|  |  |
| --- | --- |
| **Российский университет транспорта (МИИТ)**  **Институт транспортной техники и систем управления**  **Кафедра «Управление и защита информации»** | |
| **Отчет**  **по лабораторной работе №7**  **по дисциплине «Web-программирование» Вариант 12** | |
|  | Выполнили:  Студенты группы ТКИ-541  Кузнецов К. К.  Степанов Д. Е. |
|  | Проверил:  Доцент кафедры УиЗИ к.т.н. с.н.с.  Сафронов А. И. |
| Москва 2024 | |

**Оглавление**

[1 Цель работы 3](#_Toc185533051)

[2 Рабочее задание 4](#_Toc185533052)

[3 Таблица соответствия переменных и методов, используемых в приложениях 5](#_Toc185533053)

[4 Спецификация оборудования, на котором выполнялась работа 6](#_Toc185533054)

[5 Содержательная часть 7](#_Toc185533055)

[5.1 Код spwa\_Кузнецов\_Степанов.html 7](#_Toc185533056)

[5.2 Код App.vue 13](#_Toc185533057)

[5.3 Код main.js 14](#_Toc185533058)

[5.4 Код router/index.js 15](#_Toc185533059)

[5.5 Код components/Fractal.vue 16](#_Toc185533060)

[5.6 Отображение в браузере 21](#_Toc185533061)

[5.7 Математическая формула для расчета фрактала Жюлиа 23](#_Toc185533062)

[6 Анализ быстродействия системы 24](#_Toc185533063)

[6.1 Устранение избыточности 24](#_Toc185533064)

[6.2 Анализ влияния количества пикселей 27](#_Toc185533065)

[6.3 Сравнение производительности программного обеспечения на различных устройствах 29](#_Toc185533066)

[7 Вывод 32](#_Toc185533067)

1. Цель работы

Средствами *SVG* или *Canvas* под управлением фреймворка *Vue.js* построить программное, браузерное *web*-обеспечение, реализующее анализ быстродействия формируемого фрактального изображения по варианту.

1. Рабочее задание

1. Устранить избыточность в предложенном базовом коде. Обосновать предложенные альтернативы.

2. Модифицировать серверную часть *web*-приложения согласно таблице вариантов индивидуального задания.

3. Модифицировать клиентскую часть *web*-приложения согласно таблице вариантов индивидуального задания.

4. Реализовать *SPWA* и *LSPWA* – сопоставить их быстродействие на как минимум двух различных персональных компьютерах (домашнем, аудиторном, ноутбуке, моноблоке и пр.).

5. Восстановить математическую форму записи, предложенную для расчёта фрактала, по коду согласно варианту индивидуального задания.

6. Построить столбчатые диаграммы влияния изменяемых параметров на скорость расчёта и визуализации фрактальной графики.

7. При значениях *R* (определение в **Приложении 1**), больших единицы, предусмотреть возможность вывода областей построения как порознь, так и совместно с применением тега <*template*> или компонентов *Vue.js*.

8. Обосновывать посредством предоставления соответствующих расчётов, что выбранные величины ширин и высот удовлетворяют заданному количеству единиц, заданных в Мегапикселях (МПкс).

***Базовая структура отчёта по работе:***

1. Цель работы.

2. Формулировка задачи.

3. Таблица соответствия переменных.

4. Спецификация оборудования, на котором выполнялась работа.

5. Содержательная часть (код *web*-приложения + отображение в браузере + Сеть Петри).

6. Анализ быстродействия программного обеспечения.

7. Развёрнутый вывод.

***Не использовать:*** *jQuerry* и «чистый» *JavaScript*.

1. Таблица соответствия переменных и методов, используемых в приложениях

|  |  |
| --- | --- |
| Переменная/метод | Описание |
| props | Валидация параметров, передающихся на страницу |
| pixels | Массив пикселей фрактала |
| calculationTimes | Массив миллисекунд, за которые был посчитан фрактал |
| maxIter | Количество итераций, использующихся при построении фрактала |
| zoom | Масштаб фракталов |
| moveX | Сдвиг области построения фрактала по оси X |
| moveY | Сдвиг области построения фрактала по оси Y |
| drawJulia() | Расчет пикселей фрактала |
| updateChart() | Обновление данных в столбчатой диаграмме |

1. Спецификация оборудования, на котором выполнялась работа

На рисунке 1 изображена спецификация каждого из устройств, на которых было протестировано приложение:



1. – Спецификация устройств
2. Содержательная часть
   1. Код spwa\_Кузнецов\_Степанов.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Фрактал Жюлиа</title>

<script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>

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

<style>

svg {

border: 1px solid #000000;

}

input {

width: 60px;

}

.chart-container {

margin-top: 20px;

}

button {

margin-left: 20px;

margin-bottom: 10px;

}

</style>

</head>

<body>

<div id="jlf">

<h1>Фрактал Жюлиа</h1>

<svg :width="wdt" :height="hgt">

<rect v-for="p in pixels"

:key="p.x + '-' + p.y"

:fill="p.color"

:fill-opacity="p.alpha"

:x="p.x"

