関係コード５

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

A.I.VOICEの制御コード

20210723/u公開時のもの

ScDeviceDriver.cs

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading;
using System.Threading.Tasks;
using System.Windows.Forms;
using Codeer.Friendly;
using Codeer.Friendly.Dynamic;
using Codeer.Friendly.Windows;
using Codeer.Friendly.Windows.Grasp;
using Codeer.Friendly.Windows.NativeStandardControls;
using RM.Friendly.WPFStandardControls;

namespace ScDriver.AIVOICE
{
    public class ScDeviceDriver : ScBaseDriver, IScBaseDriver
    {
        private const string DrvName =
"AiVoice.Driver@echoseika.hgotoh.jp";
        private const string DrvVersion = "20210517/c";
        private const string DrvProdName = "AIVOICE";
        private const int DrvTextMaxLength = 0;
        private const int CidBase = 5200;
        private const int MaxAvators = 500;
        private const int SplitLine = 100;

        public SemaphoreSlim Semaphore = new SemaphoreSlim(1, 1);

        private WindowsAppFriend _app = null;
```

```
        private WindowControl uiTreeTop = null;
        private WPFTabControl VoicePresetTab = null;
        private WPFTabControl TuneTab = null;
        private WPFTextBox TalkTextBox = null;
        private WPFButtonBase PlayButton = null;
        private WPFButtonBase SaveButton = null;
        private WPFListView AvatorListView_std = null;
        private WPFListView AvatorListView_usr = null;
        private WPFTabControl TextBoxTab = null;

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

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

        private readonly titles[] AIVoice2Names =
        {
            new titles(0, "琴葉 茜",                CidBase + 1),
            new titles(0, "琴葉 茜（蕾）",          CidBase + 2),
            new titles(0, "琴葉 葵",                CidBase + 3),
            new titles(0, "琴葉 葵（蕾）",          CidBase + 4),
            new titles(0, "伊織 弓鶴",              CidBase + 5),
            new titles(0, "伊織 弓鶴（冥）",        CidBase + 6),
            new titles(0, "結月 ゆかり",            CidBase + 7),
            new titles(0, "結月ゆかり",             CidBase + 7),
            new titles(0, "紲星 あかり",            CidBase + 9),
            new titles(0, "紲星あかり",             CidBase + 9),
            new titles(0, "民安 ともえ",            CidBase + 11),
            new titles(0, "民安ともえ",             CidBase + 11),
            new titles(0, "結城 香",                CidBase + 13),
            new titles(0, "日ノ出 賢",              CidBase + 14),
            new titles(0, "潮崎 かずき",            CidBase + 15)
        };

        public ScDeviceDriver()
        {
            ScDrvName = DrvName;
            ScDrvVersion = DrvVersion;
```

```
            ScDrvProdName = DrvProdName;
            ScDrvTextMaxLength = DrvTextMaxLength;
            CidBaseIndex = CidBase;
            AvatorParams = new Dictionary<int, ScDriver.AvatorParam>();

            IsAlive = false;

            Process p = GetVoiceroidEditorProcess();

            if (p != null)
            {
                try
                {
                    _app = new WindowsAppFriend(p);
                }
                catch (Exception)
                {
                    p = null;
                }
            }

            if (p != 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 + AIVoice2Names.Select(v => v.FixedCid).Max();
                if (cbaseCounter < (CidBase + SplitLine)) cbaseCounter = CidBase + SplitLine;

                try
                {
                    uiTreeTop = WindowControl.FromZTop(_app);

                    //判明しているGUI要素特定（暫定判定）

                    var tabs = uiTreeTop.GetFromTypeFullName("AI.Framework.Wpf.Controls.TitledTabControl");

                    if (tabs.Length == 2) // 1.1.0 より前
                    {
                        VoicePresetTab = new WPFTabControl(tabs[0]);  //ボイス（プリセット）のタブコントロール
                        TuneTab = new WPFTabControl(tabs[1]);         //チューニングのタブコントロール

                        var editUis = uiTreeTop.GetFromTypeFullName("AI.Talk.Editor.TextEditView")[0].LogicalTree();
                        TalkTextBox = new WPFTextBox(editUis[4]);     //テキストボックス
```

