Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение   
высшего образования   
«Алтайский государственный технический университет им. И.И. Ползунова»

Факультет информационных технологий

Кафедра прикладной математики

Отчет защищен с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Руководитель от вуза\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.Ю. Андреева

(подпись) (и.о., фамилия)

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_2022 г.

(дата)

Отчет об учебной (технологической) практике

«Модель развития одноклеточных организмов»

УП 09.03.04. 1.35 О

Студент группы ПИ-02 Д.В. Остроухов

и.о., фамилия

Руководитель практики доцент, к.ф.-м.н. А.Ю. Андреева

должность, ученое звание и.о., фамилия

Барнаул 2022

ФГБОУ ВО «Алтайский государственный технический университет   
им. И. И. Ползунова»

Кафедра «Прикладная математика»

**Индивидуальное задание**

**На учебную (технологическую (проектно-технологическую) практику)**

(вид и тип практики по УП)

студенту Остроухов Денис Владимирович группы ПИ-02

(Ф.И.О.)

**График проведения практики**

|  |  |  |
| --- | --- | --- |
| **№**  **п/п** | **Наименование работ, выполняемых на практике** | **Сроки выполнения** |
| 1 | Разработка техническое задание | 23.06.2022 |
| 2 | Проектирование объектной модели | 30.06.2022 |
| 3 | Реализация модели в виде программы с графическим интерфейсом | 10.07.2022 |
| 4 | Написание отчета и защита практики | 14.07.2022 |

Руководитель практики от университета АндрееваА.Ю., доцент каф. ПМ

(подпись) (Ф.И.О., должность)

Задание принял к исполнению  Д.В. Остроухов

(подпись) (Ф.И.О.)

**Инструктаж по ОТ, ТБ, ПБ, ПВТР**

Инструктаж обучающегося по ознакомлению с требованиями охраны труда, техники безопасности, пожарной безопасности, а также правилами внутреннего трудового распорядка проведен «20» июня 2022 г.

Руководитель практики от

Университета Андреева А.Ю. доцент каф. ПМ

(подпись) (Ф.И.О., должность)

**Аннотация**

Отчет об учебной практике содержит описание программы, реализующей моделирование развития одноклеточных организмов: техническое задание, структуру данных, описание программного продукта. Код программы на языке JavaScript размещен в репозитории на Github и в приложении Б. В приложении А приведены снимки программы.

Отчет содержит 26 страниц, 8 рисунков, 3 источника литературы.

**Оглавление**

[Введение 5](#_Toc108750048)

[1. Техническое задание 6](#_Toc108750049)

[1.1. Терминология 6](#_Toc108750050)

[1.2. Описание процесса функционирования модели 7](#_Toc108750051)

[1.3. Требования к функциональности программы 7](#_Toc108750052)

[2. Проект программного продукта 8](#_Toc108750053)

[2.1. Математическая модель 8](#_Toc108750054)

[2.2. Диаграмма классов 8](#_Toc108750055)

[2.3. Жизненный цикл объектов модели 10](#_Toc108750056)

[3. Описание программного продукта 11](#_Toc108750057)

[3.1. Выбор средств реализации 11](#_Toc108750058)

[3.2. Описание классов 11](#_Toc108750059)

[Заключение 13](#_Toc108750060)

[Список использованных источников 13](#_Toc108750061)

[Приложение A. Снимки экранных форм пользовательского интерфейса 14](#_Toc108750062)

[Приложение Б. Исходный код 17](#_Toc108750063)

Введение

Моделирование – основной метод исследований объектов, процессов или явлений, с целью получения объяснений этих явлений, а также для предсказания развития явлений и процессов, интересующих исследователей. Задача моделирования - выявить главные, характерные черты явления или процесса, его особенности, его поведение. Моделирование применяется в разных областях человеческой жизни: в медицине, демографии, страховании, социологии, в научно-исследовательской деятельности, в повседневной жизни и даже в компьютерных играх.

Модель (объекта – оригинала) – вспомогательный объект, отражающий наиболее существенные для исследования закономерности, суть, свойства, особенности строения и функционирования объекта-оригинала. Задачи моделирования:

* Понять сущность изучаемого объекта
* Научиться управлять объектом и определять наилучшие способы управления
* Решать прикладные задачи

1. Техническое задание
   1. Терминология

**Поле** – это модель мира, представленная в виде прямоугольника, в границах которого происходит моделирование: передвигаются клетки, отображается еда и взаимодействия.

**Клетка** – это отдельный, двигающийся круг на поле. Её цвет варьируется в зависимости от потребляемой еды, размер от времени жизни на поле.

**Популяция** – количество клеток (еды) на поле.

**Еда –** это отдельный статичный круг на поле. Еда может быть мясом (красным) или травой (зеленым).

**Продолжительность жизни –** срок, в течение которого клетка отображается на поле, двигается и взаимодействует. По истечении срока – исчезает. В модели продолжительность жизни клетки равна 30 секунд или же 30. Жизнь состоит из 5 периодов, в течение которых клетка увеличивается в размере.

**Период жизни** – отрезок жизни у клетки, равный 6 секундам. Жизнь разбивается на 5 периодов. С переходом на следующий период клетка увеличивается в размере.

**Размножение** – появление новой клетки, которая наследует тип потребляемой еды от родителей. Размножение возможно с 3 до 27 лет. Клетка не может размножаться больше 2 раз за жизнь.

* 1. Описание процесса функционирования модели

Модель функционирует на плоском поле и имитирует процесс сумбурного передвижения клеток, случайное взаимодействие этих клеток друг с другом и поедание клетками еды.

После появления в случайном месте карты клетки начинают двигаться в случайном направлении и поедать пищу на своём пути.

