Implementing a visual representation of analytic data using a radar chart

E R Kalyuzhny\textsuperscript{1,2}, A V Poteryaeva\textsuperscript{1,2}, G A Shpakovsky\textsuperscript{1,2}, L V Bykreev\textsuperscript{1,2} and N V Zarikovskaya\textsuperscript{1,2}

\textsuperscript{1} Tomsk State University of Control Systems and Radioelectronics, 40 Lenina Prospect, Tomsk, 634050, Russia
\textsuperscript{2} AlderaSoft LLC, 9, block 1, Mokrushina Street, of. 3007, Tomsk, 634045, Russia

E-mail: kalyuzhnyer@gmail.com

Abstract. This article considers the implementation of one of the ways to visually display analytical data using a radar chart. The implementation of scaling and displaying inscriptions and images by sectors using Bezier curves is described in detail.

1. Introduction

The use of information technology is widely entrenched in all spheres of human and organizational activities. One of the most frequently used areas is data processing, data visualization, analysis, and formation of any analytical reports [1]. Given the capabilities of modern information tools, each company seeks to find the most optimal for an ordinary person to visually easy to perceive this information.

The purpose of this work is to implement a data analysis visualization system in the form of a radar, which is a pie chart. Let's consider the application of the "Radar" diagram to analyze the innovation activities of a large enterprise.

In this case, information about various technological areas of the enterprise (the radar sector) is subject to visualization [2], the investors of which are directly the enterprise itself. The radar consists of the following elements:

- sectors - technological directions of the first, second or third level, the number of which varies from two to ten;
- cells - elements located on the radar in a particular technological direction and at a particular stage of the technology life cycle;
- cell boundaries - stages of the life cycle representing rings on the radar;
- markers - events and news in the form, which are in a particular radar cell.

Figure 1 shows the radar described above.
One of the main requirements of the implemented system is the ability to view detailed information about events and news, as well as filtering information according to the following parameters:

- technical directions;
- time parameters;
- type of events displayed.

2. Technologies used in the implementation

The system is implemented as a client-server application.

The server part is implemented in PHP programming language using Laravel framework.

The client part of the system is implemented using TypeScript programming language, React library and Konva framework. With all the advantages of Konva.js framework, there are many disadvantages of this tool. They are that when the environment is properly configured, elements drawn within the same canvas are converted to svg format during the build process, which leads to some image distortions.

The most frequently identified problem is the drawing of text along an arc, in our case it is the name of the sectors. The problem of "dancing letters" (Fig. 2.a) is revealed during the implementation of this problem.

![Incorrect display](image1.png) ![Correct display](image2.png)

**Figure 2. Displaying sector labels**
To solve this problem, the trajectory of lettering can be set using Bezier curves [5, 6]. Bezier curves are used in computer graphics to draw smooth curves. To draw captions correctly it is necessary to define a curve that would be as close as possible to describing an arc from one point to another on a circle.

To draw a line in svg there are several commands for <path> nodes. The basic command is "Move to", or M (start of path). It takes as parameters the coordinates of the point it moves to.

A cubic Bezier (C) takes two control points. Thus, to create a cubic curve, you need to specify three sets of coordinates, the pattern is shown in formula 1.

\[ C \ x_3, y_3, x_4, y_4, x_2, y_2 \] (1)

The last set of coordinates \((x_2, y_2)\) is the point where the line ends. The other two are control points: \((x_3, y_3)\) the control point for the beginning of the curve, and \((x_4, y_4)\) for the end of the curve. The Bezier function creates a smooth curve that leads from the slope set at the beginning of the line to the slope at the other end.

Thus, the final view of the given trajectory will have the form presented in formula 2.

\[ M \ x_1, y_1 \ C \ x_3, y_3, x_4, y_4, x_2, y_2 \] (2)

As part of the problem, we need to draw a Bezier curve as close as possible to the trajectory of an arc of a circle. From Figure 3 we can conclude that the problem is reduced to finding the coordinates \(x_3, y_3, x_4, y_4\) using the distance L (line).

![Figure 3](image)

**Figure 3.** Graphical representation of the coordinate search for a Bezier curve

The angles between the segments connecting the endpoints and the control points and the radii of the segment are marked in the diagram as straight lines (L). It follows from the condition of collinearity, which must be satisfied for the curve to be smooth, that the segment connecting the end point to the control point will be perpendicular to the radius of the circle.

The approximation is made by the following formula 3.

\[ L = \frac{4}{3} \tan \left( \frac{1}{4} \alpha \right) \] (3)

The further calculation must be translated into coordinate form.

At realization of web systems coordinate axes are located from left to right from top to bottom. When drawing the radar, we know the number of sectors and radius, the coordinates of the center coincide with the beginning of the coordinate system.

