関係コード４

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

かんたん AITalk 3 2話者版の制御コード

ScDeviceDriver.cs

```
using Codeer.Friendly;
using Codeer.Friendly.Dynamic;
using Codeer.Friendly.Windows;
using Codeer.Friendly.Windows.Grasp;
using Codeer.Friendly.Windows.NativeStandardControls;
using Ong.Friendly.FormsStandardControls;
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Text;
using System.Threading;
using System.Threading.Tasks;

namespace ScDriver.AItalk3Kansai
{
    public class ScDeviceDriver : ScBaseDriver, IScBaseDriver
    {
        private const string DrvName =
"AItalk3Kansai.Driver@echoseika.hgotoh.jp";
        private const string DrvVersion = "20210327/c";
        private const string DrvProdName = "AITalk3Kansai";
        private const int DrvTextMaxLength = 0;
        private const int CidBase = 6010;
        private const int MaxAvators = 10;
        private const int SplitLine = 6;

        private Process          AItalk3Process = null;
        private SemaphoreSlim    Semaphore      = new SemaphoreSlim(1,
1);
        private WindowsAppFriend _app           = null;
```

```
        private WindowControl    uiTreeTop      = null;
        private FormsComboBox    AvatorListx     = null;

        class titles
        {
            public string Avator { get; private set; }
            public int FixedCid { get; private set; }
            public int AvatorIndex { get; private set; }

            public titles(string avator, int cid)
            {
                Avator = avator;
                FixedCid = cid;
                AvatorIndex = 0;
            }
        }

        private readonly titles[] AITalk2Names =
        {
            new titles( "みやび",             CidBase + 1),
            new titles( "やまと",             CidBase + 2)
        };

        public ScDeviceDriver()
        {
            ScDrvName = DrvName;
            ScDrvVersion = DrvVersion;
            ScDrvProdName = DrvProdName;
            ScDrvTextMaxLength = DrvTextMaxLength;
            CidBaseIndex = CidBase;
            AvatorParams = new Dictionary<int, ScDriver.AvatorParam>();

            IsAlive = false;

            AItalk3Process = GetAItalk3Process();

            if (AItalk3Process != null)
            {
                //認識対象外、もしくはユーザ定義プリセットに割り当てるcidのカウンター
                int cbaseCounter = 1 + AITalk2Names.Select(v => v.FixedCid).Max();
                if (cbaseCounter < (CidBase + SplitLine)) cbaseCounter = CidBase + SplitLine;

                try
                {
                    _app = new WindowsAppFriend(AItalk3Process);
                    uiTreeTop = WindowControl.FromZTop(_app);
```

```
                    //音声効果タブ選択
                    FormsTabControl ci = new
FormsTabControl(uiTreeTop.IdentifyFromZIndex(2, 0, 0, 1, 0, 0, 1));
                    ci.EmulateTabSelect(0);  //音声効果
                    Thread.Sleep(10);

                    //リストボックスから話者一覧を取得して処理
                    titles ent = null;
                    AvatorListx = new
FormsComboBox(uiTreeTop.IdentifyFromZIndex(2, 0, 0, 1, 0, 0, 0, 0, 0));
                    for (int avatorIdx = 0; avatorIdx <
AvatorListx.ItemCount; avatorIdx++)
                    {
                        if (cbaseCounter >= (CidBase + MaxAvators))
break;

                        AvatorListx.EmulateChangeSelect(avatorIdx);

                        AItalk3Kansai.AvatorParam avator = new
AItalk3Kansai.AvatorParam();

                        avator.AvatorIndex             = avatorIdx;
                        avator.IndexOfUserList         = avatorIdx;
                        avator.AvatorUI                = new
AItalk3Kansai.AvatorUIParam();
                        avator.AvatorUI.TalkTextBox    = new
WindowControl(uiTreeTop.Dynamic().bk);
                        avator.AvatorUI.PlayButton     = new
FormsButton(uiTreeTop.Dynamic().bo);
                        avator.AvatorUI.SaveButton     = new
FormsButton(uiTreeTop.Dynamic().bq);
                        avator.AvatorUI.VolumeText     = new
FormsTextBox(uiTreeTop.Dynamic().de);
                        avator.AvatorUI.SpeedText      = new
FormsTextBox(uiTreeTop.Dynamic().dd);
                        avator.AvatorUI.PitchText      = new
FormsTextBox(uiTreeTop.Dynamic().dc);
                        avator.AvatorUI.IntonationText = new
FormsTextBox(uiTreeTop.Dynamic().db);

                        avator.AvatorName =
AvatorListx.Dynamic().SelectedItem.b();
                        ////avator.AvatorName =
AvatorListx.Dynamic().SelectedItem.b;

                        //認識している話者名ならば固定のcidを設定
                        try
                        {
                            ent = AITalk2Names.Where(v => v.Avator ==
avator.AvatorName).First();
                            avator.FixedCid = ent.FixedCid;
```

```
                        }
                        catch (Exception)
                        {
                            avator.FixedCid = cbaseCounter;
                            cbaseCounter++;
                        }

                        //このタイミングで一旦登録する
                        AvatorParams.Add(avator.FixedCid, avator);

                        avator.VoiceEffects_default = new
Dictionary<EnumVoiceEffect, EffectValueInfo>
                        {
                            { EnumVoiceEffect.volume,     new
EffectValueInfo(GetSliderValue(avator.FixedCid,
EnumVoiceEffect.volume), 0.0m, 2.0m, 0.01m)},
                            { EnumVoiceEffect.speed,      new
EffectValueInfo(GetSliderValue(avator.FixedCid, EnumVoiceEffect.speed),
0.5m, 4.0m, 0.01m)},
                            { EnumVoiceEffect.pitch,      new
