REVIEW ARTICLE

REAL-TIME PLOTTING OF BIOMEDICAL SIGNALS IN APPLICATIONS UTILIZING WINDOWS PRESENTATION FOUNDATION TECHNOLOGY

Ewelina SOBOTNICKA[1], Daniel FEIGE[1]

[1]Łukasiewicz Research Network – Krakow Institute of Technology, The Centre for Biomedical Engineering, Krakow, Poland

Source of support: Own sources

Author's address: E. Sobotnicka, Łukasiewicz Research Network – Krakow Institute of Technology, The Centre for Biomedical Engineering, Zakopianska 73 Street, 30-418 Krakow, Poland, e- mail: ewelina.sobotnicka@itam.lukasiewicz.gov.pl

Abstract: Utilization of personal computers in medicine requires strong emphasis on software performance. Contemporary programming languages give engineers great flexibility in developing applications. Since these technologies are more and more complex, there is a need to analyze methods of achieving desired results taking into account time requirements. This paper focuses mainly on development of plotting module using Windows Presentation Foundation library, which is one of the newest tool for software developers. Aspects of plotter creation for technologically limited screen are covered and conclusions are summarized giving recommendations regarding the topic. General methods of drawing signal on a raster surface are also discussed. Since medical signals in monitoring range consist of frequencies up to about 100 Hz, one has to take into account aspects of signal scaling in order to not to make its important parts invisible. The main problem is that personal computer screen, e.g. a monitor, is purely digital in nature; actually it is the set of pixels (consisting of three components). Because resolution is limited, the signal must be limited in frequency to the range which can be shown at a given plotting speed. This paper describes these aspects of signal plotting as well as the method for virtually increasing pixel resolution by using particular components of a pixel. This paper is a summary giving a general idea of the capabilities of the above mentioned WPF technology and the basis for creating such programs using methods such as: WriteableBitmap method, DrawingVisual method, Canvas Method type 1, Canvas Method type 2.

Keywords: biomedical signals, ECG, Windows Presentation Foundation, WPF, visualization, real-time

INTRODUCTION

Personal computers are playing an increasingly important role in modern medicine, more and more often helping to diagnose and treat diseases. The most common area of application is to display signals received from patients and process them according to the developed algorithm. There is a requirement that the plot being shown must not include significant delay in order not to mislead the doctor. In fact, commonly used PC operating systems are not RTOS (Real-time operating system), thus software engineer has to take application performance into account.

One of the newest technology allowing the development of applications consisting of graphical user interface is Windows Presentation Foundation (WPF) [16]. Despite its name, it allows to create programs designed to run on many popular operating systems used on personal computers [1,7,14]. In fact, at first this library was only available as a part of .Net technology, however, nowadays there are ports on other operating systems. Software engineer may use several programming languages, such as C#, J#, F#, etc. Graphical interface is intended to be designed using XAML (Extensible Application Markup Language) [8,10,16], however, the idea allows any part of the program to be created this way. This technology implements graphical operations with the use of graphic processing unit, allowing fast execution speed of the whole application [7,8,16].

The library described offers many ways of achieving this particular aim. As an example, one can draw line on a screen defining it in XAML, writing the code in C#, and further this code can be written in many different ways, utilizing available controls and modules [12]. One can think of it as just several levels of abstractions, e.g. drawing 10 consecutive pixels is the same as executing function that draws line of length 10. In this case, however, one can use many possibilities of achieving the result in apparently the same way, but after investigation it turns out that there are differences in efficiency.

Because medical signals in monitoring range consist of frequencies up to about 100 Hz, one has to take into account aspects of signal scaling in order to not to make its important parts invisible. The main problem is the fact, that personal computer screen, e.g. monitor, is purely digital in nature; actually it is the set of pixels (consisting of three components). Because resolution is limited, the signal must be limited in frequency to the

range which can be shown in a given plotting speed. This paper describes these aspects of signal plotting as well as the method of virtually increasing pixel resolution, by using particular components of a pixel. This article is addressed to programmers who create software dedicated to medical devices and create utility applications for doctors which presenting in real time medical signals. In this article compared signal drawing methods like: WriteableBitmap method, DrawingVisual method, Canvas Method type 1, Canvas Method type 2.