Когда клетка поедает траву, то она ближется к становлению травоядным, при поедании мяса клетка ближется к становлению мясоядным. При примерно одинаковом поедании травы и мяса клетка становится всеядной. После поедания пищи она появляется в случайном месте поля.

Клетки соударяются друг с другом. Если во время соударения обе клетки находились в периоде размножения, то помимо отскакивания происходит появление дочерней клетки такого же типа потребляемой пищи.

Если клетка находится на поле больше 30 секунд, то она умирает.

* 1. Требования к функциональности программы

В программе имеется графический интерфейс: поле для отображения карты, таймер, отсчитывающий, сколько лет прошло с запуска модели, кнопка запуска и ускорения.

При нажатии на кнопку СТАРТ на поле появляются клетки и еда, при этом по мере выполнения программы они осуществляют случайное взаимодействие друг с другом. Все они при запуске располагаются в пределах поля случайным образом.

Предусмотрены следующие типы объектов:

* Клетка;
* Мясо;
* Трава.

При нажатии на кнопку Х1 модель проигрывается с обычной скоростью.

При нажатии на кнопки Х2 и Х5 модель проигрывается в 2 и в 5 раз быстрее соответственно.

1. Проект программного продукта
   1. Математическая модель

**Поле модели** – прямоугольный контур шириной *W* и высотой *H* пикселей.

**Взаимодействие объектов:**

**Клетка - Еда**

Клетка поглощает пищу, их взаимодействие происходит, если расстояние между их центрами меньше суммы их радиусов.

В таком случае клетка поглощает встреченную ей на пути пищу.

**Клетка – Клетка**

Клетки могут размножаться, если они столкнулись, то есть расстояние между их центрами меньше суммы их радиусов.

Клетки размножаются и передают своим потомкам тип поедаемой пищи. Каждая клетка с 3 до 27 лет может размножиться до 2 раз жизнь. Если на поле достигнут лимит клеток, то размножения не происходит.

