|  |
| --- |
| МБОУ Лицей №17 |
| Индивидуальный итоговый проект |
| Моделирование жизни одноклеточных организмов |
|  |
| **Подготовили: Белоконь Даниил, Соколов Артём** |
| **Под руководством: Виноградовой Юлии Николаевны** |

|  |
| --- |
|  |

**Кострома, 2020 год**

Оглавление

[Введение 3](#_Toc33732870)

[Концепция 3](#_Toc33732871)

[Актуальность 3](#_Toc33732872)

[Польза 3](#_Toc33732873)

[Цель 3](#_Toc33732874)

[Задачи 3](#_Toc33732875)

[Глава I. Теоретическая часть. 5](#_Toc33732876)

[Устройство клетки 5](#_Toc33732877)

[Деление клетки 5](#_Toc33732878)

[Выполнение действий 5](#_Toc33732879)

[Выбор языка программирования 6](#_Toc33732880)

[Глава II. Реализация. 6](#_Toc33732881)

[Планирование 6](#_Toc33732882)

[Структура файлов 7](#_Toc33732883)

[Распределение задач 7](#_Toc33732884)

[Написание кода 7](#_Toc33732885)

[Мир (доска) 7](#_Toc33732886)

[Клетка (бот) 7](#_Toc33732887)

[Стены и органика (мёртвые клетки) 8](#_Toc33732888)

# Введение

## Концепция

Идея проекта заключается в создании небольшого приложения, способного по заданной в её коде модели симулировать развитие одноклеточных организмов (ботов). Организмы живут по программе, прописанной в их «генном коде», массиве чисел, где каждое значение подразумевает то или иное действие, которое необходимо выполнить. В процессе размножения могут происходить случайные изменения в списке команд – мутации. Вследствие постоянной смены поколений должны оставаться только те экземпляры, которые лучше всего подходят под данные условия и способны дать наибольшее количество потомков, а также показывать наиболее выдающийся и впечатляющий результат.

## Актуальность

Футуристы задумывались о подобных симуляциях достаточно давно, но технологии не позволяли обрабатывать большое количество информации, чтобы реализовать эти мечты. В наши дни до сих пор не представляется возможным предусмотреть все известные человеку процессы: малейшие округления способны привести к большим изменениям в предсказаниях учёных, а системы не могут содержать настолько большое количество переменных и законов. Это называется эффектом бабочки или эффектом крыла бабочки[[1]](#footnote-1). Тем не менее, общие модели уже существуют и активно применяются практически во всех сферах деятельности человека, особенно в программировании, где каждый серьёзный проект – это продуманная схема с большим количеством взаимодействий.

## Польза

Данная идея при любой реализации способна стать наглядным примером эволюции, что при профессиональном исполнении может использоваться в качестве ознакомительной площадки для знакомства с упрощёнными принципами развития организмов. Добавление возможности редактирования стартовых условий пользователем позволит проводить домашние эксперименты с искусственно воссозданной моделью мира. В свою очередь, это увеличит понимание изменчивости жизни, тонкости и сложности её настройки.

Для нас, как учеников 9 класса, данный проект является отличным опытом в сфере программирования, так как это большая работа по моделированию реальных объектов и их функций. Каждая пропущенная деталь ведёт к большому обобщению результатов работы программы и как следствие – ухудшению результата.

## Цель

Разработка и реализация упрощённой модели мира, одноклеточных организмов и принципов эволюции.

## Задачи

Поскольку работа по проектированию сложных биологических систем достаточно объёмна, учитывая то, что после создания концепции необходимо всё оформить в программном коде, было принято решение разделять работу по мере появления новых задач и совместной работой над ошибками. По итогу вышло так, как описано ниже.

Белоконь Даниил:

* Работа с самим проектом:
  + Структурирование файлов. Оформление. Создание документации.
  + Выведение результатов на экран.
* Моделирование мира:
  + Смена времён года.
  + Зонирование по уровням света.
  + Изменение условий при смене сезонов.
  + Организация хранения данных.
  + Смена ходов.
  + Получение энергии клетками.
  + Создание клетки и взаимодействий между ними:
    - Поедание.
    - Размножение.
    - Перемещение.
    - Передача энергии.
    - Осмотр территории.
    - Выявление близкой родственности между ботами.
    - Обработка генов.
    - Смена направлений.
    - Проверка направлений и списка на возможность проведения операций.
    - Фотосинтез.

