C1A_frame.py

```
# ================================================================================
"""FRAME : demo program for the 'Win' and 'Frame' widgets"""
# ================================================================================
__author__  = "Christophe Schlick"
__version__ = "1.0" # one single 'Button' widget
__date__    = "2018-01-15"
# ================================================================================
from ezTK import *
# --------------------------------------------------------------------------------
def main():
  """create the main window and pack the widgets"""
  win = Win(title='FRAME') # create main window
  #Button(win, text='OOO\nOOO') # create a 'Button' widget in 'win'
  # Possible values for anchor: C (= default), N, NE, E, SE, S, SW, W, NW
  Button(win, text='OOO\nOOO', anchor='SW') # anchor text at South-West
  win.loop() # start event loop
# ================================================================================
if __name__ == "__main__":
  main()
# ================================================================================
```

C1B_frame.py

```
# ================================================================================
"""FRAME : demo program for the 'Win' and 'Frame' widgets"""
# ================================================================================
__author__  = "Christophe Schlick"
__version__ = "2.0" # 3 'Button' widgets with 1D packing
__date__    = "2018-01-15"
# ================================================================================
from ezTK import *
# --------------------------------------------------------------------------------
def main():
  """create the main window and pack the widgets"""
  # default 1D packing flow direction is South : flow='S'
  win = Win(title='FRAME') # use default packing flow (direction = S)
  #win = Win(title='FRAME', flow='N') # packing flow direction = N
  #win = Win(title='FRAME', flow='E') # packing flow direction = E
  #win = Win(title='FRAME', flow='W') # packing flow direction = W
  #win = Win(title='FRAME', ip=20) # inner padding in pixel units
  #win = Win(title='FRAME', op=5)  # outer padding in pixel units
  #win = Win(title='FRAME', bg='blue', fg='white') # background and foreground
  # ------------------------------------------------------------------------------
  #Button(win, text='OOOOOO')
  #Button(win, text='XXX\nXXX')
  #Button(win, text='IIIIII')
  Button(win, grow=False, text='OOOOOO')
  Button(win, bg='green', text='XXX\nXXX')
  Button(win, grow=False, text='IIIIII')
  # ------------------------------------------------------------------------------
  print("Number of widgets :", win.size) # get number of widgets used in 'win'
  for n in range(win.size): # loop over widgets and get their properties
    print("Text property for win[%s] = %r" % (n, win[n]['text']))
  # ------------------------------------------------------------------------------
  win[1]['text'] = 'ZZZ\nZZZ' # edit widget property (use widget index)
  # ------------------------------------------------------------------------------
  win.loop()
# ================================================================================
if __name__ == "__main__":
  main()
# ================================================================================
```

C1C_frame.py

```
# ================================================================================
"""FRAME : demo program for the 'Win' and 'Frame' widgets"""
# ================================================================================
__author__  = "Christophe Schlick"
__version__ = "3.0" # 24 'Button' widgets with 2D packing
__date__    = "2018-01-15"
# ================================================================================
from ezTK import *
# --------------------------------------------------------------------------------
def main():
  """create the main window and pack the widgets"""
  # default 2D packing flow direction is first East then South : flow='ES'
  win = Win(title='FRAME', fold=6) # fold the packing every 6 widgets
  #win = Win(title='FRAME', flow='NE', fold=3) # pack N then E, fold every 3
  #win = Win(title='FRAME', flow='WN', fold=12) # pack W then N, fold every 12
  # ------------------------------------------------------------------------------
  for loop in range(24):
    Button(win, text=loop, width=4)
  # ------------------------------------------------------------------------------
  #print("Number of widgets :", win.size) # get number of widgets as a tuple
  #rows, cols = win.size # number of rows, number of cols
  #for row in range(rows): # loop over widgets and get properties
  #  for col in range(cols):
  #    print("Button (%s,%s): text=%r" % (row, col, win[row][col]['text']))
  # ------------------------------------------------------------------------------
  #win[2][1]['bg'] = 'red' # edit widget properties (use widget coordinates)
  # ------------------------------------------------------------------------------
  win.loop()
# ================================================================================
if __name__ == "__main__":
  main()
# ================================================================================
```

