|  |  |
| --- | --- |
| **Российский университет транспорта (МИИТ)**  **Институт транспортной техники и систем управления**  **Кафедра «Управление и защита информации»** | |
| **Задание №7**  **по дисциплине «Web-программирование»** | |
|  | Выполнил:  Студент группы ТКИ-542  Пономарев А. Д.  Дроздов А. Д.  Проверил:  Доцент кафедры УиЗИ, к.т.н.  Сафронов А. И. |
| Москва 2024 | |

Оглавление

[1. Цель работы 3](#_Toc184126063)

[2. ФОРМУЛИРОВАКА ЗАДАЧИ 3](#_Toc184126064)

[3. таблица соответсвия переменных 4](#_Toc184126065)

[4. Спецификация оборудования 4](#_Toc184126066)

[5. Web-страница 4](#_Toc184126067)

[5.1. Базовый код (до оптимизации) 4](#_Toc184126068)

[5.1.1 Результат отображения (до оптимизации) 6](#_Toc184126069)

[5.2. SPWA (код после оптимизации) 6](#_Toc184126070)

[5.2.1 Результат отображения SPWA 11](#_Toc184126071)

[5.2.2 Диаграмма построения 12](#_Toc184126072)

[5.2.3 Обоснование размеров 12](#_Toc184126073)

[5.2.4 Сети Петри 13](#_Toc184126074)

[5.3. LSPWA (код после оптимизации) 14](#_Toc184126075)

[5.3.1 SierpinskiTriangle.vue 14](#_Toc184126076)

[5.3.2 server.js 17](#_Toc184126077)

[5.3.3 Результат отображения LSPWA 18](#_Toc184126078)

[5.3.4 Диаграмма построения 20](#_Toc184126079)

[5.3.5 Обоснование размеров 20](#_Toc184126080)

[5.3.6 Сети Петри 21](#_Toc184126081)

[5.4. Анализ быстродействия программного обеспечения 22](#_Toc184126082)

[6. вывод по работе 25](#_Toc184126083)

1. Цель работы

Целью работы является разработка и анализ быстродействия веб-приложения, реализующего фрактальную графику с использованием фреймворка Vue.js и технологий SVG или Canvas. Работа включает в себя модификацию базового кода, анализ производительности, а также сравнение быстродействия различных реализаций (SPWA и LSPWA) на разных устройствах.

1. ФОРМУЛИРОВАКА ЗАДАЧИ

Необходимо:

1. Устранить избыточность в базовом коде.
2. Модифицировать серверную и клиентскую части веб-приложения согласно индивидуальному варианту.
3. Реализовать Single Page Web Application (SPWA) и Lazy Single Page Web Application (LSPWA), сравнив их быстродействие на разных устройствах.
4. Восстановить математическую форму записи для расчета фрактала.
5. Построить столбчатые диаграммы влияния изменяемых параметров на скорость расчета и визуализации фрактальной графики.
6. Предусмотреть возможность вывода областей построения фрактала при значениях R > 1.
7. Обосновать выбор ширины и высоты изображения в соответствии с заданным количеством мегапикселей.
8. таблица соответсвия переменных

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ***№*** | ***N*** | ***R, шт.*** | ***S, МПкс*** | ***B*** | ***F*** | ***V*** |
| **8** | Треугольник Серпинского | *2* | *1,5; 2,5* | Ввести изменение точки начала отсчёта для области построения фрактала на левую нижнюю – (0;0) считать там (выполнить только для первой из двух областей построения). Изменения подразумевают сохранение исходных строчек кода, размещаемых в недостижимой компилятором и вычислительным процессом алгоритмической ветви и организации новой, основной ветви вычислений | Предоставить пользователю возможность изменения координат вершин опорного треугольника Серпинского (*А, В, С*) | **05** |

1. Спецификация оборудования

* **Стационарный компьютер (Пономарев)**:
  + Процессор: Intel(R) Core(TM) i7-9700KF CPU @ 3.60GHz
  + Оперативная память: 16 ГБ
  + Видеокарта: NVIDIA GeForce RTX 2060
  + Операционная система: Windows 10
* **Ноутбук (Дроздов)**:
  + Процессор: Intel(R) Celeron(R) N4000 CPU @ 1.10GHz
  + Оперативная память: 8 ГБ
  + Видеокарта: Intel Graphics
  + Операционная система: Kali Linux