:y="p.y"

width="1"

height="1"/>

</svg>

<div>

<label for="maxIter">Max Iterations:</label>

<input type="number" id="maxIter" v-model="maxIter">

<label for="zoom">Zoom:</label>

<input type="number" id="zoom" v-model="zoom">

<label for="moveX">Move X:</label>

<input type="number" id="moveX" v-model="moveX">

<label for="moveY">Move Y:</label>

<input type="number" id="moveY" v-model="moveY">

<button @click="drawJulia()">Update Fractal</button>

</div>

</div>

<div class="chart-container">

<h2>Время расчёта</h2>

<div id="chart"></div>

</div>

<script>

const app = Vue.createApp({

data() {

return {

wdt: 1000,

hgt: 1000,

pixels: [],

calculationTimes: [], // Время расчёта

labels: [], // Метки для диаграммы

maxIter: 1000, // Default max iterations

zoom: 1, // Default zoom

moveX: 0, // Default move X

moveY: 0 // Default move Y

};

},

methods: {

drawJulia() {

const pixels = [];

const cIm = 0.62662;

const cRe = -0.38135;

let newRe, newIm, oldRe, oldIm;

const zoom = this.zoom;

const moveX = this.moveX;

const moveY = this.moveY;

const maxIter = this.maxIter;

const startTime = performance.now();

for (let x = 0; x < this.wdt; x++) {

for (let y = 0; y < this.hgt; y++) {

newRe = 1.5 \* (x - this.wdt / 2) / (0.5 \* zoom \* this.wdt) + moveX;

newIm = (y - this.hgt / 2) / (0.5 \* zoom \* this.hgt) + moveY;

let i;

for (i = 0; i < maxIter; i++) {

oldRe = newRe;

oldIm = newIm;

newRe = (oldRe \* oldRe) - (oldIm \* oldIm) + cRe;

newIm = 2 \* oldRe \* oldIm + cIm;

if ((newRe \* newRe + newIm \* newIm) > 4) {

break;

}

}

const pix = {};

pix.x = x;

pix.y = y;

const R = (i \* 9) % 255;

const G = 0;

const B = (i \* 9) % 255;

pix.color = "#" + (R.toString(16).length < 2 ? "0" + R.toString(16) : R.toString(16))

+ (G.toString(16).length < 2 ? "0" + G.toString(16) : G.toString(16))

+ (B.toString(16).length < 2 ? "0" + B.toString(16) : B.toString(16));

pix.alpha = 1;

pixels.push(pix);

}

}

const endTime = performance.now();

this.calculationTimes.push(endTime - startTime);

this.pixels = pixels;

this.labels.push(`Run ${this.calculationTimes.length}`);

this.updateChart();

},

updateChart() {

const options = {

chart: {

type: 'bar',

height: 350,

},

series: [{

name: 'Calculation Time (ms)',

data: this.calculationTimes.map(time => time.toFixed(3)),

}],

xaxis: {

categories: this.labels,

},

};

if (this.chart) {

this.chart.destroy();

}

this.chart = new ApexCharts(document.getElementById('chart'), options);

this.chart.render();

}

},

created() {

this.drawJulia();

}

});

app.mount('#jlf');

</script>

</body>

</html>

* 1. Код App.vue

<script setup>

import { RouterLink, RouterView } from 'vue-router'

</script>

<template>

<header>

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

<div class="wrapper">

<nav>

<RouterLink to="/">Fractal 1000x1000</RouterLink> |

<RouterLink to="/large">Fractal 1732x1732</RouterLink>

</nav>

</div>

</header>

<RouterView />

</template>

* 1. Код main.js

import './assets/main.css'

import { createApp } from 'vue'

import App from './App.vue'

import router from './router'

const app = createApp(App)

app.use(router)

app.mount('#app')

* 1. Код router/index.js

import { createRouter, createWebHistory } from 'vue-router';

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

const routes = [

{ path: '/', name: 'Fractal1000', component: Fractal, props: { wdt: 1000, hgt: 1000 } },

{ path: '/large', name: 'Fractal1732', component: Fractal, props: { wdt: 1732, hgt: 1732 } },

];

const router = createRouter({

history: createWebHistory(),

routes,

});

export default router;

* 1. Код components/Fractal.vue

<template>

<div>

<h1>Фрактал Жюлиа ({{ wdt }} x {{ hgt }})</h1>

<svg :width="wdt" :height="hgt">

<rect v-for="p in pixels"

:key="p.x + '-' + p.y"

:fill="p.color"

:fill-opacity="p.alpha"

:x="p.x"

:y="p.y"

width="1"

height="1"/>

</svg>

<div>

<label for="maxIter">Max Iterations:</label>

<input type="number" id="maxIter" v-model="maxIter">

<label for="zoom">Zoom:</label>

<input type="number" id="zoom" v-model="zoom">

<label for="moveX">Move X:</label>

<input type="number" id="moveX" v-model="moveX">

<label for="moveY">Move Y:</label>

<input type="number" id="moveY" v-model="moveY">

<button @click="drawJulia">Update Fractal</button>

</div>

</div>

<div>

<h2>Calculation Time</h2>

<ApexChart type="bar" :options="chartOptions" :series="chartOptions.series" />

</div>

</template>

<script>

import VueApexCharts from 'vue3-apexcharts';

export default {

components: {

ApexChart: VueApexCharts,

},

name: 'Fractal',

props: {

wdt: { type: Number, required: true },

hgt: { type: Number, required: true },

},

data() {

return {

pixels: [],

calculationTimes: [],

labels: [],

maxIter: 1000,

zoom: 1,

moveX: 0,

moveY: 0

};

