

PS/1 操作マニュアル

目次

1. wrapperについて.....	2
2. 起動	3
3. 再開始	4
4. メニュー・パネルと目的画面	5
5. 効率的な作業手順	6
6. ウィジェットの編集	
(1) 貼り付け	7
(2) 移動	8
(3) 重なり変更	8
(4) コピー	8
(5) 属性編集	9
7. メニューの編集	12
8. コンフィグレーション	13
9. インフォメーション	16
10. セーブと終了	17
11. 目的スクリプトの使い方	18
12. 注意事項	19

用語

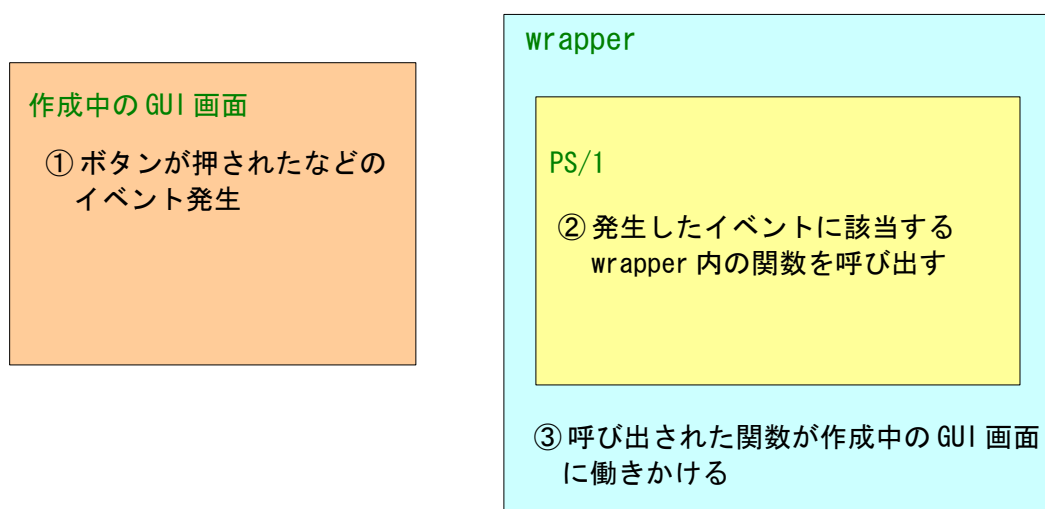
目的画面	作成する GUI 画面
目的スクリプト	目的画面を生成するためのスクリプト (PS/1 が出力する)
wrapper	コールバック関数や変数などを定義した Perl スクリプト (ユーザが作成する)
クリック	マウスの左ボタンを押すこと
右クリック	マウスの右ボタンを押すこと
実行時	目的スクリプトを実行している状態のこと

1. wrapper について

ユーザが関数や変数などを記述した Perl スクリプト・ファイルを **wrapper** と呼びます。

wrapper を使うと、コールバック関数やイベント処理の動作を、**PS/1** の実行中に確認することができます。

下図は、**PS/1** と **wrapper** の関係を示すイメージです。



wrapper の内容は、PS/1 の実行中でも変更することができます。（リロード機能）

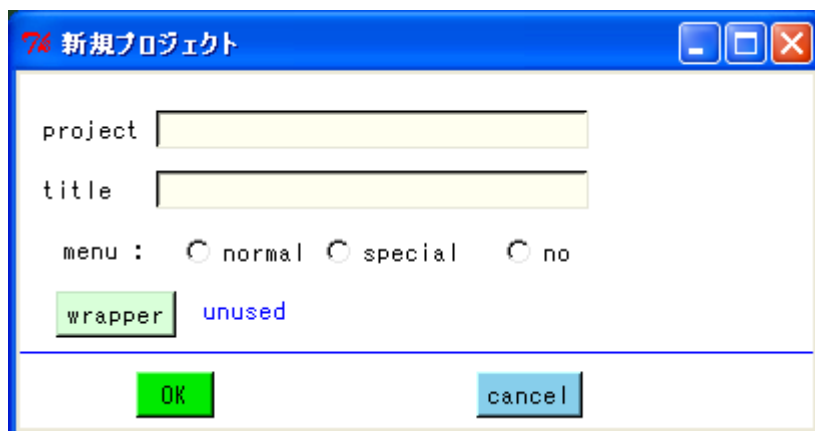
wrapper でエラーが発生しても PS/1 が異常終了することはありません。（デバッグ機能）

wrapper はオプションです。

2. 起動

PS/1 は `startps1.bat` (Linux は `startps1.csh`) で起動します。下図の起動パネルが開きます。

図 1 : 起動パネル



- ① **project** プロジェクト名 (フォルダ名に成り得る名前を入力する)
- ② **title** 目的画面のタイトル (option)
- ③ **menu ボタン** メニューの種類 (normal/special/no のいずれかを選択する)
 - normal** Perl/Tk で作られる通常のメニュー
 - special** 画像の貼り付けや、フォント/色指定が可能なメニュー
 - no** メニューを使わない
- ④ **wrapper ボタン** wrapper を指定 (option)

`title` と `wrapper` は、後で(再)設定することができます。(「8. コンフィグレーション」参照)

項目を指定した状態の起動パネル



OK ボタンを押して作業を開始します。

- ⑤ **encode ボタン** PS/1 による wrapper 文字コードの判定が誤りの場合に、正しい encode を指定する

3. 再開始

作業は、任意の時点で中断し、`startps1.bat` (Linux は `startps1.csh`) で中断した状態から再開始できます。

中断の方法については「10. セーブと終了」のページで説明します。

下図は、作成中のプロジェクトが存在するときに開く再開始パネルです。

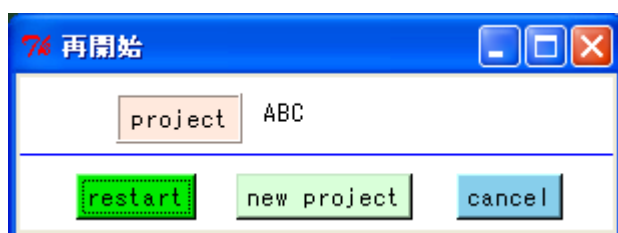
図 2 : 再開始パネル



restart ボタン ABC プロジェクトを再開始

new project ボタン 新たにプロジェクトを作成 (→ 図 1 の起動パネルが開く)

