|  |  |
| --- | --- |
| **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  Круглов В.А.  Зинченко Б.А.  Проверил:  Доцент кафедры УиЗИ, к.т.н.  Сафронов А. И. |
| Москва 2024 | |

Оглавление

[1. Цель работы 3](#_Toc184657888)

[2. ФОРМУЛИРОВАКА ЗАДАЧИ 3](#_Toc184657889)

[3. таблица соответсвия переменных 4](#_Toc184657890)

[4. Спецификация оборудования 4](#_Toc184657891)

[5. Web-страница 4](#_Toc184657892)

[5.1. Базовый код (до оптимизации) 4](#_Toc184657893)

[5.1.1 Результат отображения (до оптимизации) 7](#_Toc184657894)

[5.2. SPWA (код после оптимизации) 7](#_Toc184657895)

[5.2.1 Результат отображения SPWA 13](#_Toc184657896)

[5.2.2 Диаграмма построения 13](#_Toc184657897)

[5.2.3 Обоснование размеров 13](#_Toc184657898)

[5.2.4 Сети Петри 14](#_Toc184657899)

[5.3. LSPWA (код после оптимизации) 15](#_Toc184657900)

[5.3.1 SierpinskiTriangle.vue 15](#_Toc184657901)

[5.3.2 App.vue 20](#_Toc184657902)

[5.3.3 server.js 20](#_Toc184657903)

[5.3.4 Результат отображения LSPWA 22](#_Toc184657904)

[5.3.5 Диаграмма построения 23](#_Toc184657905)

[5.3.6 Обоснование размеров 23](#_Toc184657906)

[5.3.7 Сети Петри 24](#_Toc184657907)

[5.4. Анализ быстродействия программного обеспечения 25](#_Toc184657908)

[6. вывод по работе 28](#_Toc184657909)

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*** |
| **24** | *N*-угольник Серпинского | *2* | *0,6; 0,9* | Менять на новый, выставляемый случайным образом, цвет очередной точки построения  *N*-угольника Серпинского, если расстояние до этой точки от предыдущей составляет более половины длины диагонали области построения фрактального изображения. | Предоставить возможность управления всеми возможными прямыми (не косвенными) параметрами *N*-угольника Серпинского | **13** |

1. Спецификация оборудования

* **Ноутбук 1:**
  + Процессор: AMD Ryzen 7840H CPU @ 3,8 GHz
  + Оперативная память: 32 ГБ
  + Видеокарта: AMD Radeon 780M
  + Операционная система: Windows 10
* **Ноутбук 2**:
  + Процессор: Apple M2 Pro
  + Оперативная память: 16 ГБ
  + Видеокарта: GPU 16-Core
  + Операционная система: macOS Sonoma v.14.4

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

<html>

<title>N-угольник Серпинского</title>

<body>

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

<div id="nnn">

<br>

<svg

:width="X"

:height="Y">

<rect v-for="r in rs"

fill="#000000"

fill-opacity="1"

:x="r.x"

:y="r.y"

width="1"

height="1"/>

</svg>

</div>

<script>

var myFigure = new Vue({

el: "#nnn",

data: {

X: 500,

Y: 500,

d: 1.6,

N: 5,

n: 20000,

R: 5,

sX: 10,

sY: 10,

rs: [],

},

methods: {

drawN:function() {

this.rs = [];

let p = [];

let s = [];

for (let i = 0; i < this.N; i++) {

let a = 2 \* Math.PI \* i / this.N;

s.push([this.R \* Math.cos(a)

+ this.sX, this.R \* Math.sin(a) + this.sY])

}

let r = new Object();

r.x = s[0][0] \* (this.X / 20);

r.y = s[0][1] \* (this.Y / 20);

this.rs.push(r);

p.push([s[0][0], s[0][1]]);

for (let i = 0; i < this.n; i++){

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

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

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

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

let nx = 0;

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

let ny = 0;

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

ny = G[1] + dy;

else

ny = G[1] - dy;

let r = new Object();

r.x = nx \* (this.X / 20);

r.y = ny \* (this.Y / 20);

this.rs.push(r);

p.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.drawN();

},

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="ru">

<head>

<meta charset="UTF-8">

<title>N-угольник Серпинского</title>

<style type="text/css">

body {

font-family: Arial, sans-serif;

background-color: #f4f4f4;

margin: 0;

padding: 20px;

