УДК 004.92
DOI https://doi.org/10.32838/2663-5941/2021.1-1/12

Качурівський В.О.
Відокремлений підрозділ Національного університету біоресурсів
і природокористування України «Бережанський агротехнічний інститут»

Качурівська Г.М.
Відокремлений підрозділ Національного університету біоресурсів
і природокористування України «Бережанський агротехнічний інститут»

ПОБУДОВА АНІМАЦІЙНИХ ДІАГРАМ ЗАСОБАМИ CANVAS API

Одним зі способів візуалізації числової інформації є використання діаграм та графіків. Діаграми за допомогою геометричних фігур представляють образи числових даних. Графіки подають розвиток певного процесу залежно від певного фактора. Будь-який графік чи діаграма – це графічний образ математичної моделі. Вебтехнології володіють засобами роботи з графічною інформацією. Поряд із вставкою графічних файлів у текстові матеріали все частіше використовується програмована графіка. Програмована графіка визначається SVG-зображеннями та побудовами за допомогою полотна <canvas>. Побудова діаграм на полотні здійснюється засобами CANVAS API. Програмований Javascript-сценарій здійснює побудову діаграми або графіка та створює в документі статичне графічне зображення. Унаочнення динаміки розвитку процесу потребує побудови діаграми, яка триває в часі. Для цього необхідно застосовувати анімацію елементів діаграми або графіка. Стандартні засоби трансформації та переходу CSS можуть бути використані для створення ефектів появи зображення на полотні. Вони не відображають динаміки побудови самої діаграми. Використання анімаційних GIF-зображень є надто об'ємним. Тому є необхідним розгляд питання про те, яким чином провести анімацію, кадрування діаграми. Одним зі способів анімації діаграми або графіка є використання методу setTimeout, що належить об'єкту WindowOrWorkerGlobalScope. Цей метод установлює таймер, що виконує функцію один раз, щойно спливе заданий час. Розбиття діаграми на складники та почерговий запуск побудови елемента діаграми через певний час створить динаміку та відповідну анімацію діаграми. У статті визначено Javascript-сценарії динамічної побудови стовпчикової та секторної діаграм. Описано програмування анімаційних ефектів: побудова за елементами діаграми, одночасна побудова всіх елементів.

Ключові слова: *анімація діаграми, секторна діаграма, комп'ютерна графіка, CANVAS API, програмована анімація.*

Постановка проблеми. Одним зі способів візуалізації числової інформації є використання діаграм та графіків. Будь-який графік чи діаграма – це графічний образ математичної моделі. Унаочнення динаміки розвиту процесу потребує побудови діаграми, яка триває у часі. Така анімація дозволяє спостерігати за розвитком процесу або явища у часі. Застосування GIF-зображень не є ефективним, оскільки вони є достатньо об'ємними та не дозволяють коригувати вхідні дані. Цю проблему можна розв'язати за допомогою програмованої графіки засобами CANVAS API.

Аналіз останніх досліджень і публікацій. У публікаціях Steve Fulton та Jeff Fulton висвітлено роботу з побудови графічних примітивів, а також розглянуто загальні принципи анімації зображення [2; 3]. Джош Маріначі розглядає методи побудови діаграм [4]. У роботі авторів розглянуто питання адаптивної побудови діаграми до ширини контейнера публікації та способи відбору числових даних до Javascript-сценарію побудови інфографіки [6; 7]. Також на програмному ринку присутні бібліотеки для побудови інтерактивних діаграм. Серед таких бібліотек слід виділити Chartist JS, C3 JS, DC JS та Google Chart. Натепер відсутні розроблення, що стосуються анімації побудови самої діаграми.

Постановка завдання. Визначити методи програмної реалізації для створення анімаційного ефекту побудови стовпчикової гістограми та секторної діаграми. Розробити javascript-сценарій для побудови анімаційного зростання стовпчиків гістограми та секторів кругової діаграми засобами CANVAS API. Розробити загальний алгоритм створення анімації елементів діаграм для створення ефектної інфографіки.

