

УДК 004.021

РАЗРАБОТКА АЛГОРИТМА ДИНАМИЧЕСКОГО РИСОВАНИЯ ФИГУР НА CANVAS

Самокиш В.В., студент, кафедра МСТ ХНУРЭ
Егорова И.Н., к.т.н. профессор, кафедра МСТ ХНУРЭ

***Аннотация.** Сформулировано несколько подходов к динамическому рисованию фигур на canvas. На примере программы для рисования овалов был рассмотрен общий алгоритм динамического рисования фигур на холсте.*

***Ключевые слова:** АЛГОРИТМ, CANVAS, JAVASCRIPT, ОВАЛ, ФИГУРА*

Новый элемент HTML5 Canvas (холст) используется для рисования графики с применением скриптов (преимущественно на языке JavaScript).

Сферы использования холста простираются от визуализации результатов статистических данных (создание диаграмм) и математических функций (графики) до создания анимации и таких web-приложений, как игры и графические редакторы.

Существует большое количество библиотек для работы с холстом. К примеру, Paper.js, Processing.js, Raphael.js, KineticJS, LibCanvas, CanvasPlus и Artisan JS. Существуют библиотеки общего назначения (Paper.js), и библиотеки, ориентированные на определенные сферы использования – анимацию (cakejs), создание графиков (Smoothie Charts) и др. Трудность использования библиотек заключается в том, что помимо знания языка JavaScript, необходимо обучиться работе с каждой библиотекой.

Целью работы является исследование и разработка алгоритма динамического рисования фигур на Canvas.

Основной проблемой динамического рисования фигур на холсте является тот факт, что холст не является запоминающей поверхностью и фигуры, нарисованные на нем, превращаются в массив цветных пикселей. Другими словами, мы не можем обратиться к нарисованным фигурам как к отдельным объектам. Для этого необходимо создать массив с нарисованными на холсте объектами и его параметрами [1]. Такой подход в данной работе не анализируется.

В работе рассмотрен алгоритм динамического рисования фигур, предполагающий решение двух основных задач:

– во-первых, фиксация момента начала процесса рисования. С этой целью вводим переменную isDrawing, которая и определяет, может ли пользователь рисовать в данный момент;

– во-вторых, динамическое изменение размеров фигур в процессе рисования в соответствии с положением курсора мыши. Одним из вариантов является использование двух холстов, один из которых будет постоянно обновляться, отображая текущие размеры фигуры, а другой – служить для фиксации конечных

результатов. Однако, такой подход не является рациональным. Другим вариантом является использование одного холста и хранение конечных результатов рисования в файле изображения. В этом случае при перемещении мыши холст постоянно обновляется и на нем прорисовывается предыдущее сохраненное состояние холста и фигура с соответствующими параметрами.

Для создания инструмента рисования «перо» нет необходимости обновлять холст в процессе рисования, а достаточно лишь знать в какой момент может происходить рисование.

Необходимо также определиться с моментом завершения процесса рисования, другими словами, определить будет ли продолжаться рисование на холсте, если пользователь вышел за его пределы. От этого зависит, для каких объектов должны определяться обработчики событий. Одним из вариантов реализации такого сценария является подключение событий рисования к холсту и завершение процесса рисования при возникновении события `onmouseout` или `onmouseup`. В этом случае рисование начинается при нажатии мыши на холсте и завершается при поднятии или выходе мыши за его пределы. Второй вариант определения момента завершения процесса рисования может быть реализован посредством подключения одних событий к холсту, а других – к объекту `window`. Как и в предыдущем случае, рисование начинается при нажатии мыши на холсте, но завершается только при поднятии клавиши мыши. В этом случае при выходе пользователя за пределы рабочей области (холст) или даже окна, рисование продолжается. Сложно сказать, какой из подходов является наиболее оптимальным. Выбор того или иного подхода зависит от поставленной задачи и личных предпочтений. К примеру, в таком программном продукте, как Paint используется второй подход – рисование не прекращается при выходе за рабочую область. Далее будет рассматриваться именно такой подход.

Процесс динамического рисования фигур целесообразно рассмотреть на примере программы для рисования овалов. HTML-разметка (рис. 1) данной программы состоит всего из двух строк и включает: элемент `canvas`, на котором непосредственно происходит рисование, и элемент `img`, в котором хранится состояние канвы

```
<canvas id='example' width='600' height='400'> </canvas>
<img id='imageCopy' src='images/empty.png'>
```

Рисунок 1 – HTML-разметка холста

При загрузке страницы код получает объект холста, контекст рисования и объект `img`. Тут же задаем значение переменной `isDrawing`, чтобы исключить возможность возникновения ошибок, и подключаем требуемые для рисования события и их обработчики (рис. 2). Данная переменная информирует остальной код программы, можно ли работать с контекстом рисования. Процесс рисования начинается в момент клика по холсту [2].

```

window.onload = function() {
    canvas = document.getElementById("example");
    context = canvas.getContext("2d");
    img = document.getElementById('imageCopy');
    isDrawing = false;
    canvas.onmousedown = startDrawing;
    window.onmousemove = draw;
    window.onmouseup = stopDrawing;
}