Соколов Артём:

* Работа с проектом:
  + Исправление ошибок, возникающих в коде.
  + Тестирование.
  + Работа с миром:
    - Разделение на зоны для получения минералов.
    - Передача минералов клеткам.
    - Ускорение эволюции на начальных этапах: радиоактивная зона.
    - Работа с клетками:
      * Переработка минералов.
      * Переработка трупов ботов.
      * Обновление состояния.
      * Обеспечение смертности.

Кроме того ещё на моменте изучения теории было начато писание отдельных функций и частей кода, которые будут приведены в пример ниже. Это позволило сэкономить время, так как эти концепции были уже использованы в самой программе.

# Глава I. Теоретическая часть.

Для реализации задуманной идеи понадобятся знания, как в области биологии, так и в сфере программирования. Прежде чем преступать к моделированию мира, необходимо изучить некоторые процессы в одноклеточных организмах.

## Устройство клетки

Во-первых, необходимо выяснить устройство клетки и решить, что именно нужно перенести в программируемую модель. Для начала стоит вспомнить то, что клетки бывают нескольких видов. В наше время выделяют основные 3: бактериальные, животные и растительные[[2]](#footnote-2).

Животные и растительные клетки находятся в структурах многоклеточных организмов и не могут выполнять большое количество отдельных задач, будучи отделенными друг от друга. Поэтому к целям проекта подходят бактерии. Они имеют малые размеры и живут независимо друг от друга.

Моделирование и изменение инициализации клетки происходило на протяжении всей работы с кодом, в итоге получился подобный конечный вариант.

Клетка в проекте обладает энергией, значение которой или заставляет клетку размножаться, или умереть при её недостатке; минералами, которые заменяют разные химические элементы, которыми клетки могут питаться; а также самим генетическим кодом, который представляет из себя массив чисел. Остальные параметры используются для работы самой программы и обеспечения процесса развития и действий.

## Деление клетки

Во-вторых, это само деление клетки. Если обобщить, то хромосомы дублируются, затем выстраиваются посередине родительской клетки, после чего расходятся в дочерние организмы. Именно на этих этапах возникают одни из самых серьёзных мутаций: происходят ошибки при копировании геномного кода или нарушается количество хромосом (в одной клетке хромосом становится больше, в другой – меньше). На картинке в приложении изображено деление не бактериальной клетки, тем не менее, процесс достаточно схож.[[3]](#footnote-3)

Для упрощения этого процесса было решено проводить деление таким образом: сначала из родительской клетки вычитается 150 энергии. Затем со случайной стороны создаётся базовая клетка, в которую передаётся половина минералов и энергии материнского бота, а затем копируется генный код. С шансом в 25% в генном коде дочерней клетки происходит мутация – просто случайное изменение в случайном месте массива. Функция размножения находится в приложении.

## Выполнение действий

В реальности каждый ген отвечает за выработку определённых белков, а действия совершаются уже вследствие их комбинаций и сложных химических процессов. Ради упрощения принципов работы ДНК её функция изменилась с содержания информации об отдельных белках до хранения чисел, каждое из которых означает одно из действий, которое необходимо выполнить[[4]](#footnote-4).

При каждом обновлении состояния клетки (при смене хода) происходит выполнение действия, обозначенного курсором в массиве чисел. К некоторым (а в перспективе и ко всем) значениям приравнено выполнение определённых функций.

Некоторые являются простыми, и после них действие передаётся следующей клетке. Однако остальные взаимодействуют с последующими «генами». Они получают их значения и работают в зависимости от аргументов. К примеру, такие функции, как перемещение и осмотр местности делят следующее за генами их вызвавшими число на 8, берут остаток и взаимодействуют с той стороной, которая соответствует этому числу. Кроме того, каждая функция смещает курсор на определённое количество единиц.