1. Web-страница
   1. Базовый код (до оптимизации)

<html>

<title>Треугольник Серпинского</title>

<body>

<script src="Vue.js"></script>

<div id="trngl">

<br>

<svg :width="w" :height="h">

<rect v-for="pixel in pixels" fill="#000000" fill-opacity="1" :x="pixel.x" :y="pixel.y" width="1" height="1"/>

</svg>

</div>

<script>

var Triang = new Vue({

el: "#trngl",

data:{

w: 500,

h: 500,

A: [0, 0],

B: [0.5, 0.86603],

C: [1, 0],

pixels: [],

},

computed:{

ABC: function(){

return [this.A, this.B, this.C];

}

},

methods: {

drawFractal: function() {

this.pixels = [];

let p = [];

let px = new Object();

let dx = (Math.random() \* 100) / 100;

let dy = (Math.random() \* (dx <= 0.5 ? dx : (1 - dx)) \* Math.tan(Math.PI / 3) \* 100) / 100;

px.x = dx \* this.w;

px.y = dy \* this.h;

this.pixels.push(px);

p.push([dx, dy]);

for (let i = 0; i < 100000; i++) {

let U = this.ABC[Math.floor(Math.random() \* this.ABC.length)];

let G = p[Math.floor(Math.random() \* p.length)];

let d00 = Math.abs(U[0] - G[0]) / 2;

let d11 = Math.abs(U[1] - G[1]) / 2;

let dxx = U[0] > G[0] ? G[0] + d00 : G[0] - d00;

let dyy = U[1] > G[1] ? G[1] + d11 : G[1] - d11;

let px = new Object();

px.x = dxx \* this.w;

px.y = dyy \* this.h;

this.pixels.push(px);

p.push([dxx, dyy]);

}

let now = new Date();

console.log("Завершено построение: " + now + " мс: " + now.getMilliseconds());

},

},

created: function() {

let now = new Date();

console.log("Начато построение: " + now + " мс: " + now.getMilliseconds());

this.drawFractal();

},

mounted: function() {

let now = new Date();

console.log("Отображено на странице: " + now + " мс: " + now.getMilliseconds());

},

});

</script>

<style type="text/css">

svg { border: 1px solid #000000; }

input { width:60px; }

</style>

</body>

</html>

Результат отображения (до оптимизации)

|  |
| --- |
|  |
| 1. – Результат отображения (до оптимизации) |

* 1. SPWA (код после оптимизации)

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Треугольник Серпинского</title>

<style>

body {

font-family: Avenir, Helvetica, Arial, sans-serif;

text-align: center;

color: #2c3e50;

margin-top: 60px;

}

svg {

border: 1px solid #000000;

}

input {

width: 60px;

}

#time {

font-size: 18px;

margin-top: 20px;

}

</style>

</head>

<body>

<h1>Треугольник Серпинского</h1>

<div id="trngl">

<br>

<svg :width="w" :height="h">

<rect v-for="pixel in pixels"

fill="#000000"

fill-opacity="1"

:x="pixel.x"

:y="pixel.y"

width="1"

height="1"/>

</svg>

<br>

<label>Вершина A (x, y):</label>

<input type="number" v-model="A[0]">

<input type="number" v-model="A[1]">

<br>

<label>Вершина B (x, y):</label>

<input type="number" v-model="B[0]">

<input type="number" v-model="B[1]">

<br>

<label>Вершина C (x, y):</label>

<input type="number" v-model="C[0]">

<input type="number" v-model="C[1]">

<br>

<button @click="drawFractal">Перерисовать фрактал</button>

<div id="time"></div>

</div>

<canvas id="performanceChart"></canvas>

<script src="https://cdn.jsdelivr.net/npm/vue@2"></script>

<script src="https://cdn.jsdelivr.net/npm/chart.js"></script>

<script>

var Triang = new Vue({

el: "#trngl",

data: {

w: 500,

h: 500,

//Вершина треугольника

A: [0, 0],

B: [0.5, 0.86603],//Math.Sqrt(3)/2 = 0.86603

C: [1, 0],

pixels: [],

startPoint: [0, 0],

chart: null,

},

//Реактивно вычисляемые функции

computed: {

ABC: function() {

return [this.A, this.B, this.C];

}

},

//Раздел методов фреймворка

methods: {

drawFractal: function() {

//коллекция точек

this.pixels = [];

let p = [];