* 1. Диаграмма классов

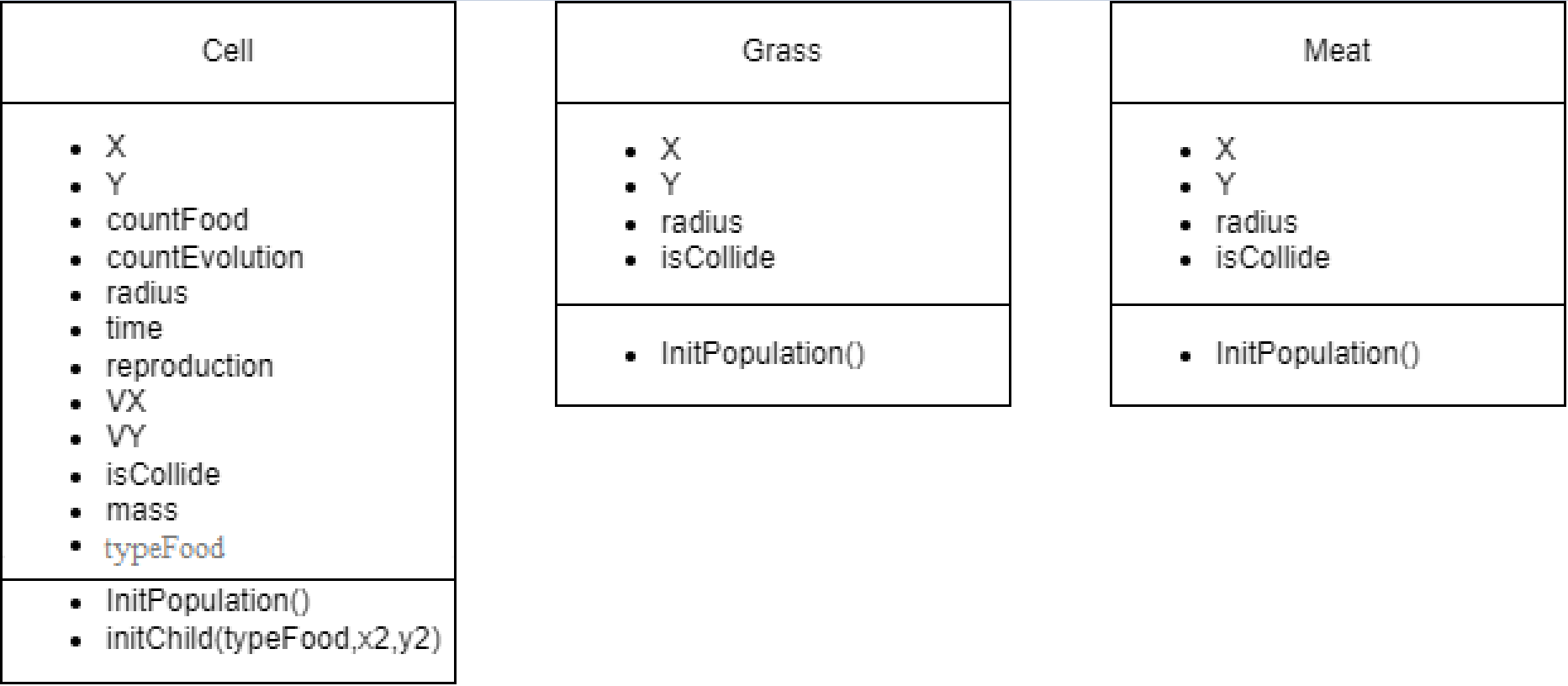


Рисунок 2.2 Диаграмма классов программы

* 1. Жизненный цикл объектов модели

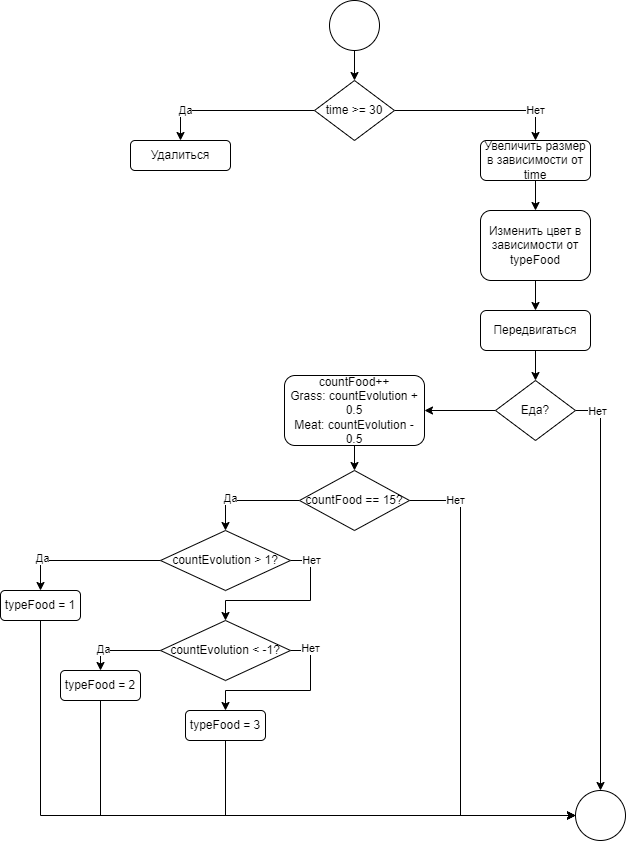


Рисунок 2.2 Диаграмма деятельности для клетки

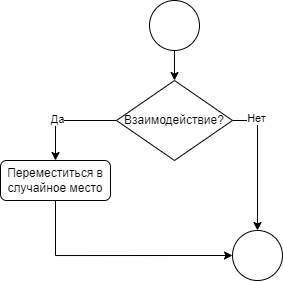


Рисунок 2.3 Диаграмма деятельности для еды

1. Описание программного продукта
   1. Выбор средств реализации

Данная модель была реализована на языке JavaScript с использованием HTML5 Canvas.

* 1. Описание классов

**Классы:**

**Cell** – клетка.

Поля:

* *x –* координата по оси ОХ
* *y -* координата по оси ОУ
* countFood – количество съеденной еды
* countEvolution – счётчик типа съеденной еды
* radius – радиус круга в пикселях
* vx – передвижение по ОХ
* vy – передвижение по ОУ
* time – прожитое время на поле
* isCollide – флаг столкновения
* mass – вес клетки для столкновений
* typeFood – тип поглощаемой пищи

Методы:

* initChild(typeFood,x2,y2) – инициализация потомка
* initPopulation() – инициализация популяции

**Grass** – положительные обстоятельства

Поля:

* *x –* координата по оси ОХ
* *y -* координата по оси ОУ
* radius – радиус круга в пикселях
* isCollide – флаг столкновения

Методы:

initPopulation() – инициализация популяции

**Meat** – отрицательные обстоятельства

Поля:

* *x –* координата по оси ОХ
* *y -* координата по оси ОУ
* radius – радиус круга в пикселях
* isCollide – флаг столкновения

Методы:

* initPopulation() – инициализация популяции

Заключение

Была разработана программа, которая обладает следующим функционалом:

* Визуализация развития одноклеточных организмов, их взаимодействие друг с другом, а также влияние поедаемой пищи на их развитие.

Возможны дальнейшие усовершенствования программы:

* Исправление багов и недоработок.
* Увеличение размеров поля модели, добавление на него других объектов для полноценности модели.
* Добавление статистики.
* Доработка интерфейса.

Список использованных источников

1. Ютуб [Электронный ресурс]. – Режим доступа: https://www.youtube.com/, свободный.
2. Википедия [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/, свободный.
3. Сайт HTMLbook [Электронный ресурс]. Режим доступа: http://htmlbook.ru/, свободный.