EffectValueInfo(GetSliderValue(avator.FixedCid, EnumVoiceEffect.pitch),
0.5m, 2.0m, 0.01m)},
                            { EnumVoiceEffect.intonation, new
EffectValueInfo(GetSliderValue(avator.FixedCid,
EnumVoiceEffect.intonation), 0.0m, 2.0m, 0.01m)}
                        };
                        avator.VoiceEffects = new
Dictionary<EnumVoiceEffect, EffectValueInfo>
                        {
                            { EnumVoiceEffect.volume,     new
EffectValueInfo(GetSliderValue(avator.FixedCid,
EnumVoiceEffect.volume), 0.0m, 2.0m, 0.01m)},
                            { EnumVoiceEffect.speed,      new
EffectValueInfo(GetSliderValue(avator.FixedCid, EnumVoiceEffect.speed),
0.5m, 4.0m, 0.01m)},
                            { EnumVoiceEffect.pitch,      new
EffectValueInfo(GetSliderValue(avator.FixedCid, EnumVoiceEffect.pitch),
0.5m, 2.0m, 0.01m)},
                            { EnumVoiceEffect.intonation, new
EffectValueInfo(GetSliderValue(avator.FixedCid,
EnumVoiceEffect.intonation), 0.0m, 2.0m, 0.01m)}
                        };
                        avator.VoiceEmotions_default = new
Dictionary<string, EffectValueInfo>();
                        avator.VoiceEmotions = new Dictionary<string,
EffectValueInfo>();

                    }
```

```
                }
                catch (Exception e)
                {
                    ThrowException(string.Format(@"{0} {1}", e.Message,
e.StackTrace));
                }

            }

            IsAlive = AvatorParams.Count != 0;

            // cidエイリアス用
            if (IsAlive)
            {
                AItalk3Kansai.AvatorParam avator = new
AItalk3Kansai.AvatorParam();
                int firsFindtCid = AvatorParams.First().Value.FixedCid;

                avator.FixedCid             = CidBase;
                avator.AvatorIndex          = AvatorParams.Count;
                avator.AvatorName           =
AvatorParams[firsFindtCid].AvatorName;
                avator.AliasCode            = true;
                avator.IndexOfUserList      =
(AvatorParams[firsFindtCid] as
AItalk3Kansai.AvatorParam).IndexOfUserList;
                avator.AvatorUI             =
(AvatorParams[firsFindtCid] as AItalk3Kansai.AvatorParam).AvatorUI;
                avator.VoiceEffects         =
(AvatorParams[firsFindtCid] as AItalk3Kansai.AvatorParam).VoiceEffects;
                avator.VoiceEmotions        =
(AvatorParams[firsFindtCid] as
AItalk3Kansai.AvatorParam).VoiceEmotions;
                avator.VoiceEffects_default  =
(AvatorParams[firsFindtCid] as
AItalk3Kansai.AvatorParam).VoiceEffects_default;
                avator.VoiceEmotions_default =
(AvatorParams[firsFindtCid] as
AItalk3Kansai.AvatorParam).VoiceEmotions_default;

                AvatorParams.Add(avator.FixedCid, avator);
            }

        }

        private Process GetAItalk3Process()
        {
            string WinTitle1 = "かんたん AITalk 3 関西風";
            string WinTitle2 = WinTitle1 + "*";

            int RetryCount = 3;
```

```
            int RetryWaitms = 500;
            Process p = null;

            for (int i = 0; i < 3; i++)
            {
                Process[] ps = Process.GetProcesses();

                foreach (Process pitem in ps)
                {
                    if ((pitem.MainWindowHandle != IntPtr.Zero) &&
                         ((pitem.MainWindowTitle.Equals(WinTitle1)) ||
(pitem.MainWindowTitle.Equals(WinTitle2))))
                    {
                        p = pitem;
                        break;
                    }
                }
                if (p != null) break;
                if (i < (RetryCount - 1)) Thread.Sleep(RetryWaitms);
            }

            return p;
        }

        public void Dispose()
        {
            Dispose(true);
        }

        /// <summary>
        ///指定話者で指定テキストで発声
        /// </summary>
        /// <param name="cid">話者CID</param>
        /// <param name="talkText">発声させるテキスト</param>
        /// <returns>発声にかかった時間（ミリ秒）</returns>
        public override double Play(int cid, string talkText)
        {
            Stopwatch stopWatch = new Stopwatch();
            int avatorIdx = ConvertAvatorIndex(cid);
            AItalk3Kansai.AvatorParam avator = AvatorParams[avatorIdx]
as AItalk3Kansai.AvatorParam;
            Semaphore.Wait();

            if (avator.AvatorUI.PlayButton == null) return 0.0;
            if (avator.AvatorUI.SaveButton == null) return 0.0;
            if (avator.AvatorUI.TalkTextBox == null) return 0.0;

            dynamic AItalk3UiTab = new
FormsTabControl(uiTreeTop.IdentifyFromZIndex(2, 0, 0, 1, 0, 0, 1));
```

```
            AItalk3UiTab.EmulateTabSelect(0);

            AvatorListx.EmulateChangeSelect(avator.IndexOfUserList);
            ApplyEffectParameters(avator.FixedCid);
            ApplyEmotionParameters(avator.FixedCid);

            //再生中なので再生終了を待つ(音声保存ボタンがEnableになるのを待つ)
            if (!avator.AvatorUI.SaveButton.Enabled)
            {
                while (!avator.AvatorUI.SaveButton.Enabled)
                {
                    Thread.Sleep(10);
                }
            }

            avator.AvatorUI.TalkTextBox["Text"](talkText);
            Thread.Sleep(10);

            stopWatch.Start();

            avator.AvatorUI.PlayButton.EmulateClick();

            //再生開始を待つ(音声保存ボタンがDisableになるのを待つ)
            if (avator.AvatorUI.SaveButton.Enabled)
            {
                while (avator.AvatorUI.SaveButton.Enabled)
                {
                    Thread.Sleep(10);
                }
            }

            //再生終了を待つ(音声保存ボタンがEnableになるのを待つ)
            if (!avator.AvatorUI.SaveButton.Enabled)
            {
                while (!avator.AvatorUI.SaveButton.Enabled)