Виклад основного матеріалу дослідження. У вебдизайні графіки є одним із кращих інстру-

ментів для візуалізації даних. Застосування ефекту анімації до інфографіки затримує увагу користувача на ній та створює позитивний момент під час перегляду текстової інформації. Під час побудови діаграм засобами CANVAS API доцільно розробити способи програмної анімації елементів діаграм.

Припустимо, що діаграма складається з чотирьох елементів. Визначимо такі сценарії анімації діаграм.

Перший спосіб – почергова побудова елементів діаграми. До кожного елемента діаграми застосуємо спосіб збільшення його розміру від 0 (нуля) до заданого числового значення. Спочатку будується перший елемент, за ним другий, третій та четвертий. Час побудови одного елемента діаграми залежить від фактичного числового значення, на основі якого будується елемент. Чим більше числове значення, тим довше будується елемент.

Другий спосіб – одночасна побудова елементів діаграми. Усі елементи будуються одночасно від нульового до відповідного числового значення масиву вхідних даних. Побудова закінчується тоді, коли вичерпано всі значення. Найдовше будується елемент із максимальним числовим значенням.

Розглянемо детальніше перший спосіб (почергова побудова елементів діаграми).

Алгоритм для створення анімації такий:

1) визначити, який складник діаграми буде програмно анімуватися;

2) сформувати числовий масив показників, на основі яких будується діаграма. Про способи відбору числових значень до діаграми детальніше описано у праці [6]. Провести масштабування числових значень відповідно до висоти полотна побудови canvas. Масштаб визначається коефіцієнтом k, як співвідношення висоти полотна H до максимального числового значення;

3) провести конструювання функції побудови статичного елемента діаграми. У параметрах функції необхідно передбачити передання параметрів зміни розміру елемента;

4) здійснити циклічне застосування методу setTimeout() для створення кадрування елемента діаграми та часову затримку в його відображенні. Це створить ефект анімації нашої інфографіки.

Цей алгоритм є універсальним та може бути застосований для довільного типу діаграми чи графіка.

Розглянемо реалізацію цього алгоритму для побудови вертикальної гістограми. Елементами гістограми є прямокутники, які будуть створюватися командою ctx.fillRect(x,y,w,h). Необхідно провести програмування ***анімації одного стовпчика гістограми***, який буде основою для всієї інфографіки. Для прикладу візьмемо полотно canvas розміру 300px на 150px. Висоту полотна зафіксуємо у змінній H=150. Числове значення стовпчика гістограми записуємо у масив, наприклад: var y=[300]. Коефіцієнт масштабування k буде дорівнювати k=H/300=0,5. Формуємо новий масштабований масив значень yy[0]=y[0]*k.

Здійснюємо програмування функції користувача plot для побудови відображення прямокутника:

```
function plot(xx, hh) {
    ctx.fillStyle = '#67828E';
    ctx.fillRect(xx,H-hh,ww,hh);}
```

де xx та hh – формальні параметри початкової точки на осі X та висота прямокутника відповідно.

Здійснюємо побудову стовпчика зі зростання висоти h прямокутника від 1px до масштабованого раніше значення yy[0] з кроком 1px.

Для часової затримки у побудові наступного прямокутника застосуємо метод setTimeout()[4]. Числове значення часової затримки для побудови наступного прямокутника формуємо у змінній time. Змінну часу побудови збільшуємо на 10мс лічильником time+=10.

Фрагмент програмного коду для анімаційної побудови стовпчика гістограми є таким:

```
var time=0;
for(h=1;h<=yy[0];h++)
    {time+=10;setTimeout(plot,time,x,h);}
```

Параметрами методу setTimeout() є такі: функція побудови прямокутника plot, змінна time та параметри для побудови x – координата на осі X та h – висота чергового прямокутника, якій передаються функції plot. Оскільки значення координати x є незмінною величиною, то прямокутники будуються таким чином, що кожен наступний перекриває попередній. Таким чином, створюється анімаційний ефект плавного зростання прямокутника.

