AnalogClock

Software Documentation

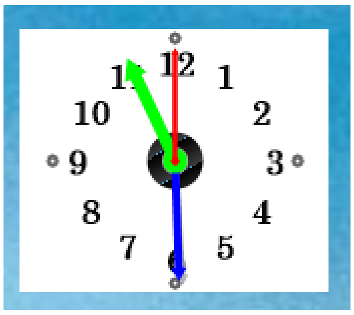
Author: matiwa

Table of contents

Introduction

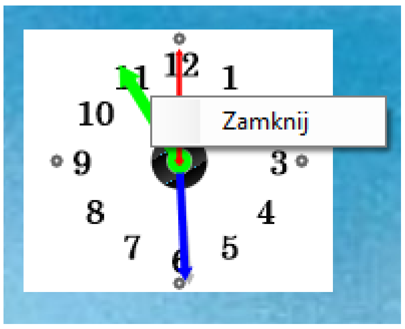
This software documentation includes: description of the application's operation, what is needed for use, algorithms used, interface description and source code description. This application is used to display the current time.

Describing of the application’s operation



Drawing 1: The beginning of the application's operation [own study]

This application is implemented as a gadget for the user's personal computer and laptop.



Drawing 2: The ability to close the gadget [own study]

The only aspect where there is an operation that the user can perform is shutting down the gadget. He can do this by right-clicking within the dial and then left-clicking. This is what the "Close" option from Polish is for.

The only drawback of the application is the lack of movement, i.e. the rotation of the hand for seconds.

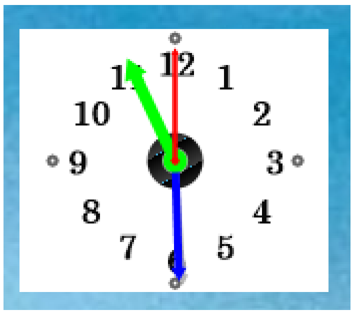
What is needed for use?

The application does not require installation. It only needs the Windows operating system.

Algorithm used

The basic form of the algorithm can be deduced from the previous section. All in all, the app is a typical analog clock gadget.

Interface description



Drawing 3: Graphical interface [own study]

The interface is typical for a Windows Forms Application without title bar with minimize, maximize or shrink and close.. There are essential components: timer, contextmenustrip with toolstripitem. Outside of it there is a bitmap.

Source code description

The project was made in the C# programming language, in the Visual Studio Community 2017 programming environment. All work was done on the Windows 10 operating system. The application's source code looks like this.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Drawing.Drawing2D;
using System.Drawing.Text;

namespace AnalogClock
{
    public partial class Form1 : Form
    {
        private Rectangle rect;
        private LinearGradientBrush obramowanieKolor;
        private SolidBrush tarczaKolor;
        private SolidBrush liczbyKolor;
        private SolidBrush podpisKolor;
        private Pen cienTarczyKolor;
        private Pen pioro;
        private Pen pioroG;
        private Pen pioroMin;
        private Pen pioroSek;
        private int srednica=10;

        private Point MouseAktualnaPoz, MouseNowaPoz, formPoz, formNowaPoz;
        private bool mouseDown = false;

        public Form1()
        {
            InitializeComponent();
            this.SetStyle(ControlStyles.DoubleBuffer, true);
            this.SetStyle(ControlStyles.UserPaint, true);
            this.SetStyle(ControlStyles.AllPaintingInWmPaint, true);

            this.ShowInTaskbar = false;
            Refresh();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            rect = new Rectangle(this.ClientSize.Width / 2 - srednica / 2, 
this.ClientSize.Height / 2 - srednica
                / 2, srednica, srednica);

            obramowanieKolor = new LinearGradientBrush(rect, Color.FromArgb(0, 0, 0), 
Color.FromArgb(60, 60, 60),
                60);
            tarczaKolor = new SolidBrush(Color.WhiteSmoke);
            liczbyKolor = new SolidBrush(Color.FromArgb(10, 10, 10));
            podpisKolor = new SolidBrush(Color.Blue);
            cienTarczyKolor = new Pen(Color.FromArgb(180, 180, 180), 3);
            pioro = new Pen(Color.FromArgb(100, 100, 100), 4);
            pioroG = new Pen(Color.FromArgb(0, 0, 255), 4);
            pioroMin = new Pen(Color.FromArgb(0, 255, 0), 6);
            pioroSek = new Pen(Color.FromArgb(255, 0, 0), 2);

            pioro.EndCap = LineCap.ArrowAnchor;
            pioro.StartCap = LineCap.RoundAnchor;
            pioroG.EndCap = LineCap.ArrowAnchor;
            pioroG.StartCap = LineCap.RoundAnchor;
            pioroMin.EndCap = LineCap.ArrowAnchor;
            pioroMin.StartCap = LineCap.RoundAnchor;
            pioroSek.EndCap = LineCap.ArrowAnchor;
            pioroSek.StartCap = LineCap.RoundAnchor;
            cienTarczyKolor.EndCap = LineCap.ArrowAnchor;
            cienTarczyKolor.StartCap = LineCap.RoundAnchor;
        }

        private void Form1_Paint(object sender, PaintEventArgs e)
        {
            Rysuj(e.Graphics);
        }

        public void Rysuj(Graphics graphics)
        {
            graphics.SmoothingMode = SmoothingMode.AntiAlias;
            graphics.TextRenderingHint = TextRenderingHint.AntiAlias;
            graphics.InterpolationMode = InterpolationMode.HighQualityBicubic;
            //wspolrzedne srodka okna
            int srWidth;
            int srHeight;