```
                        PlayButton = new WPFButtonBase(editUis[6]);   //
再生ボタン
                        SaveButton = new WPFButtonBase(editUis[24]);  //
音声保存ボタン
                    }
                    else if (tabs.Length == 3) //1.1.0以降
                    {
                        VoicePresetTab = new WPFTabControl(tabs[0]);  //
ボイス（プリセット）のタブコントロール
                        TextBoxTab = new WPFTabControl(tabs[1]);      //
テキスト/リストのタブコントロール
                        TuneTab = new WPFTabControl(tabs[2]);         //
チューニングのタブコントロール

                        TextBoxTab.EmulateChangeSelectedIndex(0); //テキ
ストタブに切り替えておく

                        var editUis =
TextBoxTab.LogicalTree().ByType("System.Windows.Controls.TabItem")[0].L
ogicalTree();
                        TalkTextBox = new WPFTextBox(editUis[4]);     //
テキストボックス
                        PlayButton = new WPFButtonBase(editUis[6]);   //
再生ボタン
                        SaveButton = new WPFButtonBase(editUis[28]);  //
音声保存ボタン
                    }

                    //標準タブにいる各話者毎のGUI要素データを取得
                    TuneTab.EmulateChangeSelectedIndex(1);
                    VoicePresetTab.EmulateChangeSelectedIndex(0);
                    AvatorListView_std = new
WPFListView(uiTreeTop.GetFromTypeFullName("System.Windows.Controls.List
View")[0]);
                    ScanPreset(AvatorListView_std, 0, 0, ref
cbaseCounter);

                    //ユーザータブにいる各話者プリセット毎のGUI要素データを取得
                    TuneTab.EmulateChangeSelectedIndex(1);
                    VoicePresetTab.EmulateChangeSelectedIndex(1);
                    AvatorListView_usr = new
WPFListView(uiTreeTop.GetFromTypeFullName("System.Windows.Controls.List
View")[1]);
                    ScanPreset(AvatorListView_usr,
AvatorListView_std.ItemCount, 1, ref cbaseCounter);
                }
                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)
            {
                AIVOICE.AvatorParam avator = new AIVOICE.AvatorParam();
                int firsFindtCid = AvatorParams.First().Value.FixedCid;

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

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

        private Process GetVoiceroidEditorProcess()
        {
            string winTitle1 = "A.I.VOICE Editor - ";

            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.StartsWith(winTitle1))
                    {
                        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)
        {
            Semaphore.Wait();

            Stopwatch sw = new Stopwatch();
            bool iconFind = false;
            int avatorIndex = ConvertAvatorIndex(cid);

            //最後の確認ダイアログを殺し切れていなかった時のための処理
            var infoOldDlgs = WindowControl.GetFromWindowText(_app, "情
報");
            try
            {
                if ((infoOldDlgs.Length != 0) &&
(infoOldDlgs[0].WindowClassName == "#32770"))
                {
                    //OKボタンを押す
                    NativeButton btn = new
NativeButton(infoOldDlgs[0].IdentifyFromWindowClass("Button"));
                    btn.EmulateClick(new Async());
                    Thread.Sleep(10);
                }
            }
            catch (Exception)
            {
                //
            }

            if (PlayButton == null) return 0.0;
```

```
            if (SaveButton == null) return 0.0;
            if (TalkTextBox == null) return 0.0;

            TuneTab.EmulateChangeSelectedIndex(1);

            //話者切り替え
            AvatorSelect(avatorIndex);

            //音声保存ボタンを使った再生終了判定を止めて、再生ボタンのアイコンの状態で判定する方法に変更する。
            var items01 = PlayButton.LogicalTree(TreeRunDirection.Descendants).ByType("System.Windows.Controls.Image");
            dynamic playButtonImage1 = items01[0].Dynamic(); //再生アイコン。このアイコンが有効時？の判定はまだないな…
            dynamic playButtonImage2 = items01[1].Dynamic(); //停止アイコン。処理ではこのアイコンのプロパティを持ている

