за уже нарисованными, а шаблонное тестирование использует шаблоны рисования для ограничения отрисовываемой части кадрового буфера. Фрагменты, которые остались после такой фильтрации имеют собственное значение цвета, смешанное с перезаписанным значением и финальные значения цвета, глубины и шаблона записываются в соответствующий буфер.

Этот процесс обработки данных, от вершинного буфера до кадрового, выполняется каждый раз, при вызове функции отрисовки. Обычно рендеринг включает в себя сразу несколько задач отрисовки, и полученные в итоге результаты объединяются.

Таким образом, с использованием данного инструмента и была разработана требуемая 3D модель.

3.3 Технологии создания графиков, гистограмм и графов

Для решения данной задачи была выбрана свободная независимая библиотека QCustomPlot [13]. Она предназначена для рисования 2D графиков, гистограмм и графов и предоставляющая высокопроизводительную визуализацию в приложениях в реальном времени. Каждый виджет, созданный с помощью данной библиотеки, обладает возможностью масштабирования и перемещения изображения, а также отображением легенды.

Все созданные в данной работе графики и гистограммы созданы с помощью данной библиотеки. В качестве входных параметров, функция отрисовки принимает два вектора, один для оси абсцисс и один для оси ординат. Для построения верного графика, каждому i-му элементу одного вектора, должен соответствовать i-й элемент из второго вектора. Таким образом, все используемые данные, подготавливались в соответствующем формате.

4. Анализ результатов работы

Разработанная система управления и визуализации является надстройкой над существующим программным комплексом «Гаплоидный эволюционный конструктор 3D» (рис.11).

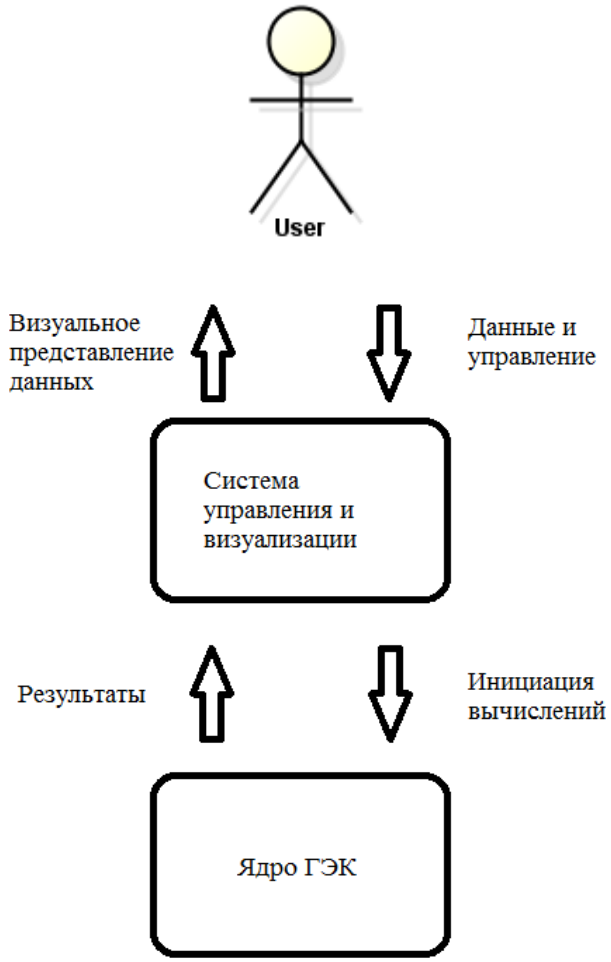

Рис.11 Схема взаимодействия пользователя с системой.

Вычислительное ядро ГЭК представляет собой динамическую библиотеку, взаимодействие с которой происходит посредством специальных скриптов, загружаемых пользователем. Система управления инициализирует начало вычислений с параметрами, указанными в скрипте. Вычислительное ядро возвращает результаты, которые обрабатываются системой визуализации и выводятся на экран.

Общая архитектура системы управления и визуализации программного комплекса «Гаплоидный эволюционный конструктор 3D» выглядит следующим образом (рис.12):

