WindowSystem Documentation

リリース ***2.00***

Seikichi Kanemura,KsSoft

2019 年 **05** 月 **21** 日

目次

第 1 章

特徴:

1.1 特徴

ウィンドウベースのシステム

ウィンドウベースのシステムによって、画面遷移などが楽になります。また、ウィンドウ同士をスタックすることが可能なので、ウィンドウを開く前の状態に戻すことが簡単です。

複数人で分業しても破綻しないシステム

複数人で開発するとき、多くの場合バージョンコントロールソフト (git や svn) を使います。この時、マージがし辛い従来の GUI システムでは問題が発生しやすいです。当ウィンドウシステムは、独自の GUI 構築スクリプト言語を用意し、ウィンドウを簡易に記述できるようにしています。これによってマージしやすい環境を提供します。

ウィンドウデータとプログラムを完全に分離

強力で簡単/シンプルなウィンドウ記述言語

ウィンドウを記述するためのスクリプトの文法はシンプルです。このスクリプト言語の特徴は、簡単なプロパティを設定する程度のものです。ただ、単純といえども十分パワフルで、ラジオボタンの排他制御、タブ制御等記述可能になっています。また、ウィンドウオープン/クローズ時のアニメーションなどもスクリプト内に記述できます。

柔軟なレイアウト

画面サイズが変わってもレイアウトが崩れないように、座標の指定を柔軟に設定可能になっています。ほとんどの位置、オフセット、サイズに親のサイズからの割合率を指定することが可能になっています。

開発効率の良いテクスチャアトラス

当ウィンドウシステムがサポートするテクスチャアトラスは、従来の Unity の持つテクスチャアトラスを一段進化させたものになっています。テクスチャアトラス化するときパッチ方法を個別に指定することができます。パッチを指定することによって、スプライトを拡大/縮小してレンダリングする際に、両

端を固定し真ん中だけ伸ばしたりすることが可能になります。また、アトラス化する際、テクスチャを 32bit から 16bit に自動的にディザリングを用いて減色することも可能です。

高速なレンダリング

複数のコントロールを一つのメッシュにまとめてレンダリングするようになっています。ドローコールの最適化は自動的に行われ、複雑な画面でも十分快適な GUI を提供することが可能になっています。

多言語対応可能なキャプション

文字リソースを外部から渡すことが可能になっており、多言語対応が可能なシステムになっています。

ウィンドウシステムは、ウィンドウを編集する環境とウィンドウを実際使ってアプリケーションを構築する環境を分離 (別プロジェクトに) することが可能になっています。これによって、ウィンドウのレイアウトを修正している間も、アプリケーションを自由に作ることが可能になっています。当然、一つの環境で開発することも可能です。

第 2 章

チュートリアル:

2.1 環境構築方法

2.1.1 MS-Windows 環境:cygwin インストール

本ウィンドウシステムでは、スクリプトをコンパイルしてウィンドウデータとします。スクリプトをコンパイルするときに、プリプロセッサとして gcc を使用しています。

MS-Windows 環境では、以下の手順で cygwin をインストールしてください。

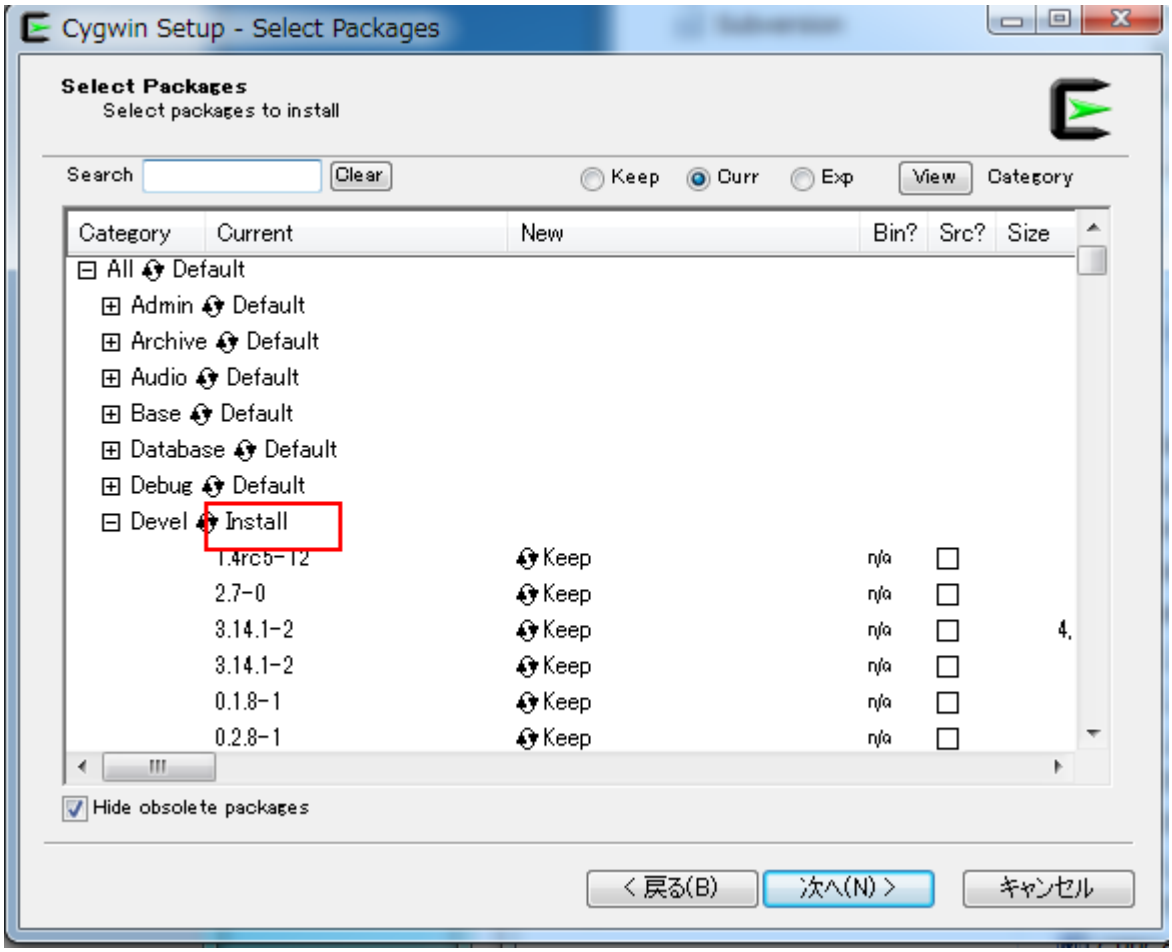
cygwin をダウンロード

次のアドレスから setup-x86.exe 若しくは setup-x86_64.exe をダウンロードしてください。

http://www.cygwin.com/

Devel パッケージを選択しインストール

Select Packages の項目で図のように、Devel を **Default** → **Install** に変更してください。



インストール

後は指定通りインストールしてください。

パスを通す

インストールしたフォルダを基準に、bin のパスを環境変数のシステムパスに追加してください。

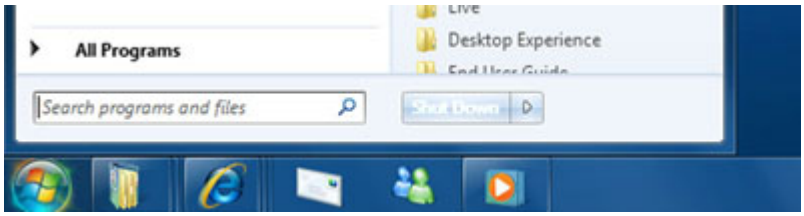
デフォルトでは、下記のフォルダになっています。

```
C:\cygwin\bin or  c:\cygwin64\bin(64bit version)
```

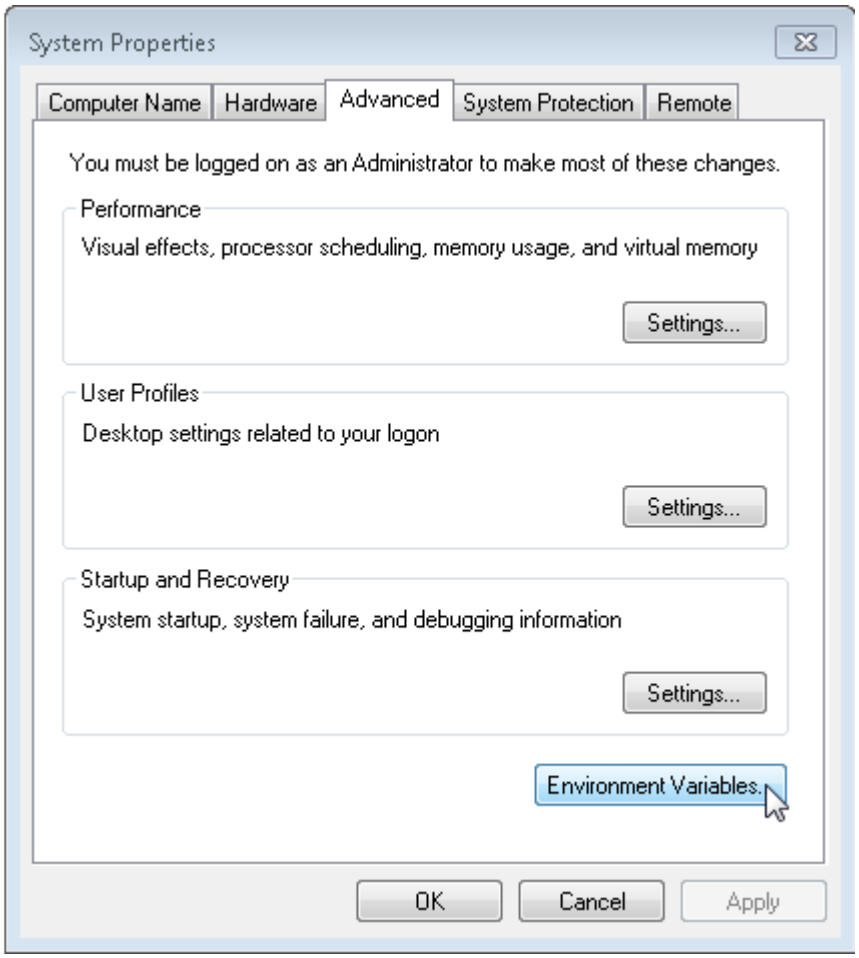
注釈: この環境変数は、cygwin の環境変数ではなく、Windows のシステム環境変数です。

詳しい手順は次の通りです。

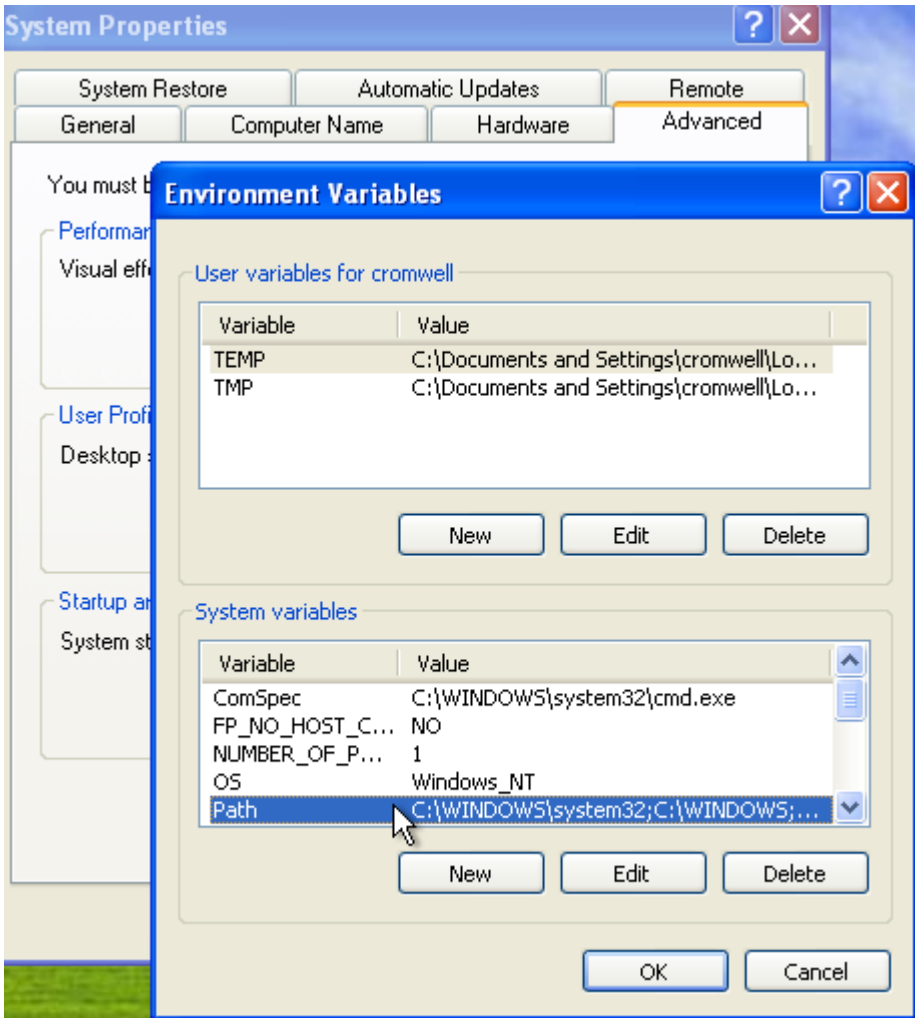
スタートメニューを押してください。



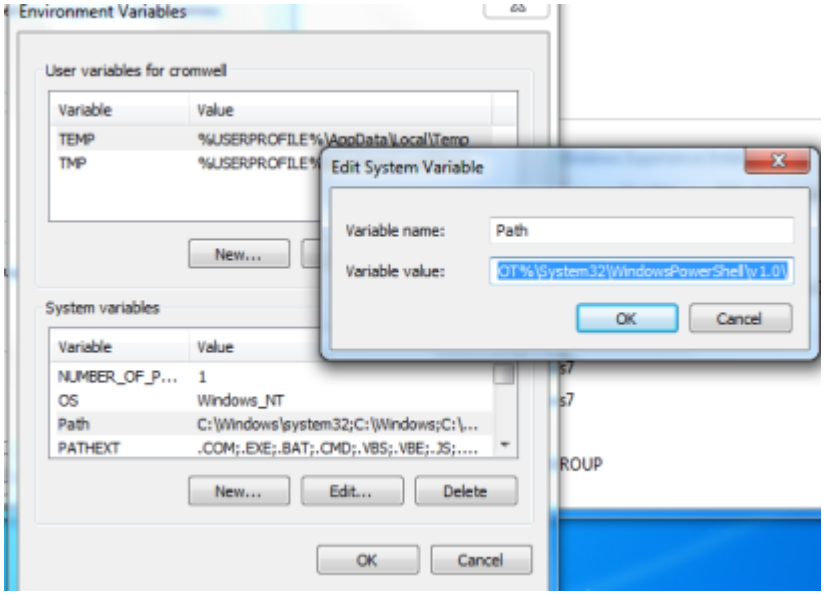
「プログラムとファイルの検索」に、sysdm.cpl を入力し、「システムプロパティ」を開いてください。



詳細設定パネルを選択し、「環境変数」ボタンをクリックしてください。



次に、システム環境変数のリストから、スクロールダウンして Path という値を探してください。見つけたら Path という値をクリックし選択後、もう一度クリックすると次のようなエディットボタンが開きます。



ウィンドウが開いたら、現在の値がハイライトされているはずです。現在の値を消したり、変更しないように気を付けてください。あなたの設定を破壊してしまいます。

恐らくデフォルトの設定では以下のような値が設定されているかと思われます。

```
%SystemRoot%\system32;%SystemRoot%;%SystemRoot%\System32\Wbem
```

セミコロンで値が区切られているので、デフォルトでは三つの要素から構成されています。

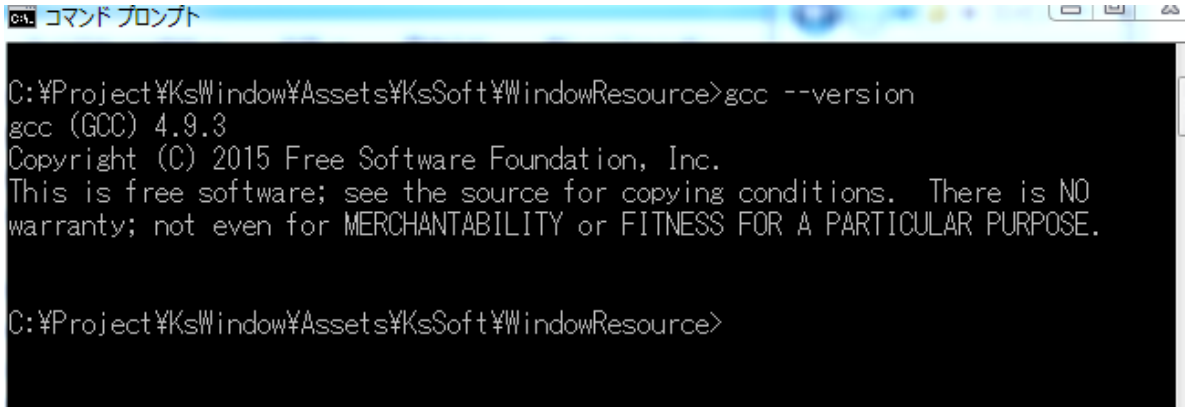
```
%SystemRoot%\system32
%SystemRoot%
%SystemRoot%\System32\Wbem
```

あなたが ";c:\cygwin\bin" を最後に追加したいなら、次のようになります。しかしこれはあくまでも例であって、あなたの環境に合わせて従来の設定を変えずに最後に追加するようにしてください。

```
%SystemRoot%\system32;%SystemRoot%;%SystemRoot%\System32\Wbem;c:\cygwin\bin
```

確認

ウィンドウズコマンドプロンプト (cygwin の bash では確認になりません) を開いて、gcc –version と打ち込んでみてください。正しくインストールできていれば、以下のように表示されるはずです。



同様の結果を得られないときは、二つのことを確認してください。

1. gcc.exe が C:\cygwin\bin ディレクトリに存在するか確認してください。もし存在しなければ、 インストール手順 を見直してください。
2. gcc.exe が所定のディレクトリに存在しているなら、 パス が正しく設定されているかもう一度確認しなおして下さい。

2.1.2 OSX 環境:Xcode のコマンドラインツールをインストール

ターミナル上で gcc が見つからないときは、各自 Xcode からコマンドラインツールをインストールしてください。

2.1.3 多言語対応を行うための準備 (MS 環境専用)

Python 3.x インストール

多言語対応のための文字リソースを作るときに、MS-Excel を使います。MS-Excel から、データコンバートするためのツールは、Python で記述されています。

最初に Python3.x をインストールしてください。

Python のパスを通す

インストールしたフォルダを基準に、パスを環境変数のシステムパスに追加してください。

パスの設定方法は、こちら を参照ください。

デフォルトでは、次のパスになっています。パスは、インストールした Python のバージョンによって異なります。

```
C:\Python3x
```

pywin32 インストール

python から Windows API に対してアクセスするためのファイルを読み込むために、pywin32 を使っています。各自インストールしてください。コマンドラインから次ようにしてインストールしてください。

```
pip install pywin32
```

2.2 サンプル

2.2.1 サンプルシーン

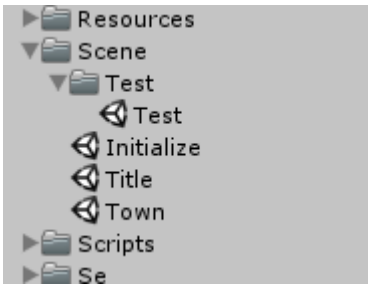
二つのサンプルシーンを用意しています。

- 各コントロールの振る舞い確認できるサンプル

  KsSoft/Scene/Test/Test.unity

- 実際のゲームを想定したサンプル

  KsSoft/Scene/Initialize.unity



各コントロールの振る舞い確認できるサンプル (Test.unity)

ほぼすべてのコントロールの振る舞いを確認できるサンプルです。

Console ログには各種コールバック (onClick,onDrag 等々) のログが出力するように作られています。コールバックが実際にどのタイミング発行されるかを確認できます。また、ドラッグ可能なコントロールは、全てドラッグできるようにスクリプト上で設定されています。

関係するコードは、次の通りです。

- KsSoft/Script/Test/Test.cs
- KsSoft/Script/Test/CWinTest.cs
- KsSoft/Script/Test/CWinTestBase.cs
- KsSoft/WindowResource/CWinTest.wra

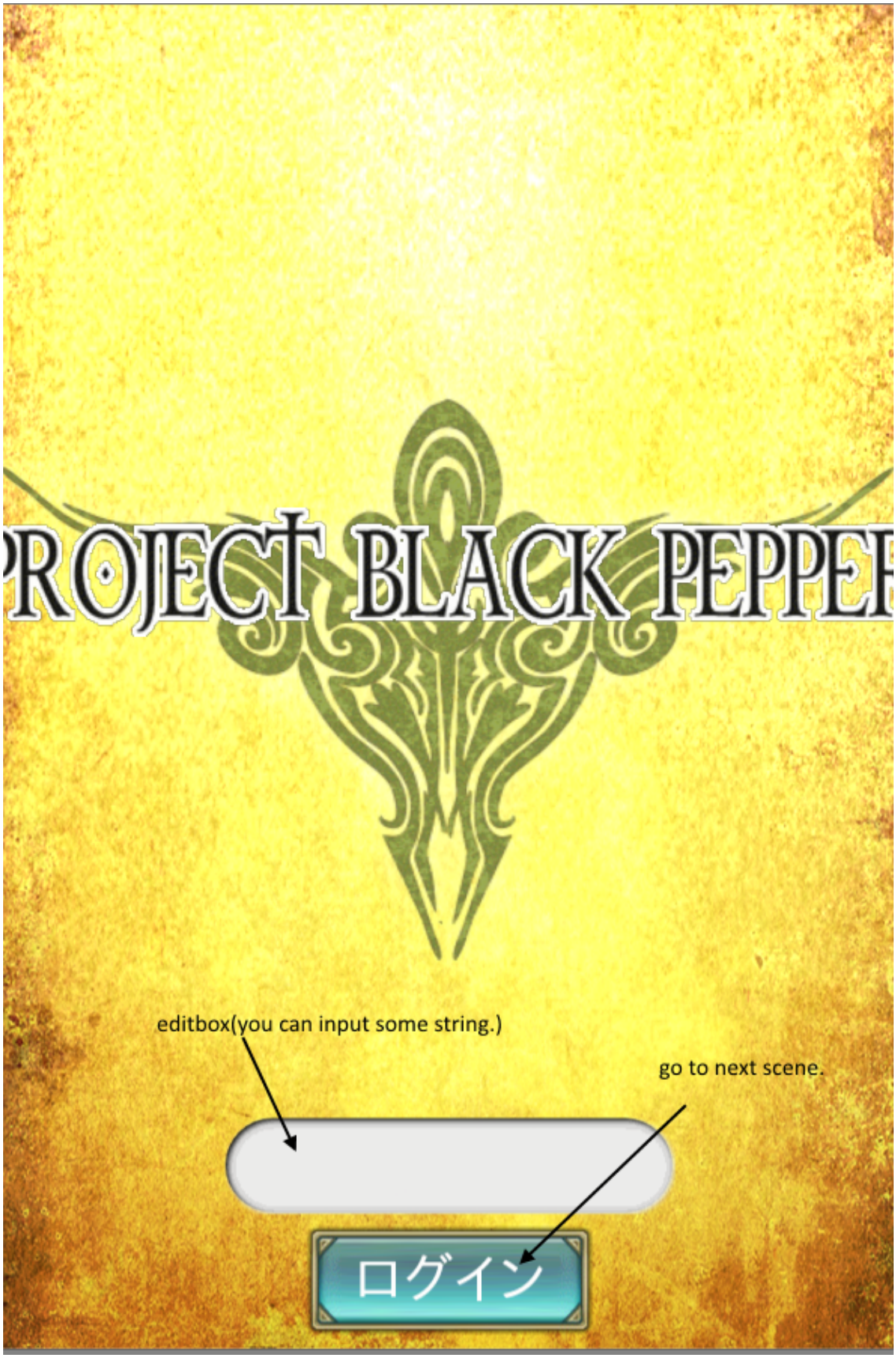
注釈: CWinTitle.wra を編集後、その動きを確認したいときは、実行前に *Export Window Resource* を呼び忘れないようにしてください。

実際のゲームを想定したサンプル **(Initialize.unity)**

2.2.2 タイトル画面

CWinTitle.cs

- KsSoft/Script/Scene/Title/CWinTitle.cs
- KsSoft/Script/Scene/Title/CWinTitleBase.cs
- KsSoft/WindowResource/CWinTitle.wra


PROJECT BLACK PEPPER
editbox(you can input some string.)
go to next scene.
ログイン

2.2.3 ホーム画面

CWinHome

- KsSoft/Script/Scene/Town/Home/CWinHome.cs
- KsSoft/Script/Scene/Town/Home/CWinHomeBase.cs
- KsSoft/WindowResource/CWinHome.wra

ギルドボタン、フレンドボタン、トレジャーボタン、画面中央のキャラクタをレンダリングしています。*RENDER* の使い方の参考になります。

CWinTopPart

- KsSoft/Script/Scene/Town/CWinTopPart.cs
- KsSoft/Script/Scene/Town/CWinTopPartBase.cs
- KsSoft/WindowResource/CWinTopPart.wra

画面上部のキャラステータスや、レンダーアイコン部分を担当しています。

RENDERICON の使い方の参考になります。

CWinBottomPart

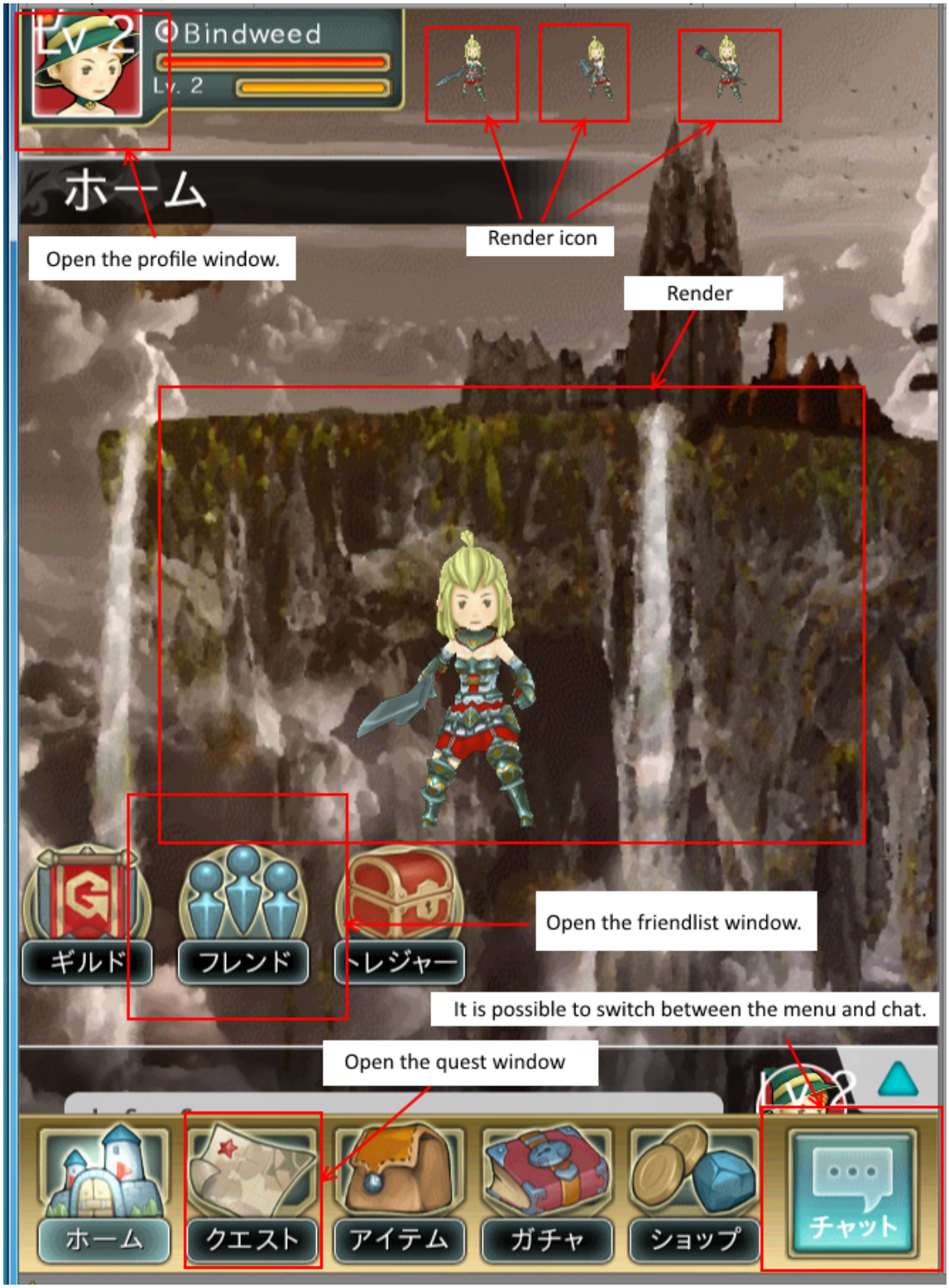
- KsSoft/Script/Scene/Town/CWinBottomPart.cs
- KsSoft/Script/Scene/Town/CWinBottomPartBase.cs
- KsSoft/WindowResource/CWinBottomPart.wra

画面下部のチャットボタンとメニューボタンのトグル部分を担当しています。

CWinTabbar

- KsSoft/Script/Scene/Town/CWinTabbar.cs
- KsSoft/Script/Scene/Town/CWinTabbarBase.cs
- KsSoft/WindowResource/CWinTabbar.wra

画面下部のメニューアイコン部分を担当してます。



2.2.4 チャットウィンドウ

CWinChat

- KsSoft/Script/Chat/CWinChat.cs
- KsSoft/Script/Chat/CWinChatBase.cs
- KsSoft/WindowResource/CWinChat.wra

CWinChatInput

- KsSoft/Script/Chat/CWinChatInput.cs
- KsSoft/Script/Chat/CWinChatInputBase.cs
- KsSoft/WindowResource/CWinChatInput.wra

CWinChatPerson

- KsSoft/Script/Chat/CWinChatPerson.cs
- KsSoft/Script/Chat/CWinChatPersonBase.cs
- KsSoft/WindowResource/CWinChatPerson.wra

チャットウィンドウの作り方の一例です。チャットウィンドウの機能としては、ログウィンドウサイズの変更や、発言者に対応するアイコンを表示したりする方法のサンプルになっています。

2.2.5 プロフィールウィンドウ

CWinProfile

- KsSoft/Script/Scene/Town/Profile/CWinProfile.cs
- KsSoft/Script/Scene/Town/Profile/CWinProfileBase.cs
- KsSoft/WindowResource/CWinProfile.wra

横方向のリストボックスや、スワイプによるページ送りのサンプルになっています。また、 *RENDER* の参考にもなります。

2.2.6 フレンドリストウィンドウ

CWinFriend,CWinFriendList,CWinFriendApply,CWinFriendBalcklist

- KsSoft/Script/Scene/Town/Friend/CWinFriend.cs
- KsSoft/Script/Scene/Town/Friend/CWinFriendBase.cs
- KsSoft/WindowResource/CWinFriend.wra

- KsSoft/Script/Scene/Town/Friend/CWinFriendList.cs
- KsSoft/Script/Scene/Town/Friend/CWinFriendListBase.cs
- KsSoft/WindowResource/CWinFriendList.wra
- KsSoft/Script/Scene/Town/Friend/CWinFriendApply.cs
- KsSoft/Script/Scene/Town/Friend/CWinFriendApplyBase.cs
- KsSoft/WindowResource/CWinFriendApply.wra
- KsSoft/Script/Scene/Town/Friend/CWinFriendBalcklist.cs
- KsSoft/Script/Scene/Town/Friend/CWinFriendBalcklistBase.cs
- KsSoft/WindowResource/CWinFriendBalcklist.wra

リストボックスのコンテンツ内容をアップデートする方法の一例です。

2.2.7 フレンドリストウィンドウ

CWinQuest

- KsSoft/Script/Friend/CWinFriend.cs
- KsSoft/Script/Friend/CWinFriendBase.cs
- KsSoft/WindowResource/CWinFriend.wra
- KsSoft/Script/Friend/CWinFriendList.cs
- KsSoft/Script/Friend/CWinFriendListBase.cs
- KsSoft/WindowResource/CWinFriendList.wra
- KsSoft/Script/Friend/CWinFriendApply.cs
- KsSoft/Script/Friend/CWinFriendApplyBase.cs
- KsSoft/WindowResource/CWinFriendApply.wra
- KsSoft/Script/Friend/CWinFriendBalcklist.cs
- KsSoft/Script/Friend/CWinFriendBalcklistBase.cs
- KsSoft/WindowResource/CWinFriendBalcklist.wra

リストボックスのコンテンツ内容をアップデートする方法の一例です。

2.2.8 クエストウィンドウ

CWinQuest

- KsSoft/Script/Scene/Quest/CWinQuest.cs
- KsSoft/Script/Scene/Quest/CWinQuestBase.cs
- KsSoft/WindowResource/CWinQuest.wra

ラジオボタンにコンテンツを関連付けて、タブのように振る舞わせる方法の例です。


Lv 2
Bindweed
Lv. 2
クエスト
ソロクエスト
ソロ
マルチ
イベント
ホーム
クエスト
アイテム
ガチャ
ショップ
チャット

2.3 チュートリアル

2.3.1 アセットバンドルをサーバーから落としてくる VS ストリーミングアセットを使う

ストリーミングアセットを使う (デフォルト)

クライアントにアセットバンドルを封入しておくには、StreamingAssets を使うことになります。アセットバンドルをストリーミングするときは、次のように設定してください。

Assets/KsSoft/Plugins/KsSoftConfig.cs

```
public class KsSoftConfig : KsSoftConfigBase {
    static public void initialize() {
      m_UseStreaming = true;
    }
```

デフォルトではこちらになっています。

アセットバンドルをサーバーからダウンロードする.

あなたのアプリケーションが、アセットバンドルを用いて HTTP 経由でデータを落としてくる必要があるなら次のように設定してください。

Assets/KsSoft/Plugins/KsSoftConfig.cs

```
public class KsSoftConfig : KsSoftConfigBase {
    static public void initialize() {
      m_UseStreaming = false;
    }
```

2.3.2 wra ファイルをコンパイルする

環境設定 を、済ましているか確認してください。

済ましているなら、KsSoft/WindowResource/を選択した状態で、右クリック →Export を選んでください。

全ての wra ファイルがコンパイルされます。

エラーが出でたときには、 環境設定 を見直してください。

2.3.3 全てのアセットバンドルを出力する

[メニュー] から [Tools]->[KsSoft]->[Export All]->[Windows,Mac]

注釈: Mac 環境で試すときは、Xcode のコマンドラインツールがインストールしているか確認してください。

注釈: リソースを使う 設定 になっているときは、アセットバンドルではなく、リソースデータを出力します。

2.3.4 スクリプト作成/編集方法

1. Unity Project を起動します。
2. プロジェクト内の WindowResource フォルダ (Assets/KsSoft/WindowResource) の直下に wra ファイルを作成します。
3. wra ファイル上で右クリックし、Export を選びます。

   好みのエディタで編集してください。

   ファイル名は、 ウィンドウ名.wra(例: CMessageWindow.wra) という形式でつけてください。ファイルのエンコーディングは、UTF8 です。

wra ファイル/wrb ファイル

wra ファイルは、Window Resource Ascii の略です。ウィンドウのコントロールとビューを定義するためのスクリプトファイルです。

wrb ファイルは、Window Resource Binary の略です。**wra** ファイルを Unity Editor 上でコンパイルに成功すると、対応した wrb ファイルが生成されます。

インクルードパス

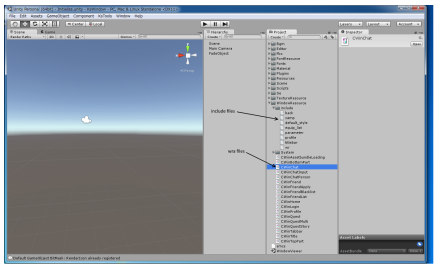
以下のパスにインクルードパスが、通っています。インクルードファイルを作成する際は、ファイルをここに置いてください。

```
Assets/KsSoft/WindowResource/include
```

wra ファイルをコンパイルするときにインクルードされるファイル

次のファイルは、wra ファイルをコンパイルするときに、自動的にインクルードされます。共通の設定や定数は、このファイルに設定されています。

```
Assets/KsSoft/WindowResource/include/wr.h
```



コンパイルされたファイルの場所

Assets/KsSoft/WindowResource/wrb というフォルダの直下に、*.wrb というファイルが生成されています。このファイルがコンパイルされたファイルです。

2.3.5 ウィンドウビューワー

コンパイルしたウィンドウをビューワーですぐ確認できます。

また、一度コンパイルに成功したウィンドウはビューワーを起動しっ放しでも変更を確認できます。

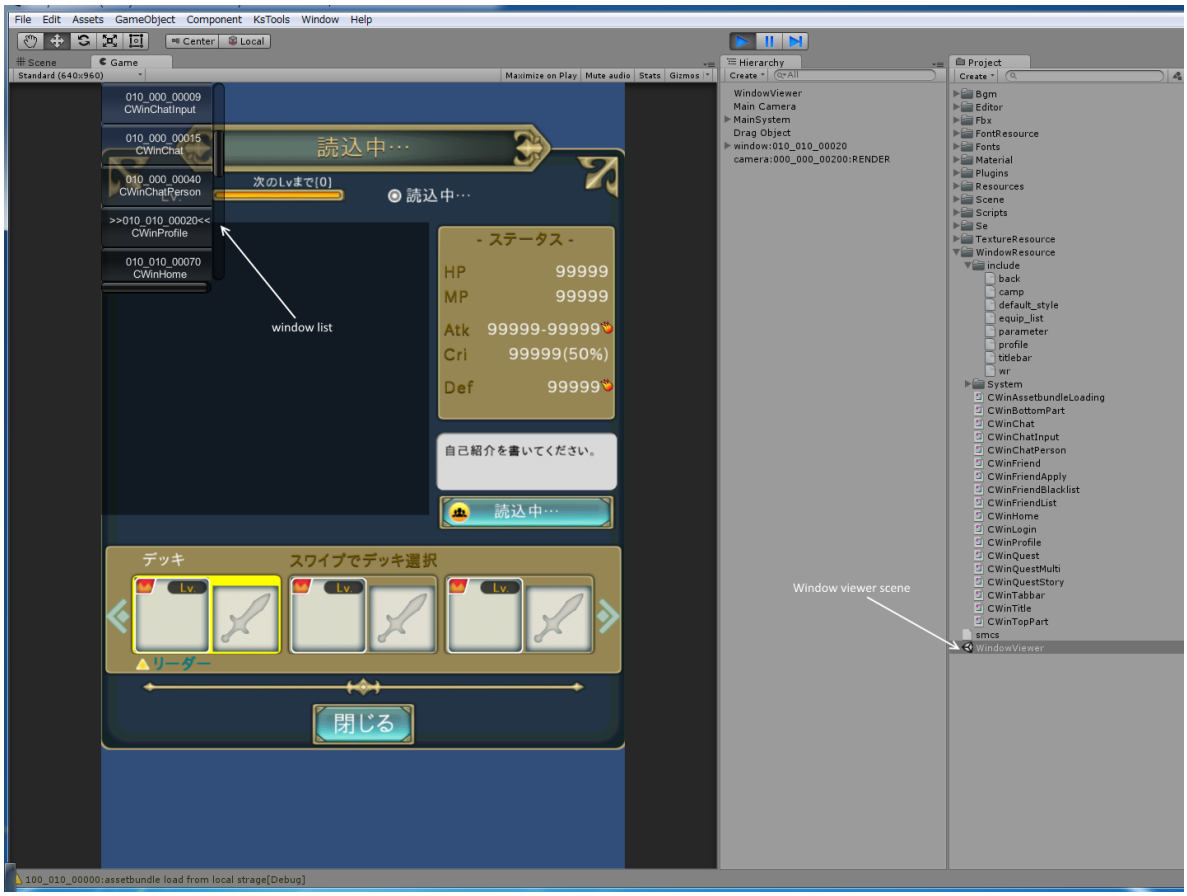
ただし、新規で追加したウィンドウを確認したい場合は、一度ウィンドウビューワを終わらせて再起動してください。

1. シーン WWindowViewer を選択し、実行してください。

   シーン実行中もスクリプトのコンパイルはできます。

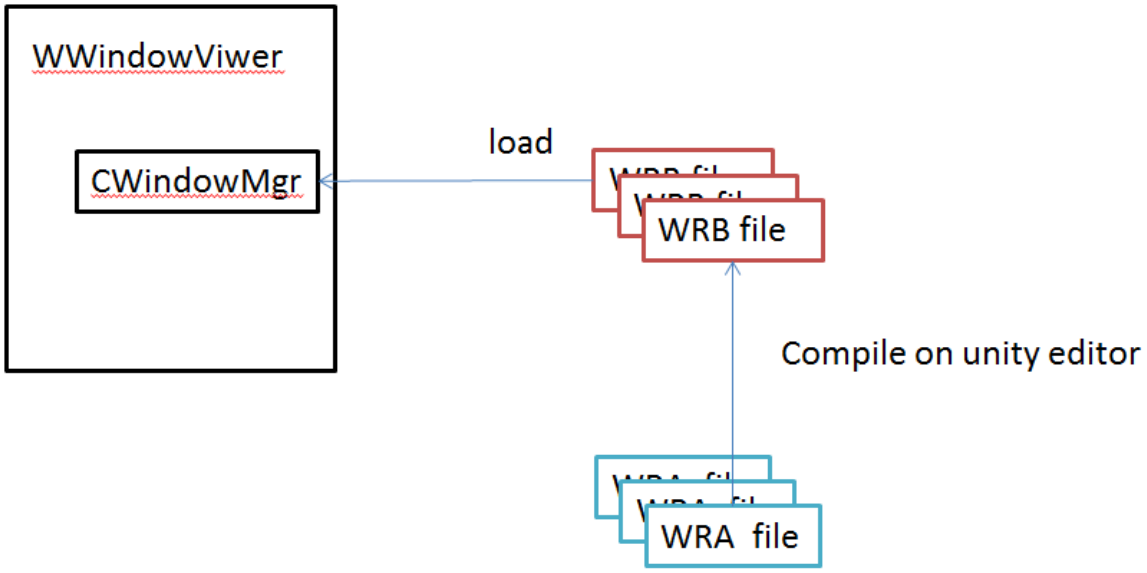
2. リストが表示されるので、所定のウィンドウを選んでください。

3. コンパイル結果のウィンドウが表示されます。



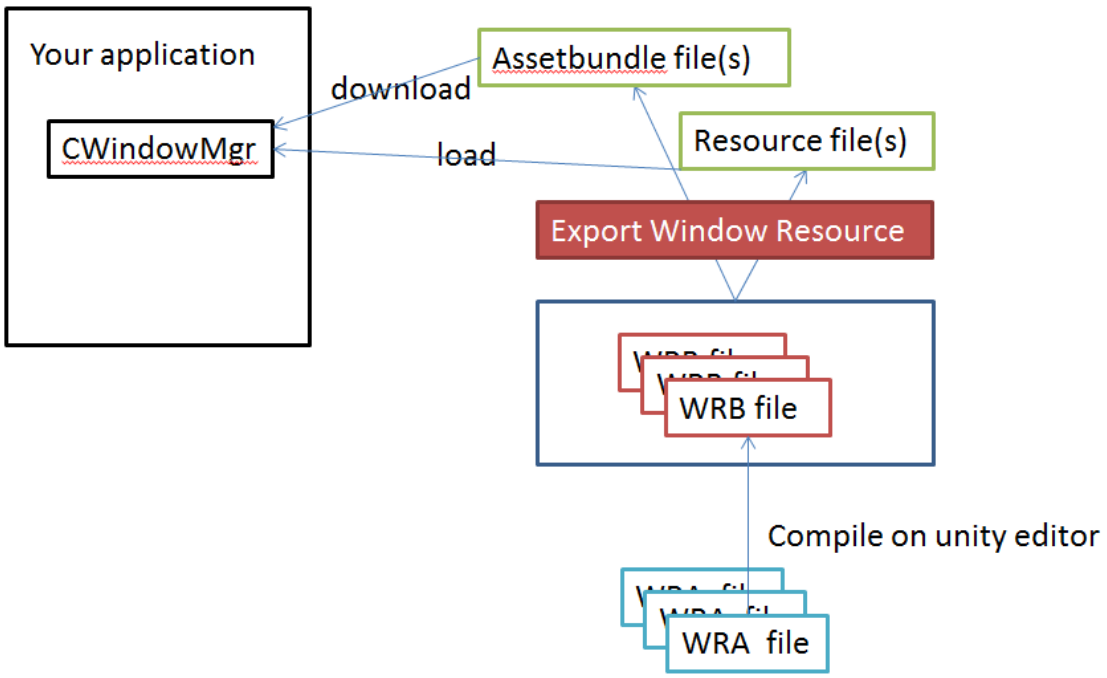
2.3.6 アセットバンドル/リソースデータ化

WWindowViewer は、各 wra ファイルから、コンパイルした wrb ファイルを直接読み込んで動作します。よって、wra ファイルをコンパイルした後、即座にその変更をチェックすることができます。



しかし、実際のアプリケーションでは、各 wrb ファイルをアプリケーションに入れたり、読み込んだりするのは管理上煩雑になります。そこで、本ウィンドウシステムでは、コンパイル済みのファイルを纏めて、一つ、若しくは複数のアセットバンドルやリソースファイルに出力することが可能になっています。アプリケーションでは、その纏められたファイルを予め読み込んでおけば、いつでも全てのウィンドウを好きなタイミングでクリエイトすることが可能になります。

この纏められたファイルは、HTTP 経由でダウンロードしたり、Resource 経由で読み込んだりすることが可能です。



Unity 上のメニューの [Tools]->[KsSoft]->[Export Window Resource] を選ぶと assetbundles フォルダにアセットファイルが出力されます。

ファイル名は、 デフォルトで **000_000_00010.unity3d** です。

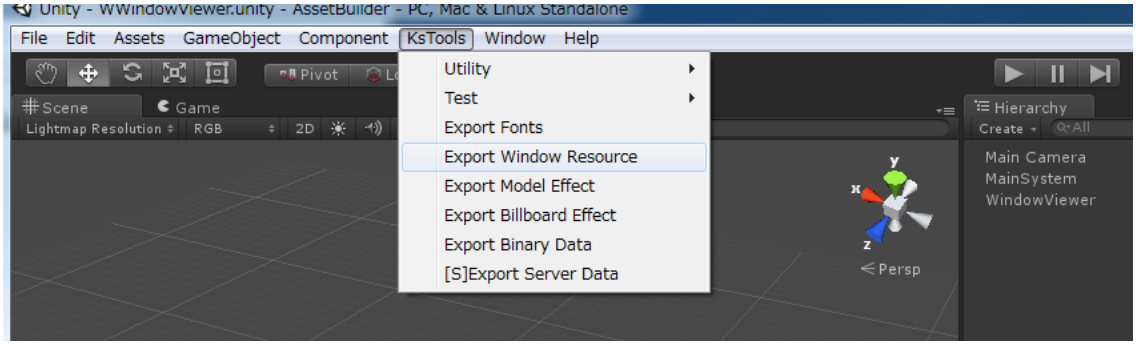
読み込むファイルを変更したいときは、こちらを 参考 にしてください。

WINDOW プロパティ: *RESOURCE = マルチ ID* を用いて、出力先を変更できます。

同じ マルチ *ID* を指定したウィンドウデータは、一つのアセットとして纏めた状態で出力します。また、アセットバンドルの情報を格納した **version.unity3d** も併せて更新されます。

CAssetBundleMgr は、 **version.unity3d** を起動時に一度読み込み、アセットバンドルが更新されているかどうかをチェックします。

WINDOW プロパティ: *RESOURCE = パス* を用いて、アセットデータとして出力することもできます。同じパスを指定したウィンドウデータは、一つのアセットバンドルとして纏めた状態で出力します。



2.3.7 実際のゲーム内でウィンドウを扱う方法

ウィンドウ作成環境とアプリケーションを分ける/分けないにかかわらず同じ方法で扱えます。

CMainSystem を作成/編集する

```
public class CMainSystem : CMainSystemBase {
    //=========================================================================
    /*!Awake
     * @brief    Unity Callback
    */
    new void Awake() {
        Application.targetFrameRate = 60;
        base.Awake();

        if (m_instance != null) {
            Debug.LogError("already exist CMainSystem");
            return;
        }
        m_instance = this;

        // Add Component
        gameObject.AddComponent<CInput>();
        gameObject.AddComponent<CSpriteFontMgr>();
        gameObject.AddComponent<CTextureResourceMgr>();
        gameObject.AddComponent<CWindowMgr>();
    }
    //=========================================================================
    /*!Initialize.
     * @brief    initialize
    */
    override protected void initialize() {
        base.initialize();
    }
    //=========================================================================
    /*!Instance.
```

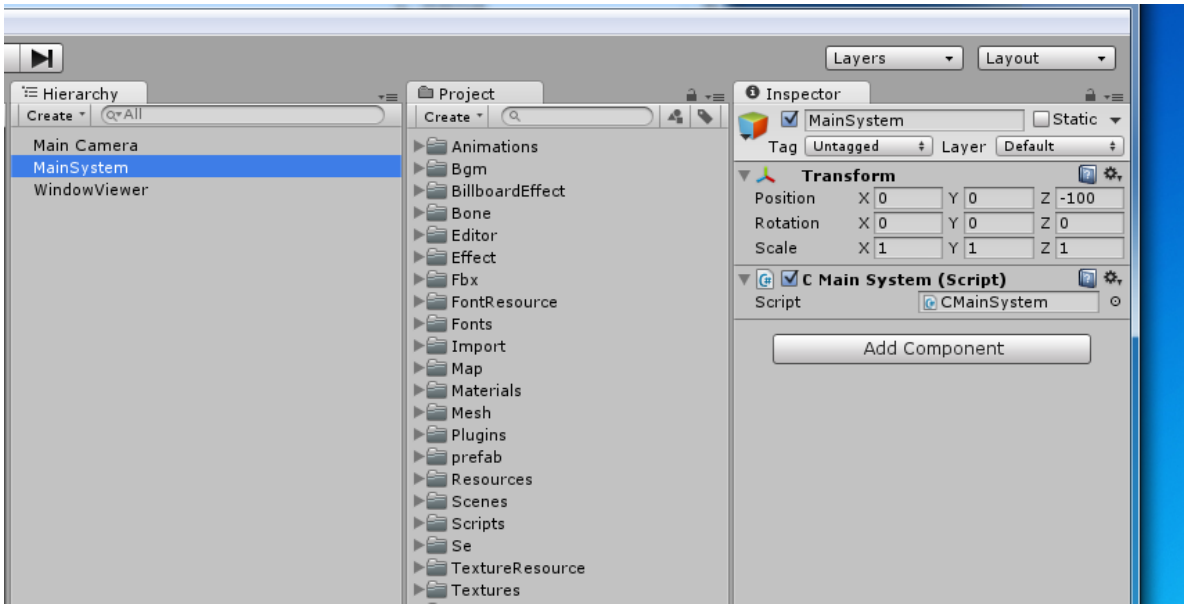
(次のページに続く)

(前のページからの続き)

```
        @brief    Instance.
    */
    static private CMainSystem    m_instance = null;
    new public static CMainSystem Instance {
        get {
            return m_instance;
        }
    }
}
```

CMainSystem をゲームオブジェクトに登録

下記の通り、MainSystem という空のゲームオブジェクトを作成し、そこに CMainSystem をスクリプトとして追加します。



CMainSystem は、*CMainSystemBase* を継承するようにしてください。CMainSystem を張り付けたゲームオブジェクトは、シーンを読み込んでも自動的に破壊されないように設定されています (DontDestroyOnLoad 参照)。

最低限ウィンドウシステムを使用するために必要なスクリプトは下記の４つです。

詳しくは、こちら を参照ください。

CInput

入力を監視するスクリプトです。必須です。

CSpriteFontMgr

フォントテクスチャ、フォントデータを管理しています。文字を出すために必須です。

CAssetBundleMgr をゲームオブジェクトにアタッチし、アセットデータが存在するときは、フォントデータをダウンロードして使うことができます。

詳しくは、こちら を参照ください。

CTextureResourceMgr

アトラス化したテクスチャリソースを管理しています。必須です。

CAssetBundleMgr をゲームオブジェクトにアタッチし、アセットデータが存在するときは、テクスチャデータをダウンロードして使うことができます。

詳しくは、こちら を参照ください。

CAssetBundleMgr

アセットバンドルを管理しています。

効果音、ウィンドウレイアウトデータ、テクスチャデータ、フォントデータ等全てアセットバンドル化し、読み込むことが可能です。

アセットバンドルを使用しないときは、必要ありません。

マネージャ自身はアセットバンドルの中身については関与しません。

詳しくは、こちら を参照ください。

CWindowMgr

ウィンドウを管理しています。必須です。

詳しくは、こちら を参照ください。

リソースから読み込むとき

Resources フォルダに配置した、アセットを読み込みます。

一度、読み込むと、ウィンドウデータは常駐します。

```
CWindowMgr cWindowMgr = CWindowMgr.Instance;
cWindowMgr.load("windows");
```

アセットバンドルから読み込むとき

[Export Window Resource] を行うと次の二つのファイルが更新されます (PC,Mac & Linux Standalone 環境の場合)。

- assetbundles/Windows/000_000_00010.unity3d
- assetbundles/Windows/version.unity3d

KsSoftConfig.httpserver が、正しい HTTP サーバのフォルダを指しているかを確認します。

注釈: 出力するファイル ID を変更したいときは、こちらを 参考 にしてください。

ウィンドウをクリエイトする方法

例えば、このような名前でウィンドウを定義したとします。

```
#include "default_style.h"

#define         WIN_WIDTH       640
$y = 160;

WINDOW(001_000_00000) {
   RESOURCE = "Assets/KsSoft/Resources/windows";
   PATH = NETWORKPATH;
   TEX_ID = 100_000_00000;
   CAPTION = 000_000_0000;
   STYLE = NOTITLEBAR|NOFRAME|ANCHOR_CENTER|TOP;
   SIZE = {100},{100};
   PRIORITY = PROGRESSBAR_PRIORITY;
};

METER(ProgressTotal) {
   ID = 000_001_00000;
   STYLE = ANCHOR_BOTTOM;
   POSITION = 0,$y;
   SIZE = -128{100};
   TEX_ID = 0,"MTRB";
   COLOR = COLOR32(255.0,255.0,255.0,255.0);
   SIZE1 = -128{100};
   TEX_ID1 = 0,"MTR";
   COLOR1 = COLOR32(255.0,255.0,255.0,255.0);
};
$y -=64;
METER(ProgressPart) {
```

(次のページに続く)

(前のページからの続き)

```
   ID = 000_001_00010;
   STYLE = ANCHOR_BOTTOM;
   SIZE = -128{100},-32;
   POSITION = 0,$y;
   TEX_ID = 0,"MTRB";
   COLOR = COLOR32(255.0,255.0,255.0,255.0);
   SIZE1 = -128{100},32;
   TEX_ID1 = 0,"MTR";
   COLOR1 = COLOR32(255.0,255.0,255.0,255.0);
};
TEXTURE(Wait) {
   ID = 000_002_00020;
   STYLE = ANCHOR_RIGHTBOTTOM;
   TEX_ID = 0,"LD00";
   POSITION = -48,-32;
};
TEXT(Message) {
   ID = 000_002_00030;
   STYLE = TEXT_RIGHT|ANCHOR_RIGHTTOP;
   CAPTION = 000_000_00250;
   FONT_KIND = "cfn20";
   POSITION = -48,32;
   COLOR = 1,1,1,1;
};
```

そうすると、CWinAssetbundleLoadingBase.cs という C#スクリプトが生成されています。これを継承した、CWinAssetbundleLoading クラスを作ります。

次のように CWinAssetbundleLoadingBase.create() を使ってウィンドウを作成できます。

```
CWinAssetbundleLoading.create();
```

