関係コード６

ソース解説をすることはしません。最新版との同期もとれていません。

概要

音声合成製品制御コードサンプル。

ソースコード

VOICEVOXの制御コード

20211205/u公開時のもの。このソースではマルチスレッドを考慮していません。
doubleの定義はdecimalへ変更するかもしれません。

VoiceVoxProxy.cs

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Net.Http;
using System.Net.Http.Headers;
using System.Runtime.Serialization;
using System.Runtime.Serialization.Json;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace ScDriver.VoiceVox : IDisposable
{
    public class VoiceVoxProxy
    {
        private static HttpClient client;

        public VoiceVoxProxy()
        {
            try
            {
                client = new HttpClient();
            }
            catch (Exception)
            {
                client = null;
            }
        }
        public void Dispose()
        {
            client?.Dispose();
        }

        private void SettingJsonHeader()
        {
            client.DefaultRequestHeaders.Accept.Clear();
            client.DefaultRequestHeaders.Accept.Add(new
MediaTypeWithQualityHeaderValue("application/json"));
            client.DefaultRequestHeaders.Accept.Add(new
MediaTypeWithQualityHeaderValue("audio/wav"));
            client.DefaultRequestHeaders.Add("User-Agent",
"AssistantSeika Driver");
        }

        private void PostSynthesisQuery(VoiceVoxAudioQuery aq, int
speaker, string saveFileName = "")
        {
            var json = new
DataContractJsonSerializer(typeof(VoiceVoxAudioQuery));
            MemoryStream ms = new MemoryStream();
            json.WriteObject(ms, aq);

            var content = new
StringContent(Encoding.UTF8.GetString(ms.ToArray()), Encoding.UTF8,
"application/json");

            Task.Run(async () => {

                SettingJsonHeader();

                try
                {
                    var response = await
client.PostAsync(string.Format(@"http://localhost:50021/synthesis?speak
er={0}", speaker), content);

                    if(response.StatusCode==
System.Net.HttpStatusCode.OK)
                    {
                        string tempFileName = saveFileName == "" ?
Path.GetTempFileName() : saveFileName;
                        using (FileStream tempfile = new
FileStream(tempFileName, FileMode.Create, FileAccess.Write,
FileShare.None))
                        {
                            await
response.Content.CopyToAsync(tempfile);
                        }

                        if (saveFileName == "")
                        {
                            var player = new
System.Media.SoundPlayer(tempFileName);
                            player.PlaySync();
                            File.Delete(tempFileName);
                        }
                    }
                }
                catch (Exception e)
                {
                    MessageBox.Show(string.Format("**
PostSynthesisQuery2 [{0}:{1},{2}]", speaker, saveFileName, e.Message));
                }
            }).Wait();
        }

        private VoiceVoxAudioQuery GetAudioQuery(string text, int
speaker)
        {
            var content = new StringContent("{}", Encoding.UTF8,
@"application/json");
            VoiceVoxAudioQuery ans = null;
            DataContractJsonSerializerSettings settings = new
DataContractJsonSerializerSettings();

            settings.UseSimpleDictionaryFormat = true;

            Task.Run(async () => {
                SettingJsonHeader();

                try
                {
                    var response = await
client.PostAsync(string.Format(@"http://localhost:50021/audio_query?tex
t={0}&speaker={1}", text, speaker), content);

                    if (response.StatusCode ==
System.Net.HttpStatusCode.OK)
                    {
                        var json = new
DataContractJsonSerializer(typeof(VoiceVoxAudioQuery), settings);
                        ans = (VoiceVoxAudioQuery)json.ReadObject(await
response.Content.ReadAsStreamAsync());
                    }
                }
                catch (Exception e)
                {
                    MessageBox.Show(string.Format("** GetAudioQuery
{0}:{1},{2}", speaker, text, e.Message));
                }
            }).Wait();

            return ans;
        }

        /// <summary>
        /// 話者パラメタの取り出し
        /// </summary>
        /// <param name="speaker">話者番号</param>
        /// <returns>パラメタ情報</returns>
        public VoiceVoxParams GetAvatorParams(int speaker)
        {
            VoiceVoxParams ans = new VoiceVoxParams();

            try
            {
                VoiceVoxAudioQuery aq = GetAudioQuery("あ", speaker);

                if (aq!=null)
                {
                    ans.intonationScale = (double)aq.intonationScale;
                    ans.pitchScale = (double)aq.pitchScale;
                    ans.speedScale = (double)aq.speedScale;
                    ans.volumeScale = (double)aq.volumeScale;
                }
            }
            catch(Exception e)
            {
                MessageBox.Show(string.Format("** GetAvatorParams
[{0}:{1}]", speaker, e.Message));
            }

            return ans;
        }

        /// <summary>
        /// 利用可能な話者の取り出し
        /// </summary>
        /// <returns>話者番号と名称の組み合わせのリスト</returns>
        public List<KeyValuePair<int, string>> AvailableCasts()
        {
            DataContractJsonSerializerSettings settings = new
DataContractJsonSerializerSettings();
            List<VoiceVoxSpeaker> speakers = new
List<VoiceVoxSpeaker>();
            var ans = new List<KeyValuePair<int, string>>();
            Task.Run(async () => {
                SettingJsonHeader();

                try
                {
                    var response = await
client.GetAsync("http://localhost:50021/speakers");

                    if (response.StatusCode ==
System.Net.HttpStatusCode.OK)
                    {
                        var json = new
DataContractJsonSerializer(typeof(List<VoiceVoxSpeaker>), settings);

