Insert up to four players demo

by Halvard Tislavoll, 2024

"nameplayers" is a project that follows from the project "fourplayer". This time in Page I have loaded fourplayer.tcl and saved it under this new project named nameplayers.tcl

The Tcl file was loaded into Page and the development work started. I made a cover over the player buttons with a Label. But first I moved them into a new TFrame. It is as narrow as two of the game board's canvases. I made the buttons a little bigger, from 75 to 100 pixels to make room for longer names.

In order to end the program via an Exit button, a new Exit button was created.

The entry function for four players was organized via four TEntry boxes on a new TFrame. The frame is the same size as the one with nine canvases and is placed on top of it. At the top of the new TFrame there is a guiding text Label. A label is placed above each Entrybox with information about which spler numbers are registered. Next to the entry box is also placed an empty label which is intended to show the color of this player.

At the bottom of the frame there is a button labeled 'Go On' to register the players.

Here is the startup code for this project.

```
def nameplayers_startup():
    _top1.title("up to four players")
    sh.im_dict = {}   # a dict to hold all images from disc
    sh.rnd_im_dict={}   # a dict to hold all images in random
orderinit_players_image_dict(sh.rnd_im_dict)
    # init a image dict for each player
    sh.player1_im={}
    sh.player2_im={}
    sh.player3_im={}
    sh.player4_im={}
    sh.player_lst=[sh.player1_im, sh.player2_im, sh.player3_im, sh.player4_im]
    sh.player_name={}
    sh.player=0
    sh.form_color_lst=['blue', 'yellow', 'red', 'green']
    random.shuffle(sh.form_color_lst)
    _w1.Label1.configure(background=sh.form_color_lst[0])
    _w1.Label2.configure(background=sh.form_color_lst[1])
    _w1.Label3.configure(background=sh.form_color_lst[2])
    _w1.Label4.configure(background=sh.form_color_lst[3])
    sh.style = ttk.Style()
    make_mystyle()
    sh.id_lst=
['MyTButton1.TButton','MyTButton2.TButton','MyTButton3.TButton','MyTButton4.TButton']
    sh.button_lst=[_w1.TBtnPlayer1, _w1.TBtnPlayer2, _w1.TBtnPlayer3, _w1.TBtnPlayer4]
    set_form_color(sh.player)
    set_button_color()
    init_canvas()
    init_path()
    img_number = 36   # set number of images to load
    # make a list of all image names by List Comprehension
    sh.im_lst=[f'{x}.png' for x in range(1,img_number+1)]
    ## print(im_lst)   # ['1.png', '2.png', '3.png', '4.png', '5.png', '6.png', '7.png',
'8.png', '9.png', '10.png', '11.png', '12.png', '13.png', '14.png', '15.png', '16.png',
'17.png', '18.png', '19.png', '20.png', '21.png', '22.png', '23.png', '24.png', '25.png',
'26.png', '27.png', '28.png', '29.png', '30.png', '31.png', '32.png', '33.png', '34.png',
'35.png', '36.png']
    init_image_dict()
    sh.rnd_im_dict = dict_randomization(sh.im_dict)
    init_players_image_dict(sh.rnd_im_dict)
    player_dict=sh.player_lst[sh.player]
    # set binding to entry/return key
    _w1.TEntry1.bind('<Return>', retur_entry1)
    _w1.TEntry2.bind('<Return>', retur_entry2)
    _w1.TEntry3.bind('<Return>', retur_entry3)
    _w1.TEntry4.bind('<Return>', retur_entry4)
    # Trace changes in the entry value
    _w1.player1var.trace_add("write", validate_input1)
```

```
    _w1.player2var.trace_add("write", validate_input2)
    _w1.player3var.trace_add("write", validate_input3)
    _w1.player4var.trace_add("write", validate_input4)
    # set widget fokus
    _w1.TEntry1.tk_focusNext().focus_set()
    _w1.TEntry1.config(takefocus=1)
    display_image(player_dict) # player 1 at init time
```

Much of this is inherited from the previous project, but here the changes are commented.