自動生成された、CWinAssetbundleLoadingBase には、次のような関数が生成されています。

```
static public CWinAssetbundleLoading create(CWindowBase cParent = null) {
    return CWindowMgr.Instance.create<CWinAssetbundleLoading>(windowId,cParent);
}
```

クリエイト済のウィンドウを取得する方法

クリエイト済のウィンドウをウィンドウマネージャ経由で *find* を使って取得することができます。

```
CWinAssetbundleLoading cLoading = CWindowMgr.Instance.find<CWinAssetbundleLoading>
↪(CWinAssetbundleLoading.windowId);
if (cLoading != null) {
   //find created this window.
```

(次のページに続く)

(前のページからの続き)

```
} else {
   //can't find this window.
}
```

特定のウィンドウを全面に出す方法

ウィンドウマネージャが持つ、bringToTop を使って、特定のウィンドウを最前面に出すことができます。

```
CWindowMgr.Instance.bringToTop(cWindow);
```

効果音について

効果音を鳴らすには、CWindowMgr.soundeffect を書き換えます。

soundeffect は、IWinSoundEffect というインターフェースを持つオブジェクトです。

```
public interface IWinSoundEffect {
    void play(uint mSE);
}
```

付属している *CSeResourceMgr* と連動させる一つの例です。

CSeResourceMgr 経由で、アセットバンドル化された SE を取得します (一つのアセットバンドルに複数の SE が入っています)。

取得した、*CSeResource* は、IWinSoundEffect インターフェースを持っています。(音を実際、鳴らすには *CSoundEffectMgr* も CMainSystem に AddComponent しておく必要があります)。

下記の例では、052_000_00000 内にウィンドウ系の効果音がパックされているという前提の実装になっています。

注釈: 自作した IWinSoundEffect を持つオブジェクトと関連付けることも可能です。

```
public class CMainSystem : CMainSystemBase {
    //=============================================================================
    /*!Awake
    * @brief   Unity Callback
    */
    new void Awake() {
        base.Awake();

        if (m_instance != null) {
```

(次のページに続く)

(前のページからの続き)

```
            Debug.LogError("already exist CMainSystem");
            return;
        }
        m_instance = this;

        // Add Component
        gameObject.AddComponent<CInput>();
        if (KsSoftConfig.UseAssetBundle) {
             gameObject.AddComponent<CAssetBundleMgr>();
            }
        gameObject.AddComponent<CSpriteFontMgr>();
        gameObject.AddComponent<CTextureResourceMgr>();
        gameObject.AddComponent<CWindowMgr>();
        gameObject.AddComponent<CBgmResourceMgr>();
        gameObject.AddComponent<CSeResourceMgr>();
        gameObject.AddComponent<CSoundEffectMgr>();
    }
    //=========================================================================
    /*!Initialize.
    * @brief   initialize
    */
    override protected void initialize() {
        base.initialize();
        //-----------------------------------------
        // WindowMgr initilaize.
        //-----------------------------------------
        CWindowMgr     cWindowMgr = CWindowMgr.Instance;
        // Assign standard SEs.
        cWindowMgr.soundeffect = CSeResourceMgr.Instance.reference(new MulId(52,0,0),
↪true);    //2D sound;
        cWindowMgr.clickSE = new MulId(52,0,20);
        cWindowMgr.scrollSE = new MulId(52,0,110);
    }
    //=========================================================================
    /*!Instance.
    @brief     Instance.
    */
    static private CMainSystem m_instance = null;
        new public static CMainSystem Instance {
            get {
                 return m_instance;
            }
        }
    }
}
```

多言語対応、文字リソースをアセットバンドルにする方法

キャプション ID から、文字列に変換する際、CWindowMgr.captiondata に設定されたオブジェクトを通して変換されます。

captiondata は、IWinCaptionData インターフェースです。

現在のロケールに応じて返す文字列を変えることによって多言語対応が可能になっています。

```
public interface IWinCaptionData {
    string find(uint mCaptionId);
}
```

付属している *CMessageDataSheetMgr* と連動させるための一つの例です。

CMessageDataSheet が、IWinCaptionData インターフェースを持っているので次のようなコードでキャプションデータを取得できます。

```
public class CMainSystem : CMainSystemBase {
    //=============================================================================
    /*!Awake
    * @brief   Unity Callback
    */
    new void Awake() {
        base.Awake();
        if (m_instance != null) {
            Debug.LogError("already exist CMainSystem");
            return;
        }
        m_instance = this;

        // Add Component
        gameObject.AddComponent<CInput>();
        if (KsSoftConfig.UseAssetBundle) {
                gameObject.AddComponent<CAssetBundleMgr>();
            }
        gameObject.AddComponent<CSpriteFontMgr>();
        gameObject.AddComponent<CTextureResourceMgr>();
        gameObject.AddComponent<CWindowMgr>();
        gameObject.AddComponent<CBgmResourceMgr>();
        gameObject.AddComponent<CSeResourceMgr>();
        gameObject.AddComponent<CSoundEffectMgr>();

        addManager(new CMessageDataSheetMgr(Utility.getSystemLocale()));
    }
    //=============================================================================
    /*!Initialize
    * @brief   initialize
```

(次のページに続く)

(前のページからの続き)

```
    */
    override protected void initialize() {
        base.initialize();
        //-----------------------------------------
        // WindowMgr initialize.
        //-----------------------------------------
        CWindowMgr     cWindowMgr = CWindowMgr.Instance;
        cWindowMgr.captiondata = CMessageDataSheetMgr.Instance.find(new FiveCC("WNDW
↪"));
        // Assign standard SEs
        cWindowMgr.soundeffect =  CSeResourceMgr.Instance.reference(new MulId(52,0,0));
↪;
        cWindowMgr.clickSE = new MulId(52,0,20);
        cWindowMgr.scrollSE = new MulId(52,0,110);
    }
    //=========================================================================
    /*!Instance.
    @brief     Instance.
    */
    static private CMainSystem m_instance = null;
        new public static CMainSystem Instance {
            get {
                 return m_instance;
            }
        }
    }
}
```

注釈: 自作した IWinCaptionData を持つオブジェクトと関連付けることも可能です。

第 3 章

マルチ ID,FiveCC:

3.1 マルチ ID,FiveCC

ウィンドウスクリプト内で使われる ID の形式について、説明します。

3.1.1 マルチ ID

32bit 整数を、8bit,8bit,16bit に分割して扱う ID をマルチ ID と定義します。

次のようなフォーマットで記述します。

8bit_8bit_16bit

次の範囲で値が取れます。

000_000_00000 ～ 255_255_65535

ただし、000_000_00000 は使わないようにしてください。

```
000_010_00000
123_123_12345
123_456_78910  //invalid(It is beyond the range of value)
```

C#からのアクセス方法

クラス **MulId** を用意しています。

```
//Constructor
MulId(string sId);
MulId(uint uppper,uint middle,uint lower);
MulId(uint val);
```

(次のページに続く)

(前のページからの続き)

```
// Properties
uint Upper;     //Get the upper 8bit
uint Middle;    //Get the middle 8bit
uint Lower;     //Get the lower 16bit

//cast to uint
static implicit operator uint (MulId mulId);
```

3.1.2 FiveCC

5 文字以内のアスキー文字から構成される ID です。使える文字は、以下の通りです。

- 0〜9
- a〜z
- A〜Z
- ?,_
- 半角スペース

ただし、'?' と'_' は、同じ数値でエンコーディングされるので気を付けてください。

```
"BTN00"
"BTN0?" = "BTN0_"
"Test"
```

C#からのアクセス方法

クラス **FiveCC** を用意しています。

```
//Constructor
FiveCC(string sId);
FiveCC(uint uppper,uint middle,uint lower);
FiveCC(uint val);

// Get the character pointed to by the index
public char this[int index];

//cast to uint
static implicit operator uint (FiveCC FiveCC);
```

第 4 章

テクスチャ:

4.1 テクスチャについて

C#上でテクスチャリソースを扱うには、 こちら を参照ください。

ウィンドウシステムで使うテクスチャは、幾つかのテクスチャをアトラス化することによって一つのメッシュとしてまとめてレンダリングできるようにしています。

また、アトラス化するときに、自動的に 16bit に減色し、ディザリングも行ってくれます。

パーツ毎にディザリングを行うか/単純な色変換を行うか選択可能になっています。

■テクスチャパーツ

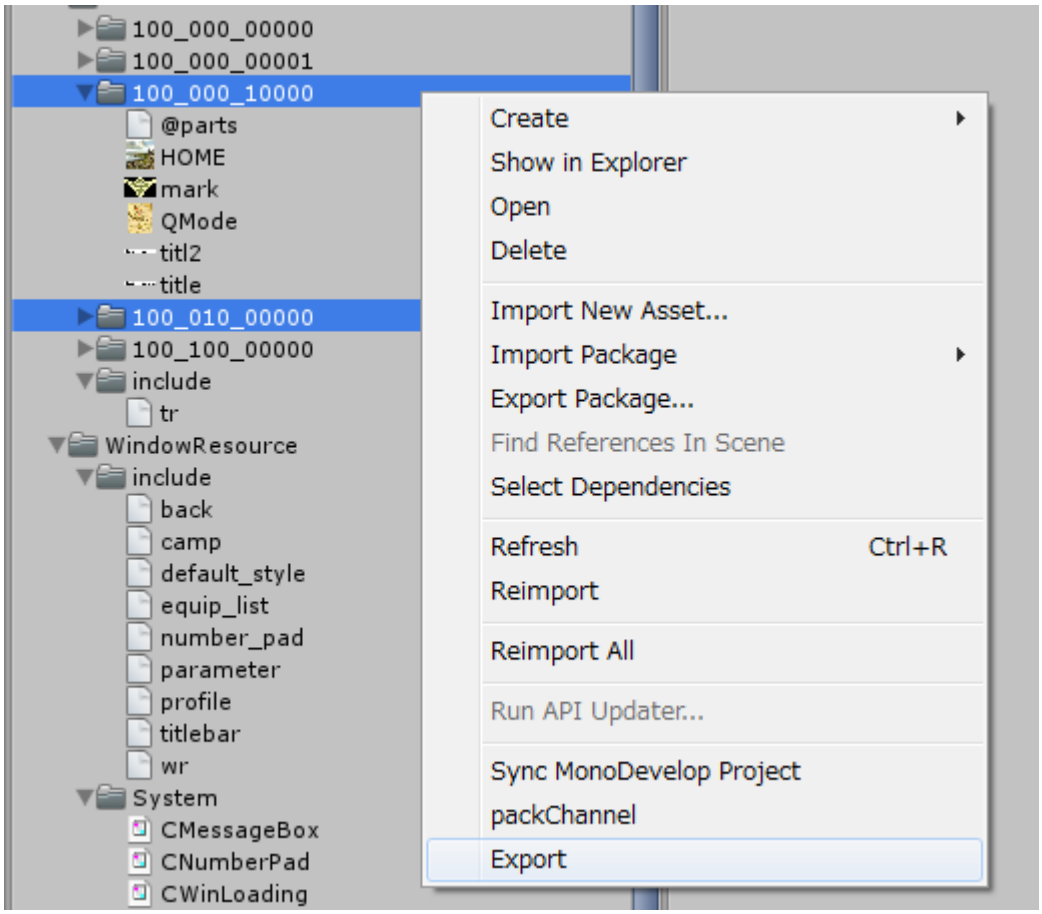
単純にパーツとも呼びます。アトラス化する前のテクスチャをテクスチャパーツと呼びます。

■テクスチャ

本ドキュメントでテクスチャと呼ぶときは、アトラス化された後のテクスチャを指します。また、通常のテクスチャと明確に分けるために、テクスチャリソースと呼ぶこともあります。

4.1.1 テクスチャリソースを作る方法

1. Unity プロジェクトを起動します。
2. プロジェクト内の TextureResource フォルダ (Assets/KsSoft/TextureResource) の直下に マルチ *ID* で フォルダを作ります。
3. そのフォルダ内に、テクスチャを配置していきます。
4. フォルダを選んで、[右クリック]→[Export] を選んでください (複数選択可能)。



これによって、100_000_00000.unity3d というアセットバンドルが、assetbundles の下に生成されます。

■テクスチャ名の制約

アトラス化するテクスチャの名前は、拡張子を除いて 5 文字である必要があります。

これは、 *FiveCC* としてエンコーディングされるためです。

また、 マルチ *ID* ファイル名も使用できます。

```
BTN00.pgn
000_000_00010.png
```

■フォルダ名の制約

マルチ *ID* である必要があります。

アセットバンドルは、 マルチ *ID* で管理されるためです。

4.1.2 定義ファイル @parts.def

@parts.def をテクスチャフォルダ内に置いておくと、定義ファイルに沿ってアトラス化してくれます。

単純なテキストファイルなので、お好みのエディタを使って編集してください。

ファイルのエンコーディングは、UTF8 です。

リソース化するかどうかを選択

```
RESOURCE = ON/OFF;
```

次のように、RESOURCE を有効にすると、KsSoft/Resources/の下に次のようなファイルを生成します。

- 001_000_00000.spr
- 001_000_00000.tex

アセットバンドル経由で読み込まず、この二つのファイルをリソース経由で読み込みます。デフォルトでは、アセットバンドルを出力するようになっています。

```
RESOURCE = ON;

PART("partA") {
    COLOR = 0.5,0.5,0,5,1;
};
PART("partB") {
   COLOR = 0.5,0.5,0,5,1;
};
```

テクスチャフォーマット.

```
FORMAT = Texture Format;
```

テクスチャフォーマットのデフォルトは RGBA4444 です。

それ以外に、*PNG,JPG* を指定することが可能です。

16bit テクスチャを指定するとき以外は、ディザリングを OFF にしてください。

```
DITHER = OFF;
FORMAT = RGBA32;
PART("partA") {
    COLOR = 0.5,0.5,0,5,1;
};
PART("partB") {
   DITHER = OFF;
   COLOR = 0.5,0.5,0,5,1;
};
```

PNG,JPG ファイルとしてアトラス化する方法.

幾つかのテクスチャを纏め (アトラス化) た後、テクスチャをアセットバンドルに変換するときに、テクスチャを PNG や JPG ファイルに変換することが可能です。

ただし、実際にアプリケーションに組み込んで使うとき、これらテクスチャは 32bit テクスチャとして展開されます。

メリットとデメリットを考えて使ってください。

フォーマット	実機上でのテクスチャフォーマット
FORMAT = PNG;	ARGB32
FORMAT = JPG;	RGB24

利点

アセットサイズが小さくなる (=ダウンロードサイズが小さくなる)

弱点

32bit テクスチャとして展開されるため、VRAM やメインメモリを圧迫します。

また、レンダリング速度も犠牲になります。

注釈: 特にメモリ帯域の狭いスマートフォン上では、著しくパフォーマンスが落ちるため、気を付けてください。

ディザリングのデフォルト値

```
DITHER = ON/OFF;
```

ディザリングをするかしないかは、パーツ毎に設定できます。

特に指定しないとき、どちらをデフォルト値にするか選択可能です。

この例では partA は、ディザリングを行い、partB は、ディザリングがキャンセルされます。

```
DITHER = ON;

PART("partA") {
    COLOR = 0.5,0.5,0,5,1;
};
PART("partB") {
   DITHER = OFF;
```

(次のページに続く)

(前のページからの続き)

```
   COLOR = 0.5,0.5,0,5,1;
};
```

シェーダの切り替え

```
SHADER = "SHADER PATH";
```

シェーダは、テクスチャリソース一つに対して、一つ指定できます。

シェーダはパーツ単位で切り替えることはできません。

ただし、テクスチャリソース単位では切り替えることが可能です。

```
SHADER = "Custom/Billboard";

PART("blck") {
    COLOR = 1,1,1,1;
};
```

パーツ定義とエイリアス

パーツの定義は、次のような形で設定可能です。

```
PART(Texture part file name) {
   Property 0;
   Property 1;
       :
   Property n;
};

PART(Alias name,Texture part file name) {
   Property 0;
   Property 1;
       :
   Property n;
};
```

後者は同一のテクスチャパーツに対して名前をエイリアスすることによって、あたかも同一パーツで別のプロパティを持つパーツとして振る舞うことが可能です。

例えば、次のように定義すると、"BTN00"は、テクスチャカラーがそのまま出力されますが、"BTN01"は暗くレンダリングされます。

```
PART("BTN00") {
  COLOR = 1,1,1,1;
};
PART("BTN01","BTN00") {
  COLOR = 0.5,0.5,0.5,1;
};
```

パーツ定義プロパティ

COLOR = R,G,B,A;

カラーを変更します。

各色の要素は、0〜1 で指定します。

デフォルトは、1,1,1,1 になっています。

DITHER = ON/OFF;

パーツをディザリングするかしないかを選択できます。

ディザリングを ON にすると誤差拡散によってディザリングします。

デフォルトは、ON になっています。

NODIVIDE;

NOPATCH;

パーツを拡大表示するとき通常の拡大を行う。

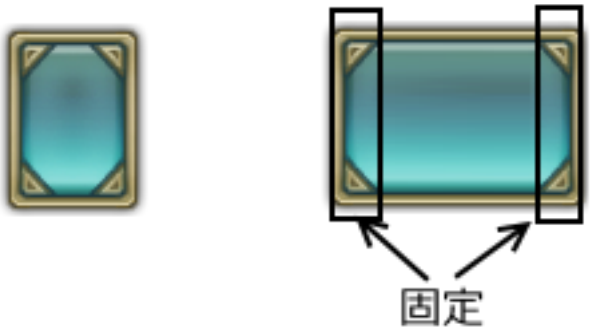
パーツの分割はデフォルトは、NODIVIDE になっています。

DIVIDE3H = 左固定幅, 右固定幅;

PATCH3H = 左固定幅, 右固定幅;

パーツを拡大するとき、左右の指定幅だけ固定し、真ん中を拡大表示する。

```
PART("BTN01") {
   DIVIDE3H = 24,24;
   DITHER = ON;
};
```

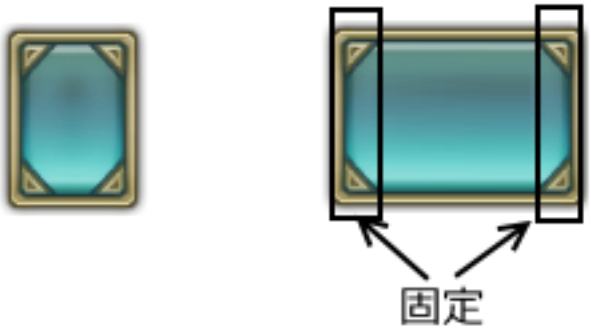


DIVIDE3V = 上固定高, 下固定高;

PATCH3V = 上固定高, 下固定高;

パーツを拡大するとき、上下の指定高だけ固定し、真ん中を拡大表示する。

```
PART("BTN01") {
   DIVIDE3V = 24,24;
   DITHER = ON;
};
```

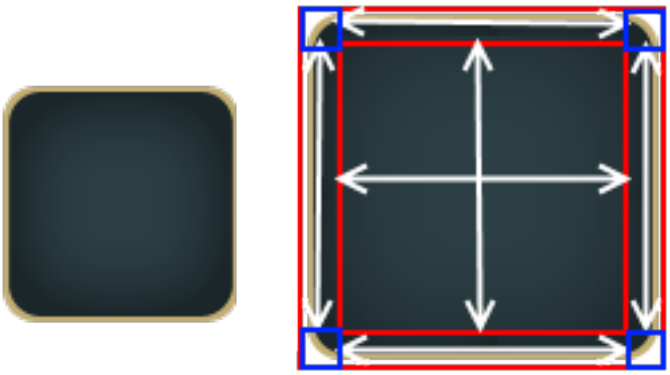


DIVIDE9 = 左上固定幅, 左上固定高, 右下固定幅, 右下固定高;

PATCH9 = 左上固定幅, 左上固定高, 右下固定幅, 右下固定高;

パーツを拡大するとき、4 辺の指定サイズだけ固定し、真ん中を拡大表示する。

```
PART("FRAME") {
   DIVIDE9 = 40,40,40,40;
   DITHER = OFF;
};
```



インクルードパス

以下のパスにインクルードパスが、通っています。インクルードファイルを作成する際は、ファイルをここに置いてください。

```
Assets/KsSoft/TextureResource/include
```

@parts.def をコンパイルするときにインクルードされるファイル

次のファイルは、@parts.def をコンパイルするときに、自動的にインクルードされます。共通の設定は、このファイルに記述すると便利です。

```
Assets/KsSoft/TextureResource/include/tr.h
```

4.1.3 プリプロセッサ

C 言語,C++ 言語とほぼ同じプリプロセッサが使えます。

プリプロセッサ	説明
// コメント	ラインコメント
/* コメント */	ブロックコメント
#include ファイル名	ファイルをインクルードする
#define 記号定数の定義	記号定数の定義
#define 関数マクロ	関数マクロ
#if defined(記号定義) 〜 #endif	条件コンパイル
#ifdef 〜 #endif	条件コンパイル
#ifndef 〜 #endif	条件コンパイル
#pragma once	多重インクルード防止

BuildTarget がマクロとして自動定義されています。

```
#if defined(StandaloneWindows)
   MS-Windows
#else
   Other
#endif
```

第 5 章

スプライトフォント:

5.1 フォントについて

最大ビットマップフォント４枚を一つのフォントとして扱えます。

ただし、これら４枚のビットマップをアセット化する際に、ビットマップファイルの α 成分のみを参照します。４枚の α 成分を一枚のテクスチャの (R,G,B,A) 成分にそれぞれマッピングし保存します。

４枚を超えるテクスチャを一つのフォントデータとして扱うことができません。

5.1.1 リソースデータとして作成

Assets/KsSoft/フォルダの下に、FontResource というフォルダを作ります。

そのフォルダの下に *FiveCC* で命名したテキストファイルとビットマップファイルを置きます。

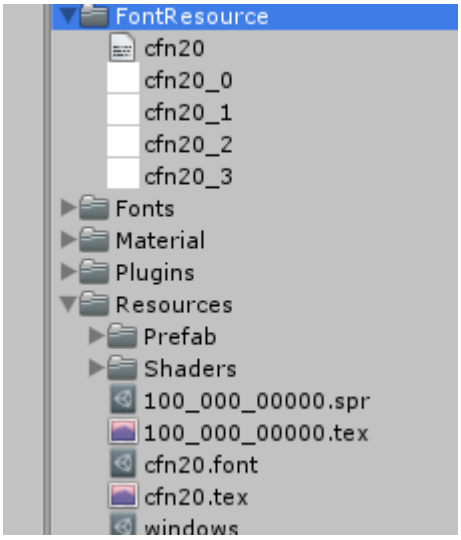
例えば、cfn20 というフォントを作りたいときは、次のようなファイルを用意し、指定の場所に置きます。

- cfn20.txt
- cfn20_0.png
- cfn20_1.png
- cfn20_2.png
- cfn20_3.png

この状態で cfn20.txt を選択し、[右クリック]→[Export] を選ぶと、Resources フォルダの下に次のようなファイルが生成されます。

- cfn20.font
- cfn20.tex

以上の手順で、cfn20 というフォントがアプリケーション内で使用可能になります。



5.1.2 アセットバンドルデータとして作成

Assets/KsSoft/のフォルダの下に、Fonts というフォルダを作ります。

そのフォルダの下に *FiveCC* で命名したテキストファイルとテクスチャファイルを置きます。

例えば、fn16 というフォントを作りたいときは、次のようなファイルを用意し、指定の場所に置きます。

- fn16.txt
- fn16_0.png
- fn16_1.png
- fn16_2.png
- fn16_3.png

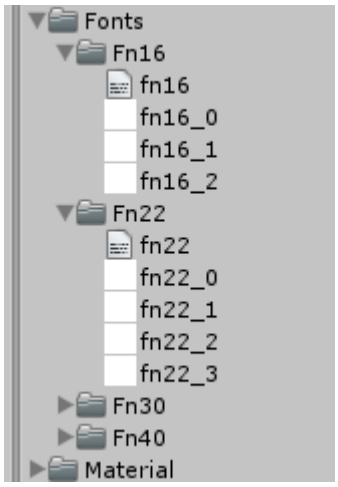
また、複数のフォントを使いたいときは、同じ手順で複数ファイルを配置しておきます。

準備ができましたら、[Tools]->[KsSoft]->[Export Fonts] を実行してください。

以下のアセットバンドルが生成されます。

    assetbundles/Windows/000_001_00000.unity3d

このアセットバンドルの中に必要なフォントデータがすべて格納されています。



注釈: デフォルト値を変更したいときは、 こちら を参考ください。

5.1.3 BMFont によるフォントデータ作成方法

BMFont(Bitmap Font Generator) を使ってビットマップフォントを作る方法を紹介します。

ビットマップフォントの作成

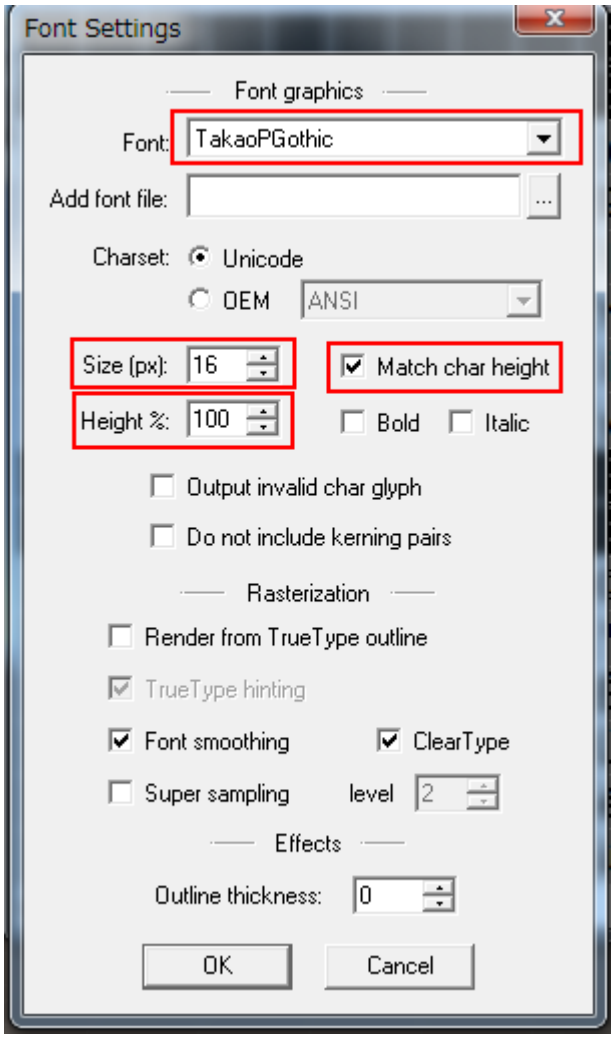
■ Font Settings

まずは使用するフォントを選択します。メニューから [Options]→[Font Settings] を選択します。

次にフォントの種類とフォントサイズを選びます。

[Match char height] にはチェックを入れておいてください。

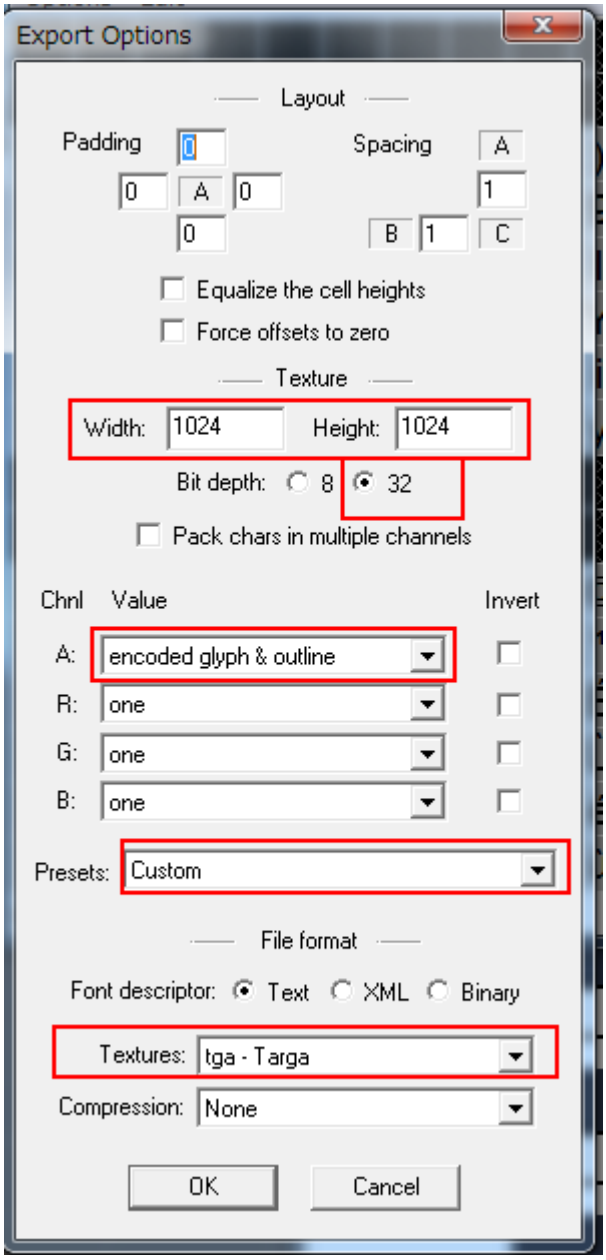
[Hieght%] には 100 を入れておきます。



■ Export Options

次にメニューから [Options]→[Export Options] を選択して Export の設定を行います。

まずは以下の赤枠の部分の値を変更します。



■含める文字を選択

次にビットマップフォントに含める文字を指定します。

■確認

設定された項目にそって正しく出力されるか確認します。メニューから [Options]→[Visualize] を選び、プレビューします。

テクスチャが 4 枚以内に収まっているかを確認します。

もし 1 枚に全て収まっているならビットマップサイズが 4 枚になるようにテクスチャサイズを調整してみるとデータサイズを軽くできる可能性があります。

■出力

これで設定は完了です。

メニューから [Options]→[Save bitmap font as...] を選択してビットマップを出力します。

ファイルは、 *FiveCC* を使って名前を付けてください。

最後に、出力された fnt ファイルを txt ファイルにリネームするとビットマップフォントに必要なファイルがすべて揃います。

第 6 章

効果音,BGM:

6.1 効果音、BGM について

音関連を制御するためのマネージャとして、次のものを用意しています。

- *CSoundEffectMgr*
- *CSeResourceMgr*
- *CBgmResourceMgr*

これらを用いて、アセットバンドルの管理、SE のグループ化、プライオリティ、同時発声数制限等を行います。また、BGM は、イントロ＋ループ部分の再生を行ってくれます。

注釈: グループ化、プライオリティ、同時発声数制限は、3D のみ対応しています。2D の音声は、同じ音が同時にならないような制御のみ行っています。

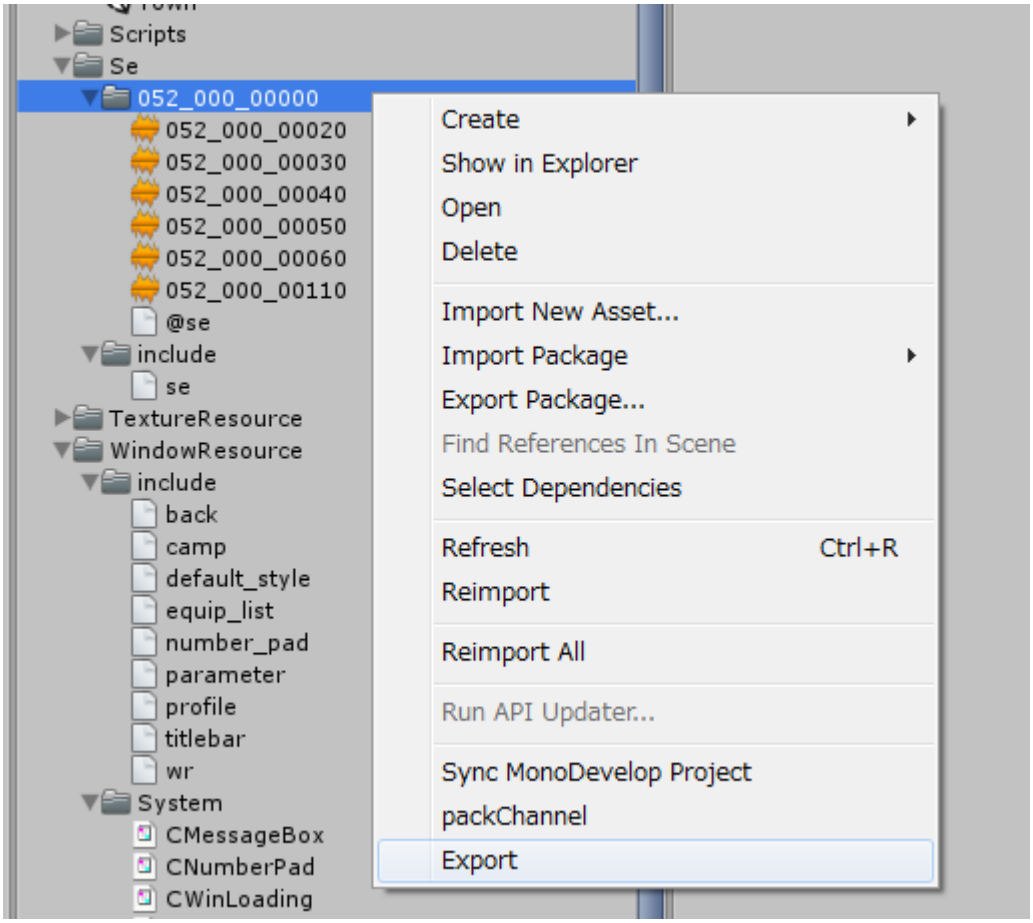
6.1.1 SE のアセットバンドルを作成する方法

次の手順で、効果音に使う音声ファイルを配置してください。

1. Assets/KsSoft/の下に、Se というフォルダを作ります。
2. 作成した Se フォルダの下に、 マルチ *ID* で更にフォルダを作ってください。
3. マルチ *ID* 名で作成したフォルダの下に音声ファイルを置いてください。
4. 必要ならば、@se.def というファイルも作成し置いてください。
5. フォルダを選んで、[右クリック]→[Export] を選んでください (複数選択可能)。

2. で作成したフォルダ名と同名のアセットバンドルが生成されます。場所は、assetbundles/XXXX/です。

配置する音声ファイルは、 マルチ *ID* か、 *FiveCC* である必要があります。



6.1.2 @se.def 記述方法

アセットバンドルを作成するときに、各 SE に様々な情報を埋め込むことができます。以下に例を挙げておきます。ファイルは、UTF-8(BOM 無し) で作成してください。

```
// click sound effect
SE(052_000_00020) {
    VOLUME = 1;
    PRIORITY = 1;
    GROUP = 200;
    POLYPHONY = 1;
};
// decide sound effect
SE(052_000_00030) {
    VOLUME = 50%;
    PRIORITY = 1;
    GROUP = 200;
    POLYPHONY = 1;
```

(次のページに続く)

(前のページからの続き)

```
};
```

基本的な記述方法は、次の通りです。エリアスも使え、同一の音声ファイルに別名を与え、違うパラメータを割り振ることが可能です。

```
SE(se file name) {
   Property 0;
   Property 1;
       :
   Property n;
};

SE(Alias name,SE file name) {
   Property 0;
   Property 1;
       :
   Property n;
};
```

エリアス名/ファイル名は、 マルチ *ID* か *FiveCC* (半角英数 5 文字) である必要があります。

次のようにルールがあります。

ID	ファイル名
000_000_00010	000_000_00010.wav 等
"click"	click.wav 等

注釈: *FiveCC* (半角英数 5 文字) を使うときは、"〜"でくくるのを忘れないでください

インクルードパス

以下のパスにインクルードパスが、通っています。インクルードファイルを作成する際は、ファイルをここに置いてください。

```
Assets/KsSoft/Se/include
```

@se.def をコンパイルするときにインクルードされるファイル

次のファイルは、@se.def をコンパイルするときに、自動的にインクルードされます。共通の設定は、このファイルに記述すると便利です。

```
Assets/KsSoft/Se/include/se.h
```

@se.def に使えるプロパティ

VOLUME = 数字

ボリュームを設定します。

値は、0〜1 の少数で指定するか、0〜100% の百分率が使えます。

- VOLUME = 1;

GROUP = グループ番号

3D として音を鳴らした時のみ有効です。

グループは、0〜255 の整数で指定してください。同グループ内で同時発声制限や、プライオリティなどを指定できます。

PRIORITY = プライオリティ

3D として音を鳴らした時のみ有効です。

発声する際のグループ内でのプライオリティを 0〜255 の整数で設定します。値が大きいほど優先度が高くなります。

POLYPHONY = 同時発声数

3D として音を鳴らした時のみ有効です。

1〜255 の間で設定して下さい。同時発声数が許された数だけ、同グループ内の音声を鳴らそうとします。同時発声数を超えたときは、鳴っている音声と優先順位を比べ高ければ、鳴っている音声を停止させ再生を開始します。

DISTANCE = 最少距離, 最大距離

3D として音を鳴らした時のみ有効です。

最少距離よりも近いところで発生したとき、最大のボリュームで再生します。

逆に最大距離よりも離れると音が聞こえなくなります。

プリプロセッサ

C 言語,C++ 言語とほぼ同じプリプロセッサが使えます。

プリプロセッサ	説明
// コメント	ラインコメント
/* コメント */	ブロックコメント
#include ファイル名	ファイルをインクルードする
#define 記号定数の定義	記号定数の定義
#define 関数マクロ	関数マクロ
#if defined(記号定義) 〜 #endif	条件コンパイル
#ifdef 〜 #endif	条件コンパイル
#ifndef 〜 #endif	条件コンパイル
#pragma once	多重インクルード防止

BuildTarget がマクロとして自動定義されています。

```
#if defined(StandaloneWindows)
   MS-Windows
#else
   Other
#endif
```

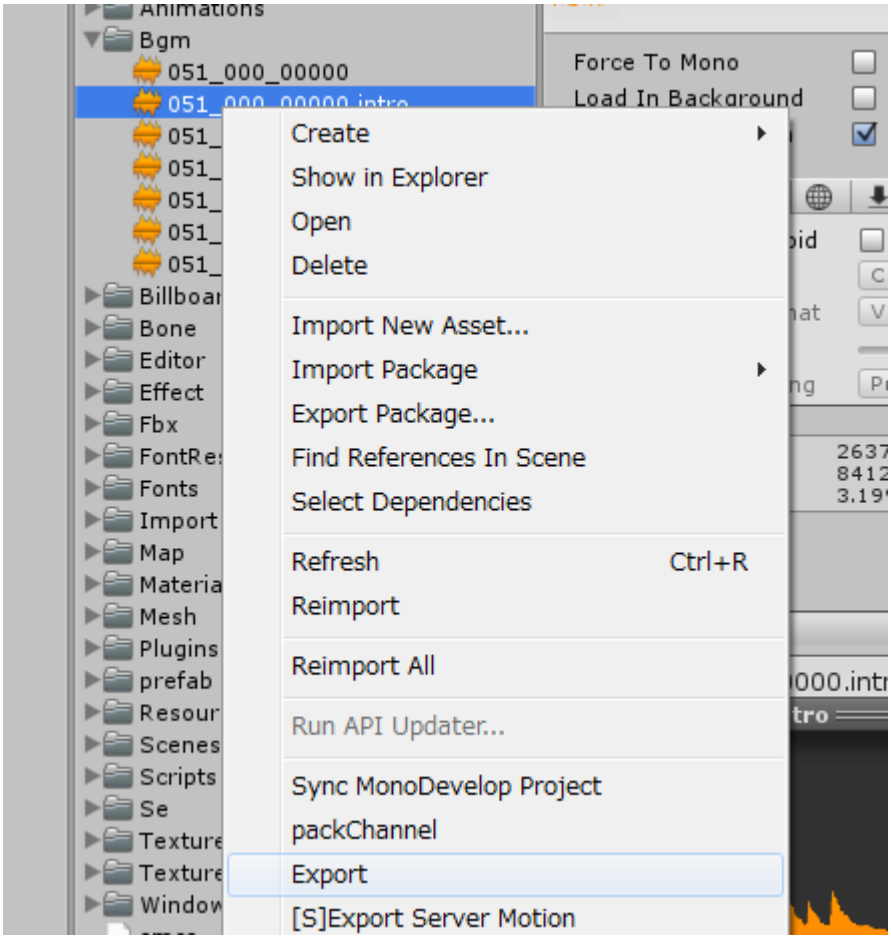
6.1.3 BGM のアセットバンドルを作成する方法

BGM を、予めイントロ部分、ループ部分に分けておいてアセットバンドル化することによって、マネージャ側が自動的に順番に再生を行ってくれます。

次の手順でアセットバンドル化してください。

1. Assets/KsSoft/の下に、Bgm というフォルダを作ります。
2. ループ部分は、 マルチ *ID* .mp3 という名前で Bgm フォルダ内においてください。
3. イントロ部分は、 マルチ *ID* .intro.mp3 という名前で同フォルダ内においてください。
4. 準備ができたら、アセットバンドル化したいループファイルか、イントロファイルのどちらかを選択します。
5. [右クリック]→[Export] を選択してください。

   マルチ *ID* 部分が、アセットバンドル名になります。


Animations
Bgm
051_000_00000
Force To Mono
Load In Background
Create
Show in Explorer
Open
Delete
Import New Asset...
Import Package
Export Package...
Find References In Scene
Select Dependencies
Refresh Ctrl+R
Reimport
Reimport All
Run API Updater...
Sync MonoDevelop Project
packChannel
Export
[S]Export Server Motion
Billboar
Bone
Editor
Effect
Fbx
Fonts
Import
Map
Mesh
Plugins
prefab
Scenes
Scripts
Texture
Window

第 7 章

文字リソース, 多言語対応:

7.1 多言語対応について

文字リソースをアセットデータとしてアプリケーションと分離する機能を持っています。

この文字リソースを切り替えることによって多言語対応が可能になっています。

ただし、ウィンドウシステムを使うに当たって必須ではありません。

文字データのアセットデータを作成する際、MS-Excel を使用します。

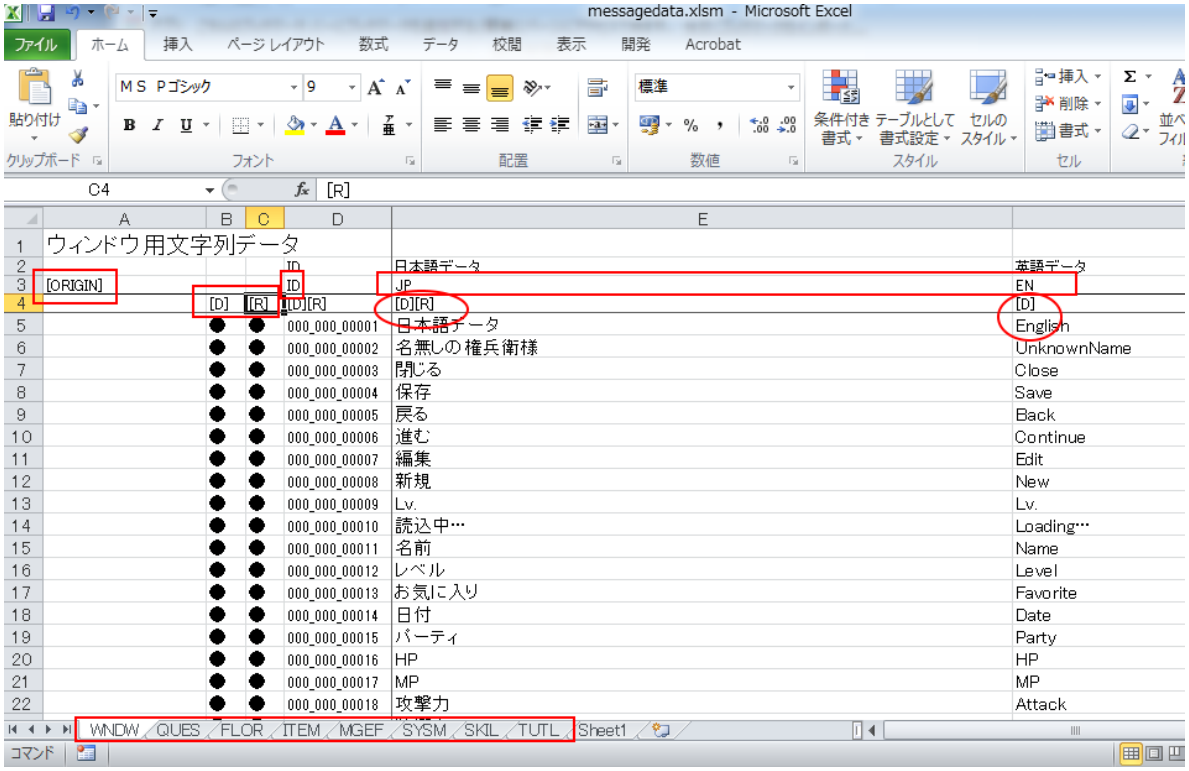
作成した Excel データからリソースデータ用のバイナリデータを出力するためのツールを用意しています。

出力されたデータを使用するためのオブジェクトとして、以下を用意してあります。

- `CMessageDataSheetMgr`
- `CMessageDataSheet`

7.1.1 文字リソース Excel 書式

Excel ファイルフォーマットは次の図のようになります。



[ORIGIN]

[ORIGIN] というカラムを基準にデータを取得してきます。

必須ですので必ず右下のデータが始まる先頭においてください。

[ORIGIN] と同じ行はコメント扱いになり、データ出力に影響を与えません。

[D],[R]

B4 カラム,C4 カラムの [D],[R] は、バージョンごとに出力するかしないかを選択します。ここの列が空欄の時は、その行はデータ出力を抑止します。

- [D] はデバッグバージョン
- [R] はリリースバージョン

D や R の文字は自由に設定でき、コンバータに渡す引数でどのデータを出力するかしないかを選択可能になっています。細かくバージョン管理したいときは、行を追加して下さい。

ID

D3 カラムの ID は、文字リソースの ID を マルチ *ID* で指定することになります。その下 (D4) の、[D][R] は、デバッグバージョン、リリースバージョンでも必要なのでこのような記述になっています。

ID を使って、文字リソースにアクセスするので、ID カラムは全てのバージョンで必須です。

JP,EN

E3 ,F3 カラムの JP,EN は文字リソースのロケール名です。出力するとき次のようにデータが出力されます。

- messagedata.JP.bin
- messagedata.EN.bin

自由に列を追加可能です。

E4 の [D][R] は、ロケールが JP の時は、リリース、デバッグ両方で出力することを意味します。

それに対して、F4 の [D] は、デバッグバージョンでのみ出力することを意味しています。

シート名 [WNDW]

シート名は、 *FiveCC* でエンコードされます。

よって、半角英数字 5 文字まででシート名を付けてください。

シート名 +ID の組み合わせで、文字リソースにアクセスすることになります。

7.1.2 文字リソース Excel コンバート方法

messagedata.exe、若しくは messagedata.py を使ってください。

インストール手順は、 こちら を参照ください。

python(2.6 以降) を用いて記述されています。

Excel データの読み込みには、openpyxl を使用しています。

exe にするときに、pyinstaller を使っています。

[使用方法] messagedata.exe or messagedata.py

messagedata [-v version] Excel ファイル名 [出力ファイル名]

- -v version

version で指定した、文字列のデータを出力します。デフォルトでは、D になっています。

- Excel ファイル名

  エクセルファイルを指定してください。必須です。

- 出力ファイル名

  省略したときは、エクセルファイル名から生成します。

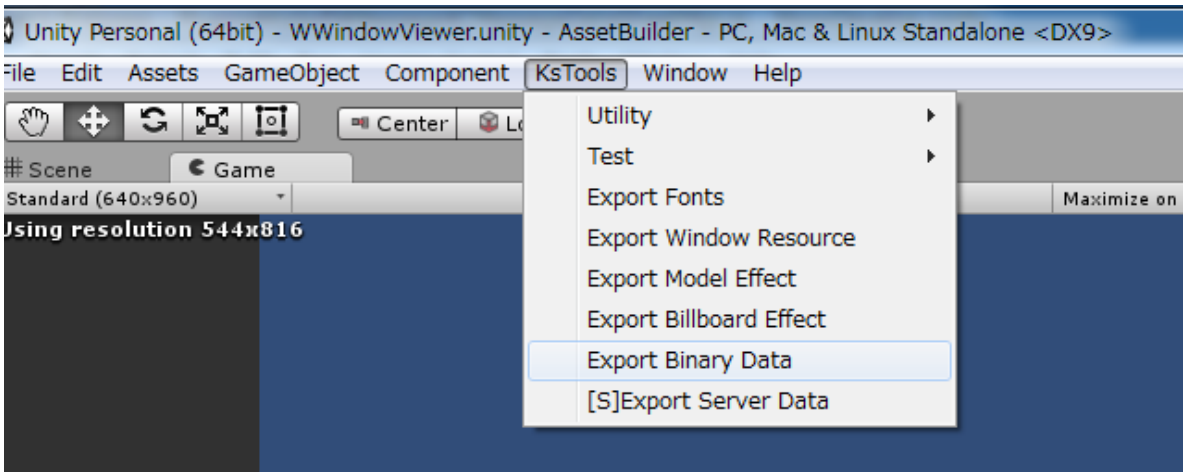
```
messagedata -v R messagedata.xlsx caption

output file names:
  messagedata.JP.bin
  messagedata.EN.bin
  etc
```

アセットバンドル化

標準で用意しているアセットバンドル化の手順です。

バイナリを自前で読み込みデータを展開する方法もあります。

メニュータブから、[Tools]->[KsSoft]->[Export Binary Data] を選択してください。



ロケールデータデータファイルを追加したときは、 *KsSoftConfig* を編集/追加してください。

第 8 章

ウィンドウスクリプト:

8.1 ウィンドウスクリプト (wra) 文法

8.1.1 wra 文法

wra は、大きく二つのブロックから構成されています。

- ウィンドウ定義ブロック
- コントロール定義ブロック

必ず、ウィンドウ定義ブロックが先頭にきて、その後にコントロール定義ブロックが続くことになります。
ウィンドウ ID は、 マルチ *ID* を使って指定してください。

```
// window define block
WINDOW(Window ID) {
   Property 0;
   Property 1;
       :
   Property n;
};

// control define blocks
ControlKind(name 0) {
   Property 0;
   Property 1;
       :
   Property n;
};
ControlKind(name 1) {
   Property 0;
   Property 1;
       :
   Property n;
```

(次のページに続く)

(前のページからの続き)

```
};
      :
ControlKind(name n) {
   Property 0;
   Property 1;
       :
   Property n;
};
```

```
#include "wr.h"
$y = 160;
#pragma RESOURCE ON

WINDOW(250_000_00010) {
    RESOURCE;
    PATH = NETWORKPATH;
    ID = 001_000_00000;
    TEX_ID = 100_000_00000;
    CAPTION = 000_000_0000;
    STYLE = NOTITLEBAR|NOFRAME|ANCHOR_CENTER|TOP;
    SIZE = {100},{100};
    PRIORITY = PROGRESSBAR_PRIORITY;
};
METER(ProgressTotal) {
    ID = 000_001_00000;
    STYLE = ANCHOR_BOTTOM;
    SIZE = -128{100};
    POSITION = 0,$y;
    TEX_ID = 0,"MTRB";
    COLOR = COLOR32(255.0,255.0,255.0,255.0);
    TEX_ID1 = 0,"MTR";
    COLOR1 = COLOR32(255.0,255.0,255.0,255.0);
};
$y -=64;
METER(ProgressPart) {
   ID = 000_001_00010;
   STYLE = ANCHOR_BOTTOM;
   SIZE = -128{100},32;
   SIZE = -128{100},32;
   POSITION = 0,$y;
   TEX_ID = 0,"MTRB";
   COLOR = COLOR32(255.0,255.0,255.0,255.0);
   TEX_ID1 = 0,"MTR";
   COLOR1 = COLOR32(255.0,255.0,255.0,255.0);
};
```

8.1.2 変数と式

wra ファイル内で変数とその変数を使った式が使えます。**$** から始まるものが変数となります。使える変数の型は実数になります。また、式をプロパティ内にも記述可能です。

```
ID = 000_000_00010 + (3 * $val);
```

変数の宣言

変数を使うときは、必ず初期値が付いている状態で宣言する必要があります。未初期化な変数はエラーになります。

```
$i = 3;              //integer(Decimal number)
$h = 0xfff123;       //integer(Hexadecimal number)
$f = 3.14;           //real number
$m = 001_002_00003; //Multi (8bit,8bit,16bit)
```

式において使える演算子

演算子	説明
+,-,*,/	四則演算
%	剰余
|,&,^,~	ビット演算 (or,and,xor,not)
<<,>>	シフト演算
**	べき乗演算

式においての定数表現

定数例	説明
14	10 進整数
0x12ffffff	16 進整数
000_001_00010	マルチ ID(8bit,8bit,16bit 形式)
0.345	浮動小数点

文字列の演算

文字列は、数字と違い変数に代入することはできませんが、連結するための演算は行えます。連結するには、以下の形式で行います。

- 文字 + 文字
- 文字 . 文字

```
PATH = "test/" + "folder/";
```

8.1.3 プリプロセッサ

C 言語,C++ 言語とほぼ同じプリプロセッサが使えます。

プリプロセッサ	説明
// コメント	ラインコメント
/* コメント */	ブロックコメント
#include ファイル名	ファイルをインクルードする
#define 記号定数の定義	記号定数の定義
#define 関数マクロ	関数マクロ
#if defined(記号定義) 〜 #endif	条件コンパイル
#ifdef 〜 #endif	条件コンパイル
#ifndef 〜 #endif	条件コンパイル
#pragma once	多重インクルード防止

次に、wra 独自のプリプロセッサ定義について列挙しておきます。

プリプロセッサ	説明
#pragma RESOURCE マルチ *ID*	アセットバンドルとして出力する 例 #pragma RESOURCE 000_014_000000
#pragma RESOURCE パス	リソースデータとして指定されたパスに出力する 例 #pragma RESOURCE = Resources/windows.asset
#pragma PATH 出力パス	出力パスを変更する デフォルトは、以下のフォルダ #pragma PATH ../wrc/base
#pragma BASECLASS ベースクラス	ベースクラスを変更する デフォルトは、以下のクラス #pragma BASECALSS CWindowBase

第 9 章

柔軟な座標、サイズ指定方法:

9.1 割合指定:柔軟な座標、サイズ指定方法

ウィンドウシステムでは、ほぼすべてのサイズ、オフセット、そして座標を指定する際、親のサイズの任意割合を指定することができます。

例えば、特定のコントロールの横幅はウィンドウサイズの 75% のサイズに指定したりすることができます。

これによって画面サイズが変わってもレイアウトが崩れずらいウィンドウを記述することができます。

9.1.1 記述方法

割合を { 〜 }で囲ってあげて、記述します。値は百分率を用います。オフセット及び割合の値に、式を記述することも可能です。

オフセット

オフセットのみ指定する方法です。

以下の例では、40x40 のサイズが指定されます。

```
SIZE = 20 * 2,16 + 24;
```

{割合}

割合のみ指定することもできます。

以下の例では、スクリーンサイズが 960x640 である場合、480x320 のサイズが指定されます。

```
SIZE = {50},{25 + 25};
```

オフセット + {割合}

{割合} オフセット

オフセットと割合を指定する場合に用います。

以下の例では、スクリーンサイズが 960x640 である場合、520x360 のサイズが指定されます。

```
SIZE = 40 + {20 + 30},20 * 2{50};
   or
SIZE = {20 + 30} + 40,{50} + 20 * 2;
```

9.1.2 間違った記述例

次のように割合やオフセットを複数入れることはできません。

```
SIZE = 10 + {20} + 30,40;

SIZE = 40 + {20} + {30},40;

SIZE = 40 + 0.5 * {20},40;
```

いずれも間違った記述です。

9.2 仮想的な GUI スクリーンサイズの設定方法

実機のスクリーンサイズに依存せずに、開発者が指定したサイズに仮想的に固定することができます。

設定した横サイズ (若しくは縦サイズ) と実スクリーンサイズが異なるときは自動的に全ての GUI が拡大/縮小されます。

これによって様々な画面サイズを持った携帯端末に比較的容易に対応が可能になります。

こちらの *setUIResolution* を参考にしてください。

第 10 章

ウィンドウ:

10.1 ウィンドウの配置について

10.1.1 スクリーン:

SCREEN を設定することによって、画面内に仮想的なスクリーンを定義することができます。

ウィンドウ座標やサイズはこのスクリーンを基準に決定されます。

これによって、アンカーの中心位置や基準位置を調整することができます。

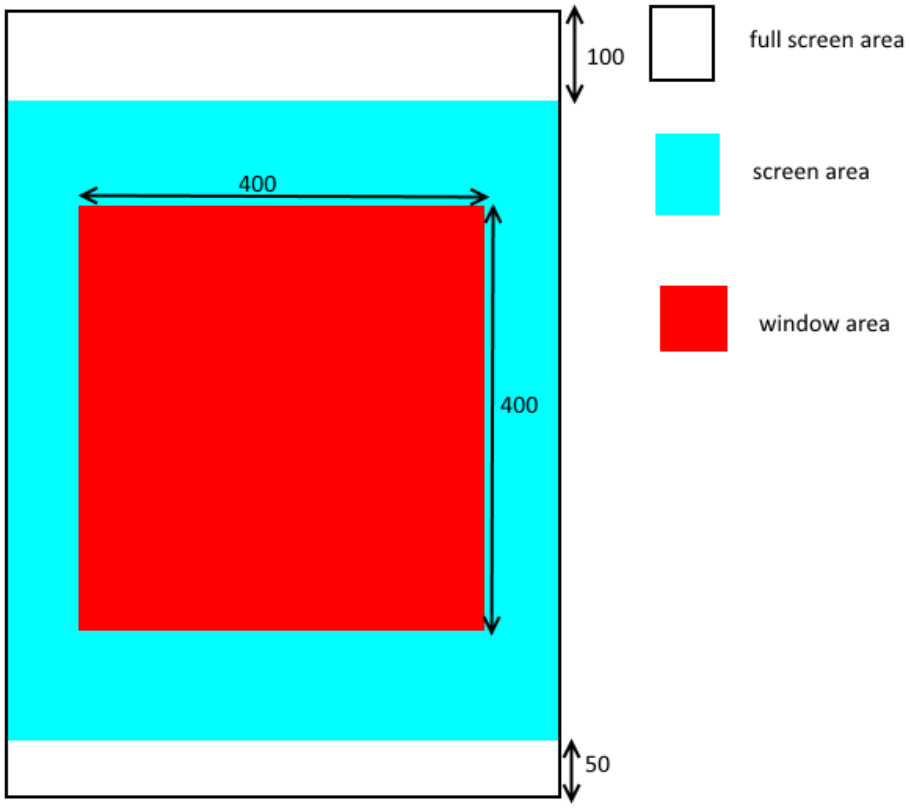
スクリーンは、特に明示的に設定しない限り、画面サイズと同じになり、以下と同等の記述になります。

```
SCREEN = 0,0,{100},{100};   //this mean is full screen.
```

割合指定 を使うことによって、画面サイズの割合を用いて指定できます。

```
STYLE = ANCHOR_CENTER;
SCREEN = 0,100,0{100},-50{100};
SIZE = 400,400;
```

この場合、下図のようにウィンドウが配置されます。



10.1.2 アンカー：

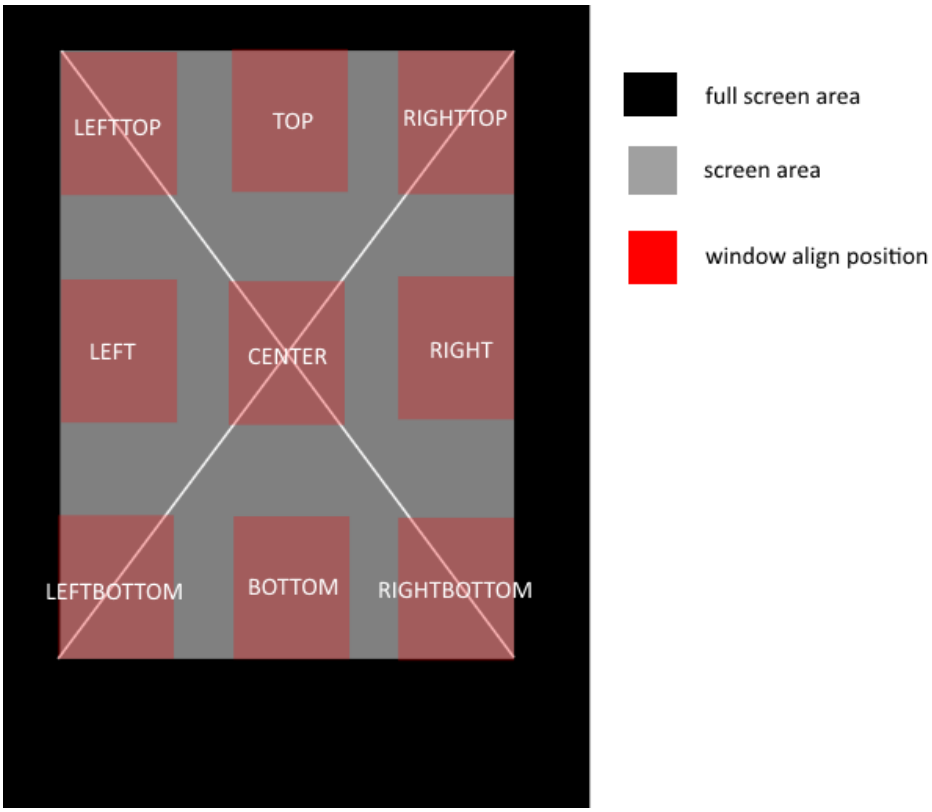
ウィンドウを配置する際、スクリーン上の原点座標をそれぞれ設定可能です。

その原点座標から、相対位置で座標を指定できます。

これによって、スクリーンサイズが変わってもウィンドウの配置が崩れずらくなります。

ウィンドウシステムでは、この原点位置の指定をアンカーと呼んでいます。

アンカーは、下図の通り 9 か所から自由に設定可能です。



10.2 ウィンドウプライオリティについて

通常のウィンドウは、後にクリエイトしたものほど優先度が高く表示されます。

明示的に優先度を制御したいときは、 *PRIORITY* に値を設定する必要があります。

大きいほど手前に表示します。

ただし、次のスタイルが指定されていると、*PRIORITY* を設定した通常のウィンドウよりも優先度が高くなります。

- TOP
- POPUP
- TOPMOST

ただし、同じスタイルが指定されているウィンドウ間なら、*PRIORITY* を使って優先度制御ができます。

例えば、二つのウィンドウが TOP を指定してクリエイトされているとき、そのウィンドウ間の優先順位は、*PRIORITY* で決定されます。

また、TOP,POPUP,TOPMOST 間の優先順位は次の通りとなります。

TOP < POPUP < TOPMOST

ウィンドウ描画優先順位フラグ	説明
TOP	ウィンドウの表示優先度を最大にする。 POPUP、TOPMOST よりは優先度は低い。 TOP 同士では、PRIORITY プロパティに設定されている優先度に従います。
POPUP	ウィンドウの表示優先度を最大にします。 TOP より優先され、TOPMOST よりは優先度は低い。 POPUP 同士では、PRIORITY プロパティに設定されている優先度に従う。 ウィンドウ以外をタッチしたとき、自動的に onClose が呼ばれる。
TOPMOST	ウィンドウの表示優先度を最大にする。 TOP、POPUP よりは優先度が高い。 TOPMOST 同士では、PRIORITY プロパティに設定されている優先度に従う。
NOECLIPSE	ウィンドウが開いたときに他のウィンドウを暗くしない (TOP/POPUP 時のみ有効)

10.3 WINDOW

WINDOW(ウィンドウ ID) {

プロパティ 1;

プロパティ 2;

:

:

プロパティ n;

};

ウィンドウ ID は、 マルチ *ID* で指定してください。

wra ファイル名が **TestWindow.wra** ならば、TestWindowBase.cs というファイルを自動生成します。

```
TestWindowBase.cs
```

使うときは、TestWindow.cs を作り、TestWindowBase を継承してください。

TestWindowBase.cs が出力されるパスは、スクリプト上で *PATH* を使って指定できます。

```
using UnityEngine;
using System;

public class TestWindow : TestWindowBase {
```

(次のページに続く)

(前のページからの続き)

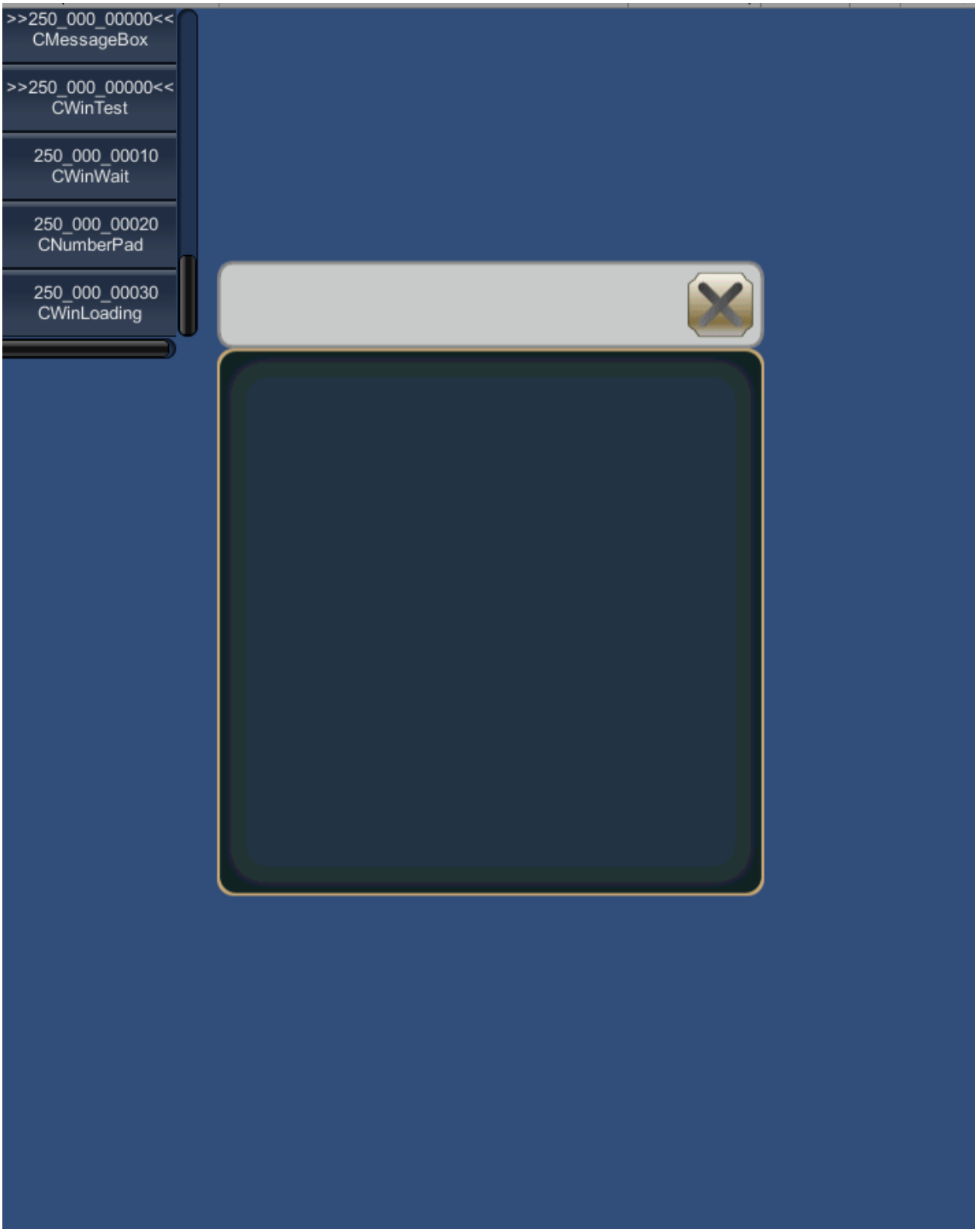
```
    :
    :
}
```

10.3.1 記述例

最少のウィンドウスクリプト

```
WINDOW(250_000_00000) {
  STYLE = ANCHOR_CENTER;
  SIZE = 400,400;
};
```

このスクリプトで次のようなウィンドウが表示されます。



ウィンドウの原点位置は、フレームの左上です (タイトルバーの左上ではありません)。

次のように記述するとタイトルバーが見えなくなります。

```
WINDOW(250_000_00000) {
  STYLE = ANCHOR_TOP;
  SIZE = 400,400;
};
```

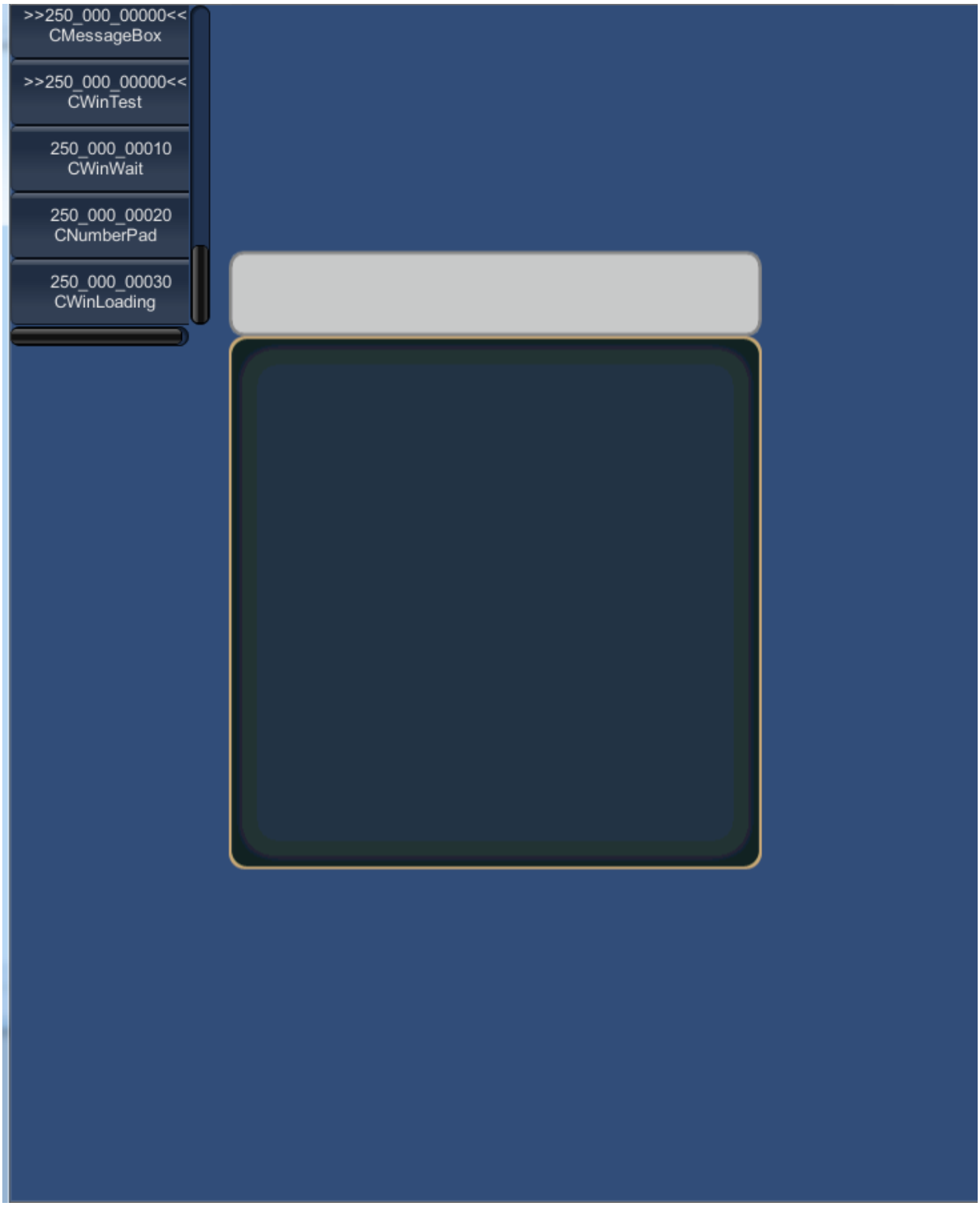
次のように記述するとタイトルバーが表れます。

```
WINDOW(250_000_00000) {
  STYLE = ANCHOR_TOP;
  POSITION = 0,-64;
  SIZE = 400,400;
};
```

クローズボタンのないウィンドウスクリプト

クローズボタンを削除するには次のようにスタイルを設定します。

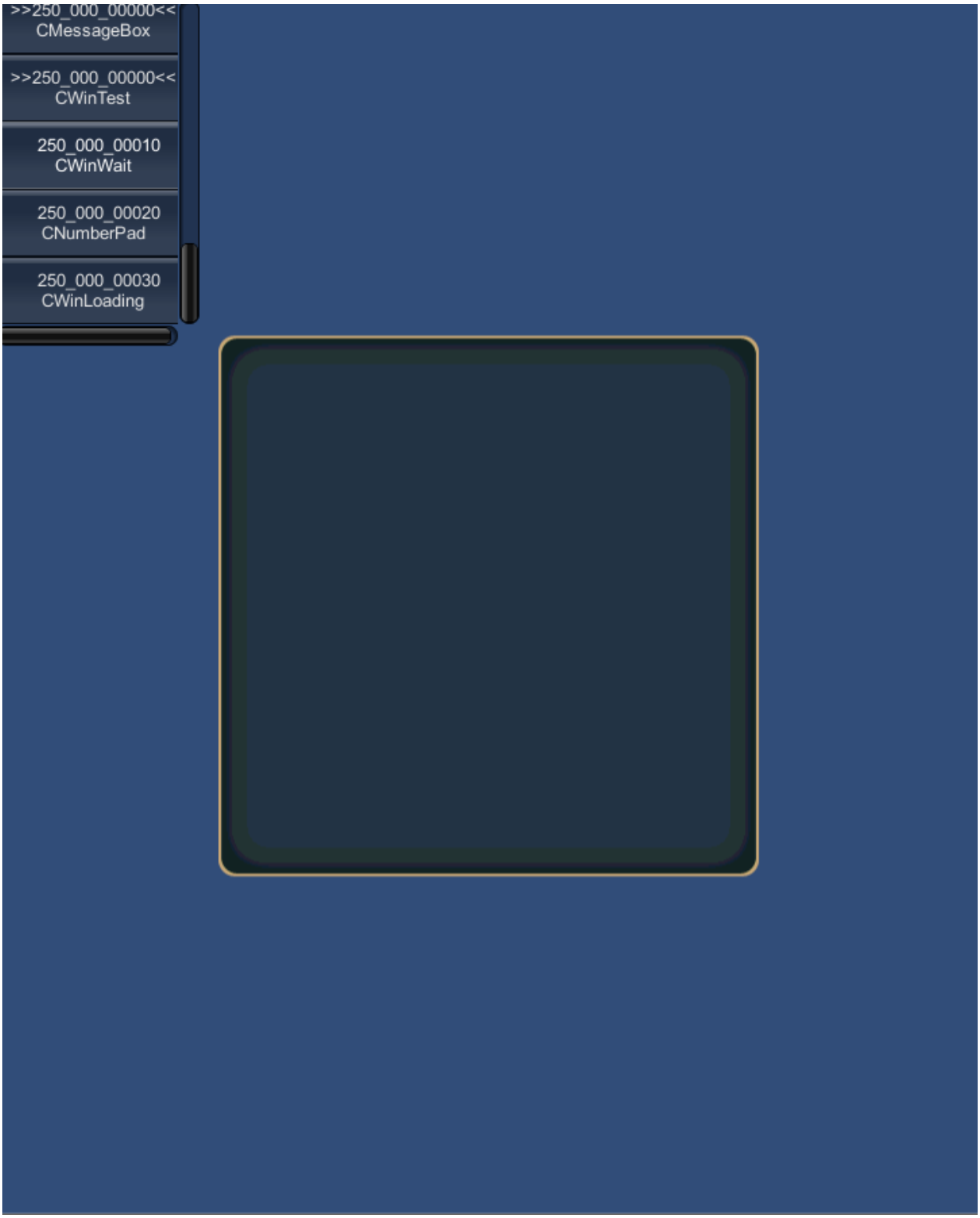
```
WINDOW(250_000_00000) {
  STYLE = NOCLOSE|ANCHOR_CENTER;
  SIZE = 400,400;
};
```



タイトルバーを無くしたウィンドウスクリプト

タイトルバーを無くしたウィンドウスクリプト

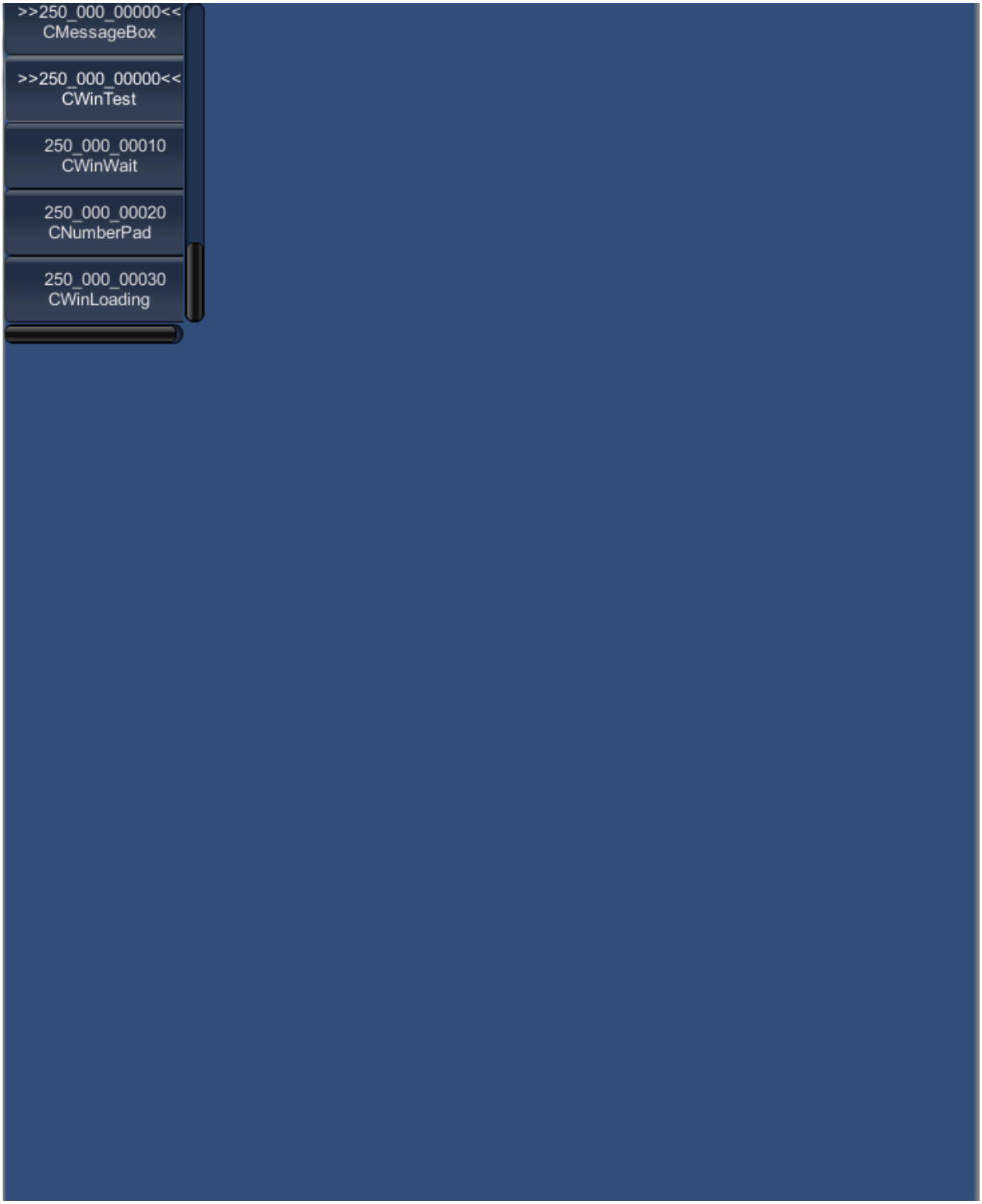
```
WINDOW(250_000_00000) {
  STYLE = NOTITLEBAR|ANCHOR_CENTER;
  SIZE = 400,400;
};
```



全てのデフォルト表示を消したウィンドウスクリプト

全てのデフォルト表示を消したウィンドウスクリプト

```
WINDOW(250_000_00000) {
  STYLE = NOTITLEBAR|NOFRAME|ANCHOR_CENTER;
  SIZE = 400,400;
};
```



全ての表示をコントロールを直接記述することで見た目のカスタマイズを最大限行えます。

ウィンドウのドラッグ移動について

STYLE に DRAG を付けることによって、ドラッグによるウィンドウの移動を許可します。

タイトルバー、フレームやユーザー定義のコントロールをドラッグしたときに、ウィンドウを移動できます。ただし、コントロールのスタイルに DRAG が付いているときは、ウィンドウの移動は発生しません。以下の例を試してください。

```
WINDOW(250_000_00000) {
  STYLE = ANCHOR_CENTER|DRAG|NOFRAME;
  POSITION = 0,0;
  CLOSE_POSITION = 400,0;
  SIZE = 400,400;
};
FRAME(Test) {
  STYLE = DRAG;
  SIZE = 400,400;
};
```

この例では、フレームをドラッグすると、フレームが複製され、その複製が移動します。その反面、タイトルバーをドラッグするとウィンドウを移動することができます。

10.3.2 プロパティ

RESOURCE = マルチ ID

[Tools]->[KsSoft]->[Export Window Resource] を行ったとき、アセットバンドルとして出力します。

同一の マルチ *ID* を指定していると、一つにまとめてアセットバンドル化してくれます。

```
RESOURCE = 000_014_00000;
```

この例では、000_014_00000.unity3d というアセットバンドルに出力します。

RESOURCE = パス

指定したパスに、リソースデータとして出力されます。

同一のパスを指定したときは、一つにまとめたリソースを出力してくれます。

```
RESOURCE = "Assets/KsSoft/Resources/windows";
```

この例では、Assets/KsSoft/Resources/windows.asset というリソースを出力します。

PATH = パス文字列

ウィンドウベースクラスを出力するパスを設定します。

カレントパスは、Unity プロジェクトの直下になります。

```
PATH = "../../../client/Assets/Script/TestWindow.cs";
```

TEX_ID = テクスチャ ID

デフォルトのテクスチャ ID を設定します。

ここで設定した値は、各コントロールにおいて、テクスチャ ID を省略したとき、適用されます。

```
TEX_ID = 010_000_00010;
```

TEX_ID 0～7 = テクスチャ ID

デフォルトのテクスチャ ID を設定します。

コントロール内のテクスチャ ID を省略したとき、この値が使われます。

また、フレーム、タイトルバーに使うテクスチャもここで指定できます。

CAPTION = キャプション ID

ウィンドウのキャプション文字を設定します。

STYLE に対して **NOTITLEBAR** が付いているときは、無視されます。

```
CAPTION = 020_000_00010;
```

POSITION = X,Y

ウィンドウ表示位置を決定します。表示位置は、STYLE で指定するアンカーによって変わります。

座標は スクリーン からの割合指定が可能です。

```
display position = (x,y) + screen size * ratio/100;
```

```
POSITOIN = 30{50},-40{50};
```

この例では、スクリーンサイズが 640x960 であるならば、実際のウィンドウの座標は次のように計算されます。

:: x = 30 + 640 * 50/100 = 350 y = -40 + 960 * 20/100 = 440

SIZE = 横サイズ, 縦サイズ

ウィンドウサイズを指定します。

サイズは スクリーン からの割合指定が可能です。

```
display size = size + screen size * ratio/100;
```

```
SIZE = 100{50},50{20};
```

この例では、スクリーンサイズが 640x960 であるならば、実際のウィンドウサイズは次のように計算されます。

:: width = 100 + 640 * 50/100 = 420 height = 50 + 960 * 20/100 = 242

SCREEN = 左上 X, 左上 Y, 横サイズ, 縦サイズ

スクリーンの位置とサイズを指定します。

ウィンドウは、指定されたスクリーンに沿って大きさや位置が決定されます。

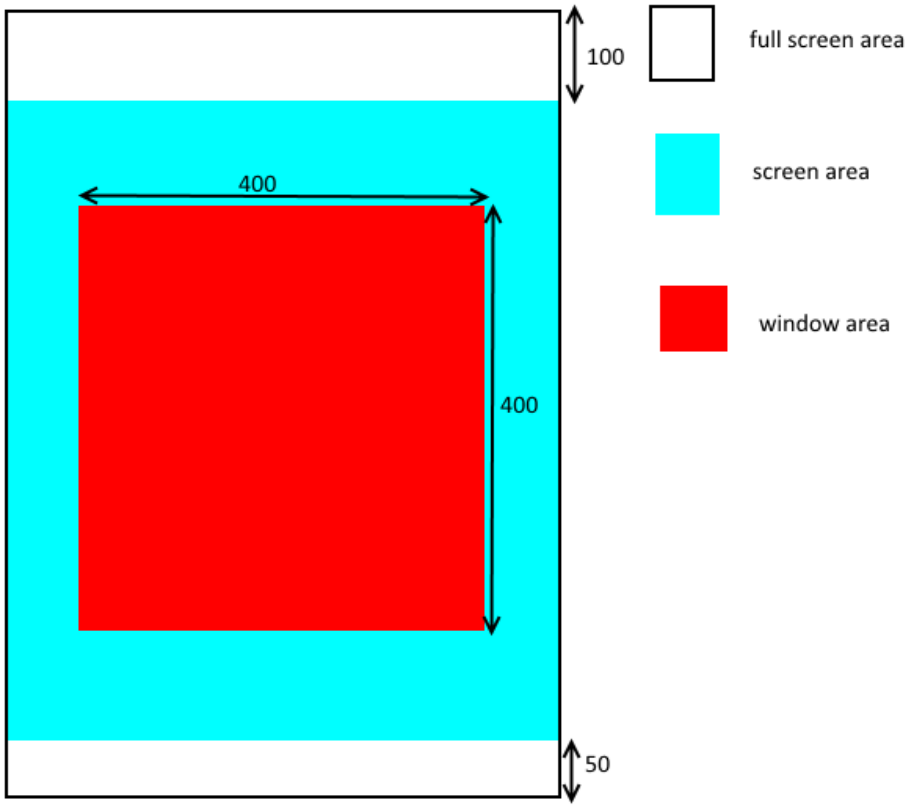
スクリーンを指定しないときは、全画面と同義になります。

```
SCREEN = 0,0,0{100},0{100};      //Same as Full Screen
```

４つのパラメータ全て、割合指定が可能です。

```
STYLE = ANCHOR_CENTER;
SCREEN = 0,100,0{100},-50{100};
SIZE = 400,400;
```

この例では、下図のようにウィンドウを配置します。



セーフエリア

ノッチ付の端末に対応するためにセーフエリアを基準に SCREEN を設定できます。

```
SCREEN = SAFEAREA(0,0,{100},{100});
```

PRIORITY = 表示プライオリティ

ウィンドウの表示プライオリティを設定します。大きいほど、表示優先度が高くなり、手前に表示されます。*TEXTURE_ZOFFSET* との違いに気を付けてください。

```
PRIORITY = 32;
```

TEXTURE_ZOFFSET = テクスチャ ID, Z オフセット

本ウィンドウシステムは、レンダリング最適化のために、同一テクスチャを使っているコントロールは全て単一のメッシュでレンダリングしようとします。

よって、各コントロール間のプライオリティは、あくまでも同一テクスチャを用いているコントロール間の表示優先度になります。

各テクスチャ間の表示優先度を変更するためには、TEXTURE_ZOFFSET を使います。これによって、特定のテクスチャを用いるメッシュの表示優先順位を変更できます。

小さいほど、表示優先度が高くなります。

```
TEXTURE_ZOFFSET = 014_000_00010,-1;     //Change to display on front
```

STYLE = スタイルフラグ **0**|スタイルフラグ **1**|..|スタイルフラグ **n**

表示位置アンカーフラグ	説明
ANCHOR_DEFAULT	アンカー位置を左上に設定 ANCHOR_CENTER と同じ
ANCHOR_LEFTTOP	アンカー位置を左上に設定
ANCHOR_LEFT	アンカー位置を左に設定 縦にはセンタリング
ANCHOR_LEFTBOTTOM	アンカー位置を左に設定 縦には下辺を基準に配置
ANCHOR_TOP	アンカー位置を上辺に設定 横方向にはセンタリング
ANCHOR_CENTER	アンカー位置を画面中央にセンタリング
ANCHOR_BOTTOM	アンカー位置を底辺に設定 横方向にはセンタリング
ANCHOR_RIGHTTOP	アンカー位置を右上に設定
ANCHOR_RIGHT	アンカー位置を右に設定 縦にはセンタリング
ANCHOR_RIGHTBOTTOM	アンカー位置を右に設定 縦には下辺を基準に配置

ウィンドウ描画優先順位フラグ	説明
TOP	ウィンドウの表示優先度を最大にする。 POPUP、TOPMOST よりは優先度は低い。 TOP 同士では、PRIORITY プロパティに設定されている優先度に従います。
POPUP	ウィンドウの表示優先度を最大にします。 TOP より優先され、TOPMOST よりは優先度は低い。 POPUP 同士では、PRIORITY プロパティに設定されている優先度に従う。 ウィンドウ以外をタッチしたとき、自動的に onClose が呼ばれる。
TOPMOST	ウィンドウの表示優先度を最大にする。 TOP、POPUP よりは優先度が高い。 TOPMOST 同士では、PRIORITY プロパティに設定されている優先度に従う。
NOECLIPSE	ウィンドウが開いたときに他のウィンドウを暗くしない(TOP/POPUP 時のみ有効)

ウィンドウ機能制御用フラグ	説明
NOCLOSE	クローズボタンを配置しない。 NOTITLEBAR が付いているとクローズボタンは配置されない
NOMINIMIZATION	未実装
NOHELP	未実装
NOTITLEBAR	タイトルバーを表示しない。
NOFRAME	フレームを表示しない
DISABLE	機能を奪い、入力を受け付けなくする
DRAG	ドラッグによるウィンドウの移動を許可する
NOACTIVE	アクティブにならない
HIDE	ウィンドウ表示をオフにする
NOBRINGTOTOP	アクティブになった時にプライオリティを自動的にあげない。
OPENBOTTOM	背面にウィンドウを開く

```
STYLE = NOTITLEBAR|NOFRAME;
```

第 11 章

コントロール:

共通項目:

11.1 コントロールの配置について

11.1.1 座標の 割合指定：

コントロールの位置を親のサイズ (多くはウィンドウサイズ) の 割合 から計算することが可能です。

```
POSITION = 0{20},50{30};
```

X = 0 + ウィンドウ幅 * 50/100

Y = 50 + ウィンドウ高 * 30/100

11.1.2 アンカー：

ウィンドウ上の原点位置をコントロール毎に設定可能です。

その原点位置から、相対座標を用いてコントロールの位置を指定できます。

これによって、ウィンドウサイズが変わっても、レイアウトが崩れずらいウィンドウを作ることが可能になっています。

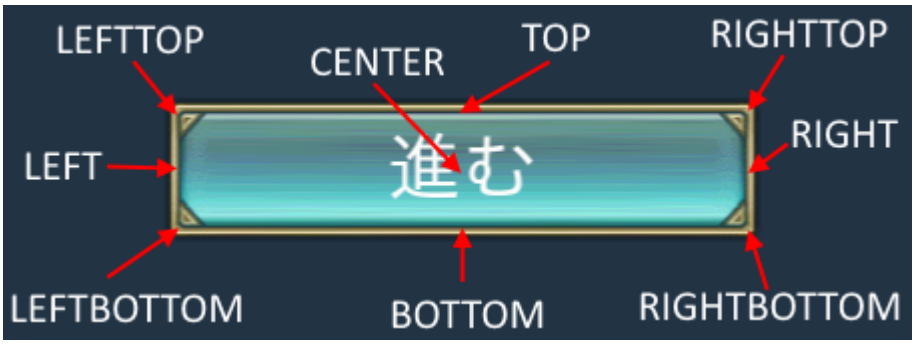
ウィンドウシステムでは、この原点位置の指定をアンカーと呼んでいます。

アンカーは、下図の通り 9 か所から自由に設定可能です。

11.1.3 ベースポジション:

コントロールの中心位置を自由に設定できます。

中心位置の指定は、下図の通り 9 か所から設定可能です。



11.1.4 アンカーとベースポジションの関係

アンカーに応じて、ベースポジションのデフォルトが決まっています。

例えば、アンカーが左上に設定されるとベースポジションも左上に設定されます。

自動的に設定される値以外のベースポジションを指定したいときは、明示的に設定してください。

以下にアンカーが設定されたときに自動的に設定されるベースポジションの値を列挙しておきます。

アンカー	ベースポジション
ANCHOR_DEFAULT	BASE_DEFAULT
ANCHOR_LEFTTOP	BASE_LEFTTOP
ANCHOR_LEFT	BASE_LEFT
ANCHOR_LEFTBOTTOM	BASE_LEFTBOTTOM
ANCHOR_TOP	BASE_TOP
ANCHOR_CENTER	BASE_CENTER
ANCHOR_BOTTOM	BASE_BOTTOM
ANCHOR_RIGHTTOP	BASE_RIGHTTOP
ANCHOR_RIGHT	BASE_RIGHT
ANCHOR_RIGHTBOTTOM	BASE_RIGHTBOTTOM

11.2 コントロールのプライオリティについて

ウィンドウをレンダリングする際、メッシュを用います。

メッシュはテクスチャ一つに対して一つ用意されます。

これによって、コントロールの数が増えてもドローコールが増えないようになっています。

ただし、コントロール間のプライオリティについては気を付ける必要があります。

同一テクスチャ (＝同一メッシュ) 内で 1 回のドローコールでレンダリングされるコントロール間では、柔軟にプライオリティの変更が可能です。

それに対して、別のテクスチャ (＝異なるメッシュ) 間でのプライオリティは、テクスチャ間のプライオリティに依存します。

最初に、テクスチャでソートされ、次に同一テクスチャ内でプライオリティでソートされることになります。

例えば、次のウィンドウはメッシュ (テクスチャ) が 4 つで構成されています。

このウィンドウは、フォントメッシュとテクスチャメッシュから構成されています。

- fn40
- fn30
- 100_000_00001
- 100_010_00000

11.2.1 同一のテクスチャ ID が指定されているとき

コントロールのプロパティである、 *PRIORITY* によって設定可能です。

PRIORITY の値が大きいほど手前に表示されます。

```
BUTTON(Button) {
  ID = 001_000_00010;
  POSITION = 100,100;
  SIZE = 64;
  PRIORITY = 5;
};
```

プライオリティが同一の時

プライオリティが同一の時は、コントロールの種類に応じて自動的にソートされます。

上にあるものほど、手前にレンダリングされます。

- SCROLLBAR
- LISTBOX
- LISTBOXEX
- CONTAINER
- TEXT
- RICHTEXT
- LOG
- LOGTEXT
- BUTTON
- CHECKBOX
- RADIO
- HELPBUTTON
- EDITBOX
- TEXTBOX
- METER
- ICON

- CARD
- TEXTURE
- RENDER
- RENDERICON
- RECASTICON
- LINE
- LABEL
- FRAME
- BAR

11.2.2 テクスチャ ID が異なるとき

テクスチャ ID が異なるときは、コントロールのプライオリティよりもテクスチャのＺオフセットが優先されます。

テクスチャのＺオフセットを設定するには、WINDOW のプロパティである、*TEXTURE_ZOFFSET* を使います。

値が大きいものほど、奥に表示されます。

```
WINDOW(100_100_00010) {
  TEX_ID = DEFAULT_TEXTURE;
  STYLE = POPUP|NOFRAME|NOTITLEBAR;
  TEXTURE_ZOFFSET = 100_100_00000,-1;
  TEXTURE_ZOFFSET = 100_010_00000,1;
  POSITION = 0,0;
  SCREEN = 0,0,{100},{-BASEY};
  SIZE = WIN_W,WIN_H;
};
```

この例では、次の順番に表示されます。

- 100_100_00000 手前
- DEFAULT_TEXTURE
- 100_010_00000 再背面

注釈: Ｚオフセットを明示的に指定しなかったとき、デフォルトテクスチャとして設定したメッシュより手前に、レンダリングされるようにシステム側でＺオフセットを設定します。その順番はシステム側で処理されるため、厳

密に制御が困難です。3 枚以上のテクスチャを一つのウィンドウで使うときは、明示的に *TEXTURE_ZOFFSET* を指示するようにしてください。

注釈: レンダーターゲットとして生成されたメッシュについても、 *TEXTURE_ZOFFSET* でレンダリング順序を指定可能です。*RENDER* 、 *RENDERICON* を参照ください。

11.3 コントロールのサイズについて

11.3.1 サイズの 割合指定：

コントロールのサイズを親のサイズ (多くはウィンドウサイズ) の割合から計算することが可能です。

```
SIZE = -16 + {50},16 + {20};
```

この例では次のように設定されます。

横サイズ = -16 + ウィンドウ幅 * 50/100

縦サイズ = 16 + ウィンドウ高 * 50/100

11.3.2 サイズのデフォルト値

コントロールのサイズに 0 を指定したり省略したとき、デフォルト値が設定されます。

デフォルト値は、テクスチャパーツのサイズを参照するようになっています。

ドットバイドットで表示したいときは、デフォルト値を設定すればよいでしょう。

11.4 コントロールの共通プロパティ

コントロールの共通プロパティについて列挙しておきます。

ただし、ここで紹介されているプロパティでもコントロールごとに値を設定しても意味のないものもあります。

詳しくは、各コントロールの説明を見てください。

11.4.1 ID = コントロール ID

コントロール ID を定義します。

```
ID = 001_000_00010;
```

注釈: 設定しなかったときは、自動的にハッシュ値から生成します。

11.4.2 POSITION = X, Y

表示位置を決定します。STYLE の アンカー 指定に応じて、基準位置が変わります。

```
POSITION = 32,{50} + 64;
```

座標は 割合指定 が可能です。

11.4.3 SIZE = 横サイズ, 縦サイズ

表示サイズを変更します。省略したとき、若しくは、0 を指定すると、テクスチャパーツからサイズを取得してきます。

```
SIZE = 64,32; //64x32
SIZE = ,32;   //Set the width of the texture part width
SIZE = 64;            //Set the height of the texture part height
SIZE = {50} - 25;
```

サイズは 割合指定 が可能です。

11.4.4 CONTENTS = { コントロール定義 … }

コンテンツを持つことができるコントロールには関連付けたいコントロールを列挙します。

```
CONTENTS = {
      BUTTON(A) {
          :
      };
      CHECKBOX(B) {
          :
      };
      LISTBOX(C) {
```

(次のページに続く)

(前のページからの続き)

```
        :
    };
```

11.4.5 CONTENTS_SIZE = 横サイズ, 縦サイズ

コンテンツのサイズを指定します。

```
CONTENTS_SIZE = {50},64;
```

コンテンツサイズは 割合指定 が可能です。

11.4.6 LINE_SPACE = 行間ピクセル値

コンテンツとコンテンツ間の隙間をこのプロパティで設定可能です。

```
LINE_SPACE = 8;        //Put an 8-dot space.
```

割合指定はできません。

11.4.7 GROUP = コントロール ID,…

グループ化したいコントロール ID を指定します。リストボックスとスクロールバーを 関連 付けたいときにも使います。

```
GROUP = RADIO(Spring),RADIO(Summer),RADIO(Autumn),RADIO(Winter),001_002_30000;
```

11.4.8 PRIORITY = プライオリティ値

プライオリティ値を設定します。値が大きいほど手前に表示されます。

```
PRIORITY = 3;
```

11.4.9 TEX_ID = テクスチャ ID, パーツ ID

11.4.10 TEX_ID = パーツ ID

テクスチャ ID とパーツ ID を指定します。

11.4.11 TEX_ID n = テクスチャ ID, パーツ ID

11.4.12 TEX_ID n = パーツ ID

テクスチャ ID とパーツ ID を指定します。

n = [0..7] で指定可能です。

n = 0 の時は、TEX_ID と同じテクスチャを操作します。

11.4.13 TEXTURE_OFFSET n = オフセット X, オフセット Y

テクスチャの表示オフセットを指定します。

n = [0..7] で指定可能です。

テクスチャオフセットは 割合指定 が可能です。

11.4.14 TEXTURE_SIZE n = 横サイズ, 縦サイズ

テクスチャ ID とパーツ ID を指定します。

n = [0..7] で指定可能です。

n = 0 の時は、 **SIZE** と同じ意味です。

```
TEXTURE_SIZE1 = 64,32;         //64x32
TEXTURE_SIZE2 = ,32;  //Set the width of the texture part width
TEXTURE_SIZE3 = 64;   //Set the height of the texture part height
```

テクスチャサイズは 割合指定 が可能です。

11.4.15 COLOR = R,G,B,A

カラーを指定します。

R,G,B については、0〜2 の間で指定してください。

1 を超えたとき、そのカラー成分が 2 倍までかけることができます。

A については、0〜1 の間で指定してくさい。

11.4.16 COLOR n = R,G,B,A

カラーを指定します。

R,G,B については、0〜2 の間で指定してください。

1 を超えたとき、そのカラー成分が 2 倍までかけることができます。

A については、0〜1 の間で指定してくさい。

n = [0..7] で指定可能です。

n = 0 の時は、 **COLOR** と同じプロパティです。

11.4.17 CAPTION = キャプション ID

文字列の ID を指定してください。

こちら を参照ください。

```
CAPTION = 010_000_00100;
```

11.4.18 CAPTION = "文字列"

文字列を設定します。

```
CAPTION = "Hellow world!";
```

11.4.19 CAPTION_COLOR = R,G,B,A

キャプションのカラーを指定できます。

0〜1 の間で指定してください。

11.4.20 CAPTION_OFFSET = X, Y

キャプションの位置を移動することができます。

キャプションオフセットは 割合指定 が可能です。

11.4.21 SE_ID = SE_ID

押されたときに鳴らす音の ID を設定します。デフォルトは、0 になっており、音がなりません。

11.4.22 STYLE = フラグ 0|フラグ 1|..|フラグ n

コントロールの表示アンカーを指定できます。

表示位置アンカーフラグ	説明
ANCHOR_DEFAULT	アンカー位置を左上に設定 ANCHOR_LEFTTOP と同じ コントロールの中心位置はデフォルトで、BASE_LEFTTOP になる。
ANCHOR_LEFTTOP	アンカー位置を左上に設定 コントロールの中心位置はデフォルトで、BASE_LEFTTOP になる。
ANCHOR_LEFT	アンカー位置を左に設定 縦にはセンタリング コントロールの中心位置はデフォルトで、BASE_LEFT になる。
ANCHOR_LEFTBOTTOM	アンカー位置を左に設定 縦には下辺を基準に配置 コントロールの中心位置はデフォルトで、BASE_LEFTBOTTOM になる。
ANCHOR_TOP	アンカー位置を上辺に設定 横方向にはセンタリング コントロールの中心位置はデフォルトで、BASE_TOP になる。
ANCHOR_CENTER	アンカー位置を画面中央にセンタリング コントロールの中心位置はデフォルトで、BASE_CENTER になる。
ANCHOR_BOTTOM	アンカー位置を底辺に設定 横方向にはセンタリング コントロールの中心位置はデフォルトで、BASE_BOTTOM になる。
ANCHOR_RIGHTTOP	アンカー位置を右上に設定 コントロールの中心位置はデフォルトで、BASE_RIGHTBOTTOM になる。
ANCHOR_RIGHT	アンカー位置を右に設定 縦にはセンタリング コントロールの中心位置はデフォルトで、BASE_RIGHT になる。
ANCHOR_RIGHTBOTTOM	アンカー位置を右に設定 縦には下辺を基準に配置 コントロールの中心位置はデフォルトで、BASE_RIGHTBOTTOM になる。

コントロールの中心位置を指定できます。

中心位置変更フラグ	説明
BASE_DEFAULT	アンカーフラグに応じて変わる。 特に指定しなければ、これになります。 各アンカーフラグの説明を参照
BASE_LEFTTOP	中心位置をコントロールの左上に設定
BASE_LEFT	中心位置を左に設定 縦にはコントロールの真中
BASE_LEFTBOTTOM	中心位置を左下に設定
BASE_TOP	中心位置を上辺に設定 横にはコントロールの中心
BASE_CENTER	中心位置をコントロールの中心
BASE_BOTTOM	中心位置を底辺に設定 横方向にはコントロールの真中
BASE_RIGHTTOP	中心位置をコントロールの右上に設定
BASE_RIGHT	中心位置を右に設定 横にはコントロールの中心
BASE_RIGHTBOTTOM	中心位置を右下に設定

キャプションの中心位置を指定できます。

ボタン、ラベル、バー等が持つキャプションの位置を調整できます。

TEXT 、*RICHTEXT* や *LOGTEXT* の位置調整には使えません。

テキストアンカー	説明
TEXT_CENTER	テキスト表示を中心に合わせます。
TEXT_LEFT	テキスト表示を左に合わせます。
TEXT_RIGHT	テキスト表示を右に合わせます。

キャプションのフォント装飾を変更できます。

キャプション装飾種類	説明
TEXT_NORMAL	装飾なし
TEXT_BOLD	太字
TEXT_DENT	へこんだ感じの文字
TEXT_SHADOW	影付き文字

機能を制限するスタイルは以下のものがあります。

機能制限スタイル	説明
HIDE	表示を隠す。
DRAG	ドラッグ可能にする。
DISABLE	押せなくし、暗くする。
NOHIT	押せなくする。
HIT	押せるようにする。

各コントロール詳細:

11.5 TEXT

C#: *CWinCtrlText*

テキストを表示するためのコントロールです。

途中で文字色を変えたりしたいときは、 *RICHTEXT* を使ってください。改行は可能です。

デフォルトでは、押したことを検出できません。コールバックが必要なときは、STYLE に HIT を指定してください。

```
TEXT(Control name) {

  Property 1;

  Property 2;

     :

     :

  Property n;

};
```

11.5.1 記述例

```
WINDOW(255_000_00001) {
  STYLE = NOTITLEBAR|ANCHOR_CENTER;
  POSITION = 0,100;
  TEX_ID = 100_000_00001;
  CAPTION = 000_000_00010;
  SIZE = 256,256;
```

(次のページに続く)

(前のページからの続き)

```
};

TEXT(TEST) {
  ID = 000_002_00001;
  CAPTION = 000_000_00006;
  FONT_KIND="fn40";
  POSITION = 0,0;
};
```

11.5.2 プロパティ

ID = コントロール ID

コントロール ID を定義します。

```
ID = 001_000_00010;
```

注釈: 設定しなかったときは、自動的にハッシュ値から生成します。

POSITION = X, Y

表示位置を決定します。STYLE に応じて、表示位置の基準を柔軟に変更可能です。

注釈: 表示位置アンカーフラグと中心位置変更フラグによって、位置が決定します。

テキストアンカーを設定しても表示に影響与えません

```
POSITION = 32,64;
```

座標は 割合指定 が可能です。

FONT_KIND = フォント種類

フォントの種類を指定します。

```
FONT_KIND = "fnt32";
```

CAPTION = キャプション ID

文字列の ID を指定してください。

こちら を参照ください。

```
CAPTION = 010_000_00100;
```

CAPTION = "文字列"

文字列を設定します。

```
CAPTION = "Hellow world!";
```

CAPTION_COLOR = R,G,B,A

キャプションのカラーを指定できます。

0〜1 の間で指定してください。

CAPTION_OFFSET = X, Y

キャプションの位置を移動することができます。

キャプションオフセットは 割合指定 が可能です。

CONTENTS_SIZE = 改行サイズ

文字列が改行サイズを超えたとき、そこから自動で改行をします。指定しないときは、自動的に改行しません。

```
CONTENTS_SIZE = 128;  //128pixel
```

改行サイズは 割合指定 が可能です。

LINE_SPACE = 行間ピクセル値

行間をこのプロパティで設定可能です。

割合指定はできません。

```
LINE_SPACE = 8;       //Put an 8-dot space.
```

STYLE = フラグ 0|フラグ 1|..|フラグ n

コントロールの表示アンカーを指定できます。

表示位置アンカーフラグ	説明
ANCHOR_DEFAULT	アンカー位置を左上に設定 ANCHOR_LEFTTOP と同じ コントロールの中心位置はデフォルトで、BASE_LEFTTOP になる。
ANCHOR_LEFTTOP	アンカー位置を左上に設定 コントロールの中心位置はデフォルトで、BASE_LEFTTOP になる。
ANCHOR_LEFT	アンカー位置を左に設定 縦にはセンタリング コントロールの中心位置はデフォルトで、BASE_LEFT になる。
ANCHOR_LEFTBOTTOM	アンカー位置を左に設定 縦には下辺を基準に配置 コントロールの中心位置はデフォルトで、BASE_LEFTBOTTOM になる。
ANCHOR_TOP	アンカー位置を上辺に設定 横方向にはセンタリング コントロールの中心位置はデフォルトで、BASE_TOP になる。
ANCHOR_CENTER	アンカー位置を画面中央にセンタリング コントロールの中心位置はデフォルトで、BASE_CENTER になる。
ANCHOR_BOTTOM	アンカー位置を底辺に設定 横方向にはセンタリング コントロールの中心位置はデフォルトで、BASE_BOTTOM になる。
ANCHOR_RIGHTTOP	アンカー位置を右上に設定 コントロールの中心位置はデフォルトで、BASE_RIGHTBOTTOM になる。
ANCHOR_RIGHT	アンカー位置を右に設定 縦にはセンタリング コントロールの中心位置はデフォルトで、BASE_RIGHT になる。
ANCHOR_RIGHTBOTTOM	アンカー位置を右に設定 縦には下辺を基準に配置 コントロールの中心位置はデフォルトで、BASE_RIGHTBOTTOM になる。

コントロールの中心位置を指定できます。

中心位置変更フラグ	説明
BASE_DEFAULT	アンカーフラグに応じて変わる。 特に指定しなければ、これになります。 各アンカーフラグの説明を参照
BASE_LEFTTOP	中心位置をコントロールの左上に設定
BASE_LEFT	中心位置を左に設定 縦にはコントロールの真中
BASE_LEFTBOTTOM	中心位置を左下に設定
BASE_TOP	中心位置を上辺に設定 横にはコントロールの中心
BASE_CENTER	中心位置をコントロールの中心
BASE_BOTTOM	中心位置を底辺に設定 横方向にはコントロールの真中
BASE_RIGHTTOP	中心位置をコントロールの右上に設定
BASE_RIGHT	中心位置を右に設定 横にはコントロールの中心
BASE_RIGHTBOTTOM	中心位置を右下に設定

キャプションのフォント装飾を変更できます。

キャプション装飾種類	説明
TEXT_NORMAL	装飾なし
TEXT_BOLD	太字
TEXT_DENT	へこんだ感じの文字
TEXT_SHADOW	影付き文字

機能を制限するスタイルは以下のものがあります。

機能制限スタイル	説明
HIDE	表示を隠す。
DRAG	ドラッグ可能にする。
DISABLE	押せなくし、暗くする。
HIT	押せるようにする。

11.6 RICHTEXT

C#: *CWinCtrlRichText*

リッチテキストを表示するためのコントロールです。

途中で文字色を変えたり、テクスチャを埋め込んだりすることが可能です。

RICHTEXT(コントロール名) {

プロパティ 1;

プロパティ 2;

:

:

プロパティ n;

};

11.6.1 記述例

```
WINDOW(255_000_00001) {
  STYLE = NOTITLEBAR|ANCHOR_LEFT;
  POSITION = 0,100;
  SIZE = 512,256;
};

RICHTEXT(Message) {
  ID = 00_000_00010;
  POSITION = 0,-64;
  CAPTION = "\a[3]TESTabc\f[fn16]smallFont\t123[100_000_00001,ATTR1,32,32,0]" +
            "\a[4]\w[001_002_00003,click here!]";
  CONTENTS_SIZE = 380, 64;
};
```