},

methods: {

drawJulia() {

const pixels = [];

const cIm = 0.62662;

const cRe = -0.38135;

let newRe, newIm, oldRe, oldIm;

const zoom = this.zoom;

const moveX = this.moveX;

const moveY = this.moveY;

const maxIter = this.maxIter;

const startTime = performance.now();

for (let x = 0; x < this.wdt; x++) {

for (let y = 0; y < this.hgt; y++) {

newRe = 1.5 \* (x - this.wdt / 2) / (0.5 \* zoom \* this.wdt) + moveX;

newIm = (y - this.hgt / 2) / (0.5 \* zoom \* this.hgt) + moveY;

let i;

for (i = 0; i < maxIter; i++) {

oldRe = newRe;

oldIm = newIm;

newRe = (oldRe \* oldRe) - (oldIm \* oldIm) + cRe;

newIm = 2 \* oldRe \* oldIm + cIm;

if ((newRe \* newRe + newIm \* newIm) > 4) {

break;

}

}

const pix = {};

pix.x = x;

pix.y = y;

const R = (i \* 9) % 255;

const G = 0;

const B = (i \* 9) % 255;

pix.color = "#" + (R.toString(16).length < 2 ? "0" + R.toString(16) : R.toString(16))

+ (G.toString(16).length < 2 ? "0" + G.toString(16) : G.toString(16))

+ (B.toString(16).length < 2 ? "0" + B.toString(16) : B.toString(16));

pix.alpha = 1;

pixels.push(pix);

}

}

const endTime = performance.now();

this.calculationTimes.push(endTime - startTime);

this.pixels = pixels;

this.labels.push(`Run ${this.calculationTimes.length}`);

this.updateChart();

},

updateChart() {

this.chartOptions = {

chart: {

type: 'bar',

height: 350,

},

series: [{

name: 'Calculation Time (ms)',

data: this.calculationTimes.map(time => time.toFixed(3)),

}],

xaxis: {

categories: this.labels,

},

};

}

},

created() {

this.drawJulia();

},

};