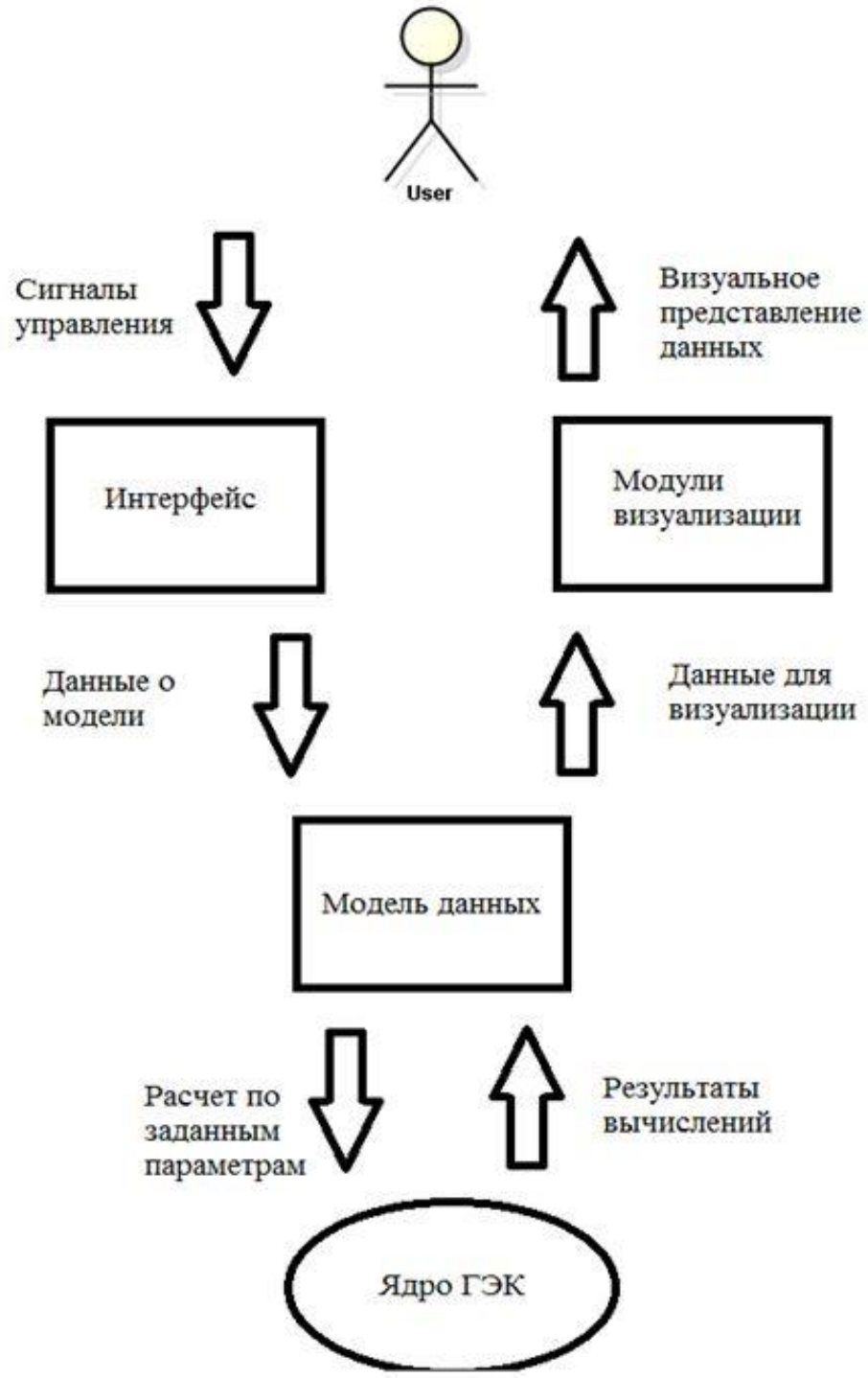

Рис.12 Архитектура системы управления и визуализации.

Взаимодействие пользователя с системой управления и визуализации реализуется следующим образом:

Пользователь загружает собственный сценарий (скрипт, описывающий исследуемую модель (пример скрипта показан на рис.14)), происходит считывание и разбор скрипта, заполнение параметров создаваемой модели. Рассчитываются и сохраняются результаты вычислений на протяжении указанного количества итераций (в рамках данного программного комплекса – количество поколений). После того, как компьютерная модель создана, при помощи модулей визуализации из нее извлекаются данные, подлежащие отображению. Размеры популяций нормируются и задают радиус для сфер. Размерность, заданная в скрипте, определяет форму исследуемой 3D модели. Полученные данные поступаю на вход функции, использующую технологию OpenGL для отрисовки модели. Полученная модель центрируется относительно начала координат и рисуются прозрачные кубы, для каждой ячейки. В центр каждого куба помещается сфера, с радиусом, соответствующим концентрации популяции в конкретной ячейке. При включении анимации по количеству итераций, из модели данных извлекается ранее рассчитанные данные о размерностях популяций на каждой итерации, которые и отображаются на виджете с заданным интервалом времени.

Изменения размерности популяций и концентрации субстратов с течением времени записываются в соответствующие вектора, предназначенные для передачи в функцию отрисовки графиков с использованием библиотеки QCustomPlot. Обозначения трофических взаимосвязей, хранящиеся в векторах, также используются для отображения текущих цепей питания в виде графа на отдельном виджете. Информация о генетическом спектре отображается в двух разных видах на отдельных виджетах. Соответствующая информация извлекается из созданной модели данных, приводится к требуемому формату и передается в функцию отрисовки.

Таким образом, все взаимодействия интерфейса с ядром ГЭК происходит через создаваемую в результате считывания скрипта и расчета необходимого числа итераций модель данных. В процессе визуализации, необходимая информация извлекается из модели соответствующим модулем, обрабатывается и выводится на экран пользователю в виде 3D объектов, графиков, гистограмм и графов.