11.6.2 プロパティ

ID = コントロール ID

コントロール ID を定義します。

```
ID = 001_000_00010;
```

注釈: 設定しなかったときは、自動的にハッシュ値から生成します。

POSITION = X, Y

表示位置を決定します。STYLE の アンカー 指定に応じて、基準位置が変わります。

```
POSITION = 32,{50} + 64;
```

座標は 割合指定 が可能です。

FONT_KIND = フォント種類

フォントの種類を指定します。

```
FONT_KIND = "fnt32";
```

CAPTION = キャプション ID

文字列の ID を指定してください。

こちら を参照ください。

```
CAPTION = 010_000_00100;
```

CAPTION = "文字列"

文字列を設定します。

```
CAPTION = "Hellow world!";
```

リッチテキストのキャプションに コマンド を埋め込むことによって、様々な装飾が可能になっています。

CAPTION_COLOR = R,G,B,A

キャプションのカラーを指定できます。

0〜1 の間で指定してください。

CONTENTS_SIZE = 改行サイズ

文字列が改行サイズを超えたとき、そこから自動で改行をします。

```
CONTENTS_SIZE = 128;  //128pixel

CONTENTS_SIZE = {50} + 32;     // screen width * 0.5 + 32
```

改行サイズは 割合指定 が可能です。

LINE_SPACE = 行間ピクセル値

行間をこのプロパティで設定可能です。

```
LINE_SPACE = 8;        //Put an 8-dot space.
```

行間は割合指定はできません。

CAPTION_OFFSET = X, Y

キャプションの位置を移動することができます。

キャプションオフセットは 割合指定 が可能です。

STYLE = フラグ 0|フラグ 1|..|フラグ n

コントロールの表示アンカーを指定できます。

表示位置アンカーフラグ	説明
ANCHOR_DEFAULT	アンカー位置を左上に設定 ANCHOR_LEFTTOP と同じ コントロールの中心位置はデフォルトで、BASE_LEFTTOP になる。
ANCHOR_LEFTTOP	アンカー位置を左上に設定 コントロールの中心位置はデフォルトで、BASE_LEFTTOP になる。
ANCHOR_LEFT	アンカー位置を左に設定 縦にはセンタリング コントロールの中心位置はデフォルトで、BASE_LEFT になる。
ANCHOR_LEFTBOTTOM	アンカー位置を左に設定 縦には下辺を基準に配置 コントロールの中心位置はデフォルトで、BASE_LEFTBOTTOM になる。
ANCHOR_TOP	アンカー位置を上辺に設定 横方向にはセンタリング コントロールの中心位置はデフォルトで、BASE_TOP になる。
ANCHOR_CENTER	アンカー位置を画面中央にセンタリング コントロールの中心位置はデフォルトで、BASE_CENTER になる。
ANCHOR_BOTTOM	アンカー位置を底辺に設定 横方向にはセンタリング コントロールの中心位置はデフォルトで、BASE_BOTTOM になる。
ANCHOR_RIGHTTOP	アンカー位置を右上に設定 コントロールの中心位置はデフォルトで、BASE_RIGHTBOTTOM になる。
ANCHOR_RIGHT	アンカー位置を右に設定 縦にはセンタリング コントロールの中心位置はデフォルトで、BASE_RIGHT になる。
ANCHOR_RIGHTBOTTOM	アンカー位置を右に設定 縦には下辺を基準に配置 コントロールの中心位置はデフォルトで、BASE_RIGHTBOTTOM になる。

コントロールの中心位置を指定できます。

中心位置変更フラグ	説明
BASE_DEFAULT	アンカーフラグに応じて変わる。 特に指定しなければ、これになります。 各アンカーフラグの説明を参照
BASE_LEFTTOP	中心位置をコントロールの左上に設定
BASE_LEFT	中心位置を左に設定 縦にはコントロールの真中
BASE_LEFTBOTTOM	中心位置を左下に設定
BASE_TOP	中心位置を上辺に設定 横にはコントロールの中心
BASE_CENTER	中心位置をコントロールの中心
BASE_BOTTOM	中心位置を底辺に設定 横方向にはコントロールの真中
BASE_RIGHTTOP	中心位置をコントロールの右上に設定
BASE_RIGHT	中心位置を右に設定 横にはコントロールの中心
BASE_RIGHTBOTTOM	中心位置を右下に設定

キャプションの中心位置を指定できます。

テキストアンカー	説明
TEXT_CENTER	テキスト表示を中心に合わせます。
TEXT_LEFT	テキスト表示を左に合わせます。
TEXT_RIGHT	テキスト表示を右に合わせます。

キャプションのフォント装飾を変更できます。

キャプション装飾種類	説明
TEXT_NORMAL	装飾なし
TEXT_BOLD	太字
TEXT_DENT	へこんだ感じの文字
TEXT_SHADOW	影付き文字

機能を制限するスタイルは以下のものがあります。

機能制限スタイル	説明
HIDE	表示を隠す。
DISABLE	押せなくし、暗くする。
NOHIT	押せなくする。

リッチテキストで使用可能なコントロールコード

リッチテキストのキャプションに コマンド を埋め込むことによって、様々な装飾が可能になっています。

例えば、途中にテクスチャを挟み込むには次のような文字列をキャプションに設定します。

```
"texture test\t[001_001_01000,icon0,32,32,0]\ndisplay test"
```

コマンド	説明
\n or \r	改行
\ \	\文字
\f[フォント名]	以降のフォントを指定したものに切り替える
\a[0] or \a[1]	横:左揃え 縦:上辺
\a[2]	横:左揃え 縦:中心
\a[3]	横:左揃え 縦:下辺
\a[4]	横:センタリング 縦:上辺
\a[5]	横:センタリング 縦:中心
\a[6]	横:センタリング 縦:下辺
\a[7]	横:右揃え 縦:上辺
\a[8]	横:右揃え 縦:中心
\a[9]	横:右揃え 縦:下辺
\c[0]	カラーを白色
\c[1]	カラーを水色
\c[2]	カラーを紫色
\c[3]	カラーを青色
\c[4]	カラーを黄色
\c[5]	カラーを緑色
\c[6]	カラーを赤色
\c[7]	カラーをピンク色
\c[8]	カラーをオレンジ色
\c[9]	カラーをスカイブルー
\c[10]	カラーを黒
\C[00000000] ～ \C[FFFFFFFF]	カラーを 16 進数 (AARRGGBB) で指定
\wUserID[WindowID, キャプション]	指定した ID のウィンドウコマンドを発行する
\tUserID[テクスチャ ID, パーツ ID, 横, 縦,Y オフセット]	指定したテクスチャ内のパーツ ID の パーツ を埋め込む。
\B or \b	ボールド
\N	装飾無

\a コマンドを使ったときに、右揃え、センタリング、左揃えのアンカーが変化したとき、自動的に改行します。

ウィンドウコマンド \w

指定されたキャプション文字列をタッチすると *CWindowBase.onClick(CWinCtrlRichText cCtrl,CRichTextOne cText)* が発生します。

次のようなフォーマットになっています。

\wUserID[WindowID, キャプション]

- UserID: 整数で指定してください (省略可能です。デフォルトで 0 が設定されます)
- WindowID: マルチ ID で指定してください。
- キャプション: 文字列を指定してください。この文字列を押すとコールバックが発生します。

```
CAPTION = "\a[4]\w123[001_002_00003,click here!]";
```

テクスチャコマンド \t

文字列の途中にテクスチャを埋め込めます。

テクスチャをタッチすると *CWindowBase.onClick(CWinCtrlRichText cCtrl,CRichTextOne cText)* が発生します。

次のようなフォーマットになっています。

\tUserID[テクスチャ ID, パーツ ID, 横, 縦,Y オフセット]

- UserID: 整数で指定してください (省略可能です。デフォルトで 0 が設定されます)
- テクスチャ ID: マルチ ID で指定してください。
- 横, 縦: ピクセルサイズを指定してください。
- Y オフセット: 表示するときの Y 方向へのオフセットです (省略可能です)。

```
CAPTION = "Texture test\t456[001_001_01000,icon0,32,32,0]";
```

11.7 LOG

C#: *CWinCtrlLog*

チャットログのためのコントロールです。

コンテンツを複製し、それを一行とします。

リストボックスと違い、コンテンツの最大数が設定でき、それ以上のコンテンツを追加しようとしたとき、内部で循環 (一番古いものが破棄されます) するようになっています。

LOGTEXT 以外のコントロールもコンテンツとして入れることができます。

このコントロールはスクロール可能なコントロールです。

STYLE に *SCROLL_LOCK* をつけておくと、onDrag コールバックが呼び出され、ドラッグした方向を検出できます。DRAG をつけても、これらコントロールには効果がないことに気を付けてください。

注釈: LOG 内の **CONTENTS** には、必ず一つの **LOGTEXT** を入れておく必要があります。

LOG(コントロール名) {

　プロパティ 1;

　プロパティ 2;

　　:

　　:

　プロパティ n;

};

11.7.1 記述例

```
LOG(Chat) {
  ID = 000_000_00030;
  STYLE = ANCHOR_LEFTBOTTOM;
  POSITION = 0,OFFSET_Y;
  SIZE = WINDOW_SIZE_FULL, 80-2;
  CONTENTS_SIZE = WINDOW_SIZE_FULL, 80-2;
  CONTENTS = {
    ICON(Chat) {
         :
    }
    TEXT(Name) {
         :
    }
    FRAME(Balloon) {
         :
    }
    LOGTEXT(Chat) {
         :
```

(次のページに続く)

(前のページからの続き)

```
    }
  }

  COLOR = COLOR32(0,0,0,255);
  LINE_SPACE = 10;
  GROUP = SCROLLBAR(Chat);
};
SCROLLBAR(Chat) {
  ID = 000_001_00001;
  STYLE = ANCHOR_RIGHTTOP;
  COLOR = 1,1,1,0.5;
  POSITION = -2,-5;
  SIZE = 0,-10;
};
```

11.7.2 プロパティ

代表的なデフォルト値

```
COLOR = 1,1,1,1;
CAPTION_COLOR = 1,1,1,1;
```

ID = コントロール ID

コントロール ID を定義します。

```
ID = 001_000_00010;
```

注釈: 設定しなかったときは、自動的にハッシュ値から生成します。

POSITION = X, Y

表示位置を決定します。STYLE の アンカー 指定に応じて、基準位置が変わります。

```
POSITION = 32,{50} + 64;
```

座標は 割合指定 が可能です。

SIZE = 横幅, 縦幅

リストボックスの表示領域を定義します。

表示領域をはみ出たコントロールは、クリッピングされます。

```
SIZE = {100} - 32,400;                  //display size
CONTENTS_SIZE = 800,80;                 //one content size
```

サイズは 割合指定 が可能です。

CONTENTS_SIZE = 横幅, 縦幅

一行のサイズを決定します。垂直方向に並んでいるリストボックスにおいては、縦方向にスクロールします。

ただし、コンテンツサイズの横幅のほうが表示エリアよりも大きいときは、左右にもスクロールします。

```
// only vertical direction to scroll
SIZE = 400,400;                 //display size
CONTENTS_SIZE = {100},40;       //virtual screen size

// Scroll to the left and right direction
SIZE = 400,400;                 //display size
CONTENTS_SIZE = {200},40;       //virtual screen size
```

コンテンツサイズは 割合指定 が可能です。

CONTENTS = { コントロール定義 … }

LOG の中の一行を定義するためのコンテンツを列挙します。

コントロールの座標は、LOG の座標から相対で配置されます。

全てのコントロールを関連付けることが可能です。

注釈: **LOG** 内の **CONTENTS** には、必ず一つの **LOGTEXT** を入れておく必要があります。

```
CONTENTS = {
  ICON(Chat) {
        :
  }
  TEXT(Name) {
        :
  }
```

(次のページに続く)

(前のページからの続き)

```
FRAME(Balloon) {
      :
}
LOGTEXT(Chat) {
      :
}
```

C#からコンテンツリストの特定のコントロールに対してアクセスする方法 は こちら を参考にしてください。

ただし、内部のコンテンツは循環リストになっているため、直接アクセスするのは危険です。コンテンツに対するカスタマイズは、 ログを追加 したときに行ってください。

注釈: コンテンツ内のコントロールの座標やサイズの 割合指定 は、リストボックスのサイズを基準に計算されます。

ウィンドウサイズではないことに気を付けてください。

LINE_SPACE = 行間ピクセル値

行間をこのプロパティで設定可能です。

割合指定はできません。

```
LINE_SPACE = 8;        //Put an 8-dot space.
```

GROUP = SCROLLBAR コントロール ID,…

SCROLLBAR コントロールを GROUP として定義しておくと、自動的にログと連動してスクロールバーを表示します。

CONTENTS に入れないように気を付けてください。

複数のスクロールバーを関連付けることも可能です。

```
GROUP = SCROLLBAR(Horizon),SCROLLBAR(Vertical);
```

COLOR = R,G,B,A

カラーを指定します。カラー変更することによって、含まれるコントロール全てに影響を与えます。

R,G,B については、0～2 の間で指定してください。

1 を超えたとき、そのカラー成分を 2 倍まで上げて表示することができます。

A については、0〜1 の間で指定してください。

STYLE = フラグ 0|フラグ 1|..|フラグ n

リストボックス固有のスタイルは以下のものがあります。

リストボックス制御フラグ	説明
NOBOUNCES	端までスクロールしたとき、勢いが残っていても跳ね返らない
SCROLL_UNLOCK	リストボックスのスクロールを許可する。
SCROLL_LOCK	リストボックスのスクロールを抑制する。

コントロールの表示アンカーを指定できます。

表示位置アンカーフラグ	説明
ANCHOR_DEFAULT	アンカー位置を左上に設定 ANCHOR_LEFTTOP と同じ コントロールの中心位置はデフォルトで、BASE_LEFTTOP になる。
ANCHOR_LEFTTOP	アンカー位置を左上に設定 コントロールの中心位置はデフォルトで、BASE_LEFTTOP になる。
ANCHOR_LEFT	アンカー位置を左に設定 縦にはセンタリング コントロールの中心位置はデフォルトで、BASE_LEFT になる。
ANCHOR_LEFTBOTTOM	アンカー位置を左に設定 縦には下辺を基準に配置 コントロールの中心位置はデフォルトで、BASE_LEFTBOTTOM になる。
ANCHOR_TOP	アンカー位置を上辺に設定 横方向にはセンタリング コントロールの中心位置はデフォルトで、BASE_TOP になる。
ANCHOR_CENTER	アンカー位置を画面中央にセンタリング コントロールの中心位置はデフォルトで、BASE_CENTER になる。
ANCHOR_BOTTOM	アンカー位置を底辺に設定 横方向にはセンタリング コントロールの中心位置はデフォルトで、BASE_BOTTOM になる。
ANCHOR_RIGHTTOP	アンカー位置を右上に設定 コントロールの中心位置はデフォルトで、BASE_RIGHTBOTTOM になる。
ANCHOR_RIGHT	アンカー位置を右に設定 縦にはセンタリング コントロールの中心位置はデフォルトで、BASE_RIGHT になる。
ANCHOR_RIGHTBOTTOM	アンカー位置を右に設定 縦には下辺を基準に配置 コントロールの中心位置はデフォルトで、BASE_RIGHTBOTTOM になる。

コントロールの中心位置を指定できます。

中心位置変更フラグ	説明
BASE_DEFAULT	アンカーフラグに応じて変わる。 特に指定しなければ、これになります。 各アンカーフラグの説明を参照
BASE_LEFTTOP	中心位置をコントロールの左上に設定
BASE_LEFT	中心位置を左に設定 縦にはコントロールの真中
BASE_LEFTBOTTOM	中心位置を左下に設定
BASE_TOP	中心位置を上辺に設定 横にはコントロールの中心
BASE_CENTER	中心位置をコントロールの中心
BASE_BOTTOM	中心位置を底辺に設定 横方向にはコントロールの真中
BASE_RIGHTTOP	中心位置をコントロールの右上に設定
BASE_RIGHT	中心位置を右に設定 横にはコントロールの中心
BASE_RIGHTBOTTOM	中心位置を右下に設定

機能を制限するスタイルは以下のものがあります。

機能制限スタイル	説明
HIDE	表示を隠す。
DISABLE	押せなくし、暗くする。
NOHIT	押せなくする。
NOBOUNCES	端までスクロールしたとき、勢いが残っていても跳ね返らない
SCROLL_UNLOCK	スクロールを許可する。
SCROLL_LOCK	スクロールを抑制する。

11.8 LOGTEXT

C#: *CWinCtrlLogText*

チャットログの一文を保持するためのコントロールです。

ほぼ、振る舞いは *TEXT* と同じです。

デフォルトでは、押したことを検出できません。コールバックが必要なときは、STYLE に HIT を指定してください。

注釈: *LOG* 内の *CONTENTS* には、必ず一つの *LOGTEXT* を入れておく必要があります。

LOGTEXT(コントロール名) {

プロパティ 1;

プロパティ 2;

:

:

プロパティ n;

};

11.8.1 記述例

```
LOG(Chat) {
  ID = 000_000_00030;
  STYLE = ANCHOR_LEFTBOTTOM;
  POSITION = 0,OFFSET_Y;
  SIZE = WINDOW_SIZE_FULL, 80-2;
  CONTENTS_SIZE = WINDOW_SIZE_FULL, 80-2;
  CONTENTS = {
    LOGTEXT(Chat) {
      ID = 000_001_00100;
      STYLE = NOHIT;
      POSITION = 105, -44;
      CAPTION_COLOR = 0,0,0,1;
      FONT_KIND = "fn24";
      CAPTION = 000_000_00010;
      CONTENTS_SIZE = WINDOW_SIZE_FULL -235;
      LINE_SPACE = 2;
    };
  };
  COLOR = COLOR32(0,0,0,255);
  LINE_SPACE = 10;
  GROUP = SCROLLBAR(Chat);
};
SCROLLBAR(Chat) {
  ID = 000_001_00001;
  STYLE = ANCHOR_RIGHTTOP;
  COLOR = 1,1,1,0.5;
  POSITION = -2,-5;
```

(次のページに続く)

(前のページからの続き)

```
  SIZE = 0,64-10;
};
```

11.8.2 プロパティ

代表的なデフォルト値

```
COLOR = 1,1,1,1;
CAPTION_COLOR = 1,1,1,1;
```

ID = コントロール ID

コントロール ID を定義します。

```
ID = 001_000_00010;
```

注釈: 設定しなかったときは、自動的にハッシュ値から生成します。

POSITION = X, Y

表示位置を決定します。STYLE に応じて、表示位置の基準を柔軟に変更可能です。

注釈: 表示位置アンカーフラグと中心位置変更フラグによって、位置が決定します。

テキストアンカーを設定しても表示に影響与えません

```
POSITION = 32,64;
```

座標は 割合指定 が可能です。

FONT_KIND = フォント種類

フォントの種類を指定します。

```
FONT_KIND = "fnt32";
```

CAPTION = キャプション ID

文字列の ID を指定してください。

こちら を参照ください。

```
CAPTION = 010_000_00100;
```

CAPTION = "文字列"

文字列を設定します。

```
CAPTION = "Hellow world!";
```

CAPTION_COLOR = R,G,B,A

キャプションのカラーを指定できます。

0〜1 の間で指定してください。

CAPTION_OFFSET = X, Y

キャプションの位置を移動することができます。

キャプションオフセットは 割合指定 が可能です。

CONTENTS_SIZE = 改行サイズ

文字列が改行サイズを超えたとき、そこから自動で改行をします。

```
CONTENTS_SIZE = 128;  //128pixel

CONTENTS_SIZE = {50} + 32;    // screen width * 0.5 + 32
```

改行サイズは 割合指定 が可能です。

LINE_SPACE = 行間ピクセル値

行間をこのプロパティで設定可能です。

```
LINE_SPACE = 8;       //Put an 8-dot space.
```

行間は割合指定はできません。

STYLE = フラグ 0|フラグ 1|..|フラグ n

コントロールの表示アンカーを指定できます。

表示位置アンカーフラグ	説明
ANCHOR_DEFAULT	アンカー位置を左上に設定 ANCHOR_LEFTTOP と同じ コントロールの中心位置はデフォルトで、BASE_LEFTTOP になる。
ANCHOR_LEFTTOP	アンカー位置を左上に設定 コントロールの中心位置はデフォルトで、BASE_LEFTTOP になる。
ANCHOR_LEFT	アンカー位置を左に設定 縦にはセンタリング コントロールの中心位置はデフォルトで、BASE_LEFT になる。
ANCHOR_LEFTBOTTOM	アンカー位置を左に設定 縦には下辺を基準に配置 コントロールの中心位置はデフォルトで、BASE_LEFTBOTTOM になる。
ANCHOR_TOP	アンカー位置を上辺に設定 横方向にはセンタリング コントロールの中心位置はデフォルトで、BASE_TOP になる。
ANCHOR_CENTER	アンカー位置を画面中央にセンタリング コントロールの中心位置はデフォルトで、BASE_CENTER になる。
ANCHOR_BOTTOM	アンカー位置を底辺に設定 横方向にはセンタリング コントロールの中心位置はデフォルトで、BASE_BOTTOM になる。
ANCHOR_RIGHTTOP	アンカー位置を右上に設定 コントロールの中心位置はデフォルトで、BASE_RIGHTBOTTOM になる。
ANCHOR_RIGHT	アンカー位置を右に設定 縦にはセンタリング コントロールの中心位置はデフォルトで、BASE_RIGHT になる。
ANCHOR_RIGHTBOTTOM	アンカー位置を右に設定 縦には下辺を基準に配置 コントロールの中心位置はデフォルトで、BASE_RIGHTBOTTOM になる。

コントロールの中心位置を指定できます。

中心位置変更フラグ	説明
BASE_DEFAULT	アンカーフラグに応じて変わる。 特に指定しなければ、これになります。 各アンカーフラグの説明を参照
BASE_LEFTTOP	中心位置をコントロールの左上に設定
BASE_LEFT	中心位置を左に設定 縦にはコントロールの真中
BASE_LEFTBOTTOM	中心位置を左下に設定
BASE_TOP	中心位置を上辺に設定 横にはコントロールの中心
BASE_CENTER	中心位置をコントロールの中心
BASE_BOTTOM	中心位置を底辺に設定 横方向にはコントロールの真中
BASE_RIGHTTOP	中心位置をコントロールの右上に設定
BASE_RIGHT	中心位置を右に設定 横にはコントロールの中心
BASE_RIGHTBOTTOM	中心位置を右下に設定

キャプションのフォント装飾を変更できます。

キャプション装飾種類	説明
TEXT_NORMAL	装飾なし
TEXT_BOLD	太字
TEXT_DENT	へこんだ感じの文字
TEXT_SHADOW	影付き文字

機能を制限するスタイルは以下のものがあります。

機能制限スタイル	説明
HIDE	表示を隠す。
DRAG	ドラッグ可能にする。
DISABLE	押せなくし、暗くする。
HIT	押せるようにする。

11.9 EDITBOX

C#: *CWinCtrlEditbox*

編集可能なエディットボックスの為のコントロールです。

行数指定及び、文字列の最大サイズを指定可能です。

このコントロールはスクロール可能なコントロールです。

STYLE に *SCROLL_LOCK* をつけておくと、onDrag コールバックが呼び出され、ドラッグした方向を検出できます。DRAG をつけても、これらコントロールには効果がないことに気を付けてください。

EDITBOX(コントロール名) {

プロパティ 1;

プロパティ 2;

:

:

プロパティ n;

};

11.9.1 記述例

一行入力のエディットボックスの例

```
EDITBOX(LoginID) {
  ID = 001_000_00010;
  STYLE = ANCHOR_BOTTOM;
  SIZE = 300;
  EDIT = 255,0;        //255 characters,one line
  POSITION = 0,$y;
  FONT_KIND = "fn24";
};
```

複数行 (この例では 10 行) 入力のエディットボックスの例.

```
EDITBOX(Sentence) {
  ID = 000_000_00010;
  STYLE = ANCHOR_LEFT;
  CAPTION_COLOR = COLOR32(50,50,50,255);
  FONT_KIND="fn16";
  EDIT = 255,10;              //255 characters,10 lines
  TEX_ID = 0,"EDBT0";
  POSITION = $x,$y;
  SIZE = $w,$h;
};
```

11.9.2 編集機能

Android,iOS 環境以外は、簡単な編集機能を持ちます。

Unity Editor の制限により、いくつかの機能はエディター上のキー入力とアプリケーション上のものとでは入力が異なります。

カーソル移動	アプリケーション上	Unity Editor
カーソルを右へ移動	右	右
カーソルを左へ移動	左	左
カーソルを上へ移動	上	上
カーソルを下へ移動	下	下
カーソルを先頭へ移動	Home	Home
カーソルを末尾へ移動	End	End

選択範囲	アプリケーション上	Unity Editor
選択範囲を右に広げる	Shift + 右	Shift + 右
選択範囲を左へ広げる	Shift + 左	Shift + 左
選択範囲を上へ広げる	Shift + 上	Shift + 上
選択範囲を下へ広げる	Shift + 下	Shift + 下
選択範囲を先頭へ広げる	Shift + Home	Shift + Home
選択範囲を末尾へ広げる	Shift + End	Shift + End

編集機能	アプリケーション上	Unity Editor
バックスペース	BackSpace	BackSpace
デリート	Delete	Delete
カット	Ctrl + X	Ctrl + **Shift** + X
ペースト	Ctrl + V	Ctrl + **Shift** + V
コピー	Ctrl + C	Ctrl + **Shift** + **Z**

11.9.3 プロパティ

代表的なデフォルト値

```
TEX_ID = "TXFD?";
TEX_ID1 = "CURSR"; //Cursor parts id
COLOR = 1,1,1,1;
COLOR1 = 1,1,1,1;  //Cursor color
CAPTION_COLOR = 1,1,1,1;
```

パーツ ID ルール

パーツ ID の 5 文字目を自動的に 0/1 に入れ替えます。

- XXXX0: OFF の状態
- XXXX1: ON の状態

```
TEX_ID = "TXFD?";
```

パーツ ID がこのように設定されているとすると以下のようなパーツを用意しておく必要があります。

- TXFD0:フォーカス OFF の状態
- TXFD1:フォーカス ON の状態

ID = コントロール ID

コントロール ID を定義します。

```
ID = 001_000_00010;
```

注釈: 設定しなかったときは、自動的にハッシュ値から生成します。

EDIT = 入力文字最大数, 入力可能行数

入力文字最大数と、入力可能行数を設定します。

入力可能行数が、0 または 1 の場合、一行入力エディットボックスになります。

POSITION = X, Y

表示位置を決定します。STYLE の アンカー 指定に応じて、基準位置が変わります。

```
POSITION = 32,{50} + 64;
```

座標は 割合指定 が可能です。

SIZE = 横サイズ, 縦サイズ

表示サイズを変更します。省略したとき、若しくは、0 を指定すると、テクスチャパーツからサイズを取得してきます。

```
SIZE = 64,32; //64x32
SIZE = ,32;   //Set the width of the texture part width
SIZE = 64;            //Set the height of the texture part height
SIZE = {50} - 25;
```

サイズは 割合指定 が可能です。

TEX_ID = テクスチャ ID, パーツ ID

TEX_ID = パーツ ID

ベースのテクスチャ ID とパーツ ID を指定します。

TEX_ID1 = テクスチャ ID, パーツ ID

TEX_ID1 = パーツ ID

カーソルに使うテクスチャ ID とパーツ ID を指定します。

COLOR = R,G,B,A

カラーを指定します。

R,G,B については、0〜2 の間で指定してください。

1 を超えたとき、そのカラー成分を 2 倍まで上げて表示することができます。

A については、0〜1 の間で指定してください。

LINE_SPACE = 行間ピクセル値

コンテンツとコンテンツ間の隙間をこのプロパティで設定可能です。

```
LINE_SPACE = 8;       //Put an 8-dot space.
```

割合指定はできません。

GROUP = SCROLLBAR コントロール ID,…

SCROLLBAR コントロールを GROUP として定義しておくと、自動的にエディットボックスと連動してスクロールバーを表示します。

CONTENTS に入れないように気を付けてください。

複数のスクロールバーを関連付けることも可能です。

```
GROUP = SCROLLBAR(Horizon),SCROLLBAR(Vertical);
```

CAPTION = キャプション ID

文字列の ID を指定してください。

こちら を参照ください。

CAPTION = "文字列"

文字列を設定します。

```
CAPTION = "Hellow world!";
```

CAPTION_COLOR = R,G,B,A

キャプションのカラーを指定できます。

EDITBOX では、入力文字列のカラーになります。

0〜1 の間で指定してください。

CAPTION_OFFSET = X, Y

キャプションの位置を移動することができます。

キャプションオフセットは 割合指定 が可能です。

STYLE = スタイル 0|スタイル 1|..|スタイル n

エディットボックスのスタイルを指定できます。

エディットボックススタイル	説明
EDIT_BLIND	入力文字をマスクする
EDIT_ALL	デフォルト、全ての文字入力が可能
EDIT_TYPE_ASCIICAPABLE	ASCII 配列のキーボード
EDIT_TYPE_URL	URL 入力を行うキーボード
EDIT_TYPE_NUMBERANDPUNCTUATION	数字と句読点が含まれるキーボード
EDIT_TYPE_NUMBERPAD	PIN の入力用にデザインされたテンキーキーボード
EDIT_TYPE_PHONEPAD	電話番号入力を行うキーボード
EDIT_TYPE_NAMEPHONEPAD	番号入力と文字入力を行うキーボード
EDIT_TYPE_EMAILADDRESS	メールアドレスを入力するキーボード

コントロールの表示アンカーを指定できます。

表示位置アンカーフラグ	説明
ANCHOR_DEFAULT	アンカー位置を左上に設定 ANCHOR_LEFTTOP と同じ コントロールの中心位置はデフォルトで、BASE_LEFTTOP になる。
ANCHOR_LEFTTOP	アンカー位置を左上に設定 コントロールの中心位置はデフォルトで、BASE_LEFTTOP になる。
ANCHOR_LEFT	アンカー位置を左に設定 縦にはセンタリング コントロールの中心位置はデフォルトで、BASE_LEFT になる。
ANCHOR_LEFTBOTTOM	アンカー位置を左に設定 縦には下辺を基準に配置 コントロールの中心位置はデフォルトで、BASE_LEFTBOTTOM になる。
ANCHOR_TOP	アンカー位置を上辺に設定 横方向にはセンタリング コントロールの中心位置はデフォルトで、BASE_TOP になる。
ANCHOR_CENTER	アンカー位置を画面中央にセンタリング コントロールの中心位置はデフォルトで、BASE_CENTER になる。
ANCHOR_BOTTOM	アンカー位置を底辺に設定 横方向にはセンタリング コントロールの中心位置はデフォルトで、BASE_BOTTOM になる。
ANCHOR_RIGHTTOP	アンカー位置を右上に設定 コントロールの中心位置はデフォルトで、BASE_RIGHTBOTTOM になる。
ANCHOR_RIGHT	アンカー位置を右に設定 縦にはセンタリング コントロールの中心位置はデフォルトで、BASE_RIGHT になる。
ANCHOR_RIGHTBOTTOM	アンカー位置を右に設定 縦には下辺を基準に配置 コントロールの中心位置はデフォルトで、BASE_RIGHTBOTTOM になる。

コントロールの中心位置を指定できます。

中心位置変更フラグ	説明
BASE_DEFAULT	アンカーフラグに応じて変わる。 特に指定しなければ、これになります。 各アンカーフラグの説明を参照
BASE_LEFTTOP	中心位置をコントロールの左上に設定
BASE_LEFT	中心位置を左に設定 縦にはコントロールの真中
BASE_LEFTBOTTOM	中心位置を左下に設定
BASE_TOP	中心位置を上辺に設定 横にはコントロールの中心
BASE_CENTER	中心位置をコントロールの中心
BASE_BOTTOM	中心位置を底辺に設定 横方向にはコントロールの真中
BASE_RIGHTTOP	中心位置をコントロールの右上に設定
BASE_RIGHT	中心位置を右に設定 横にはコントロールの中心
BASE_RIGHTBOTTOM	中心位置を右下に設定

キャプションのフォント装飾を変更できます。

キャプション装飾種類	説明
TEXT_NORMAL	装飾なし
TEXT_BOLD	太字
TEXT_DENT	へこんだ感じの文字
TEXT_SHADOW	影付き文字

機能を制限するスタイルは以下のものがあります。

機能制限スタイル	説明
HIDE	表示を隠す。
DISABLE	押せなくし、暗くする。
NOHIT	押せなくする。
NOBOUNCES	端までスクロールしたとき、勢いが残っていても跳ね返らない
SCROLL_UNLOCK	スクロールを許可する。
SCROLL_LOCK	スクロールを抑制する。

11.10 TEXTBOX

C#: *CWinCtrlTextbox*

編集できないエディットボックスです。

キャプションが、コントロールサイズに収まらないときは、テキストボックスをタッチするとスクロールします。

このコントロールはスクロール可能なコントロールです。

STYLE に *SCROLL_LOCK* をつけておくと、onDrag コールバックが呼び出され、ドラッグした方向を検出できます。DRAG をつけても、これらコントロールには効果がないことに気を付けてください。

TEXTBOX(コントロール名) {

プロパティ 1;

プロパティ 2;

:

:

プロパティ n;

};

11.10.1 記述例

一行のテキストボックスの例 (TEXT とほぼ同じ振る舞いになります)

```
TEXTBOX(Sentence) {
  ID = 001_000_00010;
  STYLE = ANCHOR_BOTTOM;
  SIZE = 300;
  EDIT = 255,0;       //255 characters,one line
  POSITION = 0,$y;
  FONT_KIND = "fn24";
};
```

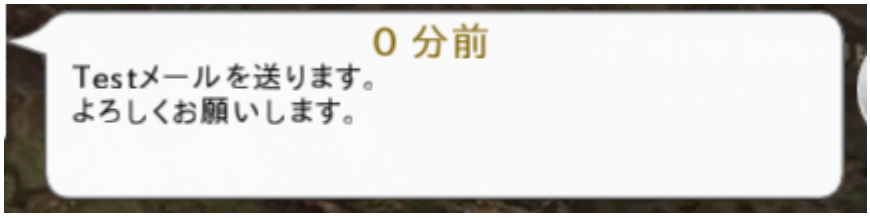
複数行 (この例では 10 行) のテキストボックスの例.

```
TEXTBOX(Sentence) {
  ID = 000_000_00010;
  STYLE = ANCHOR_LEFT;
  CAPTION_COLOR = COLOR32(50,50,50,255);
  FONT_KIND="fn16";
```

(次のページに続く)

(前のページからの続き)

```
  EDIT = 255,10;              //255 characters,10 lines
  TEX_ID = 0,"EDBT0";
  POSITION = $x,$y;
  SIZE = $w,$h;
};
```



11.10.2 プロパティ

```
TEX_ID = "TXFD?";
COLOR = 1,1,1,1;
CAPTION_COLOR = 1,1,1,1;
```

パーツ ID ルール

パーツ ID の 5 文字目を自動的に 0/1 に入れ替えます。

- XXXX0: OFF の状態
- XXXX1: ON の状態

```
TEX_ID = "TXFD?";
```

パーツ ID がこのように設定されているとすると以下のようなパーツを用意しておく必要があります。

- TXFD0:フォーカス OFF の状態
- TXFD1:フォーカス ON の状態

ID = コントロール ID

コントロール ID を定義します。

```
ID = 001_000_00010;
```

注釈: 設定しなかったときは、自動的にハッシュ値から生成します。

EDIT = キャプション文字最大数, キャプション最大行数

キャプション文字最大数と、キャプション最大行数を設定します。

POSITION = X, Y

表示位置を決定します。STYLE の アンカー 指定に応じて、基準位置が変わります。

```
POSITION = 32,{50} + 64;
```

座標は 割合指定 が可能です。

SIZE = 横サイズ, 縦サイズ

表示サイズを変更します。省略したとき、若しくは、0 を指定すると、テクスチャパーツからサイズを取得してきます。

```
SIZE = 64,32; //64x32
SIZE = ,32;   //Set the width of the texture part width
SIZE = 64;            //Set the height of the texture part height
SIZE = {50} - 25;
```

サイズは 割合指定 が可能です。

TEX_ID = テクスチャ ID, パーツ ID

TEX_ID = パーツ ID

テクスチャ ID とパーツ ID を指定します。

COLOR = R,G,B,A

カラーを指定します。

R,G,B については、0〜2 の間で指定してください。

1 を超えたとき、そのカラー成分を 2 倍まで上げて表示することができます。

A については、0〜1 の間で指定してください。

CAPTION = キャプション ID

文字列の ID を指定してください。

こちら を参照ください。

```
CAPTION = 010_000_00100;
```

CAPTION = "文字列"

文字列を設定します。

```
CAPTION = "Hellow world!";
```

CAPTION_COLOR = R,G,B,A

キャプションのカラーを指定できます。

EDITBOX では、入力文字列のカラーになります。

0〜1 の間で指定してください。

CAPTION_OFFSET = X, Y

キャプションの位置を移動することができます。

キャプションオフセットは 割合指定 が可能です。

LINE_SPACE = 行間ピクセル値

コンテンツとコンテンツ間の隙間をこのプロパティで設定可能です。

```
LINE_SPACE = 8;        //Put an 8-dot space.
```

割合指定はできません。

GROUP = SCROLLBAR コントロール ID,…

SCROLLBAR コントロールを GROUP として定義しておくと、自動的にテキストボックスと連動してスクロールバーを表示します。

CONTENTS に入れないように気を付けてください。

複数のスクロールバーを関連付けることも可能です。

```
GROUP = SCROLLBAR(Horizon),SCROLLBAR(Vertical);
```

STYLE = スタイル **0**|スタイル **1**|..|スタイル **n**

コントロールの表示アンカーを指定できます。

表示位置アンカーフラグ	説明
ANCHOR_DEFAULT	アンカー位置を左上に設定 ANCHOR_LEFTTOP と同じ コントロールの中心位置はデフォルトで、BASE_LEFTTOP になる。
ANCHOR_LEFTTOP	アンカー位置を左上に設定 コントロールの中心位置はデフォルトで、BASE_LEFTTOP になる。
ANCHOR_LEFT	アンカー位置を左に設定 縦にはセンタリング コントロールの中心位置はデフォルトで、BASE_LEFT になる。
ANCHOR_LEFTBOTTOM	アンカー位置を左に設定 縦には下辺を基準に配置 コントロールの中心位置はデフォルトで、BASE_LEFTBOTTOM になる。
ANCHOR_TOP	アンカー位置を上辺に設定 横方向にはセンタリング コントロールの中心位置はデフォルトで、BASE_TOP になる。
ANCHOR_CENTER	アンカー位置を画面中央にセンタリング コントロールの中心位置はデフォルトで、BASE_CENTER になる。
ANCHOR_BOTTOM	アンカー位置を底辺に設定 横方向にはセンタリング コントロールの中心位置はデフォルトで、BASE_BOTTOM になる。
ANCHOR_RIGHTTOP	アンカー位置を右上に設定 コントロールの中心位置はデフォルトで、BASE_RIGHTBOTTOM になる。
ANCHOR_RIGHT	アンカー位置を右に設定 縦にはセンタリング コントロールの中心位置はデフォルトで、BASE_RIGHT になる。
ANCHOR_RIGHTBOTTOM	アンカー位置を右に設定 縦には下辺を基準に配置 コントロールの中心位置はデフォルトで、BASE_RIGHTBOTTOM になる。

コントロールの中心位置を指定できます。

中心位置変更フラグ	説明
BASE_DEFAULT	アンカーフラグに応じて変わる。 特に指定しなければ、これになります。 各アンカーフラグの説明を参照
BASE_LEFTTOP	中心位置をコントロールの左上に設定
BASE_LEFT	中心位置を左に設定 縦にはコントロールの真中
BASE_LEFTBOTTOM	中心位置を左下に設定
BASE_TOP	中心位置を上辺に設定 横にはコントロールの中心
BASE_CENTER	中心位置をコントロールの中心
BASE_BOTTOM	中心位置を底辺に設定 横方向にはコントロールの真中
BASE_RIGHTTOP	中心位置をコントロールの右上に設定
BASE_RIGHT	中心位置を右に設定 横にはコントロールの中心
BASE_RIGHTBOTTOM	中心位置を右下に設定

キャプションのフォント装飾を変更できます。

キャプション装飾種類	説明
TEXT_NORMAL	装飾なし
TEXT_BOLD	太字
TEXT_DENT	へこんだ感じの文字
TEXT_SHADOW	影付き文字

機能を制限するスタイルは以下のものがあります。

機能制限スタイル	説明
HIDE	表示を隠す。
DISABLE	押せなくし、暗くする。
NOHIT	押せなくする。
NOBOUNCES	端までスクロールしたとき、勢いが残っていても跳ね返らない
SCROLL_UNLOCK	スクロールを許可する。
SCROLL_LOCK	スクロールを抑制する。

11.11 BUTTON

C#: *CWinCtrlButton*

ボタンを実現するためのコントロールです。

一つのキャプションと、複数のバッジを同時に表示することが可能です。

バッジは最大 7 つまで対応可能です。

BUTTON(コントロール名) {

プロパティ 1;

プロパティ 2;

:

:

プロパティ n;

};

11.11.1 記述例

```
WINDOW(255_000_00001) {
  STYLE = NOTITLEBAR|ANCHOR_CENTER;
  POSITION = 0,100;
  TEX_ID = 100_000_00001;
  CAPTION = 000_000_00010;
  SIZE = 512,256;
};

BUTTON(Button) {
  ID = 000_002_00001;
  CAPTION = 000_000_00006;
  STYLE = ANCHOR_CENTER;
  POSITION = 0,0;
  FONT_KIND= "fn40";
  TEX_ID1 = "bd01";
  TEXTURE_OFFSET1 = 0,10;
  SIZE = 256;
};
```

11.11.2 プロパティ

代表的なデフォルト値

```
TEX_ID = "BTN0?";
COLOR = 1,1,1,1;
CAPTION_COLOR = 1,1,1,1;
SE_ID = clickSE;
```

パーツ ID ルール

パーツ ID の 5 文字目を自動的に 0/1 に入れ替えます。

- XXXX0: OFF の状態
- XXXX1: ON の状態

```
TEX_ID = "BTN0?";
```

パーツ ID がこのように設定されているとすると以下のようなパーツを用意しておく必要があります。

- BTN00:OFF の状態
- BTN01:ON の状態

ID = コントロール ID

コントロール ID を定義します。

```
ID = 001_000_00010;
```

注釈: 設定しなかったときは、自動的にハッシュ値から生成します。

POSITION = X, Y

表示位置を決定します。STYLE の アンカー 指定に応じて、基準位置が変わります。

```
POSITION = 32,{50} + 64;
```

座標は 割合指定 が可能です。

SIZE = 横サイズ, 縦サイズ

表示サイズを変更します。省略したとき、若しくは、0 を指定すると、テクスチャパーツからサイズを取得してきます。

```
SIZE = 64,32; //64x32
SIZE = ,32;   //Set the width of the texture part width
SIZE = 64;            //Set the height of the texture part height
SIZE = {50} - 25;
```

サイズは 割合指定 が可能です。

TEX_ID = テクスチャ ID, パーツ ID

TEX_ID = パーツ ID

テクスチャ ID とパーツ ID を指定します。

COLOR = R,G,B,A

カラーを指定します。

R,G,B については、0〜2 の間で指定してください。

1 を超えたとき、そのカラー成分を 2 倍まで上げて表示することができます。

A については、0〜1 の間で指定してください。

CAPTION = キャプション ID

文字列の ID を指定してください。

こちら を参照ください。

CAPTION = "文字列"

文字列を設定します。

```
CAPTION = "Hellow world!";
```

CAPTION_COLOR = R,G,B,A

キャプションのカラーを指定できます。

0〜1 の間で指定してください。

CAPTION_OFFSET = X, Y

キャプションの位置を移動することができます。

キャプションオフセットは 割合指定 が可能です。

TEX_ID n = テクスチャ ID, パーツ ID

TEX_ID n = パーツ ID

バッジのテクスチャ ID とパーツ ID を指定します。

n = [1..7] で指定可能です。

TEXTURE_OFFSET n = オフセット X, オフセット Y

テクスチャの表示オフセットを指定します。

n = [0..7] で指定可能です。

テクスチャオフセットは 割合指定 が可能です。

TEXTURE_SIZE n = 横サイズ, 縦サイズ

テクスチャ ID とパーツ ID を指定します。

n = [0..7] で指定可能です。

n = 0 の時は、 **SIZE** と同じ意味です。

```
TEXTURE_SIZE1 = 64,32;         //64x32
TEXTURE_SIZE2 = ,32;  //Set the width of the texture part width
TEXTURE_SIZE3 = 64;   //Set the height of the texture part height
```

テクスチャサイズは 割合指定 が可能です。

COLOR n = R,G,B,A

バッジのカラーを指定します。

R,G,B については、0〜2 の間で指定してください。

1 を超えたとき、そのカラー成分が 2 倍までかけることができます。

A については、0〜1 の間で指定してくさい。

n = [1..7] で指定可能です。

SE_ID = SE_ID

押されたときに鳴らす音の ID を設定します。デフォルトは、clickSE です。0 を指定すると音が鳴らなくなります。

STYLE = フラグ 0|フラグ 1|..|フラグ n

コントロールの表示アンカーを指定できます。

表示位置アンカーフラグ	説明
ANCHOR_DEFAULT	アンカー位置を左上に設定 ANCHOR_LEFTTOP と同じ コントロールの中心位置はデフォルトで、BASE_LEFTTOP になる。
ANCHOR_LEFTTOP	アンカー位置を左上に設定 コントロールの中心位置はデフォルトで、BASE_LEFTTOP になる。
ANCHOR_LEFT	アンカー位置を左に設定 縦にはセンタリング コントロールの中心位置はデフォルトで、BASE_LEFT になる。
ANCHOR_LEFTBOTTOM	アンカー位置を左に設定 縦には下辺を基準に配置 コントロールの中心位置はデフォルトで、BASE_LEFTBOTTOM になる。
ANCHOR_TOP	アンカー位置を上辺に設定 横方向にはセンタリング コントロールの中心位置はデフォルトで、BASE_TOP になる。
ANCHOR_CENTER	アンカー位置を画面中央にセンタリング コントロールの中心位置はデフォルトで、BASE_CENTER になる。
ANCHOR_BOTTOM	アンカー位置を底辺に設定 横方向にはセンタリング コントロールの中心位置はデフォルトで、BASE_BOTTOM になる。
ANCHOR_RIGHTTOP	アンカー位置を右上に設定 コントロールの中心位置はデフォルトで、BASE_RIGHTBOTTOM になる。
ANCHOR_RIGHT	アンカー位置を右に設定 縦にはセンタリング コントロールの中心位置はデフォルトで、BASE_RIGHT になる。
ANCHOR_RIGHTBOTTOM	アンカー位置を右に設定 縦には下辺を基準に配置 コントロールの中心位置はデフォルトで、BASE_RIGHTBOTTOM になる。

コントロールの中心位置を指定できます。

中心位置変更フラグ	説明
BASE_DEFAULT	アンカーフラグに応じて変わる。 特に指定しなければ、これになります。 各アンカーフラグの説明を参照
BASE_LEFTTOP	中心位置をコントロールの左上に設定
BASE_LEFT	中心位置を左に設定 縦にはコントロールの真中
BASE_LEFTBOTTOM	中心位置を左下に設定
BASE_TOP	中心位置を上辺に設定 横にはコントロールの中心
BASE_CENTER	中心位置をコントロールの中心
BASE_BOTTOM	中心位置を底辺に設定 横方向にはコントロールの真中
BASE_RIGHTTOP	中心位置をコントロールの右上に設定
BASE_RIGHT	中心位置を右に設定 横にはコントロールの中心
BASE_RIGHTBOTTOM	中心位置を右下に設定

キャプションの中心位置を指定できます。

テキストアンカー	説明
TEXT_CENTER	テキスト表示を中心に合わせます。
TEXT_LEFT	テキスト表示を左に合わせます。
TEXT_RIGHT	テキスト表示を右に合わせます。

キャプションのフォント装飾を変更できます。

キャプション装飾種類	説明
TEXT_NORMAL	装飾なし
TEXT_BOLD	太字
TEXT_DENT	へこんだ感じの文字
TEXT_SHADOW	影付き文字

機能を制限するスタイルは以下のものがあります。

機能制限スタイル	説明
HIDE	表示を隠す。
DRAG	ドラッグ可能にする。
DISABLE	押せなくし、暗くする。
NOHIT	押せなくする。

11.12 RADIO

C#: *CWinCtrlRadio*

ラジオボタンとは、複数のラジオボタン同士をグループ化し、一つが ON になるとグループ化されたラジオボタンが自動的に OFF になるボタンです。

CHECKBOX 同様、*CONTENTS* を持つことができ、タブボタンとして振る舞うことも可能です。

RADIO(コントロール名) {

プロパティ 1;

プロパティ 2;

:

:

プロパティ n;

};

11.12.1 記述例

```
WINDOW(255_000_00001) {
  STYLE = NOTITLEBAR|ANCHOR_CENTER;
  POSITION = 0,100;
  TEX_ID = 100_000_00001;
  CAPTION = 000_000_00010;
  SIZE = 512,256;
};
RADIO(Party) {
  ID = 000_000_00100;
  POSITION = 16,10;
  TEX_ID=0,"BTNT?";
  CAPTION_COLOR=COLOR32(30,30,30,220);
  FONT_KIND="fn22";
```

(次のページに続く)

(前のページからの続き)

```
  SIZE=44*4+16,64;
  CAPTION = 000_000_00015;
  GROUP = RADIO(Party),RADIO(Friend),RADIO(Guild);
};
RADIO(Friend) {
  ID = 000_000_00110;
  POSITION = 16,90;
  TEX_ID=0,"BTNT?";
  CAPTION_COLOR=COLOR32(30,30,30,220);
  FONT_KIND="fn22";
  SIZE=44*4+16,64;
  CAPTION = 010_000_00020;
  GROUP = RADIO(Party),RADIO(Friend),RADIO(Guild);
};
RADIO(Guild) {
  ID = 000_000_00120;
  POSITION = 16,170;
  TEX_ID=0,"BTNT?";
  CAPTION_COLOR=COLOR32(30,30,30,220);
  FONT_KIND="fn22";
  SIZE=44*4+16,64;
  CAPTION = 010_000_00010;
  GROUP = RADIO(Party),RADIO(Friend),RADIO(Guild);
};
```

この例では、パーティボタン、フレンドボタン、ギルドボタンがそれぞれ排他的に ON になります。

11.12.2 プロパティ

代表的なデフォルト値

```
TEX_ID = "BTN0?";
COLOR = 1,1,1,1;
CAPTION_COLOR = 1,1,1,1;
```

パーツ ID ルール

パーツ ID の 5 文字目を自動的に 0/1 に入れ替えます。

- XXXX0: OFF の状態
- XXXX1: ON の状態

```
TEX_ID = "BTN0?";
```

パーツ ID がこのように設定されているとすると以下のようなパーツを用意しておく必要があります。

- BTN00:OFF の状態
- BTN01:ON の状態

ID = コントロール ID

コントロール ID を定義します。

```
ID = 001_000_00010;
```

注釈: 設定しなかったときは、自動的にハッシュ値から生成します。

POSITION = X, Y

表示位置を決定します。STYLE の アンカー 指定に応じて、基準位置が変わります。

```
POSITION = 32,{50} + 64;
```

座標は 割合指定 が可能です。

SIZE = 横サイズ, 縦サイズ

表示サイズを変更します。省略したとき、若しくは、0 を指定すると、テクスチャパーツからサイズを取得してきます。

```
SIZE = 64,32; //64x32
SIZE = ,32;   //Set the width of the texture part width
SIZE = 64;            //Set the height of the texture part height
SIZE = {50} - 25;
```

サイズは 割合指定 が可能です。

TEX_ID = テクスチャ ID, パーツ ID

TEX_ID = パーツ ID

テクスチャ ID とパーツ ID を指定します。

COLOR = R,G,B,A

カラーを指定します。

R,G,B については、0〜2 の間で指定してください。

1 を超えたとき、そのカラー成分を 2 倍まで上げて表示することができます。

A については、0〜1 の間で指定してください。

SE_ID = SE_ID

押されたときに鳴らす音の ID を設定します。デフォルトは、clickSE です。0 を指定すると音が鳴らなくなります。

CAPTION = キャプション ID

文字列の ID を指定してください。

こちら を参照ください。

```
CAPTION = 010_000_00100;
```

CAPTION = "文字列"

文字列を設定します。

```
CAPTION = "Hellow world!";
```

CAPTION_COLOR = R,G,B,A

キャプションのカラーを指定できます。

0〜1 の間で指定してください。

CAPTION_OFFSET = X, Y

キャプションの位置を移動することができます。

キャプションオフセットは 割合指定 が可能です。

TEX_ID n = テクスチャ ID, パーツ ID

TEX_ID n = パーツ ID

バッジのテクスチャ ID とパーツ ID を指定します。

n = [1..7] で指定可能です。

TEXTURE_OFFSET n = オフセット X, オフセット Y

テクスチャの表示オフセットを指定します。

n = [0..7] で指定可能です。

テクスチャオフセットは 割合指定 が可能です。

TEXTURE_SIZE n = 横サイズ, 縦サイズ

テクスチャ ID とパーツ ID を指定します。

n = [0..7] で指定可能です。

n = 0 の時は、 **SIZE** と同じ意味です。

```
TEXTURE_SIZE1 = 64,32;         //64x32
TEXTURE_SIZE2 = ,32;  //Set the width of the texture part width
TEXTURE_SIZE3 = 64;   //Set the height of the texture part height
```

テクスチャサイズは 割合指定 が可能です。

COLOR n = R,G,B,A

バッジのカラーを指定します。

R,G,B については、0〜2 の間で指定してください。

1 を超えたとき、そのカラー成分を 2 倍まで上げて表示することができます。

A については、0〜1 の間で指定してください。

n = [1..7] で指定可能です。

CONTENTS = { コントロール定義 … }

ON 状態の時に、自動的にアクティブになるコントロールを関連付けます。

OFF 状態の時には、自動的に HIDE 状態になります。全てのコントロールを関連付けることが可能です。

```
CONTENTS = {
   BUTTON(A) {
      :
   }
   TEXTURE(B) {
      :
   }
   LISTBOX(C) {
      :
   }
}
```

C#からコンテンツに対してアクセスする方法は こちら を参考にしてください。

注釈: コンテンツ内のコントロールの座標やサイズの 割合指定 には、リストボックスと異なり影響を与えません。

GROUP = ID0,ID1,…,IDn

ラジオボタン同士を関連付けておくと、グループの中の一つが ON になった時、他のラジオボタンを自動的に OFF 状態に変更してくれます。

```
GRUOP = RADIO(A),RADIO(B),RADIO(C);
```

RADIO(A) が ON になると、自動的に RADIO(B),RADIO(C) は、OFF になります。

STYLE = スタイル 0|スタイル 1|..|スタイル n

コントロールの表示アンカーを指定できます。

表示位置アンカーフラグ	説明
ANCHOR_DEFAULT	アンカー位置を左上に設定 ANCHOR_LEFTTOP と同じ コントロールの中心位置はデフォルトで、BASE_LEFTTOP になる。
ANCHOR_LEFTTOP	アンカー位置を左上に設定 コントロールの中心位置はデフォルトで、BASE_LEFTTOP になる。
ANCHOR_LEFT	アンカー位置を左に設定 縦にはセンタリング コントロールの中心位置はデフォルトで、BASE_LEFT になる。
ANCHOR_LEFTBOTTOM	アンカー位置を左に設定 縦には下辺を基準に配置 コントロールの中心位置はデフォルトで、BASE_LEFTBOTTOM になる。
ANCHOR_TOP	アンカー位置を上辺に設定 横方向にはセンタリング コントロールの中心位置はデフォルトで、BASE_TOP になる。
ANCHOR_CENTER	アンカー位置を画面中央にセンタリング コントロールの中心位置はデフォルトで、BASE_CENTER になる。
ANCHOR_BOTTOM	アンカー位置を底辺に設定 横方向にはセンタリング コントロールの中心位置はデフォルトで、BASE_BOTTOM になる。
ANCHOR_RIGHTTOP	アンカー位置を右上に設定 コントロールの中心位置はデフォルトで、BASE_RIGHTBOTTOM になる。
ANCHOR_RIGHT	アンカー位置を右に設定 縦にはセンタリング コントロールの中心位置はデフォルトで、BASE_RIGHT になる。
ANCHOR_RIGHTBOTTOM	アンカー位置を右に設定 縦には下辺を基準に配置 コントロールの中心位置はデフォルトで、BASE_RIGHTBOTTOM になる。