            ApplyEffectParameters(avatorIndex);
            ApplyEmotionParameters(avatorIndex);

            //再生中なので再生終了を待つ
            //※再生ボタンのアイコンが再生アイコンに切り替わるのを待つ
            iconFind = playButtonImage2.IsVisible;
            if (iconFind)
            {
                while (iconFind)
                {
                    Thread.Sleep(10);
                    iconFind = playButtonImage2.IsVisible;
                }
            }

            TextBoxTab?.EmulateChangeSelectedIndex(0); //テキストタブに切り替えておく
            TalkTextBox.EmulateChangeText(talkText);
            Thread.Sleep(10);

            sw.Start();

            PlayButton.EmulateClick();

            //再生開始を待つ
            //再生ボタンのアイコンが停止アイコンに切り替わるのを待つ
            iconFind = playButtonImage2.IsVisible;
            if (!iconFind)
            {
                while (!iconFind)
                {
                    Thread.Sleep(10);
                    iconFind = playButtonImage2.IsVisible;
                }
```

```
            }

            //再生終了を待つ
            //再生ボタンのアイコンが再生アイコンに切り替わるのを待つ
            iconFind = playButtonImage2.IsVisible;
            if (iconFind)
            {
                while (iconFind)
                {
                    Thread.Sleep(10);
                    iconFind = playButtonImage2.IsVisible;
                }
            }

            sw.Stop();
            Semaphore.Release();

            return sw.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>
        /// <returns>発声にかかった時間（ミリ秒）</returns>
        public override void PlayAsync(int cid, string talkText)
        {
            Task.Run(() =>
            {
                Semaphore.Wait();

                bool iconFind = false;
                int avatorIndex = ConvertAvatorIndex(cid);

                //最後の確認ダイアログを殺し切れていなかった時のための処理
                var infoOldDlgs = WindowControl.GetFromWindowText(_app,
```

```
"情報");
                try
                {
                    if ((infoOldDlgs.Length != 0) &&
(infoOldDlgs[0].WindowClassName == "#32770"))
                    {
                        //OKボタンを押す
                        NativeButton btn = new
NativeButton(infoOldDlgs[0].IdentifyFromWindowClass("Button"));
                        btn.EmulateClick(new Async());
                        Thread.Sleep(10);
                    }
                }
                catch (Exception)
                {
                    //
                }

                if (PlayButton == null) return;
                if (SaveButton == null) return;
                if (TalkTextBox == null) return;

                TuneTab.EmulateChangeSelectedIndex(1);

                //話者切り替え
                AvatorSelect(avatorIndex);

                //音声保存ボタンを使った再生終了判定を止めて、再生ボタンのアイコンの状態で判定する方法に変更する。
                var items01 =
PlayButton.LogicalTree(TreeRunDirection.Descendants).ByType("System.Windows.Controls.Image");
                dynamic playButtonImage1 = items01[0].Dynamic(); //再生アイコン。このアイコンが有効時？の判定はまだないな…
                dynamic playButtonImage2 = items01[1].Dynamic(); //停止アイコン。処理ではこのアイコンのプロパティを持ている

                ApplyEffectParameters(avatorIndex);
                ApplyEmotionParameters(avatorIndex);

                //再生中なので再生終了を待つ
                //※再生ボタンのアイコンが再生アイコンに切り替わるのを待つ
                iconFind = playButtonImage2.IsVisible;
                if (iconFind)
                {
                    while (iconFind)
                    {
                        Thread.Sleep(10);
                        iconFind = playButtonImage2.IsVisible;
                    }
                }
```

```
                TextBoxTab?.EmulateChangeSelectedIndex(0); //テキストタブに切り替えておく
                TalkTextBox.EmulateChangeText(talkText);
                Thread.Sleep(10);

                PlayButton.EmulateClick();

                //再生開始を待つ
                //再生ボタンのアイコンが停止アイコンに切り替わるのを待つ
                iconFind = playButtonImage2.IsVisible;
                if (!iconFind)
                {
                    while (!iconFind)
                    {
                        Thread.Sleep(10);
                        iconFind = playButtonImage2.IsVisible;
                    }
                }