4.1 Общий вид приложения

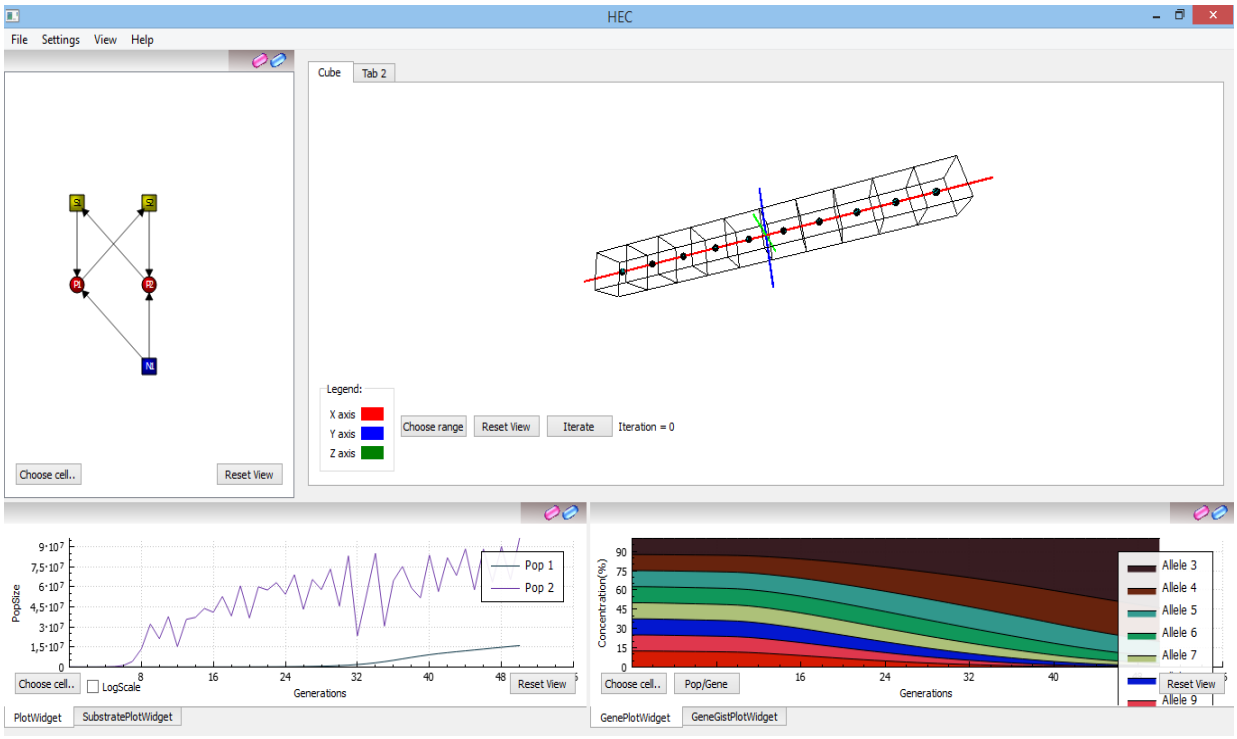

Рис.13 Общий вид приложения.

На рисунке 13 изображен общий вид приложения ГЭК 3D. Система визуализации предоставляет возможность пользователю выбрать скрипт для загрузки и представления данных в графическом виде. Каждый элемент располагается на независимом виджете, который можно перемещать по рабочему полю приложения и располагать в удобном для пользователя месте. Все виджеты разработаны с помощью инструмента среды разработки Qt Designer – специального виджета QDockWidget. Также данные виджеты можно выносить в отдельные окна, для удобства просмотра и закрывать их. Вновь отобразить закрытые элементы можно выбрав в строке меню кнопку “View” и поставить галочку напротив нужного элемента. На основном экране приложения располагаются несколько виджетов, отображающие 3D модель рассматриваемой системы, граф трофических отношений в модели, график динамики роста популяций на протяжении установленных итераций, концентрации субстратов и гистограммы, визуализирующие генетический спектр.

Пользователь может открыть и загрузить новый скрипт, нажатием кнопки мыши на пункт меню “File – Open..” и выбрать требуемый скрипт, как показано на рисунке. Программный комплекс произведет все необходимые расчеты и представит данные в таком виде, как изображено на предыдущем рисунке.