METHODS OF DRAWING IN WPF

The following assumptions have been made in order to compare methods of drawing:

- measurements of function execution time in ms;
- drawing functions designed to be as simple as possible;
- multiple tests of drawing 1, 20, 50, 100, 200, 500, 900 sample lines at a time;
- multiple tests of drawing sine waveform (1 Hz, 5 Hz, 50 Hz at 1 kHz sampling rate, assuming record speed 1 sample per 1 vertical line), random noise, ECG demo (all signals must have an amplitude of 200 pixels);
- plot area: 1000 x 500 pixels;
- testing PC: Intel Quad 2.4GHz, 4GB RAM, NVIDIA GeForce 8600 GT;
- after reaching right edge of the plot area, the plotter must start drawing again from the left side and erasing of old samples present on screen has to take place, e.g. eraser located 50 pixels before current sample. The idea is visible in Fig. 1.

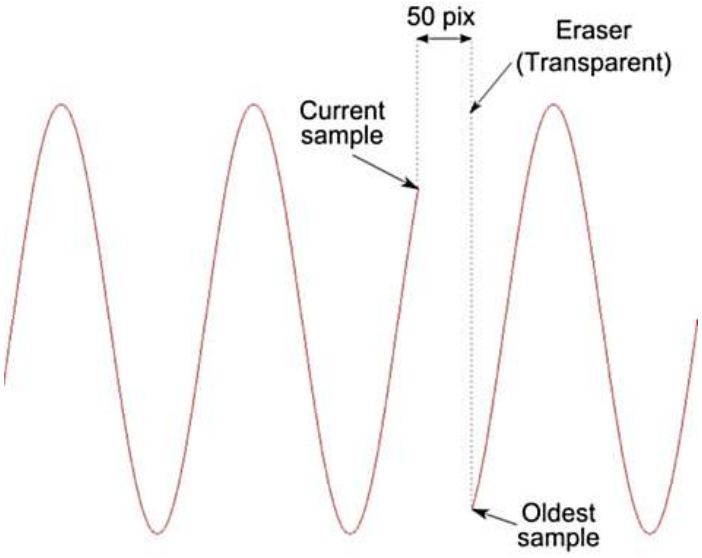

Fig. 1. Idea of signal plotting and erasing.

It is worth to mention that there is no need to draw whole 50 pixel wide eraser rectangle if a function draws plot sample by sample. Then only one vertical eraser line at a time is necessary. In case of drawing many samples at a time one may use eraser wider than 1 pixel line, however, problems which can arise at the right edge of the plot area have to be taken into account.

WriteableBitmap method

The method utilizes Image surface. **WriteableBitmap** class allows direct pixel manipulation by writing data to BackBuffer. There is an automatic mechanism performing copying this data to FrontBuffer and further to the screen. The main idea is to fill BackBuffer with pixel alpha, red, green and blue values according to the signal samples on input [4]. In fact, each sample should generate vertical line in order to create continuous plot.

Improvement can be achieved by creating buffer containing vertical line pattern and copying appropriate part of this data to the FrontBuffer, at a specified offset. Such block transfer is for sure faster than writing each pixel of vertical line to the buffer separately. The same idea may be applied to the eraser line by simple copying buffer containing "empty" pixels to the FrontBuffer at a specified offset. The idea is presented in Fig. 2.

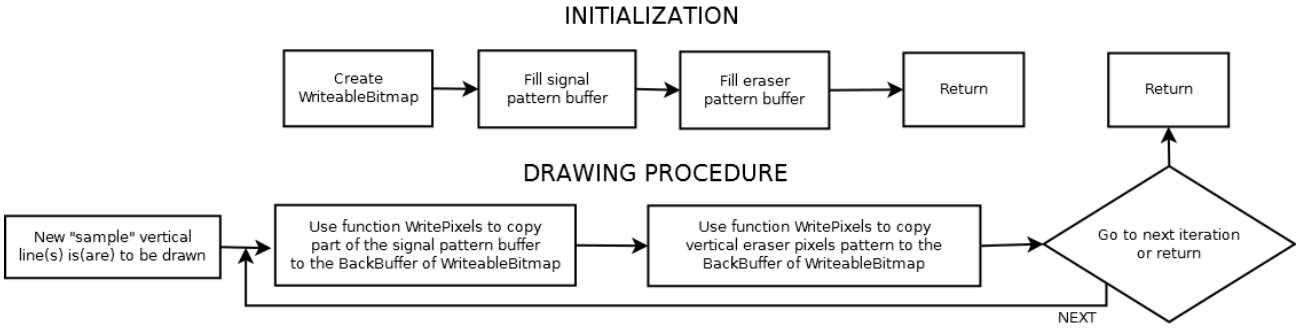

Fig. 2. WriteableBitmap method flowchart.