The coordinates \((x_1, y_1)\) and \((x_2, y_2)\) are calculated following formulas:

\[ x_1 = 0 \]
\[ y_1 = -R \]
\[ x_2 = R \times \cos \alpha \]
\[ y_2 = -R \times \sin \alpha \]
The angle of each sector is calculated following formula:
\[ a = \frac{360}{n}, \]
where \( n \) is the number of sectors, \( R \) is the radius of the circle.
Calculation of coordinates \((x_3, y_3)\) and \((x_4, y_4)\) is made following formulas:
\[
\begin{align*}
x_3 &= x_1 - k \cdot y_1 \\
y_3 &= y_1 + k \cdot x_1 \\
x_4 &= x_2 + k \cdot y_2 \\
y_4 &= y_2 - k \cdot x_2,
\end{align*}
\]
where:
\[
\begin{align*}
k &= \frac{4 \cdot \sqrt{2} \cdot q_1 \cdot q_2 - q_2}{3 \cdot x_1 \cdot y_2 - y_1 \cdot x_2} \\
q_1 &= x_1^2 + y_1^2 \\
q_2 &= q_1 + x_1 \cdot x_2 + y_1 \cdot y_2.
\end{align*}
\]
Thus, for the Bezier curve (2) all unknowns are found.
This solution corrects the defect of the used framework, the result of the application is shown in Figure 2.b.
In the implementation process, the most important problem besides the problem described above was the problem of multiple scaling.

3. Mathematical model of scaling
There are three common ways of interacting with the zoom object:
- the user zooms using only the interface buttons;
- the user moves around the element by holding the mouse button;
- the user scales and moves the item relative to the cursor using the mouse or touch screen.
Analyzing the problem in more detail, when scaling with buttons, we can say that the cursor is in the center of the screen - in the point \((0; 0)\).
When zooming with the mouse we can get the current coordinates of the cursor - \(xc\) and \(yc\) relative to the center of the screen. For button scaling \(xc = 0\) and \(yc = 0\).
An important criterion for scaling with the mouse is that the cursor must remain at the same point of the diagram as before, that is, the diagram must be positioned relative to the cursor, in case of scaling by buttons - relative to the screen center.

![Figure 4](image1.png)
**Figure 4.** Scaling of the radar, initial size; C - middle of the screen, start of reference; K - cursor; A - middle of the circle

![Figure 5](image2.png)
**Figure 5.** Scaling of the radar, size doubled; C - middle of the screen, the beginning of counting; K - cursor location from the figure 4; K1 - current cursor location; A - middle of the circle
Figure 6. Radar scaling, size increased by three times:

C - middle of the screen, the beginning of counting; C - cursor location from the figure 3; A - middle of the circle.

Figure 4 shows an example of a pie chart, which shows the display of the initial state, at two (Figure 5) and three times (Figure 6) magnification.

Since to solve the problem, you need to find the shift along the abscissa and ordinate axes, the problem is reduced to finding the coordinates of point A after scaling with known coordinates before scaling. First, you need to find the coordinates of point A relative to the center of the screen before scaling by following formulas:

\[
\begin{align*}
    x_{AC} &= x_K - x_A \\
    y_{AC} &= y_K - y_A
\end{align*}
\]

(4)

Since the scaling is x1, x2, x3 and x4 times, you need to find the ratio of scale before and after by the following formula:

\[
s' = \frac{s_{i+1}}{s_i}
\]

The next step to convert the values found in formulas 4 to new values taking into account the scaling ratio, calculated by the following formulas:

\[
\begin{align*}
    x_{AC}' &= x_{AC} \times s' \\
    y_{AC}' &= y_{AC} \times s'
\end{align*}
\]

The next step is to find the required coordinates of the new circle center by the following formulas:

\[
\begin{align*}
    x_A' &= x_K - x_{AC}' \\
    y_A' &= y_K - y_{AC}'
\end{align*}
\]

The formulas below used to scale down, they are equivalent to formulas above.

\[
\begin{align*}
    x_{AC} &= x_A - x_K \\
    y_{AC} &= y_A - y_K \\
    s' &= \frac{s_{i-1}}{s_i} \\
    x_{AC}' &= x_{AC} \times s' \\
    y_{AC}' &= y_{AC} \times s' \\
    x_A' &= x_{AC}' + x_K
\end{align*}
\]
\[ y'_A = y'_{AC} + y_K \]

During testing, it was found that the data obtained by mathematical calculations coincide with the data obtained by the graphical method. Consequently, the developed algorithm can be considered correct.

4. Conclusion
This paper considers the implementation of a visualization system for analytical data in the form of a pie chart (radar). In the process of work, the problems of scaling, filtering data, displaying markers depending on the value of the event were solved. The project was successfully implemented and put into experienced exploitation.

References
[1] Iliinsky N and Steele J 2011 Designing Data Visualizations (Sebastopol, CA: O'Reilly) p 142
[2] Krum R 2014 Cool infographics: effective communication with data visualization and design (Indianapolis: Wiley) p 348
[3] Tukey J 1977 Exploratory Data Analysis (Reading, Mass: Pearson) p 688
[4] Paklin N and Oreshkov V 2013 Data visualization. Business Analytics. From data to knowledge (St. Petersburg: Peter) p 210.
[5] Rogers D and Adams J 2001 Mathematical Foundations of Computer Graphics (Moscow: Mir) p 604
[6] Grigoriev and Malozemov V 2010 Compound Curves and Bezier Surfaces (Moldova: LAP Lambert Academic Publishing) p 132