}

.container {

display: flex;

flex-direction: column;

align-items: center;

}

.controls {

display: flex;

flex-wrap: wrap;

justify-content: center;

margin-bottom: 20px;

}

.controls label {

margin-right: 10px;

font-weight: bold;

}

.controls input {

width: 60px;

margin-right: 20px;

padding: 5px;

border: 1px solid #ccc;

border-radius: 4px;

}

.controls button {

padding: 10px 20px;

background-color: #4CAF50;

color: white;

border: none;

border-radius: 4px;

cursor: pointer;

}

.controls button:hover {

background-color: #45a049;

}

.svg-container {

display: flex;

justify-content: space-around;

width: 100%;

}

svg {

border: 1px solid #000000;

background-color: #ffffff;

box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

margin: 10px;

}

.chart-container {

width: 80%;

margin-top: 20px;

}

</style>

</head>

<body>

<div id="app" class="container">

<div class="controls">

<label for="N">Количество вершин (N): </label>

<input type="number" v-model.number="N" min="3" max="10" id="N">

<label for="n">Количество точек (n): </label>

<input type="number" v-model.number="n" min="1000" max="50000" id="n">

<label for="d">Коэффициент (d): </label>

<input type="number" v-model.number="d" min="1" max="10" step="0.1" id="d">

<label for="R">Радиус (R): </label>

<input type="number" v-model.number="R" min="1" max="20" step="0.1" id="R">

<label for="sX">Смещение по X (sX): </label>

<input type="number" v-model.number="sX" min="0" max="20" step="0.1" id="sX">

<label for="sY">Смещение по Y (sY): </label>

<input type="number" v-model.number="sY" min="0" max="20" step="0.1" id="sY">

<button @click="drawN">Перерисовать</button>

</div>

<div class="svg-container">

<svg :width="X \* 0.6" :height="Y \* 0.6">

<rect v-for="r in rs"

:fill="r.color"

fill-opacity="1"

:x="r.x"

:y="r.y"

width="1"

height="1"/>

</svg>

<svg :width="X \* 0.9" :height="Y \* 0.9">

<rect v-for="r in rs"

:fill="r.color"

fill-opacity="1"

:x="r.x"

:y="r.y"

width="1"

height="1"/>

</svg>

</div>

<div class="chart-container">

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

</div>

</div>

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

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

<script>

new Vue({

el: "#app",

data: {

X: 500,

Y: 500,

d: 1.6,

N: 5,

n: 20000,

R: 5,

sX: 10,

sY: 10,

rs: [],

previousPoint: null,

diagonalLength: Math.sqrt(500 \* 500 + 500 \* 500) / 2, // Половина длины диагонали

performanceData: {

N: [],

n: [],

d: [],

R: [],

sX: [],

sY: [],

time: []

}

},

methods: {

drawN() {

this.rs = [];

let vertices = this.calculateVertices();

let points = [vertices[0]];

let startTime = performance.now();

for (let i = 0; i < this.n; i++) {

let randomVertex = vertices[Math.floor(Math.random() \* vertices.length)];

let randomPoint = points[Math.floor(Math.random() \* points.length)];

let newPoint = this.calculateNewPoint(randomVertex, randomPoint);

let color = this.getPointColor(newPoint);

this.rs.push({ x: newPoint.x, y: newPoint.y, color: color });

points.push([newPoint.x / (this.X / 20), newPoint.y / (this.Y / 20)]);

if (i % 1000 === 0) {

this.$forceUpdate(); // Обновление DOM для плавного рендеринга

}

}

let endTime = performance.now();

let timeTaken = endTime - startTime;

this.performanceData.N.push(this.N);

this.performanceData.n.push(this.n);

this.performanceData.d.push(this.d);

this.performanceData.R.push(this.R);

this.performanceData.sX.push(this.sX);

this.performanceData.sY.push(this.sY);

this.performanceData.time.push(timeTaken);

this.updateChart();

let now = new Date();

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

},

calculateVertices() {

let vertices = [];

for (let i = 0; i < this.N; i++) {

let angle = 2 \* Math.PI \* i / this.N;

let x = this.R \* Math.cos(angle) + this.sX;

let y = this.R \* Math.sin(angle) + this.sY;

vertices.push([x, y]);

}

return vertices;