Із результатом роботи JavaScript-сценарію анімації прямокутника детальніше можна ознайомитися за вказівкою https://codepen.io/volodimir-kachurwskij/pen/qBOyEWN.

Тепер розглянемо програмну реалізацію ***почергової побудови елементів гістограми***.

Припустимо, що наша гістограма повинна мати чотири стовпчики, які представляють такі числові дані: 200, 180, 100, 150. Формуємо масив вхідних даних:

var y = [200, 180, 100, 150]; var n = y.length;

де n – кількість числових значень масиву.

Побудову елементів гістограми будемо здійснювати поступово від першого стовпчика до останнього за допомогою циклу. Для створення анімаційної побудови застосуємо змінну time, у якій будемо формувати час запізнення побудови наступного кадру анімації. Після побудови відповідного кадру збільшимо значення time на 5 мілісекунд. Дана змінна буде накопичувати числове значення запізнення для побудови другого, третього та четвертого стовпчиків.

Фрагмент коду почергової побудови стовпчиків гістограми з анімаційним ефектом зростання висоти кожного прямокутника буде таким.

```
var time = 0;
for(j=0; j<n;j++)
    for (h = 1; h<=yy[j]; h++) {
    time+=5;setTimeout(plot, time, x, h, j);}
```

Із результатом роботи JavaScript-сценарію почергової побудови стовпчиків гістограми з анімаційним ефектом зростання висоти прямокутника детальніше можна ознайомитися за вказівкою https://codepen.io/volodimir-kachurwskij/pen/oNjPGGd

Розглянемо другий спосіб анімації діаграми: одночасна побудова елементів діаграми.

Для конкретизації JavaScript-сценарію розглянемо побудову гістограми на основі чотирьох числових значень.

Алгоритм побудови буде таким:

1) формування числового масиву показників гістограми. Припустимо var y = [200, 180, 100, 150];

2) обчислення коефіцієнта масштабування. k= H/max, де max – найбільше числове значення масиву даних max = Math.max(…);

3) формування масштабованого масиву для побудови діаграми

```
var n=y.length; for(i=0; i<n; i++) yy[i]=y[i]*k;
```

де n=y.length – кількість елементів масиву;

4) зміна функції plot() для побудови чотирьох прямокутників. Побудову прямокутників визначимо у конструкції циклу.

```
function plot(xx, hh) {
for(j=0; j<n; j++)
    if(hh<=yy[j])
    {ctx.fillStyle='#67828E';
    ctx.fillRect(xx+j*70,H-hh,ww,hh);}}
```

Для зміщення побудови чергового прямокутника використаємо конструкцію xx+j*70, де j*70 значення на зміщення. Число 70 – це крок зміщення на осі X;

5) застосування методу setTimeout() для створення кадрування та анімації прямокутників:

```
for (h = 1; h<=max*k; h++ )
{time+=10;setTimeout(plot, time, x, h);}
```

Щодо тривалості анімації. Оскільки висота побудови наступного прямокутника збільшується на одиницю, часова затримка збільшується на 10мс, то час анімації триватиме max*k*10 секунд, що створює приємну та м'яку анімацію.

Із результатом роботи JavaScript-сценарію одночасної побудови чотирьох прямокутників гістограми детальніше можна ознайомитися за вказівкою https://codepen.io/volodimir-kachurwskij/pen/yLYqMBw.

Анімація побудови секторної діаграми.

Програмна анімація секторної діаграми буде залежати від способу анімації:

1) почергова побудова кожного повного сектора діаграми із затримкою визначеного часу;

2) плавна побудова одного сектора діаграми з переходом до побудови кожного наступного сектора.