                //再生終了を待つ
                //再生ボタンのアイコンが再生アイコンに切り替わるのを待つ
                iconFind = playButtonImage2.IsVisible;
                if (iconFind)
                {
                    while (iconFind)
                    {
                        Thread.Sleep(10);
                        iconFind = playButtonImage2.IsVisible;
                    }
                }

                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>
        /// <returns>発声にかかった時間（ミリ秒）</returns>
        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 avatorIndex = ConvertAvatorIndex(cid);

            //最後の確認ダイアログを殺し切れていなかった時のための処理
            var infoOldDlgs = WindowControl.GetFromWindowText(_app, "情
報");
            try
            {
                if ((infoOldDlgs.Length != 0) &&
(infoOldDlgs[0].WindowClassName == "#32770"))
                {
                    //OKボタンを押す
                    NativeButton btn = new
NativeButton(infoOldDlgs[0].IdentifyFromWindowClass("Button"));
                    btn.EmulateClick(new Async());
                    Thread.Sleep(10);
                }
            }
            catch (Exception)
            {
                //
            }

            if (PlayButton == null) return 0.0;
            if (SaveButton == null) return 0.0;
            if (TalkTextBox == null) return 0.0;

            TuneTab.EmulateChangeSelectedIndex(1);

            //話者切り替え
            AvatorSelect(avatorIndex);

            ApplyEffectParameters(avatorIndex);
            ApplyEmotionParameters(avatorIndex);

            if (!SaveButton.IsEnabled)
            {
                while (!SaveButton.IsEnabled)
                {
                    Thread.Sleep(10);
                }
            }

            TextBoxTab?.EmulateChangeSelectedIndex(0); //テキストタブに切
```

```
り替えておく
            TalkTextBox.EmulateChangeText(talkText);
            Thread.Sleep(10);

            //AIVOICE Editorの音声保存ボタンを押す
            SaveButton.EmulateClick(new Async());

            bool finish_savefileSetup = false;
            bool skip_saveOptionDlg = false;
            while (finish_savefileSetup == false)
            {
                //フォルダー参照の設定ダイアログ処理
                //音声保存の設定ダイアログ処理
                var folderDlgs = WindowControl.GetFromWindowText(_app,
"フォルダーの参照");
                var settingDlgs = WindowControl.GetFromWindowText(_app,
"音声保存");
                try
                {
                    if ((folderDlgs.Length != 0) &&
(folderDlgs[0].WindowClassName == "#32770"))
                    {
                        //OKボタンを押す
                        NativeButton btn = new
NativeButton(folderDlgs[0].IdentifyFromZIndex(2));
                        btn.EmulateClick(new Async());
                    }

                    if ((!skip_saveOptionDlg) && (settingDlgs.Length !=
0))
                    {
                        //ファイル保存ダイアログで選択させる設定に切替てOKボタン
を押す
                        WPFToggleButton radioBtn = new
WPFToggleButton(settingDlgs[0].IdentifyFromLogicalTreeIndex(0, 0, 0, 0,
11, 1, 8));
                        WPFButtonBase btn = new
WPFButtonBase(settingDlgs[0].IdentifyFromLogicalTreeIndex(0, 1, 0));
                        radioBtn.EmulateCheck(true);
                        btn.EmulateClick(new Async());
                        skip_saveOptionDlg = true;
                    }
                }
                catch (Exception)
                {
                    //
                }

                //名前を付けて保存 ダイアログで名前を設定
                var fileDlgs1 = WindowControl.GetFromWindowText(_app, "
```

```
名前を付けて保存");
                Thread.Sleep(10);
                try
                {
                    if ((fileDlgs1.Length != 0) &&
(fileDlgs1[0].WindowClassName == "#32770"))
                    {
                        // https://github.com/mikoto2000/TTSController
UI特定の記述を参照
                        NativeButton btn = new
NativeButton(fileDlgs1[0].IdentifyFromDialogId(1));
                        NativeEdit saveNameText = new
NativeEdit(fileDlgs1[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
                        btn.EmulateClick(new Async());
                        Thread.Sleep(100);  //短くするとまずいかも