},

calculateNewPoint(vertex, point) {

let dx = Math.abs(vertex[0] - point[0]) / this.d;

let dy = Math.abs(vertex[1] - point[1]) / this.d;

let nx = vertex[0] > point[0] ? point[0] + dx : point[0] - dx;

let ny = vertex[1] > point[1] ? point[1] + dy : point[1] - dy;

return {

x: nx \* (this.X / 20),

y: ny \* (this.Y / 20)

};

},

getPointColor(newPoint) {

if (this.previousPoint) {

let distance = Math.sqrt(Math.pow(newPoint.x - this.previousPoint.x, 2) + Math.pow(newPoint.y - this.previousPoint.y, 2));

if (distance > this.diagonalLength) {

return this.getRandomColor();

}

}

this.previousPoint = newPoint;

return "#000000";

},

getRandomColor() {

return '#' + Math.floor(Math.random() \* 16777215).toString(16);

},

updateChart() {

if (this.chart) {

this.chart.destroy();

}

this.chart = new Chart(document.getElementById('performanceChart').getContext('2d'), {

type: 'bar',

data: {

labels: this.performanceData.time.map((\_, i) => `Run ${i + 1}`),

datasets: [

{

label: 'Время выполнения (мс)',

data: this.performanceData.time,

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

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

borderWidth: 1

}

]

},

options: {

scales: {

y: {

beginAtZero: true

}

}

}

});

}

},

created() {

let now = new Date();

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

this.drawN();

},

mounted() {

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)

FractalComponent.vue

<template>

<div class="container">

<div class="controls">

<label for="N">Количество вершин (N): </label>

<input type="number" v-model.number="N" min="3" max="10" id="N">

<label for="n">Количество точек (n): </label>

<input type="number" v-model.number="n" min="1000" max="50000" id="n">

<label for="d">Коэффициент (d): </label>

<input type="number" v-model.number="d" min="1" max="10" step="0.1" id="d">

<label for="R">Радиус (R): </label>

<input type="number" v-model.number="R" min="1" max="20" step="0.1" id="R">

<label for="sX">Смещение по X (sX): </label>

<input type="number" v-model.number="sX" min="0" max="20" step="0.1" id="sX">

<label for="sY">Смещение по Y (sY): </label>

<input type="number" v-model.number="sY" min="0" max="20" step="0.1" id="sY">

<button @click="drawN">Перерисовать</button>

</div>

<div class="svg-container">

<svg :width="X \* 0.6" :height="Y \* 0.6">

<rect v-for="(r, index) in rs"

:key="index"

:fill="r.color"

fill-opacity="1"

:x="r.x"

:y="r.y"

width="1"

height="1"/>

</svg>

<svg :width="X \* 0.9" :height="Y \* 0.9">

<rect v-for="(r, index) in rs"

:key="index"

:fill="r.color"

fill-opacity="1"

:x="r.x"

:y="r.y"

width="1"

height="1"/>

</svg>

</div>

<div class="chart-container">

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

</div>

</div>

</template>

<script>

import { ref, markRaw } from 'vue';

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

Chart.register(...registerables);

export default {

setup() {

const X = ref(500);

const Y = ref(500);

const d = ref(1.6);

const N = ref(5);

const n = ref(20000);

const R = ref(5);

const sX = ref(10);

const sY = ref(10);

const rs = ref([]);

const previousPoint = ref(null);

const diagonalLength = ref(Math.sqrt(500 \* 500 + 500 \* 500) / 2); // Половина длины диагонали

const performanceData = markRaw({

N: [],

n: [],

d: [],

R: [],

sX: [],

sY: [],

time: []

});

const drawN = async () => {

rs.value = [];

let startTime = performance.now();

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

method: 'POST',

headers: {

'Content-Type': 'application/json'

},

body: JSON.stringify({

N: N.value,

n: n.value,

d: d.value,

R: R.value,

sX: sX.value,

sY: sY.value

})

});

rs.value = await response.json();

let endTime = performance.now();

let timeTaken = endTime - startTime;

performanceData.N.push(N.value);

performanceData.n.push(n.value);

performanceData.d.push(d.value);

performanceData.R.push(R.value);

performanceData.sX.push(sX.value);

performanceData.sY.push(sY.value);

performanceData.time.push(timeTaken);

updateChart();

let now = new Date();

