ООП

Подход реализуемый в данном примере не предполагает изменений базовых возможностей языка, и по своей семантике он более похож на классические методы ООП. В нём реализованы такие принципы ООП как инкапсуляция и наследование. Данные находящиеся в локальной области видимости, по средствам анонимного замыкания, защищены как от неконтролируемого изменения, так и от добавления новых данных, в отличие от стандартных объектов JS, в которые вносить изменения и добавлять свойства и методы можно произвольно, что несёт за собой известные риски.

Безопасный объект и наследование.

В данном примере реализован объект в котором хранятся свойства и методы необходимые для работы с html элементом canvas. Реализованы всего два метода: инициализация контекста и метод для отрисовки прямоугольника с заливкой. В методы можно передавать параметры, если они не заданы, то есть свойства по умолчанию. Далее этот объект присваивается переменной, в нашем случае gr и в коде реализующем пример работы с данным объектом через точку идёт обращение, непосредственно к методам объекта.

Сцепленный вызов.

Необязательной и не очевидной особенностью данной конструкции является, так называемый сцепленный вызов методов, при котором после вызова метода можно сразу, через точку вызывать следующий метод. Это возможно так как в конце все методы возвращают this, который в нашем случае ссылается на объект gr со всеми его методами. Слабым местом данного подхода является то обстоятельство, что в случае возникновения ошибки в одном месте цепочки вызовов не выполнится все последующие методы. Тем не менее именно такой подход реализован в авторитетной и проверенной годами библиотеке jQuery.

Код самого объекта выглядит следующим образом

```
//Graph.js
//конструктор объекта
    var gr = (function(){
        var myArg, canvas, ctx;
        //свойства по умолчанию, если не передал пользователь
            var DEFAULTS = {
                    height: '200px',
                    width: '300px',
                    colorRect: '#b9b9c8',
                    colorline: '#535362',
                    x: 0,
                    y: 0,
                    w: 0,
                    h: 0
                }
        return {
        //инициализация холста по обязательному id = 'canvas'
            init: function(){
            myArg = arguments[0] || '';
                canvas = document.getElementById('canvas');
                ctx = canvas.getContext('2d');
                canvas.setAttribute('height', myArg.height || DEFAULTS.height);
                canvas.setAttribute('width', myArg.width || DEFAULTS.width);
                return this
            },,
      //функция отрисовки прямоугольника с заливкой
          fRect: function(){
              myArg = arguments[0] || '';
              ctx.fillStyle = myArg.colorRect || DEFAULTS.colorRect;
              ctx.fillRect(myArg.x || DEFAULTS.x, myArg.y || DEFAULTS.y, myArg.w || DEFAULTS.w, myArg.h ||
              DEFAULTS.h);
              return this}
              }
  })()
```

Для упрощения в этой конструкции предполагается наличие на странице только одного элемента canvas со стандартизированным атрибутом id = 'canvas'. Нужно учесть, что на странице может быть больше одного элемента, но реализовать такую возможность несложно.

Салфетка Серпинского и игра Хаос.

Для демонстрации работы этого подхода реализован алгоритм построения салфетки Серпинского, с помощью игры Хаос. Этот изящный алгоритм является простым в реализации, но даёт эффектный результат. Для работы алгоритма задаются 3 вершины треугольника и одна случайная точка, которая может находится где угодно на холсте. Затем случайным образом выбирается вершина и посередине между выбранной вершиной и случайной точкой рисуется новая точка, это будет новая внешняя точка для следующего шага алгоритма. На следующем шаге снова случайным образом выбирается вершина и рисуется точка между вершиной и внешней точкой. Так алгоритм повторяется заданное количество раз. Множество внешних точек образуют фигуру эмитирующую фрактальный объект, который называется салфетка Серпинского.