                        //上書き確認ダイアログがある？
                        var FileDlgs2 =
WindowControl.GetFromWindowText(_app, "名前を付けて保存");
                        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);
            }

            bool finish_fileSave = false;
            int waitCount = 0;
```

```
            while ((finish_fileSave == false) && (waitCount < 200)) //
メッセージ表示レベルを「簡潔」にしてしまう人対策
            {
                //最後の確認ダイアログの処理
                var infoDlgs = WindowControl.GetFromWindowText(_app, "情
報");
                try
                {
                    if ((infoDlgs.Length != 0) &&
(infoDlgs[0].WindowClassName == "#32770"))
                    {
                        //OKボタンを押す
                        NativeButton btn = new
NativeButton(infoDlgs[0].IdentifyFromWindowClass("Button"));
                        btn.EmulateClick(new Async());
                        finish_fileSave = true;
                    }
                }
                catch (Exception)
                {
                    //
                }

                Thread.Sleep(10);
                waitCount++;
            }

            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 avatorIndex = ConvertAvatorIndex(cid);
            AvatorParam avator = AvatorParams[avatorIndex] as
AvatorParam;

            foreach (KeyValuePair<string, EffectValueInfo> item in
avator.VoiceEmotions_default)
            {
                avator.VoiceEmotions[item.Key].value =
item.Value.value;
            }

            ApplyEmotionParameters(avatorIndex);
        }

        /// <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 avatorIndex = ConvertAvatorIndex(cid);
            AvatorParam avator = AvatorParams[avatorIndex] as
AvatorParam;

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

            ApplyEffectParameters(avatorIndex);
        }

        /// <summary>
        /// 話者切り替え
        /// </summary>
        /// <param name="cid">話者CID</param>
        public override void SetAvator(int cid)
        {
            int avatorIndex = ConvertAvatorIndex(cid);

            AvatorSelect(avatorIndex);
        }

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

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

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

                AvatorParams.Clear();
                _app.Dispose();
            }

            Disposed = true;
        }

        private void ScanPreset(WPFListView avatorListView, int
idxBase, int presetTabIndex, ref int cbaseCounter)
        {
            titles ent = null;

            for (int avatorIdx = 0; avatorIdx <
avatorListView.ItemCount; avatorIdx++)
            {
                if (cbaseCounter >= (CidBase + MaxAvators)) break;

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

                avator.AvatorIndex = idxBase + avatorIdx;

                avator.AvatorUI = new AIVOICE.AvatorUIParam();
                avator.AvatorUI.PresetTabIndex = presetTabIndex;
                avator.AvatorUI.IndexOnPresetTab = avatorIdx;
                avator.AvatorUI.EmotionSliderIndexs = new
Dictionary<string, int>();
                avator.AvatorUI.EmotionSliders = new Dictionary<int,
WPFSlider>();

                avator.VoiceEmotions = new Dictionary<string,
EffectValueInfo>();
                avator.VoiceEmotions_default = new Dictionary<string,
EffectValueInfo>();

                avatorListView.EmulateChangeSelectedIndex(avatorIdx); //
リスト内のインデクスになる
                TuneTab.EmulateChangeSelectedIndex(1); //ボイスタブ

                //プリセット名取得（話者名）
                var params1 =
TuneTab.VisualTree(TreeRunDirection.Descendants).ByType("AI.Framework.W
pf.Controls.TextBoxEx")[0];
                WPFTextBox nameTextBox = new WPFTextBox(params1);

                avator.AvatorName = nameTextBox.Text;
```

```
                    //認識している話者名ならば固定のcidを設定
                    try
                    {
                        ent = AIVoice2Names.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);