複数のプロジェクトがある場合



project ボタン プロジェクトを選択

4. メニュー・パネルと目的画面

作業を開始すると、下図のようにメニュー・パネルと目的画面が開きます。
 左側のメニュー・パネルを使って、ウィジェットの貼り付けや編集を行います。
 目的画面の大きさは、画面の四隅や縁をドラッグして変更します。

図 3 : メニュー・パネルと目的画面

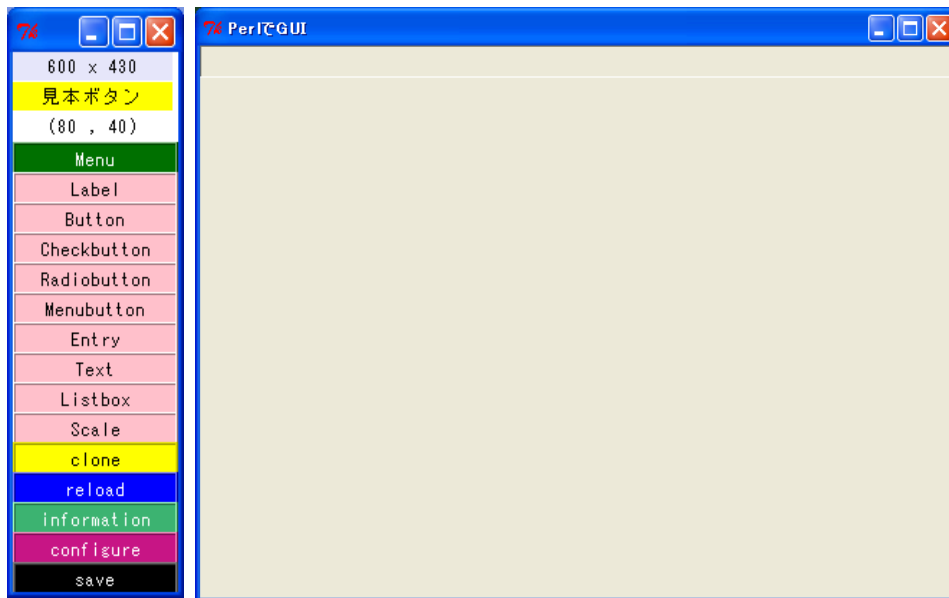
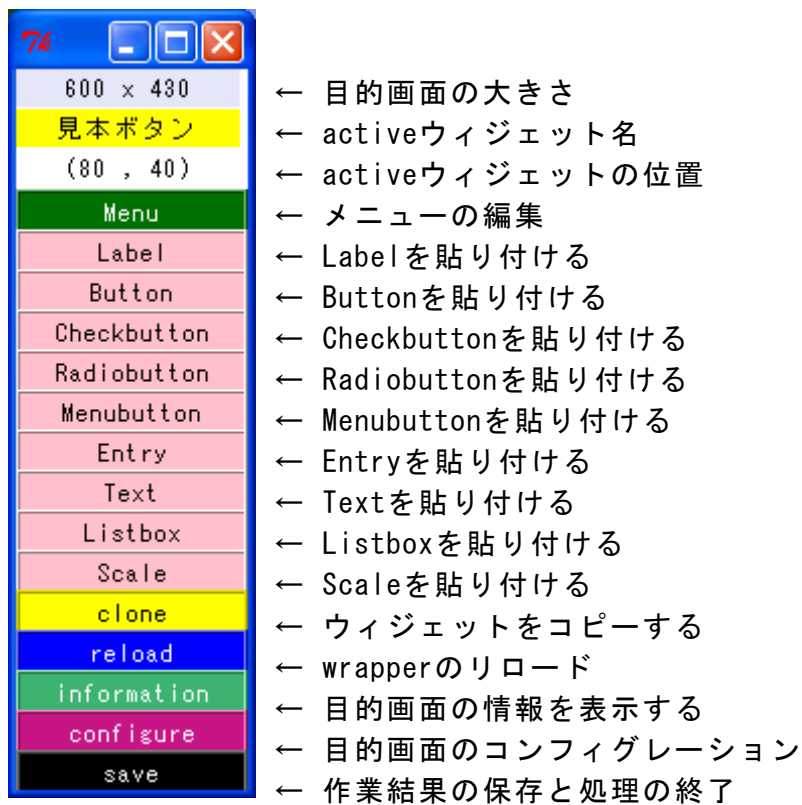


図 4 : メニュー・パネル



5. 効率的な作業手順

ウィジェットを貼り付ける前に、メニュー・パネルの **configure** を使って、目的画面/ウィジェット/メニュー/ポップアップなどのデフォルト属性を設定しておくとは効率的です。

configure については「8. コンフィグレーション」のページで説明します。

次の手順で操作します。

- (1) 画面の大きさを決める。(目的画面の四隅や縁をドラッグ)
- (2) 背景色を設定する。(configure)
- (3) ウィジェットのデフォルト属性を設定する。(configure → widget)
- (4) menubar のデフォルト属性を設定する。(configure → menubar)
- (5) ポップアップのデフォルト属性を設定する。(configure → pop-up)

これらの設定は、随時、変更することができます。

【注意】 (4)は menu=special の場合、(5)は menu≠no の場合のみ操作可能です。

6. ウィジェットの編集

(1) ウィジェットの貼り付け

次の方法で貼り付けます。

- ① メニュー・パネルの **Label**～**Scale** の任意のボタンをクリック (→ カーソルが '**+**' になる)
- ② 目的画面の貼り付けたい場所をクリック (→ カーソルが '**矢印**' に戻る)

