PointOnACircle

Software Documentation

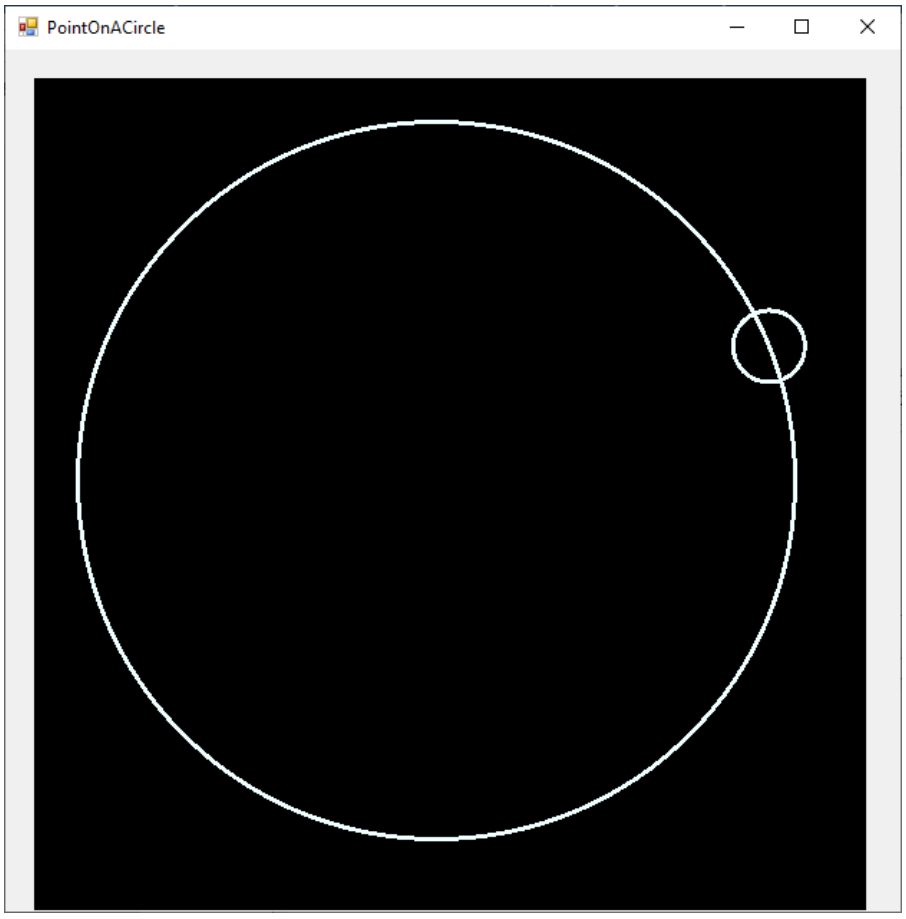
Author: matiwa

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 as a relaxation circle of a circle which is the orbit for the smaller circle and the smaller one moves clockwise around it.

Describing of the application’s operation



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

A bitmap-only program, with no user-accessible component, relies on maintenance to run it and shut it down when redundant. The application itself is endless. The time of her work may be endless.

The smaller circle traverses the larger circle as if it were an orbit.

What is needed for use?