C1D_frame.py

```
# ================================================================================
"""FRAME : demo program for the 'Win' and 'Frame' widgets"""
# ================================================================================
__author__  = "Christophe Schlick"
__version__ = "4.0" # use sub-frames to get 2D packing
__date__    = "2018-01-15"
# ================================================================================
from ezTK import *
# --------------------------------------------------------------------------------
def main():
  """create the main window and pack the widgets"""
  font1, font2 = 'Arial 14', 'Arial 32 bold' # define fonts
  #win = Win(title='FRAME', font=font1, op=2) # main window default flow='SE'
  win = Win(title='FRAME', font=font1, flow='WN') # pack W then N
  # ------------------------------------------------------------------------------
  Button(win, grow=False, text='OOOOOO')
  # ------------------------------------------------------------------------------
  frame = Frame(win) # inner Frame with default flow='ES' (orthogonal flow)
  Button(frame, grow=False, text='XXX\nXXX')
  Button(frame, font=font2, text='YYY\nYYY') # use specific font
  Button(frame, grow=False, text='ZZZ\nZZZ')
  # ------------------------------------------------------------------------------
  Button(win, grow=False, text='IIIIII')
  # ------------------------------------------------------------------------------
  print("Number of widgets for win :", win.size)
  print("Number of widgets for frame :", frame.size)
  print("Text property for win[0] : %r" % win[0]['text'])
```

```
  print("Text property for frame[0] : %r" % frame[0]['text'])
  # ------------------------------------------------------------------------
  frame[1]['text'] = '\u2660\u2663\u2665\u2666' # edit widget property
  # ------------------------------------------------------------------------
  win.loop()
# ============================================================================
if __name__ == "__main__":
  main()
# ============================================================================
```

C1E_frame.py

```
# ============================================================================
"""FRAME : demo program for the 'Win' and 'Frame' widgets"""
# ============================================================================
__author__  = "Christophe Schlick"
__version__ = "5.0" # complex packing by using several levels of sub-frames
__date__    = "2018-01-15"
# ============================================================================
from ezTK import *
# ----------------------------------------------------------------------------
def main():
  """create the main window and pack the widgets"""
  #win = Win(title='FRAME', font='Arial 14', op=2) # default flow='SE'
  win = Win(title='FRAME', font='Arial 14', flow='NE', op=2)
  # ------------------------------------------------------------------------
  fr1 = Frame(win, grow=False) # default flow='ES' (orthogonal flow)
  Button(fr1, text='OOOOOO')
  Button(fr1, grow=False, text='XXX\nXXX')
  Button(fr1, text='IIIIII')
  # ------------------------------------------------------------------------
  fr2 = Frame(win) # default flow='ES' (orthogonal flow)
  Button(fr2, grow=False, text='OOOOOO')
  fr3 = Frame(fr2, op=0) # default flow='SE' (orthogonal flow again)
  Button(fr3, text='XXX\nXXX')
  fr4 = Frame(fr3, op=2) # default flow='ES' (orthogonal flow again)
  Button(fr4, text='XYZ\nXYZ')
  Button(fr4, text='ZYX\nZYX')
  Button(fr2, grow=False, text='IIIIII')
  # ------------------------------------------------------------------------
  # Each widget can be accessed starting at any level of its parent hierarchy
  #win[0][2]['bg'], fr1[0]['bg'], fr2[2]['bg'] = 'cyan','magenta','yellow'
  #fr3[0]['bg'], fr3[1][0]['bg'], fr4[1]['bg'] = 'green','red','blue'
  # ------------------------------------------------------------------------
  win.loop()
# ============================================================================
if __name__ == "__main__":
  main()
# ============================================================================
```