下図は、上記の方法で、いろいろなウィジェットを貼り付けた状態です。

図 5 : ウィジェットの貼り付け状態



ウィジェットは、configure で設定した属性、または、Perl/Tk のデフォルト属性で作られます。

ラベル属性を持つウィジェットには、**Label1**, **Button2** のように、便宜的に「**クラス名+クラスごとの生成番号**」のラベルが付けられます。これは、後で任意のラベルや画像などに変更することができます。

ラベル属性を持たない **Entry**, **Text**, **Listbox** にも内部的には同様の名前が付けられて、ユーザへのメッセージなどに使われますが、目的画面には表示されません。

上図の **Entry**, **Text**, **Listbox** の文字列は、説明のために表示したもので、実際は空白になります。

貼り付けたウィジェットを**右クリック**すると、ウィジェットの形状やインタフェースなどの属性を設定するための**編集パネル**が開きます。

編集パネルについては「(5)ウィジェットの編集」で説明します。

(2) ウィジェットの移動

貼り付けたウィジェットの配置を変更するには、次の3つの方法があります。

下の4つの図は画像ウィジェットの例ですが、画像以外のウィジェットも同様です。

① 矢印キーによる移動

ウィジェットを active 状態にして、**矢印キー**（上下左右）で移動させる。

ただし、文字列が挿入されている Text は無効、

Scale と、文字列が挿入されている Entry は上下だけが有効です。

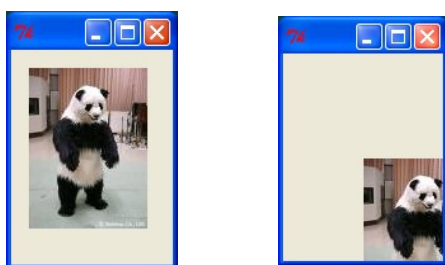
② ドラッグによる移動

ウィジェットを **shift+左ボタン** でドラッグして移動させる。

③ 移動ボタンによる移動

ウィジェットの**編集パネル**を開いて、**移動ボタン**で移動させる。

この方法だけ、ウィジェットを画面から**はみ出させる**ことができます。



(3) ウィジェットの重なり変更

他のウィジェットと位置が重なった場合、後から貼り付けたウィジェットが上部に表示されます。この重なり具合は、次の方法で変更することが出来ます。

① 上または下に置きたいウィジェットの**編集パネル**を開く。

② **inside** または **outside** ボタンをクリックする。



(4) ウィジェットのコピー

貼り付けたウィジェットは次の方法でコピーすることができます。

① コピーするウィジェットをクリック。（active 状態にする）

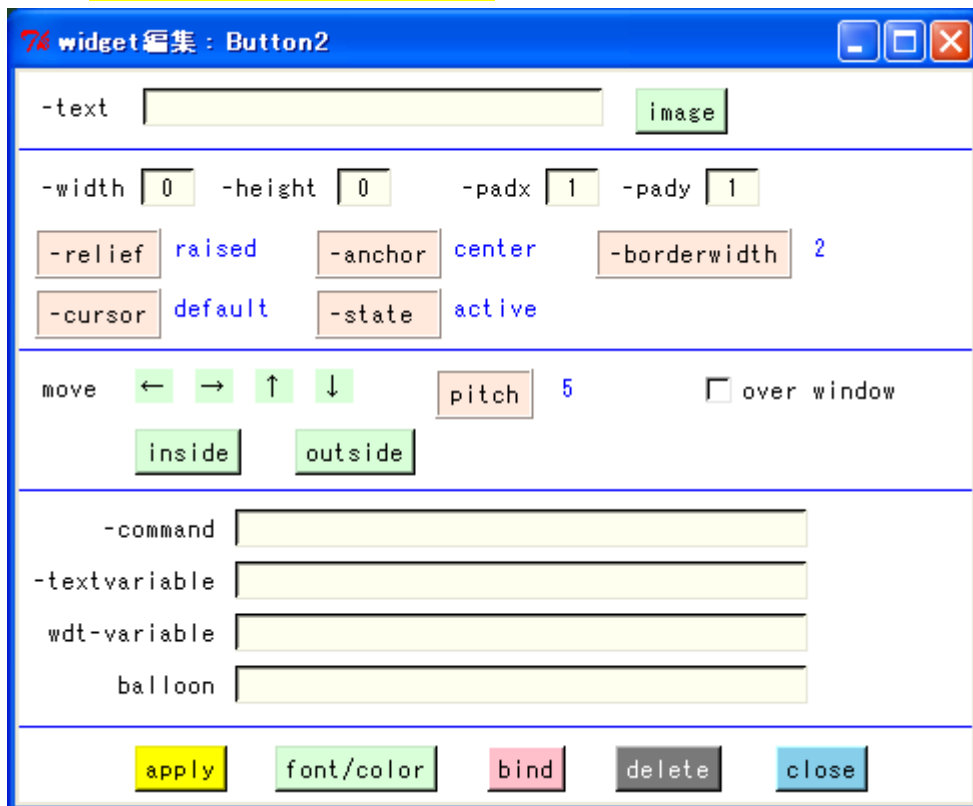
② メイン・パネルの **clone** ボタンを押す。（→ カーソルが ' + ' になる）

③ 貼り付けたい場所をクリック（→ ウィジェットがコピーされ、カーソルが ' 矢印 ' に戻る）

(5) ウィジェットの属性編集

下図は、貼り付けたウィジェットを右クリックすると開く編集パネルです。

図 6 : ウィジェット編集パネル (Button ウィジェットの場合)



編集パネルにはウィジェットのクラスに応じて、必要な項目が表示されます。

また、ウィジェットの状態に応じて、表示される項目が変化します。

例えば `-textvariable` が定義されている場合は、`-text` 項目は表示されません。

① Perl/Tk 属性項目

`-text`, `-width` のように、先頭が `'-'` で始まる項目は Perl/Tk で使われる属性です。

PS/1 では、次に示す Perl/Tk 属性を取り扱います。

`-text` `-width` `-height` `-padx` `-pady` `-relief` `-anchor` `-borderwidth` `-orient` `-scrollbars`
`-justify` `-cursor` `-state` `-wrap` `-direction` `-indicatoron` `-command` `-variable` `-onvalue`
`-offvalue` `-value` `-textvariable`

font/color ボタンや bind ボタンを押したときに開くパネルにも同様の項目があります。

各属性の意味や使い方については Perl/Tk のマニュアルなどを参照してください。

`-command` および `bind` 指定でのコールバック処理には、次のように、ユーザ関数以外の Perl スクリプトを書くこともできます。 `print "abc\n"; &xyz;...`

