УДК 004.4

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

ПРОГРАММНОЕ СРЕДСТВО ДЛЯ ГЕНЕРАЦИИ ВЕКТОРНЫХ ЗАЩИТНЫХ ИЗОБРАЖЕНИЙ

Векторные защитные изображения могут быть сформированы из простейших геометрических фигур путем таких преобразований, как поворот, смещение, масштабирование. Программное приложение для генерации векторных защитных элементов методом автотипного синтеза цвета разрабатывалось, используя JavaScript-фреймворк Vue.js. Автотипный синтез цвета – получение оттенков цвета на оттиске путем совмещения растровых или штриховых изображений, отпечатанных красками разных цветов, например, триадными красками: голубой, пурпурной и желтой (CMY). В разрабатываемом программном продукте красный цвет получается путем чередования желтого и пурпурного цветов. Аналогичным образом получается зеленый цвет – чередование желтого и голубого и синий – чередование пурпурного и голубого.

Программное приложение состоит из нескольких классов. Первым из них является класс Image, который хранит параметры изображения, а именно размеры по вертикали и горизонтали (this.dimensions = {x:400, y:400} – по умолчанию 400x400px), а также цвет фона (прозрачный, сплошной цвет, линейный или радиальный градиент). Класс Shape хранит информацию о выбранном примитиве и его параметрах: dimensions – размеры по горизонтали и вертикали, transform translate – смещение по горизонтали и вертикали, transform scale – масштабирование по горизонтали и вертикали, transform rotate – поворот, strokeWidth – толщина обводки. Класс FinalShape, который содержит параметры результирующего примитива, наследуется от класса примитива Shape (class FinalShape extends Shape). Класс Transition хранит данные о переходе между примитивами. Содержит shape – тип примитива, customPoints – хранит координаты точек, заданных пользователем, если был выбран произвольный примитив, steps – количество шагов перехода, pivot – опорная точка, относительно которой поворачиваются и масштабируются примитивы. Класс UI хранит информацию об элементах интерфейса.

Внешний вид блока для ввода параметров изображения представлен на рисунке 1.

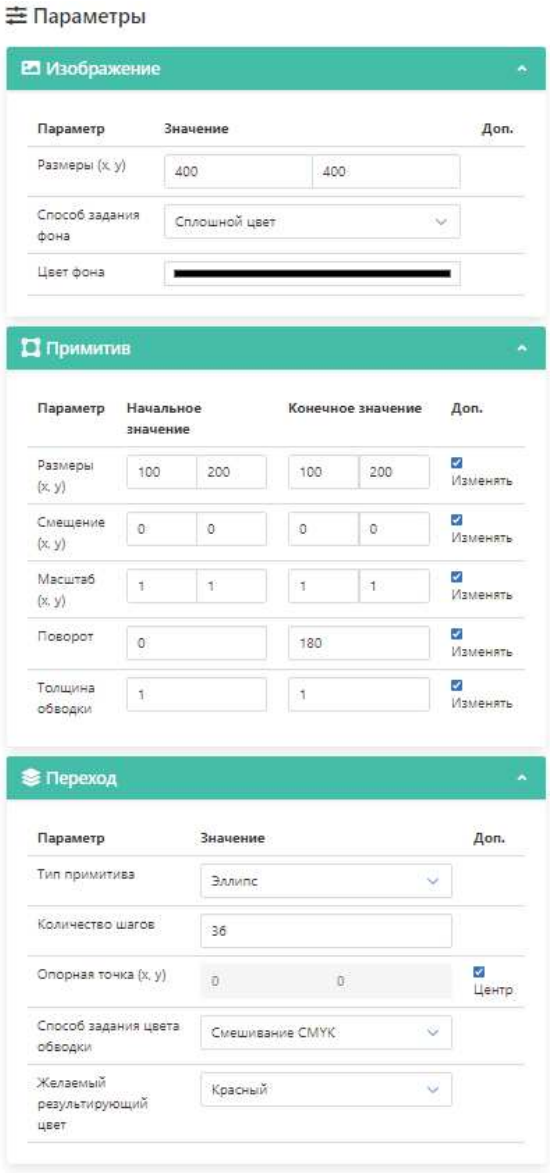

Рисунок 1 – Внешний вид блока для ввода параметров

Были реализованы следующие вычисляемые свойства:

1) center() вычисляет координаты центра изображения;

2) widthDelta() / heightDelta() вычисляет шаг изменения ширины примитива / высоты примитива;

3) translationXDelta() / translationYDelta() вычисляет шаг изменения смещения примитива по горизонтали / по вертикали;

4) scaleXDelta() / scaleYDelta() вычисляет шаг масштабирования примитива по горизонтали / по вертикали;

5) rotationDelta() вычисляет шаг поворота примитива;

6) strokeWidthDelta() вычисляет шаг изменения толщины обводки.

