

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

Преподаватель _____ (подпис
ь)
«__» _____ 2023 г.

Отчет
по расчетному заданию
по дисциплине «Современные средства разработки Web приложений»

Студент гр. ПИ-02
Чередов Р.А.

Старший преподаватель,
Третьяков А. А.

Оглавление

Введение	3
1. Техническое задание	4
1.1. Терминология	4
1.2. Описание процесса функционирования модели	6
1.3. Требования к функциональности программы	7
2. Проект программного продукта.....	7
2.1. Математическая модель.....	7
2.2. Диаграмма классов	9
2.3. Жизненный цикл объектов модели.....	10
3. Описание программного продукта	11
3.1. Выбор средств реализации	11
3.2. Описание классов	11
Заключение	13
Приложение А. Снимки экранных форм пользовательского интерфейса	14
Приложение Б. Исходный код.....	17

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

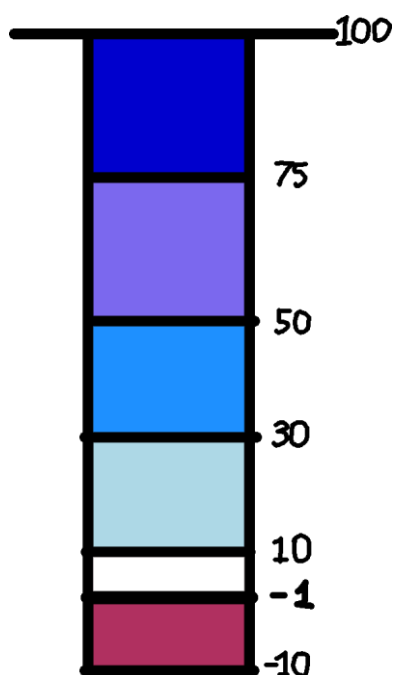
1.1. Терминология

Поле – это модель мира, в виде прямоугольника, в котором передвигаются бактерии и отображаются еда, а также происходит взаимодействие

Бактерия – это движущийся объект, выглядит как круг, изменяет цвет и размер в зависимости от съеденной еды.

Еда – это отдельный статичный круг на поле. Обстоятельство может быть отрицательным (красным) или положительным (зеленым), а также смертельным (черным).




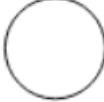
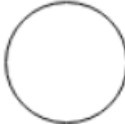
Показатель сытости– показатель от -10 до 100. Уровень сытости отображается цветом бактерии (рис. 1).



(рис. 1) Соотношение цвета и уровня сытости

Продолжительность жизни – срок, в течение которого бактерия отображается на поле, двигается и взаимодействует. По истечении срока – исчезает. В модели продолжительность жизни равна 25 секунд иначе, если голод меньше – 10. Жизнь состоит из 5 периодов.

Период жизни – отрезок жизни, равный 5 секундам (15 лет). Жизнь разбивается на 5 периодов:

1. От 0 до 5 секунд 
2. От 6 до 10 секунд 
3. От 11 до 15 секунд 
4. От 16 до 20 секунд 
5. От 21 до 25 секунд 

С каждым периодом жизни бактерия увеличивается в размере, во время 3-го и 4-го периодов появляется возможность размножаться.

Взаимодействие – столкновение с бактериями или едой

Размножение – появление новой бактерии, наследует половину сытости от родителей

1.2. Описание процесса функционирования модели

Модель функционирует на плоском поле и имитирует процесс взаимодействия бактерий,

с различными обстоятельствами в их короткий промежуток жизни.

В границах рабочей области в случайном месте появляются бактерии с нулевым уровнем развития и начинают хаотичное движение.

А также, появляются еда для бактерий, а именно хорошая, отрицательная и смертельная.

После потребления еды, она появляется в случайном месте.

Когда бактерии кушают, то происходит следующее, зеленая-положительная +5 к насыщению, красная-плохая -5 к насыщению и смертельная - 1000 к насыщению.

Если бактерия вступает в контакт с другой бактерий, то происходит передача насыщения по принципу:

1. Положительно развитый увеличивает уровень насыщения менее развитого на 5.

При этом отнимается у положительного на 5.

2. Менее развитый положительный никак не влияет на более развитого.

3. Более развитый отрицательный никак не влияет на менее развитого.

4. Отрицательно развитый уменьшает уровень насыщения более развитого на 5 от своего насыщения.

Так же при взаимодействии людей 3-го или 4-го периода жизни происходит размножение.

При увеличении периода жизни бактерия увеличивается в размере, при изменении уровня развития меняется цвет. Если продолжительность жизни бактерии закончилась, то она исчезает.