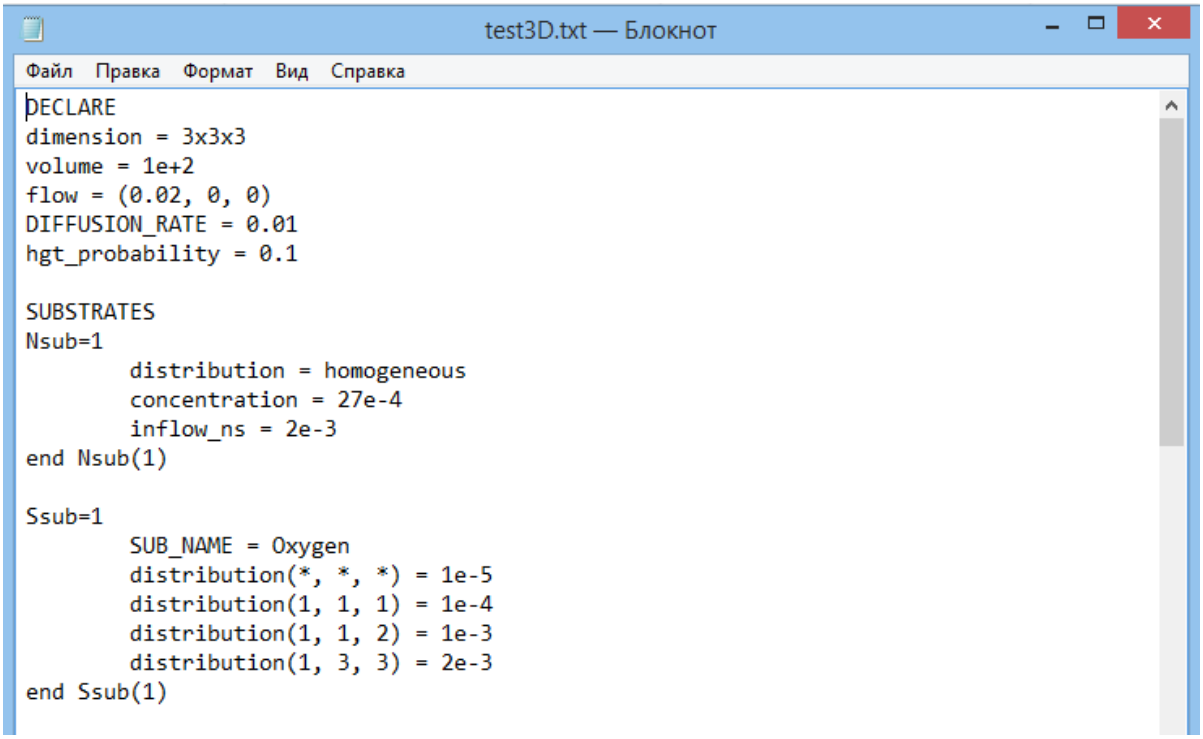

Рис.14 Пример скрипта.

4.2 Граф трофических взаимоотношений

Все популяции связаны между собой трофическими цепями. Каждая популяция может производить или потреблять субстрат, производимый другой популяцией и потреблять неспецифические субстраты, приносимые протоком.

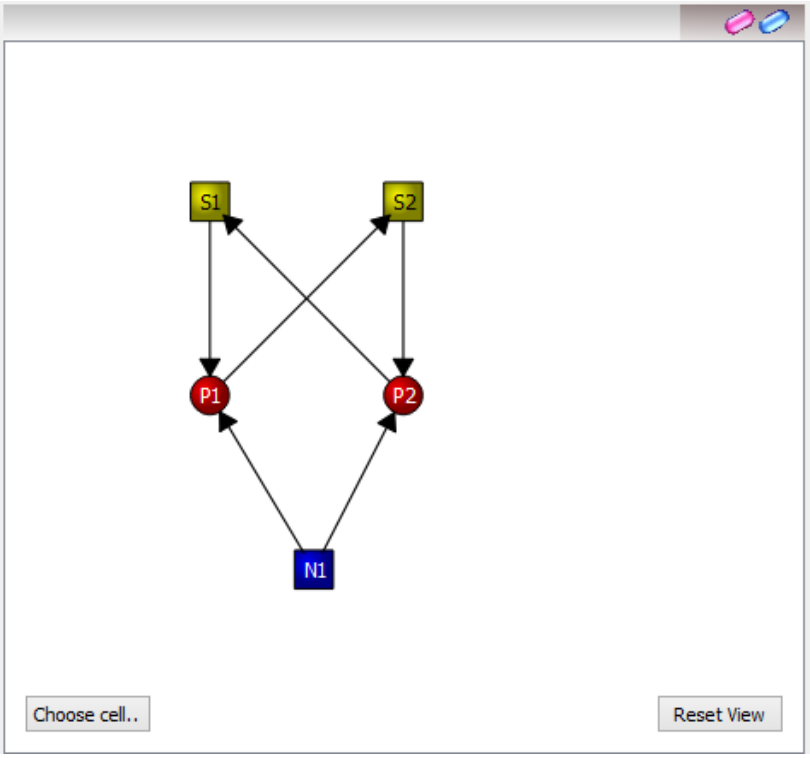

Рис.15 Виджет для отображения графа трофических связей.

На рисунке 15 изображен виджет, визуализирующий трофическую цепь в загруженной модели. Желтыми квадратами обозначены специфические субстраты, синими квадратами – неспецифические. Красные кружки обозначают популяции. Входящие стрелки отображают потребление субстратов, исходящие – производство. Для удобства при просмотре можно перемещать курсором элементы графа и приближать/удалять граф при помощи колесика мыши.

По умолчанию, граф отображается для всего объема заданной модели в целом. Но поскольку на протяжении нескольких поколений данный граф может меняться в конкретной ячейке, было предусмотрено специальное меню, вызывающееся кнопкой «Choose cell..».

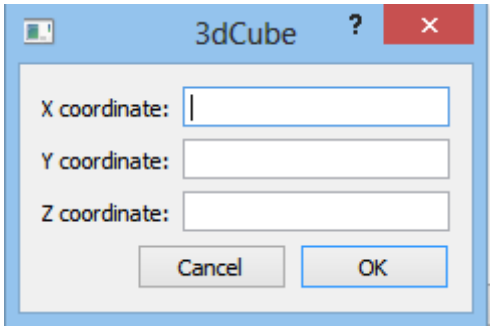

Рис.16 Меню выбора ячейки для виджета с графом трофических отношений.