This method is the most complicated (out of described in this paper) from the point of view of software developer since it requires operations on buffers using pointers, calculation of buffer offsets, etc. However it offers at the same time great flexibility and control of drawing. It is easy to create multicolour plots and apply pixel operators such as a pixel shader.

DrawingVisual method

The method also uses Image surface, although **DrawingVisual** class and RenderTargetBitmap class are used. The latter is a source for Image. The former

is a surface on which one can draw using DrawingContext class. This class contains DrawLine method suitable for the purpose of signal plotting [6]. This method is simple from the developer point of view, however, flexibility is lower comparing to WriteableBitmap. General idea is shown in Fig. 3.

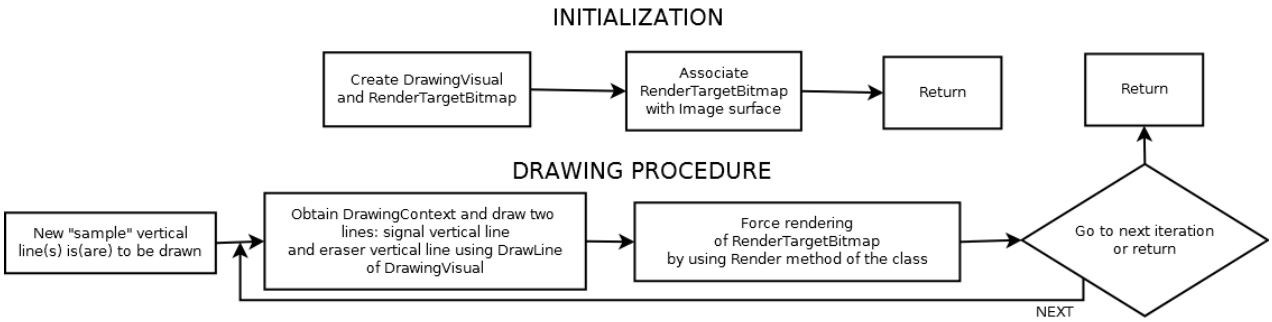

Fig. 3. DrawingVisual method flowchart.

Canvas Method type 1

In contrast to previously described method, this one utilizes **Canvas** class as a surface for drawing. It does not require any initialization (except definition of surface in xaml) and relies on adding children to the Canvas [2,3]. These children may by Polylines created with PointCollection. The idea is shown in Fig. 4. For the purpose of this paper, this method has been treated separately from the next one which is slightly different. The method always sets two points for given signal sample to draw vertical line. One multipoint requirement for loop has been used. In contrast, the next method always draws consecutive samples by defining multipoint Polyline. Since the method relies on adding children to the surface, it is relatively easy to erase old part of the plot by calling Remove method of Canvas.

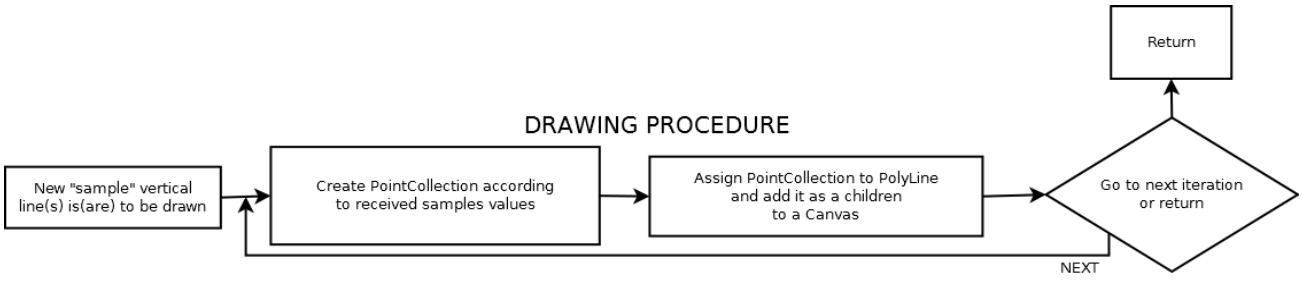

Fig. 4. Canvas Method flowchart.