Приложение A. Снимки экранных форм   
пользовательского интерфейса

Модель перед запуском

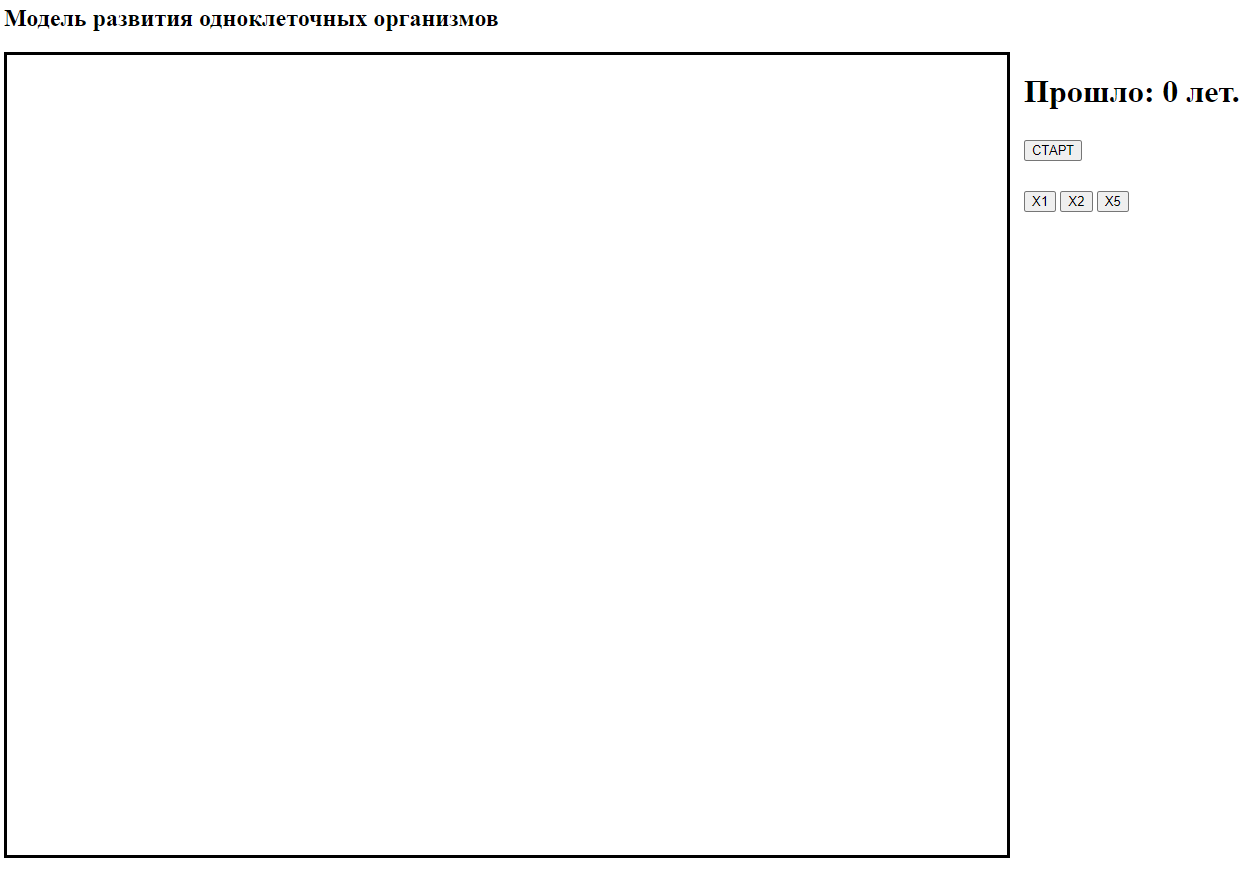


Рисунок A. 1

Старт модели

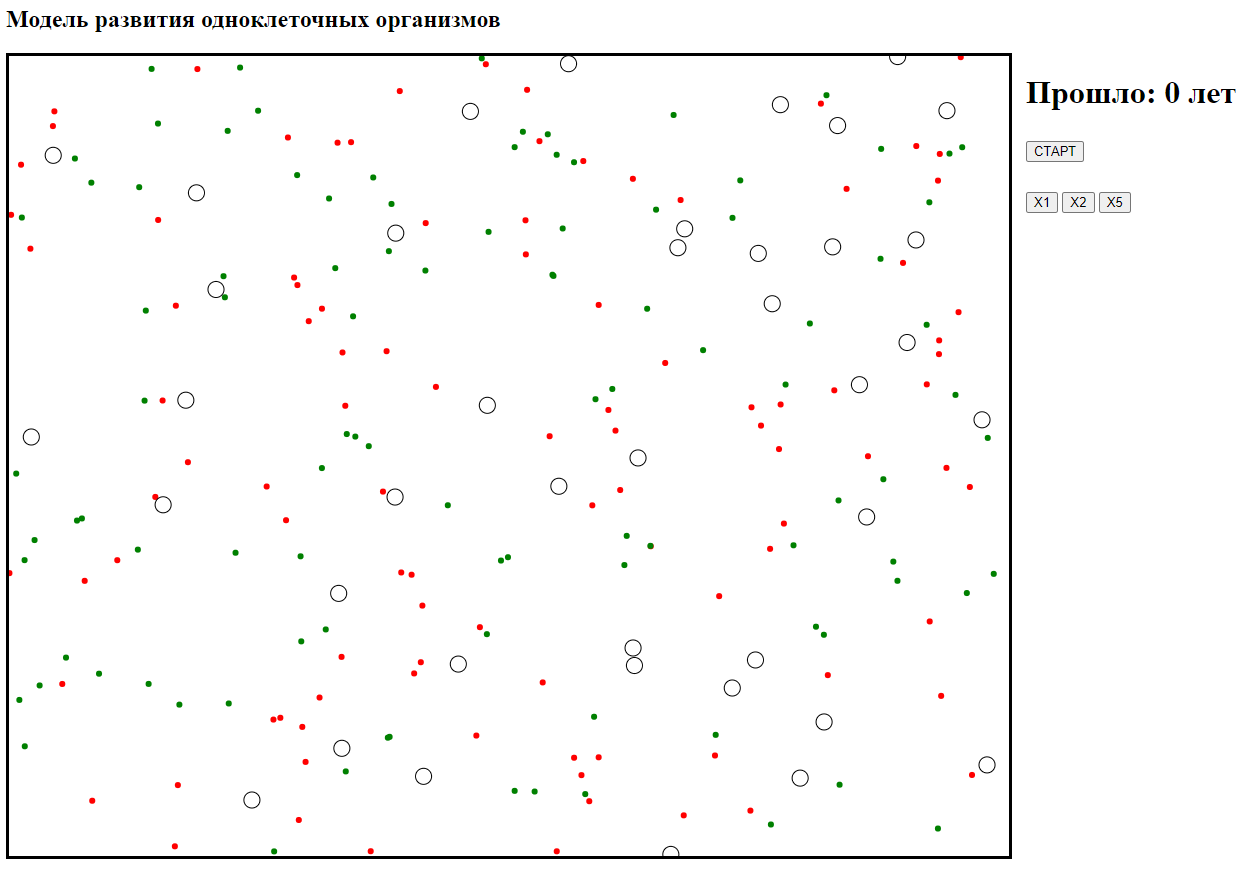


Рисунок A. 2

Развитие клеток

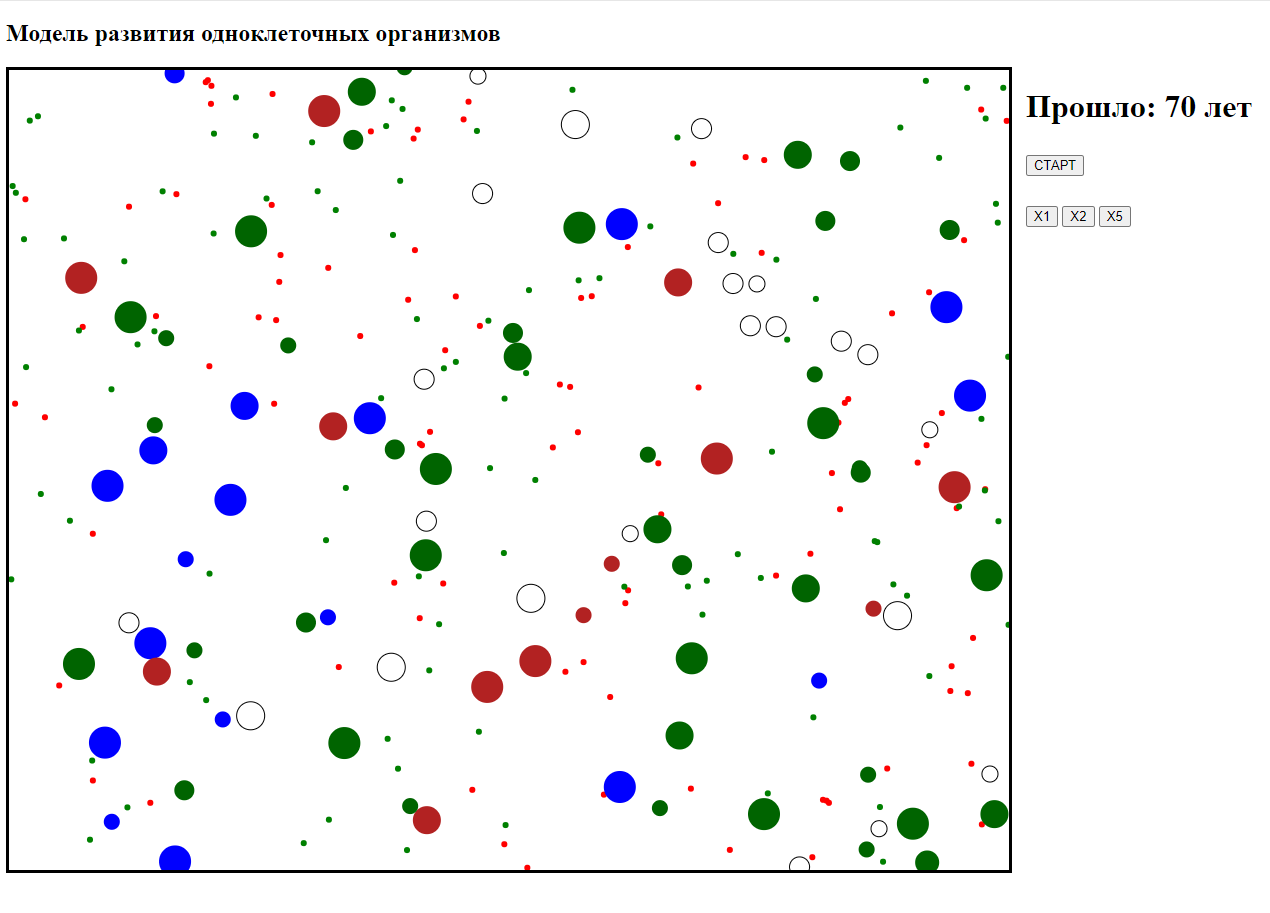


Рисунок A. 3

Доминантным типом стали травоядные

