The Tk Packer

SAM
sam@cs.vu.nl

version 1.0

With thanks to Jaap Kreykamp and Jacco van Ossenbruggen

June 20, 1996

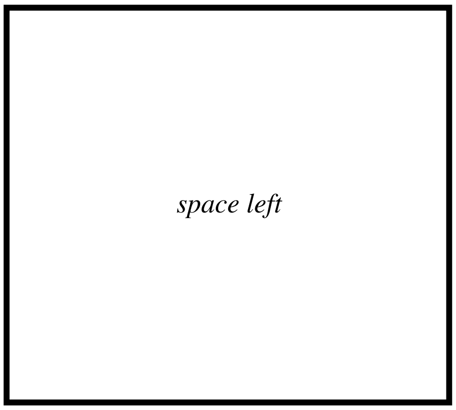
1 Introduction

Every time a pack-function in Hush is called, the Packer of the underlying Tk window toolkit is activated. The Packer takes care of adding (child) widgets to other (parent) widgets. How the Packer exactly functions is not very obvious at first sight. This document will try to shed some light on the Packer.

2 Some basics

Imagine the window to which widgets will be added as a big rectangle. Every time a widget is added by calling a pack-function, the remaining rectangle will be divided in two: one part will contain the added widget, the other part will be used to add other widgets. When a widget is packed on the right side of the rectangle, room will be left on the left side for future packs. An example will make this more clear.

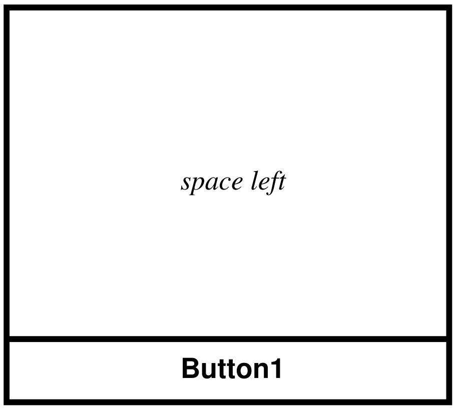
An empty window:



Add a widget (for example a button b1) to the bottom side by calling the pack() member function:

```
b1 = new button(".b1", "-text Button1");
b1 -> pack("-side bottom -fill x");
```

The window will now have the following form [1]:

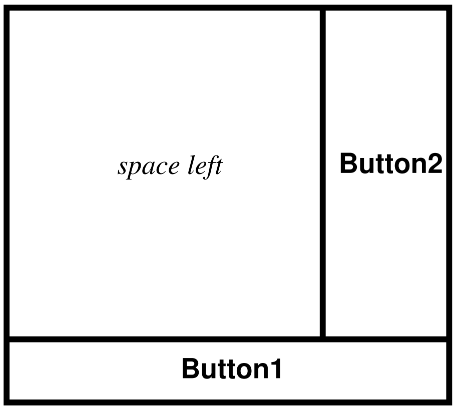


Next, we can add a button b2 to the right side of the remaining rectangle with

```
b2 = new button(".b2", "-text Button2 -height 10");
b2 -> pack("-side right -fill y");
```

The window now shows:

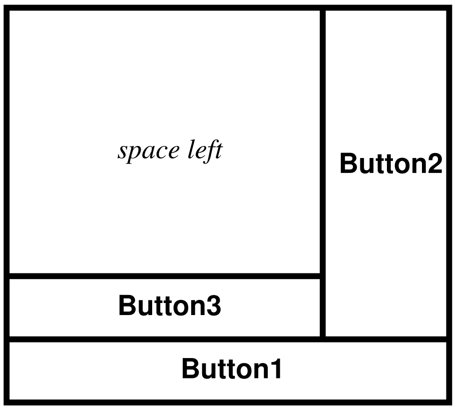
[1] This is not entirely true, because the Packer will try to leave out the space that's left. In this case the actual window will only contain the button. But for a more clear understanding of the Packer, it's better to represent it this way.