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

};

const getPointColor = (newPoint) => {

if (previousPoint.value) {

let distance = Math.sqrt(Math.pow(newPoint.x - previousPoint.value.x, 2) + Math.pow(newPoint.y - previousPoint.value.y, 2));

if (distance > diagonalLength.value) {

return getRandomColor();

}

}

previousPoint.value = newPoint;

return "#000000";

};

const getRandomColor = () => {

return '#' + Math.floor(Math.random() \* 16777215).toString(16);

};

let chart = null;

const updateChart = () => {

if (chart) {

chart.destroy();

}

chart = new Chart(document.getElementById('performanceChart').getContext('2d'), {

type: 'bar',

data: {

labels: performanceData.time.map((\_, i) => `Run ${i + 1}`),

datasets: [

{

label: 'Время выполнения (мс)',

data: performanceData.time,

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

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

borderWidth: 1

}

]

},

options: {

scales: {

y: {

beginAtZero: true

}

}

}

});

};

return {

X,

Y,

d,

N,

n,

R,

sX,

sY,

rs,

previousPoint,

diagonalLength,

performanceData,

drawN,

getPointColor,

getRandomColor,

updateChart

};

},

created() {

let now = new Date();

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

this.drawN();

},

mounted() {

let now = new Date();

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

}

};

</script>

<style scoped>

.container {

display: flex;

flex-direction: column;

align-items: center;

}

.controls {

display: flex;

flex-wrap: wrap;

justify-content: center;

margin-bottom: 20px;

}

.controls label {

margin-right: 10px;

font-weight: bold;

}

.controls input {

width: 60px;

margin-right: 20px;

padding: 5px;

border: 1px solid #ccc;

border-radius: 4px;

}

.controls button {

padding: 10px 20px;

background-color: #4CAF50;

color: white;

border: none;

border-radius: 4px;

cursor: pointer;

}

.controls button:hover {

background-color: #45a049;

}

.svg-container {

display: flex;

justify-content: space-around;

width: 100%;

}

svg {

border: 1px solid #000000;

background-color: #ffffff;

box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

margin: 10px;

}

.chart-container {

width: 80%;

margin-top: 20px;

}

</style>

App.vue

<template>

<div id="app">

<FractalComponent />

</div>

</template>

<script>

import FractalComponent from './components/FractalComponent.vue';

export default {

name: 'App',

components: {

FractalComponent

}

};

</script>

<style>

#app {

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

-webkit-font-smoothing: antialiased;

-moz-osx-font-smoothing: grayscale;

text-align: center;

color: #2c3e50;

margin-top: 60px;

}

</style>

server.js

const express = require('express');

const cors = require('cors');

const app = express();

const port = 3000;

app.use(cors());

app.use(express.json());

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

const { N, n, d, R, sX, sY } = req.body;

let vertices = calculateVertices(N, R, sX, sY);

let points = [vertices[0]];

let rs = [];

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

let randomVertex = vertices[Math.floor(Math.random() \* vertices.length)];

let randomPoint = points[Math.floor(Math.random() \* points.length)];

let newPoint = calculateNewPoint(randomVertex, randomPoint, d);

rs.push({ x: newPoint.x, y: newPoint.y });

points.push([newPoint.x, newPoint.y]);

}

res.json(rs);

});

function calculateVertices(N, R, sX, sY) {

let vertices = [];

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

let angle = 2 \* Math.PI \* i / N;

let x = R \* Math.cos(angle) + sX;

let y = R \* Math.sin(angle) + sY;

vertices.push([x, y]);

}

return vertices;

}

function calculateNewPoint(vertex, point, d) {

let dx = Math.abs(vertex[0] - point[0]) / d;

let dy = Math.abs(vertex[1] - point[1]) / d;

let nx = vertex[0] > point[0] ? point[0] + dx : point[0] - dx;

let ny = vertex[1] > point[1] ? point[1] + dy : point[1] - dy;

return { x: nx, y: ny };

}

app.listen(port, () => {

console.log(`Server is running on http://localhost:${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 ноутбуке 1 |
|  |
|  |
| 1. – Результат запуска SPWA на ноутбуке 2 |
|  |
|  |
| 1. – Результат запуска LSPWA на ноутбуке 1 |
|  | |
|  | |
|  |
| 1. – Результат запуска LSPWA на ноутбуке 2 |

На основе приведенных результатов, 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.