УДК 004.021

А.Н. Щербакова, ассист.; Д.М. Романенко, доц. (БГТУ, г. Минск)

ОСОБЕННОСТИ ПРОГРАММНОЙ ГЕНЕРАЦИИ ЗАЩИТНЫХ ИЗОБРАЖЕНИЙ ПО ТЕХНОЛОГИИ АВТОТИПНОГО ЦВЕТОВОГО СИНТЕЗА

В настоящее время задача защиты авторских прав на изображения имеет большую важность. Одним из методов защиты является генерация изображений по технологии автотипного синтеза.

Автотипный синтез цвета – это получение оттенков цвета путем пространственного совмещения растровых или векторных изображений (линий, точек) разных цветов. Реализация программного продукта по данной теме подразумевает разработку нескольких модулей: для установки параметров генерируемого изображения, выбранного примитива, переходов, реализация алгоритма генерации защитного изображения, используя выбранные параметры, и сохранение сгенерированного изображения в формате SVG.

Для разработки программы используется javascript-фреймворка vue.js.

Структура приложения представлена на рисунке 1.

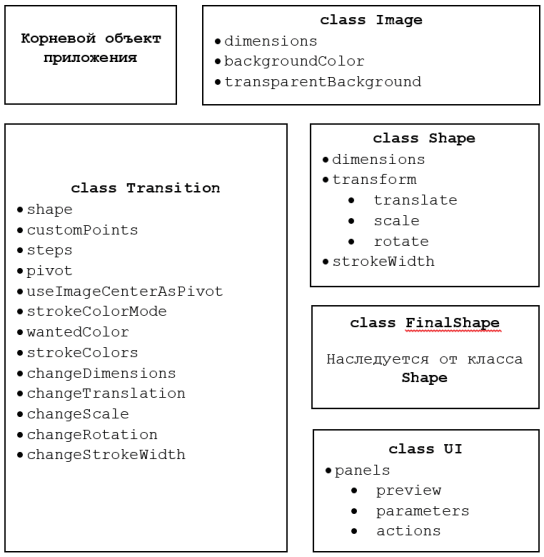

Рисунок 1 – Структура программы для генерации защитного изображения

Класс Image хранит данные об изображении и содержит такие параметры, как размеры изображения по горизонтали и вертикали, цвет фона, прозрачный фон.

Класс Shape хранит данные о примитиве и содержит такие параметры, как размеры по горизонтали и вертикали, трансформации (смещение, масштабирование, поворот), толщина обводки.

Класс FinalShape наследуется от класса примитива.

Класс Transition хранит данные о переходе между примитивами и содержит параметры: тип примитива (эллипс, круг, квадрат, прямоугольник, треугольник, линия и произвольный примитив); координаты точек, если пользователь выбрал произвольный примитив; количество шагов перехода; опорная точка, относительно которой поворачиваются и масштабируются примитивы; параметры для настройки отображения панелей.

Класс UI хранит данные об отображении элементов интерфейса.

Фрагмент кода класса Image представлен на листинге 1.

```
class Image {
    constructor() {
        this.dimensions = {
            x: 400, y: 400
        }
        this.backgroundColor = "#000000"
        this.transparentBackground = false
    }
}
```

Листинг 1 – Фрагмент кода класса Image

Фрагмент кода класса Shape представлен на листинге 2.

```
class Shape {
    constructor() {
        this.dimensions = {x: 100, y: 200},
        this.transform = {
            translate: {x: 0, y: 0},
            scale: {x: 1, y: 1},
            rotate: 0,
        }
        this.strokeWidth = 1
    }
}
```

Листинг 2 – Фрагмент кода класса Shape

В третьем блоке можно выбрать тип примитива, количество шагов и желаемый результирующий цвет. Желаемый результирующий цвет формируется линиями разных цветов (голубой, желтый, пурпурный).

Например, чтобы получить зеленый цвет, нужно совместить голубой и желтые цвета. Фрагменты кода методов для расчетов представлены на листинге 3. Примеры генерации изображений приведены на рисунках 2, 3.

```
computed: {
        center() {
            return {
                x: this.image.dimensions.x / 2,
                y: this.image.dimensions.y / 2
            }
        },
        widthDelta() {
            if (!this.transition.changeDimensions) return 0
            return (this.shape.to.dimensions.x -
      this.shape.from.dimensions.x) / this.transition.steps
        },
        heightDelta() {
            if (!this.transition.changeDimensions) return 0
            return (this.shape.to.dimensions.y -
this.shape.from.dimensions.y) / this.transition.steps
        },
        strokeWidthDelta() {
            if (!this.transition.changeStrokeWidth) return 0
            return (this.shape.to.strokeWidth -
this.shape.from.strokeWidth) / this.transition.steps
        },
}
```

Листинг 3 – Фрагмент кода расчетов

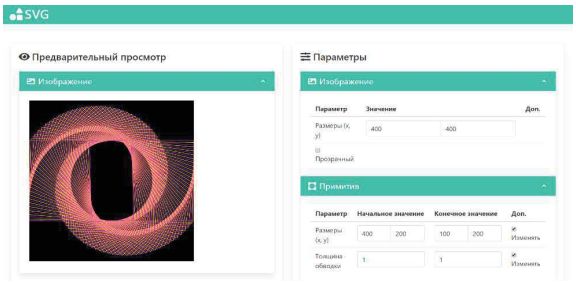

Рисунок 2 – Пример 1 генерации изображения

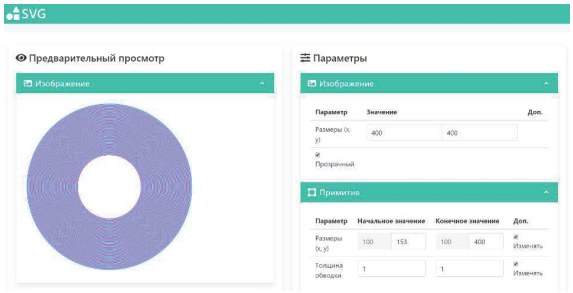

Рисунок 3 – Пример 2 генерации изображения

Реализована возможность сохранить сгенерированное изображение в формате SVG и возможность посмотреть исходный код.