</script>

<style scoped>

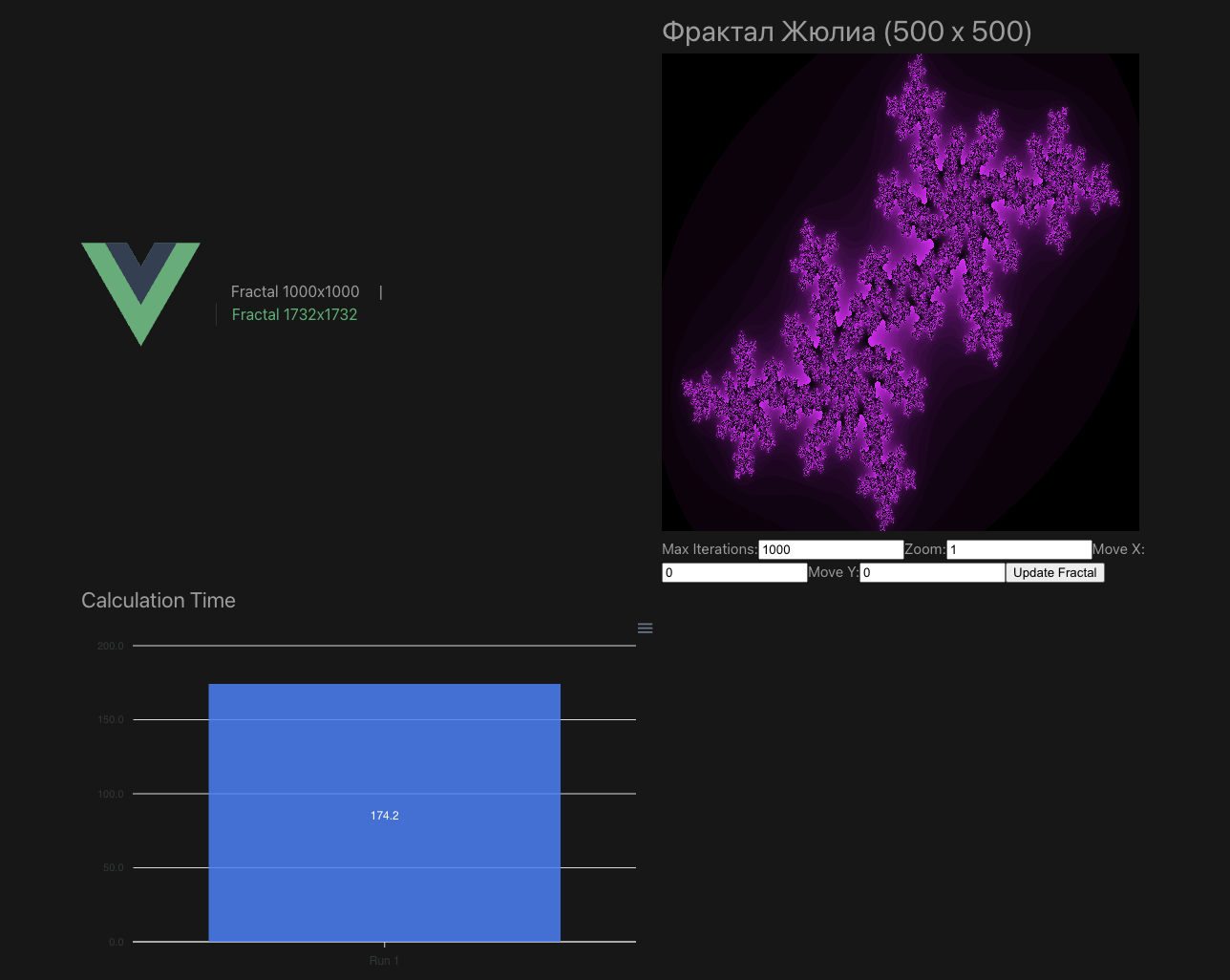
.chart-container {

margin-top: 20px;

