2019/04/07
.NET 4.6.2

eselipsync.zip 2019-04-07
　lipsync

2016/02/15

　.NET Framework 4.5.2

Proguram.cs

```csharp
using System;
using System.Speech.Recognition;

namespace eselipsync
{
    class Program
    {
        static void Main(string[] args)
        {
            audioparse a = null;

            if (args.Length == 1)
            {
                a = new audioparse(args[0]);
            }
            else
            {
                a = new audioparse();
            }

            try
            {
                a.parse();
                Console.ReadKey();
            }
            catch (Exception e)
            {
                Console.WriteLine("{0},{1}", e.Message, e.StackTrace);
            }

            a.close();

        }
    }
    class audioparse
    {
        public bool finish { get; private set; }
        public bool sourceSw { get; private set; }

        SpeechRecognitionEngine sre;
        string wavpath;

        public audioparse(string wav)
        {
            finish = false;
            wavpath = wav;
            sourceSw = true;
        }
        public audioparse()
```

```csharp
        {
            finish = false;
            wavpath = "";
            sourceSw = false;
        }
        public void close()
        {
            sre.Dispose();
        }
        public SpeechRecognitionEngine parse()
        {
            sre = new SpeechRecognitionEngine();

            sre.LoadGrammar(new DictationGrammar());

            sre.SpeechDetected += sre_SpeechDetected;
            sre.SpeechRecognized += sre_SpeechRecognized;
            sre.RecognizeCompleted += sre_RecognizeCompleted;
            sre.AudioStateChanged += sre_AudioStateChanged;

            //sre.SpeechHypothesized += sre_SpeechHypothesized;
            //sre.SpeechRecognitionRejected +=
sre_SpeechRecognitionRejected;

            //sre.BabbleTimeout = new TimeSpan(Int32.MaxValue);
            //sre.InitialSilenceTimeout = new TimeSpan(Int32.MaxValue);
            //sre.EndSilenceTimeout = new TimeSpan(0,0,10);
            //sre.EndSilenceTimeoutAmbiguous = new
TimeSpan(Int32.MaxValue);

            if (sourceSw)
            {
                sre.SetInputToWaveFile(wavpath);
            }
            else
            {
                sre.SetInputToDefaultAudioDevice();
            }

            Console.WriteLine("s        ");

            sre.RecognizeAsync(RecognizeMode.Multiple);

            return sre;
        }

        private void sre_AudioStateChanged(object sender,
AudioStateChangedEventArgs e)
        {
            Console.WriteLine("d     ,{0}", sre.AudioState);
        }
```

```
        void sre_RecognizeCompleted(object sender,
RecognizeCompletedEventArgs e)
        {
            finish = true;
            sre.RecognizeAsyncCancel();
            sre.RecognizeAsyncStop();
            Console.WriteLine("r        ");
        }
        void sre_SpeechDetected(object sender, SpeechDetectedEventArgs
e)
        {
            Console.WriteLine("d     ");
        }
        void sre_SpeechRecognized(object sender,
SpeechRecognizedEventArgs e)
        {
            Console.WriteLine("a {0,0:f9}, {1}", e.Result.Confidence,
e.Result.Text);
            for(int idx =0; idx<e.Result.Words.Count; idx++)
            {
                Console.WriteLine("p {0:f9}, {1} -> {2} -> {3}",
e.Result.Words[idx].Confidence,
e.Result.Words[idx].Pronunciation,
e.Result.Words[idx].LexicalForm,
e.Result.Words[idx].Text);
            }
        }
    }
}
```