② 変更した内容がウィジェットに適用されるタイミング

文字列や値を入力する項目は、リターン・キーまたは apply ボタンを押したとき、その他の項目は、当該ボタンを押したときに、ウィジェットに適用されます。

③ image ボタン ウィジェットに画像を貼り付けます。

画像だけを画面に置く場合でも、Labelなどを貼り付けてから image ボタンで変更します。

- ④ **move 欄** ウィジェットを移動するためのボタン類です。
pitch 上下左右ボタンの1回のクリックで移動する単位を指定します。
over window ウィジェットを画面からはみ出して移動します。
inside ウィジェットを最下部に配置します。
outside ウィジェットを最上部に配置します。
- ⑤ **wdt-variable** wrapper でウィジェットを操作するための変数名を指定します。
 例えば \$v1 と指定すると、\$v1->configure(...)などと、wrapper でウィジェットを操作することができるようになります。
- ⑥ **balloon** ウィジェット上にマウスがあるときに表示するバルーン・ヘルプの文字列を指定します。
 文字列中に '\n' があると、そこで改行します。
- ⑦ **apply ボタン** 文字列や値を入力する項目の適用処理を行います。
 内容が変更され、リターン・キーを押されていない項目が対象です。
- ⑧ **delete ボタン** ウィジェットを削除します。
- ⑨ **close ボタン** 編集パネルを閉じます。
- ⑩ **font/color ボタン** フォントと色属性を変更するためのパネルを開きます。(図7を参照)
- ⑪ **bind ボタン** イベントのbind 処理を設定するためのパネルを開きます。(図8を参照)
- ⑫ **pop-up ボタン** ポップアップ・メニューを編集するためのパネルを開きます。(図9, 10を参照)