//Случайная точка внутри треугольника

let dx = Math.random();

let dy = dx <= 0.5 ? Math.random() \* dx \* Math.tan(Math.PI / 3) : Math.random() \* (1 - dx) \* Math.tan(Math.PI / 3);

this.pixels.push({ x: dx \* this.w, y: dy \* this.h });

p.push([dx, dy]);

//Числовой подсчет времени работы

const startTime = performance.now();

const startDate = new Date().toLocaleTimeString();

for (let i = 0; i < 100000; i++) {

let U = this.ABC[Math.floor(Math.random() \* 3)];

let G = p[Math.floor(Math.random() \* p.length)];

let dxx = (U[0] + G[0]) / 2;

let dyy = (U[1] + G[1]) / 2;

this.pixels.push({ x: dxx \* this.w, y: dyy \* this.h });

p.push([dxx, dyy]);

}

const endTime = performance.now();

const endDate = new Date().toLocaleTimeString();

const elapsedTime = endTime - startTime;

document.getElementById('time').innerHTML = `

<p>Время начала построения: ${startDate}</p>

<p>Время окончания построения: ${endDate}</p>

<p>Время построения: ${elapsedTime.toFixed(2)} мс</p>

`;

this.updateChart([1000000], [elapsedTime.toFixed(2)]);

},

updateChart: function(labels, data) {

if (!this.chart) {

// Инициализация диаграммы, если она еще не создана

var ctx = document.getElementById('performanceChart').getContext('2d');

this.chart = new Chart(ctx, {

type: 'bar',

data: {

labels: labels,

datasets: [{

label: 'Время выполнения (мс)',

data: data,

backgroundColor: 'rgba(75, 192, 192, 0.2)',

borderColor: 'rgba(75, 192, 192, 1)',

borderWidth: 1

}]

},

options: {

scales: {

y: {

beginAtZero: true

}

}

}

});

} else {

// Обновление данных диаграммы

this.chart.data.labels = labels;

this.chart.data.datasets[0].data = data;

this.chart.update();

}

}

},

created: function() {

let now = new Date();

console.log("Начато построение: " + now + " мс: " + now.getMilliseconds());

this.drawFractal();

},

mounted: function() {

let now = new Date();

console.log("Отображено на странице: " + now + " мс: " + now.getMilliseconds());

},

});

</script>

</body>

</html>

Результат отображения SPWA

|  |
| --- |
|  |
| 1. – Результат отображения |

Диаграмма построения

|  |
| --- |
|  |
| 1. – Диаграмма |

Обоснование размеров

=

Сети Петри

|  |
| --- |
|  |
| 1. – Сети Петри |

* 1. LSPWA (код после оптимизации)

SierpinskiTriangle.vue

<template>

<div>

<h1>Треугольник Серпинского</h1>

<div>

<br>

<svg :width="w" :height="h">

<rect v-for="(pixel, index) in pixels"

:key="index"

fill="#000000"

fill-opacity="1"

:x="pixel.x"

:y="pixel.y"

width="1"

height="1"/>

</svg>

<br>

<label>Вершина A (x, y):</label>

<input type="number" v-model="A[0]">

<input type="number" v-model="A[1]">

<br>

<label>Вершина B (x, y):</label>

<input type="number" v-model="B[0]">

<input type="number" v-model="B[1]">

<br>

<label>Вершина C (x, y):</label>

<input type="number" v-model="C[0]">

<input type="number" v-model="C[1]">

<br>

<button @click="drawFractal">Перерисовать фрактал</button>

<div id="time"></div>

</div>

<canvas id="performanceChart"></canvas>

</div>

</template>

<script>

import { Chart, registerables } from 'chart.js';

Chart.register(...registerables);

export default {

data() {

return {

w: 500,

h: 500,

A: [0, 0],

B: [0.5, 0.86603], // Math.Sqrt(3)/2 = 0.86603

C: [1, 0],

pixels: [],

chart: null,

};

},

computed: {

ABC() {

return [this.A, this.B, this.C];

}

},

methods: {

async drawFractal() {

const response = await fetch('http://localhost:3000/api/drawFractal', {

method: 'POST',

headers: {

'Content-Type': 'application/json'

},

body: JSON.stringify({

A: this.A,

B: this.B,

C: this.C

})