C1F_frame.py

```
# ============================================================================
"""FRAME : demo program for the 'Win' and 'Frame' widgets"""
# ============================================================================
__author__  = "Christophe Schlick"
__version__ = "6.0" # complex packing by using folding and sub-frames
__date__    = "2018-01-15"
# ============================================================================
from ezTK import *
# ----------------------------------------------------------------------------
def main():
  """create the main window and pack the widgets"""
  win = Win(title='FRAME', font='Arial 14', fold=3, op=2)
  # ------------------------------------------------------------------------
  Button(win, text='OOOOOO')
  Button(win, grow=False, text='XXX\nXXX')
  Button(win, text='IIIIII')
  # ------------------------------------------------------------------------
  Button(win, grow=False, text='OOOOOO')
  frm = Frame(win, flow='EN', fold=2)
  Button(frm, text='XYZ\nXYZ')
  Button(frm, text='ZYX\nZYX')
  Button(frm, text='XXX\nXXX') # last widget fills all remaining space in row
  Button(win, grow=False, text='IIIIII')
  # ------------------------------------------------------------------------
  # Each widget can be accessed starting at any level of its parent hierarchy
  #win[0][2]['bg'], win[0][0]['bg'], win[1][2]['bg'] = 'cyan','magenta','yellow'
  #frm[1][0]['bg'], frm[0][0]['bg'], frm[0][1]['bg'] = 'green','red','blue'
  # ------------------------------------------------------------------------
  win.loop()
# ============================================================================
if __name__ == "__main__":
  main()
# ============================================================================
```

C1G_frame.py

```
# ============================================================================
"""FRAME : demo program for the 'Win' and 'Frame' widgets"""
# ============================================================================
__author__  = "Christophe Schlick"
__version__ = "7.0" # add callback function with 'command' option
__date__    = "2018-01-15"
# ============================================================================
from ezTK import *
# ----------------------------------------------------------------------------
def main():
  """create the main window and pack the widgets"""
  font1, font2 = 'Arial 14', 'Arial 48 bold'
  win = Win(title='FRAME', font=font1, op=2)
  # ------------------------------------------------------------------------
  fr1 = Frame(win, grow=False)
  Button(fr1, text='IIIIII')
  Button(fr1, grow=False, text='XXX\nXXX')
  Button(fr1, text='IIIIII')
  # ------------------------------------------------------------------------
  fr2 = Frame(win)
  fr3 = Frame(fr2, grow=False)
  Button(fr3, grow=False, text='OOOOOO')
  Button(fr3, text='XXX\nXXX')
  Button(fr3, grow=False, text='OOOOOO')
  Button(fr2, text='EXIT', font=font2, fg='#FFF', bg='#00F', command=win.exit)
  Button(fr2, grow=False, text='OOOOOO')
  # ------------------------------------------------------------------------
  fr4 = Frame(win, grow=False)
  Button(fr4, grow=False, text='XXX\nXXX')
  Button(fr4, text='IIIIII')
  Button(fr4, grow=False, text='XXX\nXXX')
  # ------------------------------------------------------------------------
  win.loop()
# ============================================================================
if __name__ == "__main__":
  main()
# ============================================================================
```

C2A_toggle.py

```
# ============================================================================
"""TOGGLE : demo program for simple animation of multi-state widgets"""
```

```
# ================================================================================
__author__  = "Christophe Schlick"
__version__ = "1.0" # perform animation by manual editing of widget properties
__date__    = "2018-01-15"
# ================================================================================
from ezTK import *
# --------------------------------------------------------------------------------
def tick():
  """update function for widget animation"""
  # toggle the background color of the widget located at coordinates (2,2)
  win[2][2]['bg'] = '#F00' if win[2][2]['bg'] == '#00F' else '#00F'
  win.after(100, tick) # launch 'tick' again in 1000 ms
# --------------------------------------------------------------------------------
def main():
  """create the main window and pack the widgets"""
  global win # always define 'win' as a global variable
  win = Win(title='TOGGLE', fold=5) # fold every 5 widgets
  # ------------------------------------------------------------------------------
  for loop in range(25) :
    Brick(win, height=64, width=64, bg='#00F')
    #Brick(win, height=64, width=64, bg='#00F', border=0) # remove borders
  # ------------------------------------------------------------------------------
  win.after(5000, tick) # launch 'tick' in 1000 ms
  win.loop()
# ================================================================================
if __name__ == "__main__":
  main()
# ================================================================================
```

