|  |  |
| --- | --- |
| **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  Волочинский И.О.  Ефремов Д.Ю.  Проверил:  Доцент кафедры УиЗИ, к.т.н.  Сафронов А. И. |
| Москва 2024 | |

Оглавление

[1. Цель работы 3](#_gjdgxs)

[2. ФОРМУЛИРОВАКА ЗАДАЧИ 3](#_1fob9te)

[3. таблица соответсвия переменных 4](#_3znysh7)

[4. Спецификация оборудования 4](#_2et92p0)

[5. Web-страница 4](#_tyjcwt)

[5.1. Базовый код (до оптимизации) 4](#_3dy6vkm)

[5.1.1](#_1t3h5sf) Результат отображения (до оптимизации) 6

[5.2. SPWA (код после оптимизации) 6](#_4d34og8)

[5.2.1](#_2s8eyo1) Результат отображения SPWA 11

[5.2.2](#_17dp8vu) Диаграмма построения 12

[5.2.3](#_3rdcrjn) Обоснование размеров 12

[5.2.4](#_26in1rg) Сети Петри 13

[5.3. LSPWA (код после оптимизации) 14](#_lnxbz9)

[5.3.1](#_35nkun2) SierpinskiTriangle.vue 14

[5.3.2](#_1ksv4uv) server.js 17

[5.3.3](#_44sinio) Результат отображения LSPWA 18

[5.3.4](#_2jxsxqh) Диаграмма построения 20

[5.3.5](#_z337ya) Обоснование размеров 20

[5.3.6](#_3j2qqm3) Сети Петри 21

[5.4. Анализ быстродействия программного обеспечения 22](#_1y810tw)

[6. вывод по работе 25](#_2xcytpi)

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*** |
| **17** | Ковер Серпинского | *1* | *1,0* | Предусмотреть полный отказ от использования тернарного оператора в коде там, где это возможно реализовать | Управлять количеством выводимых в область построения точек ковра Серпинского, начиная с одной указанной точки завершая второй указанной точкой | **15** |

1. **Спецификация оборудования**

* **Стационарный компьютер (Волочинский)**:
  + Процессор: Intel(R) Core(TM) i7-13000KF CPU @ 3.60GHz
  + Оперативная память: 32 ГБ
  + Видеокарта: NVIDIA GeForce RTX 2080
  + Операционная система: Windows 10
* **Ноутбук (Ефремов)**:
  + Процессор: Apple M3 Pro @ 4.05GHz
  + Оперативная память: 18 ГБ
  + Видеокарта: Apple M4 GPU
  + Операционная система: MacOS Sonoma 14.1

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

<html>

<title>**Ковёр Серпинского**</title>

<body>

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

<div id**="crpt"**>

<br>

<svg

:width**="hor"**

:height**="ver"**>

<rect v-for**="el in els"**

fill**="#000000"**

fill-opacity**="1"**

:x**="el.x"**

:y**="el.y"**

width**="1"**

height**="1"**/>

</svg>

</div>

<script>

var *Carpet* = new Vue({

el: "#crpt",

data: {

hor: 500,

ver: 500,

A: [0, 0],

B: [0, 1],

C: [1, 1],

D: [1, 0],

els: [],

},

computed: {

ABCD: function() {

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

}

},

methods: {

drawCarpet: function() {

this.els = [];

let e = [];

let p = new Object();

p.x = this.A[0] \* this.hor;

p.y = this.A[1] \* this.ver;

this.els.push(p);

e.push([this.A[0], this.A[1]]);

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

let U = this.ABCD[*Math*.floor(*Math*.random() \*

this.ABCD.length)];

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

let dx = *Math*.abs(U[0] - G[0]) / 1.7;

let dy = *Math*.abs(U[1] - G[1]) / 1.7;

let nx = U[0] > G[0] ? G[0] + dx : G[0] - dx;

let ny = U[1] > G[1] ? G[1] + dy : G[1] - dy;

let p = new Object();

p.x = nx \* this.hor;

p.y = ny \* this.ver;

this.els.push(p);

e.push([nx, ny]);

}

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.drawCarpet();

},

mounted: function() {

let now = new *Date*();

*console*.log("Отображено на странице: "

+ now + " мс: " + now.getMilliseconds());

},

});

</script>

<style type**="text/css"**>

svg { *border*: 1**px solid** #000000; }

input { *width*:60**px**; }

</style>

</body>

</html>

* + 1. **Результат отображения (до оптимизации)**

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

* 1. **SPWA (код после оптимизации)**

<html>

<title>**Ковër Серпинского**</title>

<body>

<script src**="https://cdnjs.cloudflare.com/ajax/libs/vue/1.0.18/vue.min.js"**></script>

<div id**="crpt"**>

<br />

<input type**="number"** v-model.number**="pointStart"** placeholder**="Начало"** />

<input type**="number"** v-model.number**="pointEnd"** placeholder**="Конец"** />