Побудова секторної діаграми здійснюється за допомогою команди побудови дуги void ctx.arc (x, y, radius, startAngle, endAngle, anticlockwise) [1];

Для побудови секторів діаграми використаємо заливку ctx.fill(); певним кольором ctx.fillStyle = 'color';

Для побудови кожного сектора діаграми необхідно мати початковий та кінцевий кути. Проведемо формування кутів секторів у масиві angel. Для зручності обчислення кутів формуємо питому вагу кожного значення числового масиву, на основі яких необхідно будувати діаграму. Наприклад, необхідно побудувати секторну діаграму, яка буде мати шість секторів, які відповідають таким числовим даним: 14, 9, 7, 20, 28, 34. Знайдемо загальну суму.

```
var znach=[14,9,7,20,28,34];
var s=0; znach.forEach((item)=>{s+=item});
```

Для зручності обчислень – кути розраховуємо в діапазоні від 0 до 1. Перший кут рівний 0, кінцевий – 1. Кожен наступний кут побудови визначається як попередній, до якого додано величину питомої ваги числового значення.

```
var n=znach.length; var angel=[];
angel[0]=0; angel[n+1]=1;
for(i=0;i<n;i++){
    angel[i+1]=angel[i]+znach[i]/s;}
```

У результаті сформовано масив питомих значень кутів для побудови секторів діаграми.

Побудову конкретного сектора здійснюємо командою ctx.arc(x,y,r,angel[i]*Math.PI*2, angel[i+1]*Math.PI*2,false); де i – це індекс сектора побудови. Побудова секторів діаграми здійснюється у циклі. Попередньо сформуємо кольорову заливку секторів у масиві color_s.

```
var color_s=['#FF7C00', '#BF7630', '#A65100', '#FF9D40','#FFB773','#8F4600'];
```

Функція plot_sector () побудови одного сектора діаграми є такою:

```
function plot_sector(ii) {
ctx.beginPath();
ctx.fillStyle = color_s[ii];
ctx.moveTo(200,200); // центр дуги
ctx.arc(200,200,180,angel[ii]*Math.PI*2, angel[ii+1]*Math.PI*2,false);
ctx.fill();}
```

де ii – це формальний параметр, який визначає номер сектора побудови.

Для анімації елементів діаграми застосуємо метод setTimeout(), до якого передаємо значення визначеного номера сектора та час затримки для побудови наступного. Побудова секторів проводиться циклічно від першого до шостого.

Програмна реалізація почергової побудови кожного повного сектора діаграми із затримкою визначеного часу буде такою:

```
var time=0;
    for (i= 0; i<=n; i++ ){
      time+=200;
      setTimeout(plot_sector, time, i); }
```

Із результатом роботи JavaScript-сценарію детальніше можна ознайомитися за вказівкою https://codepen.io/volodimir-kachurwskij/pen/JjYaZpp.

Більш привабливою буде анімація секторної діаграми з плавною побудовою одного сектора діаграми з переходом до побудови наступного сектора.

Для плавної побудови секторів діаграми будемо здійснювати побудову одиничного сектора величиною в 1 градус тобто π/180 радіан. Колір одиночного сектора буде визначатися поточним кольором доти, доки значення буде у кутових межах відповідного сектора, кути якого задані у масиві angel.

Умова для визначення кольору одиничного сектора є такою:

```
if(aa>=angel[i]*Math.PI*2&&aa<=angel[i+1]*Math.PI*2)
c=i; ctx.fillStyle = color_s[c];
```

де aa – кут одиничного сектора.

Функція побудови сектора буде такою:

```
function plot_sector(aa) {
ctx.beginPath ();
for(i=0;i<=n;i++)
if(aa>=angel[i]*Math.PI*2&&  aa<=angel[i+1]*Math.PI*2)
c=i;
ctx.fillStyle = color_s[c];
ctx.moveTo(cen_x,cen_y);
ctx.arc(cen_x,cen_y,r,aa-Math.PI/180,aa,false);
ctx.fill();}
```

Побудова одиничного сектора здійснюється під час зміни кута побудови від 0 до 360 градусів. Затримка побудови одиничного сектора 5 мілісекунд. Уся побудова триватиме близько двох секунд.