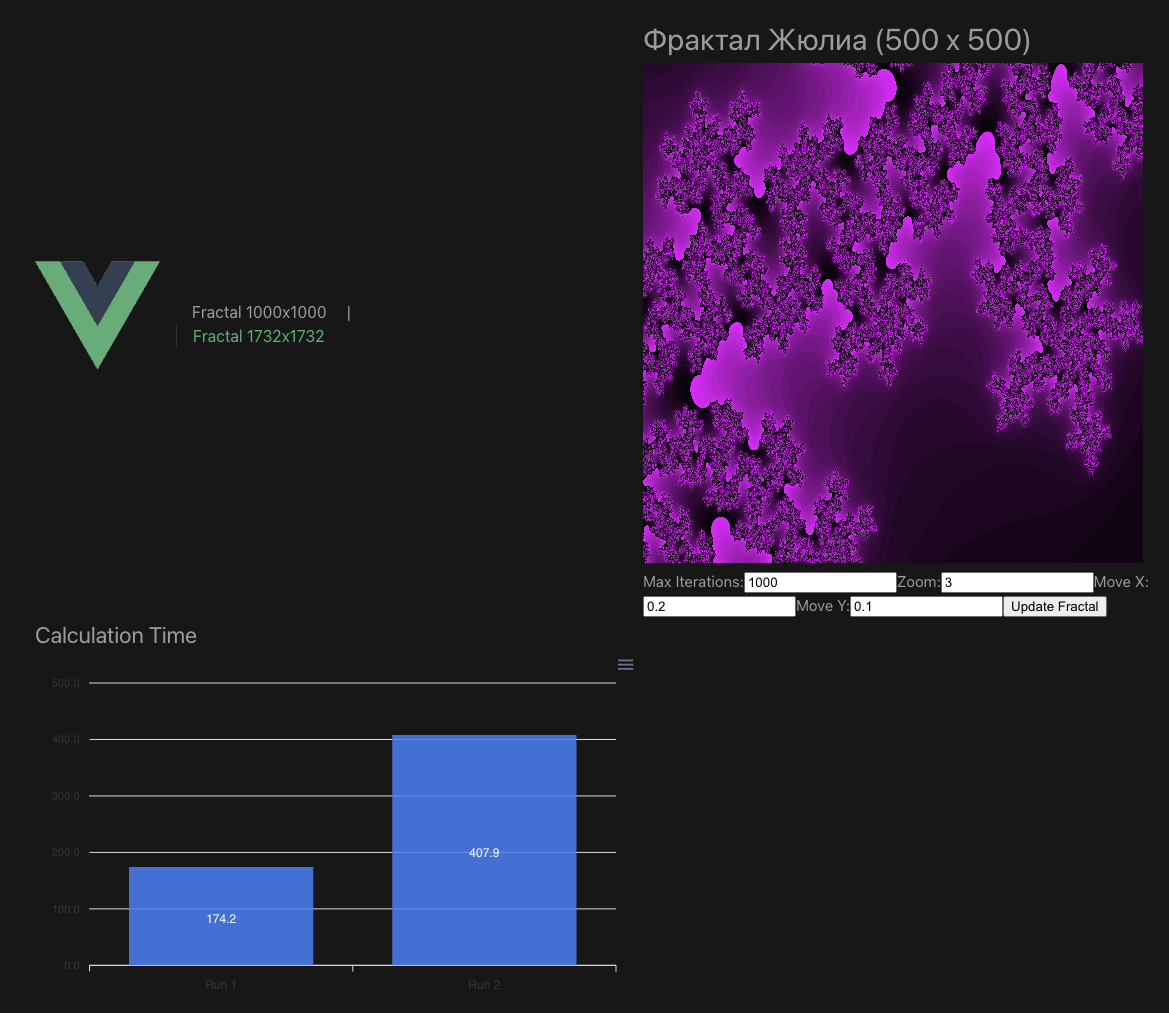
}

</style>

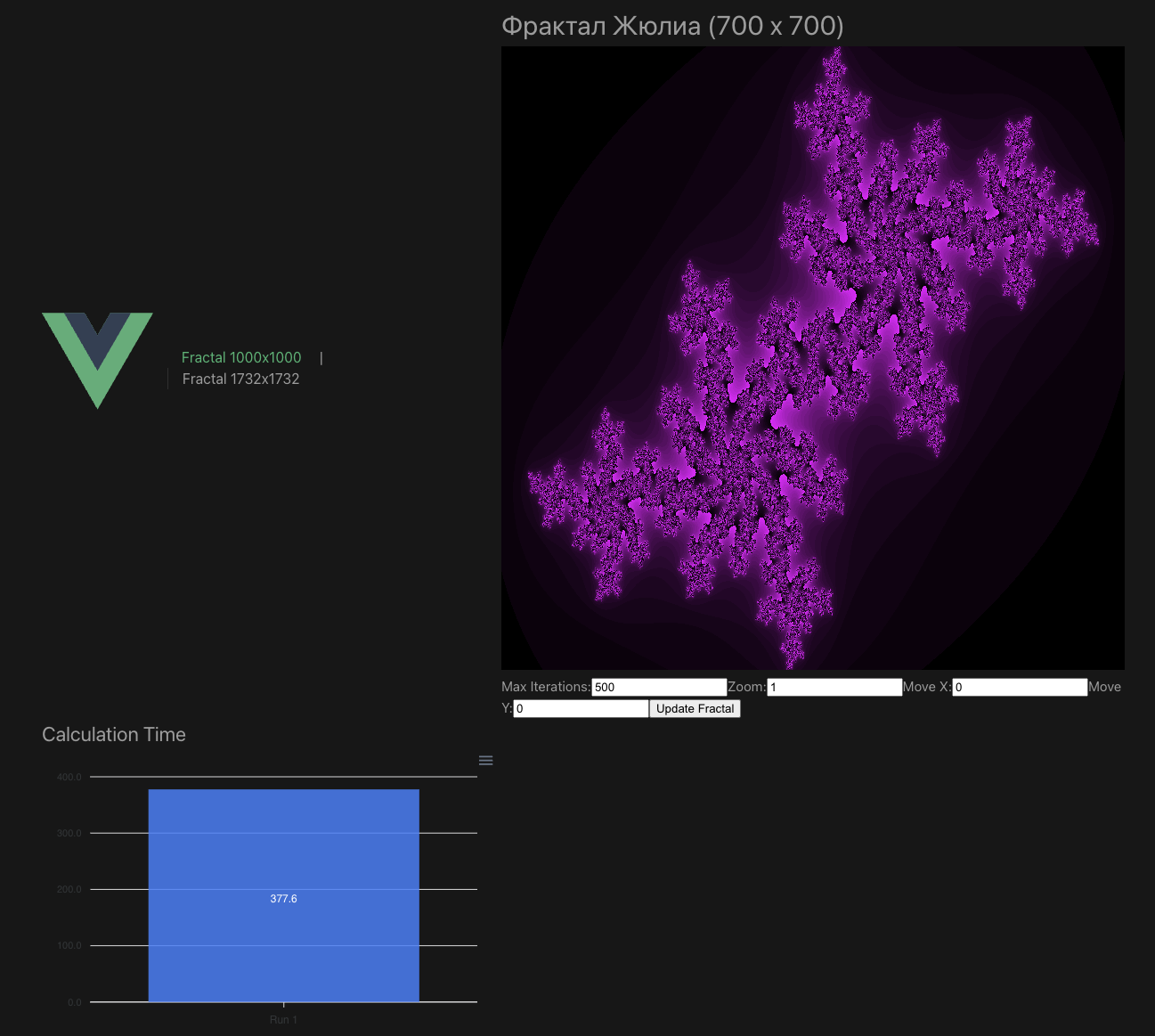
* 1. Отображение в браузере



1. – Начальное состояние при 250 кПкс



1. – Состояние при измененных масштабе и области



1. – Состояние при 490 кПкс
   1. Математическая формула для расчета фрактала Жюлиа
2. Анализ быстродействия системы
   1. Устранение избыточности

В качестве изначального алгоритма был представлен вариант, где в переменную приложения pixels происходила вставка каждого посчитанного в цикле пикселя по одному. Ниже представлен фрагмент данного кода:

for {

…

pix = …;

this.pixels.push(pix);

};