            //czas w danej chwili
            int minuty;
            int godziny;
            double sekundy;

            //rotacje wskazowek
            double minutyTic;
            double godzinyTic;
            double sekundyTic;

            float promien; //dlugosc wskazowki
            int ramka; //szerokość czarnej ramki
            int stopnie; //odleglosc miedzy liczbami

            DateTime czas;

            srWidth = this.ClientSize.Width / 2;
            srHeight = this.ClientSize.Height / 2;

            ramka = 18;

            graphics.FillEllipse(obramowanieKolor, srWidth - srednica / 2 - ramka / 2,
srHeight - srednica / 2
                - ramka / 2, srednica + ramka, srednica + ramka);
            graphics.FillEllipse(tarczaKolor, rect);
            graphics.DrawEllipse(cienTarczyKolor, rect);

            graphics.TranslateTransform(srWidth, srHeight);

            StringFormat format = new StringFormat();
            format.Alignment = StringAlignment.Center;
            format.LineAlignment = StringAlignment.Center;
            Font textFont = new Font("Century", 12F, FontStyle.Bold);

            stopnie = 360 / 12;
            promien = 49;

            for (int i = 1; i <= 12; i++)
            {
                graphics.DrawString(i.ToString(), textFont, liczbyKolor, -1 *
ObliczX(i * stopnie + 90, promien)
                    + 1, -1 * ObliczY(i * stopnie + 90, promien) + 2, format);
            }

            promien = 61;
            stopnie = 360 / 4;

            for (int i = 1; i <= 4; i++)
            {
                graphics.DrawEllipse(pioro, -1 * ObliczX(i * stopnie + 90, promien) -
1, -1 * ObliczY(i * stopnie + 90, promien) - 1, 2, 2);
            }

            czas = DateTime.Now;
            godziny = czas.Hour;
            minuty = czas.Minute;
            sekundy = czas.Second + (czas.Millisecond * 0.001);
            godzinyTic = 2.0 * Math.PI * (godziny + minuty / 60.0) / 12.0;

            Point pktSrodek = new Point(0, 0);
            Point pktCienGodzina = new Point((int)((promien * Math.Sin(godzinyTic)) +
2), (int)((-(promien) *
                Math.Cos(godzinyTic)) + 2));
            graphics.DrawLine(cienTarczyKolor, pktSrodek, pktCienGodzina);

            Point pktWskGodzina = new Point((int)(promien * Math.Sin(godzinyTic)),
(int)(-(promien) *
                Math.Cos(godzinyTic)));
            graphics.DrawLine(pioroG, pktSrodek, pktWskGodzina);

            minutyTic = 2.0 * Math.PI * (minuty + sekundy / 60.0) / 60.0; promien =
57;
            Point pktCienMinuta = new Point((int)(promien * Math.Sin(minutyTic) + 2),
(int)(-(promien) *
                Math.Cos(minutyTic) + 2));
            graphics.DrawLine(cienTarczyKolor, pktSrodek, pktCienMinuta);
            Point pktWskMinuta = new Point((int)(promien * Math.Sin(minutyTic)),
(int)(-(promien) *
                Math.Cos(minutyTic)));
            graphics.DrawLine(pioroMin, pktSrodek, pktWskMinuta);

            sekundyTic = 2.0 * Math.PI * (sekundy+czas.Millisecond+0.001) / (60.0 *
60.0) /3600;
            Point pktCienSekunda = new Point((int)(promien * Math.Sin(sekundyTic)),
(int)(-(promien) *
                Math.Cos(sekundyTic)));
            graphics.DrawLine(pioroSek, pktSrodek, pktCienSekunda);

            Point pktWskSekunda = new Point((int)(promien * Math.Sin(sekundyTic)),
(int)(-(promien) *
                Math.Cos(sekundyTic)));
            graphics.DrawLine(pioroSek, pktSrodek, pktWskSekunda);
        }

        private float ObliczX(float stopnie, float r)
        {
            return (float)(r * Math.Cos((Math.PI / 180) * stopnie));
        }

        private float ObliczY(float stopnie, float r)
        {
            return (float)(r * Math.Sin((Math.PI / 180) * stopnie));
        }

        private void Form1_MouseDown(object sender, MouseEventArgs e)
        {
            if (e.Button == MouseButtons.Left)
            {
                mouseDown = true;
                MouseAktualnaPoz = Control.MousePosition;
                formPoz = Location;
            }
        }

        private void Form1_MouseMove(object sender, MouseEventArgs e)
        {
            if (mouseDown == true)
            {
                MouseNowaPoz = Control.MousePosition;
                formNowaPoz.X = MouseNowaPoz.X - MouseAktualnaPoz.X + formPoz.X;
                formNowaPoz.Y = MouseNowaPoz.Y - MouseAktualnaPoz.Y + formPoz.Y;
                Location = formNowaPoz;
                formPoz = formNowaPoz;
                MouseAktualnaPoz = MouseNowaPoz;
            }
        }

        private void Form1_MouseUp(object sender, MouseEventArgs e)
        {
            if (e.Button == MouseButtons.Left)
                mouseDown = false;
        }

        private void Timer1_tick(object sender, EventArgs e)
        {
            Invalidate();
        }

        private void ZamknijToolStripMenuItem_Click(object sender, EventArgs e)
        {
            Close();
        }

        private void Form1_MouseClick(object sender, MouseEventArgs e)
        {
            if (e.Button == MouseButtons.Right)
            {
                this.contextMenuStrip.Show(Control.MousePosition);
            }
        }
    }
}
```

Listing 1: Source code [own study]

List of drawings

List of listings