Canvas Method type 2

As mentioned above, this method is almost the same as the previous one. When multipoint approach is required, it utilizes multipoint Polyline. Erasing is also as simple as in Canvas Method type 1, however, when multipoint approach is taken, it is not easy

to erase a single sample [2,3]. In this case only group of samples can be simply erased from screen and its size is the same as the number of polyline points.

MEASUREMENTS

Measurements have been made according to the assumptions given before. Each method has been tested by 6 test signals. The signals selected for testing are: sine 1HZ, sine 5HZ, sine 50Hz, random signal, ECG 60ppm, ECG 180ppm. The frequency components of a typical ECG signal are in the range of 0.05–100 Hz, therefore, most approaches use filters to eliminate unwanted signals that fall outside this range [15]. Typically 1Hz, 5Hz and 50Hz signals are used for tests [15]. Additionally, random test signal has been generated by system function NextDouble of Random class. The article is a review and its purpose is to compare the methods of signal presentation, therefore a randomly generated signal was taken into account, which is not a standard signal used for tests. The aim of this procedure was to obtain objective results and to indicate a universal method that could be used to present signals of different quality. In addition, ECG 60ppm and ECG 180ppm signals were selected for testing, showing normal rhythm and tachycardia in order to compare the drawing methods under near-real conditions [5]. ECG signal used (generated demo signal) is shown in Fig. 5.

1, 20, 50, 100, 200, 500, 900 multipasses have been used. The tests were run twice and the results were averaged. Averaged test results (in ms) are given in Tab. 1. Multipass means that the function draws given number of samples (vertical lines) at one call. Value 0 in tables means: execution time is less than 1 ms. For example, for the Sine 1Hz signal for the Canvas Type 2 method, the drawing time for multipass 1, 20, 50, 100, 200 is less than 1 ms, while for multipass 500 time is 1 ms and for 900 time is 3 ms. The tests were conducted using a PC equipped with: Intel Quad 2.4GHz, 4GB RAM, NVIDIA GeForce 8600 GT.

Fig. 5. Demo ECG signal (sampling 1kHz).

Tab. 1. Averaged test results [ms].

Signal	WriteableBitmap	DrawingVisual	Canvas Type 1	Canvas Type 2	WriteableBitmap	DrawingVisual	Canvas Type 1	Canvas Type 2	WriteableBitmap	DrawingVisual	Canvas Type 1	Canvas Type 2	WriteableBitmap	DrawingVisual	Canvas Type 1	Canvas Type 2	WriteableBitmap	DrawingVisual	Canvas Type 1	Canvas Type 2	WriteableBitmap	DrawingVisual	Canvas Type 1	Canvas Type 2
	Number of samples drawn in one function call (passes)																							
	1				20				50				100				200				500				900			
Sine 1Hz	0	0	0	0	0	0	0	0	0	2	1	0	0	17	5	0	0	31	22	0	12	79	58	1	25	152	102	3
Sine 5Hz	0	0	0	0	0	0	0	0	0	2	0	0	0	17	5	0	0	31	23	0	12	79	65	1	26	157	102	3
Sine 50Hz	0	0	0	0	0	0	0	0	0	3	0	0	0	17	6	0	0	31	23	0	13	80	61	1	29	158	105	3
Random	0	0	0	0	1	1	0	0	0	5	1	0	0	18	6	0	2	32	23	1	17	86	61	1	31	150	103	3
ECG 60ppm	0	0	0	0	0	0	0	0	0	1	0	0	0	15	6	0	3	32	21	1	17	78	60	1	28	145	102	3
ECG 180ppm	0	0	0	0	0	0	0	0	0	1	1	0	0	15	6	0	2	33	24	1	11	79	61	1	29	147	100	3

RESULTS

Looking at the results above, one can easily conclude that Canvas Method type 2 is the best for desired purpose. Event 900 samples multipass call takes only 3 ms. From the point of view of execution speed, the result is quite robust. Such method is able to process signal sampled at 300 kHz showing each sample as a separate vertical line. Of course, when one tries to show more than one signal at a time, this result is worse; however, assuming existence of 10 concurrent signals on screen, the method can be for sure able to process 30 kHz sampled data with no visible delay. Anyway, the method relies on the fact that multiple points are drawn using polyline, which makes it hard to erase only part of such a group and produces several problems concerning handling screen edges when its size is not multiplication of sample group.