Данное меню (рис.16) позволяет ввести координаты ячейки и подробно рассмотреть соответствующий граф. Для возврата к исходному виду нажмите кнопку «Reset View».

4.3 Визуализация динамики численности популяций

При исследовании модели, бывает необходимо проследить динамику изменения численности конкретной популяции на протяжении некоторого интервала времени. Такую информацию можно посмотреть для всего объема и для конкретной его ячейки.

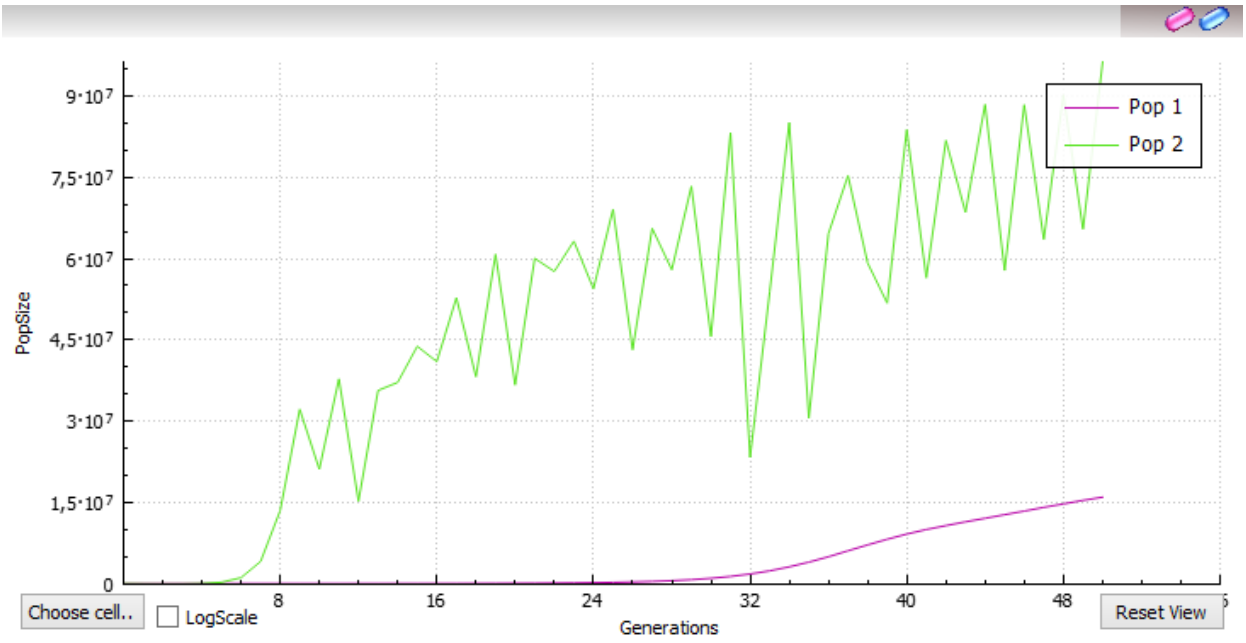

Рис.17 Виджет визуализации динамики численности популяций.

Данный виджет (рис.17) визуализирует динамику роста популяций исследуемой модели на протяжении нескольких итераций. По умолчанию, график изображает динамику популяций по всему объему. Аналогично предыдущему случаю, существует меню для выбора координат конкретной ячейки. Для включения логарифмического масштаба нужно выбрать пункт “LogScale”. Для возврата к исходному виду нажмите кнопку «Reset View».

4.4 Визуализация концентрации субстратов

Не менее важным является наблюдение за изменением концентрации субстратов в объеме.