В предложенной мной альтернативе был представлен вариант, где сначала высчитываются все пиксели, а потом присваиваются в массив pixels – переменную приложения:

const calculatedPixels = [];

for {

…

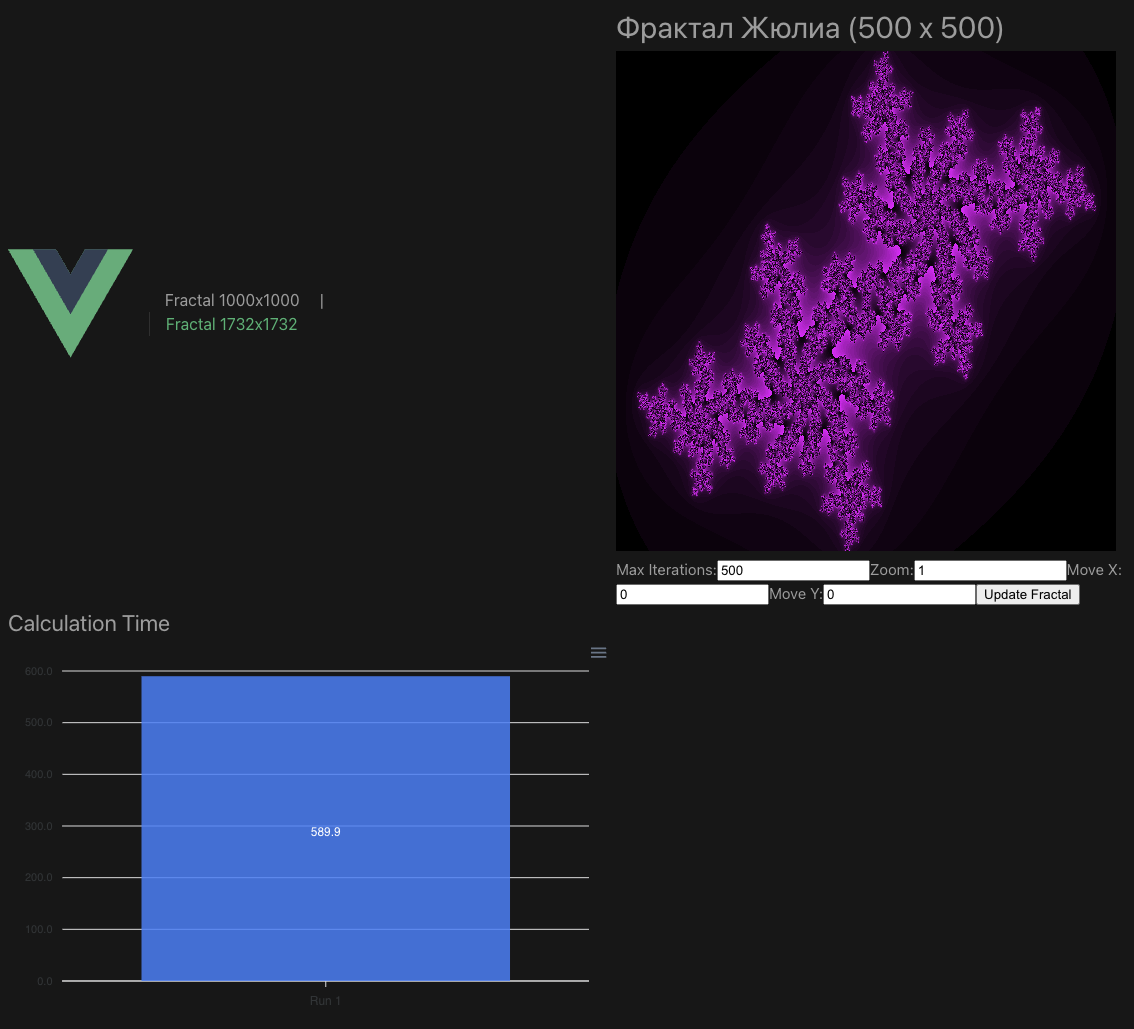
pix = …;

calculatedPixels.push(pix);

};