<button @click**="drawCarpet"**>**Построить**</button>

<svg :width**="hor"** :height**="ver"**>

<rect

v-for**="el in els"**

:key**="el.id"**

fill**="#000000"**

fill-opacity**="1"**

:x**="el.x"**

:y**="el.y"**

width**="1"**

height**="1"**

/>

</svg>

</div>

<script>

var *Carpet* = new Vue({

el: "#crpt",

data: {

hor: 500,

ver: 500,

A: [0, 0],

B: [0, 1],

C: [1, 1],

D: [1, 0],

els: [],

pointStart: 0,

pointEnd: 10000,

},

computed: {

ABCD: function () {

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

},

},

methods: {

drawCarpet: function () {

this.els = [];

let e = [];

let p = { x: this.A[0] \* this.hor, y: this.A[1] \* this.ver };

this.els.push(p);

e.push([this.A[0], this.A[1]]);

let totalPoints = this.pointEnd - this.pointStart;

for (let i = this.pointStart; i < this.pointEnd; i++) {

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

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

let dx = *Math*.abs(U[0] - G[0]) / 1.7;

let dy = *Math*.abs(U[1] - G[1]) / 1.7;

let nx = G[0];

let ny = G[1];

if (U[0] > G[0]) {

nx += dx;

} else {

nx -= dx;

}

if (U[1] > G[1]) {

ny += dy;

} else {

ny -= dy;

}

let p = { x: nx \* this.hor, y: ny \* this.ver };

this.els.push(p);

e.push([nx, ny]);

}

let now = new *Date*();

*console*.log(

"Завершено построение: " +

now +

" мс: " +

now.getMilliseconds()

);

},

},

mounted: function () {

let now = new *Date*();

*console*.log(

"Отображено на странице: " +

now +

" мс: " +

now.getMilliseconds()

);

},

});

</script>

<style type**="text/css"**>

svg {

*border*: 1**px solid** #000000;

}

input {

*width*: 60**px**;

}

</style>

</body>

</html>

* + 1. **Результат отображения SPWA**

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

* + 1. **Диаграмма построения**

|  |
| --- |
|  |
| 1. – Диаграмма |

* + 1. **Обоснование размеров**

=

* + 1. **Сети Петри**

|  |
| --- |
|  |
| 1. – Сети Петри |

* 1. **LSPWA (код после оптимизации)**
     1. **SierpinskiPalace.vue**

<template>

<div>

<h1>**Ковёр 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>**Начальная точка:**</label>

<input type**="number"** v-model**="**pointStart**"** />

<label>**Конечная точка:**</label>

<input type**="number"** v-model**="**pointEnd**"** />

<br />

<button @click**="**drawCarpet**"**>**Перерисовать ковёр**</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, 1],

C: [1, 1],

D: [1, 0],

pointStart: 0,

pointEnd: 100000,

pixels: [],

chart: null,

};

},

methods: {

async drawCarpet() {

const response = await fetch("http://localhost:3000/api/drawCarpet", {

method: "POST",

headers: {

"Content-Type": "application/json",

},

body: *JSON*.stringify({

A: [0, 0],

B: [0, 1],

C: [1, 1],

D: [1, 0],

pointStart: this.pointStart,

pointEnd: this.pointEnd,

}),

});

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(["Точки"], [data.elapsedTime]);

},

updateChart(*labels*, *data*) {

if (!this.chart) {

const 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 {

this.chart.data.labels = *labels*;

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

this.chart.update();

}

},

},

};

</script>

<style scoped>

body {

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

*text-align*: **center**;

*color*: #2c3e50;

*margin-top*: 60**px**;

}

svg {

*border*: 1**px solid** #000000;

}

input {

*width*: 60**px**;

*margin*: 5**px**;

}

#time {

*font-size*: 18**px**;

*margin-top*: 20**px**;

}

</style>

* + 1. **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/drawCarpet', (req, res) => {

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

// Проверка входных данных

if (!A || !B || !C || !D || pointStart === undefined || pointEnd === undefined) {

return res.status(400).json({ error: "Некорректные входные данные" });

}

const pixels = [];

const e = []; // Хранилище координат

let startTime = performance.now();

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

// Добавляем первую точку

let p = { x: A[0] \* 500, y: A[1] \* 500 };

pixels.push(p);

e.push([A[0], A[1]]);

const ABCD = [A, B, C, D];

for (let i = pointStart; i < pointEnd; i++) {

const U = ABCD[Math.floor(Math.random() \* ABCD.length)];

const G = e[Math.floor(Math.random() \* e.length)];

let dx = Math.abs(U[0] - G[0]) / 1.7;

let dy = Math.abs(U[1] - G[1]) / 1.7;

let nx = G[0];

let ny = G[1];

if (U[0] > G[0]) {

nx += dx;

} else {

nx -= dx;

}

if (U[1] > G[1]) {

ny += dy;

} else {

ny -= dy;

}

let newPoint = { x: nx \* 500, y: ny \* 500 };

pixels.push(newPoint);

e.push([nx, ny]);

}

const endTime = performance.now();

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

const elapsedTime = (endTime - startTime).toFixed(2);

res.json({

pixels,

startDate,

endDate,

elapsedTime

});

});

app.listen(port, () => {

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

});

* + 1. **Результат отображения LSPWA**

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

* + 1. **Диаграмма построения**

|  |
| --- |
|  |
| 1. – Диаграмма |

* + 1. **Сети Петри**

|  |
| --- |
|  |
| 1. – Сети Петри |

* 1. **Анализ быстродействия программного обеспечения**

|  |
| --- |
|  |
| 1. – Результат запуска SPWA на компьютере Волочинского |
|  |
| 1. – Результат запуска SPWA на ноутбуке Ефремова |
|  |
| 1. – Результат запуска LSPWA на компьютере Волочинского |
|  |
| 1. – Результат запуска LSPWA на ноутбуке Ефремова |

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.