После расчета шага изменения параметров примитива необходимо рассчитать параметры примитива на i-том шаге перехода. Для этого были реализованы следующие методы:

1) widthI(i) / heightI(i) для вычисления ширины /высоты примитива на i-том шаге перехода.

2) translationXI(i) / translationYI(i) для вычисления смещения примитива по горизонтали / вертикали на i-том шаге перехода.

3) scaleXI(i) / scaleYI(i) для масштабирования примитива по горизонтали / вертикали на i-том шаге перехода.

4) rotationI(i) для поворота примитива на i-том шаге перехода.

5) addGradientStop() / removeGradientStop(i) для создания градиента.

6) strokeWidthI(i) для расчета толщины обводки на i-том шаге перехода.

Генерация изображения начинается с ввода размера холста, можно выбрать сплошной фоновый цвет, сделать фон прозрачным либо выбрать линейный / радиальный градиент. Основой алгоритма является выбор базового примитива: круг, эллипс, прямоугольник, квадрат, треугольник, линия либо произвольный примитив. При выборе произвольного примитива необходимо ввести координаты точек для его построения. Далее алгоритм предполагает ввод результирующего цвета. При выборе красного цвета четные примитивы будут иметь желтую обводку, а нечетные – пурпурную, при выборе зеленого цвета четные примитивы будут иметь голубую обводку, а нечетные – желтую, при выборе синего цвета четные примитивы будут иметь голубую обводку, а нечетные – пурпурную. Далее необходимо ввести параметры начального и конечного размера для выбранного базового примитива, а также масштаб, поворот и толщину обводки. По умолчанию в качестве опорной точки выступает центр, но при желании опорную точку можно изменить, введя параметры x и y.

Следующим этапом задается количество шагов. Если данного количество достаточно для формирования защитного изображения, то происходит расчет i-ого шага, изменения выбранного примитива на i-ом шаге и завершение генерации элемента. Изображение можно сохранить в формате SVG либо EPS и при необходимости модифицировать его в любом графическом редакторе.

Если же количества шагов недостаточно, то необходимо ввести новое значение, при этом происходит перерасчет изменения примитива и генерация нового изображения.

Для сохранения полученного изображения в формате EPS использовался язык описания страниц PostScript. Для этого были напи-

саны методы hexToCmyk(hex) для преобразования шестнадцатеричной строки RGB в CMYK и getEpsSource() для формирования содержимого EPS-файла.

Далее представлен пример генерации защитного изображений с использованием базового примитива – прямоугольник.

Для этого выли выбраны следующие параметры:

1) тип примитива: прямоугольник;

2) количество шагов: 250;

3) желаемый результирующий цвет: красный;

4) размеры (x, y): начальное значение (250;300), конечное значение (50;100);

5) смещение (x, y): начальное значение (0; 0), конечное значение (0; 0);

6) масштаб (x, y): начальное значение (1; 1), конечное значение (1; 1);

7) поворот: начальное значение (0), конечное значение (180);

8) толщина обводки: начальное значение (3), конечное значение (0);

9) цвет фона: прозрачный.

Вариант векторного защитного изображения представлен на рисунке 2.

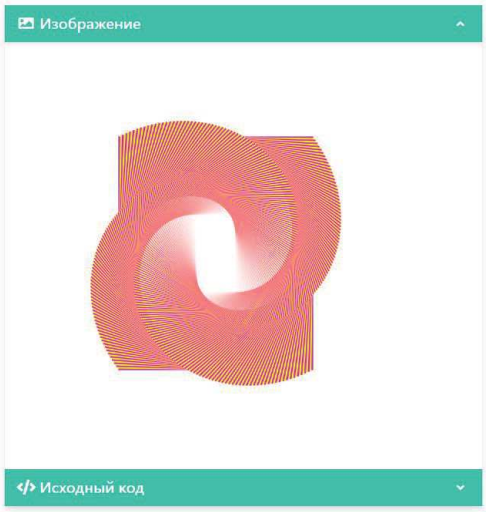

Рисунок 2 – Пример векторного защитного изображения

Таким образом, используя различные комбинации параметров, можно создавать векторные защитные элементы.