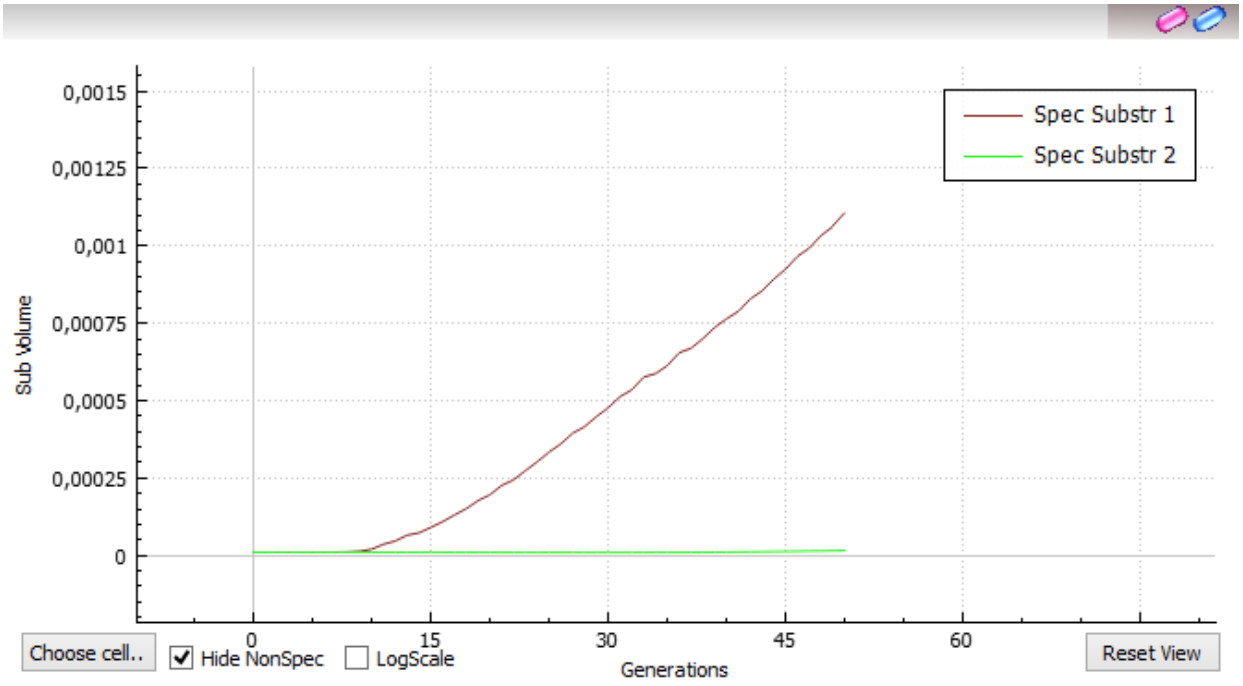

Рис.18 Виджет визуализации концентрации субстратов.

Подобным образом выглядит виджет, предназначенный для визуализации концентрации субстратов (рис.18). Виджет имеет функционал, аналогичный предыдущему, с одним дополнением. Добавлена функция отключения графика неспецифических субстратов, для более наглядного и удобного просмотра графиков специфических субстратов.

4.5 Динамика частот аллелей

Каждая популяция имеет различные вариации одного или нескольких генов, причем частота встречаемости конкретной вариации может меняться с течением времени.

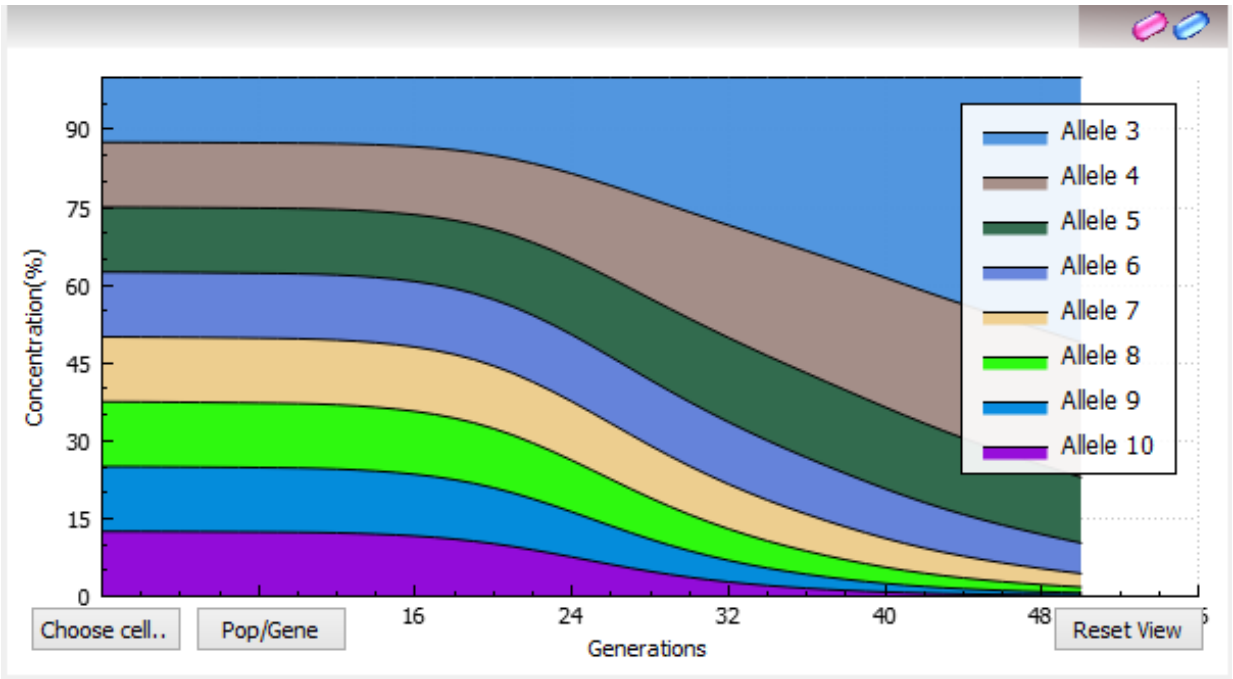

Рис.19 Виджет визуализации динамики частот аллелей в популяции.

На данном рисунке (рис.19) изображена динамика изменений доли аллелей для выбранной популяции конкретного гена по всему объему на протяжении нескольких итераций. Имеется возможность выбрать для просмотра конкретную ячейку, популяцию и ген.