});

const data = await response.json();

this.pixels = data.pixels;

document.getElementById('time').innerHTML = `

<p>Время начала построения: ${data.startDate}</p>

<p>Время окончания построения: ${data.endDate}</p>

<p>Время построения: ${data.elapsedTime} мс</p>

`;

this.updateChart([1000000], [data.elapsedTime]);

},

updateChart(labels, data) {

if (!this.chart) {

var canvasElement = document.getElementById('performanceChart');

console.log('Canvas element:', canvasElement); // Добавьте этот лог

var ctx = canvasElement.getContext('2d');

console.log('Canvas context:', ctx); // Добавьте этот лог

this.chart = new Chart(ctx, {

type: 'bar',

data: {

labels: labels,

datasets: [{

label: 'Время выполнения (мс)',

data: data,

backgroundColor: 'rgba(75, 192, 192, 0.2)',

borderColor: 'rgba(75, 192, 192, 1)',

borderWidth: 1

}]

},

options: {

scales: {

y: {

beginAtZero: true

}

}

}

});

} else {

this.chart.data.labels = labels;

this.chart.data.datasets[0].data = data;

this.chart.update();

}

}

},

created() {

let now = new Date();

console.log("Начато построение: " + now + " мс: " + now.getMilliseconds());

this.drawFractal();

},

mounted() {

let now = new Date();

console.log("Отображено на странице: " + now + " мс: " + now.getMilliseconds());

},

};

</script>

<style scoped>

body {

font-family: Avenir, Helvetica, Arial, sans-serif;

text-align: center;

color: #2c3e50;

margin-top: 60px;

}

svg {

border: 1px solid #000000;

}

input {

width: 60px;

}

#time {

font-size: 18px;

margin-top: 20px;

}

</style>

server.js

const express = require('express');

const bodyParser = require('body-parser');

const cors = require('cors');

const app = express();

const port = 3000;

app.use(cors());

app.use(bodyParser.json());

app.post('/api/drawFractal', (req, res) => {

const { A, B, C } = req.body;

// Логика для вычисления треугольника Серпинского

const pixels = [];

let p = [];

let dx = Math.random();

let dy = dx <= 0.5 ? Math.random() \* dx \* Math.tan(Math.PI / 3) : Math.random() \* (1 - dx) \* Math.tan(Math.PI / 3);

pixels.push({ x: dx \* 500, y: dy \* 500 });

p.push([dx, dy]);

const startTime = performance.now();

const startDate = new Date().toLocaleTimeString();

for (let i = 0; i < 100000; i++) {

let U = [A, B, C][Math.floor(Math.random() \* 3)];

let G = p[Math.floor(Math.random() \* p.length)];

let dxx = (U[0] + G[0]) / 2;

let dyy = (U[1] + G[1]) / 2;

pixels.push({ x: dxx \* 500, y: dyy \* 500 });

p.push([dxx, dyy]);

}

const endTime = performance.now();

const endDate = new Date().toLocaleTimeString();

const elapsedTime = endTime - startTime;

res.json({

pixels,

startDate,

endDate,

elapsedTime: elapsedTime.toFixed(2)

});

});

app.listen(port, () => {

console.log(`Server is running on port ${port}`);});

Результат отображения LSPWA

|  |
| --- |
|  |
| 1. – Запуск |
|  |
|  |
| 1. – Результат отображения |

Диаграмма построения

|  |
| --- |
|  |
| 1. – Диаграмма |

Обоснование размеров

=

Сети Петри

|  |
| --- |
|  |
| 1. – Сети Петри |

* 1. Анализ быстродействия программного обеспечения

|  |
| --- |
|  |
| 1. – Результат запуска SPWA на компьютере Пономарева |
|  |
| 1. – Результат запуска SPWA на ноутбуке Дроздова |
|  |
| 1. – Результат запуска LSPWA на компьютере Пономарева |
|  |
| 1. – Результат запуска LSPWA на ноутбуке Дроздова |

На основе приведенных результатах, можно сделать вывод, что на результативность влияет, как техническая составляющая оборудования, так и различия между SPWA и LSPWA. LSPWA обрабатывает быстрее чем SPWA.

1. вывод по работе

В результате выполнения задания было разработано web-приложение для анализа быстродействия формирования фрактального изображения. Были реализованы SPWA и LSPWA, проведено сравнение их производительности на различных устройствах. Выявлено, что LSPWA демонстрирует лучшую производительность при больших объемах данных. Рекомендуется использовать LSPWA для приложений с большим объемом данных и SPWA для приложений с небольшим объемом данных.