コントロールの中心位置を指定できます。

中心位置変更フラグ	説明
BASE_DEFAULT	アンカーフラグに応じて変わる。 特に指定しなければ、これになります。 各アンカーフラグの説明を参照
BASE_LEFTTOP	中心位置をコントロールの左上に設定
BASE_LEFT	中心位置を左に設定 縦にはコントロールの真中
BASE_LEFTBOTTOM	中心位置を左下に設定
BASE_TOP	中心位置を上辺に設定 横にはコントロールの中心
BASE_CENTER	中心位置をコントロールの中心
BASE_BOTTOM	中心位置を底辺に設定 横方向にはコントロールの真中
BASE_RIGHTTOP	中心位置をコントロールの右上に設定
BASE_RIGHT	中心位置を右に設定 横にはコントロールの中心
BASE_RIGHTBOTTOM	中心位置を右下に設定

キャプションの中心位置を指定できます。

テキストアンカー	説明
TEXT_CENTER	テキスト表示を中心に合わせます。
TEXT_LEFT	テキスト表示を左に合わせます。
TEXT_RIGHT	テキスト表示を右に合わせます。

キャプションのフォント装飾を変更できます。

キャプション装飾種類	説明
TEXT_NORMAL	装飾なし
TEXT_BOLD	太字
TEXT_DENT	へこんだ感じの文字
TEXT_SHADOW	影付き文字

機能を制限するスタイルは以下のものがあります。

機能制限スタイル	説明
HIDE	表示を隠す。
DRAG	ドラッグ可能にする。
DISABLE	押せなくし、暗くする。
NOHIT	押せなくする。

11.13 CHECKBOX

C#: *CWinCtrlCheckbox*

チェックボックスボタンを実現するためのコントロールです。

ボタン同様、 7 つまでバッジを表示することができます。

コンテンツ を持つことができ、ON の時だけ関連付いたコンテンツをアクティブにすることも可能です。

CHECKBOX(コントロール名) {

プロパティ 1;

プロパティ 2;

:

:

プロパティ n;

};

11.13.1 記述例

```
WINDOW(255_000_00001) {
  STYLE = NOTITLEBAR|ANCHOR_CENTER;
  POSITION = 0,100;
  TEX_ID = 100_000_00001;
  CAPTION = 000_000_00010;
  SIZE = 512,256;
};

CHECKBOX(TEST) {
  ID = 000_002_00001;
  CAPTION = 000_000_00006;
  STYLE = ANCHOR_CENTER;
  POSITION = 0,32;
```

(次のページに続く)

(前のページからの続き)

```
  FONT_KIND= "fn40";
  TEX_ID1 = "bd01";
  TEXTURE_OFFSET1 = 0,10;
  SIZE = 256;
  CONTENTS = {
    ICON(TEST) {
      ID = 000_001_00010;
      POSITION = 256,16;
      SIZE =80,80;
      TEX_ID = 100_000_00001,"CRYS";
    };
  }

};
```

ON になったとき、ICON(TEST) が表示されます。

OFF のときは、ICON(TEST) は表示されません。

11.13.2 プロパティ

代表的なデフォルト値

```
TEX_ID = "BTN0?";
COLOR = 1,1,1,1;
CAPTION_COLOR = 1,1,1,1;
```

パーツ ID ルール

パーツ ID の 5 文字目を自動的に 0/1 に入れ替えます。

- XXXX0: OFF の状態
- XXXX1: ON の状態

```
TEX_ID = "BTN0?";
```

パーツ ID がこのように設定されているとすると以下のようなパーツを用意しておく必要があります。

- BTN00:OFF の状態
- BTN01:ON の状態

ID = コントロール ID

コントロール ID を定義します。

```
ID = 001_000_00010;
```

注釈: 設定しなかったときは、自動的にハッシュ値から生成します。

POSITION = X, Y

表示位置を決定します。STYLE の アンカー 指定に応じて、基準位置が変わります。

```
POSITION = 32,{50} + 64;
```

座標は 割合指定 が可能です。

SIZE = 横サイズ, 縦サイズ

表示サイズを変更します。省略したとき、若しくは、0 を指定すると、テクスチャパーツからサイズを取得してきます。

```
SIZE = 64,32; //64x32
SIZE = ,32;   //Set the width of the texture part width
SIZE = 64;            //Set the height of the texture part height
SIZE = {50} - 25;
```

サイズは 割合指定 が可能です。

TEX_ID = テクスチャ ID, パーツ ID

TEX_ID = パーツ ID

テクスチャ ID とパーツ ID を指定します。

COLOR = R,G,B,A

カラーを指定します。

R,G,B については、0〜2 の間で指定してください。

1 を超えたとき、そのカラー成分を 2 倍まで上げて表示することができます。

A については、0〜1 の間で指定してください。

CAPTION = キャプション ID

文字列の ID を指定してください。

こちら を参照ください。

CAPTION = "文字列"

文字列を設定します。

```
CAPTION = "Hellow world!";
```

CAPTION_COLOR = R,G,B,A

キャプションのカラーを指定できます。

0〜1 の間で指定してください。

CAPTION_OFFSET = X, Y

キャプションの位置を移動することができます。

キャプションオフセットは 割合指定 が可能です。

TEX_ID n = テクスチャ ID, パーツ ID

TEX_ID n = パーツ ID

バッジのテクスチャ ID とパーツ ID を指定します。

n = [1..7] で指定可能です。

TEXTURE_OFFSET n = オフセット X, オフセット Y

テクスチャの表示オフセットを指定します。

n = [0..7] で指定可能です。

テクスチャオフセットは 割合指定 が可能です。

TEXTURE_SIZE n = 横サイズ, 縦サイズ

テクスチャ ID とパーツ ID を指定します。

n = [0..7] で指定可能です。

n = 0 の時は、 **SIZE** と同じ意味です。

```
TEXTURE_SIZE1 = 64,32;         //64x32
TEXTURE_SIZE2 = ,32;  //Set the width of the texture part width
TEXTURE_SIZE3 = 64;   //Set the height of the texture part height
```

テクスチャサイズは 割合指定 が可能です。

COLOR n = R,G,B,A

バッジのカラーを指定します。

R,G,B については、0〜2 の間で指定してください。

1 を超えたとき、そのカラー成分が 2 倍までかけることができます。

A については、0〜1 の間で指定してくさい。

n = [1..7] で指定可能です。

SE_ID = SE_ID

押されたときに鳴らす音の ID を設定します。デフォルトは、clickSE です。0 を指定すると音が鳴らなくなります。

CONTENTS = { コントロール定義 … }

ON 状態の時に、自動的にアクティブになるコントロールを関連付けます。

OFF 状態の時には、自動的に HIDE 状態になります。全てのコントロールを関連付けることが可能です。

```
CONTENTS = {
   BUTTON(A) {
      :
   }
   TEXTURE(B) {
      :
   }
   LISTBOX(C) {
      :
   }
}
```

C#からコンテンツに対してアクセスする方法は こちら を参考にしてください。

注釈: コンテンツ内のコントロールの座標やサイズの 割合指定 には、リストボックスと異なり影響を与えません。

STYLE = スタイル 0|スタイル 1|..|スタイル n

コントロールの表示アンカーを指定できます。

表示位置アンカーフラグ	説明
ANCHOR_DEFAULT	アンカー位置を左上に設定 ANCHOR_LEFTTOP と同じ コントロールの中心位置はデフォルトで、BASE_LEFTTOP になる。
ANCHOR_LEFTTOP	アンカー位置を左上に設定 コントロールの中心位置はデフォルトで、BASE_LEFTTOP になる。
ANCHOR_LEFT	アンカー位置を左に設定 縦にはセンタリング コントロールの中心位置はデフォルトで、BASE_LEFT になる。
ANCHOR_LEFTBOTTOM	アンカー位置を左に設定 縦には下辺を基準に配置 コントロールの中心位置はデフォルトで、BASE_LEFTBOTTOM になる。
ANCHOR_TOP	アンカー位置を上辺に設定 横方向にはセンタリング コントロールの中心位置はデフォルトで、BASE_TOP になる。
ANCHOR_CENTER	アンカー位置を画面中央にセンタリング コントロールの中心位置はデフォルトで、BASE_CENTER になる。
ANCHOR_BOTTOM	アンカー位置を底辺に設定 横方向にはセンタリング コントロールの中心位置はデフォルトで、BASE_BOTTOM になる。
ANCHOR_RIGHTTOP	アンカー位置を右上に設定 コントロールの中心位置はデフォルトで、BASE_RIGHTBOTTOM になる。
ANCHOR_RIGHT	アンカー位置を右に設定 縦にはセンタリング コントロールの中心位置はデフォルトで、BASE_RIGHT になる。
ANCHOR_RIGHTBOTTOM	アンカー位置を右に設定 縦には下辺を基準に配置 コントロールの中心位置はデフォルトで、BASE_RIGHTBOTTOM になる。

コントロールの中心位置を指定できます。

中心位置変更フラグ	説明
BASE_DEFAULT	アンカーフラグに応じて変わる。 特に指定しなければ、これになります。 各アンカーフラグの説明を参照
BASE_LEFTTOP	中心位置をコントロールの左上に設定
BASE_LEFT	中心位置を左に設定 縦にはコントロールの真中
BASE_LEFTBOTTOM	中心位置を左下に設定
BASE_TOP	中心位置を上辺に設定 横にはコントロールの中心
BASE_CENTER	中心位置をコントロールの中心
BASE_BOTTOM	中心位置を底辺に設定 横方向にはコントロールの真中
BASE_RIGHTTOP	中心位置をコントロールの右上に設定
BASE_RIGHT	中心位置を右に設定 横にはコントロールの中心
BASE_RIGHTBOTTOM	中心位置を右下に設定

キャプションの中心位置を指定できます。

テキストアンカー	説明
TEXT_CENTER	テキスト表示を中心に合わせます。
TEXT_LEFT	テキスト表示を左に合わせます。
TEXT_RIGHT	テキスト表示を右に合わせます。

キャプションのフォント装飾を変更できます。

キャプション装飾種類	説明
TEXT_NORMAL	装飾なし
TEXT_BOLD	太字
TEXT_DENT	へこんだ感じの文字
TEXT_SHADOW	影付き文字

機能を制限するスタイルは以下のものがあります。

機能制限スタイル	説明
HIDE	表示を隠す。
DRAG	ドラッグ可能にする。
DISABLE	押せなくし、暗くする。
NOHIT	押せなくする。

11.14 TEXTURE

C#: *CWinCtrlTexture*

テクスチャパーツを表示するためのコントロールです。

最大 8 枚まで重ねて表示することが可能です。

デフォルトでは、押したことを検出できません。コールバックが必要なときは、STYLE に HIT を指定してください。

TEXTURE(コントロール名) {

プロパティ 1;

プロパティ 2;

:

:

プロパティ n;

};

11.14.1 記述例

```
WINDOW(255_000_00001) {
  STYLE = NOTITLEBAR|ANCHOR_CENTER;
  POSITION = 0,100;
  TEX_ID = 100_000_00001;
  TEXTURE_ZOFFSET = 100_010_00000,1;
  SIZE = 512,256;
};
TEXTURE(DecoLeft) {
  ID = 000_000_00040;
  STYLE = NOHIT;
  TEX_ID = 100_010_00000,"DECOL";
```

(次のページに続く)

(前のページからの続き)

```
  POSITION = 24,24;
};
```

11.14.2 プロパティ

代表的なデフォルト値

```
COLOR = 1,1,1,1;
CAPTION_COLOR = 1,1,1,1;
SE_ID = 0;
```

ID = コントロール ID

コントロール ID を定義します。

```
ID = 001_000_00010;
```

注釈: 設定しなかったときは、自動的にハッシュ値から生成します。

POSITION = X, Y

表示位置を決定します。STYLE の アンカー 指定に応じて、基準位置が変わります。

```
POSITION = 32,{50} + 64;
```

座標は 割合指定 が可能です。

SIZE = 横サイズ, 縦サイズ

表示サイズを変更します。省略したとき、若しくは、0 を指定すると、テクスチャパーツからサイズを取得してきます。

```
SIZE = 64,32; //64x32
SIZE = ,32;   //Set the width of the texture part width
SIZE = 64;            //Set the height of the texture part height
SIZE = {50} - 25;
```

サイズは 割合指定 が可能です。

TEX_ID = テクスチャ ID, パーツ ID

TEX_ID = パーツ ID

テクスチャ ID とパーツ ID を指定します。

TEX_ID n = テクスチャ ID, パーツ ID

TEX_ID n = パーツ ID

テクスチャ ID とパーツ ID を指定します。

n = [0..7] で指定可能です。

n = 0 の時は、TEX_ID と同じテクスチャを操作します。

TEXTURE_OFFSET n = オフセット X, オフセット Y

テクスチャの表示オフセットを指定します。

n = [0..7] で指定可能です。

テクスチャオフセットは 割合指定 が可能です。

TEXTURE_SIZE n = 横サイズ, 縦サイズ

テクスチャ ID とパーツ ID を指定します。

n = [0..7] で指定可能です。

n = 0 の時は、 **SIZE** と同じ意味です。

```
TEXTURE_SIZE1 = 64,32;         //64x32
TEXTURE_SIZE2 = ,32;  //Set the width of the texture part width
TEXTURE_SIZE3 = 64;   //Set the height of the texture part height
```

テクスチャサイズは 割合指定 が可能です。

COLOR = R,G,B,A

カラーを指定します。

R,G,B については、0〜2 の間で指定してください。

1 を超えたとき、そのカラー成分を 2 倍まで上げて表示することができます。

A については、0〜1 の間で指定してください。

COLOR n = R,G,B,A

カラーを指定します。

R,G,B については、0〜2 の間で指定してください。

1 を超えたとき、そのカラー成分を 2 倍まで上げて表示することができます。

A については、0〜1 の間で指定してください。

n = [0..7] で指定可能です。

n = 0 の時は、 **COLOR** と同じプロパティです。

SE_ID = SE_ID

押されたときに鳴らす音の ID を設定します。デフォルトは、0 になっており、音がなりません。

STYLE = フラグ 0|フラグ 1|..|フラグ n

コントロールの表示アンカーを指定できます。

表示位置アンカーフラグ	説明
ANCHOR_DEFAULT	アンカー位置を左上に設定 ANCHOR_LEFTTOP と同じ コントロールの中心位置はデフォルトで、BASE_LEFTTOP になる。
ANCHOR_LEFTTOP	アンカー位置を左上に設定 コントロールの中心位置はデフォルトで、BASE_LEFTTOP になる。
ANCHOR_LEFT	アンカー位置を左に設定 縦にはセンタリング コントロールの中心位置はデフォルトで、BASE_LEFT になる。
ANCHOR_LEFTBOTTOM	アンカー位置を左に設定 縦には下辺を基準に配置 コントロールの中心位置はデフォルトで、BASE_LEFTBOTTOM になる。
ANCHOR_TOP	アンカー位置を上辺に設定 横方向にはセンタリング コントロールの中心位置はデフォルトで、BASE_TOP になる。
ANCHOR_CENTER	アンカー位置を画面中央にセンタリング コントロールの中心位置はデフォルトで、BASE_CENTER になる。
ANCHOR_BOTTOM	アンカー位置を底辺に設定 横方向にはセンタリング コントロールの中心位置はデフォルトで、BASE_BOTTOM になる。
ANCHOR_RIGHTTOP	アンカー位置を右上に設定 コントロールの中心位置はデフォルトで、BASE_RIGHTBOTTOM になる。
ANCHOR_RIGHT	アンカー位置を右に設定 縦にはセンタリング コントロールの中心位置はデフォルトで、BASE_RIGHT になる。
ANCHOR_RIGHTBOTTOM	アンカー位置を右に設定 縦には下辺を基準に配置 コントロールの中心位置はデフォルトで、BASE_RIGHTBOTTOM になる。

コントロールの中心位置を指定できます。

中心位置変更フラグ	説明
BASE_DEFAULT	アンカーフラグに応じて変わる。 特に指定しなければ、これになります。 各アンカーフラグの説明を参照
BASE_LEFTTOP	中心位置をコントロールの左上に設定
BASE_LEFT	中心位置を左に設定 縦にはコントロールの真中
BASE_LEFTBOTTOM	中心位置を左下に設定
BASE_TOP	中心位置を上辺に設定 横にはコントロールの中心
BASE_CENTER	中心位置をコントロールの中心
BASE_BOTTOM	中心位置を底辺に設定 横方向にはコントロールの真中
BASE_RIGHTTOP	中心位置をコントロールの右上に設定
BASE_RIGHT	中心位置を右に設定 横にはコントロールの中心
BASE_RIGHTBOTTOM	中心位置を右下に設定

機能を制限するスタイルは以下のものがあります。

機能制限スタイル	説明
HIDE	表示を隠す。
DRAG	ドラッグ可能にする。
DISABLE	押せなくし、暗くする。
HIT	押せるようにする。

11.15 LINE

C#: *CWinCtrlLine*

線分を引くためのコントロールです。

デフォルトでは、押したことを検出できません。コールバックが必要なときは、STYLE に HIT を指定してください。

LINE(コントロール名) {

プロパティ 1;

プロパティ 2;

:

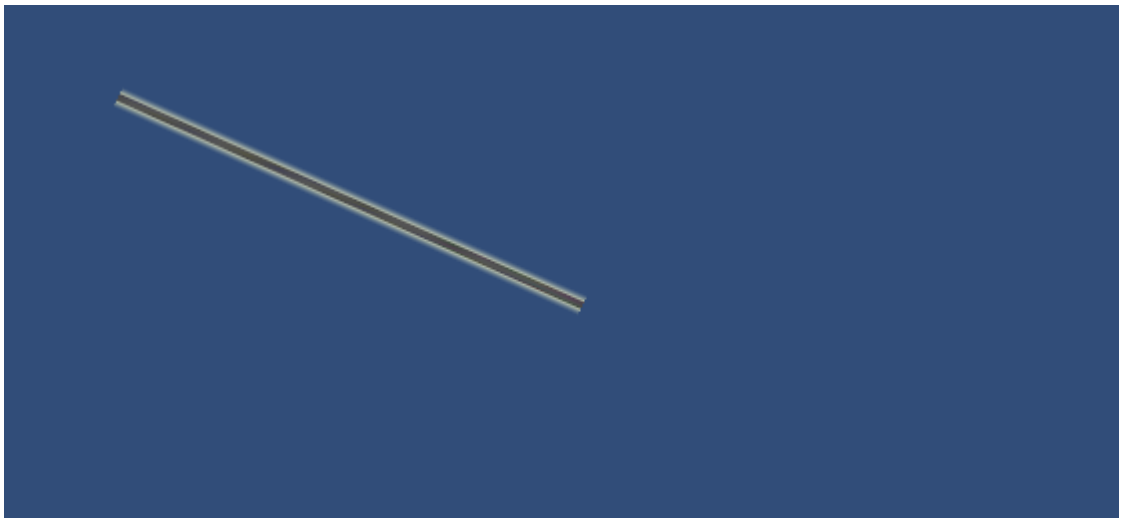
:

プロパティ n;

};

11.15.1 記述例

```
WINDOW(255_000_00001) {
  STYLE = NOTITLEBAR|NOFRAME|ANCHOR_CENTER;
  POSITION = 0,100;
  TEX_ID = 100_010_00000;
  SIZE = 512,256;
};
LINE(TEST) {
  ID = 000_000_00040;
  TEX_ID = "LINE0";
  POSITION = 24,24;
  POSITION2 = 256,128;
};
```



11.15.2 プロパティ

代表的なデフォルト値

```
TEX_ID = "LINE";
COLOR = 1,1,1,1;
CAPTION_COLOR = 1,1,1,1;
```

ID = コントロール ID

コントロール ID を定義します。

```
ID = 001_000_00010;
```

注釈: 設定しなかったときは、自動的にハッシュ値から生成します。

POSITION = X, Y

開始座標を決定します。STYLE の アンカー 指定に応じて、基準位置が変わります。

```
POSITION = 32,{50} + 64;
```

開始座標は 割合指定 が可能です。

POSITION2 = X, Y

終端座標を決定します。STYLE の アンカー 指定に応じて、基準位置が変わります。

```
POSITION2 = 32,{50} + 64;
```

終端座標は 割合指定 が可能です。

TEX_ID = テクスチャ ID, パーツ ID

TEX_ID = パーツ ID

テクスチャ ID とパーツ ID を指定します。

SIZE = 線分長さ, 線分太さ

X に、線分の長さが自動的に入ります。

Y に、線分の太さを指定してください。

線分の太さには 割合指定 が可能です。

```
SIZE = ,32;    //line width = 32
```

COLOR = R,G,B,A

カラーを指定します。

R,G,B については、0〜2 の間で指定してください。

1 を超えたとき、そのカラー成分を 2 倍まで上げて表示することができます。

A については、0〜1 の間で指定してください。

STYLE = スタイル 0|スタイル 1|..|スタイル n

コントロールの表示アンカーを指定できます。

表示位置アンカーフラグ	説明
ANCHOR_DEFAULT	アンカー位置を左上に設定 ANCHOR_LEFTTOP と同じ コントロールの中心位置はデフォルトで、BASE_LEFTTOP になる。
ANCHOR_LEFTTOP	アンカー位置を左上に設定 コントロールの中心位置はデフォルトで、BASE_LEFTTOP になる。
ANCHOR_LEFT	アンカー位置を左に設定 縦にはセンタリング コントロールの中心位置はデフォルトで、BASE_LEFT になる。
ANCHOR_LEFTBOTTOM	アンカー位置を左に設定 縦には下辺を基準に配置 コントロールの中心位置はデフォルトで、BASE_LEFTBOTTOM になる。
ANCHOR_TOP	アンカー位置を上辺に設定 横方向にはセンタリング コントロールの中心位置はデフォルトで、BASE_TOP になる。
ANCHOR_CENTER	アンカー位置を画面中央にセンタリング コントロールの中心位置はデフォルトで、BASE_CENTER になる。
ANCHOR_BOTTOM	アンカー位置を底辺に設定 横方向にはセンタリング コントロールの中心位置はデフォルトで、BASE_BOTTOM になる。
ANCHOR_RIGHTTOP	アンカー位置を右上に設定 コントロールの中心位置はデフォルトで、BASE_RIGHTBOTTOM になる。
ANCHOR_RIGHT	アンカー位置を右に設定 縦にはセンタリング コントロールの中心位置はデフォルトで、BASE_RIGHT になる。
ANCHOR_RIGHTBOTTOM	アンカー位置を右に設定 縦には下辺を基準に配置 コントロールの中心位置はデフォルトで、BASE_RIGHTBOTTOM になる。

コントロールの中心位置を指定できます。

中心位置変更フラグ	説明
BASE_DEFAULT	アンカーフラグに応じて変わる。 特に指定しなければ、これになります。 各アンカーフラグの説明を参照
BASE_LEFTTOP	中心位置をコントロールの左上に設定
BASE_LEFT	中心位置を左に設定 縦にはコントロールの真中
BASE_LEFTBOTTOM	中心位置を左下に設定
BASE_TOP	中心位置を上辺に設定 横にはコントロールの中心
BASE_CENTER	中心位置をコントロールの中心
BASE_BOTTOM	中心位置を底辺に設定 横方向にはコントロールの真中
BASE_RIGHTTOP	中心位置をコントロールの右上に設定
BASE_RIGHT	中心位置を右に設定 横にはコントロールの中心
BASE_RIGHTBOTTOM	中心位置を右下に設定

機能を制限するスタイルは以下のものがあります。

機能制限スタイル	説明
HIDE	表示を隠す。
DRAG	ドラッグ可能にする。
DISABLE	押せなくし、暗くする。
HIT	押せるようにする。

11.16 RENDER

C#: *CWinCtrlRender*

レンダーテクスチャに対して自由にレンダリングするためのコントロールです。

コントロールが生成されたとき、一緒にカメラも生成されます。

そのカメラにレンダリングした対象が描画されることになります。

注釈: **RENDER** を用いてレンダリングするためのリソースを確保したときは、クローズ時には忘れずに削除するようにしてください。

削除し忘れると、ウィンドウクローズの度にリークすることになります。

RENDER(コントロール名) {

プロパティ 1;

プロパティ 2;

:

:

プロパティ n;

};

11.16.1 記述例

```
WINDOW(255_000_00001) {
  STYLE = NOTITLEBAR|ANCHOR_CENTER;
  POSITION = 0,100;
  SIZE = 512,256;
};

RENDER(AVATAR) {
  ID = 000_000_00200;
  STYLE = ANCHOR_CENTER;
  POSITION = 0,0;
  SIZE = {100}-64,{100}-64;
  CONTENTS_SIZE = 512,512;
};
```

11.16.2 プロパティ

代表的なデフォルト値

```
COLOR = 1,1,1,1;
SE_ID = 0;
```

ID = コントロール ID

コントロール ID を定義します。

```
ID = 001_000_00010;
```

注釈: 設定しなかったときは、自動的にハッシュ値から生成します。

POSITION = X, Y

表示位置を決定します。STYLE の アンカー 指定に応じて、基準位置が変わります。

```
POSITION = 32,{50} + 64;
```

座標は 割合指定 が可能です。

SIZE = 横サイズ, 縦サイズ

表示サイズを変更します。

```
SIZE = 128,128;        //64x32
```

サイズは 割合指定 が可能です。

注釈: 必須のプロパティです。

CONTENTS_SIZE = 横サイズ, 縦サイズ

レンダーテクスチャのサイズを指定します。

正方形で指定し、2 のべき乗の値で指定してください。

```
CONTENTS_SIZE = 1024,1024;    //render texture size
```

注釈: 必須のプロパティです。正方形で指定し、幅、高さ共に 2 のべき乗を指定するようにしてください。また、最大 2048 以内に収めておくと互換性の問題が出ずらいです。

注釈: 割合指定も可能ですが、互換性の問題もあり、直値を指定するようにしてください。

COLOR = R,G,B,A

カラーを指定します。

R,G,B については、0〜2 の間で指定してください。

1 を超えたとき、そのカラー成分を 2 倍まで上げて表示することができます。

A については、0〜1 の間で指定してください。

COLOR1 = R,G,B,A

レンダリングする際の背景のクリアカラーを指定します。

R,G,B,A については、0〜1 の間で指定してください。

SE_ID = SE_ID

押されたときに鳴らす音の ID を設定します。デフォルトは、0 になっており、音がなりません。

TEX_ID = レンダーテクスチャ ID 名

レンダーテクスチャを生成する際の名前として使用されます。この値を、WINDOW のプロパティ: **TEXTURE_ZOFFSET** に指定することによって、レンダリング順序を制御することができます。

STYLE = フラグ 0|フラグ 1|..|フラグ n

コントロールの表示アンカーを指定できます。

表示位置アンカーフラグ	説明
ANCHOR_DEFAULT	アンカー位置を左上に設定 ANCHOR_LEFTTOP と同じ コントロールの中心位置はデフォルトで、BASE_LEFTTOP になる。
ANCHOR_LEFTTOP	アンカー位置を左上に設定 コントロールの中心位置はデフォルトで、BASE_LEFTTOP になる。
ANCHOR_LEFT	アンカー位置を左に設定 縦にはセンタリング コントロールの中心位置はデフォルトで、BASE_LEFT になる。
ANCHOR_LEFTBOTTOM	アンカー位置を左に設定 縦には下辺を基準に配置 コントロールの中心位置はデフォルトで、BASE_LEFTBOTTOM になる。
ANCHOR_TOP	アンカー位置を上辺に設定 横方向にはセンタリング コントロールの中心位置はデフォルトで、BASE_TOP になる。
ANCHOR_CENTER	アンカー位置を画面中央にセンタリング コントロールの中心位置はデフォルトで、BASE_CENTER になる。
ANCHOR_BOTTOM	アンカー位置を底辺に設定 横方向にはセンタリング コントロールの中心位置はデフォルトで、BASE_BOTTOM になる。
ANCHOR_RIGHTTOP	アンカー位置を右上に設定 コントロールの中心位置はデフォルトで、BASE_RIGHTBOTTOM になる。
ANCHOR_RIGHT	アンカー位置を右に設定 縦にはセンタリング コントロールの中心位置はデフォルトで、BASE_RIGHT になる。
ANCHOR_RIGHTBOTTOM	アンカー位置を右に設定 縦には下辺を基準に配置 コントロールの中心位置はデフォルトで、BASE_RIGHTBOTTOM になる。

コントロールの中心位置を指定できます。

中心位置変更フラグ	説明
BASE_DEFAULT	アンカーフラグに応じて変わる。 特に指定しなければ、これになります。 各アンカーフラグの説明を参照
BASE_LEFTTOP	中心位置をコントロールの左上に設定
BASE_LEFT	中心位置を左に設定 縦にはコントロールの真中
BASE_LEFTBOTTOM	中心位置を左下に設定
BASE_TOP	中心位置を上辺に設定 横にはコントロールの中心
BASE_CENTER	中心位置をコントロールの中心
BASE_BOTTOM	中心位置を底辺に設定 横方向にはコントロールの真中
BASE_RIGHTTOP	中心位置をコントロールの右上に設定
BASE_RIGHT	中心位置を右に設定 横にはコントロールの中心
BASE_RIGHTBOTTOM	中心位置を右下に設定

機能を制限するスタイルは以下のものがあります。

機能制限スタイル	説明
HIDE	表示を隠す。
DRAG	ドラッグ可能にする。
DISABLE	押せなくし、暗くする。
NOHIT	押せなくする。

11.17 ICON

C#: *CWinCtrlIcon*

アイコンを表示するためのコントロールです。

最大 8 枚のテクスチャパーツを重ねて表示することが可能です。

TEXTURE コントロールと振る舞いはほぼ同じですが、タッチしたとき暗くなります。

また、*TEXTURE* コントロールと違い、キャプションが持てます。

```
ICON(コントロール名) {
    プロパティ 1;
```

プロパティ 2;

:

:

プロパティ n;

};

11.17.1 記述例

```
WINDOW(001_002_00003) {
  STYLE = NOTITLEBAR|ANCHOR_CENTER;
  POSITION = 0,100;
  SIZE = 512,256;
};

ICON(TEST) {
  ID = 000_001_00010;
  POSITION = 256,24;
  SIZE =80,80;
  TEX_ID = 100_000_00001,"BNAD0";
  TEX_ID1 = 100_000_00001,"bd01";
  TEX_ID3 = 100_000_00001,"CRYS";
};
```

11.17.2 プロパティ

代表的なデフォルト値

```
COLOR = 1,1,1,1;
CAPTION_COLOR = 1,1,1,1;
SE_ID = 0;
```

ID = コントロール ID

コントロール ID を定義します。

```
ID = 001_000_00010;
```

注釈: 設定しなかったときは、自動的にハッシュ値から生成します。

POSITION = X, Y

表示位置を決定します。STYLE の アンカー 指定に応じて、基準位置が変わります。

```
POSITION = 32,{50} + 64;
```

座標は 割合指定 が可能です。

SIZE = 横サイズ, 縦サイズ

表示サイズを変更します。省略したとき、若しくは、0 を指定すると、テクスチャパーツからサイズを取得してきます。

```
SIZE = 64,32; //64x32
SIZE = ,32;   //Set the width of the texture part width
SIZE = 64;            //Set the height of the texture part height
SIZE = {50} - 25;
```

サイズは 割合指定 が可能です。

TEX_ID = テクスチャ ID, パーツ ID

TEX_ID = パーツ ID

テクスチャ ID とパーツ ID を指定します。

TEX_ID n = テクスチャ ID, パーツ ID

TEX_ID n = パーツ ID

テクスチャ ID とパーツ ID を指定します。

n = [0..7] で指定可能です。

n = 0 の時は、TEX_ID と同じテクスチャを操作します。

TEXTURE_OFFSET n = オフセット X, オフセット Y

テクスチャの表示オフセットを指定します。

n = [0..7] で指定可能です。

テクスチャオフセットは 割合指定 が可能です。

TEXTURE_SIZE n = 横サイズ, 縦サイズ

テクスチャ ID とパーツ ID を指定します。

n = [0..7] で指定可能です。

n = 0 の時は、 **SIZE** と同じ意味です。

```
TEXTURE_SIZE1 = 64,32;         //64x32
TEXTURE_SIZE2 = ,32;  //Set the width of the texture part width
TEXTURE_SIZE3 = 64;   //Set the height of the texture part height
```

テクスチャサイズは 割合指定 が可能です。

COLOR = R,G,B,A

カラーを指定します。

R,G,B については、0〜2 の間で指定してください。

1 を超えたとき、そのカラー成分を 2 倍まで上げて表示することができます。

A については、0〜1 の間で指定してください。

SE_ID = SE_ID

押されたときに鳴らす音の ID を設定します。デフォルトは、0 になっており、音がなりません。

COLOR n = R,G,B,A

カラーを指定します。

R,G,B については、0〜2 の間で指定してください。

1 を超えたとき、そのカラー成分を 2 倍まで上げて表示することができます。

A については、0〜1 の間で指定してください。

n = [0..7] で指定可能です。

n = 0 の時は、 **COLOR** と同じプロパティです。

CAPTION = キャプション ID

文字列の ID を指定してください。

こちら を参照ください。

CAPTION = "文字列"

文字列を設定します。

```
CAPTION = "Hellow world!";
```

CAPTION_COLOR = R,G,B,A

キャプションのカラーを指定できます。

0〜1 の間で指定してください。

CAPTION_OFFSET = X, Y

キャプションの位置を移動することができます。

キャプションオフセットは 割合指定 が可能です。

STYLE = フラグ 0|フラグ 1|..|フラグ n

コントロールの表示アンカーを指定できます。

表示位置アンカーフラグ	説明
ANCHOR_DEFAULT	アンカー位置を左上に設定 ANCHOR_LEFTTOP と同じ コントロールの中心位置はデフォルトで、BASE_LEFTTOP になる。
ANCHOR_LEFTTOP	アンカー位置を左上に設定 コントロールの中心位置はデフォルトで、BASE_LEFTTOP になる。
ANCHOR_LEFT	アンカー位置を左に設定 縦にはセンタリング コントロールの中心位置はデフォルトで、BASE_LEFT になる。
ANCHOR_LEFTBOTTOM	アンカー位置を左に設定 縦には下辺を基準に配置 コントロールの中心位置はデフォルトで、BASE_LEFTBOTTOM になる。
ANCHOR_TOP	アンカー位置を上辺に設定 横方向にはセンタリング コントロールの中心位置はデフォルトで、BASE_TOP になる。
ANCHOR_CENTER	アンカー位置を画面中央にセンタリング コントロールの中心位置はデフォルトで、BASE_CENTER になる。
ANCHOR_BOTTOM	アンカー位置を底辺に設定 横方向にはセンタリング コントロールの中心位置はデフォルトで、BASE_BOTTOM になる。
ANCHOR_RIGHTTOP	アンカー位置を右上に設定 コントロールの中心位置はデフォルトで、BASE_RIGHTBOTTOM になる。
ANCHOR_RIGHT	アンカー位置を右に設定 縦にはセンタリング コントロールの中心位置はデフォルトで、BASE_RIGHT になる。
ANCHOR_RIGHTBOTTOM	アンカー位置を右に設定 縦には下辺を基準に配置 コントロールの中心位置はデフォルトで、BASE_RIGHTBOTTOM になる。

コントロールの中心位置を指定できます。

中心位置変更フラグ	説明
BASE_DEFAULT	アンカーフラグに応じて変わる。 特に指定しなければ、これになります。 各アンカーフラグの説明を参照
BASE_LEFTTOP	中心位置をコントロールの左上に設定
BASE_LEFT	中心位置を左に設定 縦にはコントロールの真中
BASE_LEFTBOTTOM	中心位置を左下に設定
BASE_TOP	中心位置を上辺に設定 横にはコントロールの中心
BASE_CENTER	中心位置をコントロールの中心
BASE_BOTTOM	中心位置を底辺に設定 横方向にはコントロールの真中
BASE_RIGHTTOP	中心位置をコントロールの右上に設定
BASE_RIGHT	中心位置を右に設定 横にはコントロールの中心
BASE_RIGHTBOTTOM	中心位置を右下に設定

キャプションの中心位置を指定できます。

テキストアンカー	説明
TEXT_CENTER	テキスト表示を中心に合わせます。
TEXT_LEFT	テキスト表示を左に合わせます。
TEXT_RIGHT	テキスト表示を右に合わせます。

キャプションのフォント装飾を変更できます。

キャプション装飾種類	説明
TEXT_NORMAL	装飾なし
TEXT_BOLD	太字
TEXT_DENT	へこんだ感じの文字
TEXT_SHADOW	影付き文字

機能を制限するスタイルは以下のものがあります。

機能制限スタイル	説明
HIDE	表示を隠す。
DRAG	ドラッグ可能にする。
DISABLE	押せなくし、暗くする。
NOHIT	押せなくする。

11.18 RECASTICON

C#: *CWinCtrlRecastIcon*

リキャストアイコンを表示するためのコントロールです。

最大 8 枚のテクスチャパーツを重ねて表示することが可能です。

ICON とほぼ同じ振る舞いを行います。

リキャスト設定を行うと、指定したテクスチャパーツが、下から徐々に充填されるようなアニメーションを行います。リキャスト時間は、0 の時、*TEXTURE* と同等の表示を行い、1 以上の時何も表示されないようになっています。

11.18.1 記述例

```
WINDOW(255_000_00001) {
  STYLE = NOTITLEBAR|NOFRAME|ANCHOR_CENTER;
  POSITION = 0,100;
  SIZE = 512,256;
};

RECASTICON(Test) {
  ID = 000_000_00010;
  POSITION = 5,5;
  STYLE = ANCHOR_LEFTBOTTOM|INPUT_NOBLOCK;
  TEX_ID0 = 100_200_00000,"btrc";
  TEX_ID1 = 100_200_00000,"SKLWT";
  TEX_ID2 = 100_200_00000,"aF01";
  SIZE = 70,70;
  TEXTURE_OFFSET1 = 5,5;
  TEXTURE_OFFSET2 = 6,6;
  SE_ID = 0;
  PRIORITY = 1;
};
```

11.18.2 プロパティ

代表的なデフォルト値

```
COLOR = 1,1,1,1;
CAPTION_COLOR = 1,1,1,1;
SE_ID = 0;
```

ID = コントロール ID

コントロール ID を定義します。

```
ID = 001_000_00010;
```

注釈: 設定しなかったときは、自動的にハッシュ値から生成します。

POSITION = X, Y

表示位置を決定します。STYLE の アンカー 指定に応じて、基準位置が変わります。

```
POSITION = 32,{50} + 64;
```

座標は 割合指定 が可能です。

SIZE = 横サイズ, 縦サイズ

表示サイズを変更します。省略したとき、若しくは、0 を指定すると、テクスチャパーツからサイズを取得してきます。

```
SIZE = 64,32; //64x32
SIZE = ,32;   //Set the width of the texture part width
SIZE = 64;            //Set the height of the texture part height
SIZE = {50} - 25;
```

サイズは 割合指定 が可能です。

TEX_ID = テクスチャ ID, パーツ ID

TEX_ID = パーツ ID

テクスチャ ID とパーツ ID を指定します。

TEX_ID n = テクスチャ ID, パーツ ID

TEX_ID n = パーツ ID

テクスチャ ID とパーツ ID を指定します。

n = [0..7] で指定可能です。

n = 0 の時は、TEX_ID と同じテクスチャを操作します。

TEXTURE_OFFSET n = オフセット X, オフセット Y

テクスチャの表示オフセットを指定します。

n = [0..7] で指定可能です。

テクスチャオフセットは 割合指定 が可能です。

TEXTURE_SIZE n = 横サイズ, 縦サイズ

テクスチャ ID とパーツ ID を指定します。

n = [0..7] で指定可能です。

n = 0 の時は、 **SIZE** と同じ意味です。

```
TEXTURE_SIZE1 = 64,32;         //64x32
TEXTURE_SIZE2 = ,32;  //Set the width of the texture part width
TEXTURE_SIZE3 = 64;   //Set the height of the texture part height
```

テクスチャサイズは 割合指定 が可能です。

COLOR = R,G,B,A

カラーを指定します。

R,G,B については、0〜2 の間で指定してください。

1 を超えたとき、そのカラー成分を 2 倍まで上げて表示することができます。

A については、0〜1 の間で指定してください。

COLOR n = R,G,B,A

カラーを指定します。

R,G,B については、0〜2 の間で指定してください。

1 を超えたとき、そのカラー成分を 2 倍まで上げて表示することができます。

A については、0〜1 の間で指定してください。

n = [0..7] で指定可能です。

n = 0 の時は、 **COLOR** と同じプロパティです。

FONT_KIND = フォント種類

フォントの種類を指定します。

```
FONT_KIND = "fnt32";
```

CAPTION = キャプション ID

文字列の ID を指定してください。

こちら を参照ください。

```
CAPTION = 010_000_00100;
```

CAPTION = "文字列"

文字列を設定します。

```
CAPTION = "Hellow world!";
```

CAPTION_COLOR = R,G,B,A

キャプションのカラーを指定できます。

0〜1 の間で指定してください。

CAPTION_OFFSET = X, Y

キャプションの位置を移動することができます。

キャプションオフセットは 割合指定 が可能です。

SE_ID = SE_ID

押されたときに鳴らす音の ID を設定します。デフォルトは、0 になっており、音がなりません。

STYLE = フラグ **0**|フラグ **1**|..|フラグ **n**

コントロールの表示アンカーを指定できます。

表示位置アンカーフラグ	説明
ANCHOR_DEFAULT	アンカー位置を左上に設定 ANCHOR_LEFTTOP と同じ コントロールの中心位置はデフォルトで、BASE_LEFTTOP になる。
ANCHOR_LEFTTOP	アンカー位置を左上に設定 コントロールの中心位置はデフォルトで、BASE_LEFTTOP になる。
ANCHOR_LEFT	アンカー位置を左に設定 縦にはセンタリング コントロールの中心位置はデフォルトで、BASE_LEFT になる。
ANCHOR_LEFTBOTTOM	アンカー位置を左に設定 縦には下辺を基準に配置 コントロールの中心位置はデフォルトで、BASE_LEFTBOTTOM になる。
ANCHOR_TOP	アンカー位置を上辺に設定 横方向にはセンタリング コントロールの中心位置はデフォルトで、BASE_TOP になる。
ANCHOR_CENTER	アンカー位置を画面中央にセンタリング コントロールの中心位置はデフォルトで、BASE_CENTER になる。
ANCHOR_BOTTOM	アンカー位置を底辺に設定 横方向にはセンタリング コントロールの中心位置はデフォルトで、BASE_BOTTOM になる。
ANCHOR_RIGHTTOP	アンカー位置を右上に設定 コントロールの中心位置はデフォルトで、BASE_RIGHTBOTTOM になる。
ANCHOR_RIGHT	アンカー位置を右に設定 縦にはセンタリング コントロールの中心位置はデフォルトで、BASE_RIGHT になる。
ANCHOR_RIGHTBOTTOM	アンカー位置を右に設定 縦には下辺を基準に配置 コントロールの中心位置はデフォルトで、BASE_RIGHTBOTTOM になる。

コントロールの中心位置を指定できます。

中心位置変更フラグ	説明
BASE_DEFAULT	アンカーフラグに応じて変わる。 特に指定しなければ、これになります。 各アンカーフラグの説明を参照
BASE_LEFTTOP	中心位置をコントロールの左上に設定
BASE_LEFT	中心位置を左に設定 縦にはコントロールの真中
BASE_LEFTBOTTOM	中心位置を左下に設定
BASE_TOP	中心位置を上辺に設定 横にはコントロールの中心
BASE_CENTER	中心位置をコントロールの中心
BASE_BOTTOM	中心位置を底辺に設定 横方向にはコントロールの真中
BASE_RIGHTTOP	中心位置をコントロールの右上に設定
BASE_RIGHT	中心位置を右に設定 横にはコントロールの中心
BASE_RIGHTBOTTOM	中心位置を右下に設定

キャプションの中心位置を指定できます。

テキストアンカー	説明
TEXT_CENTER	テキスト表示を中心に合わせます。
TEXT_LEFT	テキスト表示を左に合わせます。
TEXT_RIGHT	テキスト表示を右に合わせます。

キャプションのフォント装飾を変更できます。

キャプション装飾種類	説明
TEXT_NORMAL	装飾なし
TEXT_BOLD	太字
TEXT_DENT	へこんだ感じの文字
TEXT_SHADOW	影付き文字

機能を制限するスタイルは以下のものがあります。

機能制限スタイル	説明
HIDE	表示を隠す。
DRAG	ドラッグ可能にする。
DISABLE	押せなくし、暗くする。
NOHIT	押せなくする。

11.19 RENDERICON

C#: *CWinCtrlRenderIcon*

アイコンにレンダリングするためのコントロールです。

指定されたレンダーテクスチャから自動的に矩形を切り出し、レンダリングします。

複数のレンダリング結果を一枚のレンダーテクスチャ内に含めることができるため、一度レンダリングした後のアイコンの描画が高速です。

例えば、リストボックス内のコンテンツに着せ替え可能アバターアイコンを表示したいときなどに使うと効果的です。

また、通常の *ICON* と同様、通常のテクスチャも一緒にレンダリング可能です。

ただし、 *TEXTURE_ZOFFSET* には気を付ける必要があります。

```
RENDERICON(コントロール名) {
    プロパティ 1;
    プロパティ 2;
        :
        :
    プロパティ n;
};
```

注釈: レンダリング対象に割り当てられるのは、 1 枚のみです。

11.19.1 記述例

```
RENDERICON(Avatar) {
   ID = 000_000_00330;
   POSITION = -315,-22;
   TEX_ID = 255_000_00010;            //render texture id
   CONTENTS_SIZE = 1024,1024; //render texture size
   SIZE = 64,64;                              //display size
};
```

この例では、1024/64 x 1024/64 = 256 個のアイコンを一枚のテクスチャに共有させることができます。例えば、リストにアイコンを列挙したとき、256 個まで一枚のレンダーテクスチャにまとめて、レンダリング可能です。ただし、256 個以上の共有を行うと、領域の確保に失敗し、レンダリングできなくなります。

11.19.2 プロパティ

代表的なデフォルト値

```
COLOR = 1,1,1,1;
SE_ID = 0;
```

ID = コントロール ID

コントロール ID を定義します。

```
ID = 001_000_00010;
```

注釈: 設定しなかったときは、自動的にハッシュ値から生成します。

POSITION = X，Y

表示位置を決定します。STYLE の アンカー 指定に応じて、基準位置が変わります。

```
POSITION = 32,{50} + 64;
```

座標は 割合指定 が可能です。

TEX_ID = テクスチャ ID

TEX_ID n = テクスチャ ID

レンダーテクスチャの ID を指定します。同じ ID を使うとレンダーテクスチャを共有します。共有したときは、レンダーテクスチャ内部をアイコンのサイズに分割して使用してくれます。

この値を、*TEXTURE_ZOFFSET = テクスチャ ID, Z オフセット* に指定することによって、レンダリング順序を制御することができます。

確保されるレンダーテクスチャのサイズは、CONTENTS_SIZE から取得します。

注釈: 必須のプロパティです。

注釈: 共有するのは同一ウィンドウ内においての話です。ウィンドウを超えて共有はしません。

注釈: コントロール毎に、レンダリング対象のテクスチャ ID を一つだけ指定可能です。

TEX_ID n = テクスチャ ID, パーツ ID

TEX_ID n = パーツ ID

テクスチャ ID とパーツ ID を指定します。

n = [0..7] で指定可能です。

n = 0 の時は、TEX_ID と同じテクスチャを操作します。

パーツ ID を指定したときは、レンダリング対象にならず、通常のテクスチャとしてレンダリングされます。

CONTENTS_SIZE = レンダーテクスチャサイズ, レンダーテクスチャサイズ

レンダーテクスチャのサイズを指定します。共有するアイコン間では、共通の値を指定するようにしてください。

違う値を設定したとき、動作の保証はありません。

注釈: 必須のプロパティです。

共有するレンダーテクスチャ間で同じ値を設定するようにしてください。

正方形で指定し、幅、高さ共に 2 のべき乗を指定するようにしてください。また、最大 2048 以内に収めておくと互換性の問題が出ずらいです。

注釈: 割合指定も可能ですが、互換性の問題もあり、直値を指定するようにしてください。

TEXTURE_OFFSET n = オフセット X, オフセット Y

テクスチャの表示オフセットを指定します。

n = [0..7] で指定可能です。

テクスチャオフセットは 割合指定 が可能です。

SIZE = 横サイズ, 縦サイズ

表示サイズを変更します。

レンダーテクスチャから、どれ位の矩形を切り出すかを決定するために使われます。

共有しているアイコン間で違う値を設定するとエラーになります。

```
SIZE = 64,32; //64x32
```

サイズは 割合指定 が可能です。

注釈: 必須のプロパティです。

共有するレンダーテクスチャ間で同じ値を設定するようにしてください。

COLOR = R,G,B,A

カラーを指定します。

R,G,B については、0〜2 の間で指定してください。

1 を超えたとき、そのカラー成分を 2 倍まで上げて表示することができます。

A については、0〜1 の間で指定してください。

COLOR1 = R,G,B,A

レンダリングする際の背景のクリアカラーを指定します。

R,G,B,A については、0〜1 の間で指定してください。

SE_ID = SE_ID

押されたときに鳴らす音の ID を設定します。デフォルトは、0 になっており、音がなりません。

STYLE = フラグ 0|フラグ 1|..|フラグ n

コントロールの表示アンカーを指定できます。

表示位置アンカーフラグ	説明
ANCHOR_DEFAULT	アンカー位置を左上に設定 ANCHOR_LEFTTOP と同じ コントロールの中心位置はデフォルトで、BASE_LEFTTOP になる。
ANCHOR_LEFTTOP	アンカー位置を左上に設定 コントロールの中心位置はデフォルトで、BASE_LEFTTOP になる。
ANCHOR_LEFT	アンカー位置を左に設定 縦にはセンタリング コントロールの中心位置はデフォルトで、BASE_LEFT になる。
ANCHOR_LEFTBOTTOM	アンカー位置を左に設定 縦には下辺を基準に配置 コントロールの中心位置はデフォルトで、BASE_LEFTBOTTOM になる。
ANCHOR_TOP	アンカー位置を上辺に設定 横方向にはセンタリング コントロールの中心位置はデフォルトで、BASE_TOP になる。
ANCHOR_CENTER	アンカー位置を画面中央にセンタリング コントロールの中心位置はデフォルトで、BASE_CENTER になる。
ANCHOR_BOTTOM	アンカー位置を底辺に設定 横方向にはセンタリング コントロールの中心位置はデフォルトで、BASE_BOTTOM になる。
ANCHOR_RIGHTTOP	アンカー位置を右上に設定 コントロールの中心位置はデフォルトで、BASE_RIGHTBOTTOM になる。
ANCHOR_RIGHT	アンカー位置を右に設定 縦にはセンタリング コントロールの中心位置はデフォルトで、BASE_RIGHT になる。
ANCHOR_RIGHTBOTTOM	アンカー位置を右に設定 縦には下辺を基準に配置 コントロールの中心位置はデフォルトで、BASE_RIGHTBOTTOM になる。

コントロールの中心位置を指定できます。

中心位置変更フラグ	説明
BASE_DEFAULT	アンカーフラグに応じて変わる。 特に指定しなければ、これになります。 各アンカーフラグの説明を参照
BASE_LEFTTOP	中心位置をコントロールの左上に設定
BASE_LEFT	中心位置を左に設定 縦にはコントロールの真中
BASE_LEFTBOTTOM	中心位置を左下に設定
BASE_TOP	中心位置を上辺に設定 横にはコントロールの中心
BASE_CENTER	中心位置をコントロールの中心
BASE_BOTTOM	中心位置を底辺に設定 横方向にはコントロールの真中
BASE_RIGHTTOP	中心位置をコントロールの右上に設定
BASE_RIGHT	中心位置を右に設定 横にはコントロールの中心
BASE_RIGHTBOTTOM	中心位置を右下に設定

機能を制限するスタイルは以下のものがあります。

機能制限スタイル	説明
HIDE	表示を隠す。
DRAG	ドラッグ可能にする。
DISABLE	押せなくし、暗くする。
NOHIT	押せなくする。

11.20 METER

C#: *CWinCtrlMeter*

メータを表示するためのコントロールです。

プログレスバーなどに使えます。

最大 8 枚まで重ねて表示することが可能です。

任意のテクスチャパーツの伸縮を設定可能です。

長さを 1 に設定すると、ちょうど *TEXTURE* と同じ見た目で表示されるようになっています。

デフォルトでは、押したことを検出できません。コールバックが必要なときは、STYLE に HIT を指定して

ください。

METER(コントロール名) {

プロパティ 1;

プロパティ 2;

:

:

プロパティ n;

};

11.20.1 記述例

```
WINDOW(255_000_00001) {
  STYLE = NOTITLEBAR|ANCHOR_CENTER;
  POSITION = 0,100;
  TEX_ID = 100_000_00001;
  CAPTION = 000_000_00010;
  SIZE = 512,256;
};
METER(Progress) {
  ID = 000_001_00000;
  STYLE = ANCHOR_CENTER;
  SIZE = {100}-128;
  POSITION = 0,0;
  TEX_ID = 0,"MTR";
  COLOR = 1,1,1,1;
  TEX_ID1 = 0,"MTRB";
  COLOR1 = 1,1,1,1;
};
```

11.20.2 プロパティ

代表的なデフォルト値

```
TEX_ID0 = "MTR";
TEX_ID1 = "MTRB";
COLOR = 1,1,1,1;
CAPTION_COLOR = 1,1,1,1;
```

ID = コントロール ID

コントロール ID を定義します。

```
ID = 001_000_00010;
```

注釈: 設定しなかったときは、自動的にハッシュ値から生成します。

POSITION = X, Y

表示位置を決定します。STYLE の アンカー 指定に応じて、基準位置が変わります。

```
POSITION = 32,{50} + 64;
```

座標は 割合指定 が可能です。

SIZE = 横サイズ, 縦サイズ

表示サイズを変更します。省略したとき、若しくは、0 を指定すると、テクスチャパーツからサイズを取得してきます。

```
SIZE = 64,32; //64x32
SIZE = ,32;   //Set the width of the texture part width
SIZE = 64;            //Set the height of the texture part height
SIZE = {50} - 25;
```

サイズは 割合指定 が可能です。

TEX_ID = テクスチャ ID, パーツ ID

TEX_ID = パーツ ID

テクスチャ ID とパーツ ID を指定します。

TEX_ID n = テクスチャ ID, パーツ ID

TEX_ID n = パーツ ID

テクスチャ ID とパーツ ID を指定します。

n = [0..7] で指定可能です。

n = 0 の時は、TEX_ID と同じテクスチャを操作します。

TEXTURE_OFFSET n = オフセット X, オフセット Y

テクスチャの表示オフセットを指定します。

n = [0..7] で指定可能です。

テクスチャオフセットは 割合指定 が可能です。

TEXTURE_SIZE n = 横サイズ, 縦サイズ

テクスチャ ID とパーツ ID を指定します。

n = [0..7] で指定可能です。

n = 0 の時は、 **SIZE** と同じ意味です。

```
TEXTURE_SIZE1 = 64,32;         //64x32
TEXTURE_SIZE2 = ,32;  //Set the width of the texture part width
TEXTURE_SIZE3 = 64;   //Set the height of the texture part height
```