                {
                    Thread.Sleep(10);
                }
            }

            stopWatch.Stop();
            Semaphore.Release();

            return stopWatch.ElapsedMilliseconds;
        }

        /// <summary>
        ///指定話者で指定テキストで発声
        /// </summary>
        /// <param name="cid">話者CID</param>
        /// <param name="talkTexts">発声させるテキスト</param>
```

```
        /// <returns>発声にかかった時間（ミリ秒）</returns>
        public override double Play(int cid, string[] talkTexts)
        {
            string s = String.Join("", talkTexts);
            return Play(cid, s);
        }

        /// <summary>
        ///指定話者で指定テキストで発声
        /// </summary>
        /// <param name="cid">話者CID</param>
        /// <param name="talkText">発声させるテキスト</param>
        public override void PlayAsync(int cid, string talkText)
        {
            int avatorIdx = ConvertAvatorIndex(cid);
            AItalk3Kansai.AvatorParam avator = AvatorParams[avatorIdx]
as AItalk3Kansai.AvatorParam;

            Task.Run(() =>
            {
                Semaphore.Wait();

                if (avator.AvatorUI.PlayButton == null) return;
                if (avator.AvatorUI.SaveButton == null) return;
                if (avator.AvatorUI.TalkTextBox == null) return;

                dynamic AItalk3UiTabUiTab = new
FormsTabControl(uiTreeTop.IdentifyFromZIndex(2, 0, 0, 1, 0, 0, 1));
                AItalk3UiTabUiTab.EmulateTabSelect(0);

AvatorListx.EmulateChangeSelect(avator.IndexOfUserList);
                ApplyEffectParameters(avator.FixedCid);
                ApplyEmotionParameters(avator.FixedCid);

                //再生中なので再生終了を待つ(音声保存ボタンがEnableになるのを待つ)
                if (!avator.AvatorUI.SaveButton.Enabled)
                {
                    while (!avator.AvatorUI.SaveButton.Enabled)
                    {
                        Thread.Sleep(10);
                    }
                }

                avator.AvatorUI.TalkTextBox["Text"](talkText);
                Thread.Sleep(10);

                avator.AvatorUI.PlayButton.EmulateClick();

                //再生開始を待つ(音声保存ボタンがDisableになるのを待つ)
```

```
                if (avator.AvatorUI.SaveButton.Enabled)
                {
                    while (avator.AvatorUI.SaveButton.Enabled)
                    {
                        Thread.Sleep(10);
                    }
                }

                //再生終了を待つ(音声保存ボタンがEnableになるのを待つ)
                if (!avator.AvatorUI.SaveButton.Enabled)
                {
                    while (!avator.AvatorUI.SaveButton.Enabled)
                    {
                        Thread.Sleep(10);
                    }
                }

                Semaphore.Release();
            });
        }

        /// <summary>
        ///指定話者で指定テキストで発声
        /// </summary>
        /// <param name="cid">話者CID</param>
        /// <param name="talkTexts">発声させるテキスト</param>
        public override void PlayAsync(int cid, string[] talkTexts)
        {
            string s = String.Join("", talkTexts);
            PlayAsync(cid, s);
        }

        /// <summary>
        ///指定話者で指定テキストで発声した結果をファイルに保存
        /// </summary>
        /// <param name="cid">話者CID</param>
        /// <param name="talkText">発声させるテキスト</param>
        /// <param name="saveFilename">保存先ファイル名</param>
        /// <returns>0.0ミリ秒固定</returns>
        public override double Save(int cid, string talkText, string
saveFilename)
        {
            int avatorIdx = ConvertAvatorIndex(cid);
            AItalk3Kansai.AvatorParam avator = AvatorParams[avatorIdx]
as AItalk3Kansai.AvatorParam;

            if (avator.AvatorUI.PlayButton == null) return 0.0;
            if (avator.AvatorUI.SaveButton == null) return 0.0;
            if (avator.AvatorUI.TalkTextBox == null) return 0.0;

            dynamic AItalk3UiTabUiTab = new
```

```
FormsTabControl(uiTreeTop.IdentifyFromZIndex(2, 0, 0, 1, 0, 0, 1));
            AItalk3UiTabUiTab.EmulateTabSelect(0);

            AvatorListx.EmulateChangeSelect(avator.IndexOfUserList);
            ApplyEffectParameters(avator.FixedCid);
            ApplyEmotionParameters(avator.FixedCid);

            if (!avator.AvatorUI.SaveButton.Enabled)
            {
                while (!avator.AvatorUI.SaveButton.Enabled)
                {
                    Thread.Sleep(10);
                }
            }

            avator.AvatorUI.TalkTextBox["Text"](talkText);
            Thread.Sleep(10);

            avator.AvatorUI.SaveButton.EmulateClick(new Async());

            bool finish_savefileSetup = false;
            while (finish_savefileSetup == false)
            {
                //名前を付けて保存 ダイアログで名前を設定
                var FileDlgs = WindowControl.GetFromWindowText(_app, "音声ファイルの保存");
                try
                {
                    if ((FileDlgs.Length != 0) && (FileDlgs[0].WindowClassName == "#32770"))
                    {
                        // https://github.com/mikoto2000/TTSController UI特定の記述を参照
                        NativeButton OkButton = new NativeButton(FileDlgs[0].IdentifyFromDialogId(1));
                        NativeEdit SaveNameText = new NativeEdit(FileDlgs[0].IdentifyFromZIndex(11, 0, 4, 0, 0));

                        //ファイル名を設定
                        SaveNameText.EmulateChangeText(saveFilename);
                        Thread.Sleep(100);

                        //OKボタンを押す
                        OkButton.EmulateClick(new Async());
                        Thread.Sleep(100);

                        //上書き確認ダイアログがある？
                        var FileDlgs2 = WindowControl.GetFromWindowText(_app, "音声ファイルの保存");
```

```