                    //スライダーの配列を取得(共通)
                    try
                    {
                        TuneTab.EmulateChangeSelectedIndex(1); //チューニング:
ボイスタブ
                        var params2 =
TuneTab.VisualTree(TreeRunDirection.Descendants).ByType("System.Windows
.Controls.Slider");
                        avator.AvatorUI.VolumeSlider = new
WPFSlider(params2[0]);
                        avator.AvatorUI.SpeedSlider = new
WPFSlider(params2[1]);
                        avator.AvatorUI.PitchSlider = new
WPFSlider(params2[2]);
                        avator.AvatorUI.IntonationSlider = new
WPFSlider(params2[3]);
                        avator.AvatorUI.ShortPauseSlider = new
WPFSlider(params2[4]);
                        avator.AvatorUI.LongPauseSlider = new
WPFSlider(params2[5]);
                        avator.AvatorUI.WithEmotionParams = false;

                        avator.VoiceEffects_default = new
Dictionary<EnumVoiceEffect, EffectValueInfo>
                        {
                            {EnumVoiceEffect.volume,     new
EffectValueInfo(Convert.ToDecimal(avator.AvatorUI.VolumeSlider.Value),
Convert.ToDecimal(avator.AvatorUI.VolumeSlider.Minimum),
Convert.ToDecimal(avator.AvatorUI.VolumeSlider.Maximum), 0.01m)},
                            {EnumVoiceEffect.speed,      new
EffectValueInfo(Convert.ToDecimal(avator.AvatorUI.SpeedSlider.Value),
Convert.ToDecimal(avator.AvatorUI.SpeedSlider.Minimum),
Convert.ToDecimal(avator.AvatorUI.SpeedSlider.Maximum), 0.01m)},
                            {EnumVoiceEffect.pitch,      new
```

```
EffectValueInfo(Convert.ToDecimal(avator.AvatorUI.PitchSlider.Value),
Convert.ToDecimal(avator.AvatorUI.PitchSlider.Minimum),
Convert.ToDecimal(avator.AvatorUI.PitchSlider.Maximum), 0.01m)},
                        {EnumVoiceEffect.intonation, new
EffectValueInfo(Convert.ToDecimal(avator.AvatorUI.IntonationSlider.Valu
e), Convert.ToDecimal(avator.AvatorUI.IntonationSlider.Minimum),
Convert.ToDecimal(avator.AvatorUI.IntonationSlider.Maximum), 0.01m)},
                        {EnumVoiceEffect.shortpause, new
EffectValueInfo(Convert.ToDecimal(avator.AvatorUI.ShortPauseSlider.Valu
e), Convert.ToDecimal(avator.AvatorUI.ShortPauseSlider.Minimum),
Convert.ToDecimal(avator.AvatorUI.ShortPauseSlider.Maximum), 1.00m)},
                        {EnumVoiceEffect.longpause,  new
EffectValueInfo(Convert.ToDecimal(avator.AvatorUI.LongPauseSlider.Value
), Convert.ToDecimal(avator.AvatorUI.LongPauseSlider.Minimum),
Convert.ToDecimal(avator.AvatorUI.LongPauseSlider.Maximum), 1.00m)}
                    };
                    avator.VoiceEffects = new
Dictionary<EnumVoiceEffect, EffectValueInfo>
                    {
                        {EnumVoiceEffect.volume,     new
EffectValueInfo(Convert.ToDecimal(avator.AvatorUI.VolumeSlider.Value),
Convert.ToDecimal(avator.AvatorUI.VolumeSlider.Minimum),
Convert.ToDecimal(avator.AvatorUI.VolumeSlider.Maximum), 0.01m)},
                        {EnumVoiceEffect.speed,      new
EffectValueInfo(Convert.ToDecimal(avator.AvatorUI.SpeedSlider.Value),
Convert.ToDecimal(avator.AvatorUI.SpeedSlider.Minimum),
Convert.ToDecimal(avator.AvatorUI.SpeedSlider.Maximum), 0.01m)},
                        {EnumVoiceEffect.pitch,      new
EffectValueInfo(Convert.ToDecimal(avator.AvatorUI.PitchSlider.Value),
Convert.ToDecimal(avator.AvatorUI.PitchSlider.Minimum),
Convert.ToDecimal(avator.AvatorUI.PitchSlider.Maximum), 0.01m)},
                        {EnumVoiceEffect.intonation, new
EffectValueInfo(Convert.ToDecimal(avator.AvatorUI.IntonationSlider.Valu
e), Convert.ToDecimal(avator.AvatorUI.IntonationSlider.Minimum),
Convert.ToDecimal(avator.AvatorUI.IntonationSlider.Maximum), 0.01m)},
                        {EnumVoiceEffect.shortpause, new
EffectValueInfo(Convert.ToDecimal(avator.AvatorUI.ShortPauseSlider.Valu
e), Convert.ToDecimal(avator.AvatorUI.ShortPauseSlider.Minimum),
Convert.ToDecimal(avator.AvatorUI.ShortPauseSlider.Maximum), 1.00m)},
                        {EnumVoiceEffect.longpause,  new
EffectValueInfo(Convert.ToDecimal(avator.AvatorUI.LongPauseSlider.Value
), Convert.ToDecimal(avator.AvatorUI.LongPauseSlider.Minimum),
Convert.ToDecimal(avator.AvatorUI.LongPauseSlider.Maximum), 1.00m)}
                    };
                }
                catch (Exception ep2)
                {
                    ThrowException(string.Format("ep2 fail. unknown
gui(LinearFader capture).{0}", ep2.Message));
```