1.3. Требования к функциональности программы

В программе имеется графический интерфейс: поле для отображения карты, счетчик, отсчитывающий, сколько прошло секунд с момента запуска модели, кнопки запуска, остановки и ускорения, а также поля для ввода исходных данных (Изначальное количество бактерий, Максимальное количество бактерий, Кол-во хорошей еды, Кол-во плохой еды, Кол-во смертельной еды).

При нажатии на кнопку СТАРТ на поле отображаются объекты четырех типов, при этом по мере выполнения программы они осуществляют взаимодействие друг с другом. Все они при запуске располагаются в пределах поля случайным образом.

Предусмотрены следующие типы объектов:

- Бактерия;
- Положительная еда;
- Отрицательная еда.
- Смертельная еда.

При нажатии на кнопку СТОП движению объектов останавливается, при еще одном нажатии поле очищается.

При нажатии на кнопку X2 модель проигрывается в 2 раза быстрее.

При нажатии на кнопку X1 модель проигрывается с обычной скоростью.

2. Проект программного продукта

2.1. Математическая модель

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

Взаимодействие объектов:

Бактерия - еда

Еда влияет на бактерию, если они соприкоснулись. Когда расстояние между их центрами меньше суммы их радиусов.

$$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} < r_1 + r_2$$

В таком случае положительное еда прибавляет 5 к уровню сытости бактерии, а отрицательное вычитает 5. Смертельная еда, вычитает 1000.

Бактерия – Бактерия

Бактерии могут передавать свой уровень развития друг другу или размножаться, если они соприкоснулись. Когда расстояние между их центрами меньше суммы их радиусов.

$$\sqrt{(x1 - x2)^2 + (y1 - y2)^2} < r1 + r2$$

Передача опыта:

- 1) Бактерии одного знака развития (+/-):

Более развитый по модулю увеличивает или соответственно уменьшает уровень развития менее развитого на 5 от своего развития.

Менее развитый не влияет на более развитого.

- 2) Бактерии разных знаков:

Увеличивают уровень развития друг друга на 5.

При размножении появляется новая бактерия, уровень развития которого, равен среднему от уровня развития его родителей ($L_{\text{new}} = \frac{L1+L2}{2}$). Бактерии размножаются с вероятностью 0,9, а также не могут размножиться, если на поле уже максимальное кол-во бактерий.