                        if ((FileDlgs2.Length != 0) &&
(FileDlgs2[0].WindowClassName == "#32770"))
                        {
                            //
https://github.com/mikoto2000/TTSController UI特定の記述を参照
                            NativeButton YesButton = new
NativeButton(FileDlgs2[0].IdentifyFromDialogId(6)); // YES button

                            //YESボタンを押す
                            YesButton.EmulateClick(new Async());
                            Thread.Sleep(10);
                        }

                        finish_savefileSetup = true;
                    }
                }
                catch (Exception)
                {
                    //
                }

                Thread.Sleep(10);
            }

            return 0.0;
        }

        /// <summary>
        ///指定話者で指定テキストで発声した結果をファイルに保存
        /// </summary>
        /// <param name="cid">話者CID</param>
        /// <param name="talkTexts">発声させるテキスト</param>
        /// <param name="saveFilename">保存先ファイル名</param>
        /// <returns>0.0ミリ秒固定</returns>
        public override double Save(int cid, string[] talkTexts, string
saveFilename)
        {
            string s = String.Join("", talkTexts);
            return Save(cid, s, saveFilename);
        }

        /// <summary>
        ///感情パラメタをデフォルト値に戻す
        /// </summary>
        /// <param name="cid">話者CID</param>
        public override void ResetVoiceEmotion(int cid)
        {
            int avatorIdx = ConvertAvatorIndex(cid);
            AvatorParam avator = AvatorParams[avatorIdx] as
AvatorParam;
```

```
            foreach (var emotion in avator.VoiceEmotions_default)
            {
                avator.VoiceEmotions[emotion.Key].value =
emotion.Value.value;
            }

            ApplyEmotionParameters(avatorIdx);
        }

        /// <summary>
        /// 音声効果をデフォルト値に戻す
        /// </summary>
        /// <param name="cid">話者CID</param>
        public override void ResetVoiceEffect(int cid)
        {
            int avatorIdx = ConvertAvatorIndex(cid);
            AvatorParam avator = AvatorParams[avatorIdx] as
AvatorParam;

            foreach (var effect in avator.VoiceEffects_default)
            {
                avator.VoiceEffects[effect.Key].value =
effect.Value.value;
            }

            ApplyEffectParameters(avatorIdx);
        }

        /// <summary>
        /// 話者切り替え
        /// </summary>
        /// <param name="cid">話者CID</param>
        public override void SetAvator(int cid)
        {
            int avatorIdx = ConvertAvatorIndex(cid);
            AvatorParam avator = AvatorParams[avatorIdx] as
AvatorParam;

            AvatorListx.EmulateChangeSelect(avator.IndexOfUserList);
        }

        public override void Dispose(bool disposing)
        {
            if (Disposed) return;

            if (disposing)
            {
                foreach (var item in AvatorParams)
                {
```

```
                    ResetVoiceEffect(item.Key + CidBase);
                    ResetVoiceEmotion(item.Key + CidBase);
                }

                _app.Dispose();

                AvatorParams.Clear();

                //throw new NotImplementedException();
            }

            Disposed = true;
        }

        private void ApplyEmotionParameters(int avatorIndex)
        {
            //
        }

        private void ApplyEffectParameters(int avatorIdx)
        {
            FormsTextBox TargetTextBox = null;
            double value = 0.00;
            AItalk3Kansai.AvatorParam avator = AvatorParams[avatorIdx]
as AItalk3Kansai.AvatorParam;

            foreach (var effect in avator.VoiceEffects)
            {
                switch (effect.Key)
                {
                    case EnumVoiceEffect.volume:
                        TargetTextBox = avator.AvatorUI.VolumeText;
                        break;

                    case EnumVoiceEffect.speed:
                        TargetTextBox = avator.AvatorUI.SpeedText;
                        break;

                    case EnumVoiceEffect.pitch:
                        TargetTextBox = avator.AvatorUI.PitchText;
                        break;

                    case EnumVoiceEffect.intonation:
                        TargetTextBox = avator.AvatorUI.IntonationText;
                        break;
                }

                value =
Convert.ToDouble(avator.VoiceEffects[effect.Key].value);

                if (TargetTextBox != null)
```

```
                {
TargetTextBox.EmulateChangeText(string.Format("{0:0.00}", value));
                }
            }
        }

        private decimal GetSliderValue(int avatorIdx, EnumVoiceEffect
effect)
        {
            decimal value = 0.00m;
            FormsTextBox TargetTextBox = null;
            AItalk3Kansai.AvatorParam avator = AvatorParams[avatorIdx]
as AItalk3Kansai.AvatorParam;

            switch (effect)
            {
                case EnumVoiceEffect.volume:
                    TargetTextBox = avator.AvatorUI.VolumeText;
                    break;

                case EnumVoiceEffect.speed:
                    TargetTextBox = avator.AvatorUI.SpeedText;
                    break;

                case EnumVoiceEffect.pitch:
                    TargetTextBox = avator.AvatorUI.PitchText;
                    break;

                case EnumVoiceEffect.intonation:
                    TargetTextBox = avator.AvatorUI.IntonationText;
                    break;
            }

            if (TargetTextBox != null)
            {
                value = Convert.ToDecimal(TargetTextBox.Text);
            }

            return value;
        }

    }
}
```

かんたん AITalk 3 5話者版の制御コード

