Glebik Roman, AS-36

Controll two metro cars using Speed, Distance, and Brake

This markup drowing our animation.

MainWindow.xaml

```
<Window x:Class="WpfApplication1.MainWindow"
              DataContext="{Binding Main, Source={StaticResource Locator}}">
    <Grid>
             <Canvas>
            <Ellipse Width="400" Height="400" Canvas.Top="15" Canvas.Left="50"
Stroke="Aqua" StrokeThickness="30"></Ellipse>
            <Ellipse Width="30" Height="30" Canvas.Left="{Binding FirstTrain.X,
Mode=OneWay}" Canvas.Top="{Binding FirstTrain.Y, Mode=OneWay}" Fill="Orange"></Ellipse>
            <Ellipse Width="30" Height="30" Canvas.Left="{Binding SecondTrain.Y,
Mode=OneWay}" Canvas.Top="{Binding SecondTrain.Y, Mode=OneWay}" Fill="Pink"></Ellipse>
            <Button Content="tap" Width="50" Height="50" Canvas.Top="202"
Canvas.Left="237" Command="{Binding RunCarsCommand, Mode=OneWay}"></Button>
             </Canvas>
        <StackPanel HorizontalAlignment="Right">
            <TextBlock Text="Green Train:" Foreground="Orange">
                <Run Text="Distance: " Foreground="Black"/><Run Text="{Binding
FirstTrain.Dists}" Foreground="Black"/>
                <Run Text="Speed: " Foreground="Black"/><Run Text="{Binding
FirstTrain.Speed, Converter={StaticResource SpeedConverter}}" Foreground="Black"/>
            </TextBlock>
            <TextBlock Text="Red Train" Foreground="Pink">
                <Run Text="Distance: " Foreground="Black"/><Run Text="{Binding
SecondTrain.Dists}" Foreground="Black"/>
                <Run Text="Speed: " Foreground="Black"/><Run Text="{Binding
SecondTrain.Speed, Converter={StaticResource SpeedConverter}}" Foreground="Black"/>
            </TextBlock>
        </StackPanel>

    </Grid>
</Window>
```

This class working with our trains(ellipses).

VM.cs

```
using System;
using System.Threading.Tasks;
using GalaSoft.MvvmLight;
using GalaSoft.MvvmLight.Command;

namespace Trains.VM
{
    public class MainViewModel : ViewModelBase
    {
           double ellipseRadius = 200, x0 = 237, y0 = 202;

             Random rand = new Random();

             public MainViewModel()
        {
                    RunCarsCommand = new RelayCommand(RunCarsMethod);
            FirstTrain = new Train
            {
                Angle = 0,
```

```
                X = x0 + ellipseRadius * Math.Cos(0),
                Y = y0 + ellipseRadius * Math.Sin(0),
                Speed = 0.01
            };
            SecondTrain = new Train
            {
                Angle = Math.PI,
                X = x0 + ellipseRadius * Math.Cos(Math.PI),
                Y = y0 + ellipseRadius * Math.Sin(Math.PI),
                Speed = 0.01
            };
              }

           public Train FirstTrain { get; set; }
           public Train SecondTrain { get; set; }
           public RelayCommand RunCarsCommand { get; private set; }

             private void RunCarsMethod()
             {

                    Task.Run(async () =>
                    {
                           while (true)
                           {
                                  await Task.Delay(1);
                                  CalcTrain(FirstTrain); CalcTrain(SecondTrain);
CalcDist();
                    if ( (Math.Abs(FirstTrain.Angle  - SecondTrain.Angle ) % (2 *
Math.PI)) < 0.05) { FirstTrain.Dists = 1000; SecondTrain.Dists = 0; break; }
                           }
                    });
             }

           private void CalcTrain(Train car)
           {
                  car.X = x0 + ellipseRadius*Math.Cos(car.Angle);
                  car.Y = y0 + ellipseRadius*Math.Sin(car.Angle);
                     var r = rand.Next(3);
                     switch (r) { case 0: car.Speed -= 0.001; break; case 2: car.Speed +=
0.001; break; }
            if (car.Speed <= 0) car.Speed = 0;
                     car.Angle += car.Speed;
             }

        private void CalcDist()
        {
            FirstTrain.Dists += (int)(-(FirstTrain.Speed) / (2 * Math.PI) * 1000 +
(SecondTrain.Speed) / (2 * Math.PI) * 1000);
            SecondTrain.Dists += (int)(-(SecondTrain.Speed) / (2 * Math.PI) * 1000 +
(FirstTrain.Speed) / (2 * Math.PI) * 1000);
        }

       }

       public class Train : ViewModelBase
       {
        public double Angle { get; set; }
        public int Dists { get; set; }
        public double Speed { get; set; }
        public double X { get; set; }
        public double Y { get; set; }
}
```