2.2. Диаграмма классов

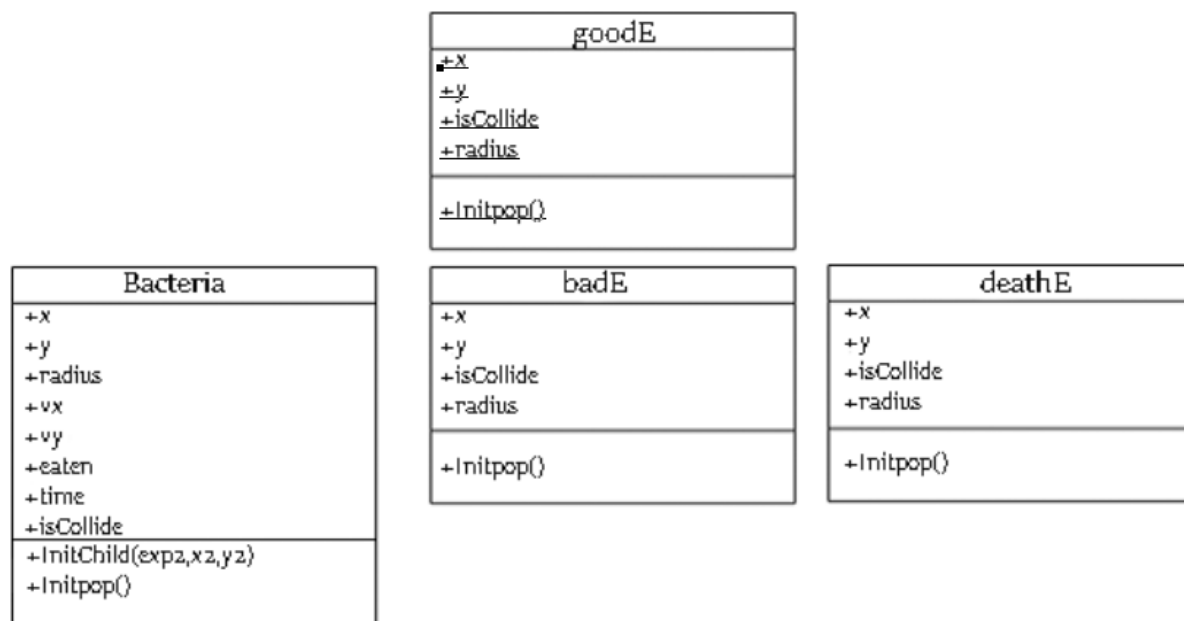


Рисунок 2.1 Диаграмма классов программы

2.3. Жизненный цикл объектов модели

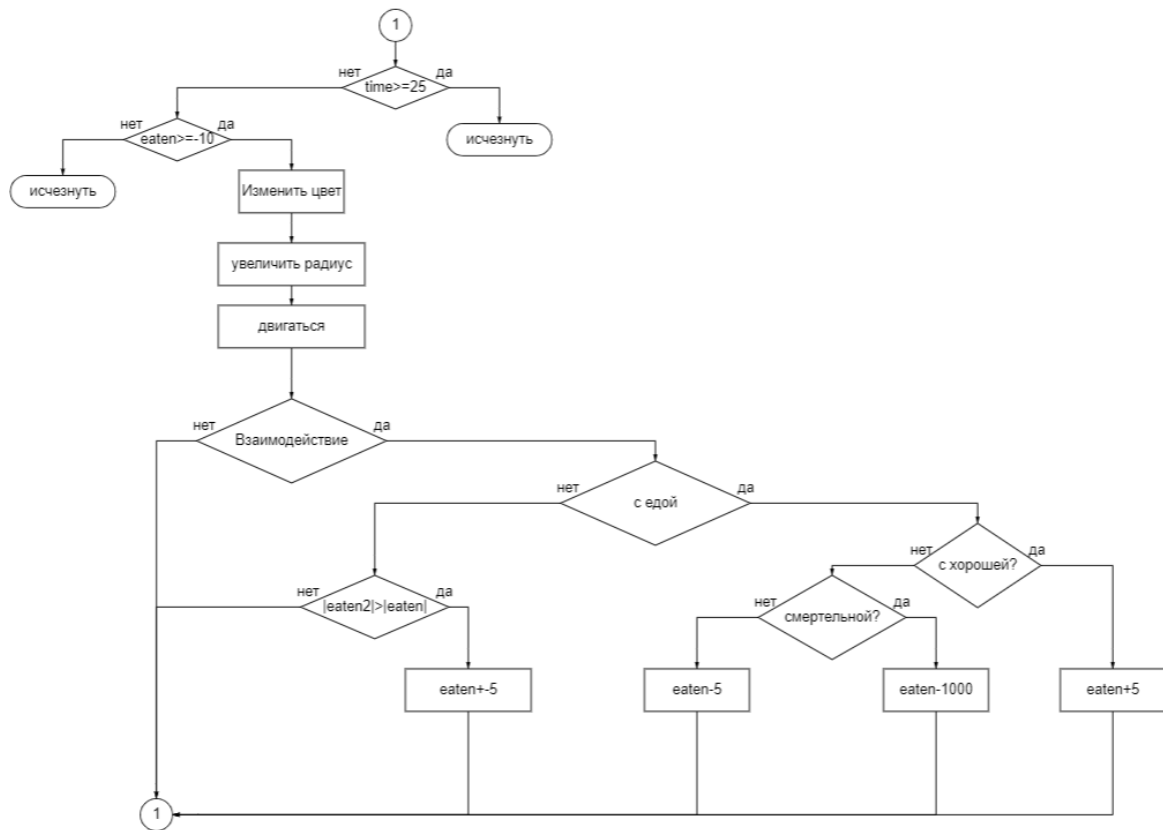


Рисунок 2.2 Диаграмма деятельности бактерии



Рисунок 2.3 Диаграмма деятельности обстоятельства

3. Описание программного продукта

3.1. Выбор средств реализации

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

3.2. Описание классов

Классы:

Bacteria – человек

Поля:

- x – координата по оси ОХ
- y - координата по оси ОУ
- $radius$ – радиус круга в пикселях
- vx – передвижение по ОХ
- vy – передвижение по ОУ
- $eaten$ – уровень развития
- $time$ – прожитое время на поле
- $isCollide$ – флаг столкновения

Методы:

- $initChild(x2,y2)$ – инициализация потомка
- $initpop()$ – инициализация популяции

goodE – положительные обстоятельства

Поля:

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

Методы:

- $initpop()$ – инициализация популяции

badE – отрицательные обстоятельства

Поля:

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

Методы:

- initpop () – инициализация популяции

deathE – отрицательные обстоятельства

Поля:

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

Методы:

- initpop () – инициализация популяции

Заключение

Была разработана программа, которая обладает следующим функционалом:

- Визуализация поведения бактерий, их взаимодействие с миром.