## Выбор языка программирования

В мире существует большое количество не только естественных языков, но и языков программирования. Практически каждый создаётся с какой-то определённой целью и имеет свою специфику. Это необходимо для выполнения различных задач. К примеру, одним из самых низкоуровневых языков (т.е. находящихся ближе всего к физическому уровню) является ассемблер. Языками более высокого уровня является, к примеру, С (Си) и его «потомки»: С# и C++. Существует ещё очень много критериев разделения языков программирования: объектно-ориентированность, функциональность и т. д.

Для нашего проекта мы использовали высокоуровневый (близкий к пониманию человеком), интерпретируемый (выполняемый «пошагово») язык программирования Python (питон). Это связано с тем, что его достаточно просто изучить, он имеет лёгкий синтаксис. Вместе с тем, к нему написано большое количество библиотек (подключаемых модулей), которые позволяют легко работать с графической составляющей проекта.

Конкретный графический модуль, который был выбран – это Pygame. Его преимущество – возможность в несколько строчек кода выводить желаемое на экран, а затем работать с результатом.

Сам Python имеет возможность работать как с функциями и классами, так и с простым исполняемым кодом. Поэтому на нём можно запрограммировать как мир и клетку, так и отдельные куски кода, помогающие выполнять программу.

# Глава II. Реализация.

## Планирование

На момент начала работы уже имелись базовые концепции того, как должен быть структурирован код и что должен включать в себя проект.

### Структура файлов

Во-первых, был создан файл “main.py”, с помощью которого запускалась остальная программа. Рядом с ним располагается файл конфигурации, в котором находятся настройки для приложения, которые можно изменять во время работы с кодом.

В той же директории располагается папка “Classes” где хранятся классы, которые импортируются в главный файл или в другие объекты (между собой).

В итоге вышло так, как показано на изображениях, находящихся в приложении.

### Распределение задач

Затем прошло обсуждение зоны ответственности каждого человека. В итоге план не был составлен, но появилось решение, по которому все делали часть, о которой его просили. Также каждый старался проявлять инициативу и предупреждать об этом, чтобы не происходило никаких конфликтов в коде. Часто проводились встречи, во время которых происходило написание кода.

В начале работы были выработаны правила оформления, которые должны соблюдаться на протяжении написания всего проекта. А именно: комментирование каждой функции и каждого класса, пояснение каждой строчки, старание уместиться в 120 символов в одной линии.

## Написание кода

### Мир (доска)

Первым делом был реализован класс доски. Он включал в себя вывод изображений, смену ходов и времён года, передачу энергии и минералов, а также массив данных, в котором находятся клетки, стены и органика (мёртвые организмы).

В процессе написания этого класса использовалась библиотека Pygame. По мере необходимости приходилось обращаться к официальной документации[[5]](#footnote-5). В проекте использовалась лишь небольшая часть того, что может в целом данный модуль, а именно: создание окна, заливка, смена кадров, вывод разных видов прямоугольников по заданным параметрам и координатам.

### Клетка (бот)

Класс клетки является самым трудоёмким, так как именно в нём происходят все процессы, ведущие к размножению, мутациям и как следствие – эволюции. Поэтому после оформления задумок разработка началась именно с него, а функции в остальных файлах создавались лишь по мере необходимости. Наибольшие проблемы вызвали возникающие в процессе сложного взаимодействия ошибки. Самыми распространёнными являлись: моменты с неправильным указанием ячеек списков; выявлением типов объектов, которые являлись неверными, несмотря на пройденные проверки; а также выход за пределы списка, хотя все меры предосторожности были предприняты.

В самом классе ботов не используется библиотека Pygame – нет необходимости отображать процессы на экране. Но были использованы некоторые интересные алгоритмы проверок ходов: переключатель True-False, который должен быть согласован с родительской доской; превентивная проверка и обработка получение данных, без которых не получилось бы реализовать большинство функций.

### Стены и органика (мёртвые клетки)

Органика и стена были добавлены в процессе написания самой клетки. Органика означает умерших ботов и постепенно падает вниз. При её поедании восполняется некоторое количество энергии. Также боты умеют отличать её от других объектов. Это можно увидеть в функциях просмотра, перемещения и поедания. Стена – это ограничение по перемещению, которое наносит урон организмам, если те попытаются их съесть. Их код можно найти в приложении к проекту.