```
                }

                //スライダーの配列を取得(スタイル)
                try
                {
                    var params3 =
TuneTab.VisualTree(TreeRunDirection.Descendants).ByType("System.Windows
.Controls.ListBox").Single();

                    WPFListBox sliders = new WPFListBox(params3);

                    if ((sliders != null) && (sliders.ItemCount != 0))
                    {
                        for (int sidx = 0; sidx < sliders.ItemCount;
sidx++)
                        {
                            var sitem = sliders.GetItem(sidx);
                            var textblocks =
sitem.VisualTree().ByType("System.Windows.Controls.TextBlock");
                            WPFSlider slider = new
WPFSlider(sitem.VisualTree().ByType("AI.Framework.Wpf.Controls.LinearFa
der").Single());
                            WPFTextBlock emoname = new
WPFTextBlock(textblocks[textblocks.Count > 2 ? (textblocks.Count - 2) :
0]);

avator.VoiceEmotions_default.Add(emoname.Text, new
EffectValueInfo(Convert.ToDecimal(slider.Value),
Convert.ToDecimal(slider.Minimum), Convert.ToDecimal(slider.Maximum),
0.01m));
                            avator.VoiceEmotions.Add(emoname.Text, new
EffectValueInfo(Convert.ToDecimal(slider.Value),
Convert.ToDecimal(slider.Minimum), Convert.ToDecimal(slider.Maximum),
0.01m));
avator.AvatorUI.EmotionSliderIndexs.Add(emoname.Text, sidx);
                            avator.AvatorUI.EmotionSliders.Add(sidx,
slider); //将来のため保持しているだけ。
                        }

                        avator.AvatorUI.WithEmotionParams = true;
                    }
                }
                catch (Exception ep3)
                {
                    ThrowException(string.Format("ep3 fail. unknown
gui(Style LinearFader capture).{0}", ep3.Message));
                }

            }
        }
```

```
        private void AvatorSelect(int avatorIndex)
        {
            AIVOICE.AvatorParam avator = AvatorParams[avatorIndex] as
AIVOICE.AvatorParam;

            if (AvatorListView_std == null) return;
            if (AvatorListView_usr == null) return;

VoicePresetTab.EmulateChangeSelectedIndex(avator.AvatorUI.PresetTabInde
x);
            switch (avator.AvatorUI.PresetTabIndex)
            {
                case 0:
AvatorListView_std.EmulateChangeSelectedIndex(avator.AvatorUI.IndexOnPr
esetTab);
                    break;

                case 1:
AvatorListView_usr.EmulateChangeSelectedIndex(avator.AvatorUI.IndexOnPr
esetTab);
                    break;
            }
        }

        private void ApplyEmotionParameters(int avatorIndex)
        {
            AIVOICE.AvatorParam avator = AvatorParams[avatorIndex] as
AIVOICE.AvatorParam;

