|  |  |
| --- | --- |
| **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  Шуриков Д. А.  Кожак И. А.  Проверил:  Доцент кафедры УиЗИ, к.т.н.  Сафронов А. И. |
| Москва 2024 | |

Оглавление

[1. Цель работы 2](#_Toc185637266)

[2. ФОРМУЛИРОВАКА ЗАДАЧИ 3](#_Toc185637267)

[3. таблица соответсвия переменных 3](#_Toc185637268)

[4. Спецификация оборудования 4](#_Toc185637269)

[5. Web-страница 4](#_Toc185637270)

[5.1. SPWA (код после оптимизации) 4](#_Toc185637271)

[5.1.1 Результат отображения SPWA 10](#_Toc185637272)

[5.1.2 Диаграмма построения 10](#_Toc185637273)

[5.1.3 Обоснование размеров 11](#_Toc185637274)

[5.1.4 Сети Петри 11](#_Toc185637275)

[5.2. LSPWA (код после оптимизации) 12](#_Toc185637276)

[5.2.1 App.vue 12](#_Toc185637277)

[5.2.2 Результат отображения LSPWA 16](#_Toc185637278)

[5.2.3 Диаграмма построения 17](#_Toc185637279)

[5.2.4 Обоснование размеров 17](#_Toc185637280)

[5.2.5 Сети Петри 18](#_Toc185637281)

[5.3. Анализ быстродействия программного обеспечения 19](#_Toc185637282)

[6. вывод по работе 23](#_Toc185637283)

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*** |
| **10** | Треугольник Серпинского | *4* | *0,1; 0,2; 0,3; 0,4* | Менять на новый, выставляемый случайным образом, цвет очередной точки построения треугольника Серпинского, если расстояние до этой точки от предыдущей составляет более половины длины диагонали области построения. | Управлять значением делителя, влияющим на габариты внутренних треугольников Серпинского, в разумных пределах | **16** |

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: Intel Core i3
  + Оперативная память: 8 ГБ
  + Видеокарта: Intel Iris Plus Graphics
  + Операционная система: macOS Ventura 13.0

1. Web-страница
   1. SPWA (код после оптимизации)

<!DOCTYPE html>

<html lang="ru">

<head>

<meta charset="UTF-8">

<title>Треугольник 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;

}

.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">

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

<br>

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

<br>

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

<br>

<div class="svg-container">

<svg :width="X \* 0.1" :height="Y \* 0.1">

<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 \* 0.2" :height="Y \* 0.2">

<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 \* 0.3" :height="Y \* 0.3">

<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 \* 0.4" :height="Y \* 0.4">

<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="js/vue-2-6-14.js"></script>

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

<script>

new Vue({

el: "#app",

data: {

X: 1000,

Y: 1000,

d: 1.9,

n: 60000,

R: 4,

sX: 3,

sY: 4,

rs: [],

previousPoint: null,

diagonalLength: Math.sqrt(1000 \* 1000 + 1000 \* 1000) / 2, // Половина длины диагонали

performanceData: {

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.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 < 3; i++) {

let angle = 2 \* Math.PI \* i / 3;

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

|  |
| --- |
| Изображение выглядит как текст, снимок экрана, круг, Шрифт  Автоматически созданное описание |
| 1. – Сети Петри |

* 1. LSPWA (код после оптимизации)

App.vue

<script>

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

Chart.register(...registerables);

export default {

data() {

return {

X: 1000,

Y: 1000,

d: 1.9,

n: 60000,

R: 4,

sX: 3,

sY: 4,

rs: [],

previousPoint: null,

diagonalLength: Math.sqrt(1000 \* 1000 + 1000 \* 1000) / 2, // Половина длины диагонали

performanceData: {

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)]);

}

let endTime = performance.now();

let timeTaken = endTime - startTime;

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 < 3; i++) {

let angle = 2 \* Math.PI \* i / 3;

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

}

}

}

});

}

},

mounted() {

let now = new Date();

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

this.drawN();

now = new Date();

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

}

}

</script>

<template>

<main style="width: 80vw">

<header style="justify-self: center; padding: 0">

<img alt="Vue logo" src="./assets/logo.svg" width="125" height="125" />

</header>

<div style="display: flex; flex-direction: column; align-items: center">

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

<br>

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

<br>

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

<br>

<div class="svg-container">

<svg :width="X \* 0.1" :height="Y \* 0.1">

<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 \* 0.2" :height="Y \* 0.2">

<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 \* 0.3" :height="Y \* 0.3">

<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 \* 0.4" :height="Y \* 0.4">

<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>

</main>

</template>

<style scoped>

header {

line-height: 1.5;

}

@media (min-width: 1024px) {

header {

display: flex;

place-items: center;

padding-right: calc(var(--section-gap) / 2);

}

}

</style>

Результат отображения LSPWA

|  |
| --- |
| Изображение выглядит как текст, снимок экрана, Шрифт  Автоматически созданное описание |
| 1. – Запуск |
|  |
| Изображение выглядит как снимок экрана, линия, белый, шаблон  Автоматически созданное описание |
| 1. – Результат отображения |

Диаграмма построения

|  |
| --- |
| Изображение выглядит как текст, снимок экрана, диаграмма, График  Автоматически созданное описание |
| 1. – Диаграмма |

Обоснование размеров

=

Сети Петри

|  |
| --- |
| Изображение выглядит как текст, снимок экрана, круг, Шрифт  Автоматически созданное описание |
| 1. – Сети Петри |

* 1. Анализ быстродействия программного обеспечения

|  |
| --- |
| Изображение выглядит как текст, диаграмма, Прямоугольник, шаблон  Автоматически созданное описание |
| 1. – Результат запуска SPWA на ноутбуке 1 |
| Изображение выглядит как текст, шаблон, диаграмма, снимок экрана  Автоматически созданное описание |
|  |
| 1. – Результат запуска SPWA на ноутбуке 2 |
| Изображение выглядит как Прямоугольник, диаграмма, линия, шаблон  Автоматически созданное описание |
|  |
| 1. – Результат запуска LSPWA на ноутбуке 1 |
| Изображение выглядит как диаграмма, текст, Прямоугольник, линия  Автоматически созданное описание | |
|  | |
|  |
| 1. – Результат запуска LSPWA на ноутбуке 2 |

На основе приведенных результатов можно сделать вывод, что на результативность влияет техническая составляющая оборудования, в то время как различие между SPWA и LSPWA не значительно.

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 SPWA и LSPWA демонстрирует одинаковую производительность. Оба варианта подходят для реализации поставленной задачи.