テクスチャサイズは 割合指定 が可能です。

COLOR = R,G,B,A

カラーを指定します。

R,G,B については、0〜2 の間で指定してください。

1 を超えたとき、そのカラー成分を 2 倍まで上げて表示することができます。

A については、0〜1 の間で指定してください。

COLOR n = R,G,B,A

カラーを指定します。

R,G,B については、0〜2 の間で指定してください。

1 を超えたとき、そのカラー成分を 2 倍まで上げて表示することができます。

A については、0〜1 の間で指定してください。

n = [0..7] で指定可能です。

n = 0 の時は、 **COLOR** と同じプロパティです。

STYLE = フラグ 0|フラグ 1|..|フラグ n

コントロールの表示アンカーを指定できます。

表示位置アンカーフラグ	説明
ANCHOR_DEFAULT	アンカー位置を左上に設定 ANCHOR_LEFTTOP と同じ コントロールの中心位置はデフォルトで、BASE_LEFTTOP になる。
ANCHOR_LEFTTOP	アンカー位置を左上に設定 コントロールの中心位置はデフォルトで、BASE_LEFTTOP になる。
ANCHOR_LEFT	アンカー位置を左に設定 縦にはセンタリング コントロールの中心位置はデフォルトで、BASE_LEFT になる。
ANCHOR_LEFTBOTTOM	アンカー位置を左に設定 縦には下辺を基準に配置 コントロールの中心位置はデフォルトで、BASE_LEFTBOTTOM になる。
ANCHOR_TOP	アンカー位置を上辺に設定 横方向にはセンタリング コントロールの中心位置はデフォルトで、BASE_TOP になる。
ANCHOR_CENTER	アンカー位置を画面中央にセンタリング コントロールの中心位置はデフォルトで、BASE_CENTER になる。
ANCHOR_BOTTOM	アンカー位置を底辺に設定 横方向にはセンタリング コントロールの中心位置はデフォルトで、BASE_BOTTOM になる。
ANCHOR_RIGHTTOP	アンカー位置を右上に設定 コントロールの中心位置はデフォルトで、BASE_RIGHTBOTTOM になる。
ANCHOR_RIGHT	アンカー位置を右に設定 縦にはセンタリング コントロールの中心位置はデフォルトで、BASE_RIGHT になる。
ANCHOR_RIGHTBOTTOM	アンカー位置を右に設定 縦には下辺を基準に配置 コントロールの中心位置はデフォルトで、BASE_RIGHTBOTTOM になる。

コントロールの中心位置を指定できます。

中心位置変更フラグ	説明
BASE_DEFAULT	アンカーフラグに応じて変わる。 特に指定しなければ、これになります。 各アンカーフラグの説明を参照
BASE_LEFTTOP	中心位置をコントロールの左上に設定
BASE_LEFT	中心位置を左に設定 縦にはコントロールの真中
BASE_LEFTBOTTOM	中心位置を左下に設定
BASE_TOP	中心位置を上辺に設定 横にはコントロールの中心
BASE_CENTER	中心位置をコントロールの中心
BASE_BOTTOM	中心位置を底辺に設定 横方向にはコントロールの真中
BASE_RIGHTTOP	中心位置をコントロールの右上に設定
BASE_RIGHT	中心位置を右に設定 横にはコントロールの中心
BASE_RIGHTBOTTOM	中心位置を右下に設定

機能を制限するスタイルは以下のものがあります。

機能制限スタイル	説明
HIDE	表示を隠す。
DRAG	ドラッグ可能にする。
DISABLE	押せなくし、暗くする。
HIT	押せるようにする。

11.21 SCROLLBAR

C#: *CWinCtrlScrollbar*

スクロールバーを表示するためのコントロールです。

スクロール可能なコントロールと関連付けることによって、有効になります。

GROUP プロパティに SCROLLBAR コントロールを関連づけることによって、スクロールバーを表示することができます。

スクロール可能なコントロールに対して、複数のスクロールバーを関連付けることも可能です。

スクロール可能なコントロールは次のものがあります。

- *LISTBOX*
- *LISTBOXEX*
- *EDITBOX*
- *TEXTBOX*
- *CONTAINER*
- *LOG*

STYLE に HIT/NOHIT 共に指定することができません。

SCROLLBAR(コントロール名) {

プロパティ 1;

プロパティ 2;

:

:

プロパティ n;

};

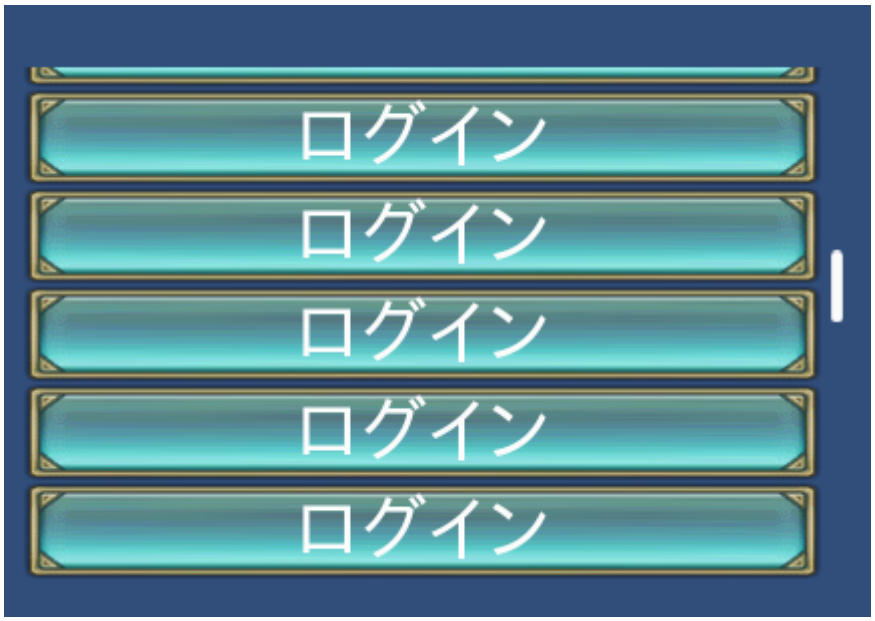
11.21.1 記述例

```
$w = 300;
LISTBOX(List) {
  ID = 001_100_00000;
  POSITION = 0,-160;
  STYLE = ANCHOR_BOTTOM;
  SIZE = $w,160 - 16;
  CONTENTS_SIZE = $w,48;
  CONTENTS = {
    CHECKBOX(IP) {
      ID = 001_000_00020;
      CAPTION = 001_000_00030;
      STYLE = ANCHOR_LEFTTOP;
      POSITION = 0,0;
      SIZE = $w,48;
    };
  }
  GROUP = SCROLLBAR(List);
};
SCROLLBAR(List) {
```

(次のページに続く)

(前のページからの続き)

```
  ID = 001_100_00010;
  DEF_SCROLLBAR;
  STYLE = ANCHOR_RIGHTTOP;
  POSITION = 0,16;
  SIZE = 0,-16 * 2;
};
```



11.21.2 位置とサイズ

通常のコントロールと違い、位置とサイズの扱いが特殊です。

位置とサイズは、関連付いているスクロール可能なコントロールからの相対になります。

また、アンカーによって、関連付いているスクロールのどの辺に沿って配置するからを指定できるようになっています。

有効なアンカーは こちら を参照ください。

```
#include "wr.h"

WINDOW(255_000_00001) {
       STYLE = NOTITLEBAR|ANCHOR_CENTER;
       POSITION = 0,0;
       SIZE = 256,256;
       TEX_ID = 100_000_00001;
};

CONTAINER(Map) {
```

(次のページに続く)

(前のページからの続き)

```
	ID = 000_000_000100;
	POSITION = 0,0;
	SIZE = RELATIVE_SIZE(0),RELATIVE_SIZE(0);
	CONTENTS_SIZE = 512,512;
	COLOR = COLOR32(255,255,255,255);
	CONTENTS = {
		BUTTON(A) {
			STYLE = BASE_LEFT;
			PRIORITY = -1;
			POSITION = 0,256;
			TEX_ID = "TQML0";
		};
		BUTTON(B) {
			STYLE = BASE_TOP;
			PRIORITY = -1;
			POSITION = 256,0;
			TEX_ID = "TQML0";
		};
		BUTTON(C) {
			STYLE = BASE_BOTTOM;
			PRIORITY = -1;
			POSITION = 256,512;
			TEX_ID = "TQML0";
		};
		BUTTON(D) {
			STYLE = BASE_RIGHT;
			PRIORITY = -1;
			POSITION = 512,256;
			TEX_ID = "TQML0";
		};
	};
	GROUP = SCROLLBAR(LEFT),SCROLLBAR(RIGHT),SCROLLBAR(TOP),SCROLLBAR(BOTTOM);
};
SCROLLBAR(LEFT) {
	STYLE = ANCHOR_LEFTTOP;
	TEX_ID = "SCBBH";
	POSITION = 0,0;
	SIZE = 0,0;
};
SCROLLBAR(RIGHT) {
	STYLE = ANCHOR_RIGHTTOP;
	TEX_ID = "SCBBH";
	POSITION = 0,0;
	SIZE = 0,0;
};
SCROLLBAR(TOP) {
	STYLE = ANCHOR_LEFTTOP|ITEM_STACK_H;
	TEX_ID = "SCBBV";
```

(次のページに続く)

(前のページからの続き)

```
        POSITION = 0,0;
        SIZE = 0,0;
};
SCROLLBAR(BOTTOM) {
        STYLE = ANCHOR_LEFTBOTTOM|ITEM_STACK_H;
        TEX_ID = "SCBBV";
        POSITION = 0,0;
        SIZE = 0,0;
};
```

11.21.3 スクロールバーの方向

縦方向のスクロールバーだけでなく横方向のスクロールバーも設定可能です。

スタイルに次の値を設定してください。

- 縦方向: ITEM_STACK_V
- 横方向: ITEM_STACK_H

11.21.4 スクロールバーの表示制御

スタイルに設定することによって、スクロールバーの表示/非表示のタイプを選択できます。

以下の３タイプから選べます。

SCROLLBAR_DISPLAY_NORMAL(デフォルト)

領域がスクロール可能で、かつその領域をタッチしてスクロールさせているときだけ表示します。

SCROLLBAR_DISPLAY_SCROLLABLE

領域がスクロール可能なときだけ表示します。

SCROLLBAR_DISPLAY_ALWAYS

常に表示します。

11.21.5 プロパティ

代表的なデフォルト値

```
TEX_ID = "SCBRH";
STYLE = ITEM_STACK_V;
COLOR = 1,1,1,1;
```

ID = コントロール ID

コントロール ID を定義します。

```
ID = 001_000_00010;
```

注釈: 設定しなかったときは、自動的にハッシュ値から生成します。

POSITION = X, Y

表示位置を決定します。GROUP によって、関連付けされたコントロールからの相対位置になります。

```
POSITION = 0,16;
```

割合指定はできません。

TEX_ID = テクスチャ ID, パーツ ID

TEX_ID = パーツ ID

テクスチャ ID とパーツ ID を指定します。

SIZE = 横幅, 縦幅

スクロールバーの表示サイズを指定します。GROUP によって、関連付けされたコントロールサイズとの相対サイズになります。

```
SIZE = 0,0;                    //display size
```

割合指定はできません。

COLOR = R,G,B,A

カラーを指定します。カラー変更することによって、含まれるコントロール全てに影響を与えます。

R,G,B については、0〜2 の間で指定してください。

1 を超えたとき、そのカラー成分を 2 倍まで上げて表示することができます。

A については、0〜1 の間で指定してください。

STYLE = フラグ 0|フラグ 1|..|フラグ n

コントロールの表示アンカーを指定できます。

表示位置アンカーフラグ	説明
ANCHOR_DEFAULT	アンカー位置を左上に設定 ANCHOR_LEFTTOP と同じ コントロールの中心位置はデフォルトで、BASE_LEFTTOP になる。
ANCHOR_LEFTTOP	アンカー位置を左上に設定 コントロールの中心位置はデフォルトで、BASE_LEFTTOP になる。
ANCHOR_LEFTBOTTOM	アンカー位置を左に設定 縦には下辺を基準に配置 コントロールの中心位置はデフォルトで、BASE_LEFTBOTTOM になる。
ANCHOR_RIGHTTOP	アンカー位置を右上に設定 コントロールの中心位置はデフォルトで、BASE_RIGHTBOTTOM になる。
ANCHOR_RIGHTBOTTOM	アンカー位置を右に設定 縦には下辺を基準に配置 コントロールの中心位置はデフォルトで、BASE_RIGHTBOTTOM になる。

実際の配置は下図を参照ください。

機能を制限するスタイルは以下のものがあります。

機能制限スタイル	説明
HIDE	表示を隠す。
ITEM_STACK_V	スクロールバーの方向を縦方向に設定する
ITEM_STACK_H	スクロールバーの方向を横方向に設定する
SCROLLBAR_DISPLAY_NORMAL	領域がスクロール可能で、かつその領域をタッチしてスクロールさせているときだけ表示します。
SCROLLBAR_DISPLAY_SCROLLABLE	領域がスクロール可能なときだけ表示します。
SCROLLBAR_DISPLAY_ALWAYS	常に表示します。

11.22 LISTBOX

C#: *CWinCtrlListbox*

リストボックスのためのコントロールです。

コンテンツを複製し、それを一行とします。

コンテンツの数はメモリが許す限り自由に設定可能です。

垂直方向にコンテンツが並んでいるリストボックスだけでなく、水平方向にも並べることも可能です。

このコントロールはスクロール可能なコントロールです。

STYLE に *SCROLL_LOCK* をつけておくと、onDrag コールバックが呼び出され、ドラッグした方向を検出できます。DRAG をつけても、これらコントロールには効果がないことに気を付けてください。

LISTBOX(コントロール名) {

プロパティ 1;

プロパティ 2;

:

:

プロパティ n;

};

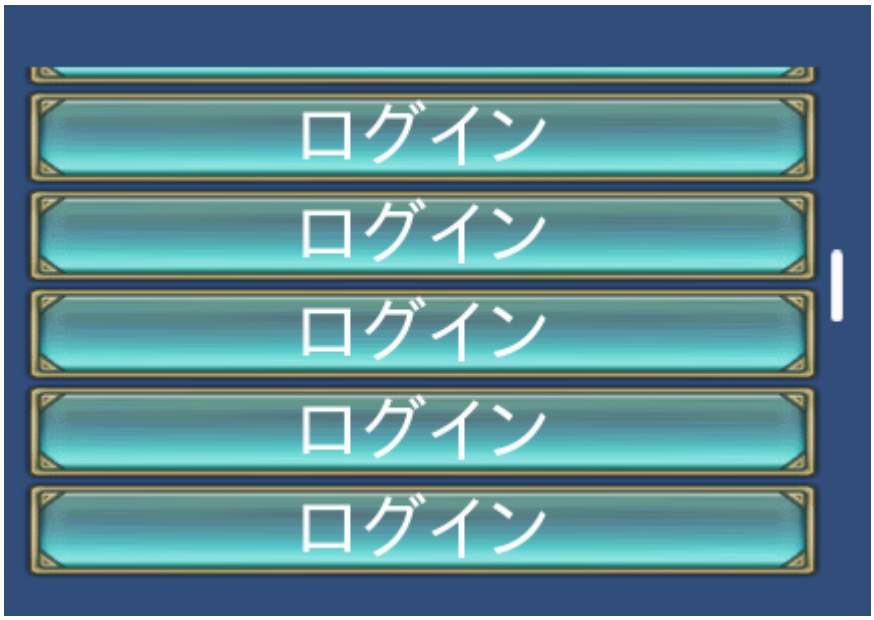
11.22.1 記述例

```
WINDOW(255_000_00001) {
  STYLE = NOTITLEBAR|NOFRAME|ANCHOR_CENTER;
  POSITION = 0,100;
  SIZE = 512,256;
  };
$w = 300;
LISTBOX(List) {
  ID = 001_100_00000;
  POSITION = 0,-160;
  STYLE = ANCHOR_BOTTOM;
  SIZE = $w,160 - 16;
  CONTENTS_SIZE = $w,48;
  CONTENTS = {
    CHECKBOX(IP) {
      ID = 001_000_00020;
      CAPTION = 001_000_00030;
      STYLE = ANCHOR_LEFTTOP;
      POSITION = 0,0;
      SIZE = $w,48;
    };
  }
  GROUP = SCROLLBAR(List);
};
SCROLLBAR(List) {
  ID = 001_100_00010;
  DEF_SCROLLBAR;
  STYLE = ANCHOR_RIGHTTOP;
  POSITION = 0,16;
```

(次のページに続く)

(前のページからの続き)

```
  SIZE = 0,-16 * 2;
};
```



11.22.2 プロパティ

代表的なデフォルト値

```
STYLE = ITEM_STACK_V; //Arranged in the vertical direction
COLOR = 1,1,1,1;
CAPTION_COLOR = 1,1,1,1;
SE_ID = scrollSE;
```

ID = コントロール ID

コントロール ID を定義します。

```
ID = 001_000_00010;
```

注釈: 設定しなかったときは、自動的にハッシュ値から生成します。

POSITION = X, Y

表示位置を決定します。STYLE の アンカー 指定に応じて、基準位置が変わります。

```
POSITION = 32,{50} + 64;
```

座標は 割合指定 が可能です。

SIZE = 横サイズ, 縦サイズ

表示サイズを変更します。省略したとき、若しくは、0 を指定すると、テクスチャパーツからサイズを取得してきます。

```
SIZE = 512,512;        //512x512
SIZE = {50} - 100,{50} + 100; //screen width * 0.5 - 100,screen height * 0.5 + 100
```

サイズは 割合指定 が可能です。

CONTENTS_SIZE = 横サイズ, 縦サイズ

一行のサイズを決定します。垂直方向に並んでいるリストボックスにおいては、縦方向にスクロールします。

ただし、コンテンツサイズの横幅のほうが表示エリアよりも大きいときは、左右にもスクロールします。

```
// only vertical direction to scroll
SIZE = 400,400;                //display size
CONTENTS_SIZE = 400,40;        //virtual screen size

// Scroll to the left and right direction
SIZE = 400,400;                //display size
CONTENTS_SIZE = 800,40;        //virtual screen size
```

コンテンツサイズは 割合指定 が可能です。

CONTENTS = { コントロール定義 … }

LISTBOX の中の一行を定義するためのコントロールを列挙します。

コントロールの座標は、LISTBOX の座標から相対で配置されます。

全てのコントロールを関連付けることが可能です。

```
// It is also possible to put the list box in the list box
CONTENTS = {
      BUTTON(A) {
          :
      };
      CHECKBOX(B) {
```

(次のページに続く)

(前のページからの続き)

```
        :
    };
    LISTBOX(C) {
        :
    };
```

C#からコンテンツリストの特定のコントロールに対してアクセスする方法 は こちら を参考にしてください。

注釈: コンテンツ内のコントロールの座標やサイズの 割合指定 は、リストボックスのサイズを基準に計算されます。

ウィンドウサイズではないことに気を付けてください。

LINE_SPACE = 行間ピクセル値

コンテンツとコンテンツ間の隙間をこのプロパティで設定可能です。

```
LINE_SPACE = 8;        //Put an 8-dot space.
```

割合指定はできません。

GROUP = SCROLLBAR コントロール ID,…

SCROLLBAR コントロールを GROUP として定義しておくと、自動的にリストボックスと連動してスクロールバーを表示します。

CONTENTS に入れないように気を付けてください。

複数のスクロールバーを関連付けることも可能です。

```
GROUP = SCROLLBAR(Horizon),SCROLLBAR(Vertical);
```

COLOR = R,G,B,A

カラーを指定します。カラー変更することによって、含まれるコントロール全てに影響を与えます。

R,G,B については、0〜2 の間で指定してください。

1 を超えたとき、そのカラー成分を 2 倍まで上げて表示することができます。

A については、0〜1 の間で指定してください。

SE_ID = SE_ID

スクロールし、コンテンツの先頭が変わった時鳴らす音を設定します。デフォルトは、scrollSE です。0 を指定すると音が鳴らなくなります。

STYLE = フラグ 0|フラグ 1|..|フラグ n

リストボックス固有のスタイルは以下のものがあります。

リストボックス制御フラグ	説明
ITEM_STACK_V	垂直方向に並んでいるリストボックス
ITEM_STACK_H	水平方向に並んでいるリストボックス
NOBOUNCES	端までスクロールしたとき、勢いが残っていても跳ね返らない
SCROLL_UNLOCK	スクロールを許可する。
SCROLL_LOCK	スクロールを抑制する。

コントロールの表示アンカーを指定できます。

表示位置アンカーフラグ	説明
ANCHOR_DEFAULT	アンカー位置を左上に設定 ANCHOR_LEFTTOP と同じ コントロールの中心位置はデフォルトで、BASE_LEFTTOP になる。
ANCHOR_LEFTTOP	アンカー位置を左上に設定 コントロールの中心位置はデフォルトで、BASE_LEFTTOP になる。
ANCHOR_LEFT	アンカー位置を左に設定 縦にはセンタリング コントロールの中心位置はデフォルトで、BASE_LEFT になる。
ANCHOR_LEFTBOTTOM	アンカー位置を左に設定 縦には下辺を基準に配置 コントロールの中心位置はデフォルトで、BASE_LEFTBOTTOM になる。
ANCHOR_TOP	アンカー位置を上辺に設定 横方向にはセンタリング コントロールの中心位置はデフォルトで、BASE_TOP になる。
ANCHOR_CENTER	アンカー位置を画面中央にセンタリング コントロールの中心位置はデフォルトで、BASE_CENTER になる。
ANCHOR_BOTTOM	アンカー位置を底辺に設定 横方向にはセンタリング コントロールの中心位置はデフォルトで、BASE_BOTTOM になる。
ANCHOR_RIGHTTOP	アンカー位置を右上に設定 コントロールの中心位置はデフォルトで、BASE_RIGHTBOTTOM になる。
ANCHOR_RIGHT	アンカー位置を右に設定 縦にはセンタリング コントロールの中心位置はデフォルトで、BASE_RIGHT になる。
ANCHOR_RIGHTBOTTOM	アンカー位置を右に設定 縦には下辺を基準に配置 コントロールの中心位置はデフォルトで、BASE_RIGHTBOTTOM になる。

コントロールの中心位置を指定できます。

中心位置変更フラグ	説明
BASE_DEFAULT	アンカーフラグに応じて変わる。 特に指定しなければ、これになります。 各アンカーフラグの説明を参照
BASE_LEFTTOP	中心位置をコントロールの左上に設定
BASE_LEFT	中心位置を左に設定 縦にはコントロールの真中
BASE_LEFTBOTTOM	中心位置を左下に設定
BASE_TOP	中心位置を上辺に設定 横にはコントロールの中心
BASE_CENTER	中心位置をコントロールの中心
BASE_BOTTOM	中心位置を底辺に設定 横方向にはコントロールの真中
BASE_RIGHTTOP	中心位置をコントロールの右上に設定
BASE_RIGHT	中心位置を右に設定 横にはコントロールの中心
BASE_RIGHTBOTTOM	中心位置を右下に設定

機能を制限するスタイルは以下のものがあります。

機能制限スタイル	説明
HIDE	表示を隠す。
DISABLE	押せなくし、暗くする。
NOHIT	押せなくする。
ITEM_STACK_V	垂直方向に並んでいるリストボックス
ITEM_STACK_H	水平方向に並んでいるリストボックス
NOBOUNCES	端までスクロールしたとき、勢いが残っていても跳ね返らない
SCROLL_UNLOCK	スクロールを許可する。
SCROLL_LOCK	スクロールを抑制する。

11.23 LISTBOXEX

C#: *CWinCtrlListboxEx*

リストボックスのためのコントロールです。

コンテンツを複製し、それを一行とします。

コンテンツの数はメモリが許す限り自由に設定可能です。

垂直方向にコンテンツが並んでいるリストボックスだけでなく、水平方向にも並べることも可能です。

LISTBOX と違う点は、コンテンツサイズを自動的に計測するところです。

垂直方向に並んでいる場合は、縦幅のみ自動計算の対象になります。

それに対して水平方向に並んでいる場合は、横幅のみ自動計算の対象になります。

このコントロールはスクロール可能なコントロールです。

STYLE に *SCROLL_LOCK* をつけておくと、onDrag コールバックが呼び出され、ドラッグした方向を検出できます。DRAG をつけても、これらコントロールには効果がないことに気を付けてください。

LISTBOXEX(コントロール名) {

プロパティ 1;

プロパティ 2;

:

:

プロパティ n;

};

11.23.1 記述例

```
WINDOW(255_000_00001) {
  STYLE = NOTITLEBAR|NOFRAME|ANCHOR_CENTER;
  POSITION = 0,100;
  SIZE = 512,256;
};

$w = 400;
LISTBOXEX(List) {
  ID = 001_100_00000;
  POSITION = 0,0;
  STYLE = ANCHOR_BOTTOM;
  SIZE = $w,{100} - 16;
  CONTENTS = {
    CHECKBOX(IP) {
      ID = 001_000_00020;
      CAPTION = 001_000_00030;
      STYLE = ANCHOR_LEFTTOP;
      POSITION = 0,0;
      SIZE = $w,48;
    };
```

(次のページに続く)

(前のページからの続き)

```
  };
  LINE_SPACE = 4;
  GROUP = SCROLLBAR(List);
};
SCROLLBAR(List) {
  ID = 001_100_00010;
  STYLE = ANCHOR_LEFTTOP;
  POSITION = 4,16;
  SIZE = 0,-16 * 2;
};
```

11.23.2 プロパティ

代表的なデフォルト値

```
STYLE = ITEM_STACK_V; //Arranged in the vertical direction
COLOR = 1,1,1,1;
CAPTION_COLOR = 1,1,1,1;
SE_ID = scrollSE;
```

ID = コントロール ID

コントロール ID を定義します。

```
ID = 001_000_00010;
```

注釈: 設定しなかったときは、自動的にハッシュ値から生成します。

POSITION = X, Y

表示位置を決定します。STYLE の アンカー 指定に応じて、基準位置が変わります。

```
POSITION = 32,{50} + 64;
```

座標は 割合指定 が可能です。

SIZE = 横サイズ, 縦サイズ

表示サイズを変更します。省略したとき、若しくは、0 を指定すると、テクスチャパーツからサイズを取得してきます。

```
SIZE = 512,512;        //512x512
SIZE = {50} - 100,{50} + 100; //screen width * 0.5 - 100,screen height * 0.5 + 100
```

サイズは 割合指定 が可能です。

CONTENTS_SIZE = 横幅, 縦幅

一行のサイズを決定します。

垂直方向に並んでいる場合は、縦幅は自動計算の対象になるので横幅のみ反映対象になります。

それに対して水平方向に並んでいる場合は、横幅は自動計算の対象になるので縦幅のみ反映対象になります。

その他は LISTBOX と同じ振る舞いをします。

```
// only vertical direction to scroll
SIZE = 400,400;               //display size
CONTENTS_SIZE = 400,40;       //virtual screen size

// Scroll to the left and right direction
SIZE = 400,400;               //display size
CONTENTS_SIZE = 800,40;       //virtual screen size
```

コンテンツサイズは 割合指定 が可能です。

CONTENTS = { コントロール定義; … }

LISTBOX の中の一行を定義するためのコンテンツを列挙します。

コントロールの座標は、LISTBOX の座標から相対で配置されます。

全てのコントロールを関連付けることが可能です。

```
// It is also possible to put the list box in the list box
CONTENTS = {
      BUTTON(A) {
          :
      };
      CHECKBOX(B) {
          :
      };
      LISTBOX(C) {
          :
      };
```

C#からコンテンツリストの特定のコントロールに対してアクセスする方法 は こちら を参考にしてください。

注釈: コンテンツ内のコントロールの座標やサイズの 割合指定 は、リストボックスのサイズを基準に計算されます。

ウィンドウサイズではないことに気を付けてください。

LINE_SPACE = 行間ピクセル値

コンテンツとコンテンツ間の隙間をこのプロパティで設定可能です。

```
LINE_SPACE = 8;        //Put an 8-dot space.
```

割合指定はできません。

GROUP = SCROLLBAR コントロール ID,…

SCROLLBAR コントロールを GROUP として定義しておくと、自動的にリストボックスと連動してスクロールバーを表示します。

CONTENTS に入れないように気を付けてください。

複数のスクロールバーを関連付けることも可能です。

```
GROUP = SCROLLBAR(Horizon),SCROLLBAR(Vertical);
```

COLOR = R,G,B,A

カラーを指定します。カラー変更することによって、含まれるコントロール全てに影響を与えます。

R,G,B については、0〜2 の間で指定してください。

1 を超えたとき、そのカラー成分を 2 倍まで上げて表示することができます。

A については、0〜1 の間で指定してください。

SE_ID = SE_ID

スクロールし、コンテンツの先頭が変わった時鳴らす音を設定します。デフォルトは、scrollSE です。0 を指定すると音が鳴らなくなります。

STYLE = フラグ 0|フラグ 1|..|フラグ n

リストボックス固有のスタイルは以下のものがあります。

リストボックス制御フラグ	説明
ITEM_STACK_V	垂直方向に並んでいるリストボックス
ITEM_STACK_H	水平方向に並んでいるリストボックス
NOBOUNCES	端までスクロールしたとき、勢いが残っていても跳ね返らない
SCROLL_UNLOCK	スクロールを許可する。
SCROLL_LOCK	スクロールを抑制する。

コントロールの表示アンカーを指定できます。

表示位置アンカーフラグ	説明
ANCHOR_DEFAULT	アンカー位置を左上に設定 ANCHOR_LEFTTOP と同じ コントロールの中心位置はデフォルトで、BASE_LEFTTOP になる。
ANCHOR_LEFTTOP	アンカー位置を左上に設定 コントロールの中心位置はデフォルトで、BASE_LEFTTOP になる。
ANCHOR_LEFT	アンカー位置を左に設定 縦にはセンタリング コントロールの中心位置はデフォルトで、BASE_LEFT になる。
ANCHOR_LEFTBOTTOM	アンカー位置を左に設定 縦には下辺を基準に配置 コントロールの中心位置はデフォルトで、BASE_LEFTBOTTOM になる。
ANCHOR_TOP	アンカー位置を上辺に設定 横方向にはセンタリング コントロールの中心位置はデフォルトで、BASE_TOP になる。
ANCHOR_CENTER	アンカー位置を画面中央にセンタリング コントロールの中心位置はデフォルトで、BASE_CENTER になる。
ANCHOR_BOTTOM	アンカー位置を底辺に設定 横方向にはセンタリング コントロールの中心位置はデフォルトで、BASE_BOTTOM になる。
ANCHOR_RIGHTTOP	アンカー位置を右上に設定 コントロールの中心位置はデフォルトで、BASE_RIGHTBOTTOM になる。
ANCHOR_RIGHT	アンカー位置を右に設定 縦にはセンタリング コントロールの中心位置はデフォルトで、BASE_RIGHT になる。
ANCHOR_RIGHTBOTTOM	アンカー位置を右に設定 縦には下辺を基準に配置 コントロールの中心位置はデフォルトで、BASE_RIGHTBOTTOM になる。

コントロールの中心位置を指定できます。

中心位置変更フラグ	説明
BASE_DEFAULT	アンカーフラグに応じて変わる。 特に指定しなければ、これになります。 各アンカーフラグの説明を参照
BASE_LEFTTOP	中心位置をコントロールの左上に設定
BASE_LEFT	中心位置を左に設定 縦にはコントロールの真中
BASE_LEFTBOTTOM	中心位置を左下に設定
BASE_TOP	中心位置を上辺に設定 横にはコントロールの中心
BASE_CENTER	中心位置をコントロールの中心
BASE_BOTTOM	中心位置を底辺に設定 横方向にはコントロールの真中
BASE_RIGHTTOP	中心位置をコントロールの右上に設定
BASE_RIGHT	中心位置を右に設定 横にはコントロールの中心
BASE_RIGHTBOTTOM	中心位置を右下に設定

機能を制限するスタイルは以下のものがあります。

機能制限スタイル	説明
HIDE	表示を隠す。
DISABLE	押せなくし、暗くする。
NOHIT	押せなくする。
ITEM_STACK_V	垂直方向に並んでいるリストボックス
ITEM_STACK_H	水平方向に並んでいるリストボックス
NOBOUNCES	端までスクロールしたとき、勢いが残っていても跳ね返らない
SCROLL_UNLOCK	スクロールを許可する。
SCROLL_LOCK	スクロールを抑制する。

11.24 CONTAINER

C#: *CWinCtrlContainer*

複数のコントロールを纏めるためのコントロールです。

SIZE によって、コンテナのサイズが決定し、CONTENTS_SIZE によって仮想画面のサイズが決定します。

仮想画面サイズがサイズより大きいときは、内包しているコントロールをスクロールさせることができます。

このコントロールはスクロール可能なコントロールです。

STYLE に *SCROLL_LOCK* をつけておくと、onDrag コールバックが呼び出され、ドラッグした方向を検出できます。DRAG をつけても、これらコントロールには効果がないことに気を付けてください。

CONTAINER(コントロール名) {

プロパティ 1;

プロパティ 2;

:

:

プロパティ n;

};

11.24.1 記述例

```
WINDOW(255_000_00001) {
  STYLE = NOTITLEBAR|ANCHOR_CENTER;
  POSITION = 0,100;
  SIZE = 512,256;
  TEX_ID = 100_010_00000;
};

$w = 400;
CONTAINER(Map) {
  ID = 000_000_000100;
  POSITION = 0,0;
  SIZE = {100},{100};
  CONTENTS_SIZE = 640,1136;
  COLOR = COLOR32(255,255,255,255);
  CONTENTS = {
    TEXTURE(Map) {
      ID = 000_000_000110;
      SIZE = 640,1136;
      PRIORITY = -1;
      POSITION = 0,0;
      TEX_ID = 0,"MAP2";
    };
  };
};
```

11.24.2 プロパティ

代表的なデフォルト値

```
COLOR = 1,1,1,1;
CAPTION_COLOR = 1,1,1,1;
```

ID = コントロール ID

コントロール ID を定義します。

```
ID = 001_000_00010;
```

注釈: 設定しなかったときは、自動的にハッシュ値から生成します。

POSITION = X, Y

表示位置を決定します。STYLE の アンカー 指定に応じて、基準位置が変わります。

```
POSITION = 32,{50} + 64;
```

座標は 割合指定 が可能です。

SIZE = 横サイズ, 縦サイズ

表示サイズを変更します。省略したとき、若しくは、0 を指定すると、テクスチャパーツからサイズを取得してきます。

```
SIZE = 512,512;        //512x512
SIZE = {50} - 100,{50} + 100; //screen width * 0.5 - 100,screen height * 0.5 + 100
```

サイズは 割合指定 が可能です。

CONTENTS_SIZE = 横幅, 縦幅

仮想画面のサイズを指定します。

仮想画面が、表示サイズより大きいときは、コンテナ内のコンテンツをスクロールさせることができます。

```
SIZE = 400,400;                //display size
CONTENTS_SIZE = 800,800;       //virtual screen size
```

コンテンツサイズは 割合指定 が可能です。

CONTENTS = { コントロール定義 … }

CONTAINER の中に内包するコントロールを列挙します。

コントロールの座標は、CONTAINER の座標から相対で配置されます。

全てのコントロールを関連付けることが可能です。

```
CONTENTS = {
   BUTTON(A) {
      :
   };
   CHECKBOX(B) {
      :
   };
   LISTBOX(C) {
      :
   };
};
```

C#からコンテンツに対してアクセスする方法は こちら を参考にしてください。

注釈: コンテンツ内のコントロールの座標やサイズの 割合指定 は、コンテナのサイズを基準に計算されます。

ウィンドウサイズではないことに気を付けてください。

COLOR = R,G,B,A

カラーを指定します。カラー変更することによって、含まれるコントロール全てに影響を与えます。

R,G,B については、0〜2 の間で指定してください。

1 を超えたとき、そのカラー成分を 2 倍まで上げて表示することができます。

A については、0〜1 の間で指定してください。

GROUP = SCROLLBAR コントロール ID,…

SCROLLBAR コントロールを GROUP として定義しておくと、自動的にコンテナと連動してスクロールバーを表示します。

CONTENTS に入れないように気を付けてください。

複数のスクロールバーを関連付けることも可能です。

```
GROUP = SCROLLBAR(Horizon),SCROLLBAR(Vertical);
```

STYLE = フラグ 0|フラグ 1|..|フラグ n

コンテナ固有のスタイルは以下のものがあります。

コンテナ制御フラグ	説明
NOBOUNCES	端までスクロールしたとき、勢いが残っていても跳ね返らない
SCROLL_UNLOCK	スクロールを許可する。
SCROLL_LOCK	スクロールを抑制する。

コントロールの表示アンカーを指定できます。

表示位置アンカーフラグ	説明
ANCHOR_DEFAULT	アンカー位置を左上に設定 ANCHOR_LEFTTOP と同じ コントロールの中心位置はデフォルトで、BASE_LEFTTOP になる。
ANCHOR_LEFTTOP	アンカー位置を左上に設定 コントロールの中心位置はデフォルトで、BASE_LEFTTOP になる。
ANCHOR_LEFT	アンカー位置を左に設定 縦にはセンタリング コントロールの中心位置はデフォルトで、BASE_LEFT になる。
ANCHOR_LEFTBOTTOM	アンカー位置を左に設定 縦には下辺を基準に配置 コントロールの中心位置はデフォルトで、BASE_LEFTBOTTOM になる。
ANCHOR_TOP	アンカー位置を上辺に設定 横方向にはセンタリング コントロールの中心位置はデフォルトで、BASE_TOP になる。
ANCHOR_CENTER	アンカー位置を画面中央にセンタリング コントロールの中心位置はデフォルトで、BASE_CENTER になる。
ANCHOR_BOTTOM	アンカー位置を底辺に設定 横方向にはセンタリング コントロールの中心位置はデフォルトで、BASE_BOTTOM になる。
ANCHOR_RIGHTTOP	アンカー位置を右上に設定 コントロールの中心位置はデフォルトで、BASE_RIGHTBOTTOM になる。
ANCHOR_RIGHT	アンカー位置を右に設定 縦にはセンタリング コントロールの中心位置はデフォルトで、BASE_RIGHT になる。
ANCHOR_RIGHTBOTTOM	アンカー位置を右に設定 縦には下辺を基準に配置 コントロールの中心位置はデフォルトで、BASE_RIGHTBOTTOM になる。

コントロールの中心位置を指定できます。

中心位置変更フラグ	説明
BASE_DEFAULT	アンカーフラグに応じて変わる。 特に指定しなければ、これになります。 各アンカーフラグの説明を参照
BASE_LEFTTOP	中心位置をコントロールの左上に設定
BASE_LEFT	中心位置を左に設定 縦にはコントロールの真中
BASE_LEFTBOTTOM	中心位置を左下に設定
BASE_TOP	中心位置を上辺に設定 横にはコントロールの中心
BASE_CENTER	中心位置をコントロールの中心
BASE_BOTTOM	中心位置を底辺に設定 横方向にはコントロールの真中
BASE_RIGHTTOP	中心位置をコントロールの右上に設定
BASE_RIGHT	中心位置を右に設定 横にはコントロールの中心
BASE_RIGHTBOTTOM	中心位置を右下に設定

機能を制限するスタイルは以下のものがあります。

機能制限スタイル	説明
HIDE	表示を隠す。
DISABLE	押せなくし、暗くする。
NOHIT	押せなくする。
NOBOUNCES	端までスクロールしたとき、勢いが残っていても跳ね返らない
SCROLL_UNLOCK	スクロールを許可する。
SCROLL_LOCK	スクロールを抑制する。

11.25 FRAME

C#: *CWinCtrlFrame*

フレームを表示するためのコントロールです。

BAR と違いキャプションが持てないです。

また、onClick イベントを受け取ることが可能です。

FRAME(コントロール名) {

プロパティ 1;

プロパティ 2;

:

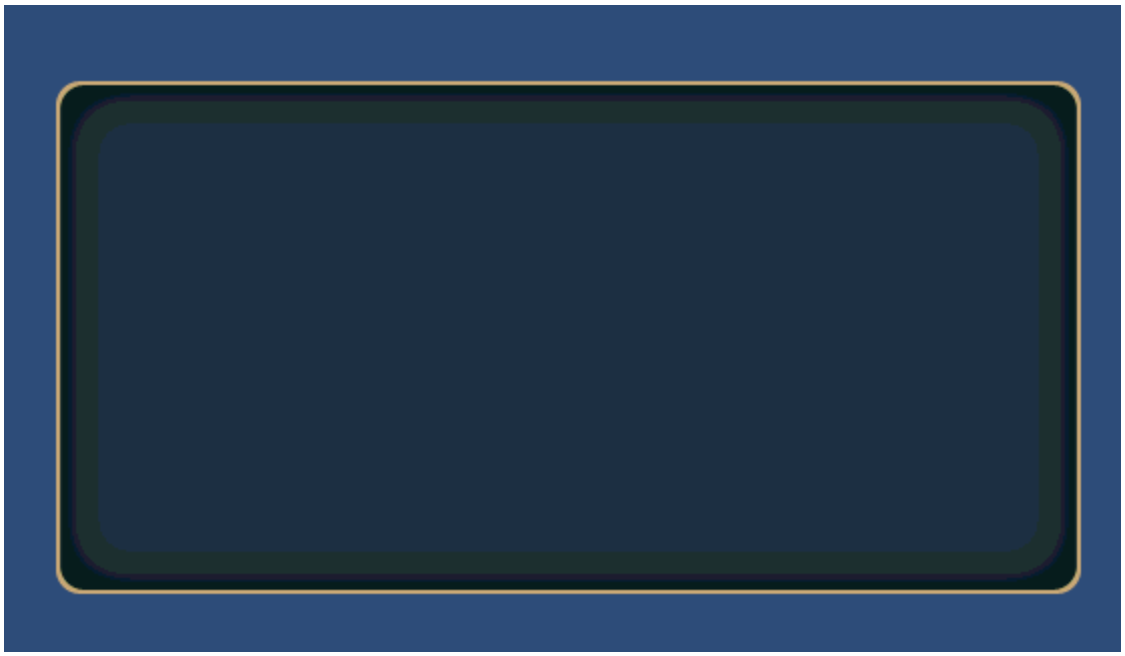
:

プロパティ n;

};

11.25.1 記述例

```
WINDOW(255_000_00001) {
  STYLE = NOTITLEBAR|NOFRAME|ANCHOR_CENTER;
  POSITION = 0,100;
  SIZE = 512,256;
};
FRAME(Test) {
  ID = 000_000_00010;
  POSITION = 0,0;
  SIZE = RELATIVE_SIZE(0),RELATIVE_SIZE(0);
};
```



11.25.2 プロパティ

代表的なデフォルト値

```
TEX_ID = "FRAME";
COLOR = 1,1,1,1;
CAPTION_COLOR = 1,1,1,1;
SE_ID = 0;
```

ID = コントロール ID

コントロール ID を定義します。

```
ID = 001_000_00010;
```

注釈: 設定しなかったときは、自動的にハッシュ値から生成します。

POSITION = X, Y

表示位置を決定します。STYLE の アンカー 指定に応じて、基準位置が変わります。

```
POSITION = 32,{50} + 64;
```

座標は 割合指定 が可能です。

SIZE = 横サイズ, 縦サイズ

表示サイズを変更します。省略したとき、若しくは、0 を指定すると、テクスチャパーツからサイズを取得してきます。

```
SIZE = 64,32; //64x32
SIZE = ,32;   //Set the width of the texture part width
SIZE = 64;            //Set the height of the texture part height
SIZE = {50} - 25;
```

サイズは 割合指定 が可能です。

TEX_ID = テクスチャ ID, パーツ ID

TEX_ID = パーツ ID

テクスチャ ID とパーツ ID を指定します。

COLOR = R,G,B,A

カラーを指定します。

R,G,B については、0〜2 の間で指定してください。

1 を超えたとき、そのカラー成分を 2 倍まで上げて表示することができます。

A については、0〜1 の間で指定してください。

SE_ID = SE_ID

押されたときに鳴らす音の ID を設定します。デフォルトは、0 になっており、音がなりません。

STYLE = フラグ 0|フラグ 1|..|フラグ n

コントロールの表示アンカーを指定できます。

表示位置アンカーフラグ	説明
ANCHOR_DEFAULT	アンカー位置を左上に設定 ANCHOR_LEFTTOP と同じ コントロールの中心位置はデフォルトで、BASE_LEFTTOP になる。
ANCHOR_LEFTTOP	アンカー位置を左上に設定 コントロールの中心位置はデフォルトで、BASE_LEFTTOP になる。
ANCHOR_LEFT	アンカー位置を左に設定 縦にはセンタリング コントロールの中心位置はデフォルトで、BASE_LEFT になる。
ANCHOR_LEFTBOTTOM	アンカー位置を左に設定 縦には下辺を基準に配置 コントロールの中心位置はデフォルトで、BASE_LEFTBOTTOM になる。
ANCHOR_TOP	アンカー位置を上辺に設定 横方向にはセンタリング コントロールの中心位置はデフォルトで、BASE_TOP になる。
ANCHOR_CENTER	アンカー位置を画面中央にセンタリング コントロールの中心位置はデフォルトで、BASE_CENTER になる。
ANCHOR_BOTTOM	アンカー位置を底辺に設定 横方向にはセンタリング コントロールの中心位置はデフォルトで、BASE_BOTTOM になる。
ANCHOR_RIGHTTOP	アンカー位置を右上に設定 コントロールの中心位置はデフォルトで、BASE_RIGHTBOTTOM になる。
ANCHOR_RIGHT	アンカー位置を右に設定 縦にはセンタリング コントロールの中心位置はデフォルトで、BASE_RIGHT になる。
ANCHOR_RIGHTBOTTOM	アンカー位置を右に設定 縦には下辺を基準に配置 コントロールの中心位置はデフォルトで、BASE_RIGHTBOTTOM になる。

コントロールの中心位置を指定できます。

中心位置変更フラグ	説明
BASE_DEFAULT	アンカーフラグに応じて変わる。 特に指定しなければ、これになります。 各アンカーフラグの説明を参照
BASE_LEFTTOP	中心位置をコントロールの左上に設定
BASE_LEFT	中心位置を左に設定 縦にはコントロールの真中
BASE_LEFTBOTTOM	中心位置を左下に設定
BASE_TOP	中心位置を上辺に設定 横にはコントロールの中心
BASE_CENTER	中心位置をコントロールの中心
BASE_BOTTOM	中心位置を底辺に設定 横方向にはコントロールの真中
BASE_RIGHTTOP	中心位置をコントロールの右上に設定
BASE_RIGHT	中心位置を右に設定 横にはコントロールの中心
BASE_RIGHTBOTTOM	中心位置を右下に設定

機能を制限するスタイルは以下のものがあります。

機能制限スタイル	説明
HIDE	表示を隠す。
DRAG	ドラッグ可能にする。
DISABLE	押せなくし、暗くする。
NOHIT	押せなくする。

11.26 LABEL

C#: *CWinCtrlLabel*

ラベルを表示するためのコントロールです。

デフォルトでは、押したことを検出できません。コールバックが必要なときは、STYLE に HIT を指定してください。

```
LABEL(コントロール名) {
    プロパティ 1;
    プロパティ 2;
```

:

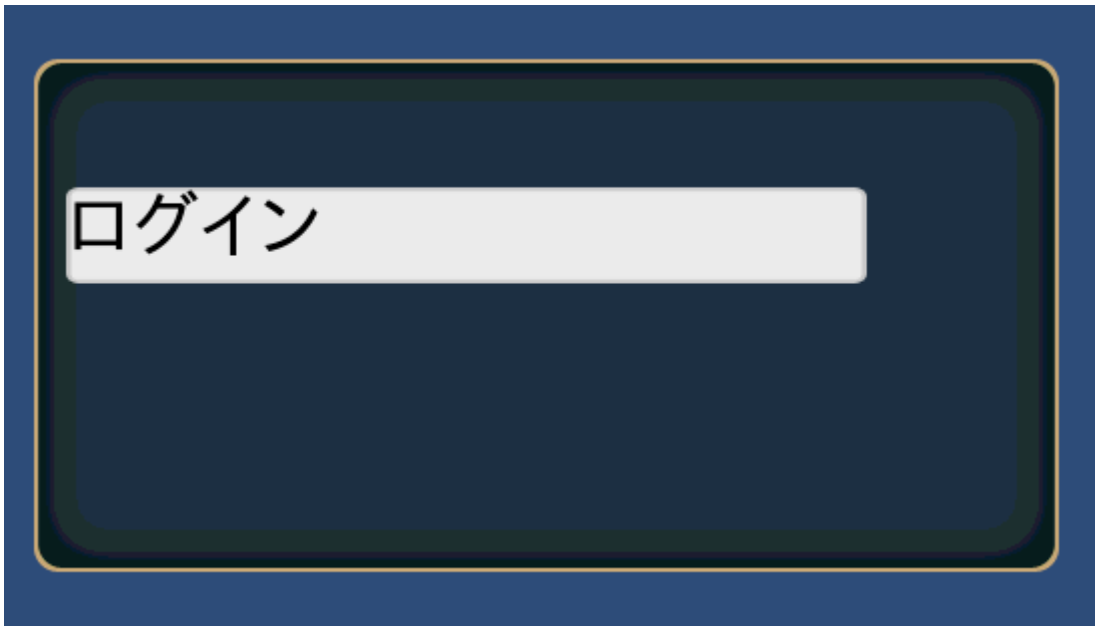
:

プロパティ n;

};

11.26.1 記述例

```
WINDOW(255_000_00001) {
  STYLE = NOTITLEBAR|ANCHOR_CENTER;
  POSITION = 0,100;
  SIZE = 512,256;
};
$w = 400;
LABEL(TEST) {
  ID = 001_000_00020;
  CAPTION = 001_000_00030;
  CAPTION_COLOR = 0,0,0,1;
  STYLE = ANCHOR_LEFTTOP;
  POSITION = 16,64;
  SIZE = $w,48;
};
```



11.26.2 プロパティ

代表的なデフォルト値

```
TEX_ID = "LABEL";
STYLE = TEXT_CENTER;
COLOR = 1,1,1,1;
CAPTION_COLOR = 1,1,1,1;
```

ID = コントロール ID

コントロール ID を定義します。

```
ID = 001_000_00010;
```

注釈: 設定しなかったときは、自動的にハッシュ値から生成します。

POSITION = X, Y

表示位置を決定します。STYLE の アンカー 指定に応じて、基準位置が変わります。

```
POSITION = 32,{50} + 64;
```

座標は 割合指定 が可能です。

SIZE = 横サイズ, 縦サイズ

表示サイズを変更します。省略したとき、若しくは、0 を指定すると、テクスチャパーツからサイズを取得してきます。

```
SIZE = 64,32; //64x32
SIZE = ,32;   //Set the width of the texture part width
SIZE = 64;            //Set the height of the texture part height
SIZE = {50} - 25;
```

サイズは 割合指定 が可能です。

TEX_ID = テクスチャ ID, パーツ ID

TEX_ID = パーツ ID

テクスチャ ID とパーツ ID を指定します。

COLOR = R,G,B,A

カラーを指定します。

R,G,B については、0〜2 の間で指定してください。

1 を超えたとき、そのカラー成分を 2 倍まで上げて表示することができます。

A については、0〜1 の間で指定してください。

CAPTION = キャプション ID

文字列の ID を指定してください。

こちら を参照ください。

CAPTION = "文字列"

文字列を設定します。

```
CAPTION = "Hellow world!";
```

CAPTION_COLOR = R,G,B,A

キャプションのカラーを指定できます。

0〜1 の間で指定してください。

CAPTION_OFFSET = X, Y

キャプションの位置を移動することができます。

キャプションオフセットは 割合指定 が可能です。

STYLE = フラグ 0|フラグ 1|..|フラグ n

コントロールの表示アンカーを指定できます。

表示位置アンカーフラグ	説明
ANCHOR_DEFAULT	アンカー位置を左上に設定 ANCHOR_LEFTTOP と同じ コントロールの中心位置はデフォルトで、BASE_LEFTTOP になる。
ANCHOR_LEFTTOP	アンカー位置を左上に設定 コントロールの中心位置はデフォルトで、BASE_LEFTTOP になる。
ANCHOR_LEFT	アンカー位置を左に設定 縦にはセンタリング コントロールの中心位置はデフォルトで、BASE_LEFT になる。
ANCHOR_LEFTBOTTOM	アンカー位置を左に設定 縦には下辺を基準に配置 コントロールの中心位置はデフォルトで、BASE_LEFTBOTTOM になる。
ANCHOR_TOP	アンカー位置を上辺に設定 横方向にはセンタリング コントロールの中心位置はデフォルトで、BASE_TOP になる。
ANCHOR_CENTER	アンカー位置を画面中央にセンタリング コントロールの中心位置はデフォルトで、BASE_CENTER になる。
ANCHOR_BOTTOM	アンカー位置を底辺に設定 横方向にはセンタリング コントロールの中心位置はデフォルトで、BASE_BOTTOM になる。
ANCHOR_RIGHTTOP	アンカー位置を右上に設定 コントロールの中心位置はデフォルトで、BASE_RIGHTBOTTOM になる。
ANCHOR_RIGHT	アンカー位置を右に設定 縦にはセンタリング コントロールの中心位置はデフォルトで、BASE_RIGHT になる。
ANCHOR_RIGHTBOTTOM	アンカー位置を右に設定 縦には下辺を基準に配置 コントロールの中心位置はデフォルトで、BASE_RIGHTBOTTOM になる。

コントロールの中心位置を指定できます。

中心位置変更フラグ	説明
BASE_DEFAULT	アンカーフラグに応じて変わる。 特に指定しなければ、これになります。 各アンカーフラグの説明を参照
BASE_LEFTTOP	中心位置をコントロールの左上に設定
BASE_LEFT	中心位置を左に設定 縦にはコントロールの真中
BASE_LEFTBOTTOM	中心位置を左下に設定
BASE_TOP	中心位置を上辺に設定 横にはコントロールの中心
BASE_CENTER	中心位置をコントロールの中心
BASE_BOTTOM	中心位置を底辺に設定 横方向にはコントロールの真中
BASE_RIGHTTOP	中心位置をコントロールの右上に設定
BASE_RIGHT	中心位置を右に設定 横にはコントロールの中心
BASE_RIGHTBOTTOM	中心位置を右下に設定

キャプションの中心位置を指定できます。

テキストアンカー	説明
TEXT_CENTER	テキスト表示を中心に合わせます。
TEXT_LEFT	テキスト表示を左に合わせます。
TEXT_RIGHT	テキスト表示を右に合わせます。

キャプションのフォント装飾を変更できます。

キャプション装飾種類	説明
TEXT_NORMAL	装飾なし
TEXT_BOLD	太字
TEXT_DENT	へこんだ感じの文字
TEXT_SHADOW	影付き文字

機能を制限するスタイルは以下のものがあります。

機能制限スタイル	説明
HIDE	表示を隠す。
DRAG	ドラッグ可能にする。
DISABLE	押せなくし、暗くする。
HIT	押せるようにする。

11.27 BAR

C#: *CWinCtrlBar*

バーを表示するためのコントロールです。

ラベルと同様キャプションを持てます。

また、onClick イベントを受け取ることが可能です。

BAR(コントロール名) {

プロパティ 1;

プロパティ 2;

:

:

プロパティ n;

};

11.27.1 記述例

```
WINDOW(255_000_00001) {
  STYLE = NOTITLEBAR|ANCHOR_CENTER;
  POSITION = 0,100;
  SIZE = 512,256;
};

$w = 400;
BAR(TEST) {
  ID = 001_000_00020;
  CAPTION = 001_000_00030;
  CAPTION_COLOR = 0,0,0,1;
  STYLE = ANCHOR_LEFTTOP;
  POSITION = 16,64;
```

(次のページに続く)

(前のページからの続き)

```
  SIZE = $w,48;
};
```

11.27.2 プロパティ

代表的なデフォルト値

```
TEX_ID = "BAR";
STYLE = TEXT_CENTER;
COLOR = 1,1,1,1;
CAPTION_COLOR = 1,1,1,1;
```

ID = コントロール ID

コントロール ID を定義します。

```
ID = 001_000_00010;
```

注釈: 設定しなかったときは、自動的にハッシュ値から生成します。

POSITION = X, Y

表示位置を決定します。STYLE の アンカー 指定に応じて、基準位置が変わります。

```
POSITION = 32,{50} + 64;
```

座標は 割合指定 が可能です。

SIZE = 横幅, 縦幅

リストボックスの表示領域を定義します。

表示領域をはみ出たコントロールは、クリッピングされます。

```
SIZE = {100} - 32,400;                  //display size
CONTENTS_SIZE = 800,80;                 //one content size
```

サイズは 割合指定 が可能です。

TEX_ID = テクスチャ ID, パーツ ID

TEX_ID = パーツ ID

テクスチャ ID とパーツ ID を指定します。

COLOR = R,G,B,A

カラーを指定します。

R,G,B については、0〜2 の間で指定してください。

1 を超えたとき、そのカラー成分を 2 倍まで上げて表示することができます。

A については、0〜1 の間で指定してください。

CAPTION = キャプション ID

文字列の ID を指定してください。

こちら を参照ください。

CAPTION = "文字列"