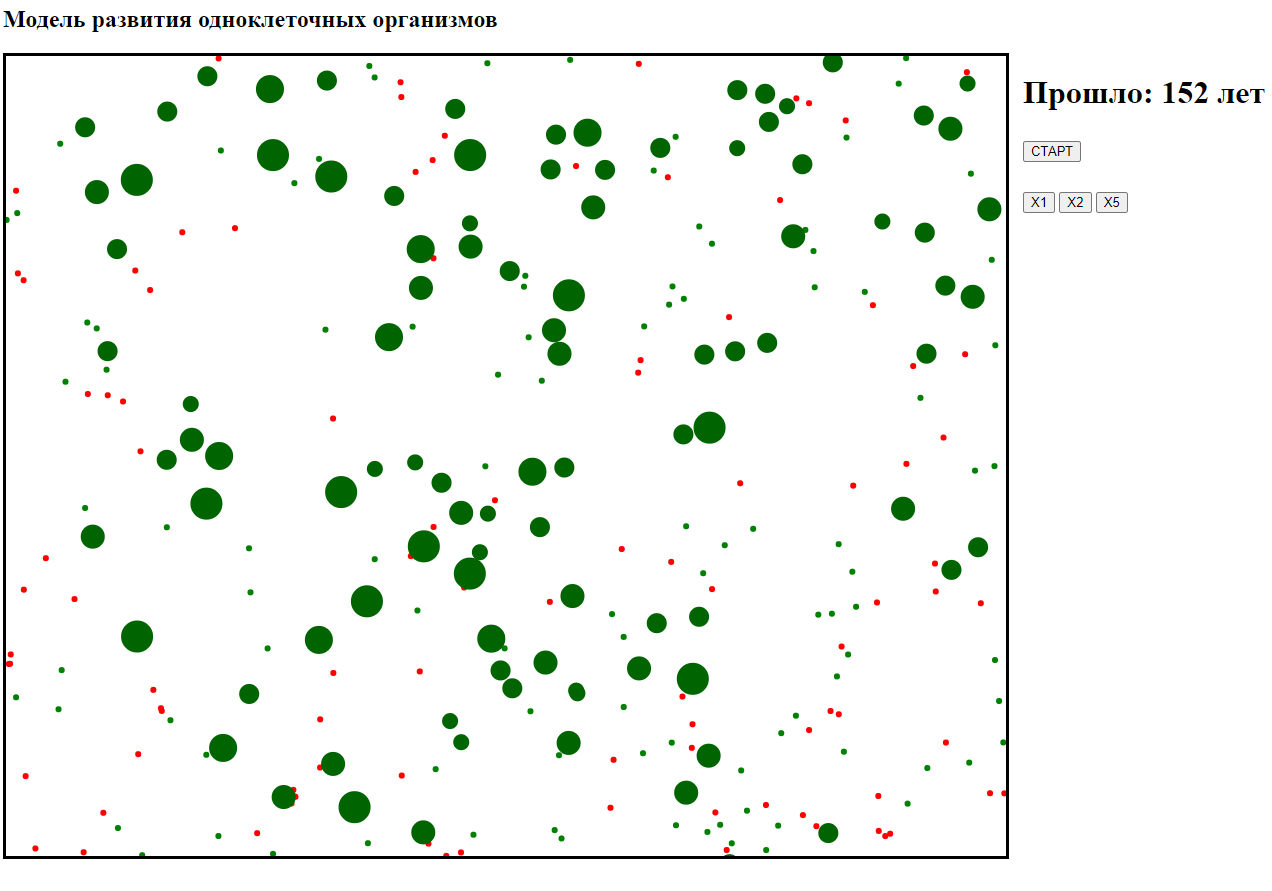


Рисунок A. 4

Смерть всех клеток

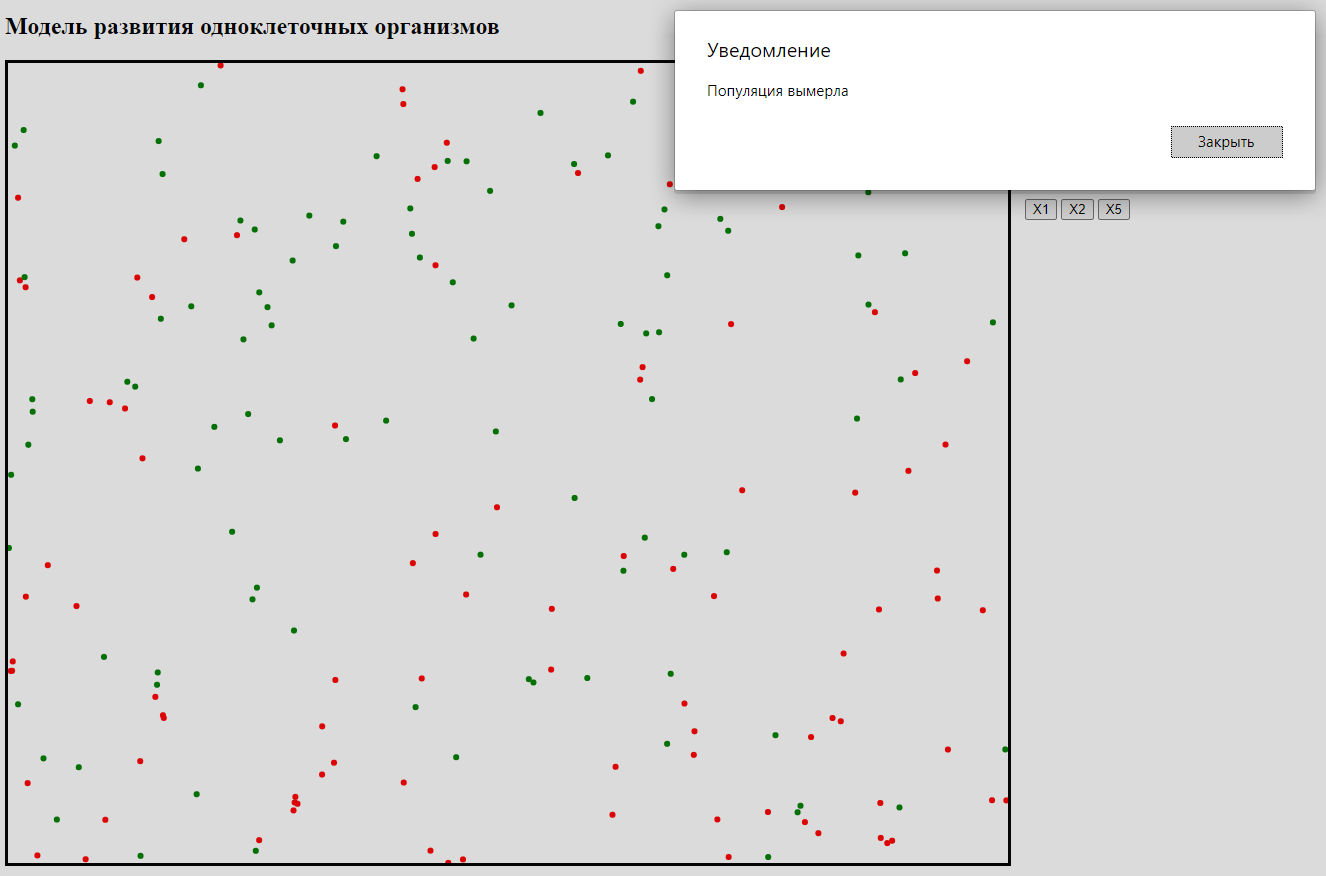


Рисунок A. 5

Приложение Б. Исходный код

<!DOCTYPE html>

<html>

<title>Остроухов</title>

<h2>Модель развития одноклеточных организмов</h2>

<style type="text/css">

TD.leftcol {

width: 1020px; /\* Ширина левой колонки с рисунком \*/

vertical-align: top; /\* Выравнивание по верхнему краю \*/

}

</style>

<body>

<table width="100%" cellspacing="0" cellpadding="0">

<tr>

<td class="leftcol"><canvas id="canvas" width="1000px" height="800px" style="border: 3px solid #000;"></canvas></td>