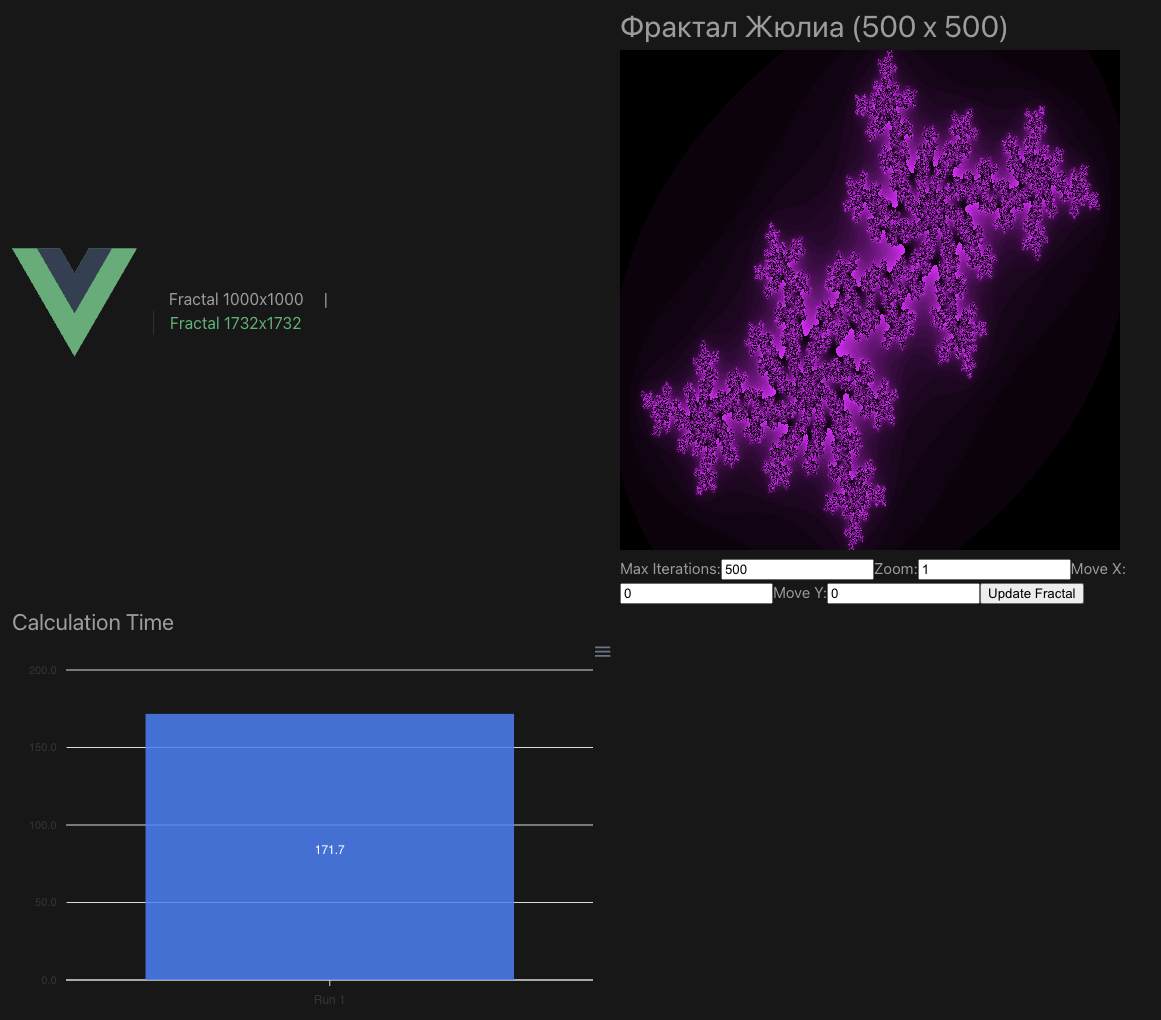
this.pixels = calculatedPixels;

На рисунке 5 представлено отображение состояния системы при использовании первоначального алгоритма:



1. – Отображение системы с избыточностью в алгоритме

На рисунке 6 представлено отображение состояния системы при использовании алгоритма с устраненной избыточностью:



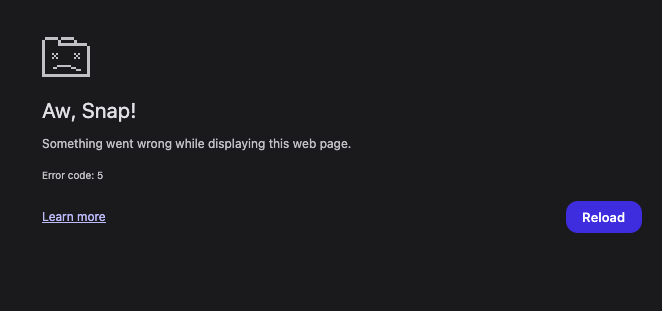
1. – Отображение системы без избыточности в алгоритме

Calculation Time в первом случае равен 589.89 мс, во втором же случае 171.7 мс. Вследствие этого следует вывод о положительном влиянии предложенного решения на производительность системы.

* 1. Анализ влияния количества пикселей

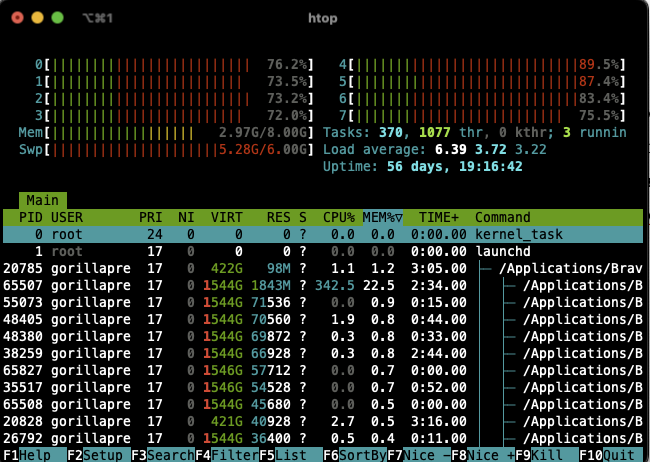
В задании для варианта 12 было предложено построить фрактал в двух разных масштабах: 1 МПкс и 3 МПкс. Для этого были выбраны следующие размеры svg:

Однако экспериментально было выяснено, что такое количество объектов слишком высоко, так как построить 1 МПкс получалось лишь один раз при первом входе на страницу с жуткими тормозами, а при попытке построить 3 МПкс приложение замирало:



1. – Состояние системы при попытке построить 3 МПкс

На рисунке 8 приведен скриншот программы htop, показывающая список запущенных процессов и их влияние на вычислительные мощности устройства:



1. – Скриншот из программы htop

Самое большое влияние на загруженность устройства оказывает процесс запущенного браузера /Applications/Brave.