WritableBitmap method also shows promising results, which is also supported by the fact that the method gives great flexibility in using additional plot operations

(like colouring parts of signal) and possibility of easy implementation of signal smoothing or pixel shaders.

Two remaining methods give worse results and there is no huge encouragement to use them to create signal plots. They are more suitable for drawing static pictures, like charts.

It is clearly visible that direct utilization of graphic processing unit, which takes place in WPF [9,16], allows usage of this technology in medical applications. Since each drawing method requires a lot of background auxiliary operations not directly related to the GPU, one has to take into account optimization of the code.

SCALING AND RESOLUTION PROBLEMS

Signal Scaling

Acquired medical signal has to be initially scaled. This is easily demonstrated when one realises that a standard 17' laptop has a screen length of about 30 cm, which gives pixel width of about 0,22 mm at resolution 1366x768. Standard record speed in medical applications is 10 mm/s, so 1s of signal can consist of 45 pixels in this case. If the signal frequency is 45 Hz, it gives only one pixel per period, making it impossible to recognize it on the screen. In fact, the maximum frequency in this case is about 22.5 Hz and it still will be only set of alternating points of value equal in magnitude. Frequency limits for particular record speeds (1 kHz sampling) are shown in Tab. 2.

Tab. 2. Frequency limits for particular record speeds (1 kHz sampling).

Record speed	Pixel width	Pixel resolution	Maximum frequency
5 mm/s	0.22 mm	23 pix/s	11.5 Hz
10 mm/s		45 pix/s	22.5 Hz
20 mm/s		91 pix/s	45.5 Hz
25 mm/s		113 pix/s	56.5 Hz
50 mm/s		227 pix/s	113.5 Hz
100 mm/s		454 pix/s	227.0 Hz
200 mm/s		909 pix/s	454.5 Hz
220 mm/s		1000 pix/s	500.0 Hz
250 mm/s		1136 pix/s	568.0 Hz

Since there is limited pixel resolution (number of pixels per second for given record speed), one has to take into consideration method of transforming input signal to the form suitable for screen buffer. Commonly used sampling speed of ECG signal is 1 kHz.

At a record speed of 10 mm/s, each second of signal (1000 samples) must be mapped to 45 pixels. Assuming initial filtering of signal cutting off at 22.5 Hz, one needs to downsample it to obtain 45 samples and then plot by drawing lines between consecutive points or alternatively create vertical lines of plot, calculating each vertical line using about 22 samples. The latter method can be easily implemented as a procedure searching for maximum and minimum value in the set of samples. Both methods are shown in Fig. 6.

Input signal is not shown in the figure, since its shape is not so important to explain the idea. There are two output screens, generated using previously described methods. Only first 11 vertical outputs are shown. As can be seen, these outputs are not identical. Each algorithm is just a way of converting set of samples to a limited in resolution raster screen, thus there is no perfect method producing the output.