Программа, несмотря на свою сложную архитектуру и большое количество деталей, которые необходимо было учитывать, вышла достаточно простой для чтения. На это также повлияло то, что на каждой строке находится по комментарию, разъясняющему то, что выполняется.

# Глава III. Результаты.

## Возникавшие трудности

Проект вышел довольно объёмным по сравнению с привычными программами, которые писались до этого. Большое количество взаимодействий создавало некоторые сложности.

На протяжении всей разработки появлялись ошибки, причину существования которых на последних этапах было довольно сложно отследить.

Во-первых, на ранней стадии развития проекта зачастую был хаос в распределении обязанностей, но с позднее эта проблема была решена путём создания списка необходимых изменений. Каждый брался за определённый пункт и при выполнении задачи удалял её оттуда, после чего обновлял нумерацию списка с помощью написанной дополнительно программы.

Во-вторых, на протяжении написания функций, взаимодействующих с миром и курсором, постоянно возникали ошибки, указывающие на то, что выполнение списка действий не зациклено или производились попытки выйти за пределы массива данных.

В-третьих, это проблемы с типизацией. А именно: недочёты в обозначении, к какому классу принадлежит экземпляр, находящийся в ячейке сбоку от клетки. Из-за этого интерпретатор пытался вызвать те функции, которые не находились в полученном объекте. Но это были проблемы во встроенных функциях самого Python.

Это были самые распространённые ошибки, которые препятствовали работе и не давали получить желаемый результат.

## Клетки

### Фотосинтезирующие

Фотосинтезирующие клетки являются самым простым видом из всех одноклеточных. Они создаются обычным нажатием левой кнопкой мыши. Тем не менее, они имеют свои особенности.

Получение энергии и некоторые мутации достаточно сбалансированы, вследствие чего они достаточно быстро размножаются и образуют небольшие колонии. Научившись перерабатывать трупы и обмениваться энергией, они начинают экономить место и оптимизировать расход энергии, тем самым выживая достаточно большое время. Вследствие большого количества размножений скопления являются колыбелью новых видов организмов.

При последующем развитии, если у них не возникает возможности постоянно создавать новые образования, они быстро погибают, так как другие клетки успешно с ними конкурируют.

### Охотники

На базе фотосинтезирующих клеток часто возникают хищные клетки, которые не избавляются от генов получения энергии «от солнца». Данная вариация команд составляет идеальный баланс, так как подобные боты (обозначенные на доске жёлтым цветом) получают большое количество энергии и могут регулировать свою численность.

Их функция в основном заключается в сокращении популяции колоний: находясь в центре, они помогают не выработавшим ген переработки трупов организмам в уничтожении «кладбищ» и старых и неэффективных клеток. Научившись передвигаться, они способны уничтожать целые колонии или фатально нарушать их систему. Кроме того, практически пропадает проблема с пищей и местом размножения.

Подводя итог, можно сказать, что хищники в проекте настолько же необходимы, как и в реальной жизни.

### Перерабатывающих минералы ботов

Фотосинтезирующие колонии часто выводят виды, стоящие между выработкой энергии от света и её получением от минералов, которые накапливаются внизу доски. Их преимущество заключается в том, что получение энергии у них не зависит от наличия других колоний (как у хищников), поэтому они способны очень быстро размножаться, при этом находясь вместе. Поэтому среди них часто возникают мутации, создающие «абсолютные клетки», обладающие всеми видами получения энергии.

Также стоит упомянуть, что минералы защищают клетки от получения урона, вследствие чего хищники способны отбивать атаки на них.

### «Абсолютные»

Неконтролируемая эволюция со временем выводит универсальные клетки, способные получать энергию всеми доступными способами. Их деятельность может лежать в любой плоскости. Но, несмотря на парадоксальность, большое разнообразие доступных механизмов действия лишь мешает им выживать на достойном уровне. Энергия часто расходуются впустую, а работа, которую они должны выполнять, прерывается исполнением других команд, что нарушает экосистему и губит их самих.

Тем не менее, иногда генетический код оптимизируется, вследствие чего возникают боты, способные выживать практически в любых условиях. Наличие большого количества условных переходов избавляет их от траты лишней энергии и совершения ненужных действий.