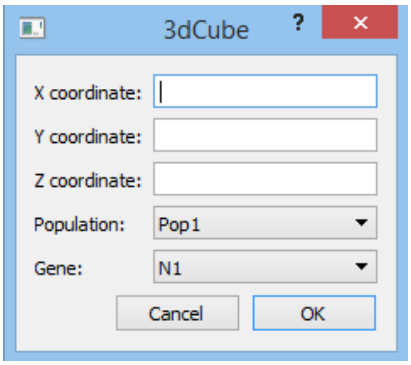

Рис.20 Меню выбора ячейки для виджета с динамикой частот аллелей.

4.6 Визуализация генетического разнообразия

В популяции может быть различное генетическое разнообразие по одному или нескольким генам. Данную информацию можно представить в виде гистограмм, изображающих состояние на текущей итерации. Также можно проследить происходящие изменения при включении анимации по итерациям.

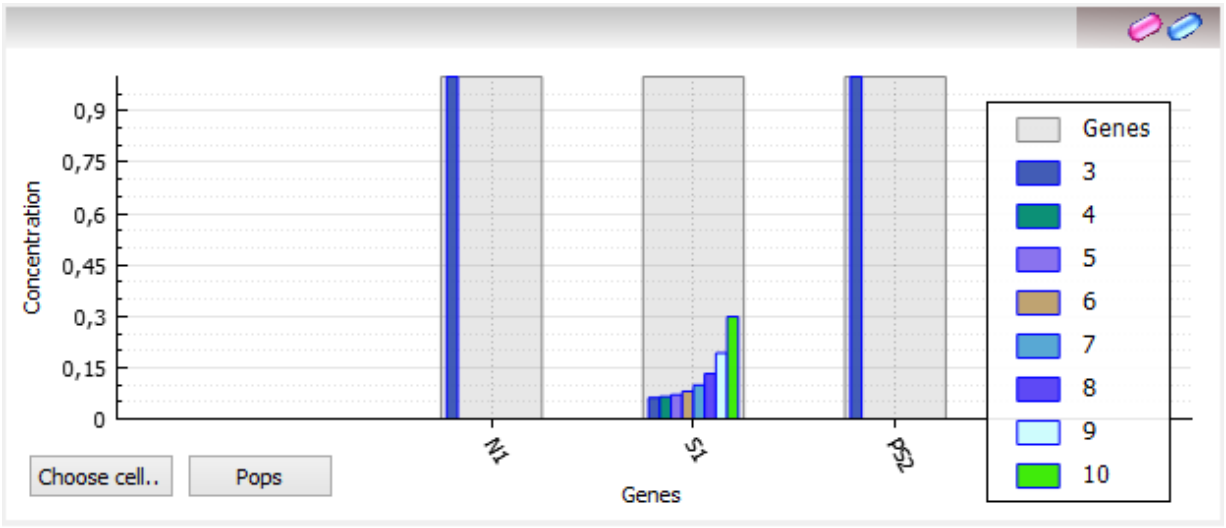

Рис.21 Генетические спектры популяции. Под гистограммами – названия генов.

На данном рисунке (рис.21) представлена визуализация генетического спектра выбранной популяции.

Поле для визуализации гистограмм разделено на категории, соответствующие генам (N1, S1, PS2) выбранной популяции. Цветом показаны значения аллелей (различные формы одного и того же гена), высота столбца соответствует доле аллели (т.е. частоте встречаемости) в совокупном геноме (совокупности генов) популяции. Система позволяет одновременно отслеживать изменение аллельных частот генов в различных популяциях. Аналогично предыдущим виджетам, начальное изображение строится для всего объема. Специальное меню позволяет выбрать конкретную ячейку и популяцию для просмотра.

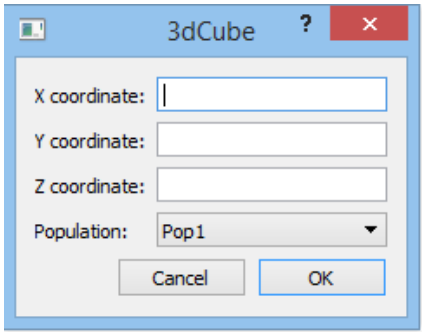

Рис.22 Меню выбора ячейки для виджета с генетическим спектром.

4.7 3D визуализация

Основной задачей при разработке системы управления и визуализации, являлось создание виджета, позволяющего визуализировать различные схемы ГЭК (0D – 3D).

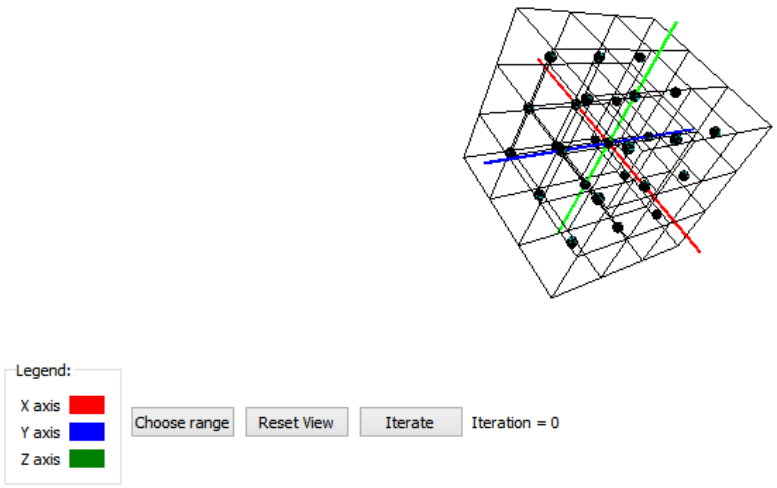

Рис.23 3D визуализация модели.