<td valign="top">

<h1><time>Прошло: 0 лет.</time></h1>

<div style="margin-top: 30px;"> </div>

<div class="row">

<div style="margin-top: 30px;"></div>

<button id="start">СТАРТ</button>

<div style="margin-top: 30px;"></div>

<button id="speed1">X1</button>

<button id="speed2">X2</button>

<button id="speed5">X5</button>

</div>

</div></td>

</tr>

</table>

</body>

<script type="text/javascript">

var canvas, ctx, W, H;

var populationLimit = 0;

var populationCount = 40;

var populationCountMeat = 100;

var populationCountGrass = 100;

var population = [];

var populationMeat = [];

var populationGrass = [];

var launch = 1;

var on\_off = 1;

var sec = 0;

var timerSpeed = 1;

var h1 = document.getElementsByTagName('h1')[0];

//КНОПКИ

var start = document.getElementById('start');

var speed1 = document.getElementById('speed1');

var speed2 = document.getElementById('speed2');

var speed5 = document.getElementById('speed5');

//Функция кнопки СТАРТ

start.onclick = function(){

populationLimit = 0;

on\_off = 1;

sec = 0;

timerSpeed = 1;

h1.textContent = 'Прошло: ' + sec + ' лет';

window.cancelAnimationFrame(null);

deletePopulation();

new Cell().initPopulation();

new Meat().initPopulation();

new Grass().initPopulation();

if (launch == 1) {

timer();

initDraw();

}

launch = 0;

}

//Функция кнопки Х1

speed1.onclick = function(){

timerSpeed = 1;

}

//Функция кнопки Х2

speed2.onclick = function(){

timerSpeed = 2;

}

//Функция кнопки Х5

speed5.onclick = function(){

timerSpeed = 5;

}

window.onload = init;

class Cell {

constructor(x,y, vx,vy, typeFood){

this.x = x;

this.y = y;

this.countFood = 0;

this.countEvolution = 0;

this.radius = 8;

this.time = 0;

this.reproduction = 0;

this.vx = vx;

this.vy = vy;

this.isCollide = false;

this.mass = 1;

this.typeFood = typeFood;

}

initPopulation() {

for (var i=0;i<populationCount;i++) {

let rndX = Math.random()\*W;

let rndY = Math.random()\*H;

var xV = parseInt(Math.random()\*2);

var yV = parseInt(Math.random()\*2);

xV == 0 ? xV = 1 : xV = -1;

yV == 0 ? yV = 1 : yV = -1;

let c = new Cell(rndX, rndY, xV, yV, 0);

population.push(c);

populationLimit++;

}

}

initChild(typeFood,x2,y2) {

let rnd = Math.random()\*2;

let xx,yy;

if (this.x > x2 && this.x + 100 < W){

xx = this.x + rnd;

yy = this.y;

}

else{

xx = x2 - rnd;

yy = y2;

}

if (this.y - 100 > 0)

yy -= rnd;

else

yy += rnd;

var xV = parseInt(Math.random()\*2);

var yV = parseInt(Math.random()\*2);

xV == 0 ? xV = 1 : xV = -1;

yV == 0 ? yV = 1 : yV = -1;

let c = new Cell(xx, yy, xV, yV, typeFood);

population.push(c);

populationLimit++;

}

}

class Meat {

constructor(x,y) {

this.x = x;

this.y = y;

this.radius = 3;

this.isCollide = false;

}

initPopulation() {

for (var i=0;i<populationCountMeat;i++) {

let rndX = Math.random()\*W;

let rndY = Math.random()\*H;

let m = new Meat(rndX, rndY);

populationMeat.push(m);

}

}

}

class Grass {

constructor(x,y) {

this.x = x;

this.y = y;

this.radius = 3;

this.isCollide = false;

}

initPopulation() {

for (var i=0;i<populationCountGrass;i++) {

let rndX = Math.random()\*W;

let rndY = Math.random()\*H;

let g = new Grass(rndX, rndY);

populationGrass.push(g);

}

}

}

function init() {

canvas = document.getElementById("canvas");

ctx = canvas.getContext('2d');

W = canvas.width;

H = canvas.height;

}

//Рисование

function initDraw() {

ctx.clearRect(0,0,W,H);

detectCollisions();

DetectMeatOrGrass();

if (!population.length){

alert('Популяция вымерла');

stop.onclick();

}

for (var i=0;i<populationMeat.length;i++) {

var m = populationMeat[i];

ctx.beginPath();

ctx.arc(m.x,m.y,m.radius,0, Math.PI\*2, 0);

ctx.closePath();

ctx.fillStyle = 'red';

ctx.strokeStyle = 'black';

ctx.fill();

}

for (var i=0;i<populationGrass.length;i++) {

var g = populationGrass[i];

ctx.beginPath();

ctx.arc(g.x,g.y,g.radius,0, Math.PI\*2, 0);

ctx.closePath();

ctx.fillStyle = 'green';

ctx.strokeStyle = 'black';

ctx.fill();

}

for (var i=0;i<population.length;i++) {

var c = population[i];

if (c.time >= 30){

population.splice(i,1);

populationLimit--;

i--;

continue;

}

c.x += timerSpeed \* c.vx;

c.y += timerSpeed \* c.vy;

if ( c.x-c.radius <= 0 || c.x+c.radius >= W ) c.vx = -c.vx;

if ( c.y-c.radius <= 0 || c.y+c.radius >= H ) c.vy = -c.vy;

ctx.beginPath();

ctx.arc(c.x,c.y,c.radius,0, Math.PI\*2, 0);

ctx.closePath();

if (c.typeFood == 0) {

ctx.fillStyle = 'white';

ctx.strokeStyle = 'black';

ctx.stroke();

}

else {

if (c.typeFood == 1) {

ctx.fillStyle = 'DarkGreen';

ctx.strokeStyle = 'black';

ctx.fill();