文字列を設定します。

```
CAPTION = "Hellow world!";
```

CAPTION_COLOR = R,G,B,A

キャプションのカラーを指定できます。

0〜1 の間で指定してください。

CAPTION_OFFSET = X, Y

キャプションの位置を移動することができます。

キャプションオフセットは 割合指定 が可能です。

SE_ID = SE_ID

押されたときに鳴らす音の ID を設定します。デフォルトは、0 になっており、音がなりません。

STYLE = フラグ 0|フラグ 1|..|フラグ n

コントロールの表示アンカーを指定できます。

表示位置アンカーフラグ	説明
ANCHOR_DEFAULT	アンカー位置を左上に設定 ANCHOR_LEFTTOP と同じ コントロールの中心位置はデフォルトで、BASE_LEFTTOP になる。
ANCHOR_LEFTTOP	アンカー位置を左上に設定 コントロールの中心位置はデフォルトで、BASE_LEFTTOP になる。
ANCHOR_LEFT	アンカー位置を左に設定 縦にはセンタリング コントロールの中心位置はデフォルトで、BASE_LEFT になる。
ANCHOR_LEFTBOTTOM	アンカー位置を左に設定 縦には下辺を基準に配置 コントロールの中心位置はデフォルトで、BASE_LEFTBOTTOM になる。
ANCHOR_TOP	アンカー位置を上辺に設定 横方向にはセンタリング コントロールの中心位置はデフォルトで、BASE_TOP になる。
ANCHOR_CENTER	アンカー位置を画面中央にセンタリング コントロールの中心位置はデフォルトで、BASE_CENTER になる。
ANCHOR_BOTTOM	アンカー位置を底辺に設定 横方向にはセンタリング コントロールの中心位置はデフォルトで、BASE_BOTTOM になる。
ANCHOR_RIGHTTOP	アンカー位置を右上に設定 コントロールの中心位置はデフォルトで、BASE_RIGHTBOTTOM になる。
ANCHOR_RIGHT	アンカー位置を右に設定 縦にはセンタリング コントロールの中心位置はデフォルトで、BASE_RIGHT になる。
ANCHOR_RIGHTBOTTOM	アンカー位置を右に設定 縦には下辺を基準に配置 コントロールの中心位置はデフォルトで、BASE_RIGHTBOTTOM になる。

コントロールの中心位置を指定できます。

中心位置変更フラグ	説明
BASE_DEFAULT	アンカーフラグに応じて変わる。 特に指定しなければ、これになります。 各アンカーフラグの説明を参照
BASE_LEFTTOP	中心位置をコントロールの左上に設定
BASE_LEFT	中心位置を左に設定 縦にはコントロールの真中
BASE_LEFTBOTTOM	中心位置を左下に設定
BASE_TOP	中心位置を上辺に設定 横にはコントロールの中心
BASE_CENTER	中心位置をコントロールの中心
BASE_BOTTOM	中心位置を底辺に設定 横方向にはコントロールの真中
BASE_RIGHTTOP	中心位置をコントロールの右上に設定
BASE_RIGHT	中心位置を右に設定 横にはコントロールの中心
BASE_RIGHTBOTTOM	中心位置を右下に設定

キャプションの中心位置を指定できます。

テキストアンカー	説明
TEXT_CENTER	テキスト表示を中心に合わせます。
TEXT_LEFT	テキスト表示を左に合わせます。
TEXT_RIGHT	テキスト表示を右に合わせます。

キャプションのフォント装飾を変更できます。

キャプション装飾種類	説明
TEXT_NORMAL	装飾なし
TEXT_BOLD	太字
TEXT_DENT	へこんだ感じの文字
TEXT_SHADOW	影付き文字

機能を制限するスタイルは以下のものがあります。

機能制限スタイル	説明
HIDE	表示を隠す。
DRAG	ドラッグ可能にする。
DISABLE	押せなくし、暗くする。
NOHIT	押せなくする。

リファレンス:

11.28 CMainSystemBase

ゲームオブジェクトに貼り付けた後、常駐します (シーンが切り替わっても削除されません)。

常駐するマネージャなどをこのスクリプト経由で追加することによって、必要なアセットをダウンロードしてくれます。

また、シーン切り替え等に、このスクリプトを使うと AssetBundle ダウンロード時などに切り替えをペンディングしてくれます。

使うときは、CMainSystem クラスを作成し、継承するようにしてください。

class CMainSystemBase

void **Awake**()

Awake 関数の中で AddComponent,addManager をすることによって、Start 関数内で自動的に必要な初期化や必要なアセットバンドルの読み込みを自動的に行います。

代表的な記述方法は次の通りです。

注釈: void Start() は、継承先で new によって上書きしないようにしてください。読み込みが正常に行えなくなります。

```
new void Awake() {
      base.Awake();

   if (m_instance != null) {
       Debug.LogError("already exist CMainSystem");
       return;
   }
   m_instance = this;
```

(次のページに続く)

(前のページからの続き)

```
    // Add Component
    gameObject.AddComponent<CInput>();
    if (KsSoftConfig.UseAssetBundle) {
        gameObject.AddComponent<CAssetBundleMgr>();
    }
    gameObject.AddComponent<CSpriteFontMgr>();
    gameObject.AddComponent<CTextureResourceMgr>();
    gameObject.AddComponent<CWindowMgr>();
    gameObject.AddComponent<CBgmResourceMgr>();
    gameObject.AddComponent<CSeResourceMgr>();
    gameObject.AddComponent<CSoundEffectMgr>();

    // Managers that need to be initialized.
    addManager(new CMessageDataSheetMgr(Utility.getSystemLocale()));
}
```

void **addManager**(*IManager mgr*)

Param IManager mgr　アセットバンドルを読み込み初期化する必要のあるオブジェクト

注釈: Awake 内で、AddComponet する必要があります。

---

virtual void **initialize**()

継承先でオーバライドしてください。アセットバンドルの読み込みが終わり、ILoader,IManager の読み込みが完了した後、呼び出される。

virtual void **OnChangeScene**()

継承先でオーバライドしてください。シーンを切り替えたときに呼び出されます。

bool **changeScene**(string *sScene*, bool *bForceChange=false*)

指定したシーンに切り替えます。

Application.LoadLevel と同等ですが、以下の点で異なります。

1. フェードオブジェクトのフェードが終わるまで待ちます。
2. アセットバンドル読み込み中などにシーン切り替えをペンディングしてくれます。
3. クリエイト済のウィンドウを自動的に閉じてくれます。

シーン切り替え直前 (Application.LoadLevel が呼び出される直前) に、OnChageScene が呼び出されます。

bool **isChangingScene** { **get;** }

シーン読み込み中かどうか判定します。

CMainSystemBase.chageScene によってシーンを切り替えているときのみ有効です。

bool **isInitialized** { **get;** }

初期化が終わっているかどうか判定します。

static int **randi**（）

整数値の乱数を取得します。

static int **randi**（int *iMax*）

指定された範囲の乱数を取得します。

static int **randi**（int *iMin*, int *iMax*）

指定された範囲の乱数を取得します。

static float **randf**（）

浮動小数点の乱数を取得します。

class ILoader

bool **isLoading**（）

Return bool　読み込み中は true, 読み込み終了状態の時は false

MonoBehaviour を継承したスクリプトにつけてください。

CMainSystemBase と同じゲームオブジェクト内のスクリプトにおいて、ILoader を持つオブジェクトを調べ、ILoader.isLoading() が false になるまで待機します。

注釈: Awake 内で、AddComponet する必要があります。

---

class IManager

Awake() 内で、CMainSystemBase.addManager で追加してください。

アセットバンドルを読み込んだのち、それを使って初期化する必要のあるオブジェクトに対してつけておく必要があります。

```
addManager(new CMessageDataSheetMgr());
```

bool **initialize**（*CAssetBundle*[] *aAssetBundles*）

Param CAssetBundle[] aAssetBundles　読み込み終わったアセットバンドルの配列

void **release**（）

ゲームオブジェクトが解放されるときに、呼ばれる解放処理

uint[] **getAssetBundleIds**（）

Return uint[]　初期化時に読み込んでおく必要のあるアセットバンドルの ID

読み込む必要のあるアセットバンドル ID の配列を返す関数です。

11.29 CAssetBundleMgr について

class CAssetBundleMgr

CMainSystem の Awake 内で AddComponent してください。

アセットバンドルを読み込んだりキャッシュしたりするためのマネージャです。

自前のマネージャを用意するときは、不必要です。

最初に、version.unity3d をダウンロードし、各アセットのバージョンを取得します。

バージョンが異なっているときはローカルにキャッシュされたデータを捨て、ダウンロードしてきます。

次のような特徴を持っています。

1. アセットバンドルはバージョン管理されており、一度読み込んだものはキャッシュから読み込むようになっている (WWW.LoadFromCacheOrDownload)
2. 読み込んだアセットバンドルはメモリにキャッシュする
3. シーン切り替え時に、キャッシュを解放する
4. 常駐フラグが付いているアセットバンドルは解放せず保持する
5. Windows,Mac 上では設定ファイルによって読み込み先を切り替え可能 (ローカルのファイルでも可)
6. 扱えるアセットバンドルの名前は、 マルチ *ID* .unity3d というフォーマットに限る

const int **MaxConcurrentLoadNum** { **get; set;** }

同時にダウンロードする最大数

この数を超えて、LoadFromCacheOrDownload を同時に発行しないようになっています。

デフォルト値は、1 になっています。

const long **MaximumAvailableDiskSpace** { **get;** }

ローカルディスクへの最大キャッシュサイズを指定します。

デフォルト値は、512 * 1024 * 1024(512Mbyte) になっています。

static string **httpServerPath** { **get; set;** }

HTTP アドレスを指定します。

static string **debugPath** { **get; set;** }

デバッグ用パスを指定します。Windows,Mac 環境でのみ有効です。こちらを指定しておくと、そのパスを優先してファイルを読み込みます。

注釈: デバッグパスを経由して読み込んだファイルは、ローカルディスクへのキャッシュを行いません。

---

int **lastError** { **get;** }

最後に発生したエラー番号が入っています。

int **errorMessage** { **get;** }

最後に発生したエラーメッセージが入っています。

CAssetBundle[] **lodings**

現在ダウンロード中のアセットバンドルのリストが返ってきます。

ダウンロードをペンディングしているものもこのリストに含まれます。

int **loadNum** { **get;** }

現在ダウンロード中のアセットバンドルの数が返ってきます。

ダウンロードをペンディングしているものも数に含まれます。

static *CAssetBundleMgr* **Instance** { **get;** }

マネージャのインスタンスを取得します。

CAssetBundle **reference**（uint *id*）

アセットバンドルを参照します。もし、キャッシュされていないときは、ダウンロードします。

ダウンロード済みでなくてもアセットバンドルが存在するときは、CAssetBundle を返します。

返ってきた CAssetBundle がダウンロード中かどうかは、CAssetBundle.isLoaded によって判別可能です。

true の時、使用可能になっています。

null が返ってくるときは、アセットバンドルがそもそも存在しないときです。

bool **isExist**（uint *id*）

アセットバンドルが存在するかどうか調べます。

void **startPreload**()

呼び出すと、存在するアセットバンドル全てをダウンロードし、ローカルディスクにキャッシュしようとします。

void **loadAssetVersion**()

アセットバージョン (version.unity3d) をダウンロードしなおし、アセットバンドルのキャッシュを更新します。

void **releaseAll**()

常駐/非常駐に関わらず、全てのアセットバンドルを解放します。

解放後は、loadAssetVersion を呼び出す必要があります。

class CAssetBundle

AssetBundle にキャスト可能です。

AssetBundle **get**()

保持している AssetBundle を取得します。

uint **id** { **get;** }

ID を取得します。

bool **isLoaded** { **get;** }

true なら読み込み済みです。

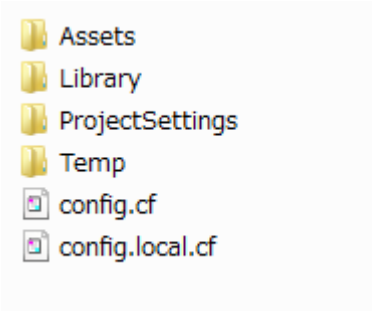
常駐しているかどうか調べます。

11.29.1 ローカルのファイルを読み込む際のパス変更方法

Windows,Mac 上でのみ有効です。

最初に Assets フォルダと同じ場所に次のファイルを作る。

- config.cf
- config.local.cf



読み込まれる順番は、config.cf => config.local.cf の順番になります。

後で設定したものが上書きされます。

config.cf に次のように記述しておきます。

```
#
# config.cf
#
DEBUG_PATH = ../../../unified/AssetBuilder/assetbundles/Windows
```

CMainSystem の Awake 内で次のように設定します。

```
//=============================================================================
/*!Awake
* @brief    Unity Callback
*/
new void Awake() {
     base.Awake();
     if (m_instance != null) {
         Debug.LogError("already exist CMainSystem");
         return;
     }
     m_instance = this;
     #if !(UNITY_IPHONE || UNITY_ANDROID || UNITY_WEBPLAYER)
         CAssetBundleMgr.debugPath = CConfig.Instance.get ("DEBUG_PATH","../Debug/");
     #endif
     if (KsSoftConfig.UseAssetBundle) {
             gameObject.AddComponent<CAssetBundleMgr>();
         }
```

11.30 アセットバンドルについて

CAssetBundleMgr は最初に version.unity3d を読み込みます。本項目では、この version.unity3d の仕組みについて説明します。

11.30.1 version.unity3d が作られるタイミング

- [Tools]->[KsSoft]->[Export Binary Data]
- [Tools]->[KsSoft]->[Export Window Resource]

何らかのアセットバンドルを作成、更新したときに自動的に更新されます。

これらツール内では、次の関数が呼ばれています。

```
ExportVersion.export(BuildTarget eTarget);
```

version.unity3d には、export 前の以下の情報を持っています。

- 存在していたアセットバンドル
- 各アセットバンドルの MD5
- 各アセットバンドルのバージョン番号
- どのアセットバンドルが読み込み後、常駐するかどうかの情報

この関数を呼び出すことによって、ビルドターゲットに対応したフォルダ内のアセットバンドルの MD5 と比較して変更が一つでもあった時は、version.unity3d を再構築しています。

この時、変更のかかっていたファイルに対して、現在時刻から生成したバージョン番号を与えています。

これによって、version.unity3d を読み込むだけで、次のことが可能になっています。

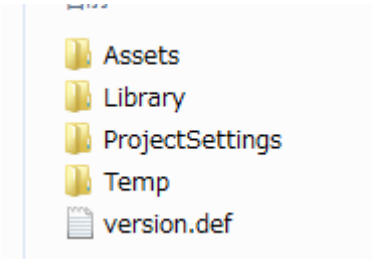
- アセットバンドルに何が存在しているか事前にわかる
- LoadFromCacheOrDownload に使う引数である version を変更のあったファイルに対してのみ新しく割り振ることができるため、更新ファイルのみをダウンロードすることが可能になる

注釈: 自前でアセットバンドルを作りたいときは、スクリプトの最後に、version.unity3d を export してあげることを忘れないでください。

11.30.2 アセットバンドルの常駐

ダウンロードしたアセットを常駐するように version.unity3d 更新時に設定可能です。

最初に Assets フォルダと同じ場所に、version.def というファイルを作ってください。




```
#Texture for Window Resource
100_000_00001  =       REMAIN:1;
#SoundEffect for 2D
052_000_00000  =       REMAIN:1;
#SoundEffect for 3D
053_000_00000  =       REMAIN:1;
```

このように指定したアセットバンドルは、読み込み後に常駐することになります。

11.30.3 version.txt

version.txt は、あくまでもデバッグの為に可視化した、version.unity3d です。

エキスポート時/ランタイム時には不必要なファイルです。

11.30.4 バージョン管理ツールなどでぶつかった時

次の手順を踏んで下さい。

アセットバンドル自体が衝突したとき

頑張って解消してください。

version.unity3d が衝突したとき

全てのアセットバンドルの競合を解消した後、version.unity3d を一旦消します。その後、アップロードされている version.unity3d を落としてきます。最後にローカル環境で以下の方法で version.unity3d を出力しなおします。

- [Tools]->[KsSoft]->[Re-Export Version]

11.31 CTextureResourceMgr について

CMainSystem の Awake 内で AddComponent してください。

テクスチャリソースを管理するためのマネージャです。

CTextureResource のコンテナとしての役割を果たします。

CTextureResoruce をダウンロード、若しくはリソースから読みます。既に読み込まれていたらキャッシュを返します。

ウィンドウシステムも、このマネージャを経由してテクスチャを読み込んでいます。

リソース作成方法は こちら を参照ください。

注釈: アセットバンドル経由で読み込むには、*CAssetBundleMgr* を登録しておく必要があります。

---

class CTextureResourceMgr

CTextureResource **reference**（uint *id*）

Param uint id　アセットバンドル ID

テクスチャリソースを参照します。もし、キャッシュされていないときは、ダウンロードします。

ダウンロード済みでなくてもテクスチャリソースが存在するときは、CTextureResource を返します。

返ってきた CTextureResource がダウンロード中かどうかは、CTextureResource.isLoaded によって判別可能です。

true の時、使用可能になっています。

アセットバンドルとしてダウンロードできないときは、リソースデータとして読み込めるか試します。

それでも存在しないときは、null が返ってきます。

static CTextureResourceMgr **Instance** { **get;** }
マネージャのインスタンスを取得します。

class CTextureResource
Material にキャスト可能です。

Material **get**（）
保持している Material を取得します。

uint **id** { **get;** }
ID を取得します。

bool **isLoaded** { **get;** }
true なら読み込み済みです。

bool **isRemain** { **get;** }
常駐しているかどうか調べます。

CSpriteDataOne[] **spriteData** { **get;** }
スプライトデータを取得します。

アトラス化されたテクスチャパーツの UV 情報、パッチ情報、カラー情報等が格納されています。

class CSpriteDataOne
アトラス化されたテクスチャパーツの UV 情報、パッチ情報、カラー情報等が格納されています。

uint **m_id** { **get;** }

WinColor **m_color** { **get;** }

e_Patch **m_ePatch** { **get;** }

```
public enum e_Patch {
    None,    //patch none
    H3,      //3 patch in a horizontal direction
    V3,      //3 patch in the vertical direction
    HV9,     //9 patch
};
```

Vector4 **m_uv** { **get; set;** }

テクスチャの UV 情報が格納されています。

Vector4[] **m_aUV** { **get; set;** }

e_Patch が H3,V3 ならば、三つの UV 情報が格納されています。e_Patch が HV9 ならば、9 つの UV 情報が格納されています。

11.32 CSoundEffectMgr

AudioSource を複数持ち、SE のグループ化、プライオリティ、同時発生数制限等を行います。

デフォルトの同時発生数制限は、このようになっています。

- 2D 音声 4 音
- 3D 音声 8 音

また、グループ毎に同時発生数制限を個別に設定可能になっています。

SE のこれら設定を行う方法は、こちらを 参照 ください。

AudioSource を管理しており、音声を発生させるためにプログラマが別途 AudioSource を作成する必要がなくなっています。

11.32.1 マネージャを組み込む方法

CSoundEffectMgr を使うには、次のようにマネージャをゲームオブジェクトに対して、Add する必要があります。

```
public class CMainSystem : CMainSystemBase {
   //=============================================================================
    /*!Awake
    * @brief  Unity Callback
    */
    new void Awake() {
        base.Awake();

        if (m_instance != null) {
            Debug.LogError("already exist CMainSystem");
            return;
        }
        m_instance = this;

        // Add Component
        gameObject.AddComponent<CInput>();
        if (KsSoftConfig.UseAssetBundle) {
                gameObject.AddComponent<CAssetBundleMgr>();
           }
        gameObject.AddComponent<CSpriteFontMgr>();
        gameObject.AddComponent<CTextureResourceMgr>();
        gameObject.AddComponent<CWindowMgr>();
        gameObject.AddComponent<CBgmResourceMgr>();
        gameObject.AddComponent<CSeResourceMgr>();
        gameObject.AddComponent<CSoundEffectMgr>();

        addManager(new CMessageDataSheetMgr(Utility.getSystemLocale()));
    }
```

class CSoundEffectMgr

void **playBgm**（*CBgmResource cBgmResource*）

Param CBgmResource　読み込んだ BGM リソース

void **playBgm**（uint *id*）

Param uint id　アセットバンドル ID

指定された BGM を再生します。もし、指定されたアセットバンドルが読み込まれていないときは、読み込み終わった後に自動的に再生を開始します。明示的に読み込みを待ちたいときは、cBgmResource.isLoaded() が、true を返すのを待ってから、再生してください。

CBgmResource を直接取得するには、次のように記述します。

```
IEnumerator load(uint id) {
 CBgmResource cBgmResource = CBgmResourceMgr.Instance.reference(id);

 if (cBgmResource == null) {
     // error!!
     yield break;
 }
 while (!cBgmResource.isLoaded()) {
     yield return 0;
 }
 CSoundEffectMgr.Instance.playBgm(cBgmResource);
};
```

void **play**（CSoundEffect *cSE*）

Param CSoundEffect cSE　CSeResource から取得可能な、SE オブジェクトです。

指定された SE を 2D 音声として再生します。CSoundEffect は、次の方法で取得可能です。

```
CSeResource   m_cSR;
          :
CSoundEffect  cSE = m_cSR.find(id);
```

void **play**（uint *mAssetBundle*, uint *id*）

Param uint mAssetBundle　アセットバンドル ID

Param uint id　SE ID

指定された SE を 2D 音声として再生します。

void **play**（CSoundEffect *cSE*, IEffectEmitter *emitter*）
void **play**（CSoundEffect *cSE*, Transform *trans*）
void **play**（CSoundEffect *cSE*, Vector3 *position*）

Param CSoundEffect cSE　CSeResource から取得可能な、SE オブジェクトです。

Param IEffectEmitter emitter　エフェクトが発生するエミッター

Param Transform trans　音が発生する場所

Param Vector3 position　音が発生する場所

指定された SE を 3D 音声として再生します。

マネージャのインスタンスを取得します。

class CSeResourceMgr

CSeResource を管理するコンテナになっています。

CAssetBundleMgr 経由で、アセットバンドルをダウンロードしてきた後、読み込んだ音声データを管理します。

CSeResource **refenrece**（uint *mAssetBundle*）

Param uint mAssetBundle　SE リソース ID(アセットバンドル ID) を指定します。

Return　SE リソースが返ってきます。

SE リソースを取得する。取得できるアセットは、こちら でコンバートしたものです。

CSoundEffect **find**（uint *mAssetBundle*, uint *mId*）

Param uint mAssetBundle　SE リソースを指定する。

Param uint id　SE ID を指定します。

Return　SE が返ってきます。指定した SE がパックされているアセットバンドルがまだ読み込み中であったり存在しないときは、null が返ります。

CSoundEffect を取得します。

static CSeResourceMgr **Instance** { **get;** }

マネージャのインスタンスを取得します。

class CSeResource

アセットバンドル化された SE を使える状態にして保持しています。

CSeResource 一つに、複数の SE がまとまった形で入っています。

IWinSoundEffect インターフェースを持ち、CWindowMgr に登録することが可能です。

実際に、SE を鳴らすには、CSoundEffectMgr 経由で行います。

CSoundEffect **find**（uint *id*）

Param uint id　取得したい SE ID を指定します。

Return　SE が返ってきます。

SE を取得します。

void **play**（uint *mSE*）

Param uint id　再生したい SE ID を指定します。

SE を再生します。

bool **isLoaded** { **get;** }

アセットバンドルが読み込み終わっているかどうか調べます。

true の時、読み込みが完了しており、使用可能な状態になっています。

class CBgmResourceMgr

CBgmResource を管理するコンテナになっています。

CAssetBundleMgr 経由で、アセットバンドルをダウンロードしてきた後、読み込んだ BGM データを管理します。

実際に、BGM を鳴らすには、CSoundEffectMgr 経由で行います。

CBgmResource **refenrece**（uint *mAssetBundle*）

Param uint mAssetBundle　BGM リソース ID(アセットバンドル ID) を指定します。

Return　BGM リソースが返ってきます。

BGM リソースを取得します。取得できるアセットは、こちら でコンバートしたものです。

CBgmResource **load**（uint *mResource*）

Param uint mResource　リソース ID を指定します。

Return　BGM リソースが返ってきます。

Resources フォルダの下に次のような形でファイルを置いておくと読み込むことができます。

Assets/Resources/001_000_000000.mp3 Assets/Resources/001_000_000000.intro.mp3

static CBgmResourceMgr **Instance** { **get;** }

マネージャのインスタンスを取得します。

class CBgmResource

アセットバンドル化された BGM を使える状態にして保持しています。

CBgmResource 内に、イントロ部分とループ部分の音声データを持っています。

実際に、BGM を鳴らすには、CSoundEffectMgr 経由で行います。

AudioClip **loopClip** { **get;** }

ループ部分のクリップ

AudioClip **introClip** { **get;** }

イントロ部分のクリップ

bool **isLoaded** { **get;** }

アセットバンドルが読み込み終わっているかどうか調べます。

true の時、読み込みが完了しており、使用可能な状態になっています。

11.33 CMessageDataSheetMgr について

文字リソースを管理するためのオブジェクトです。

CMessageDataSheet が、Excel のシートに対応しています。

CMessageDataSheetMgr は、CMessageDataSheet を束ねるコンテナになっています。

ちょうど、CMessageDataSheetMgr が、Excel の 1 ファイルに相当するようになっています。

11.33.1 アプリケーションに組み込む方法

CMainSystem の Awake の中で、addManager します。

```
public class CMainSystem : CMainSystemBase {
   //=============================================================================
    /*!Awake
    * @brief   Unity Callback
    */
    new void Awake() {
        base.Awake();

        if (m_instance != null) {
            Debug.LogError("already exist CMainSystem");
            return;
        }
        m_instance = this;

        // Add Component
        gameObject.AddComponent<CInput>();
        gameObject.AddComponent<CAssetBundleMgr>();
        gameObject.AddComponent<CSpriteFontMgr>();
        gameObject.AddComponent<CTextureResourceMgr>();
        gameObject.AddComponent<CWindowMgr>();
        gameObject.AddComponent<CBgmResourceMgr>();
        gameObject.AddComponent<CSeResourceMgr>();
        gameObject.AddComponent<CSoundEffectMgr>();

        addManager(new CMessageDataSheetMgr(Utility.getSystemLocale()));
    }
```

また、CWindowMgr のキャプションと関連付けたいときは、CMainSystem の initialize の中で次のように設

定します。

```
public class CMainSystem : CMainSystemBase {
   //=============================================================================
   /*!Initialize
   * @brief    initialize
   */
   override protected void initialize() {
       base.initialize();

       //-----------------------------------------
       // WindowMgr initialize.
       //-----------------------------------------
       CWindowMgr      cWindowMgr = CWindowMgr.Instance;
       // initialize caption interface
       cWindowMgr.captiondata = CMessageDataSheetMgr.Instance.find(new FiveCC("WNDW"));
```

CMessageDataSheetMgr が、IWinCaptionData インターフェースを持っているためこのように記述可能です。

class CMessageDataSheetMgr

CMessageDataSheetMgr は、CMessageDataSheet を複数持つ、コンテナになっています。

CMessageDataSheet は、Excel 上でシート一つに該当します。

CMessageDataSheetMgr は、Excel のシートを束ねて持っているコンテナとも言えます。

ただし、複数ロケールを一つのアセットに纏めて持つわけではなく、ロケール毎に分けて文字リソースデータを持っています。よって、アプリケーションでは一つのロケールデータのみを保持することになります。

void **initialize**（byte[] *buffer*）

Param byte[] buffer　読み込んだバイナリデータ

Excel から生成したバイナリデータを指定します。

CMessageDataSheet **find**（uint *sheetname*）

Param uint type　シート名を *FiveCC* で渡します。

シートを一つ取得します。

```
CMessageDataSheet cSheet = cMessageDataSheetMgr.find(new FiveCC("WNDW");
```

現在設定されているロケールを取得する。

CMessageDataSheetMgr のインスタンスを取得する。

class CMessageDataSheet

Excel 上のシート一つに対応します。

string **find**（uint *id*）

Param uint id　マルチ *ID* を指定し、文字リソースを取得します。

string **format**（uint *id*, *params* object[] *args*）

Param uint id　マルチ *ID* を指定し、文字リソースを取得します。

Param params object[] args　可変引数

フォーマットにそって文字リソースを生成し返します。

```
string str = cMessageDataSheetMgr.format(new MulId(0,0,1),"Test",n,value);
```

11.34 CWindowMgr について

CMainSystem の Awake 内で AddComponent してください。

ウィンドウを管理するためのマネージャです。

このマネージャを介して全てのウィンドウをクリエイトします。

class CWindowMgr

void **setUIResolution**（float *width*, float *height*）

Param width　UI のスクリーン幅を設定します。0 を指定したときは、height を基準に拡縮率を計算します。

Param height　UI のスクリーン高を設定します。width = 0 を指定した場合のみ、height を基準に拡縮率を計算します。

UI の仮想的なスクリーンサイズを設定します。アスペクト比は自動的に狂わないように拡縮率を決定するようになっています。

- width != 0 の場合

width を基準に UI スクリーンサイズを決定します。

- width == 0 and height != 0 の場合

height を基準に UI スクリーンサイズを決定します。

- width == 0 and height == 0 の場合 (デフォルト)

端末のスクリーンサイズに準拠します。

bool **initialize**（*CAssetBundle*[] *aAB*）

読み込まれたウィンドウリソースを展開し、初期化します。CAssetBundleMgr が登録されていたら、自動的にウィンドウリソースをサーバからダウンロードしてきて、この関数に渡されます。

bool **load**（byte[] *buffer*）

バイトデータとして読み込んだウィンドウリソースを展開し初期化します。

bool **load**（string *assetname*）

Param string assetname　読み込まれるアセット名

リソースデータから、指定したアセット名のウィンドウリソースを読み込み初期化します。

実際に読み込まれるアセットは、assetname + ".asset"という名前です。

WindowScript **create**<WindowScript>（uint *id*, *CWindowBase cParent* = *null*）

Param WindowScript　クリエイトするウィンドウのクラス名

Param uint id　ウィンドウ ID

Param CWindowBase cParet　親ウィンドウ

指定したクラスのウィンドウをクリエイトする。

WindowScript **find**<WindowScript>（uint *id*）

Param WindowScript　ウィドウクラス名

Param uint id　ウィンドウ ID

指定されたウィンドウを探す。まだクリエイトされていないなら、null が返る。

void **bringToTop**（*CWindowBase cWindow*）

Param CWindowBase cWindow　対象ウィンドウ

対象ウィンドウの優先順位を上げる。

string **getCaption**（uint *mCaptionId*）

Param uint mCaptionId　キャプション ID

指定されたキャプション ID から、文字列を取得します。

void **play**（uint *mSE*）

Param uint mSE　SE の ID

指定された SE ID から、効果音を再生する。

uint **clickSE** { **get; set;** }

タッチしたときに鳴らす標準音を指定します。

uint **scrollSE** { **get; set;** }

リストボックスなどをスクロールさせたときに鳴らす標準音を指定します。

static CWindowMgr **Instance** { **get;** }

ウィンドウマネージャのインスタンスを取得します。

11.35 CWindowBase

11.35.1 ウィンドウクラスを作成する方法

ウィンドウスクリプトをコンパイルし成功するとウィンドウバイナリファイルとそのウィンドウを使うためのベースファイルが生成されます。

例えば、次の通りです。

```
#include "wr.h"
WINDOW(TEST) {
  ID = 255_000_00001;
  STYLE = NOTITLEBAR|ANCHOR_CENTER;
  POSITION = 0,-100;
  SIZE = 512,256;
};

TEXTBOX(Message) {
  ID = 000_000_00010;
  POSITION = 100,-80;
  CAPTION = 000_111_00020;
  EDIT = 255,2;
  SIZE = 380, 64;
};
```

これをコンパイルすると、次のような **CTestBase.cs** が自動生成されます。

```
public class TESTBase : CWindowBase {
    public const uint windowId = 4278190081;  // 255_000_00001
        static public TEST create(CWindowBase cParent = null) {
        return CWindowMgr.Instance.create<TEST>(windowId,cParent);
  }
  public const uint TEXTBOX_Message = 10;     // 000_000_00010
};
```

アプリケーションに組み込むときは、この TestBase を継承したウィンドウを作ってください。

```
public class TEST : TestBase {
   :
};
```

そして、ウィンドウを実際にクリエイトするときは、次の通りにします。

```
TEST test = TEST.create();
```

- 各ウィンドウオブジェクトに届くコールバックについて *

class CWindowBase

ウィンドウは複数のコントロールから構成されています。

コントロールに対してタッチやドラッグ等の操作を行うと、それに応じてウィンドウ内のコールバックが呼ばれます。

受け取れるコールバックを下記に列挙しておきます。

各ウィンドウを作成するというのは、ウィンドウスクリプトをコンパイルした際に生成されたクラスを継承し、これらコールバックをオーバライドする作業になります。

virtual void **onCreate**()

ウィンドウの初期化が終わった直後に呼ばれる。

virtual void **onUpdate**()

各コントロールのアップデート前に呼ばれる。

virtual void **onAfterUpdate**()

各コントロールのアップデート後に呼ばれる。

virtual void **onPreRender**(CWindowRenderer *cRenderer*)

コントロールのレンダリング前に呼ばれる。

virtual **onRender**(CWindowRenderer *cRenderer*)

コントロールのレンダリングが終わった後に呼ばれる。

virtual bool **onClose**(int *iCloseInfo*)

ウィンドウを閉じたときに呼ばれる。

- true: ウィンドウを閉じる
- false: ウィンドウを閉じるのを抑制する

virtual void **onClick**(*CWinCtrlBase cCtrl*)

コントロールが押された。

virtual void **onHold**（*CWinCtrlBase cCtrl*）

コントロールが長押しされた。

virtual void **onClickEnter**（*CWinCtrlBase cCtrl*）

エディットボックスの入力が確定した。

virtual void **onClick**（*CWinCtrlBase cCtrl*, *CRichTextOne cText*）

リッチテキストの一部が押された

virtual void **onBeginDrag**（*CWinCtrlBase cCtrl*, Vector2 *pos*）

コントロールがドラッグされた。最初に一度、呼び出される。

virtual void **onDrag**（*CWinCtrlBase cCtrl*, Vector2 *pos*）

コントロールがドラッグされている。ドラッグ中、呼び続けられる。

スクロール可能なコントロールは、どの方向にコントロールがドラッグされたかを onDrag によって知ることができる。ただし、STYLE に *SCROLL_LOCK* をつけておく必要がある。onDrag によって、ドラッグした方向を検出できる。DRAG をつけても、これらコントロールには効果がないことに気を付ける必要がある。

virtual bool **onDragRender**（*CWinCtrlBase cCtrl*, Transform *transform*）

コントロールがドラッグされているときに、レンダリングタイミングで呼ばれる。override しなければ普通にコントロールがレンダリングされる。

戻り値の意味は以下の通りです。

- true ドラッグ中のコントロールの複製をレンダリングする。
- false ドラッグ中のコントロールの複製をレンダリングしない。

virtual void **onDrop**（*CWinCtrlBase cCtrl*, *CWindowBase cDragWindow*, *CWinCtrlBase cDragCtrl*）

特定ウィンドウのコントロール上でドラッグ中のコントロールが離されたとき呼ばれる。このコールバックは、cCtrl が含まれるウィンドウにコールバックが届く。

virtual void **onDropGround**（*CWinCtrlBase cCtrl*）

何もないところでドラッグ中のコントロールが離されたとき呼ばれる。

また、ドラッグ中のコントロールが含まれるウィンドウが消滅したときにも、消滅する直前に呼び出される。

virtual void **onOK**（int *msgBoxWinID*）

子ウィンドウとしてクリエイトしたメッセージウィンドウ上で OK ボタンが押された。

virtual void **onNext**（int *msgBoxWinID*）

子ウィンドウとしてクリエイトしたメッセージウィンドウ上で Next ボタンが押された。

virtual void **onYes**（int *msgBoxWinID*）

子ウィンドウとしてクリエイトしたメッセージウィンドウ上で Yes ボタンが押された。

virtual void **onNo**（int *msgBoxWinID*）

子ウィンドウとしてクリエイトしたメッセージウィンドウ上で No ボタンが押された。

virtual void **onCancel**（int *msgBoxWinID*）

子ウィンドウとしてクリエイトしたメッセージウィンドウ上で Cancel ボタンが押された。

virtual void **onRecreatedRenderTexture**（uint *mRenderTextureId*）

レンダーテクスチャが何らかのイベントによって、消失したとき呼ばれる。ただし、自動復帰はシステム側が行うので基本オーバーライドする必要はない。

- ウィンドウオブジェクトの持つ主なプロパティ/メソッド *

ここでは、CWindowBase が持つアクセス可能なメソッド、プロパティについて列挙しておきます。

virtual bool **onBeginRenderIcon**（*CWinCtrlRenderIcon icon*）

アイコンにレンダリング開始 *CWinCtrlRenderIcon* がウィンドウに含まれている時のみ呼び出されます。

この関数の戻り値は、次の意味を持ちます。

- true を返すとテクスチャにレンダリングします。
- false を返すとレンダリングしません。 *onEndRenderIcon* も呼ばれません。

virtual bool **onEndRenderIcon**（*CWinCtrlRenderIcon icon*）

レンダリングが終了したときに呼ばれる *CWinCtrlRenderIcon* がウィンドウに含まれている時のみ呼び出されます。

この関数の戻り値は、次の意味を持ちます。

- true を返すと、次のフレームもレンダリングを継続します (*onBeginRenderIcon* が呼ばれる)
- false を返すと、以降のフレームでは必要がない限り、レンダリングされません。

一度停止させたレンダリングを再開したいときは、 *needToRender* を true にしてください。

void **close**（int *iCloseInfo = 0*）

ウィンドウを閉じたいときに呼び出す。iCloseInfo は、onClose のパラメータとして渡されます。

WinCtrl **find**<WinCtrl>（uint *id*）

該当する ID を持つコントロールを探す。

CWinCtrlBase **find**（uint *id*）

該当する ID を持つコントロールを探す。

uint **id** { **get; set;** }

ウィンドウ ID を取得/設定する。

CWindowBase **child** { **get;** }

子ウィンドウを返す。

CWindowBase **parent** { **get;** }

親ウィンドウを返す。

e_LayerId **layer** { **get; set;** }

レンダリングレイヤーを取得/設定します。

int **priority** { **get; set;** }

描画プライオリティを取得/設定します。

uint **texid** { **get; set;** }

ウィンドウのデフォルトテクスチャ ID を返す。

float **x** { **get; set;** }

float **y** { **get; set;** }

ウィンドウ座標を取得/設定する。

Vector2 **position** { **get; set;** }

ウィンドウ座標を取得/設定する。

Transform **transform** { **get;** }

Transform を取得する。

Vector2 **scale** { **get; set;** }

x,y スケールを取得/設定する。

e_Anchor **anchor** { **get; set;** }

表示位置の基準アンカー設定を変更する。

float **width** { **get; set;** }

float **height** { **get; set;** }

ウィンドウのサイズを取得/設定する。

Vector2 **size** { **get; set;** }

ウィンドウのサイズを取得/設定する。

float **screenWidth** { **get; set;** }

float **screenHeight** { **get; set;** }

表示スクリーンサイズの取得/設定する。

string **caption** { **get; set;** }

ウィンドウのタイトルバーキャプションを取得/設定する。

WinColor **captionColor** { **get; set;** }

ウィンドウのタイトルバーキャプションカラーを取得/設定する。

Vector2 **captionOffset** { **get; set;** }

ウィンドウのタイトルバーキャプション表示オフセットを取得/設定する。

WinColor **color** { **get; set;** }

ウィンドウカラーを設定します。配下にあるコントロールは全て影響受けます。

e_WinStyle **style** { **get; set;** }

ウィンドウスタイルを取得します。

e_WinStyle **priorityStyle** { **get; set;** }

プライオリティスタイルを設定/取得する。

bool **hide** { **get; set;** }

- true

  ウィンドウ表示を隠す。

- false

  ウィンドウを表示する。

bool **disable** { **get; set;** }

- true

  ウィンドウの機能を奪い暗くする。

- false

  ウィンドウを通常表示する。

ウィンドウ機能を奪うと、タッチしても反応しなくなります。

bool **`isClose`** { **`get;`** }

ウィンドウがクローズアニメーション中なら true になります。

bool **`isLoaded`** { **`get;`** }

テクスチャ等の読み込みが行われているとき true になります。

bool **`isFade`** { **`get;`** }

ウィンドウがフェードイン/フェードアウト中なら true になります。

- その他オーバーライド可能なメソッド *

コールバック以外にオーバライド可能なメソッドが存在します。特にウィンドウをクリエイトしたとき/クローズしたときに行うフェードアニメーションをカスタマイズしたいときはオーバライドする必要があります。

protected virtual void **`startFade`**（e_FadeState *eState*）

フェードが始まった時に呼ばれる。フェードイン/フェードアウトは、e_FadeState を見て判断できる。フェード処理で必要な初期化を行ってください。

- e_FadeState.Open

  フェードイン

- e_FadeState.Close

  フェードアウト

protected virtual bool **`doFade`**（e_FadeState *eState*）

フェード中に呼ばれます。

- true

フェード中

- false

  フェード終了

フェードが終了したら、false を返すようにしてください。

また、e_FadeState.None がわたってきたときも、false を返すようにしてください。

- e_FadeState.None

  フェードしていない

- e_FadeState.Open

  フェードイン中

- e_FadeState.Close

  フェードアウト中

11.36 CWinCtrlBase

全てのコントロールは、CWinCtrlBase を継承しています。

CWinCtrlBase に各コントロールの共通メソッド/プロパティを持っています。

これを継承したコントロールはそれぞれ独自のプロパティを持つものもありますが、それについては各コントロールの説明を参照ください。

11.36.1 コンテンツについて

ウィンドウスクリプト上で、 **CONTENTS** によって関連付いたコントロールは `getContents()` でアクセス可能です。

CHECKBOX 、 *RADIO* 、 *CONTAINER* 等のコンテンツはこれに該当します。

```
// Get the CONTAINER(Map)
CWinCtrlContainer      ctnMap = find<CWinCtrlContainer>(CONTAINER_Map);
CWinContents contents = ctnMap.getContents();
```

(次のページに続く)

(前のページからの続き)

```
// Get the BUTTON(A) in contents
CWinCtrlButton btnA = contents.find<CWinCtrlButton>(BUTTON_A);
```

また、 *CWindowBase.find* 関数と、 *CWinContents.find* は、コンテンツ内も (再帰的に) 検索対象にします。よって、単純に次のようにも BUTTON_A を取り出すことができます。

```
CWinCtrlButton btnA = find<CWinCtrlButton>(BUTTON_A);
```

11.36.2 コンテンツリストについて

リストボックスはコンテンツを複数組み持つものになります。リストボックスの行を追加するというのは、コンテンツ をテンプレートとして複製するということです。よって、コンテンツリスト内には同一のコントロール ID を持つコントロールが複数存在することになります。特定のコンテンツリスト内のコントロールを取得するには、まず最初にコンテンツリスト内からコンテンツを特定する必要があります。

この時、 `getContentsFromIndex(int index)` を使います。

以下にリストボックス内のコントロールに対してアクセスする典型的なコードを掲載しておきます。

```
// get the listbox control
CWinCtrlListbox     lbFriends = find<CWinCtrlListbox>(LISTBOX_Friends);

// set the listbox contents number
lbFriends.resize(10);

// update each contents in listbox(update 10 contents).
for (int i = 0;i < lbFriends.Count;++i) {
    // get the one column(= contents) from listbox
    CWinContents contents = lbFriends.getContentsFromIndex(i);

    CWInCtrlIcon icon = contents.find<CWinCtrlIcon>(ICON_Avatar);
```

class CWinCtrlBase

コンテンツ関連

CWinContents **getContents**()

return CWinContents *CONTENTS* プロパティによって定義された コンテンツ を取得する。

コンテンツリスト関連

int **count** { **get;** }

コンテンツリストの数を取得する。

void **resize**（int *num*）

Param int num　コンテンツリストサイズ

コンテンツリストの数を設定する。

コンテンツリストの数が増えたときは、コンテンツをテンプレートとして必要な数、複製します。

既にあるコンテンツリストの要素に対しては操作は行いません。

CWinContents **insert**（int *pos*）

Param int pos　コンテンツを作成し、コンテンツリストに挿入する場所

Return CWinContents　コンテンツリスト内の コンテンツ

コンテンツリストの特定の場所にコンテンツを生成し挿入する。

CWinContents **getContentsFromIndex**（int *index*）

Param int index　コンテンツリスト内の取得したいコンテンツのインデックス

Return CWinContents　コンテンツリスト内の コンテンツ

コンテンツリストから、インデックスを用いて取得する。

int **getContentsIndex**（*CWinCtrlBase ctrl*）

特定のコントロールが何番目のコンテンツリストに含まれているか調べて返す。どのコンテンツリストの要素にも入っていなければ-1 が返る。

int **getContentsIndex**（*CWinContents contents*）

特定のコンテンツが何番目のコンテンツリストに含まれているか調べて返す。どのコンテンツリストのコンテンツのどれとも合致しなければ-1 が返る。

その他

string **ToString**（）

コントロールの種類と ID による文字列を生成する。

uint **id** { **get; set;** }

コントロールの ID を取得/設定する。

e_WinCtrlKind **kind** { **get;** }

コントロールの種類を取得する。

int **priority** { **get; set;** }

コントロールのプライオリティを取得/設定する。

Vector3 **position** { **get; set;** }

コントロールの座標を設定する。

Vector2 **absPosition** { **get; set;** }

コントロールの座標をウィンドウ基準から取得

Vector2 **screenPosition** { **get; set;** }

コントロールの座標を画面基準から取得

float **width** { **get; set;** }

float **height** { **get; set;** }

コントロールのサイズを取得/設定する。

Vector2 **size** { **get; set;** }

コントロールのサイズを取得/設定する。

Vector2 **contentsSize** { **get; set;** }

コンテンツのサイズを取得/設定する。

WinColor **getColor**（int *index*）

index = 0〜7 の間で設定する。カラーを取得する。テクスチャパーツを最大 8 まで持てるので、カラーもそれに対応して、8 個持てる。

void **setColor**（int *index*, WinColor *color*）

index = 0〜7 の間で設定する。カラーを設定する。テクスチャパーツを最大 8 まで持てるので、カラーもそれに対応して、8 個持てる。

WinColor **color** { **get; set;** }

テクスチャパーツ番号 0 番のカラーを取得/設定する。

WinColor **color1** { **get; set;** }

WinColor **color2** { **get; set;** }

WinColor **color3** { **get; set;** }

WinColor **color4** { **get; set;** }

WinColor **color5** { **get; set;** }

WinColor **color6** { **get; set;** }

WinColor **color7** { **get; set;** }

テクスチャパーツ番号 1(～7 番) のカラーを取得/設定する。

void **setTextureId**（int *index*, uint *texid*）

index = 0～7 の間で設定する。特定のテクスチャ ID を設定する。テクスチャパーツを最大８まで持てるので、テクスチャ ID もそれに対応して 8 個持てる。

WinColor **getTextureId**（int *index*）

index = 0～7 の間で設定する。特定のテクスチャ ID を取得する。テクスチャパーツを最大８まで持てるので、テクスチャ ID もそれに対応して 8 個持てる。

uint **texId** { **get; set;** }

uint **texId1** { **get; set;** }

テクスチャＩＤを取得/設定する。

void **setPartId**（int *index*, uint *partId*）

index = 0～7 の間で設定する。特定のテクスチャパーツ ID を設定する。テクスチャパーツを最大８まで持てるので、パーツ ID もそれに対応して 8 個持てる。

uint **getPartId**（int *index*）

index = 0～7 の間で設定する。特定のテクスチャパーツ ID を取得する。テクスチャパーツを最大８まで持てるので、パーツ ID もそれに対応して 8 個持てる。

void **setTextureSize**（int *index*, Vector2 *size*）

index = 0～7 の間で設定する。特定のテクスチャパーツの表示サイズを設定する。テクスチャパーツを最大８まで持てるので、サイズもそれに対応して 8 個持てる。特に index = 0 の時、width,height にアクセスするのと同等になる。

uint **parts** { **get; set;** }

uint **partId1** { **get; set;** }

uint **partId2** { **get; set;** }

uint **partId3** { **get; set;** }

uint **partId4** { **get; set;** }

uint **partId5** { **get; set;** }

uint **partId6** { **get; set;** }

uint **partId7** { **get; set;** }

テクスチャパーツＩＤを取得/設定する。

void **setTextureOffset**（int *index*, Vector2 *offset*）

index = 0〜7 の間で設定する。特定のテクスチャパーツの表示オフセットを設定する。テクスチャパーツを最大 8 まで持てるので、オフセットもそれに対応して 8 個持てる。

Vector2 **getTextureOffset**（int *index*）

index = 0〜7 の間で設定する。特定のテクスチャパーツの表示オフセットを取得する。テクスチャパーツを最大 8 まで持てるので、オフセットもそれに対応して 8 個持てる。

e_Anchor **anchor** { **get; set;** }

基準座標を決定するためのアンカーを取得/設定する。

e_Anchor **baseAnchor** { **get; set;** }

コントロールの原点をどこにするかを決定するベースアンカーを取得/設定する。

e_Anchor **textAnchor** { **get; set;** }

キャプションの原点をどこにするかを決定するテキストアンカーを取得/設定する。

uint **fontKind** { **get; set;** }

フォントの種類を取得/設定する。

string **caption** { **get; set;** }

キャプションを取得/設定する。

uint **captionId** { **get;** }

キャプション ID を取得/設定する。ただし、設定しても caption には反映しません。

WinColor **captionColor** { **get; set;** }

キャプションカラーを取得/設定する。

Vector2 **captionOffset** { **get; set;** }

キャプションの表示オフセットをずらす。

float **lineSpace** { **get; set;** }

コンテンツ間のスペースを設定する。またリッチテキストなどでは、行間の値になる。

uint **SEId** { **get; set;** }

コントロールに割り当てられた SE の ID を取得/設定する。

CWinContents **parent** { **get; set;** }

コントロールが含まれているコンテンツを取得する。

CWinCtrlBase[] **groups** { **get;** }

コントロールが持つ GROUP を取得する。

CWindowBase **window** { **get;** }

コントロールが含まれているウィンドウを取得する。

e_WinCtrlStyle **style** { **get;** }

コントロールのスタイルフラグを取得する。

bool **hide** { **get; set;** }

true: コントロールの表示を止める。

false: コントロールを表示する。

bool **disable** { **get; set;** }

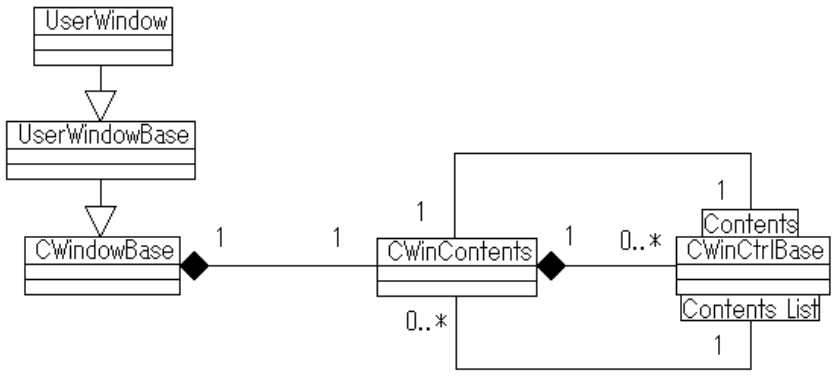
- true: コントロールの機能を奪い暗くする。
- false: コントロールを通常表示する。

bool **nohit** { **get; set;** }

- true: コントロールの機能う (disable と違い暗くしない)。
- false: コントロールを通常表示する。

11.37 CWinContents

class CWinContents



CWinContents は、各種コントロールを保持するためのコンテナとしての役割を持ちます。

CWinContents は以下の場面で使われています。

- CWindowBase 内に一つ持ち、複数のコントロールを保持します。
- CWinCtrlBase 内に一つ持ち、コンテンツと呼びます。 こちら で定義されます。
- CWinCtrlBase 内にリストでもち、コンテンツリストと呼びます。

最初の二つは ウィンドウスクリプト によって構造が定義されます。

三つ目のコンテンツリストは、コントロールが持つコンテンツをテンプレートとして、コピーすることによってリストとして持っています。

固有の関数/プロパティ

WinCtrl **find**<WinCtrl>（uint *ctrlId*）

Param uint ctrlId　コントロール ID

該当する ID のコントロールを探します。もし、コンテンツ内に存在しないときは、コントロールが持つコンテンツ内も再帰的に探します。ただし、コンテンツリスト内は検索対象に含まれません。よって、コントロールは必ず一意に決まります。

int **getIndex**（uint *ctrlId*）

Param uint ctrlId　コントロール ID

指定されたコントロール ID が何番目に格納されているか探します。見つからないときは、-1 が返ります。find と違い、再帰的には検索しません。

返ってきたインデックス値は、this[N] で使え、高速にアクセス可能になります。

CWinCtrlBase **this**`[, int N]` { **get;** }
`getIndex` によって取得したインデックス値を使えます。

int **count** { **get;** }
コンテンツ内のコントロールの数を返します。

ClipRect.State **clipState** { **get;** }
コンテンツリスト内に入っているコンテンツのみで使われます (ex: *LISTBOX*, *LISTBOXEX*)。

ClipRect.State.Inside　完全に領域に入っている

ClipRect.State.Clipped　クリッピングが必要

ClipRect.State.Outside　完全に外に出ていてレンダリング不要

float **width** { **get;** }
コンテンツの横幅を取得します。コントロールのサイズから自動計算されます。

float **height** { **get;** }
コンテンツの縦を取得します。コントロールのサイズから自動計算されます。

Vector2 **size** { **get;** }
コンテンツのサイズを取得します。コントロールのサイズから自動計算されます。

Vector3 **position** { **get;** }
コンテンツの表示位置

CWinCtrlBase **parent** { **get;** }
コンテンツの親になっているコントロールを取得します。

CWindowBase が持つコンテンツの場合、null が返ります。

11.38 CWinCtrlText

スクリプト: *TEXT*

テキストを表示するためのコントロールです。

途中で文字色を変えたりしたいときは、 *CWinCtrlRichText* を使ってください。改行は可能です。

CWindowBase に届くコールバック

このコントロールは、デフォルトで NOHIT になっています。コールバックを受け取りたいときは、STYLE に HIT を指定し忘れないでください。

```
void onClick(CWinCtrlBase cCtrl)

void onHold(CWinCtrlBase cCtrl)

void onBeginDrag(CWinCtrlBase cCtrl,Vector2 pos)

void onDrag(CWinCtrlBase cCtrl,Vector2 pos,Vector2 dragVelocity)

bool onDragRender(CWinCtrlBase cCtrl,Transform transform)

void onDrop(CWinCtrlBase cCtrl,CWindowBase cDragWindow,CWinCtrlBase
cDragCtrl)

void onDropGround(CWinCtrlBase cCtrl)
```

固有の関数/プロパティ

無し

使用例

```
CWinCtrlText   ctrlName = find<CWinCtrlText>(TEXT_Name);
ctrlName.caption = "Mike";
```

11.39 CWinCtrlRichText

スクリプト: *RICHTEXT*

リッチテキストを表示するためのコントロールです。

改行や途中で文字色を変えたりすることが可能です。

CWindowBase に届くコールバック

void onClick(CWinCtrlRichText cCtrl,CRichTextOne cText)

このコールバックが発行されるのは、テクスチャコマンド (\t) とウィンドウコマンド (\w) のみです。CRichTextOne は こちら を参照ください。

void onHold(CWinCtrlBase cCtrl)

void onDrop(CWinCtrlBase cCtrl,CWindowBase cDragWindow,CWinCtrlBase cDragCtrl)

class CRichTextOne

固有の関数/プロパティ

クリックイベントに渡される引数です。メンバを調べることによって、どこをクリックしたかを判定できます。

CRichTextOne.e_Cmd **cmd** { **get;** }

e_Cmd.String

文字列：この部位はクリックイベントが発生しない筈です。

e_Cmd.Window

ウィンドウ：\w コマンド の部位がクリックされた。

次のプロパティに有効な値が入っています。

userId

windowId

e_Cmd.Texture

テクスチャ：\t コマンド の部位がクリックされた。

次のプロパティに有効な値が入っています。

userId

texId

partId

uint **userId** { **get;** }
キャプション内で設定したユーザＩＤが格納されています。ユーザＩＤを省略すると、0 が入ります。

uint **windowId** { **get;** }
\w コマンド で指定したウィンドウ ID を取得できます。

uint **texId** { **get;** }
\t コマンド で指定したテクスチャ ID を取得できます。

uint **partId** { **get;** }
\t コマンド で指定したパーツ ID を取得できます。

11.40 CWinCtrlLog

スクリプト: *LOG*

チャットログのためのコントロールです。

コンテンツを複製し、それを一行とします。

リストボックスと違い、コンテンツの最大数が設定でき、それ以上のコンテンツを追加しようとしたとき、内部で循環 (一番古いものが破棄されます) するようになっています。