C2B_toggle.py

```
# ================================================================================
"""TOGGLE : demo program for simple animation of multi-state widgets"""
# ================================================================================
__author__  = "Christophe Schlick"
__version__ = "2.0" # animation is much easier with multi-state widgets
__date__    = "2018-01-15"
# ================================================================================
from ezTK import *
from random import randrange
# --------------------------------------------------------------------------------
def tick():
  """update function for widget animation"""
  row, col = randrange(win.size[0]), randrange(win.size[1]) # random position
  print(row,col)
  widget = win[row][col] # get widget at selected position
  widget.state = randrange(6) # modify state of widget (automatic cycling is p……
……rovided)
  win.after(500, tick) # launch 'tick' again in 20 ms
# --------------------------------------------------------------------------------
def main():
  """create the main window and pack the widgets"""
  global win # always define 'win' as a global variable
  win = Win(title='TOGGLE', fold=5) # fold the packing every 5 widgets
  # ------------------------------------------------------------------------------
  colors = ('#F00','#0F0','#00F','#0FF','#F0F','#FF0')
  for loop in range(30):
    Brick(win, height=64, width=64, bg=colors, state=3) # multi-state 'Brick'
  # ------------------------------------------------------------------------------
  print(win.size, win.size[0], win.size[1])
  win.after(1000, tick); win.loop()
# ================================================================================
if __name__ == "__main__":
  main()
```

```
# ================================================================================
```

C2C_toggle.py

```
# ================================================================================
"""TOGGLE : demo program for simple animation of multi-state widgets"""
# ================================================================================
__author__  = "Christophe Schlick"
__version__ = "3.0" # use image Labels instead of text Labels
__date__    = "2018-01-15"
# ================================================================================
from ezTK import *
from random import randrange
# --------------------------------------------------------------------------------
def tick():
  """update function for widget animation"""
  widget = win[randrange(win.size[0])][randrange(win.size[1])]
  widget.state += 1 # modify state of selected widget
  win.after(20, tick) # launch 'tick' again in 20 ms
# --------------------------------------------------------------------------------
def main():
  """create the main window and pack the widgets"""
  global win # always define 'win' as a global variable
  win = Win(title='TOGGLE', fold=5, bg='#000', op=2)
  # ------------------------------------------------------------------------------
  image = tuple(Image(file=f"Z{color}.gif") for color in 'RGBCMY')
  # alternative version without list comprehension:
  #image = (Image(file='ZR.gif'), Image(file='ZG.gif'),
  #         Image(file='ZB.gif'), Image(file='ZC.gif'),
  #         Image(file='ZM.gif'), Image(file='ZY.gif'))
  for loop in range(25):
    Label(win, image=image, state=3) # use 3 as default state for grid widgets
  # ------------------------------------------------------------------------------
  win.after(1000, tick) # launch 'tick' in 1000 ms
  win.loop()
# ================================================================================
if __name__ == "__main__":
  main()
# ================================================================================
```

C2D_toggle.py

```
# ================================================================================
"""TOGGLE : demo program for simple animation of multi-state widgets"""
# ================================================================================
__author__  = "Christophe Schlick"
__version__ = "4.0" # combine several properties in multi-state widgets
__date__    = "2018-01-15"
# ================================================================================
from ezTK import *
from random import randrange
# --------------------------------------------------------------------------------
def tick():
  """update function for widget animation"""
  widget = win[randrange(win.size[0])][randrange(win.size[1])] # random widget
  widget.state = randrange(widget.states) # random state for selected widget
  win.after(50, tick) # launch 'tick' again in 50 ms
# --------------------------------------------------------------------------------
def main(rows=3, cols=5):
  """create the main window and pack the widgets"""
  global win # always define 'win' as a global variable
  bg = ('#F00','#0F0','#00F','#0FF','#F0F','#FF0','#000')
  fg = ('#FFF','#000') # number of states may be different for each property
  text = ('RED', 'GREEN', 'BLUE', 'CYAN', 'MAGENTA', 'YELLOW','BLACK')
  win = Win(title='TOGGLE', font='Arial 16 bold', fold=cols, op=2)
```

```
  # ------------------------------------------------------------------------
  for loop in range(rows*cols): # loop over the cells of the grid
    row, col = loop//cols, loop%cols # get cell coords by Euclidian division
    state = (col+row)%len(text) # generate initial state for current cell
    Label(win, height=3, width=9, bg=bg, fg=fg, text=text, state=state)
  # ------------------------------------------------------------------------
  win.after(6000, tick) # launch 'tick' in 2000 ms
  win.loop()
# ============================================================================
if __name__ == "__main__":
  main(7,6)
# ============================================================================
```