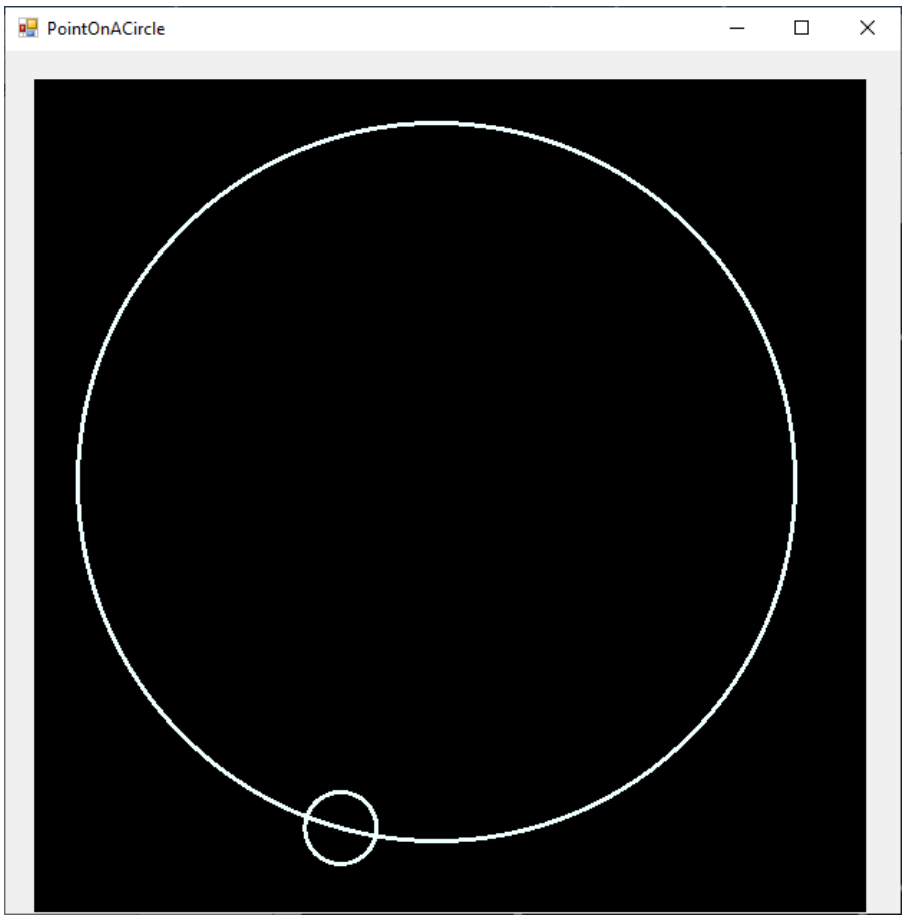
The application does not require installation. It only needs the Windows operating system.

Algorithm used

The basic form of the algorithm can be deduced from the previous section. The basic form of the algorithm can be deduced from the previous section. All in all, the app is applicable as a relaxation aid or attention trainer.

Interface description

The interface is typical for a Windows Forms Application. This program does not contain any components but is bitmap based.



Drawing 8: Graphical interface [own study]

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.Threading;

namespace PointOnACircle
{
    public partial class Form1 : Form
    {
        Thread th;
        Graphics g;
        Graphics fG;
        Bitmap btm;

        bool drawing = true;

        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            btm = new Bitmap(580, 580);
            g = Graphics.FromImage(btm);
            fG = CreateGraphics();
            th = new Thread(Draw);
            th.IsBackground = true;
            th.Start();
        }

        public void Draw()
        {
            float angle = 0.0f;
            PointF org = new PointF(250, 250);
            float rad = 250;
            Pen pen = new Pen(Brushes.Azure, 3.0f);
            RectangleF area = new RectangleF(30, 30, 500, 500);
            RectangleF circle = new RectangleF(0, 0, 50, 50);

            PointF loc = Point.Empty;
            PointF img = new PointF(20, 20);

            fG.Clear(Color.Black);

            while (drawing)
            {
                g.Clear(Color.Black);

                g.DrawEllipse(pen, area);
                loc = CirclePoint(rad, angle, org);

                circle.X = loc.X - (circle.Width / 2) + area.X;
                circle.Y = loc.Y - (circle.Height / 2) + area.Y;


                g.DrawEllipse(pen, circle);

                fG.DrawImage(btm, img);
```

```
                if (angle < 360) angle += 0.5f;
                else angle = 0;
            }
        }

        public PointF CirclePoint(float radius, float angleInDegrees, PointF origin)
        {
            float x = (float)(radius * Math.Cos(angleInDegrees * Math.PI / 180F) +
origin.X);
            float y = (float)(radius * Math.Sin(angleInDegrees * Math.PI / 180F) +
origin.Y);

            return new PointF(x, y);
        }
    }
}
```

Listing 1: Source code [own study]

List of drawings

List of listings