```

Рисунок 2 – Javascript код

Рассмотрим последовательно действия каждого из обработчиков событий. Так, функция startDrawing() вызывается событием onmousedown. Эта функция изменяет значение переменной isDrawing и фиксирует координаты курсора в момент клика мыши (рис. 3).

```

function startDrawing(e) {
    if (e.which==1) isDrawing = true;
    x = e.pageX - canvas.offsetLeft;
    y = e.pageY - canvas.offsetTop;
}

```

Рисунок 3 – Javascript код функции startDrawing

Функция draw() вызывается событием onmousemove(). Эта функция непосредственно отвечает за процесс рисования. Если в данный момент может происходить рисование (переменная isDrawing равна true), то очищаем контекст, прорисовываем предыдущее сохраненное состояние контекста рисования, определяем текущие координаты положения мыши и прорисовываем фигуру, в данном случае овал (рис. 4).

```

function draw(e) {
if (isDrawing == true) {
context.clearRect(0, 0, canvas.width, canvas.height)
context.drawImage(img, 0, 0)
current_coords(e);
draw_ellipse()
}
}

```

Рисунок 4 – Javascript код функции draw

Функция current_coords(e) вычисляет координаты точки, в которой в данный момент находится указатель мыши (рис. 5).

```

function current_coords(e){
x2 = e.pageX - canvas.offsetLeft;
y2 = e.pageY - canvas.offsetTop;
}

```

Рисунок 5 – Javascript код функции current_coords

Для динамического рисования овалов определим две функции: `current_center_radius()`, которая будет вызываться из функции рисования фигуры, и функцию для рисования `draw_ellipse()`.

Центр эллипса находится посередине между точкой клика и текущим положением курсора мыши. В зависимости от того, больше текущие координаты, чем начальные или меньше, центр определяется соответственно вычитанием или сложением начальных координат и значений проекций радиуса по горизонтали и вертикали. Радиус рассчитывается как половина расстояния между текущими и начальными координатами. Порядок вычитания задает положительное число на выходе (рис. 6).

```
function current_center_radius(){
  if (x2 > x || x2 == x)
    {R_x = (x2 - x)/2; centerX= R_x + x}
  if (x2 < x)
    {R_x = (x - x2)/2; centerX= x - R_x}
  if (y2 > y || y2 == y)
    {R_y = (y2 - y)/2; centerY= R_y + y}
  if (y2 < y)
    {R_y = (y - y2)/2; centerY= y - R_y}
}
```

Рисунок 6 – Javascript код

Функция `draw_ellipse()` вызывает функцию `current_center_radius()` и сохраняет текущее состояние контекста для того, чтобы не происходило искажение обводки и во избежание влияния уже реализованных трансформаций на последующие.

Затем происходит уточнение значений радиуса, координат центра окружности и коэффициента масштабирования.

Из проекций радиуса выбираем максимальное значение, которое и будет радиусом трансформируемой окружности. Ось с максимальным значением искажаться не будет, и коэффициент масштабирования по ней равен 1. В то время как другая ось, с минимальным значением, будет поддаваться масштабированию. Значение коэффициента масштабирования находится как отношение меньшей проекции к большей.

Следует заметить, что функция `draw_ellipse()` предоставляет искаженные координаты центра. Для того, чтобы получить исходные координаты центра до масштабирования необходимо учесть искажения после применения трансформации.

После уточнения параметров функция масштабирует контекст, начинает новый путь, переносит координаты в центр окружности, определяет окружность, восстанавливает состояние контекста и прорисовывает созданный путь (рис. 7).

Завершает процесс рисования фигуры функция `stopDrawing()`, вызываемая событием `onmouseup`. Эта функция сохраняет содержимое холста в файле изображения и устанавливает значение переменной `isDrawing` равной `false`, чтобы исключить рисование фигур при движении мыши до возникновения события `onmousedown` (рис. 8).

```
function draw_ellipse() {
current_center_radius()
context.save();
if (R_x > R_y || R_x == R_y) {
    R=R_x;
    scale_y=R_y/R_x;
    scale_x=1;
    if(scale_y!=0) centerY= centerY/scale_y
    context.scale(1, scale_y)
}
if (R_x < R_y) {
    R=R_y;
    scale_x=R_x/R_y;
    scale_y=1;
    if(scale_x!=0) centerX= centerX/scale_x
    context.scale(scale_x, 1)
}

context.beginPath();
context.translate(centerX, centerY)
if (scale_y!=0 && scale_x!=0) context.arc(0, 0, R, 0, 2*Math.PI);
context.restore();
context.stroke();
}
```

Рисунок 7 – Javascript код функции draw_ellipse

```
function stopDrawing() {
img.src=canvas.toDataURL();
isDrawing = false;
}
```

Рисунок 8 – Javascript код функция stopDrawing

Таким образом, на примере программы для рисования овалов был рассмотрен общий алгоритм динамического рисования фигур на холсте.

Литература.

1. Ерохин, А. Интерактивные фигуры / А. Ерохин. – Режим доступа: www / URL: http://professorweb.ru/my/html/html5/level4/4_7.php – 25.03.2016. – Загл. с экрана.
2. Ерохин, А. Простая программа рисования / А. Ерохин. – Режим доступа: www / URL: http://professorweb.ru/my/html/html5/level4/4_3.php – 27.01.2016. – Загл. с экрана.