Возможны дальнейшие усовершенствования программы:

- Добавление концовки, при достижении определенного результата
- Добавить возможность выбирать свою скорость.
- Запуск несколько генераций.
- Добавить возможность выставление своих параметров
- Добавить возможность сформировать отчет по окончанию симуляции.

Приложение А. Снимки экранных форм пользовательского интерфейса

При запуске:

Модель естественного отбора



0 с.

Изначальное количество бактерий:

Максимальное количество бактерий:

Кол-во хорошей еды:

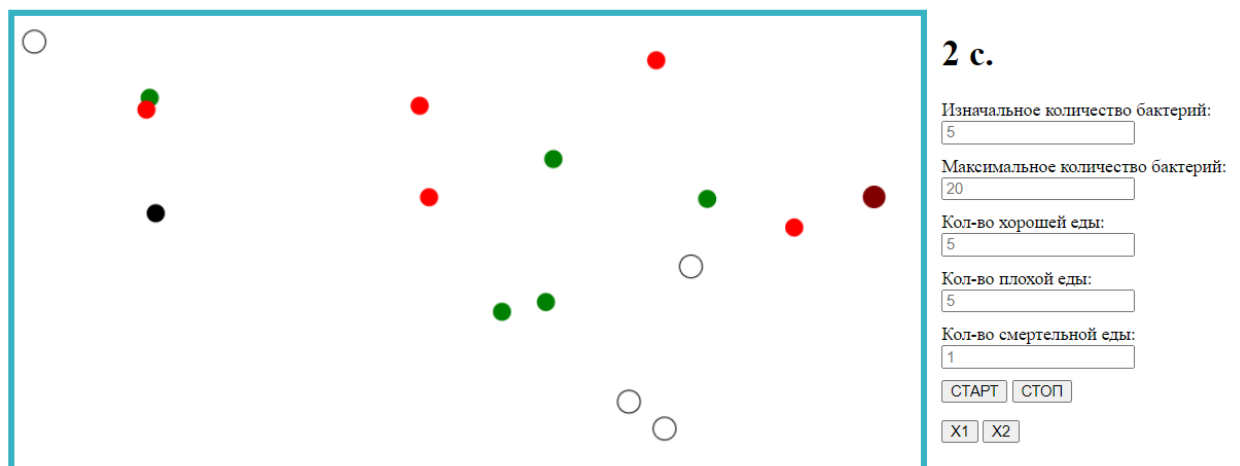
Кол-во плохой еды:

Кол-во смертельной еды:

Рисунок А. 1

При нажатии на кнопку «СТАРТ» запустится модель:

Модель естественного отбора



2 с.

Изначальное количество бактерий:

Максимальное количество бактерий:

Кол-во хорошей еды:

Кол-во плохой еды:

Кол-во смертельной еды:

Рисунок А. 2

Результат –развитие высокого уровня:

Модель естественного отбора

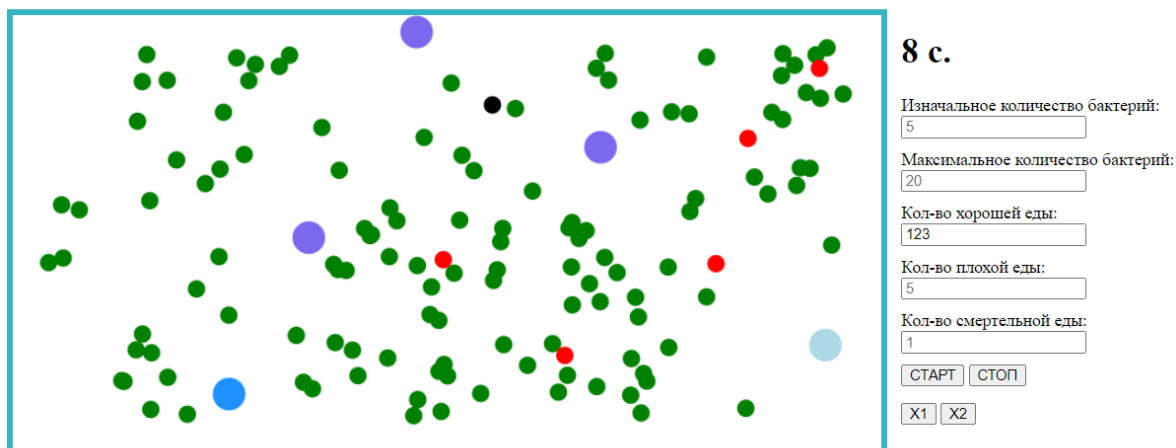


Рисунок А. 3

Результат – образование потомков:

Модель естественного отбора

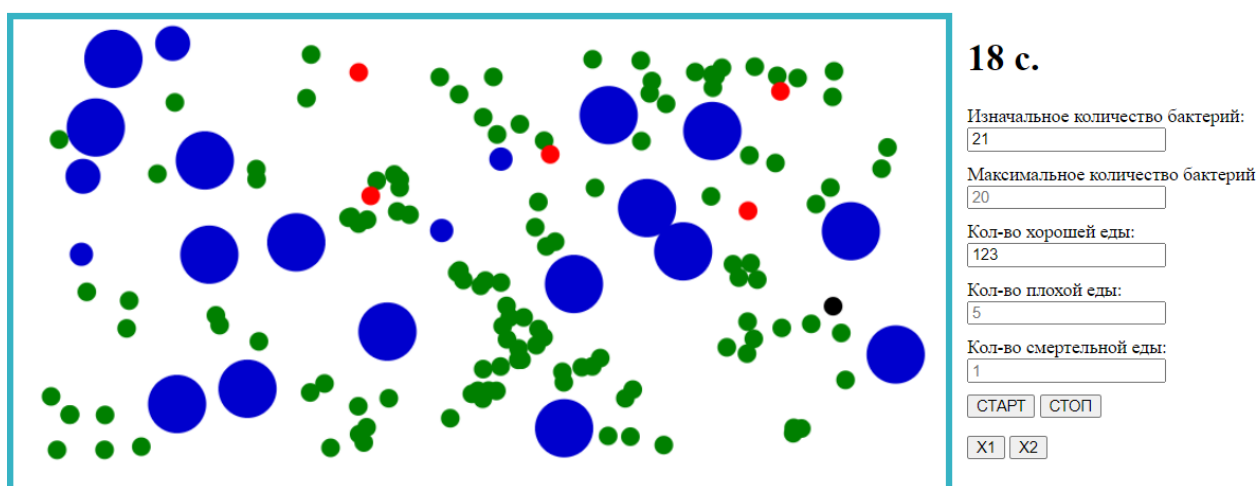


Рисунок А. 4

Результат –естественный отбор закончен:

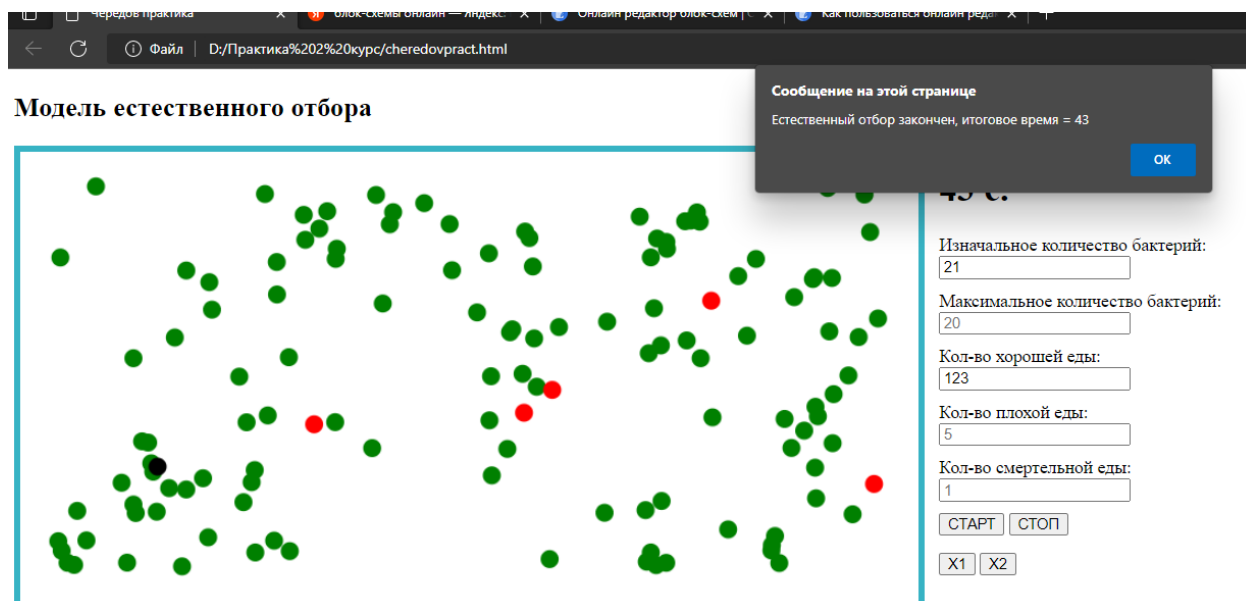


Рисунок А. 5

Приложение Б. Исходный код

```
<!DOCTYPE html>
<html>

<title>Чередов практика</title>

<h2>Модель естественного отбора</h2>
<style type="text/css">
    TD.leftcol {
        width: 825px;
        vertical-align: top;
    }
</style>
<body>
    <table width="100%" cellpadding="0" cellspacing="0">
    <tr>
        <td class="leftcol"><canvas id="canvas" width="800px" height="400px"
style="border: 6px solid rgb(55, 179, 196);"></canvas></td>

        <td valign="top">
            <h1><time>0 с.</time></h1>
            <div style="margin-top: 10px;"> </div>
            <div class="row">

                Изначальное количество бактерий:
                <div class="col-md-3">
                    <input id="fieldPopulation" type="number" placeholder="5">
                </div>

                <div style="margin-top: 10px;"></div>
                Максимальное количество бактерий:
                <div class="col-md-2">
                    <input id="fieldMAX" type="number" placeholder="20">
                </div>

                <div style="margin-top: 10px;"></div>
                Кол-во хорошей еды:
                <div class="col-md-3">
                    <input id="fieldGoodE" type="number" placeholder="5">
                </div>

                <div style="margin-top: 10px;"></div>
                Кол-во плохой еды:
                <div class="col-md-2">
                    <input id="fieldBadE" type="number" placeholder="5">
                </div>
                <div style="margin-top: 10px;"></div>
                Кол-во смертельной еды:
                <div class="col-md-2">
                    <input id="fieldDeathE" type="number" placeholder="1">
                </div>

                <div style="margin-top: 10px;"></div>
                <button id="strt">СТАРТ</button>
                <button id="stp">СТОП</button>
                <div style="margin-top: 15px;"></div>

                <button id="spd1">X1</button>

            </td>
    </tr>
    </table>
</body>
</html>
```

```

        <button id="spd2">X2</button>

    </div>
</div></td>
</tr>
</table>

</body>

<script type="text/javascript">
    var canvas, ctx, W, H;
    var h1 = document.getElementsByTagName('h1')[0];
    var sec = 0;
    var f = 1;
    var test = 1;
    var t;
    var popLimit = 20;
    var DefSpeed = 1;
    var DefRad = 10;
    var OldX=0, OldY = 0;
    //инициализация кнопок
    var start = document.getElementById('strt');
    var stop = document.getElementById('stp');
    var speed2 = document.getElementById('spd2');
    var speed1 = document.getElementById('spd1');
    //функция старта
    start.onclick = function(){
        f = 1;
        sec = 0;
        DefSpeed = 1;
        h1.textContent = sec + ' c.';

        populationCount =
parseInt(document.getElementById('fieldPopulation').value);
        if ( !populationCount ) populationCount = 5;

        popLimit = parseInt(document.getElementById('fieldMAX').value);
        if ( !popLimit ) popLimit = 20;

        GoodCount = parseInt(document.getElementById('fieldGoodE').value);
        if ( !GoodCount ) GoodCount = 5;

        badCount = parseInt(document.getElementById('fieldBadE').value);
        if ( !badCount ) badCount = 5;

        deathCount = parseInt(document.getElementById('fieldDeathE').value);
        if ( !deathCount ) deathCount = 1;

        population = [];
        popGood = [];
        popBad = [];
        popDeath = [];

        window.cancelAnimationFrame(null);
        delPopulation();
        new Bacteria().initpop();
        new badE().initpop();
        new goodE().initpop();
        new deathE().initpop();
    }

```

```

        if (test){
            timer();
            initDraw();
        }
        test = 0;
    }
    //Функция кнопки x2
    speed2.onclick = function(){
        DefSpeed = 2;
    }
    //Функция кнопки x1
    speed1.onclick = function(){
        DefSpeed = 1;
    }
    //Функция кнопки стоп
    stop.onclick = function() {
        f = 0;
        test = 1;
        clearTimeout(t);
        stopDraw();
    }

    window.onload = init;
    //создание класса бактерия
    class Bacteria {
        constructor(x,y, vx,vy, eaten){
            this.x = x;
            this.y = y;

            this.radius = DefRad;

            this.vx = vx;
            this.vy = vy;

            this.eaten = eaten;
            this.time = 0;

            this.isCollide = false;
        }
        initChild(eaten2,x2,y2) {

            let rnd = Math.random()*10+40;

            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 h = new Bacteria(xx, yy, xV, yV, (this.eaten + eaten2)/2);

        population.push(h);
    }
    initpop(){
        for (var i=0;i<populationCount;i++) {

            let rndX = Math.random()*(W-60) + 30;
            let rndY = Math.random()*(H-60) + 30;

            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 h = new Bacteria(rndX, rndY, xV, yV, 0);

            population.push(h);

        }
    }
}
//создание класса хорошей еда
class goodE {
    constructor(x,y){
        this.x = x;
        this.y = y;

        this.radius = 8;

        this.isCollide = false;
    }
    initpop(){
        for (var i=0;i<GoodCount;i++) {

            let rndX = Math.random()*(W-60) + 30;
            let rndY = Math.random()*(H-60) + 30;

            let t = new goodE(rndX, rndY);

            popGood.push(t);

        }
    }
}
//создание класса плохой еды
class badE {
    constructor(x,y){
        this.x = x;
        this.y = y;

        this.radius = 8;

        this.isCollide = false;
    }
    initpop(){
        for (var i=0;i<badCount;i++) {