図7 : font/color パネル

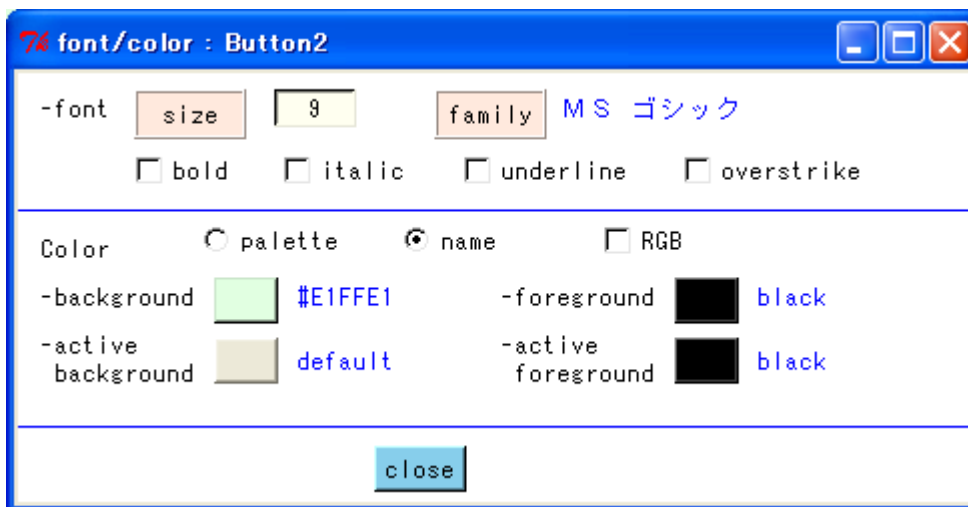


図8 : bind パネル

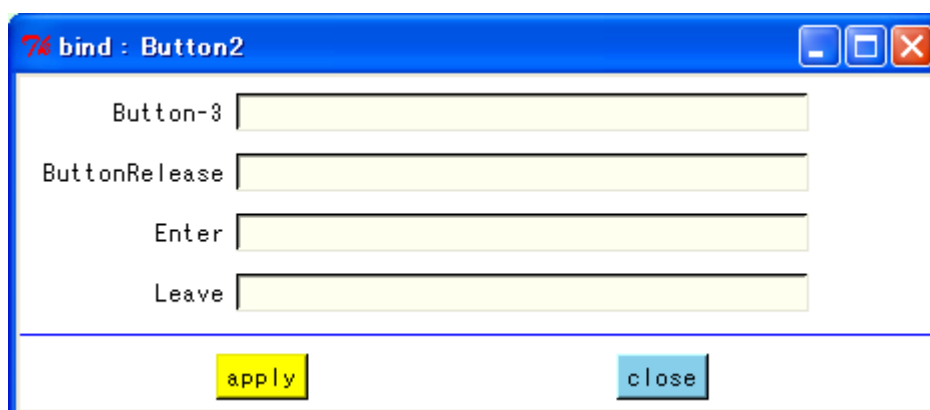


図 9 : pop-up 登録パネル

エントリが未登録状態のときに開くパネルです。

-label 欄にラベル文字列を指定し、type を選択して add ボタンでエントリを登録します。

style ボタン pop-up の形状を変更するパネルを開きます。

図 10 : pop-up 編集パネル

すでにエントリが登録されている場合に開くパネルです。

上段 : 属性を変更するエントリを選択するボタンです。

選択したエントリの編集項目が中段に表示されます。

既存エントリが 1 ッの場合は、上段部分は表示されません。

中段 : 既存エントリの属性を変更する項目です。

エントリ・タイプに応じて必要な項目が表示されます。

-label ラベル文字列を変更。

separator ボタン 直前に区切り線を挿入。

shortcut ボタン ショートカット・キーを設定。

image ボタン ラベルに画像を貼り付ける。

shift ボタン 直前のエントリと位置を入れ替える。

pop-up ボタン cascade のポップアップを編集する。

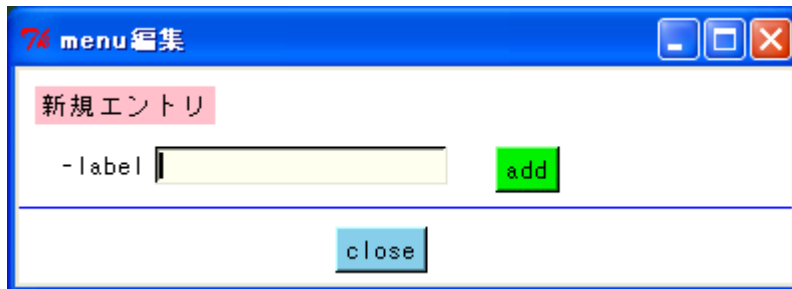
下段 : 新規エントリを登録する項目です。

-label 欄にラベル文字列を指定して type を選択し、add ボタンで登録します。

7. メニューの編集

画面最上部のメニューは、メニュー・パネルの **Menu ボタン** で編集します。

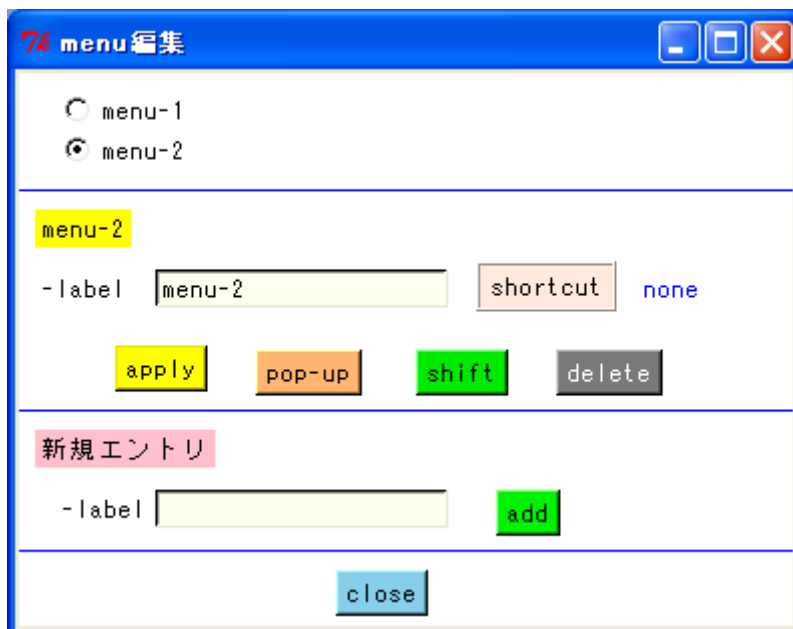
図 11：メニュー登録パネル



エントリが未登録状態のときに開くパネルです。

-label 欄にラベル文字列を入力し add ボタンで登録します。

図 12：メニュー編集パネル (menu=special の場合)



すでにエントリが登録されているときに開くパネルです。

上段：属性を変更するエントリを選択するボタンです。
 選択したエントリの編集項目が中段に表示されます。
 既存エントリが 1 ッの場合は、上段部分は表示されません。

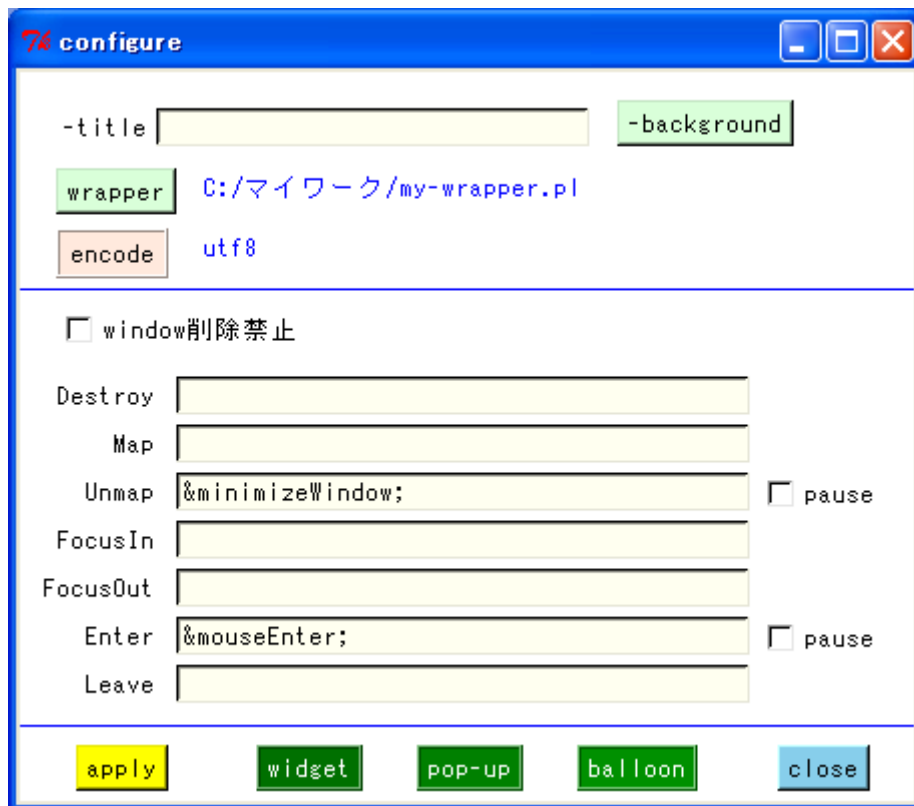
中段：既存エントリの属性を変更する項目です。
 内容は「図 10：pop-up 編集パネル」と同様です。