Такое поведение наблюдается на всех тестируемых устройствах. Наиболее вероятно, это связано с тем, что работа с отображением svg ложится на видеокарту устройства, но в каждом из тестируемых устройствах дискретная видеокарта отсутствует, поэтому вся нагрузка ложится на все ядра процессора.

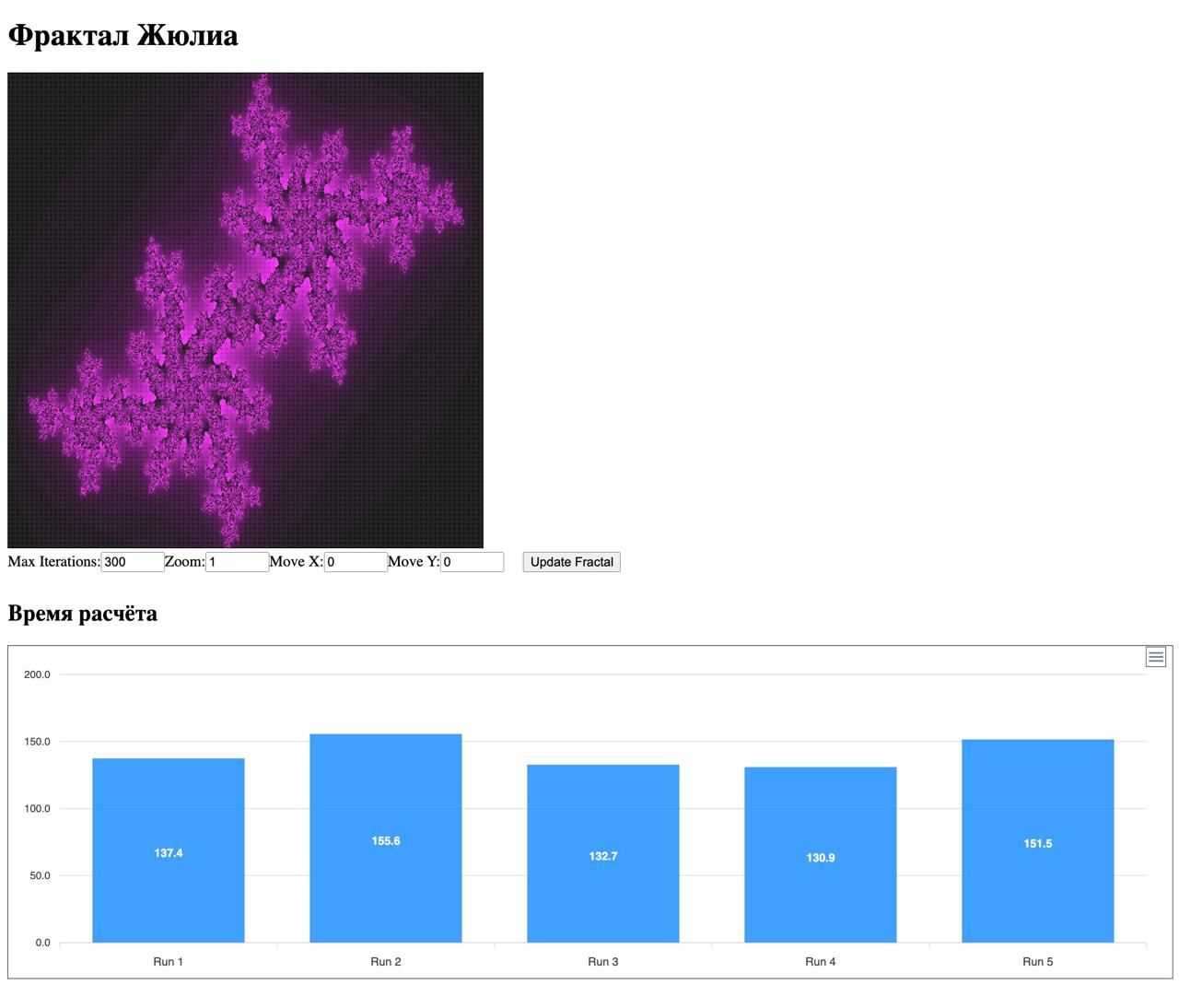
* 1. Сравнение производительности программного обеспечения на различных устройствах

На рисунке 9 приведены метрики расчета фрактала Жюлиа на Macbook Air M1:



1. – Метрики расчета фрактала

Macbook Pro M1 Pro:



1. – Метрики расчета фрактала

Macbook Pro Intel Core i7:



1. – Метрики расчета квартала
2. Вывод

В результате работы был изучен фрактал Жюлиа, а также был построен с помощью SPWA и LSPWA. Была устранена избыточность в предложенном базовом коде, были посчитаны метрики времени расчета фрактала и отображены с помощью столбчатых диаграмм. А также во время работы был произведен анализ быстродействия полученного программного обеспечения на различных устройствах, входе которого было выяснено, что скорость расчета фрактала напрямую зависит от мощности графического ядра устройства.