            //スタイルを持っている話者なら処理する
            if (avator.AvatorUI.WithEmotionParams)
            {
                TuneTab.EmulateChangeSelectedIndex(1); //ボイスタブ
                WPFListBox emoList = new
WPFListBox(TuneTab.VisualTree(TreeRunDirection.Descendants).ByType("Sys
tem.Windows.Controls.ListBox").Single());

                foreach (KeyValuePair<string, EffectValueInfo> item in
avator.VoiceEmotions)
                {
                    double p =
Convert.ToDouble(avator.VoiceEmotions[item.Key].value);
                    int eidx =
avator.AvatorUI.EmotionSliderIndexs[item.Key];
                    WPFSlider slider = new
WPFSlider(emoList.GetItem(eidx).VisualTree().ByType("AI.Framework.Wpf.C
ontrols.LinearFader").Single());
                    slider["Value"](p);
                }
```

```
            }
        }

        private void ApplyEffectParameters(int avatorIndex)
        {
            WPFSlider slider = null;
            AIVOICE.AvatorParam avator = AvatorParams[avatorIndex] as
AIVOICE.AvatorParam;

            TuneTab.EmulateChangeSelectedIndex(1);

            foreach (KeyValuePair<EnumVoiceEffect, EffectValueInfo>
item in avator.VoiceEffects)
            {
                switch (item.Key)
                {
                    case EnumVoiceEffect.volume:
                        slider = avator.AvatorUI.VolumeSlider;
                        break;

                    case EnumVoiceEffect.speed:
                        slider = avator.AvatorUI.SpeedSlider;
                        break;

                    case EnumVoiceEffect.pitch:
                        slider = avator.AvatorUI.PitchSlider;
                        break;

                    case EnumVoiceEffect.intonation:
                        slider = avator.AvatorUI.IntonationSlider;
                        break;

                    case EnumVoiceEffect.shortpause:
                        slider = avator.AvatorUI.ShortPauseSlider;
                        break;

                    case EnumVoiceEffect.longpause:
                        slider = avator.AvatorUI.LongPauseSlider;
                        break;
                }

                double p =
Convert.ToDouble(avator.VoiceEffects[item.Key].value);

                if (slider != null) slider.EmulateChangeValue(p);
            }
        }

        private decimal GetSliderValue(int avatorIndex, EnumVoiceEffect
ef)
        {
```

```
            decimal ans = 0.00m;
            WPFSlider slider = null;
            AIVOICE.AvatorParam avator = AvatorParams[avatorIndex] as
AIVOICE.AvatorParam;

            AvatorSelect(avatorIndex);
            TuneTab.EmulateChangeSelectedIndex(1);

            switch (ef)
            {
                case EnumVoiceEffect.volume:
                    slider = avator.AvatorUI.VolumeSlider;
                    break;

                case EnumVoiceEffect.speed:
                    slider = avator.AvatorUI.SpeedSlider;
                    break;

                case EnumVoiceEffect.pitch:
                    slider = avator.AvatorUI.PitchSlider;
                    break;

                case EnumVoiceEffect.intonation:
                    slider = avator.AvatorUI.IntonationSlider;
                    break;

                case EnumVoiceEffect.shortpause:
                    slider = avator.AvatorUI.ShortPauseSlider;
                    break;

                case EnumVoiceEffect.longpause:
                    slider = avator.AvatorUI.LongPauseSlider;
                    break;
            }

            ans = Convert.ToDecimal(slider.Value);

            return ans;
        }

        private decimal GetSliderValue(int avatorIndex, string emotion)
        {
            AIVOICE.AvatorParam avator = AvatorParams[avatorIndex] as
AIVOICE.AvatorParam;

            AvatorSelect(avatorIndex);
            TuneTab.EmulateChangeSelectedIndex(1);

            if
```

```
(!avator.AvatorUI.EmotionSliderIndexs.ContainsKey(emotion))
            {
                ThrowException("Effect Slider not found");
            }

            WPFSlider slider =
avator.AvatorUI.EmotionSliders[avator.AvatorUI.EmotionSliderIndexs[emot
ion]];

            return Convert.ToDecimal(slider.Value);
        }

    }
}
```

技術資料, Windows, A.I.VOICE, Codeer.Friendly