下段：新規エントリを登録する項目です。
 -label 欄にラベル文字列を指定し、add ボタンで登録します。

8. コンフィグレーション

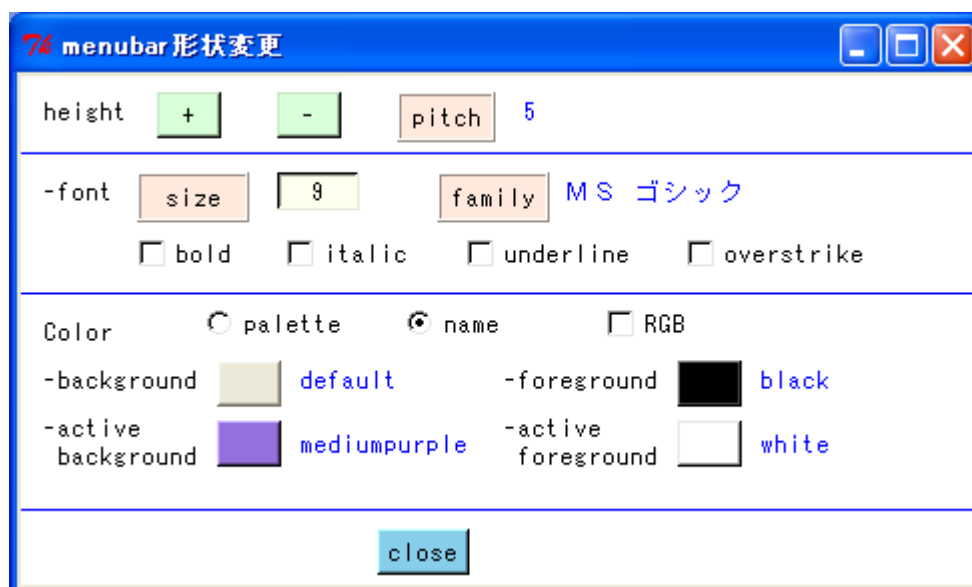
メニュー・パネルの **configure** ボタンで開くパネルです。
目的画面のさまざまな属性を変更します。

図 13 : **configure** パネル



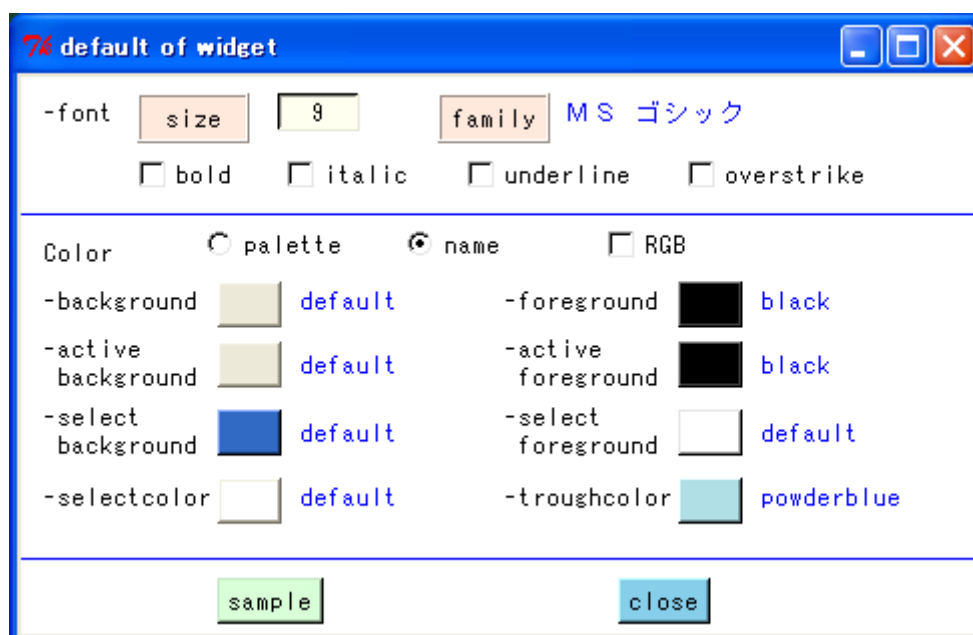
- ① **-title** ウィンドウ・タイトルの(再)定義。
- ② **-background**ボタン 目的画面の背景色を変更するためのパネルを開く。
すでに貼り付けたウィジェットの背景色を、同時に変更することもできます。
- ③ **wrapper**ボタン wrapperの(再)定義。
- ④ **encode**ボタン wrapper ファイルの文字コード指定
- ⑤ **window削除禁止**ボタン 目的画面の「閉じる」ボタンを無効にする。(実動時のみ機能)
- ⑥ **Destroy** 目的画面の「閉じる」ボタンが押されたときの処理。(実動時のみ機能)
window削除禁止ボタンのON/OFF状態に関係なく、実動時に機能します。
- ⑦ **Map~Leave** 目的画面に当該イベントが発生したときの処理。
- ⑧ **pause**ボタン PS/1実行中は当該イベント処理を抑止する。
処理が定義されているイベントの右側に表示されるボタンです。
(上図は Unmap と Enter が定義されている例です)
- ⑨ **apply**ボタン リターン・キーを押されていない項目の適用処理を行う。
- ⑩ **menubar**ボタン メニュー・バーの形状や、エントリのデフォルト属性を定義するパネルを開く。
(special menuのみ) (図14を参照)
- ⑪ **widget**ボタン ウィジェットのデフォルト属性を定義するパネルを開く。(図15を参照)
- ⑫ **pop-up**ボタン ポップアップのデフォルト属性を定義するパネルを開く。(図16を参照)
- ⑬ **balloon**ボタン バルーン・ヘルプの属性を定義するパネルを開く。(図17を参照)