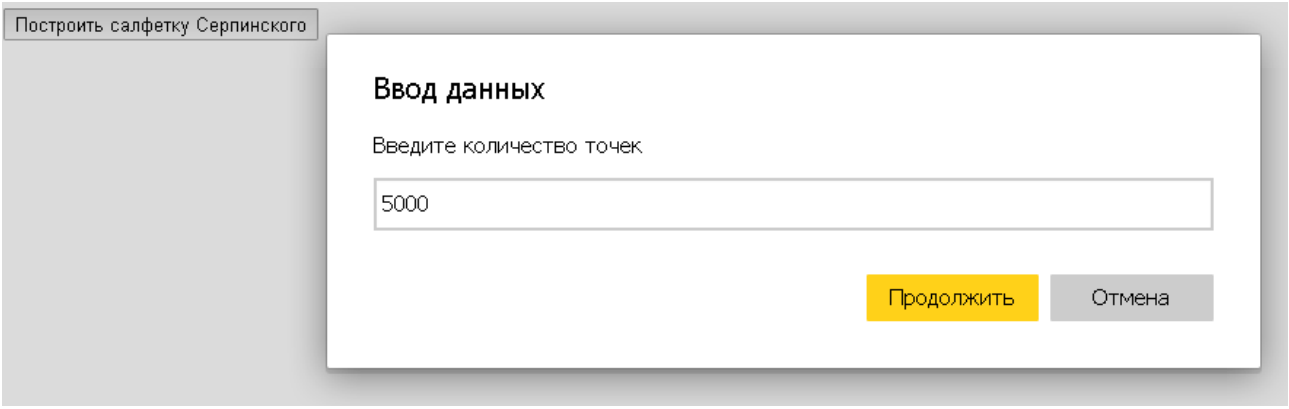
Код самого алгоритма

```
//ss.js моделирование салфетки Серпинского
var btn = document.getElementById('serp');
btn.onclick = function(){
    gr.init({width: 600, height:600})
    var A, B, C//вершины правильного треугольника
    A = {x:10, y: 10, w:3, h:3, colorRect: 'red'};
    B = {x:510, y: 10, w:3, h:3, colorRect: 'red'};
    C = {x:255, y:260, w:3, h:3, colorRect: 'red'};
    var n; //количество точек;

     n = prompt('Введите количество точек', 5000);
    //генератор случайных целых
    function getRandomInt(max){
        return Math.floor(Math.random() * Math.floor(max));
    }
    var Y = {}//случайная точка для начала алгоритма
    Y.x = getRandomInt(599);
    Y.y = getRandomInt(599);
    Y.w = 1;
    Y.h = 1;
    Y.colorRect = 'red'
    function Line(){
        gr.fRect(Y);//создадим случайную начальную точку
        //запустим цикл отрисовки салфетки
        for(var  i = 0; i < n; i++){

         function vertex(){

          var vert;
            //генерируем номер вершины
             var numV = getRandomInt(3) + 1//добавляем 1 чтобы нумерация была от 1 до 3, а не от 0 до 2;
             switch (numV){
                 case 1: vert = A;
                         break;
                 case 2: vert = B;
                         break;
                 case 3: vert = C;
                         break;
             }
             return vert;
         }
         var V = vertex();//текущая вершина
         //функция расчёта новых координат точки Y и её отрисовка
         function tmp(Y, V){
             this.Y = Y;
             this.V = V;
             Y.x = (Y.x + V.x)/2;
             Y.y = (Y.y + V.y)/2
             return Y;
         }
        Y = tmp(Y, V);
        gr.fRect(Y);
        }
    }
    Line();
```

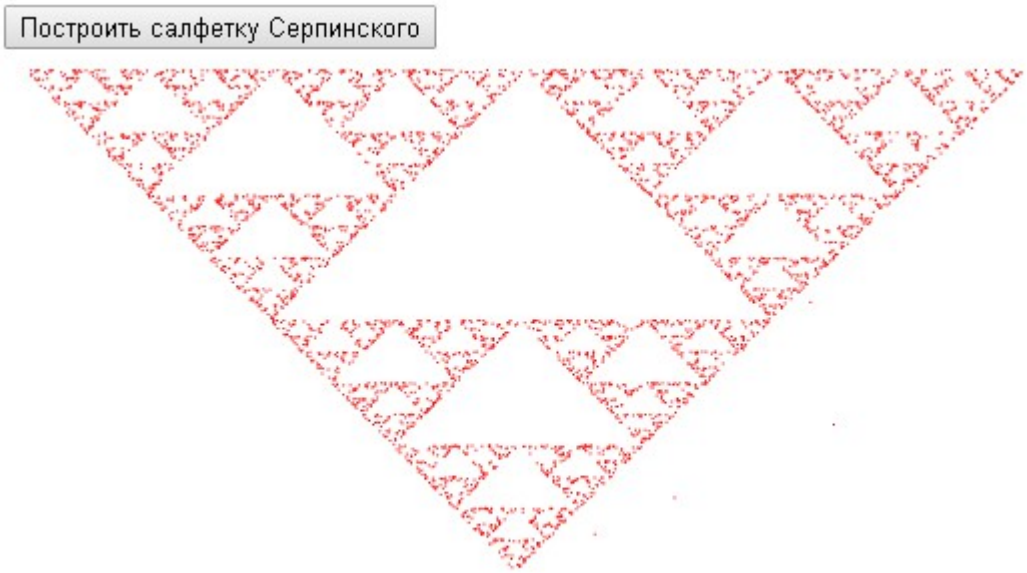
HTML реализация

```
<!doctype html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Graph</title>

</head>
<body>
<input type="button" value = 'Построить салфетку Серпинского' id = 'serp'> <br>
<canvas id="canvas"></canvas>
<script src="Graph.js"></script>
<script src="ss.js"></script>

</body>
</html>
```

Виды в браузере



Салфетка при 5000 точках



Салфетка при 50000 точках