ScDeviceDriver.cs

```
using Codeer.Friendly;
using Codeer.Friendly.Dynamic;
using Codeer.Friendly.Windows;
using Codeer.Friendly.Windows.Grasp;
using Codeer.Friendly.Windows.NativeStandardControls;
using Ong.Friendly.FormsStandardControls;
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Text;
using System.Threading;
using System.Threading.Tasks;

namespace ScDriver.AItalk3
{
    public class ScDeviceDriver : ScBaseDriver, IScBaseDriver
    {
        private const string DrvName =
"AItalk3.Driver@echoseika.hgotoh.jp";
        private const string DrvVersion = "20210327/c";
        private const string DrvProdName = "AITalk3";
        private const int DrvTextMaxLength = 0;
        private const int CidBase = 6000;
        private const int MaxAvators = 10;
        private const int SplitLine = 6;

        private Process          AItalk3Process = null;
        private SemaphoreSlim    Semaphore      = new SemaphoreSlim(1,
1);
        private WindowsAppFriend _app           = null;
        private WindowControl    uiTreeTop      = null;
        private FormsComboBox    AvatorListx     = null;

        class titles
        {
            public string Avator { get; private set; }
            public int FixedCid { get; private set; }
            public int AvatorIndex { get; private set; }

            public titles(string avator, int cid)
            {
                Avator = avator;
                FixedCid = cid;
                AvatorIndex = 0;
            }
        }

        private readonly titles[] AITalk2Names =
```

```
        {
            new titles( "あんず",             CidBase + 1),
            new titles( "かほ",               CidBase + 2),
            new titles( "ななこ",             CidBase + 3),
            new titles( "のぞみ",             CidBase + 4),
            new titles( "せいじ",             CidBase + 5)
            // new titles( "みやび",              CidBase + 6),
            // new titles( "やまと",              CidBase + 7)
        };

        public ScDeviceDriver()
        {
            ScDrvName = DrvName;
            ScDrvVersion = DrvVersion;
            ScDrvProdName = DrvProdName;
            ScDrvTextMaxLength = DrvTextMaxLength;
            CidBaseIndex = CidBase;
            AvatorParams = new Dictionary<int, ScDriver.AvatorParam>();

            IsAlive = false;

            AItalk3Process = GetAItalk3Process();

            if (AItalk3Process != null)
            {
                //認識対象外、もしくはユーザ定義プリセットに割り当てるcidのカウンター
                int cbaseCounter = 1 + AITalk2Names.Select(v => v.FixedCid).Max();
                if (cbaseCounter < (CidBase + SplitLine)) cbaseCounter = CidBase + SplitLine;

                try
                {
                    _app = new WindowsAppFriend(AItalk3Process);
                    uiTreeTop = WindowControl.FromZTop(_app);

                    //音声効果タブ選択
                    FormsTabControl ci = new FormsTabControl(uiTreeTop.IdentifyFromZIndex(2, 0, 0, 1, 0, 0, 1));
                    ci.EmulateTabSelect(0);  //音声効果
                    Thread.Sleep(10);

                    //リストボックスから話者一覧を取得して処理
                    titles ent = null;
                    AvatorListx = new FormsComboBox(uiTreeTop.IdentifyFromZIndex(2, 0, 0, 1, 0, 0, 0, 0, 0));
                    for (int avatorIdx = 0; avatorIdx < AvatorListx.ItemCount; avatorIdx++)
                    {
```

```
                        if (cbaseCounter >= (CidBase + MaxAvators))
break;

                        AvatorListx.EmulateChangeSelect(avatorIdx);

                        AItalk3.AvatorParam avator = new
AItalk3.AvatorParam();

                        avator.AvatorIndex             = avatorIdx;
                        avator.IndexOfUserList         = avatorIdx;
                        avator.AvatorUI                = new
AItalk3.AvatorUIParam();
                        avator.AvatorUI.TalkTextBox    = new
WindowControl(uiTreeTop.Dynamic().ay);
                        avator.AvatorUI.PlayButton     = new
FormsButton(uiTreeTop.Dynamic().a2);
                        avator.AvatorUI.SaveButton     = new
FormsButton(uiTreeTop.Dynamic().a4);
                        avator.AvatorUI.VolumeText     = new
FormsTextBox(uiTreeTop.Dynamic().cs);
                        avator.AvatorUI.SpeedText      = new
FormsTextBox(uiTreeTop.Dynamic().cr);
                        avator.AvatorUI.PitchText      = new
FormsTextBox(uiTreeTop.Dynamic().cq);
                        avator.AvatorUI.IntonationText = new
FormsTextBox(uiTreeTop.Dynamic().cp);

                        avator.AvatorName =
AvatorListx.Dynamic().SelectedItem.ItemText;

                        //認識している話者名ならば固定のcidを設定
                        try
                        {
                            ent = AITalk2Names.Where(v => v.Avator ==
avator.AvatorName).First();
                            avator.FixedCid = ent.FixedCid;
                        }
                        catch (Exception)
                        {
                            avator.FixedCid = cbaseCounter;
                            cbaseCounter++;
                        }

                        //このタイミングで一旦登録する
                        AvatorParams.Add(avator.FixedCid, avator);

                        avator.VoiceEffects_default = new
Dictionary<EnumVoiceEffect, EffectValueInfo>
                        {
                            { EnumVoiceEffect.volume,     new
EffectValueInfo(GetSliderValue(avator.FixedCid,
```

```
EnumVoiceEffect.volume), 0.0m, 2.0m, 0.01m)},
                            { EnumVoiceEffect.speed,      new
EffectValueInfo(GetSliderValue(avator.FixedCid, EnumVoiceEffect.speed),
0.5m, 4.0m, 0.01m)},