_top1.title("up to four players"):

I have changed the title to tell what this project does

sh.player_name={}:

A new dict to hold the player names

random.shuffle(sh.form_color_lst):

Randomize form color list to vary from time to time the application is run

_w1.TEntry1.bind('', retur_entry1):

Set binding to TEntry1..4 boxes for key. Use a call to the "retur_entry" routine

_w1.player1var.trace_add("write", validate_input1):

Trace changes made possible for TEntry1..4 boxes by using a call to the "validate_input" routine

```
_w1.TEntry1.tk_focusNext().focus_set()
_w1.TEntry1.config(takefocus=1)
```

These two lines focus on the Entry box

```
def retur_entry1(*args):
    _w1.TEntry1.tk_focusNext().focus_set()
    _w1.TEntry1.config(takefocus=1)

def retur_entry2(*args):
    _w1.TEntry2.tk_focusNext().focus_set()
    _w1.TEntry2.config(takefocus=1)

def retur_entry3(*args):
    _w1.TEntry3.tk_focusNext().focus_set()
    _w1.TEntry3.config(takefocus=1)

def retur_entry4(*args):
    _w1.TBtnGoOn.tk_focusNext().focus_set()
    _w1.TBtnGoOn.config(takefocus=1)
```

These routines make sure to change the focus to the next entry box, and finally the "Go On" button

```
def validate_input1(*args):
    # start validating this entry
    validate_input(_w1.player1var)

def validate_input2(*args):
    # start validating this entry
    validate_input(_w1.player2var)

def validate_input3(*args):
    # start validating this entry
    validate_input(_w1.player3var)

def validate_input4(*args):
    # start validating this entry
    validate_input(_w1.player4var)

def validate_input(player_var):
    # Get the current value of the entry
    input_value = player_var.get()
    # Check if the input is a number
    if input_value.isdigit():
        pass
    else:
        if input_value.endswith(" "):   # space not allowed
            player_var.set('')
        elif input_value.isalpha():     # letter is allowed
            if input_value.istitle():   # first letter must be capitalized
                pass
                #print(f"Input: {input_value} (Word)")
            else:
                player_var.set('')
        else:
            player_var.set('')
```

Here, entries in the entry boxes are checked to ensure the text is in the correct form. First capital letter, no spaces or other characters than letters

```
def on_TBtnGoOn(*args):
    if _debug:
        print('nameplayers_support.on_TBtnGoOn')
        for arg in args:
            print ('    another arg:', arg)
        sys.stdout.flush()
    # read player 1
    sh.player_name[1]=_w1.player1var.get()
    if not len(sh.player_name[1])==0:
        _w1.TBtnPlayer1.configure(text=sh.player_name[1])
    else:
        _w1.TBtnPlayer1.destroy()
    # read player 2
    sh.player_name[2]=_w1.player2var.get()
    if not len(sh.player_name[2])==0:
        _w1.TBtnPlayer2.configure(text=sh.player_name[2])
    else:
        _w1.TBtnPlayer2.destroy()
    # read player 3
    sh.player_name[3]=_w1.player3var.get()
    if not len(sh.player_name[3])==0:
        _w1.TBtnPlayer3.configure(text=sh.player_name[3])
    else:
        _w1.TBtnPlayer3.destroy()
    # read player 4
    sh.player_name[4]=_w1.player4var.get()
    if not len(sh.player_name[4])==0:
        _w1.TBtnPlayer4.configure(text=sh.player_name[4])
    else:
        _w1.TBtnPlayer4.destroy()
    # remove the lid over players button
    _w1.TLblLid.destroy()
    # lift the canvas frame to top
    _w1.TFrame1.lift()
```

Here, each entry box is inspected. If the content is greater than zero characters, this will be written to the player number button. Otherwise, the player button is removed.

Finally, the Label covered over the player buttons is removed and the Frame with the canvases lifted at the top and become visible.

```
def on_TBtnExit(*args):
    if _debug:
        print('nameplayers_support.on_TBtnExit')
        for arg in args:
            print ('    another arg:', arg)
        sys.stdout.flush()
    sys.exit()
```

Most of this routine was created by Page. I have only activated it with the last line which stops the application and removes the Toplevel form from the screen.

Now the Application is functional. The next task may be to create an application that builds on this and adds functionality for the splash screen and instruction screen