図 14 : menubar パネル



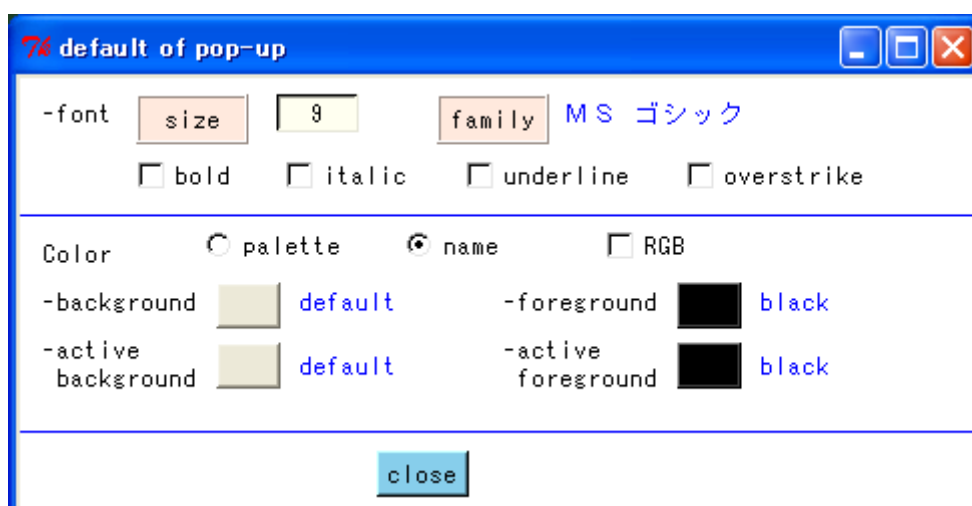
special menu のバーの高さの変更と、新規に登録するエントリのフォントと色属性のデフォルトを定義するパネルです。
 選択するフォントのサイズとファミリの組み合わせによって、バーの高さが自動で変わりますが、height ボタンでお好みの高さに調整してください。

図 15 : widget パネル



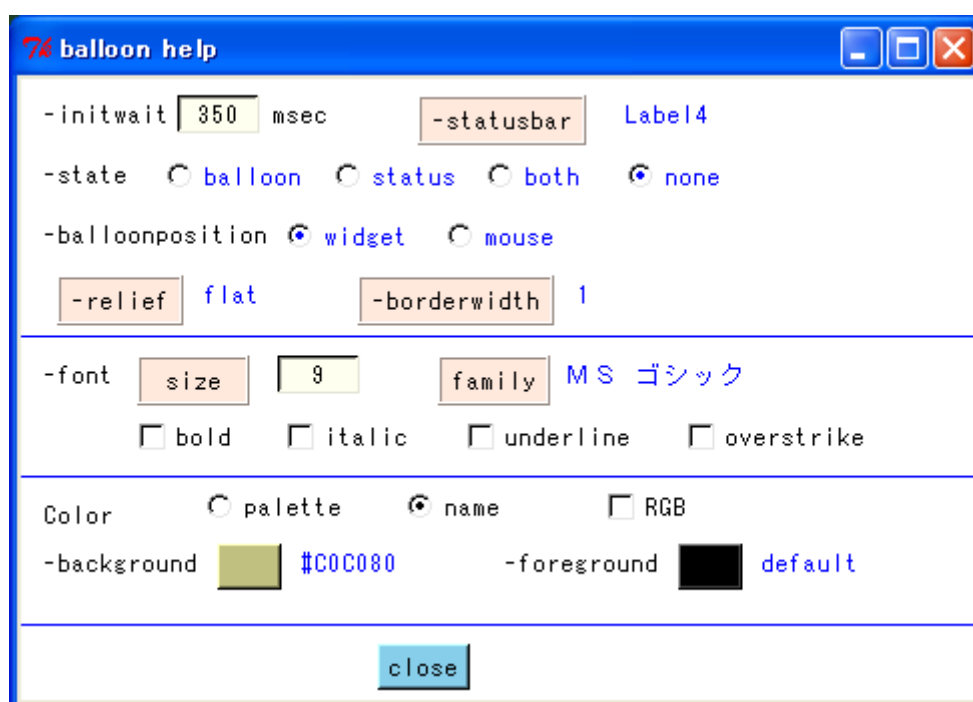
新規に貼り付けるウィジェットのフォントと色属性のデフォルトを定義するパネルです。

図 16 : pop-up パネル



新規に登録するポップアップのフォントと色属性のデフォルトを定義するパネルです。

図 18 : balloon パネル



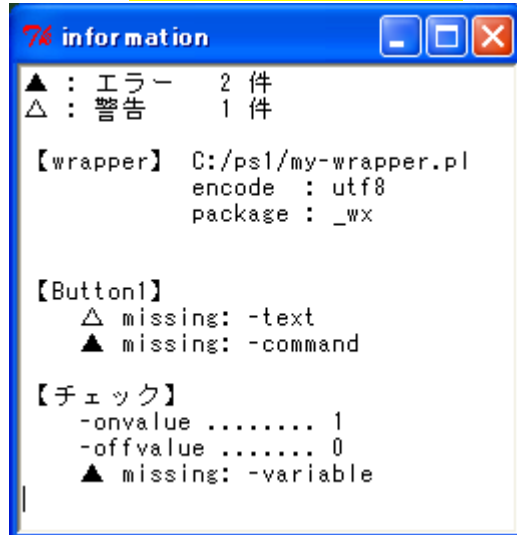
バルーン・ヘルプの属性を定義するパネルです。

9. インフォメーション

メニュー・パネルの **information** ボタンで開くパネルです。

wrapper 情報と、貼り付けたウィジェットごとに属性値やエラー/警告情報が表示されます

図 19 : information パネル



上図の例ではエラーが2件、警告が1件あります。

△ missing: -text

Button1 のラベルが未設定 (PS/1 が便宜的に付けたラベルのままになっている)

▲ missing: -command

ボタンが押された時のコールバック処理が未設定 (ボタンの意味がない)

▲ missing: -variable

ボタンのチェック状態を示す変数が未設定 (Checkbutton として機能しない)

10. セーブと終了

(1) PS/1 が出力するファイル

次の2つのファイルを出力します。

- ① 作業を再開するためのRC ファイル `rc.lis`
- ② 目的画面を生成するための目的スクリプト `code.pl` (製品版のみ)