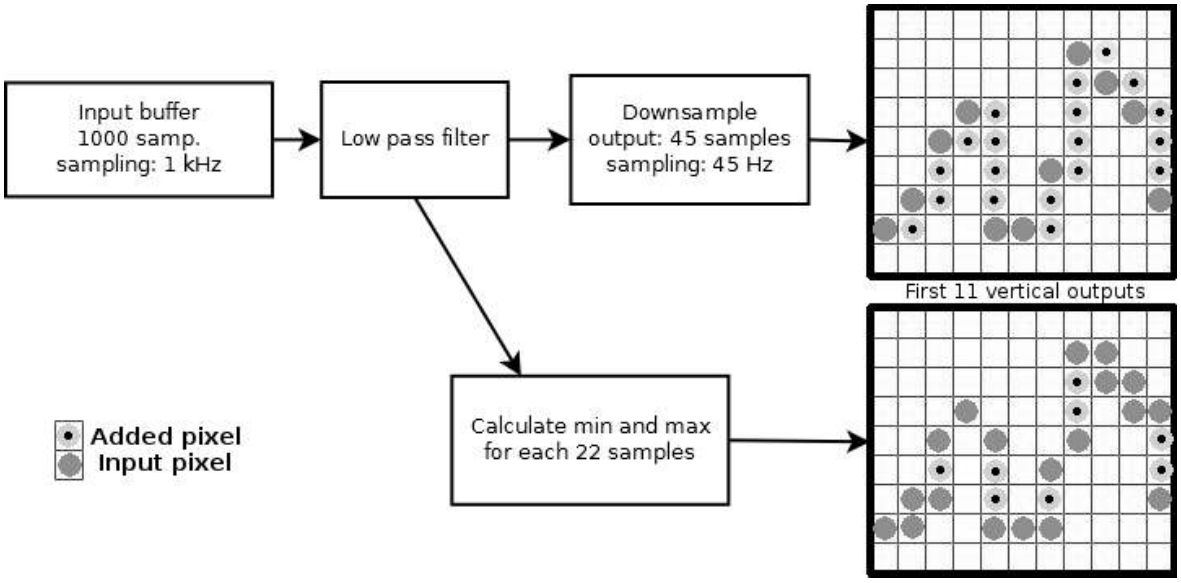

Fig. 6. Methods of signal rasterization (1s at sampling:1 kHz).

Limited resolution problem

Plot generated on a raster screen may look edgy. Particularly, the steps between consecutive vertical lines are sharp. There is a possibility of virtually increasing screen resolution in order to smooth out the transitions between lines. Modern screens, e.g. LCDs, are built of pixel triples (red, green, blue). A line step plotted using standard (left) and component (right) methods is shown in Fig. 7.

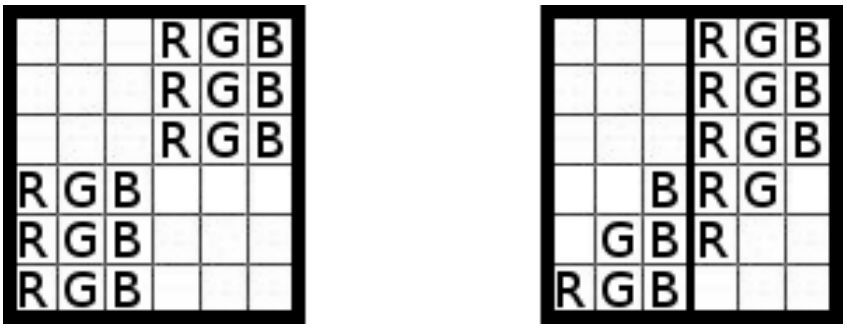

Fig. 7. Method of drawing sample points with use of pixel components.

Since the screen pixels are located near to each other, it is possible to virtually assemble a single point from two pixels. If the user is not looking at the screen with a magnifying glass, he or she will not be able to distinguish colours between pixels with RGB components, or two pixels with components B and R+G with other components adjusted to background of a picture. For example: one wants to show on the screen a line consisting of 100% red, 0% green and 100% blue. Using composition method, a pixel consisting of 0% red, 0% green, 100% blue, and other consisting of 100% red, 0% green, 0% blue can be used in the step plot. In fact, it will create a completely blue point and a completely red point, but the composition will give the impression of this fusion, resulting in desired line colour. The plot steps will be smoother then.

Drawing using frames

It turns out that better application performance can be achieved when drawing plots not one line by line, but by creating groups of lines to be drawn at one time. Canvas Method type 2 is one of those which strongly depends on such assumption. As far as the other methods described earlier are concerned, it is also better to call drawing function once for multiple inputs, not one by one, even if the function draws each line separately. The problem which may arise concerns an effect of plot bouncing. If software developer assumes drawing too huge set of lines at one function call, it is possible that this will be done much faster than the actual time length of the signal part. For example, drawing 45 lines of signal which normally shows 1s need not to be done in one call taking 100 ms. In effect, the user would see these 900 ms breaks between drawing periods. On the other hand, it is better to draw as fast as possible, because at any time the processor may become occupied by any process slowing down plotting. It is thus important to choose the length of samples frame so that it is the maximum that can be accepted by the user's eye. These a considerations may be useful in selecting the appropriate settings.

CONCLUSIONS

Windows Presentation Foundation (WPF) is nowadays quite a robust technology. Since it is build at a high level of abstraction, in contrast to libraries requiring low level programming, there are many possibilities of achieving the same result [13,14]. WPF enables the binding of user controls directly to the data in the device. In cases where medical devices aid in saving lives, technologies like WPF are extremely important because every second counts. When developing application aimed to be used for medical purposes, special attention should be paid to time requirements. This paper is a summary that gives a general idea of what the capabilities of the mentioned technology are and what the basis for developing such programs.