```

```

        let rndX = Math.random()*(W-60) + 30;
        let rndY = Math.random()*(H-60) + 30;

        let b = new badE(rndX, rndY);

        popBad.push(b);
    }
}
//создание класса смертельной еды
class deathE {
    constructor(x,y){
        this.x = x;
        this.y = y;

        this.radius = 8;

        this.isCollide = false;
    }
    initpop(){
        for (var i=0;i<deathCount;i++) {

            let rndX = Math.random()*(W-60) + 30;
            let rndY = Math.random()*(H-60) + 30;

            let k = new deathE(rndX, rndY);

            popDeath.push(k);
        }
    }
}

//Инициализация окна
function init() {
    canvas = document.getElementById("canvas");
    ctx    = canvas.getContext('2d');
    W      = canvas.width;
    H      = canvas.height;
}

//Тик
function tick(){
    sec+=1;
    h1.textContent = sec + ' с.';
    for (var i=0;i<population.length;i++) {
        var h = population[i];
        h.time ++;
    }
    timer();
}
//Таймер
function timer() {
    t = setTimeout(tick, 1000 / DefSpeed);
}
//Удалить все бактерии
function delPopulation(){
    population.splice(0,population.length);
    popGood.splice(0,popGood.length);
    popBad.splice(0,popBad.length);
    popDeath.splice(0,popDeath.length);
}

```

```

//визуализация
function initDraw() {
    ctx.clearRect(0,0,W,H);
    detectCollisions();
    DetectGB();

    if (!population.length){
        alert('Естественный отбор закончен, итоговое время = '+sec);
        stop.onclick();
    }

    for (var j = 0; j < popGood.length; j++){
        var t = popGood[j];

        ctx.beginPath();
        ctx.arc(t.x,t.y,t.radius,0,Math.PI*2, 0);
        ctx.closePath();

        ctx.fillStyle = 'green';
        ctx.strokeStyle = 'black';
        ctx.fill();
    }

    for (var j = 0; j < popBad.length; j++){
        var b = popBad[j];

        ctx.beginPath();
        ctx.arc(b.x,b.y,b.radius,0,Math.PI*2, 0);
        ctx.closePath();

        ctx.fillStyle = 'red';
        ctx.strokeStyle = 'black';
        ctx.fill();
    }

    for (var j = 0; j < popDeath.length; j++){
        var k = popDeath[j];

        ctx.beginPath();
        ctx.arc(k.x,k.y,k.radius,0,Math.PI*2, 0);
        ctx.closePath();

        ctx.fillStyle = 'black';
        ctx.strokeStyle = 'black';
        ctx.fill();
    }

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

        h.x += DefSpeed * 0.6 * h.vx;
        h.y += DefSpeed * 0.6 * h.vy;

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

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

        ctx.beginPath();
    }
}

```

```

        ctx.arc(h.x,h.y,h.radius,0, Math.PI*2, 0);
        ctx.closePath();

        if (h.x - h.radius - 20 > 0 && h.x + h.radius + 20 < W && h.y -
h.radius - 20 > 0 && h.y + h.radius + 20 < H){
            if (h.time >= 5 && h.time < 10)
                h.radius = DefRad+5;

            if (h.time >= 10 && h.time < 15)
                h.radius = DefRad+10;

            if (h.time >= 15 && h.time < 20)
                h.radius = DefRad+15;

            if (h.time >= 20 && h.time < 25)
                h.radius = DefRad+20;
        }

        if (h.eaten <= -10){
            population.splice(i,1);
            i--;
            continue;
        }
        if (h.time >= 25){
            population.splice(i,1);
            i--;
            continue;
        }

        if ( h.eaten > -10 && h.eaten <= -2) {
            ctx.fillStyle = 'maroon';
            ctx.strokeStyle = 'black';
            ctx.fill();
        }
        if ( h.eaten > -1 && h.eaten <= 10) {
            ctx.fillStyle = 'white';
            ctx.strokeStyle = 'black';
            ctx.stroke();
        }
        if ( h.eaten > 10 && h.eaten <= 30) {
            ctx.fillStyle = 'LightBlue';
            ctx.strokeStyle = 'black';
            ctx.fill();
        }
        if ( h.eaten > 30 && h.eaten <= 50) {
            ctx.fillStyle = 'DodgerBlue';
            ctx.strokeStyle = 'black';
            ctx.fill();
        }
        if ( h.eaten > 50 && h.eaten <= 75) {
            ctx.fillStyle = 'MediumSlateBlue';
            ctx.strokeStyle = 'black';
            ctx.fill();
        }
        if ( h.eaten > 75) {
            ctx.fillStyle = 'MediumBlue';
            ctx.strokeStyle = 'black';
            ctx.fill();
        }
    }
    if (f)