                            { EnumVoiceEffect.pitch,      new
EffectValueInfo(GetSliderValue(avator.FixedCid, EnumVoiceEffect.pitch),
0.5m, 2.0m, 0.01m)},
                            { EnumVoiceEffect.intonation, new
EffectValueInfo(GetSliderValue(avator.FixedCid,
EnumVoiceEffect.intonation), 0.0m, 2.0m, 0.01m)}
                        };
                        avator.VoiceEffects = new
Dictionary<EnumVoiceEffect, EffectValueInfo>
                        {
                            { EnumVoiceEffect.volume,     new
EffectValueInfo(GetSliderValue(avator.FixedCid,
EnumVoiceEffect.volume), 0.0m, 2.0m, 0.01m)},
                            { EnumVoiceEffect.speed,      new
EffectValueInfo(GetSliderValue(avator.FixedCid, EnumVoiceEffect.speed),
0.5m, 4.0m, 0.01m)},
                            { EnumVoiceEffect.pitch,      new
EffectValueInfo(GetSliderValue(avator.FixedCid, EnumVoiceEffect.pitch),
0.5m, 2.0m, 0.01m)},
                            { EnumVoiceEffect.intonation, new
EffectValueInfo(GetSliderValue(avator.FixedCid,
EnumVoiceEffect.intonation), 0.0m, 2.0m, 0.01m)}
                        };
                        avator.VoiceEmotions_default = new
Dictionary<string, EffectValueInfo>();
                        avator.VoiceEmotions = new Dictionary<string,
EffectValueInfo>();

                    }

                }
                catch (Exception e)
                {
                    ThrowException(string.Format(@"{0} {1}", e.Message,
e.StackTrace));
                }

            }

            IsAlive = AvatorParams.Count != 0;

            // cidエイリアス用
            if (IsAlive)
            {
                AItalk3.AvatorParam avator = new AItalk3.AvatorParam();
```

```
                int firsFindtCid = AvatorParams.First().Value.FixedCid;

                avator.FixedCid              = CidBase;
                avator.AvatorIndex           = AvatorParams.Count;
                avator.AvatorName            =
AvatorParams[firsFindtCid].AvatorName;
                avator.AliasCode             = true;
                avator.IndexOfUserList       =
(AvatorParams[firsFindtCid] as AItalk3.AvatorParam).IndexOfUserList;
                avator.AvatorUI              =
(AvatorParams[firsFindtCid] as AItalk3.AvatorParam).AvatorUI;
                avator.VoiceEffects          =
(AvatorParams[firsFindtCid] as AItalk3.AvatorParam).VoiceEffects;
                avator.VoiceEmotions         =
(AvatorParams[firsFindtCid] as AItalk3.AvatorParam).VoiceEmotions;
                avator.VoiceEffects_default  =
(AvatorParams[firsFindtCid] as
AItalk3.AvatorParam).VoiceEffects_default;
                avator.VoiceEmotions_default =
(AvatorParams[firsFindtCid] as
AItalk3.AvatorParam).VoiceEmotions_default;

                AvatorParams.Add(avator.FixedCid, avator);
            }

        }

        private Process GetAItalk3Process()
        {
            string WinTitle1 = "かんたん AITalk 3";
            string WinTitle2 = WinTitle1 + "*";

            int RetryCount = 3;
            int RetryWaitms = 500;
            Process p = null;

            for (int i = 0; i < 3; i++)
            {
                Process[] ps = Process.GetProcesses();

                foreach (Process pitem in ps)
                {
                    if ((pitem.MainWindowHandle != IntPtr.Zero) &&
                        ((pitem.MainWindowTitle.Equals(WinTitle1)) ||
(pitem.MainWindowTitle.Equals(WinTitle2))))
                    {
                        p = pitem;
                        break;
                    }
                }
                if (p != null) break;
```

```
                if (i < (RetryCount - 1)) Thread.Sleep(RetryWaitms);
            }

            return p;
        }

        public void Dispose()
        {
            Dispose(true);
        }

        /// <summary>
        ///指定話者で指定テキストで発声
        /// </summary>
        /// <param name="cid">話者CID</param>
        /// <param name="talkText">発声させるテキスト</param>
        /// <returns>発声にかかった時間（ミリ秒）</returns>
        public override double Play(int cid, string talkText)
        {
            Stopwatch stopWatch = new Stopwatch();
            int avatorIdx = ConvertAvatorIndex(cid);
            AItalk3.AvatorParam avator = AvatorParams[avatorIdx] as
AItalk3.AvatorParam;
            Semaphore.Wait();

            if (avator.AvatorUI.PlayButton == null) return 0.0;
            if (avator.AvatorUI.SaveButton == null) return 0.0;
            if (avator.AvatorUI.TalkTextBox == null) return 0.0;

            dynamic AItalk3UiTab = new
FormsTabControl(uiTreeTop.IdentifyFromZIndex(2, 0, 0, 1, 0, 0, 1));
            AItalk3UiTab.EmulateTabSelect(0);

            AvatorListx.EmulateChangeSelect(avator.IndexOfUserList);
            ApplyEffectParameters(avator.FixedCid);
            ApplyEmotionParameters(avator.FixedCid);

            //再生中なので再生終了を待つ(音声保存ボタンがEnableになるのを待つ)
            if (!avator.AvatorUI.SaveButton.Enabled)
            {
                while (!avator.AvatorUI.SaveButton.Enabled)
                {
                    Thread.Sleep(10);
                }
            }

            avator.AvatorUI.TalkTextBox["Text"](talkText);
            Thread.Sleep(10);
```

```
            stopWatch.Start();

            avator.AvatorUI.PlayButton.EmulateClick();

            //再生開始を待つ(音声保存ボタンがDisableになるのを待つ)
            if (avator.AvatorUI.SaveButton.Enabled)
            {
                while (avator.AvatorUI.SaveButton.Enabled)
                {
                    Thread.Sleep(10);
                }
            }

            //再生終了を待つ(音声保存ボタンがEnableになるのを待つ)
            if (!avator.AvatorUI.SaveButton.Enabled)
            {