When a button b3 is added at the bottom side of the remaining rectangle with

```
b3 = new button(".b3", "-text Button3");
b3 -> pack("-side bottom -fill x");
```

the result will be:

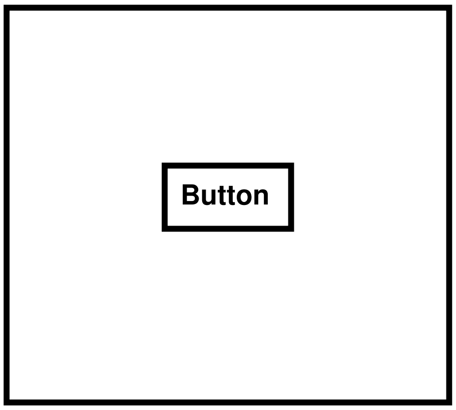


3 Option-parameters

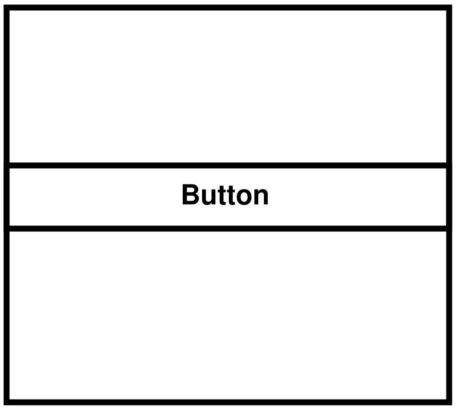
Parameters can be passed to pack-functions for changing the way widgets are packed. One example of this is the parameter '-side XXX', as shown in the examples above. This parameter will tell on which side of the remaining rectangle a smaller rectangle will contain the new widget. Possible sides are 'top', 'bottom', 'left' and 'right'.

Furthermore, a '-fill' parameter can be passed. This parameter tells the packer to expand the widget if pieces of the rectangle are not filled by the widget. Some examples:

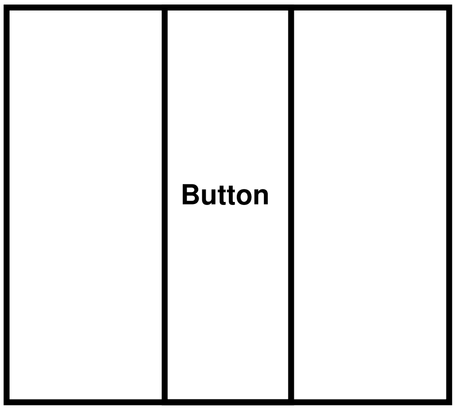
Without any filling:



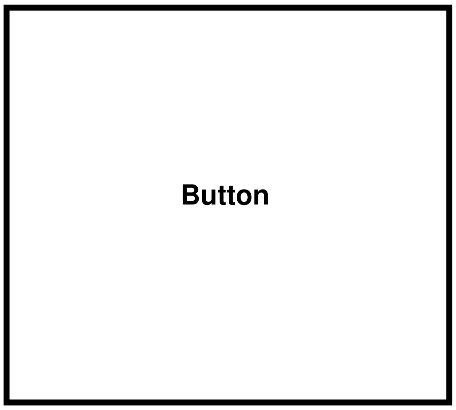
With '-fill x':



With '-fill y':



With '-fill both':



Another parameter is '-expand'. This parameter is similar to '-fill'. The difference between '-expand' en '-fill' becomes clear when some small wish-scripts are executed. Wish should be started anew for every script.

Script 1:

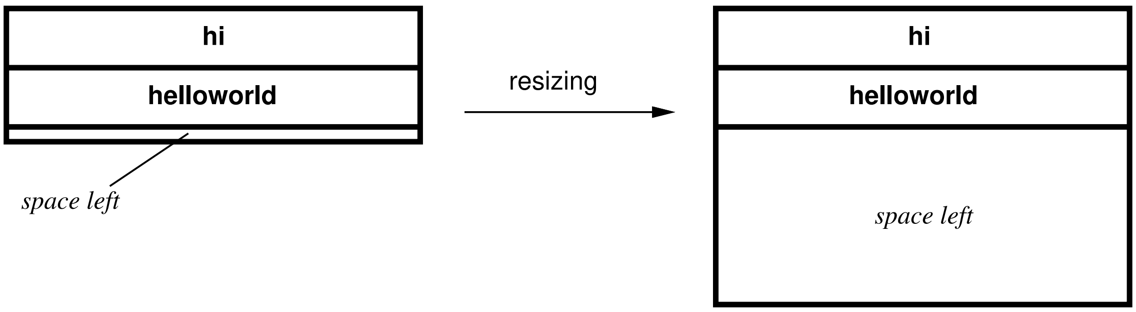
```
button .b1 -text hi
button .b2 -text helloworld
pack .b1 ; pack .b2
```