Код побудови кругової діаграми є таким:ц

```
var time=20;
    for (a= 0; a<=Math.PI*2;a+=Math.PI/360){
      time+=5;
      setTimeout(plot_sector,time,a);}
```

Із результатом роботи JavaScript-сценарію детальніше можна ознайомитися за вказівкою https://codepen.io/volodimir-kachurwskij/pen/oNjPaxo.

Описані Javascript-сценарії є валідними та можуть бути застосовані у HTML документах із прив'язкою до певної події під час формування аналітичних матеріалів.

Висновки. Розглянуті питання анімації елементів діаграми для відображення динаміки процесу не вичерпують усіх можливих підходів до розв'язання цієї проблеми. У статті розглянуто процедурний підхід до побудови діаграм. Окремої уваги заслуговує об'єктний підхід до елементів діаграми та відповідного програмування їх анімації.

Також потребують подальшого дослідження анімація звичайного декартового графіка, графіка в полярних та сферичних координатних для візуалізації об'ємних фігур.

Список літератури:

1. Canvas API. MDN web docs mozilla. URL: https://developer.mozilla.org/uk/docs/Web/API/Canvas_API (дата звернення: 11.01.2021)
2. Steve Fulton, Jeff Fulton. HTML5 Canvas. O'Reilly Media, 2011. 654 p.
3. Steve Fulton, Jeff Fulton. HTML5 Canvas: Native Interactivity and Animation for the Web. O'Reilly Media, 2011. 628 p.

4. WindowTimers.setTimeout(). MDN web docs mozilla. https://developer.mozilla.org/uk/docs/Web/API/WindowTimers/setTimeout (дата звернення: 11.01.2021)

5. Джош Мариначи. Практика: создание диаграмм. URL: https://webref.ru/dev/canvasdeepdive/chapter02 (дата звернення: 11.01.2021)

6. Качурівська Г., Качурівський В. Способи відбору даних до Javascript сценарію побудови діаграми. *«Інтернет-освіта-наука-2018*» : збірник праць одинадцята міжнар. наук-практ. конф. (Вінниця, 22–25 травня 2018 р.). Вінниця : ВНТУ, 2018. С. 238–240.

7. Качурівський В.О. Побудова адаптивних та динамічних діаграм засобами CANVAS API. *Вчені записки ТНУ імені В.І. Вернадського. Серія: Технічні науки.* 2018. Т. 29 (68) № 2. С. 132–137.

Kachurisvkii V.O., Kachurivska H.M. **CONSTRUCTION OF ANIMATION DIAGRAMS BY CANVAS API**

One way to visualize numerical information is to use charts and graphs. Diagrams with the help of geometric figures represent images of numerical data. Graphs show the development of a certain process depending on a certain factor. Any graph or diagram is a graphical representation of a mathematical model. Web technologies have the means to work with graphic information. Along with inserting graphic files into text materials, programmable graphics are increasingly used. Programmable graphics are defined by SVG images and constructions using the <canvas> canvas. Construction of diagrams on a cloth is carried out by means of CANVAS API. A programmable Javascript script builds a chart or graph and creates a static graphic image in the document. Visualizing the dynamics of the process requires the construction of a chart that lasts over time. To do this, use animation of chart or graph elements. Standard CSS transformation and transition tools can be used to create effects on the canvas. They do not reflect the dynamics of the diagram itself. Using animated GIFs is too extensive. Therefore, it is necessary to consider how to conduct animation, charting. One way to animate a chart or graph is to use the setTimeout method, which belongs to the WindowOrWorkerGlobalScope object. This method sets a timer that performs the function once as soon as the set time expires. Dividing the diagram into constituent elements and alternately starting the construction of the diagram element after a certain time will create the dynamics and the corresponding animation of the diagram. The article identifies Javascript-scenarios for dynamic construction of bar and pie charts. The programming of animation effects is described: construction on the element of the diagram, simultaneous construction of all elements.

Key words: *chart animation, pie chart, computer graphics, CANVAS API, programmable animation.*