```

```

        window.requestAnimationFrame(initDraw);

    }
    //Очистка поля
    function stopDraw(){
        ctx.clearRect(0,0,W,H);
    }

    //Столкновение с обстоятельствами
    function DetectGB(){
        var h, t, b;

        for (var i=0; i < population.length; i++){
            population[i].isCollide = false;
        }
        for (var i=0; i < popGood.length; i++){
            popGood[i].isCollide = false;
        }
        for (var i=0; i < popBad.length; i++){
            popBad[i].isCollide = false;
        }
        for (var i=0; i < popDeath.length; i++){
            popDeath[i].isCollide = false;
        }

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

            for ( var j=0;j<popGood.length;j++ ) {
                t = popGood[j];

                let dx = h.x-t.x;
                let dy = h.y-t.y;
                let distance = Math.sqrt(dx*dx+dy*dy);

                if ( distance < h.radius + t.radius ) {
                    h.eaten += 5;
                    t.x = Math.random()*(W-60) + 30;
                    t.y = Math.random()*(H-60) + 30;
                }
            }
            for ( var j=0;j<popBad.length;j++ ) {
                b = popBad[j];

                let bdx = h.x-b.x;
                let bdy = h.y-b.y;
                let bdistance = Math.sqrt(bdx*bdx+bdy*bdy);

                if ( bdistance < h.radius + b.radius ) {
                    h.eaten -= 5;
                    b.x = Math.random()*(W-60) + 30;
                    b.y = Math.random()*(H-60) + 30;
                }
            }
            for ( var j=0;j<popDeath.length;j++ ) {
                k = popDeath[j];

                let kdx = h.x-k.x;
                let kdy = h.y-k.y;
                let kdistance = Math.sqrt(kdx*kdx+kdy*kdy);
            }
        }
    }

```



```

        if ( kdistance < h.radius + k.radius ) {
            h.eaten -= 1000;
            k.x = Math.random()*(W-60) + 30;
            k.y = Math.random()*(H-60) + 30;
        }
    }
}
//Столкновение бактерий
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 srX = (h1.x+h2.x)/2;
                let srY = (h1.y+h2.y)/2;

                if (srX < OldX - 20 || srX > OldX + 20 || srY < OldY
- 20 || srY > OldY + 20){
                    //Передача опыта
                    let hh = h1.eaten;

                    if (h1.time >= 1 && h2.time >= 1){
                        if (h1.eaten * h2.eaten >= 0){
                            if(Math.abs(h2.eaten) >
Math.abs(h1.eaten))
                                if(Math.abs(h1.eaten) +
Math.abs(h2.eaten)/10 <= 100 && Math.abs(h1.eaten) + Math.abs(h2.eaten)/10 <=
Math.abs(h2.eaten))
                                    {
                                        h1.eaten +=5;
                                        h2.eaten -=5;
                                    }
                                else{
                                    if (Math.abs(h2.eaten) +
Math.abs(h1.eaten)/10 <= 100 && Math.abs(h2.eaten) + Math.abs(h1.eaten)/10 <=
Math.abs(h2.eaten))
                                        {
                                            h2.eaten += 5;
                                            h1.eaten =5;
                                        }
                                    }
                                }
                            }
                        else{
                            h1.eaten += 5;

```

```

        h2.eaten += 5;
    }
}

0 //Появление потомка
let rndRod = Math.random()*100;

    if (((h1.time >= 11 && h1.time < 15 && h2.time >= 11
&& h2.time < 15) || (h1.time >= 16 && h1.time < 20 && h2.time >= 16 && h2.time < 20))
&& population.length <= popLimit && rndRod <= 90){
        h1.initChild(h2.eaten,h2.x,h2.y);
    }
}

OldX = srX;
OldY = srY;

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;

h1.vx -= speedV*collisionNormal.x;
h1.vy -= speedV*collisionNormal.y;

h2.vx += speedV*collisionNormal.x;
h2.vy += speedV*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;
    }
}
}
}
}

}

</script>
</html>

```