After the script is executed, there are two buttons in the wish-window. The upper button is a lot smaller than the lower one. Resizing the window has a limited effect: when the window is broadened, the buttons will shift to the middle of the window; when the window is heightened, the buttons will stay where they are.

Script 2:

```
button .b1 -text hi
button .b2 -text helloworld
pack .b1 -fill both ; pack .b2 -fill both
```

After the script has been executed, there are again two buttons in the wish-window, but this time, both buttons have the same size. The Packer has filled the upper button to contain the space that was left with script 1. Resizing the window has a somewhat different effect: when the window is broadened, the buttons will become as broad as the window; they will fill the available space in the rectangle. When the window is heightened, the buttons will not become any higher. This is because buttons are packed at the top by default and the areas containing the buttons will not become larger when the window is heightened; only the space that was left after the buttons were packed will become larger:



The strip of space left in the left part of the picture is not visible when the buttons are packed, but it **is** there.

Script 3:

```
button .b1 -text hi
button .b2 -text helloworld
pack .b1 -expand true -fill both ; pack .b2 -expand true -fill both
```

The buttons are of the same proportions as with script 2. When the window is resized now, the buttons will also grow in height. This is a result of the expand parameter: if the available space is enlarged (even if everything is already packed), widgets that are packed with '-expand true' will expand to contain the added space.

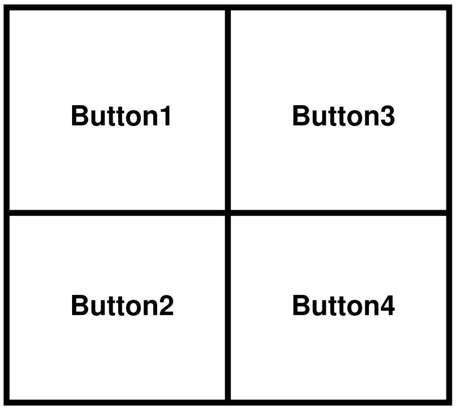
Script 4:

```
button .b1 -text hi
button .b2 -text helloworld
pack .b1 -expand true ; pack .b2 -expand true
```

The window now looks the same as after running script 1. Resizing the window has a peculiar effect: the buttons will end up in the middle of the available space. But because none of the buttons has been packed with a fill-parameter, the buttons are not enlarged. The difference between expand en fill should be clear now: 'expand' divides new available space between all widgets, 'fill' makes the widgets grow to fill the available space.

4 Using frames

Using only the abovementioned constructions it's not possible to create a window like



To create such a window, frames have to be used. A frame can be used as a kind of subwindow. Other widgets can be packed in a frame, and the frame (containing widgets) can then be packed to another window or frame. For the example this would mean that the leftmost buttons would be packed on top of eachother within a frame, and the frame would be packed at the left side of the window; space would then remain at the right side of the window. Next, the rightmost buttons would be packed at that remaining space.

For example:

```
frame* f = new frame(".frame");         // frame is a child of the
                                        // main-window
button* b1 = new button(f, ".b1", "-text Button1");
                                        // button is child of f
button* b2 = new button(f, ".b2", "-text Button2");

button* b3 = new button(".b3", "-text Button3");
                                        // buttons on rightside are
                                        // children of main-window
button* b4 = new button(".b4", "-text Button4");

b1 -> pack("-side top -fill both -expand true");
b2 -> pack("-side bottom -fill both -expand true");

f -> pack("-side left -fill both -expand true");

b3 -> pack("-side top -fill both -expand true");
b4 -> pack("-side bottom -fill both -expand true");
```

5 Other documentation

```
man -M /home/hush/man -s n pack
```

especially the section 'THE PACKER ALGORITHM'