LOGTEXT 以外のコントロールもコンテンツとして入れることができます。

CWindowBase に届くコールバック

void onDrag(CWinCtrlBase cCtrl,Vector2 pos,Vector2 dragVelocity)

領域をドラッグしたとき呼び出されます。ロック しているときだけ、このコールバックは呼ばれます。

void onDrop(CWinCtrlBase cCtrl,CWindowBase cDragWindow,CWinCtrlBase cDragCtrl)

class CWinCtrlLog
固有の関数/プロパティ

void **resize**（int *count*）

Param int count　ログのサイズ

Listbox では、コンテンツの数に相当しますが、ここでは保持可能な最大ログ数になります。

int **count** { **get;** }

Listbox では、コンテンツリスト内のコンテンツ数になりますが、ここでは保持しているログの数になります。

resize で設定した値より、大きくなることはありません。

void **clearLog**（）

現在保持しているログを全てクリアにし、空にします。

CWinCtrlLogText **add**（string *text*, WinColor *color*）

Param string text　追加する文字列

Param WinColor color　文字列のカラー

Return CWinCtrlLogText　文字列を設定したログテキストコントロール

ログを追加するために、まずコンテンツを内部的に追加します (若しくは再利用します)。

そのコンテンツ内から、 *CWinCtrlLogText* を探し、そのコントロールのキャプションに指定された文字列とカラーを設定します。

戻り値の *CWinCtrlLogText* から追加されたコンテンツを取得することができます。

```
CWinCtrlLogText ltChat = lgChat.add(strSentence,WinColor.white);
//Access to content that contains the added LogText.
CWinContents contents = ltChat.parent;
```

Vector2 **viewsize** { **get; set;** }

リストボックスのサイズとなります。size プロパティと同等です。

Vector2 **screensize** { **get; set;** }

リストボックスの仮想画面のサイズです。コンテンツリスト内の全ての高さの総和になります。

Vector2 **offset** { **get; set;** }

リストボックスが仮想画面のどこをポイントしているかを表しています。値を設定すると一気に移動します。滑らかにアニメーションさせながら移動させたいときは、 *setSmoothOffset* を使ってください。

bool **isSwipe** { **get;** }

スワイプ中かどうか判定します。

void **setSmoothOffset**（Vector2 *offset*, float *spd*）

Param Vector2 offset　設定したい最終オフセット

Param float spd　速度

Vector2 **smoothOffset** { **get; set;** }

0.25 の速度で、 *setSmoothOffset* したのと同じ振る舞いになります。

bool **isSmoothScrolled** { **get;** }

setSmoothOffset によるスクロールアニメーションかどうかを取得できます。

Vector2 **getContentsOffset**（int *index*, e_Anchor *eAnchor*）

Param int index　取得したいコンテンツオフセットのインデックス

Param e_Anchor eAnchor　コンテンツがどの位置に来ることを望むかをアンカーで指定する

特定のコンテンツが指定したアンカー位置に沿って表示されるようなオフセットを取得できます。

リストボックスが縦方向の時は次の通りです。

e_Anchor	位置
Bottom,LeftBottom,RightBottom	コンテンツが一番下に来るようなオフセット値
Top,LeftTop,RightTop	コンテンツが一番上に来るようなオフセット値
Center,Left,Right	コンテンツが真ん中に来るようなオフセット値

リストボックスが横方向の時は次の通りです。

e_Anchor	位置
Left,LeftBottom,LeftTop	コンテンツが一番左に来るようなオフセット値
Right,RightBottom,RightTop	コンテンツが一番右に来るようなオフセット値
Center,Top,Bottom	コンテンツが真ん中に来るようなオフセット値

使用例

```
CWinCtrlLog    lgChat = find<CWinCtrlLog>(LOG_Chat);

lgChat.resize(10);

string strSentence;
CWinCtrlLogText ltChat = lgChat.add(strSentence,WinColor.white);

CWinContents contents = ltChat.parent;
```

ログをクリアする方法

```
//log clear
lgChat.clearLog();
```

11.41 CWinCtrlLogText

スクリプト: *LOGTEXT*

チャットログの一文を保持するためのコントロールです。ほぼ、振る舞いは *TEXT* と同じです。

注釈: *LOG* 内の *CONTENTS* には、必ず一つの *LOGTEXT* を入れておく必要があります。

CWindowBase に届くコールバック

このコントロールは、デフォルトで NOHIT になっています。コールバックを受け取りたいときは、STYLE に HIT を指定し忘れないでください。

```
void onClick(CWinCtrlBase cCtrl)

void onHold(CWinCtrlBase cCtrl)

void onBeginDrag(CWinCtrlBase cCtrl,Vector2 pos)

void onDrag(CWinCtrlBase cCtrl,Vector2 pos,Vector2 dragVelocity)

bool onDragRender(CWinCtrlBase cCtrl,Transform transform)

void onDrop(CWinCtrlBase cCtrl,CWindowBase cDragWindow,CWinCtrlBase
cDragCtrl)

void onDropGround(CWinCtrlBase cCtrl)
```

固有の関数/プロパティ

無し

使用例

CWinCtrlLog の使用例 を参照

11.42 CWinCtrlEditbox

スクリプト: *EDITBOX*

編集可能なエディットボックスの為のコントロールです。

行数指定及び、文字列の最大サイズを指定可能です。

CWindowBase に届くコールバック

void onClick(CWinCtrlBase cCtrl)

void onDrag(CWinCtrlBase cCtrl,Vector2 pos,Vector2 dragVelocity)

void onClickEnter(CWinCtrlBase cCtrl)

文字が確定したとき呼び出されます。入力文字は、caption に格納されています。

void onDrop(CWinCtrlBase cCtrl,CWindowBase cDragWindow,CWinCtrlBase cDragCtrl)

このコールバックは、cCtrl が含まれるウィンドウにコールバックが届く。

使用例

```
// get a control.
CWinCtrlEditbox        ebSentence = find<CWinCtrlEditbox>(EDITBOX_Sentence);

// get entered text.
string strSentence = ebSentence.caption;


// Called when input is complete..
override protected void onClickEnter(CWinCtrlBase cCtrl) {
    switch (cCtrl.id) {
      case EDITBOX_Sentence:
        break;
    }
}
```

11.43 CWinCtrlTextbox

スクリプト: *TEXTBOX*

編集できないエディットボックスです。

キャプションが、コントロールサイズに収まらないときは、テキストボックスをタッチするとスクロールします。

CWindowBase に届くコールバック

```
void onClick(CWinCtrlBase cCtrl)

void onDrag(CWinCtrlBase cCtrl,Vector2 pos,Vector2 dragVelocity)

void onDrop(CWinCtrlBase cCtrl,CWindowBase cDragWindow,CWinCtrlBase
cDragCtrl)
```

固有の関数/プロパティ

無し

使用例

11.44 CWinCtrlButton

スクリプト: *BUTTON*

ボタンを実現するためのコントロールです。

一つのキャプションと、複数のバッジを同時に表示することが可能です。

バッジは最大 7 つまで対応可能です。

CWindowBase に届くコールバック

```
void onClick(CWinCtrlBase cCtrl)

void onHold(CWinCtrlBase cCtrl)

void onBeginDrag(CWinCtrlBase cCtrl,Vector2 pos)

void onDrag(CWinCtrlBase cCtrl,Vector2 pos,Vector2 dragVelocity)

bool onDragRender(CWinCtrlBase cCtrl,Transform transform)

void onDrop(CWinCtrlBase cCtrl,CWindowBase cDragWindow,CWinCtrlBase
cDragCtrl)

void onDropGround(CWinCtrlBase cCtrl)
```

使用例

```
// get control
CWinCtrlButton ctrlStart = find<CWinCtrlButton>(BUTTON_Start);

// click call back
override protected void onClick(CWinCtrlBase cCtrl) {
 switch (cCtrl.id) {
  case BUTTON_Start:
    break;
 }
// hold callback.
override protected void onHold(CWinCtrlBase cCtrl) {
 switch (cCtrl.id) {
  case BUTTON_Start:
    break;
 }
```

11.45 CWinCtrlRadio

スクリプト: *RADIO*

ラジオボタンとは、複数のラジオボタン同士をグループ化し、一つが ON になるとグループ化されたラジオボタンが自動的に OFF になるボタンです。

CHECKBOX 同様、*CONTENTS* を持つことができ、タブボタンとして振る舞うことも可能です。

CWindowBase に届くコールバック

void onClick(CWinCtrlBase cCtrl)

void onHold(CWinCtrlBase cCtrl)

void onBeginDrag(CWinCtrlBase cCtrl,Vector2 pos)

void onDrag(CWinCtrlBase cCtrl,Vector2 pos,Vector2 dragVelocity)

bool onDragRender(CWinCtrlBase cCtrl,Transform transform)

void onDrop(CWinCtrlBase cCtrl,CWindowBase cDragWindow,CWinCtrlBase cDragCtrl)

void onDropGround(CWinCtrlBase cCtrl)

class CWinCtrlRadio

固有の関数/プロパティ

bool **check** { **get; set;** }

ラジオボタンの状態を変更します。

- true:

  ON 状態にし、グループ化されているラジオボタンに対して OFF 状態にする。関連付いているコンテンツがあれば、アクティブにします。

- false:

  OFF 状態にし、関連付いているコンテンツがあれば、非アクティブにし表示を抑制します。

使用例

```
// get control
CWinCtrlRadio  rdSelect = find<CWinCtrlRadio>(RADIO_Select);

if (rdSelect.check) {
  //ON
} else {
```

(次のページに続く)

(前のページからの続き)

```
  //OFF
}

// click callback
override protected void onClick(CWinCtrlBase cCtrl) {
    switch (cCtrl.id) {
      case RADIO_Select:
        break;
    }
}
// hold callback
override protected void onHold(CWinCtrlBase cCtrl) {
    switch (cCtrl.id) {
      case RADIO_Select:
        break;
    }
}
```

11.46 CWinCtrlCheckbox

スクリプト: *CHECKBOX*

チェックボックスボタンを実現するためのコントロールです。

ボタン同様、７つまでバッジを表示することができます。

コンテンツ を持つことができ、ON の時だけ関連付いたコンテンツをアクティブにすることも可能です。

CWindowBase に届くコールバック

void onClick(CWinCtrlBase cCtrl)

void onHold(CWinCtrlBase cCtrl)

void onBeginDrag(CWinCtrlBase cCtrl,Vector2 pos)

void onDrag(CWinCtrlBase cCtrl,Vector2 pos,Vector2 dragVelocity)

bool onDragRender(CWinCtrlBase cCtrl,Transform transform)

void onDrop(CWinCtrlBase cCtrl,CWindowBase cDragWindow,CWinCtrlBase cDragCtrl)

void onDropGround(CWinCtrlBase cCtrl)

class CWinCtrlCheckBox

bool **check** { **get; set;** }

チェックボックスボタンの状態を変更します。

- true:

  ON 状態にし、関連付いているコンテンツがあれば、アクティブにします。

- false:

  OFF 状態にし、関連付いているコンテンツがあれば、非アクティブにし表示を抑制します。

使用例

```
// get control
CWinCtrlCheckbox       cbSelect = find<CWinCtrlCheckbox>(CHECKBOX_Select);

if (cbSelect.check) {
  //on
} else {
  //off
}
// Get BUTTON(A) from within content
// Content in the container can be accessed by find as follows
CWinCtrlButton btnA = find<CWinCtrlButton>(BUTTON_A);

// click callback
override protected void onClick(CWinCtrlBase cCtrl) {
    switch (cCtrl.id) {
      case CHECKBOX_Select:
        break;
    }
}
 // hold callback
override protected void onHold(CWinCtrlBase cCtrl) {
    switch (cCtrl.id) {
      case CHECKBOX_Select:
        break;
    }
}
```

11.47 CWinCtrlTexture

スクリプト: *TEXTURE*

テクスチャパーツを表示するためのコントロールです。

最大 8 枚まで重ねて表示することが可能です。

CWindowBase に届くコールバック

このコントロールは、デフォルトで NOHIT になっています。コールバックを受け取りたいときは、STYLE に HIT を指定し忘れないでください。

```
void onClick(CWinCtrlBase cCtrl)

void onHold(CWinCtrlBase cCtrl)

void onBeginDrag(CWinCtrlBase cCtrl,Vector2 pos)

void onDrag(CWinCtrlBase cCtrl,Vector2 pos,Vector2 dragVelocity)

bool onDragRender(CWinCtrlBase cCtrl,Transform transform)

void onDrop(CWinCtrlBase cCtrl,CWindowBase cDragWindow,CWinCtrlBase
cDragCtrl)

void onDropGround(CWinCtrlBase cCtrl)
```

固有の関数/プロパティ

無し

使用例

```
// get control
CWinCtrlTexture        texAttribute = find<CWinCtrlTexture>(TEXTURE_Attribute);

// click callback
override protected void onClick(CWinCtrlBase cCtrl) {
  switch (cCtrl.id) {
    case TEXTURE_Attribute:
      break;
   }
}
// hold callback
override protected void onHold(CWinCtrlBase cCtrl) {
   switch (cCtrl.id) {
     case TEXTURE_Attribute:
      break;
    }
}
```

11.48 CWinCtrlLine

スクリプト: *LINE*

線分を引くためのコントロールです。

CWindowBase に届くコールバック

このコントロールは、デフォルトで NOHIT になっています。コールバックを受け取りたいときは、STYLE に HIT を指定し忘れないでください。

void onClick(CWinCtrlBase cCtrl)

void onHold(CWinCtrlBase cCtrl)

void onBeginDrag(CWinCtrlBase cCtrl,Vector2 pos)

void onDrag(CWinCtrlBase cCtrl,Vector2 pos,Vector2 dragVelocity)

bool onDragRender(CWinCtrlBase cCtrl,Transform transform)

void onDrop(CWinCtrlBase cCtrl,CWindowBase cDragWindow,CWinCtrlBase cDragCtrl)

void onDropGround(CWinCtrlBase cCtrl)

使用例

```
// get control
CWinCtrlLine   lnTest = find<CWinCtrlLine>(LINE_TEST);
```

11.49 CWinCtrlRender

スクリプト: *RENDER*

レンダーテクスチャに対して自由にレンダリングするためのコントロールです。

コントロールが生成されたとき、一緒にカメラも生成されます。

そのカメラにレンダリングした対象が描画されることになります。

注釈: **RENDER** を用いてレンダリングするためのリソースを確保したときは、クローズ時には忘れずに削除するようにしてください。

削除し忘れると、ウィンドウクローズの度にリークすることになります。

CWindowBase に届くコールバック

void onClick(CWinCtrlBase cCtrl)

void onHold(CWinCtrlBase cCtrl)

void onBeginDrag(CWinCtrlBase cCtrl,Vector2 pos)

void onDrag(CWinCtrlBase cCtrl,Vector2 pos,Vector2 dragVelocity)

bool onDragRender(CWinCtrlBase cCtrl,Transform transform)

void onDrop(CWinCtrlBase cCtrl,CWindowBase cDragWindow,CWinCtrlBase cDragCtrl)

void onDropGround(CWinCtrlBase cCtrl)

class CWinCtrlRender

固有の関数/プロパティ

Camera **camera** { **get;** }

レンダリングテクスチャが割り当てられたカメラを取得できます。カメラの位置や角度等をカスタマイズしたいときにアクセスしてください。

注釈: 描画対象オブジェクトとカメラのレイヤーの設定に矛盾が出ないように気を付けてください。また、複数の RENDER コントロールが同時に画面にできるときは、それぞれの RENDER でレイヤー設定が重複しないように気を付けて割り当てる必要があります。

使用例

```
// get control
public override void onCreate() {
  CWinCtrlRender       rdAvatar = find<CWinCtrlRender>(RENDER_AVATAR);

  // get camera
  Camera       camera = rdAvatar.camera;

  // set display layer.
  camera.cullingMask = (int) e_Layer.OwnPlayer;

  // set camera position and angle.
  camera.transform.position = new Vector3(0f,0.85f,0f);
  camera.transform.rotation = Quaternion.Euler(new Vector3(0f,180f,0f));

  GameObject goAvatar = LoadAvatar();
  // Sets the layer of game objects to render (Make Same as Camera).
  Utility.setLayer(goAvatar,e_LayerId.OwnPlayer);
}

// click callback
override protected void onClick(CWinCtrlBase cCtrl) {
  switch (cCtrl.id) {
```

(次のページに続く)

(前のページからの続き)

```
    case RENDER_AVATAR:
      break;
   }
}
// hold callback
override protected void onHold(CWinCtrlBase cCtrl) {
   switch (cCtrl.id) {
     case RENDER_AVATAR:
      break;
    }
}
```

11.50 CWinCtrlIcon

スクリプト: *ICON*

アイコンを表示するためのコントロールです。

最大 8 枚のテクスチャパーツを重ねて表示することが可能です。

TEXTURE コントロールと振る舞いはほぼ同じですが、タッチしたとき暗くなります。

また、 *TEXTURE* コントロールと違い、キャプションが持てます。

CWindowBase に届くコールバック

void onClick(CWinCtrlBase cCtrl)

void onHold(CWinCtrlBase cCtrl)

void onBeginDrag(CWinCtrlBase cCtrl,Vector2 pos)

void onDrag(CWinCtrlBase cCtrl,Vector2 pos,Vector2 dragVelocity)

bool onDragRender(CWinCtrlBase cCtrl,Transform transform)

void onDrop(CWinCtrlBase cCtrl,CWindowBase cDragWindow,CWinCtrlBase cDragCtrl)

void onDropGround(CWinCtrlBase cCtrl)

使用例

```
// get cotrol
CWinCtrlIcon   icnItem = find(ICON_Item) as CWinCtrlIcon;

// click callback
```

(次のページに続く)

(前のページからの続き)

```
override protected void onClick(CWinCtrlBase cCtrl) {
  switch (cCtrl.id) {
    case ICON_Item:
      break;
   }
}
// hold callback
override protected void onHold(CWinCtrlBase cCtrl) {
    switch (cCtrl.id) {
      case ICON_Item:
        break;
    }
}
```

11.51 CWinCtrlRecastIcon

スクリプト: *RECASTICON*

リキャストアイコンを表示するためのコントロールです。

最大 8 枚のテクスチャパーツを重ねて表示することが可能です。

ICON とほぼ同じ振る舞いを行います。

リキャスト設定を行うと、指定したテクスチャパーツが、下から徐々に充填されるようなアニメーションを行います。リキャスト時間は、0 の時、*TEXTURE* と同等の表示を行い、1 以上の時何も表示されないようになっています。

CWindowBase に届くコールバック

void onClick(CWinCtrlBase cCtrl)

void onHold(CWinCtrlBase cCtrl)

void onBeginDrag(CWinCtrlBase cCtrl,Vector2 pos)

void onDrag(CWinCtrlBase cCtrl,Vector2 pos,Vector2 dragVelocity)

bool onDragRender(CWinCtrlBase cCtrl,Transform transform)

void onDrop(CWinCtrlBase cCtrl,CWindowBase cDragWindow,CWinCtrlBase cDragCtrl)

void onDropGround(CWinCtrlBase cCtrl)

class CWinCtrlRecastIcon

固有の関数/プロパティ

CInput::e_State **state** { **get;** }

アイコンが押されている状態を取得できます。

void **setRecastTime**（int *index*, float *tmRecast*）

Param int index　リキャストアニメーションを行うテクスチャパーツ番号

Param float tmRecast　アニメーション時間 [0〜1]

リキャスト時間を設定します。

public float **getRecastTime**（int *index*）

Param int index　リキャストアニメーションを行うテクスチャパーツ番号

Return float　残りアニメーション時間

リキャスト時間を取得します。

使用例

```
// get control
CWinCtrlRecastIcon     rcItem = find(RECASTICON_Item) as CWinCtrlRecastIcon;

// Recast Display (Display half of the texture of TEX_ID3)
rcItem.setRecastTime(3,0.5f);

// click callback
override protected void onClick(CWinCtrlBase cCtrl) {
  switch (cCtrl.id) {
    case RECASTICON_Item:
      break;
   }
}
// hold callback
override protected void onHold(CWinCtrlBase cCtrl) {
    switch (cCtrl.id) {
      case ICON_Item:
        break;
    }
}
```

11.52 CWinCtrlRenderIcon

スクリプト: *RENDERICON*

アイコンにレンダリングするためのコントロールです。

指定されたレンダーテクスチャから自動的に矩形を切り出し、レンダリングします。

複数のレンダー結果を一枚のレンダーテクスチャ内に含めることができるため、一度レンダリングした後のアイコンの描画が高速です。

例えば、リストボックス内のコンテンツに着せ替え可能アバターアイコンを表示したいときなどに使うと効果的です。

CWindowBase に届くコールバック

void onClick(CWinCtrlBase cCtrl)

void onHold(CWinCtrlBase cCtrl)

void onBeginDrag(CWinCtrlBase cCtrl,Vector2 pos)

void onDrag(CWinCtrlBase cCtrl,Vector2 pos,Vector2 dragVelocity)

bool onDragRender(CWinCtrlBase cCtrl,Transform transform)

void onDrop(CWinCtrlBase cCtrl,CWindowBase cDragWindow,CWinCtrlBase cDragCtrl)

void onDropGround(CWinCtrlBase cCtrl)

bool onBeginRenderIcon(CWinCtrlRenderIcon icon)

アイコンにレンダリング開始

この関数の戻り値は、次の意味を持ちます。

- true を返すとテクスチャにレンダリングします。
- false を返すとレンダリングしません。 *onEndRenderIcon* も呼ばれません。

bool onEndRenderIcon(CWinCtrlRenderIcon icon)

レンダリングが終了したときに呼ばれる

この関数の戻り値は、次の意味を持ちます。

- true を返すと、次のフレームもレンダリングを継続します (*onBeginRenderIcon* が呼ばれる)
- false を返すと、以降のフレームでは必要がない限り、レンダリングされません。

一度停止させたレンダリングを再開したいときは、 *needToRender* を true にしてください。

class CWinCtrlRenderIcon

固有の関数/プロパティ

ulong **regionId** { **get; set;** }

0 以外の固有の ID を割り振ってください。0 を代入すると確保されたレンダリング領域が解放されます。また、他のコントロールと同じ値を割り振ると、レンダリング領域を共有します。

Camera **camera** { **get;** }

レンダリングテクスチャが割り当てられたカメラを取得できます。カメラの位置や角度等をカスタマイズしたいときにアクセスしてください。

WinColor **color1** { **get; set;** }

レンダリングする際の背景色として割り当てられています。

デフォルトは、(0,0,0,0) です。

bool **needToRender** { **get; set;** }

レンダリングを再開したいとき、true を設定してください。

次のフレームでは *onBeginRenderIcon* が呼び出されます。

CInput::e_State **state** { **get;** }

アイコンが押されている状態を取得できます。

使用例

onBeginRenderIcon によって、レンダリングしたいオブジェクトのみアクティブにしています。そして、 *onEndRenderIcon* によって、非アクティブにしています。

この操作をしないと全てのオブジェクトが一つのアイコンにレンダリングされることになります。

また、 *onBeginRenderIcon* が true を返すと、直ちに Camera.Render() が呼び出されるので、アニメーションなどの設定をしたときは、下記の例のように、Animation.Sample() を明示的に呼んでおく必要があります。

正しくレンダリングされないときは、以下の点をチェックしてみて下さい。

- レイヤーの設定がおかしくないか？/レイヤーの設定を忘れていないか?
- カメラにレンダリング対象が収まっているか？

  当然、カメラに収まっていないとオブジェクトはレンダリングされません。

- *regionId* を設定したか？

  regionId が 0 の時は、レンダーテクスチャへのレンダリングが行われません。*regionId* をキーにしてレンダーテクスチャの空いている領域から矩形を切り出し、レンダリングします。

```
override public bool onBeginRenderIcon(CWinCtrlRenderIcon icon) {
    CWinCtrlListbox    lbAvatar = lbAvatar = find(CWinCtrlListbox>(LISTBOX_Avatar);
    int idx = lbAvatar.getContentsIndex(icon);
    if (idx < 0) {
        return false;
    }
```

(次のページに続く)

(前のページからの続き)

```
    GameObject go = m_lstGameObject[idx];
    if (go == null) {
       return false;
    }
    go.SetActive(true);
    Animation anim = go.GetComponent<Animation>();
    anim.Sample();

    return true;
}
override public bool onEndRenderIcon(CWinCtrlRenderIcon icon) {
    CWinCtrlListbox    lbAvatar = lbAvatar = find(CWinCtrlListbox>(LISTBOX_Avatar);
    int idx = lbAvatar.getContentsIndex(icon);

    GameObject go = m_lstGameObject[idx];
    if (go == null) {
       return false;
    }
    go.SetActive(false);
    return false;
}

// click callback
override public void onClick(CWinCtrlBase cCtrl) {
  switch (cCtrl.id) {
    case RENDERICON_Avatar:
      break;
   }
}
// hold callback
override public void onHold(CWinCtrlBase cCtrl) {
   switch (cCtrl.id) {
     case RENDERICON_Avatar:
      break;
    }
}
```

各レイヤーの初期値

Camera.cullingMask = e_Layer.RenderIcon;

特に必要が無ければ触らないようにしておいてください。

レンダリングするときは、レイヤー設定をしているかしっかり確認してください。

レンダリング対象のゲームオブジェクトも上記値を設定しておくようにしておいてください。

```
// If gameobjectAvatar is not multiple hierarchies, the following is sufficient.
gameobjectAvatar.layer = (int) e_LayerId.RenderIcon;
// If gameobjectAvatar is made up of multiple hierarchies, it must be recursively set␣
↪using the following function:
Utility.setLayer(gameObjectAvatar,e_LayerId.RenderIcon);
gameobjectAvatar.SetActive(false);
```

注釈: GameObject.layer は、一つの階層に対してのみレイヤーを設定します。子供の階層にまで影響を与えません。レンダーアイコンが持つカメラに選択的に子供の階層を持つオブジェクトを正しくレンダリングするためには、本アセットが用意しているレイヤーを再帰的に設定可能な、Utility.setLayer を使ってください。

レンダリングの主な流れ

RenderIcon は、複数のモデルをテクスチャにレンダリングして表示するのに適しています。ただし、レイヤーの数が限りがあるので、一つのレイヤーを使いまわすことを考える必要があります。

この問題を回避するために次のような手順でレンダリングします。

1. モデルの読み込み
2. 読み込み終わったら、SetActive(false) にしてゲームオブジェクトを非アクティブにします
3. レイヤーを RenderIcon に設定します。
4. *regionId* に、ユニークな値を設定します。
5. OnBeginRenderIcon が呼ばれたら、対象のゲームオブジェクトをアクティブにします
6. OnEndRenderIcon が呼ばれたら、対象のゲームオブジェクトを非アクティブにします

これによって必ず RenderIcon にレンダリングされるゲームオブジェクトを一つに絞ることができます。OnEndRenderIcon の戻り値は、false を返しておく方がパフォーマンスが出ます。そうしないと、毎フレーム、テクスチャへのレンダリングが発生してしまいます。

onBeginRender が呼ばれるタイミング

- *regionId* に 0 以外の値が設定されたとき.

  regionId に 0 以外のユニークな値を設定した直後から、 *onBeginRenderIcon* が呼ばれます。*onBeginRenderIcon* が true を返すまで呼ばれ続けます。false を返した時は、 *onEndRenderIcon* は呼び出されません。この仕組みを使ってモデルの非同期読み込み途中は、false を返し、レンダリング準備ができたタイミングでモデルをテクスチャにレンダリングすることが可能です (レイヤーの設定は忘れないでください)。

- レンダーテクスチャ消失後

レンダーテクスチャが消失する理由としては、スクリーンセーバが走ったり、ウィンドウが最小化されたときなどが考えられます。レンダーテクスチャは破壊されることがあるので、常にテクスチャへの再レンダーができるようにしておくべきです。

- *needToRender* に true を設定した後
- *onEndRenderIcon* が true を返した次のフレーム

毎フレームレンダリングしたいときは、true を返してください。ただし、パフォーマンスには気を付けてください。

11.53 CWinCtrlMeter

スクリプト: *METER*

メータを表示するためのコントロールです。

プログレスバーなどに使えます。

TEXTURE と同様に最大 8 枚まで重ねて表示することが可能です。

任意のテクスチャパーツの伸縮を設定可能です。

長さを 1 に設定すると、ちょうど *TEXTURE* と同じ見た目で表示されるようになっています。

CWindowBase に届くコールバック

METER は、デフォルトで NOHIT になっています。コールバックを受け取りたいときは、STYLE に HIT を指定し忘れないでください。

```
void onClick(CWinCtrlBase cCtrl)

void onHold(CWinCtrlBase cCtrl)

void onBeginDrag(CWinCtrlBase cCtrl,Vector2 pos)

void onDrag(CWinCtrlBase cCtrl,Vector2 pos,Vector2 dragVelocity)

bool onDragRender(CWinCtrlBase cCtrl,Transform transform)

void onDrop(CWinCtrlBase cCtrl,CWindowBase cDragWindow,CWinCtrlBase
cDragCtrl)

void onDropGround(CWinCtrlBase cCtrl)
```

class CWinCtrlMeter

固有の関数/プロパティ

float **meter**（int *index*）

Param int index　テクスチャパーツ番号

Return float　メータの長さ

指定されたテクスチャ番号のパーツが元のサイズから何割の長さに設定されているか取得する.

0 ～ 1 で値が返ってきます。

デフォルトでは、1 が設定されています (メータがマックスの状態)

void **setMeter**（int *index*, float *value*, float *speed* = *0.2f*）

Param int index　テクスチャパーツ番号

Param float value　メータの長さ (0～1)

Param float speed = 0.2f　アニメーション速度

指定されたテクスチャ番号のパーツのサイズをアニメーション付で変更します。

使用例

```
// get control
CWinCtrlMeter  mtrLoading = find(METER_Loading) as CWinCtrlMeter;

// Get Meter Position (0 … 1.)
float pos = mtrLoading.meter(3);

// Set 3rd texture part stretch (It can be specified as 0 … 1.)
mtrLoading.setMeter(3,0.5f);
```

11.54 CWinCtrlScrollbar

スクリプト: *SCROLLBAR*

スクロールバーを表示するためのコントロールです。

主にリストボックスの *GROUP* に設定することによって自動的に表示を制御してくれます。

CWindowBase に届くコールバック

無し

class CWinCtrlScrollbar

固有の関数/プロパティ

int **index** { **get; set;** }

スクロールバーとして振る舞うテクスチャインデックスを指定します。

この値で指定したテクスチャ以外は、長さや位置の変更なしに表示されます。

float **fadeSpeed** { **get; set;** }

スクロールバーの 表示条件 から外れたときに、消える速度を指定します。

使用例

```
CWinCtrlScrollbar     sbListbox = find<CWinCtrlScrollbar>(SCROLLBAR_Listbox);
```

11.55 CWinCtrlListbox

スクリプト: *LISTBOX*

リストボックスのためのコントロールです。

コンテンツを複製し、それを一行とします。

コンテンツの数はメモリが許す限り自由に設定可能です。

垂直方向にコンテンツが並んでいるリストボックスだけでなく、水平方向にも並べることも可能です。

CWindowBase に届くコールバック

void onDrag(CWinCtrlBase cCtrl,Vector2 pos,Vector2 dragVelocity)

領域をドラッグしたとき呼び出されます。ロック しているときだけ、このコールバックは呼ばれます。

void onDrop(CWinCtrlBase cCtrl,CWindowBase cDragWindow,CWinCtrlBase cDragCtrl)

class CWinCtrlListBox

固有の関数/プロパティ

Vector2 **viewsize** { **get; set;** }

リストボックスのサイズとなります。size プロパティと同等です。

Vector2 **screensize** { **get; set;** }

リストボックスの仮想画面のサイズです。コンテンツリスト内の全ての高さの総和になります。

Vector2 **offset** { **get; set;** }

リストボックスが仮想画面のどこをポイントしているかを表しています。値を設定すると一気に移動します。滑らかにアニメーションさせながら移動させたいときは、 *setSmoothOffset* を使ってください。

bool **isSwipe** { **get;** }

スワイプ中かどうか判定します。

void **setSmoothOffset**（Vector2 *offset*, float *spd*）

Param Vector2 offset　設定したい最終オフセット

Param float spd　速度

Vector2 **smoothOffset** { **get; set;** }

0.25 の速度で、 *setSmoothOffset* したのと同じ振る舞いになります。

bool **isSmoothScrolled** { **get;** }

setSmoothOffset によるスクロールアニメーションかどうかを取得できます。

Vector2 **getContentsOffset**（int *index*, e_Anchor *eAnchor*）

Param int index　取得したいコンテンツオフセットのインデックス

Param e_Anchor eAnchor　コンテンツがどの位置に来ることを望むかをアンカーで指定する

特定のコンテンツが指定したアンカー位置に沿って表示されるようなオフセットを取得できます。

リストボックスが縦方向の時は次の通りです。

e_Anchor	位置
Bottom,LeftBottom,RightBottom	コンテンツが一番下に来るようなオフセット値
Top,LeftTop,RightTop	コンテンツが一番上に来るようなオフセット値
Center,Left,Right	コンテンツが真ん中に来るようなオフセット値

リストボックスが横方向の時は次の通りです。

e_Anchor	位置
Left,LeftBottom,LeftTop	コンテンツが一番左に来るようなオフセット値
Right,RightBottom,RightTop	コンテンツが一番右に来るようなオフセット値
Center,Top,Bottom	コンテンツが真ん中に来るようなオフセット値

使用例

```
override public void onUpdate() {
   // get control
   CWinCtrlListbox     lbFriends = find<CWinCtrlListbox>(LISTBOX_Friends);

   // set list box contents num.
   lbFriends.resize(10);

   // Refresh the status in a list box (In this case, 10 lines will be updated.)
```

(次のページに続く)

(前のページからの続き)

```
    for (int i = 0;i < lbFriends.Count;++i) {
        // Get a single line of list box content.
        CWinContents contents = lbFriends.getContentsFromIndex(i);

        //Get BUTTON(Name)
        CWinCtrlButton btnName = contents.find<CWinCtrlButton>(BUTTON_Name);

        //Get ICON(Icon)
        CWinCtrlIcon icnName = contents.find<CWinCtrlIcon>(ICON_Icon);
    }
}
```

11.56 CWinCtrlListboxEx

スクリプト: *LISTBOXEX*

リストボックスのためのコントロールです。

コンテンツを複製し、それを一行とします。

コンテンツの数はメモリが許す限り自由に設定可能です。

垂直方向にコンテンツが並んでいるリストボックスだけでなく、水平方向にも並べることも可能です。

LISTBOX と違う点は、コンテンツサイズを自動的に計測するところです。

垂直方向に並んでいる場合は、縦幅のみ自動計算の対象になります。

それに対して水平方向に並んでいる場合は、横幅のみ自動計算の対象になります。

CWindowBase に届くコールバック

void onDrag(CWinCtrlBase cCtrl,Vector2 pos,Vector2 dragVelocity)

領域をドラッグしたとき呼び出されます。ロック しているときだけ、このコールバックは呼ばれます。

void onDrop(CWinCtrlBase cCtrl,CWindowBase cDragWindow,CWinCtrlBase cDragCtrl)

class CWinCtrlListboxEx

固有の関数/プロパティ

Vector2 **viewsize** { **get; set;** }

リストボックスのサイズとなります。size プロパティと同等です。

Vector2 **screensize** { **get; set;** }

リストボックスの仮想画面のサイズです。コンテンツリスト内の全ての高さの総和になります。

Vector2 **offset** { **get; set;** }

リストボックスが仮想画面のどこをポイントしているかを表しています。値を設定すると一気に移動します。滑らかにアニメーションさせながら移動させたいときは、 `setSmoothOffset` を使ってください。

bool **isSwipe** { **get;** }

スワイプ中かどうか判定します。

void **setSmoothOffset**（Vector2 *offset*, float *spd*）

Param Vector2 offset　設定したい最終オフセット

Param float spd　速度

Vector2 **smoothOffset** { **get; set;** }

0.25 の速度で、 `setSmoothOffset` したのと同じ振る舞いになります。

bool **isSmoothScrolled** { **get;** }

setSmoothOffset によるスクロールアニメーションかどうかを取得できます。

Vector2 **getContentsOffset**（int *index*, e_Anchor *eAnchor*）

Param int index　取得したいコンテンツオフセットのインデックス

Param e_Anchor eAnchor　コンテンツがどの位置に来ることを望むかをアンカーで指定する

特定のコンテンツが指定したアンカー位置に沿って表示されるようなオフセットを取得できます。

リストボックスが縦方向の時は次の通りです。

e_Anchor	位置
Bottom,LeftBottom,RightBottom	コンテンツが一番下に来るようなオフセット値
Top,LeftTop,RightTop	コンテンツが一番上に来るようなオフセット値
Center,Left,Right	コンテンツが真ん中に来るようなオフセット値

リストボックスが横方向の時は次の通りです。

e_Anchor	位置
Left,LeftBottom,LeftTop	コンテンツが一番左に来るようなオフセット値
Right,RightBottom,RightTop	コンテンツが一番右に来るようなオフセット値
Center,Top,Bottom	コンテンツが真ん中に来るようなオフセット値

使用例

```
override public void onUpdate() {
   // get control
   CWinCtrlListboxEx   lbPartys = find<CWinCtrlListboxEx>(LISTBOXEX_Partys);

   // set listbox contents num
   lbPartys.resize(10);

   // Refresh the status in a list box (In this case, 10 lines will be updated.)
   for (int i = 0;i < lbFriends.Count;++i) {
       // Get a single line of list box content.
       CWinContents contents = lbFriends.getContentsFromIndex(i);

       //Get BUTTON(Name)
       CWinCtrlButton btnName = contents.find<CWinCtrlButton>(BUTTON_Name);

       //Get ICON(Icon)
       CWinCtrlIcon icnName = contents.find<CWinCtrlIcon>(ICON_Icon);
   }
}
```

11.57 CWinCtrlContainer

スクリプト: *CONTAINER*

複数のコントロールを纏めるためのコントロールです。

SIZE によって、コンテナのサイズが決定し、CONTENTS_SIZE によって仮想画面のサイズが決定します。

仮想画面サイズがサイズよりも大きいときは、内包しているコントロールをスクロールさせることができます。

CWindowBase に届くコールバック

void onDrag(CWinCtrlBase cCtrl,Vector2 pos,Vector2 dragVelocity)

ロック しているときだけ、このコールバックは呼ばれます。

void onDrop(CWinCtrlBase cCtrl,CWindowBase cDragWindow,CWinCtrlBase cDragCtrl)

使用例

```
// get control
CWinCtrlContainer      ctnMap = find<CWinCtrlContainer>(CONTAINER_Map);
// Get TEXTURE(Map) from within content
// Content in the container can be accessed by find as follows
CWinCtrlTexture txMap = find<CWinCtrlTexture>(TEXTURE_Map);
```

11.58 CWinCtrlFrame

スクリプト: *FRAME*

フレームを表示するためのコントロールです。

バーと違いキャプションが持てないです。

CWindowBase に届くコールバック

void onClick(CWinCtrlBase cCtrl)

void onHold(CWinCtrlBase cCtrl)

void onBeginDrag(CWinCtrlBase cCtrl,Vector2 pos)

void onDrag(CWinCtrlBase cCtrl,Vector2 pos,Vector2 dragVelocity)

bool onDragRender(CWinCtrlBase cCtrl,Transform transform)

void onDrop(CWinCtrlBase cCtrl,CWindowBase cDragWindow,CWinCtrlBase cDragCtrl)

void onDropGround(CWinCtrlBase cCtrl)

使用例

```
// get control
CWinCtrlFrame  ctrlStart = find(FRAME_BG) as CWinCtrlFrame;

// click callback
override protected void onClick(CWinCtrlBase cCtrl) {
    switch (cCtrl.id) {
      case FRAME_BG:
        break;
     }
}
```

11.59 CWinCtrlLabel

スクリプト: *LABEL*

ラベルを表示するためのコントロールです。*BAR* と違い、onClick イベントを受け取れません。

CWindowBase に届くコールバック

このコントロールは、デフォルトで NOHIT になっています。コールバックを受け取りたいときは、STYLE に HIT を指定し忘れないでください。

```
void onClick(CWinCtrlBase cCtrl)

void onHold(CWinCtrlBase cCtrl)

void onBeginDrag(CWinCtrlBase cCtrl,Vector2 pos)

void onDrag(CWinCtrlBase cCtrl,Vector2 pos,Vector2 dragVelocity)

bool onDragRender(CWinCtrlBase cCtrl,Transform transform)

void onDrop(CWinCtrlBase cCtrl,CWindowBase cDragWindow,CWinCtrlBase
cDragCtrl)

void onDropGround(CWinCtrlBase cCtrl)
```

使用例

```
// get control
CWinCtrlLabel  ctrlStart = find(LABEL_Title) as CWinCtrlLabel;
```

11.60 CWinCtrlBar

スクリプト: *BAR*

バーを表示するためのコントロールです。

ラベルと同様キャプションを持てます。

CWindowBase に届くコールバック

void onClick(CWinCtrlBase cCtrl)

void onHold(CWinCtrlBase cCtrl)

void onBeginDrag(CWinCtrlBase cCtrl,Vector2 pos)

void onDrag(CWinCtrlBase cCtrl,Vector2 pos,Vector2 dragVelocity)

bool onDragRender(CWinCtrlBase cCtrl,Transform transform)

void onDrop(CWinCtrlBase cCtrl,CWindowBase cDragWindow,CWinCtrlBase cDragCtrl)

void onDropGround(CWinCtrlBase cCtrl)

使用例

```
// get control
CWinCtrlBar    ctrlStart = find<CWinCtrlBar>(BAR_Title);

// click callback
override public void onClick(CWinCtrlBase cCtrl) {
    switch (cCtrl.id) {
      case BAR_Title:
        break;
     }
}
```

第 12 章

付録

12.1 デフォルト値の変更方法

様々なデフォルト値を、KsSoft/Plugins/KsSoftConfig.cs 内で定義しています。

デフォルト値は、KsSoftConfig.initialize 内で値を変更可能です。

12.1.1 アセットバンドル系

bool **`m_UseStreaming`** = true

アセットバンドルをストリーミングするかどうかを選択します。こちらも 参照 ください。

- false の場合

  アセットバンドルのエキスポート先

  KsSoftConfig.m_assetbundlePath のさしている場所

  アセットのダウンロード元

  KsSoftConfig.m_httpserver のさしている場所

- true の場合

  アセットバンドルのエキスポート先

  KsSoftConfig.m_assetbundlePath のさしている場所

  アセットのダウンロード元

  Assets/StreamingAssets/ もし、StreamingAsset が存在しなければ、*KsSoftConfig.m_assetbundlePath* のさしている場所

string **m_assetbundlePath** = "assetbundles/"
　アセットバンドルの出力フォルダを変更

string **m_assetbundleExt** = ".unity3d"
　アセットバンドルファイルの拡張子

string **m_resourcePath** = "Assets/Resources/"
　リソースデータとして出力するときに使用するパス

string **deugPath**
　"http://"経由ではなく、"file://"経由でアセットバンドルを読み込むときに使用される。

注釈: 実機では無効の値です。

12.1.2 ウィンドウリソース系

string **m_windowResourcePath** = "Assets/WindowResource/"
　wra ファイルを格納するフォルダを指定する。

string **m_WindowResourceBinaryPath** = "wrb/"
　wra ファイルをコンパイルした結果を格納するフォルダを指定する。

uint **WindowAssetbundleId**
　アセットバンドル経由で読み込むときにデフォルトでダウンロードするウィンドウアセットバンドル ID

uint **m_defaultFontKind**
　フォント ID が指定されていないときに使われるデフォルト値

12.1.3 テクスチャリソース系

string **m_textureResourcePath** = "Assets/TextureResource/"
　テクスチャリソースを格納するフォルダを指定する。

12.1.4 SE リソース系

string **m_SEResourcePath** = "Assets/SE/"
　SE リソースを格納するフォルダを指定する。

12.1.5 BGM リソース系

string **m_BgmResourcePath** = "Assets/Bgm/"

BGM リソースを格納するフォルダを指定する。

12.1.6 フォント系

string **m_FontResourcePath** = "Assets/FontResource/"

リソース化するフォントデータを格納するフォルダを指定する。

string **m_FontPath** = "Assets/Fonts/"

アセットバンドル化するフォントデータを格納するフォルダを指定する。

12.1.7 メッセージデータシート系

Dictionary<e_Locale, uint> **MessageDataId**

ロケールデータを追加したときは、以下を参考に追加してください。値は、対応するロケールをアセットバンドル化するときに使われるアセットバンドル ID です。

```
static private Dictionary<e_Locale, uint> m_dicStrMessageDataMulId
                    = new Dictionary<e_Locale, uint>() {
  {e_Locale.JP, MulId.Id (0,2,0)},
  {e_Locale.EN, MulId.Id (0,2,1)},
};
```

12.1.8 アドレス/IP 系

string **httpserver** = "http://xxx.xxx.xxx.xxx/"

HTTP 経由でアセットバンドルをダウンロードするときのアドレスを指定してください。

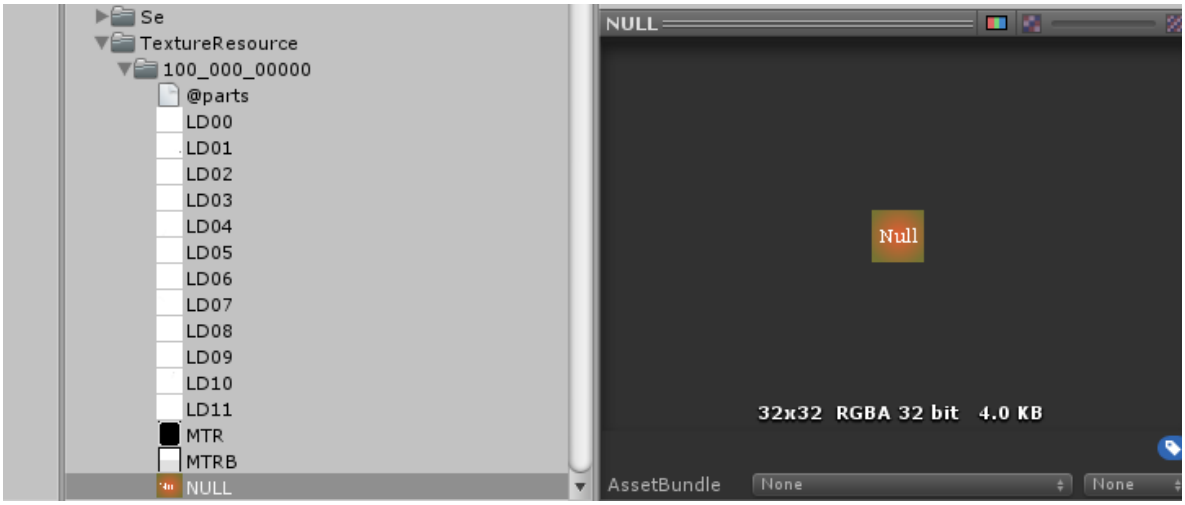
12.2 テクスチャパーツデフォルト値

コントロールの種類	デフォルト値
EDITBOX	0:"TXFD?"(TXFD0,TXFD1) 1:"CURSR"
TEXTBOX	"TXFD"
BUTTON	"BTN0?"(BTN00,BTN01)
RADIO	"BTN0?"(BTN00,BTN01)
CHECKBOX	"BTN0?"(BTN00,BTN01)
TEXTURE	""
LINE	"LINE"
ICON	"ICON"
RECASTICON	"ICON"
METER	0:"MTR" 1:"MTRB"
SCROLLBAR	"SCBRH"
FRAME	"FRAME"
LABEL	"LABEL"
BAR	"BAR"

12.3 "NULL"パーツ:テクスチャパーツが見つからないとき

設定したテクスチャパーツがテクスチャリソース内に存在しないとき、ウィンドウシステムは、代替テクスチャとして"NULL"というテクスチャパーツを探し、割り当てます。

もし、"NULL"というテクスチャパーツが存在しないときは、レンダリングはキャンセルされます。予め"NULL"というテクスチャをリソース内に入れておくことによって、テクスチャパーツの入れ忘れを視覚的に確認することができます。



12.4 CWindowBase に届くコールバック

	onClick	onHold	onClickEnter	onBeginDrag	onDrag	onDragRender
TEXT	O			O	O	O
RICHTEXT	O					
LOG					O(注)	
LOGTEXT	O			O	O	O
EDITBOX	O		O		O(注)	
TEXTBOX	O				O(注)	
BUTTON	O	O		O	O	O
RADIO	O	O		O	O	O
CHECKBOX	O	O		O	O	O
TEXTURE	O	O		O	O	O
LINE	O			O	O	O
RENDER	O	O		O	O	O
ICON	O	O		O	O	O
RECASTICON	O	O		O	O	O
RENDERICON	O	O		O	O	O
METER	O			O	O	O
SCROLLBAR						
LISTBOX					O(注)	
LISTBOXEX					O(注)	
CONTAINER					O(注)	
FRAME	O			O	O	O
LABEL	O			O	O	O
BAR	O			O	O	O

注釈: リッチテキストとスクロール可能なオブジェクトはドラッグできません。スクロール可能なコントロールは、どの方向にコントロールがドラッグされたかを onDrag によって知ることができます。ただし、STYLE に *SCROLL_LOCK* をつけておく必要があります。onDrag によって、ドラッグした方向を検出できます。DRAG をつけても、これらコントロールには効果がないことに気を付けてください。

	onDrop (dst)	onDrop (src)	onDropGround
TEXT	O	O	O
RICHTEXT	O		
LOG	O		
LOGTEXT	O	O	O
EDITBOX	O		
TEXTBOX	O		
BUTTON	O	O	O
RADIO	O	O	O
CHECKBOX	O	O	O
TEXTURE	O	O	O
LINE	O	O	O
RENDER	O	O	O
ICON	O	O	O
RECASTICON	O	O	O
RENDERICON	O	O	O
METER	O	O	O
SCROLLBAR			
LISTBOX	O		
LISTBOXEX	O		
CONTAINER	O		
FRAME	O	O	O
LABEL	O	O	O
BAR	O	O	O

12.5 ドラッグできないコントロール

リッチテキストとスクロール可能なオブジェクトはドラッグできません。スクロール可能なオブジェクトは、どの方向にコントロールがドラッグされたかを onDrag によって知ることができます。

STYLE に *SCROLL_LOCK* をつけておくと、onDrag コールバックが呼び出され、ドラッグした方向を検出できます。DRAG をつけても、これらコントロールには効果がないことに気を付けてください。

- *RICHTEXT*
- *LOG*
- *EDITBOX*
- *TEXTBOX*

- *SCROLLBAR*
- *LISTBOX*
- *LISTBOXEX*
- *CONTAINER*

12.6 有効なスタイルフラグ

	ANCHOR_*	BASE_*	HIDE	DRAG	DISABLE	NOHIT	HIT
TEXT	O	O	O	O	O		O
RICHTEXT	O	O	O	O		O	
LOG	O	O	O	O		O	
LOGTEXT	O	O	O	O	O		O
EDITBOX	O	O	O	O		O	
TEXTBOX	O	O	O	O		O	
BUTTON	O	O	O	O	O	O	
RADIO	O	O	O	O	O	O	
CHECKBOX	O	O	O	O	O	O	
TEXTURE	O	O	O	O	O		O
LINE	O	O	O	O	O		O
RENDER	O	O	O	O	O	O	
ICON	O	O	O	O	O	O	
RECASTICON	O	O	O	O	O	O	
RENDERICON	O	O	O	O	O	O	
METER	O	O	O	O	O		O
SCROLLBAR	O	O	O	O			
LISTBOX	O	O	O	O		O	
LISTBOXEX	O	O	O	O		O	
CONTAINER	O	O	O	O		O	
FRAME	O	O	O	O	O	O	
LABEL	O	O	O	O	O		O
BAR	O	O	O	O	O	O	

	TEXT_LEFT	TEXT_RIGHT	TEXT_CENTER
TEXT			
RICHTEXT	O	O	O
LOG			
LOGTEXT			
EDITBOX			
TEXTBOX			
BUTTON	O	O	O
RADIO	O	O	O
CHECKBOX	O	O	O
TEXTURE			
LINE			
RENDER			
ICON	O	O	O
RECASTICON	O	O	O
RENDERICON			
METER			
SCROLLBAR			
LISTBOX			
LISTBOXEX			
CONTAINER			
FRAME			
LABEL	O	O	O
BAR	O	O	O

	TEXT_NORMAL	TEXT_BOLD	TEXT_DENT	TEXT_SHADOW
TEXT	O	O	O	O
RICHTEXT	O	O	O	O
LOG				
LOGTEXT	O	O	O	O
EDITBOX	O	O	O	O
TEXTBOX	O	O	O	O
BUTTON	O	O	O	O
RADIO	O	O	O	O
CHECKBOX	O	O	O	O
TEXTURE				
LINE				
RENDER				
ICON	O	O	O	O
RECASTICON	O	O	O	O
RENDERICON				
METER				
SCROLLBAR				
LISTBOX				
LISTBOXEX				
CONTAINER				
FRAME				
LABEL	O	O	O	O
BAR	O	O	O	O

	EDIT_BLIND	*EDIT_TYPE_**
TEXT		
RICHTEXT		
LOG		
LOGTEXT		
EDITBOX	O	O
TEXTBOX		
BUTTON		
RADIO		
CHECKBOX		
TEXTURE		
LINE		
RENDER		
ICON		
RECASTICON		
RENDERICON		
METER		
SCROLLBAR		
LISTBOX		
LISTBOXEX		
CONTAINER		
FRAME		
LABEL		
BAR		

	NOBOUNCES	SCROLL_UNLOCK	SCROLL_LOCK
TEXT			
RICHTEXT			
LOG	O	O	O
LOGTEXT			
EDITBOX	O	O	O
TEXTBOX	O	O	O
BUTTON			
RADIO			
CHECKBOX			
TEXTURE			
LINE			
RENDER			
ICON			
RECASTICON			
RENDERICON			
METER			
SCROLLBAR			
LISTBOX	O	O	O
LISTBOXEX	O	O	O
CONTAINER	O	O	O
FRAME			
LABEL			
BAR			

	ITEM_STACK_V/H	*SCROLLBAR_DISPLAY_**
TEXT		
RICHTEXT		
LOG		
LOGTEXT		
EDITBOX		
TEXTBOX		
BUTTON		
RADIO		
CHECKBOX		
TEXTURE		
LINE		
RENDER		
ICON		
RECASTICON		
RENDERICON		
METER		
SCROLLBAR	O	O
LISTBOX	O	
LISTBOXEX	O	
CONTAINER		
FRAME		
LABEL		
BAR		

12.6.1 NOHIT/HIT について

コントロール毎に、HIT 判定の有無のデフォルトが異なります。

	Default	NOHIT	HIT
TEXT	NOHIT		O
RICHTEXT	HIT	O	
LOG	HIT	O	
LOGTEXT	NOHIT	O	
EDITBOX	HIT	O	
TEXTBOX	HIT	O	
BUTTON	HIT	O	
RADIO	HIT	O	
CHECKBOX	HIT	O	
TEXTURE	NOHIT		O
LINE	NOHIT		O
RENDER	HIT	O	
ICON	HIT	O	
RECASTICON	HIT	O	
RENDERICON	HIT	O	
METER	NOHIT		O
SCROLLBAR	NOHIT		
LISTBOX	HIT	O	
LISTBOXEX	HIT	O	
CONTAINER	HIT	O	
FRAME	HIT	O	
LABEL	NOHIT		O
BAR	HIT	O	

第 13 章

Indices and tables

- genindex
- modindex
- search