Данный виджет (рис.23) визуализирует рассматриваемую модель в виде 3D объекта по заданным параметрам. Как уже упоминалось выше, каждый куб соответствует одной ячейке, а сфера внутри – концентрация находящихся в ней популяций. Чем больше концентрация, тем больше диаметр сферы. Для удобства ориентирования, были нарисованы оси координат. Объект можно вращать и масштабировать колесиком мыши.
Создана анимация изменений концентраций популяций на заданном количестве итераций. Для запуска анимации нужно нажать кнопку “Iterate”.

При изучении сложной модели бывает сложно рассмотреть внутренние ячейки. Поэтому для удобства пользователя было создано меню, в котором можно задать координаты «подкуба» и подробно рассмотреть изменения в интересующих ячейках.

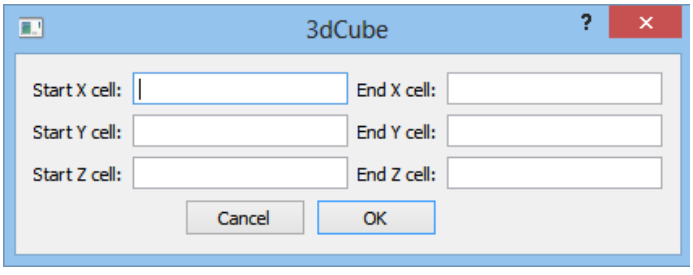

Рис.24 Меню выбора ячеек.

ЗАКЛЮЧЕНИЕ

В результате выполнения работы была достигнута поставленная цель — разработана интерактивная система управления и визуализации программного комплекса «Гаплоидный Эволюционный Конструктор», удовлетворяющая поставленным требованиям.

Для достижения цели была проделана следующая работа:

- разработано архитектурное решение для надстройки системы управления и визуализации над ядром программного комплекса;
- создана архитектура подсистемы управления и визуализации ГЭК;
- реализован графический пользовательский интерфейс для ГЭК, позволяющий настраивать параметры компьютерной модели и визуализировать различные данные в виде графиков, гистограмм, графов и 3D объектов;
- реализованы компоненты управления, обладающие всеми необходимыми функциями, указанными в требованиях к системе.

Разработанная подсистема управления и визуализации делает процесс моделирования более наглядным, обеспечивает гибкость в работе и способствует эффективному анализу получаемых результатов.

Литература

1. Evolutionary Constructor [Электронный ресурс]. – Режим доступа: http://evol-constructor.bionet.nsc.ru/ (дата обращения: 10.04.2014).
2. Ризниченко Г.Ю. Лекции по математическим моделям в биологии. Часть 1. – Ижевск: НИЦ «Регулярная и хаотическая динамика», 2002, 232 с.
3. Ризниченко Г.Ю. Математическая биология [Электронный ресурс]. – Режим доступа: http://www.library.biophys.msu.ru/ (дата обращения: 20.05.2014).
4. AgentCell [Электронный ресурс]. – Режим доступа: http://emonet.biology.yale.edu/ (дата обращения: 27.05.2014).
5. BacSim [Электронный ресурс]. – Режим доступа: http://mic.sgmjournals.org/content/ (дата обращения: 27.05.2014).
6. Aevol [Электронный ресурс]. – Режим доступа: http://www.aevol.fr (дата обращения: 27.05.2014).
7. Лашин С.А., Мамонтова Е.А., Матушкин Ю.Г. Разработка пространственно распределенной модели эволюции прокариотических сообществ //Вавиловский журнал генетики и селекции [Электронный ресурс]. – 2012. - Т 16. - № 4/1. – С. 830-832. – Режим доступа к журн.: http://www.bionet.nsc.ru/vogis/pict _pdf/2012/16_4_1/14.pdf (дата обращения: 25.05.2014).
8. Qt Project [Электронный ресурс]. – Режим доступа: http://qt-project.org/wiki/ (дата обращения: 05.03.2014).
9. Сигналы и слоты в Qt [Электронный ресурс]. – Режим доступа: http://habrahabr.ru/post/50812/ (дата обращения: 25.05.2014).
10. All about OpenGL ES 2.x [Электронный ресурс]. – Режим доступа: http://blog.db-in.com/ (дата обращения: 25.05.2014).
11. OpenGL. Официальное руководство программиста: Пер. с англ./Мейсон Ву, Джеки Нейдер, Том Девис, Дейв Шрайнер. – СПб: ООО «ДиаСофтЮП», 2002. - 592 с. – ISBN 5-93772-041-5
12. An intro to modern OpenGL [Электронный ресурс]. – Режим доступа: http://duriansoftware.com/joe/ (дата обращения: 25.05.2014).
13. QCustomPlot Library [Электронный ресурс]. – Режим доступа: http://www.qcustomplot.com/ (дата обращения: 29.05.2014).