                while (!avator.AvatorUI.SaveButton.Enabled)
                {
                    Thread.Sleep(10);
                }
            }

            stopWatch.Stop();
            Semaphore.Release();

            return stopWatch.ElapsedMilliseconds;
        }

        /// <summary>
        ///指定話者で指定テキストで発声
        /// </summary>
        /// <param name="cid">話者CID</param>
        /// <param name="talkTexts">発声させるテキスト</param>
        /// <returns>発声にかかった時間（ミリ秒）</returns>
        public override double Play(int cid, string[] talkTexts)
        {
            string s = String.Join("", talkTexts);
            return Play(cid, s);
        }

        /// <summary>
        ///指定話者で指定テキストで発声
        /// </summary>
        /// <param name="cid">話者CID</param>
        /// <param name="talkText">発声させるテキスト</param>
        public override void PlayAsync(int cid, string talkText)
        {
            int avatorIdx = ConvertAvatorIndex(cid);
            AItalk3.AvatorParam avator = AvatorParams[avatorIdx] as
AItalk3.AvatorParam;
```

```
            Task.Run(() =>
            {
                Semaphore.Wait();

                if (avator.AvatorUI.PlayButton == null) return;
                if (avator.AvatorUI.SaveButton == null) return;
                if (avator.AvatorUI.TalkTextBox == null) return;

                dynamic AItalk3UiTabUiTab = new
FormsTabControl(uiTreeTop.IdentifyFromZIndex(2, 0, 0, 1, 0, 0, 1));
                AItalk3UiTabUiTab.EmulateTabSelect(0);

AvatorListx.EmulateChangeSelect(avator.IndexOfUserList);
                ApplyEffectParameters(avator.FixedCid);
                ApplyEmotionParameters(avator.FixedCid);

                //再生中なので再生終了を待つ(音声保存ボタンがEnableになるのを待つ)
                if (!avator.AvatorUI.SaveButton.Enabled)
                {
                    while (!avator.AvatorUI.SaveButton.Enabled)
                    {
                        Thread.Sleep(10);
                    }
                }

                avator.AvatorUI.TalkTextBox["Text"](talkText);
                Thread.Sleep(10);

                avator.AvatorUI.PlayButton.EmulateClick();

                //再生開始を待つ(音声保存ボタンがDisableになるのを待つ)
                if (avator.AvatorUI.SaveButton.Enabled)
                {
                    while (avator.AvatorUI.SaveButton.Enabled)
                    {
                        Thread.Sleep(10);
                    }
                }

                //再生終了を待つ(音声保存ボタンがEnableになるのを待つ)
                if (!avator.AvatorUI.SaveButton.Enabled)
                {
                    while (!avator.AvatorUI.SaveButton.Enabled)
                    {
                        Thread.Sleep(10);
                    }
                }

                Semaphore.Release();
```

```
            });
        }

        /// <summary>
        /// 指定話者で指定テキストで発声
        /// </summary>
        /// <param name="cid">話者CID</param>
        /// <param name="talkTexts">発声させるテキスト</param>
        public override void PlayAsync(int cid, string[] talkTexts)
        {
            string s = String.Join("", talkTexts);
            PlayAsync(cid, s);
        }

        /// <summary>
        /// 指定話者で指定テキストで発声した結果をファイルに保存
        /// </summary>
        /// <param name="cid">話者CID</param>
        /// <param name="talkText">発声させるテキスト</param>
        /// <param name="saveFilename">保存先ファイル名</param>
        /// <returns>0.0ミリ秒固定</returns>
        public override double Save(int cid, string talkText, string
saveFilename)
        {
            int avatorIdx = ConvertAvatorIndex(cid);
            AItalk3.AvatorParam avator = AvatorParams[avatorIdx] as
AItalk3.AvatorParam;

            if (avator.AvatorUI.PlayButton == null) return 0.0;
            if (avator.AvatorUI.SaveButton == null) return 0.0;
            if (avator.AvatorUI.TalkTextBox == null) return 0.0;

            dynamic AItalk3UiTabUiTab = new
FormsTabControl(uiTreeTop.IdentifyFromZIndex(2, 0, 0, 1, 0, 0, 1));
            AItalk3UiTabUiTab.EmulateTabSelect(0);

            AvatorListx.EmulateChangeSelect(avator.IndexOfUserList);
            ApplyEffectParameters(avator.FixedCid);
            ApplyEmotionParameters(avator.FixedCid);

            if (!avator.AvatorUI.SaveButton.Enabled)
            {
                while (!avator.AvatorUI.SaveButton.Enabled)
                {
                    Thread.Sleep(10);
                }
            }

            avator.AvatorUI.TalkTextBox["Text"](talkText);
            Thread.Sleep(10);
```

```
            avator.AvatorUI.SaveButton.EmulateClick(new Async());

            bool finish_savefileSetup = false;
            while (finish_savefileSetup == false)
            {
                //名前を付けて保存 ダイアログで名前を設定
                var FileDlgs = WindowControl.GetFromWindowText(_app, "音声ファイルの保存");
                try
                {
                    if ((FileDlgs.Length != 0) && (FileDlgs[0].WindowClassName == "#32770"))
                    {
                        // https://github.com/mikoto2000/TTSController UI特定の記述を参照
                        NativeButton OkButton = new NativeButton(FileDlgs[0].IdentifyFromDialogId(1));
                        NativeEdit SaveNameText = new NativeEdit(FileDlgs[0].IdentifyFromZIndex(11, 0, 4, 0, 0));

                        //ファイル名を設定
                        SaveNameText.EmulateChangeText(saveFilename);
                        Thread.Sleep(100);

                        //OKボタンを押す
                        OkButton.EmulateClick(new Async());
                        Thread.Sleep(100);

                        //上書き確認ダイアログがある？
                        var FileDlgs2 = WindowControl.GetFromWindowText(_app, "音声ファイルの保存");