2回目以降のセーブ処理では、前回の `rc.lis` は `rc-xxxx.lis` にリネームされて残ります。

`xxxx` は、そのファイルの作成時刻です。

この仕組みを利用して、数世代前の状態から作業を再開することもできます。

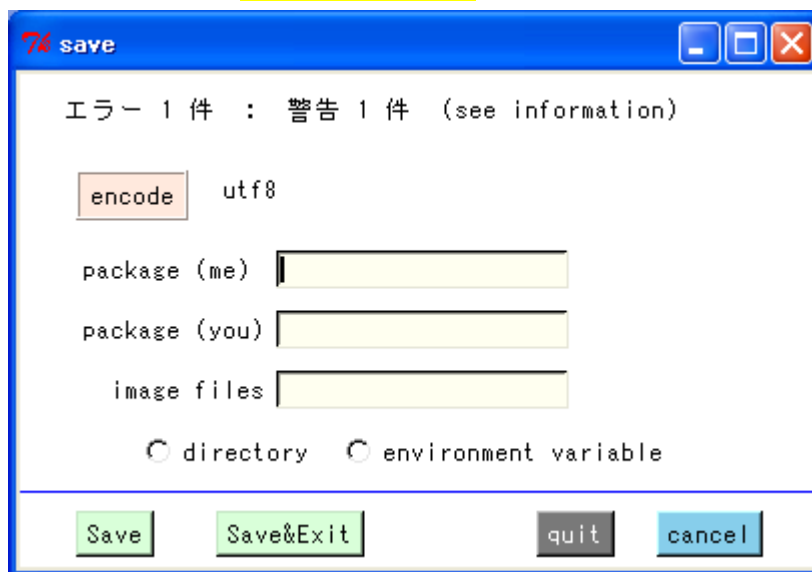
`code.pl` はセーブのたびに、常に上書きします。

2つのファイルは、PS/1 のインストール・フォルダ中の `project` フォルダ内のプロジェクト名フォルダに出力します。

例えば、インストール・フォルダが '`c:/ps1`'、プロジェクト名が '`ABC`' の場合は
`c:/ps1/project/ABC/rc.lis` , `c:/ps1/project/ABC/code.pl` となります。

(2) セーブ・終了パネル

図 20 : save パネル



メニュー・パネルの **save ボタン** で開くパネルです。

- ① **エラー/警告件数**
貼付けたウィジェットに関するエラーと警告の件数
メニュー・パネルの `information` ボタンで詳細を見ることができます。
- ② **encodeボタン** 目的スクリプトの文字コード (default: `utf8`)
- ③ **package (me)** 目的スクリプトに付けるパッケージ名 (option)
- ④ **package (you)** 目的スクリプトを呼び出すユーザ・スクリプトのパッケージ名 (option)
- ⑥ **Saveボタン** ファイルをセーブして処理を続行
- ⑦ **Save&Exitボタン** ファイルをセーブして処理を終了

⑧ **quitボタン** ファイルをセーブしないで処理を終了

⑤ **image files, directory, environment variable (option)**

これらは画像を含む目的画面を、作成したプラットフォームと異なるプラットフォームで取り扱う場合に使います。

例えば、Windows上で目的画面を作成し、**image files** に '/home/image' を入力して **directory** を選択した場合、目的スクリプトは '/home/image' から画像ファイルをアクセスします。

あるいは、**image files** に 'ABC' を入力して、**environment variable** を選択した場合、目的スクリプトは環境変数 'ABC' が指すフォルダから画像ファイルをアクセスします。

1 1. 目的スクリプトの使い方

目的スクリプトはrequire文で読み込みます。

code.pl は任意のフォルダに移動したり、リネームしても使えます。

code.pl を xxx.pm にリネームし、環境変数 INC にパスを確立すればuse文でも可能です。

インストール・マニュアルのサンプルを参照して下さい。

目的スクリプト中の関数と変数で、ユーザが使用するのは、次の2つです。

- ① **_psCreateWindow 関数** 目的画面を生成する関数
- ② **\$_psCanvas 変数** 目的画面のキャンバス・オブジェクト

これらは、セーブ・終了パネルでの package(me) の指定状態によって、使い方が異なります。

(1) package(me) を指定した場合

次のように関数/変数をパッケージ名でキャストします。(package(me)=xxx の例)

```
[任意の変数 =] &xxx::_psCreateWindow;
$xxx::_psCanvas->create('line',...);
```

(2) package(me) を指定しなかった場合

```
[任意の変数 =] &_psCreateWindow;
$_psCanvas->create('line',...);
```

関数/変数をそのまま使えますが、目的スクリプト中には、他にも '_ps' で始まる関数や変数がありますので、名前が衝突しないように注意してください。

12. 注意事項

(1) コールバック処理について

-command 欄や bind 欄にコールバック処理を記述するとき、ユーザ関数名の先頭に必ず '`&`' を付けて下さい。

これは、PS/1 が目的スクリプトを生成する際に、関数名をキャストする場合の手掛かりにするためです。

目的スクリプトを呼び出す側にパッケージ指定が無い場合は不要ですが、後々の仕様変更などのためにも、'`&`' の付与を推奨します。

(2) 画像ファイルの取り扱い

.../画像/panda.jpg や .../image/パンダ.gif などのように、画像ファイルを特定するパスにマルチバイト文字を含む画像ファイルは処理できません。(Perl/Tk の制限事項)

(3) フォントについて

PS/1 の実行には、次のフォントが必要です。

Windows = MS ゴシック

Linux = courier

(4) パネルの有効性

メニュー・パネルと目的画面は PS/1 の実行中にスクリーンに常駐しますが、そこから派生して表示されるパネルは、configure → widget、または、pop-up のネスト処理など、複数のパネルが同時に開かれる場合があります。

このような場合、原則として最後に開かれたパネルだけが有効で、その他はボタンを押すなどしても反応しませんので、前に開いたパネルを有効にするには、後で開いたパネルを閉じてください。

(自動で閉じるパネルもあります)

ただし、次の例外があります。

- ① 目的画面のイベントはいつでも発生させることが出来ます。
- ② information ボタン(メニュー・パネル)は常に有効です。