}

else {

if (c.typeFood == 2) {

ctx.fillStyle = 'FireBrick';

ctx.strokeStyle = 'DarkRed';

ctx.fill();

}

else {

if (c.typeFood == 3) {

ctx.fillStyle = 'blue';

ctx.strokeStyle = 'blue';

ctx.fill();

}

}

}

}

if (c.x - c.radius - 20 > 0 && c.x + c.radius + 20 < W && c.y - c.radius - 20 > 0 && c.y + c.radius + 20 < H){

if (c.time >= 6 && c.time < 12)

c.radius = 10;

if (c.time >= 12 && c.time < 18)

c.radius = 12;

if (c.time >= 18 && c.time < 24)

c.radius = 14;

if (c.time >= 24 && c.time < 30)

c.radius = 16;

}

}

if (on\_off == 1)

window.requestAnimationFrame(initDraw);

}

//Столкновение с едой

function DetectMeatOrGrass(){

var c, g, m;

for (var i=0; i < population.length; i++){

population[i].isCollide = false;

}

for (var i=0; i < populationGrass.length; i++){

populationGrass[i].isCollide = false;

}

for (var i=0; i < populationMeat.length; i++){

populationMeat[i].isCollide = false;

}

for ( var i=0;i<population.length;i++ ) {

c = population[i];

for ( var j=0;j<populationGrass.length;j++ ) {

g = populationGrass[j];

let gdx = c.x-g.x;

let gdy = c.y-g.y;

let gdistance = Math.sqrt(gdx\*gdx+gdy\*gdy);

if ( gdistance < c.radius + g.radius && c.typeFood != 2) {

c.countFood++;

c.countEvolution = c.countEvolution + 0.5;

g.x = Math.random()\*W;

g.y = Math.random()\*H;

if (c.countFood == 15) {

if (c.countEvolution > 1) { c.typeFood = 1}

else {if (c.countEvolution < -1) {c.typeFood = 2}

else {c.typeFood = 3}

}

}

}

}

for ( var j=0;j<populationMeat.length;j++ ) {

m = populationMeat[j];

let mdx = c.x-m.x;

let mdy = c.y-m.y;

let mdistance = Math.sqrt(mdx\*mdx+mdy\*mdy);

if ( mdistance < c.radius + m.radius && c.typeFood != 1) {

c.countFood++;

c.countEvolution = c.countEvolution - 0.5;

m.x = Math.random()\*W;

m.y = Math.random()\*H;

if (c.countFood == 15) {

if (c.countEvolution > 1) { c.typeFood = 1}

else {if (c.countEvolution < -1) {c.typeFood = 2}

else {c.typeFood = 3}

}

}

}

}

}

}

function detectCollisions()

{

var h1, h2;

for ( var i=0;i<population.length;i++ ) {

population[i].isCollide = false;

}

for ( var i=0;i<population.length;i++ ) {

h1 = population[i];

for ( var j=i+1;j<population.length;j++ ) {

h2 = population[j];

let dx = h1.x-h2.x;

let dy = h1.y-h2.y;

let distance = Math.sqrt(dx\*dx+dy\*dy);

if ( distance < h1.radius + h2.radius )

{

let rndChild = Math.random()\*100;

if ((h1.typeFood == h2.typeFood) && (h1.time >= 6 && h1.time < 27 && h2.time >= 6 && h2.time < 27) && populationLimit < 100 && rndChild <= 100 && (h1.reproduction <= 3 && h2.reproduction <=2)){

h1.reproduction++;

h2.reproduction++;

h1.initChild(h2.typeFood,h2.x,h2.y);

}

var coeffImp = 1.5;

h1.isCollide = true;

h2.isCollide = true;

let collision = {

x: h2.x-h1.x,

y: h2.y-h1.y

};

let collisionNormal = {

x: collision.x / distance,

y: collision.y / distance

};

let relV = {

x: h1.vx-h2.vx,

y: h1.vy-h2.vy

};

let speedV = relV.x\*collisionNormal.x+relV.y\*collisionNormal.y;

if ( speedV < 0 ) break;

let impulse = coeffImp\*speedV / (h1.mass+h2.mass);

h1.vx -= impulse\*h2.mass\*collisionNormal.x;

h1.vy -= impulse\*h2.mass\*collisionNormal.y;

h2.vx += impulse\*h1.mass\*collisionNormal.x;

h2.vy += impulse\*h1.mass\*collisionNormal.y;

if ( Math.abs(h1.vx) < 0.5 ) {

if ( h1.vx < 0 ) {

h1.vx = -1;

}

else {

h1.vx = 1;

}

}

if ( Math.abs(h1.vy) < 0.5 ) {

if ( h1.vy < 0 ) {

h1.vy = -1;

}

else {

h1.vy = 1;

}

}

if ( Math.abs(h2.vx) < 0.5 ) {

if ( h2.vx < 0 ) {

h2.vx = -1;

}

else {

h2.vx = 1;

}

}

if ( Math.abs(h2.vy) < 0.5 ) {

if ( h2.vy < 0 ) {

h2.vy = -1;

}

else {

h2.vy = 1;

}

}

}

}

}

}

//Тик для таймера

function tick(){

sec+=1;

h1.textContent = 'Прошло: ' + sec + ' лет';

for (var i=0;i<population.length;i++) {

var h = population[i];

h.time ++;

}

timer();

}

//Таймер

function timer() {

setTimeout(tick, 1000 / timerSpeed);

}

//Удаление всех популяций

function deletePopulation(){

population.splice(0,population.length);

populationGrass.splice(0,populationGrass.length);

populationMeat.splice(0,populationMeat.length);

}

//Остановка рисования

function stopDraw(){

ctx.clearRect(0,0,W,H);

}

</script>

</html>