                        if ((FileDlgs2.Length != 0) && (FileDlgs2[0].WindowClassName == "#32770"))
                        {
                            // https://github.com/mikoto2000/TTSController UI特定の記述を参照
                            NativeButton YesButton = new NativeButton(FileDlgs2[0].IdentifyFromDialogId(6)); // YES button

                            //YESボタンを押す
                            YesButton.EmulateClick(new Async());
                            Thread.Sleep(10);
                        }

                        finish_savefileSetup = true;
                    }
                }
                catch (Exception)
                {
```

```
                    //
                }

                Thread.Sleep(10);
            }

            return 0.0;
        }

        /// <summary>
        /// 指定話者で指定テキストで発声した結果をファイルに保存
        /// </summary>
        /// <param name="cid">話者CID</param>
        /// <param name="talkTexts">発声させるテキスト</param>
        /// <param name="saveFilename">保存先ファイル名</param>
        /// <returns>0.0ミリ秒固定</returns>
        public override double Save(int cid, string[] talkTexts, string
saveFilename)
        {
            string s = String.Join("", talkTexts);
            return Save(cid, s, saveFilename);
        }

        /// <summary>
        /// 感情パラメタをデフォルト値に戻す
        /// </summary>
        /// <param name="cid">話者CID</param>
        public override void ResetVoiceEmotion(int cid)
        {
            int avatorIdx = ConvertAvatorIndex(cid);
            AvatorParam avator = AvatorParams[avatorIdx] as
AvatorParam;

            foreach (var emotion in avator.VoiceEmotions_default)
            {
                avator.VoiceEmotions[emotion.Key].value =
emotion.Value.value;
            }

            ApplyEmotionParameters(avatorIdx);
        }

        /// <summary>
        /// 音声効果をデフォルト値に戻す
        /// </summary>
        /// <param name="cid">話者CID</param>
        public override void ResetVoiceEffect(int cid)
        {
            int avatorIdx = ConvertAvatorIndex(cid);
            AvatorParam avator = AvatorParams[avatorIdx] as
AvatorParam;
```

```
            foreach (var effect in avator.VoiceEffects_default)
            {
                avator.VoiceEffects[effect.Key].value =
effect.Value.value;
            }

            ApplyEffectParameters(avatorIdx);
        }

        /// <summary>
        ///話者切り替え
        /// </summary>
        /// <param name="cid">話者CID</param>
        public override void SetAvator(int cid)
        {
            int avatorIdx = ConvertAvatorIndex(cid);
            AvatorParam avator = AvatorParams[avatorIdx] as
AvatorParam;

            AvatorListx.EmulateChangeSelect(avator.IndexOfUserList);
        }

        public override void Dispose(bool disposing)
        {
            if (Disposed) return;

            if (disposing)
           {
                foreach (var item in AvatorParams)
                {
                    ResetVoiceEffect(item.Key + CidBase);
                    ResetVoiceEmotion(item.Key + CidBase);
                }

                _app.Dispose();

                AvatorParams.Clear();

                //throw new NotImplementedException();
            }

            Disposed = true;
        }

        private void ApplyEmotionParameters(int avatorIndex)
        {
            //
        }
```

```
        private void ApplyEffectParameters(int avatorIdx)
        {
            FormsTextBox TargetTextBox = null;
            double value = 0.00;
            AItalk3.AvatorParam avator = AvatorParams[avatorIdx] as
AItalk3.AvatorParam;

            foreach (var effect in avator.VoiceEffects)
            {
                switch (effect.Key)
                {
                    case EnumVoiceEffect.volume:
                        TargetTextBox = avator.AvatorUI.VolumeText;
                        break;

                    case EnumVoiceEffect.speed:
                        TargetTextBox = avator.AvatorUI.SpeedText;
                        break;

                    case EnumVoiceEffect.pitch:
                        TargetTextBox = avator.AvatorUI.PitchText;
                        break;

                    case EnumVoiceEffect.intonation:
                        TargetTextBox = avator.AvatorUI.IntonationText;
                        break;
                }

                value =
Convert.ToDouble(avator.VoiceEffects[effect.Key].value);

                if (TargetTextBox != null)
                {
TargetTextBox.EmulateChangeText(string.Format("{0:0.00}", value));
                }
            }
        }

        private decimal GetSliderValue(int avatorIdx, EnumVoiceEffect
effect)
        {
            decimal value = 0.00m;
            FormsTextBox TargetTextBox = null;
            AItalk3.AvatorParam avator = AvatorParams[avatorIdx] as
AItalk3.AvatorParam;

            switch (effect)
            {
                case EnumVoiceEffect.volume:
                    TargetTextBox = avator.AvatorUI.VolumeText;
```

```
                    break;

                case EnumVoiceEffect.speed:
                    TargetTextBox = avator.AvatorUI.SpeedText;
                    break;

                case EnumVoiceEffect.pitch:
                    TargetTextBox = avator.AvatorUI.PitchText;
                    break;

                case EnumVoiceEffect.intonation:
                    TargetTextBox = avator.AvatorUI.IntonationText;
                    break;
            }

            if (TargetTextBox != null)
            {
                value = Convert.ToDecimal(TargetTextBox.Text);
            }

            return value;
        }

    }
}
```

技術資料, Windows, AITalk3, Codeer.Friendly