                        speakers =
(List<VoiceVoxSpeaker>)json.ReadObject(await
response.Content.ReadAsStreamAsync());

                        ans = speakers.SelectMany(v1 =>
v1.styles.Select(v2 => new { id = v2.id, name = string.Format("{0}
□{1}□", v1.name, v2.name) }))
                                                          .OrderBy(v =>
v.id)
                                                          .Select(v =>
new KeyValuePair<int, string>(v.id, v.name)).ToList();
                    }
                }
                catch (Exception e)
                {
                    MessageBox.Show(string.Format("** AvailableCasts
[{0}]", e.Message));
                }
            }).Wait();

            return ans;
        }

        /// <summary>
        /// 発声
        /// </summary>
        /// <param name="speaker">話者番号</param>
        /// <param name="param">エフェクト</param>
        /// <param name="text">発声させるテキスト</param>
        public void Speak(int speaker, VoiceVoxParams param, string
text)
        {

            VoiceVoxAudioQuery aq = GetAudioQuery(text, speaker);

            if (param != null)
            {
                aq.volumeScale = param.volumeScale;
                aq.speedScale = param.speedScale;
                aq.pitchScale = param.pitchScale;
                aq.speedScale = param.speedScale;
            }

            PostSynthesisQuery(aq, speaker);
        }

        /// <summary>
        /// 音声保存
        /// </summary>
        /// <param name="speaker">話者番号</param>
        /// <param name="param">エフェクト</param>
        /// <param name="text">発声させるテキスト</param>
        /// <param name="WavFilePath">保存するファイル名</param>
        public void Save(int speaker, VoiceVoxParams param, string
text, string WavFilePath)
        {
            VoiceVoxAudioQuery aq = GetAudioQuery(text, speaker);

            if (param != null)
            {
                aq.volumeScale = param.volumeScale;
                aq.speedScale = param.speedScale;
                aq.pitchScale = param.pitchScale;
                aq.speedScale = param.speedScale;
            }

            PostSynthesisQuery(aq, speaker, WavFilePath);
        }

    }

    public class VoiceVoxParams
    {
        public double speedScale;
        public double pitchScale;
        public double intonationScale;
        public double volumeScale;
    }

    [DataContract]
    public class VoiceVoxSpeaker
    {
        [DataMember]
        public string name { get; set; }
        [DataMember]
        public string speaker_uuid { get; set; }
        [DataMember]
        public VoiceVoxSpeakerStyle[] styles { get; set; }
        [DataMember]
        public string version { get; set; }
    }

    [DataContract]
    public class VoiceVoxSpeakerStyle
    {
        [DataMember]
        public string name { get; set; }
        [DataMember]
        public int id { get; set; }
    }

    [DataContract]
    public class VoiceVoxAudioQuery
    {
        [DataMember]
        public string name { get; set; }
        [DataMember]
        public int? id { get; set; }
        [DataMember]
        public VoiceVoxAccentPhrase[] accent_phrases { get; set; }
        [DataMember]
        public double? speedScale { get; set; }
        [DataMember]
        public double? pitchScale { get; set; }
        [DataMember]
        public double? intonationScale { get; set; }
        [DataMember]
        public double? volumeScale { get; set; }
        [DataMember]
        public double? prePhonemeLength { get; set; }
        [DataMember]
        public double? postPhonemeLength { get; set; }
        [DataMember]
        public int? outputSamplingRate { get; set; }
        [DataMember]
        public bool outputStereo { get; set; }
        [DataMember]
        public string kana { get; set; }
    }

    [DataContract]
    public class VoiceVoxAccentPhrase
    {
        [DataMember]
        public VoiceVoxMora[] moras { get; set; }
        [DataMember]
        public int accent { get; set; }
        [DataMember]
        public VoiceVoxPauseMora pause_mora { get; set; }
    }

    [DataContract]
    public class VoiceVoxMora
    {
        [DataMember]
        public string text { get; set; }
        [DataMember]
        public string consonant { get; set; }
        [DataMember]
        public double? consonant_length { get; set; }
        [DataMember]
        public string vowel { get; set; }
        [DataMember]
        public double? vowel_length { get; set; }
        [DataMember]
        public double? pitch { get; set; }
    }

    [DataContract]
    public class VoiceVoxPauseMora
    {
        [DataMember]
        public string text { get; set; }
        [DataMember]
        public string consonant { get; set; }
        [DataMember]
        public double? consonant_length { get; set; }
        [DataMember]
        public string vowel { get; set; }
        [DataMember]
        public double? vowel_length { get; set; }
        [DataMember]
        public double? pitch { get; set; }
    }

}
```

技術資料, Windows, VoiceVox