Depending on the requirements regarding the plot appearance, it is recommended to use WriteableBitmap method or Canvas Method type 2 in connection with plot smoothing method together with proper signal scaling and filtering. Of course, there are many other ways of plotting signal in WPF, however, one has to remember that seemingly the same operation performed at a high level of abstraction can lead to many different performance results when converted by the compiler to low-level machine code. One can conclude that the methods described above are suitable and sufficient to be used in medical applications.

Given the popularity of smartphones, smart tablets, and other high-tech consumer products with context-aware or multi-input features, it is important for engineers to think how they too can leverage these technologies to advance the medical field. Developers are quickly learning that technology is transferrable across professional practices. The same features that we use to operate our computer, tablet, smartphone are now being considered as validated and useful technologies to be applied in the development of medical devices and applications.

This article may be useful for developers writing computer or mobile applications for the real time acquisition of biomedical signals to be presented in real time on the screen. They do not have to think about the waveform drawing method, they can choose the method that was found to be the best in this article.

AUTHORS' DECLARATION

Study Design: Ewelina Sobotnicka, Daniel Feige. **Data Collection:** Ewelina Sobotnicka. **Manuscript Preparation:** Ewelina Sobotnicka, Daniel Feige. The Authors declare that there is no conflict of interest.

REFERENCES

1. Advanced (Windows Presentation Foundation), Microsoft documentation, https://docs.microsoft.com/en-us/dotnet/desktop/wpf/advanced/?view=netframeworkdesktop-4.8, may 2022.
2. Canvas Class, Microsoft documentation, https://docs.microsoft.com/pl-pl/dotnet/api/system.windows.controls.canvas?view=windowsdesktop-6.0, May 2022.
3. Canvas, The Graphics Canvas element, https://developer.mozilla.org/en-US/docs/Web/HTML/Element/canvas, May 2022.
4. Class WriteableBitmap, Microsoft documentation, https://docs.microsoft.com/pl-pl/dotnet/api/system.windows.media.imaging.writeablebitmap?view=windowsdesktop-6.0, May 2022.
5. De Lucena SE. ECG simulator for testing and servicing cardiac monitors and electrocardiographs. 18th IMEKO TC4 Symposium on Measurement of Electrical Quantities 2011, Part of Metrologia. 2011; 109-112.
6. DrawingVisual.RenderOpen Method, Microsoft documentation, https://docs.microsoft.com/en-us/dotnet/api/system.windows.media.drawingvisual.renderopen?view=windowsdesktop-6.0, May 2022.
7. Janckulik D, Motalova L, Krejcar O. Software Solutions for Implementation of Biotelemetric System, IEEE 2010 International Conference on Applied Electronics. 2010; 1-4.
8. Janckulik D, Motalova L, Krejcar O. Use of Mobile Embedded Devices for Processing of Biomedical Signals, XII Mediterranean Conference on Medical and Biological Engineering and Computing Medicon, Greece, 2010.
9. Krejcar O, Janckulik D, Motalova L. Real Time Measurement and Visualization of ECG on Mobile Monitoring Stations of Biotelemetric System, Advances in Intelligent Information and Database Systems Studies in Computational Intelligence. 2010; 283: 67-78.
10. Krupka K, Dickhaus H, Katus HA, Hilbel T. Displaying computerized ECG recordings and vital signs on Windows Phone 7 smartphones, Computing in Cardiology. 2010; 1067-1070.
11. Litvinavicius T. Exploring Windows Presentation Foundation: With Practical Applications in. NET 5. Apress. 2021.

12. Mayo J. C# 3.0 Unleashed: With the .NET Framework 3.5, vol. ISBN 978-0672329814, Sams, 2008.
13. Pasztaleniec M, Skublewska-Paszkowska M. Analiza porównawcza technologii Windows Presentation Foundation i Windows Forms, JCSI. 2020; 14: 26-30.
14. Perry SC. Core C# and .NET: The Complete and Comprehensive Developer's Guide to C# 2.0 and .NET 2.0, vol. ISBN 978-0131472273, Prentice Hall. 2005.
15. Rangayyan RM. Biomedical Signal Analysis: A Case-study Approach. Wiley–Interscience; New York, NY, USA. 2001.
16. XU J. Practical WPF Charts and Graphics: Advanced chart and graphics programming with the Windows Presentation Foundation, New York. Apress. 2009; 1-11.