## Явления

Отдельным пунктом стоит рассмотреть явления, возникающие в процессе выполнения программы. Некоторые из них достаточно редки, другие появляются при каждой выполнении программы. Тем не менее, их существование свидетельствует о возможности даже в простой модели возникновения более сложных и комплексных видов жизни.

### Кометы

Иногда какая-либо быстро накапливающая энергию клетка начинает двигаться по направлению вниз или вбок. В таких случаях в «голове» этой структуры постепенно накапливаются клетки, а за ними остаётся хвост из отстающих или остановившихся организмов. Врезаясь во что-нибудь, кометы создают большие колонии, которые постепенно заполняют всю сторону, к которой они прибыли.

### Караваны

Иногда клетки не размножаются так активно, как это происходит в случае с кометами. Обычно, подобные структуры вереницей выходят из колонии и идут в каком-либо направлении, основывая в месте своего прибытия колонию. Очень напоминает караваны переселенцев в 18 – 19 вв.

### Мосты

Быстро разрастающаяся комета или постепенно оседающий широкий караван образуют между старой и новой колониями мост или дорогу, которые связывают колонии между собой.

## «Грабежи»

Охотники постепенно начинают двигаться или образовывать свои колонии. В таких случаях из них получаются «разбойники», которые нападают на соседние колонии и уничтожают их.

Подвидами разбойников являются грабители караванов, которые находят след комет, караванов или мостов и начинаются уничтожать всё на своём пути, постепенно приходя в колонии, наплывая на них большой волной.

## «Лабиринты»

Поскольку «ген поедания» реализован так, что уничтожение соседа происходит только с одной стороны, то в структурированных колониях хищников нередко появляются разнообразные фигуры, оставляющие дороги для своих потомков или других клеток, которые, появившись в этих самых пробелах, быстро уничтожаются.

## Массовые вымирания

Иногда колонии фотосинтезирующих организмов слишком рано выводят хищников, которые начинают бесконтрольно уничтожать популяцию первоначальных ботов. Отсюда следует то, что все существа, не успевшие вовремя удалиться на большое расстояние, поглощаются, а затем и сами охотники погибают от голода. Поэтому эволюция начинается практически с самого начала и ботам, которые смогли выжить, открывается целый мир.

# Заключение

Анализируя ожидания и результат, можно сказать, что получился достаточно большой проект, который удовлетворяет ожиданиям и имеет огромные перспективы на развитие.

Во-первых, стоит сказать про сам мир. Условия вышли достаточными для того, чтобы организмы успешно развивались и могли подстроиться под них. Тем не менее, клеткам достаточно сложно находиться в неприветливом для них мире, поэтому у ботов появляется лишь один шанс не исчезнуть – эволюция.

Задумка эволюции реализовалась достаточно хорошо. Это можно заметить по вымиранию несостоявшихся видов, конкуренции популяций, которые вытесняют друг друга и борются за ценный ресурс – органику. И выживают те, чем геном больше приспособлен к ситуации, которая складывается в мире.

Помимо того, приятным дополнением стало то, что клетки научились складываться не только в большие скопления, но и выводить более сложные формы совместной жизни. Всё это разбавляется яркими и непредвиденными событиями.

В целом, система становится настолько сложной, что уследить за ней на поздних этапах становится практически невозможно.

Иллюстрации, демонстрирующие результаты находятся в приложении к проекту. Видеозаписи и код можно найти в репозитории по ссылке: https://github.com/Denittka/SchoolProject

1. Эффект бабочки или эффект крыла бабочки – свойство некоторых систем, в которых малейшие изменения в переменных могут приводить к большим изменениям в результатах. Исторические заметки на эту тему: https://www.wolframscience.com/reference/notes/971c [↑](#footnote-ref-1)
2. Картинка (в приложении) и информация были взяты из статьи: <https://www.britannica.com/science/cell-biology> [↑](#footnote-ref-2)
3. Иллюстрации (находится в приложении) к делению клетки и сам материал были взяты из образовательной статьи: <https://www.britannica.com/science/cell-biology/Meiosis> [↑](#footnote-ref-3)
4. Код функции находится в приложении. [↑](#footnote-ref-4)
5. Документация по Pygame: https://www.pygame.org/docs/ [↑](#footnote-ref-5)