C3A_message.py

```
# ============================================================================
"""MESSAGE : demo program for simple callback functions"""
# ============================================================================
__author__  = "Christophe Schlick"
__version__ = "1.0" # use specific callback function for each button
__date__    = "2018-01-15"
# ============================================================================
from ezTK import *
# ----------------------------------------------------------------------------
def on_AAA():
  """callback function for button AAA"""
  win.label['text'],win.label['fg'],win.label['bg'] = 'RETRY','#FFF','#F70'
# ----------------------------------------------------------------------------
def on_BBB():
  """callback function for button BBB"""
  win.label['text'],win.label['fg'],win.label['bg'] = 'YOU WIN !','#000','#0F0'
# ----------------------------------------------------------------------------
def on_CCC():
  """callback function for button CCC"""
  win.label['text'],win.label['fg'],win.label['bg'] = 'GAME OVER','#FFF','#F00'
# ----------------------------------------------------------------------------
def main():
  """create the main window and pack the widgets"""
  global win
  font1, font2 = 'Arial 14', 'Arial 18 bold'
  win = Win(title='MESSAGE', font=font2, op=5, grow=False)
  # ------------------------------------------------------------------------
  text, fg, bg = 'Try to find the correct button', '#FFF', '#00F'
  Label(win, text=text, fg=fg, bg=bg, width=25, height=2, border=2)
  # ------------------------------------------------------------------------
  frame = Frame(win, font=font1, relief='groove', border=2, op=5)
  Button(frame, text='AAA', command=on_AAA)
  Button(frame, text='BBB', command=on_BBB)
  Button(frame, text='CCC', command=on_CCC)
  # ------------------------------------------------------------------------
  win.label = win[0] # set friendly names for all widgets used in callbacks
  win.loop()
# ============================================================================
if __name__ == '__main__':
  main()
# ============================================================================
```

C3B_message.py

```
# ============================================================================
"""MESSAGE : demo program for simple callback functions"""
# ============================================================================
__author__  = "Christophe Schlick"
__version__ = "2.0" # use the same generic callback function for all widgets
__date__    = "2018-01-15"
# ============================================================================
from ezTK import *
# ----------------------------------------------------------------------------
def on_button(index):
  """generic callback function for all three buttons"""
  win.label['text'] = ('RETRY','YOU WIN !','GAME OVER')[index]
  win.label['fg'] = ('#FFF','#000','#FFF')[index]
  win.label['bg'] = ('#F70','#0F0','#F00')[index]
# ----------------------------------------------------------------------------
def main():
  """create the main window and pack the widgets"""
  global win
  font1, font2 = 'Arial 14', 'Arial 18 bold'
  win = Win(title='MESSAGE', font=font2, op=5, grow=False)
  # ------------------------------------------------------------------------
  text, fg, bg = 'Try to find the correct button', '#FFF', '#00F'
  Label(win, text=text, fg=fg, bg=bg, width=25, height=2, border=2)
  # ------------------------------------------------------------------------
  frame = Frame(win, font=font1, relief='groove', border=2, op=5)
  Button(frame, text='AAA', command=lambda: on_button(0)) # index = 0
  Button(frame, text='BBB', command=lambda: on_button(1)) # index = 1
  Button(frame, text='CCC', command=lambda: on_button(2)) # index = 2
  # ------------------------------------------------------------------------
  win.label = win[0] # set friendly names for all widgets used in callbacks
  win.loop()
# ============================================================================
if __name__ == '__main__':
  main()
# ============================================================================
```

C3C_message.py

```
# ============================================================================
"""MESSAGE : demo program for simple callback functions"""
# ============================================================================
__author__  = "Christophe Schlick"
__version__ = "3.0" # use a multi-state 'Label' widget
__date__    = "2018-01-15"
# ============================================================================
from ezTK import *
def on_button(index):
  """generic callback function for all three buttons"""
  win.label.state = index
# ----------------------------------------------------------------------------
def main():
  """create the main window and pack the widgets"""
  global win
  font1, font2 = 'Arial 14', 'Arial 18 bold'
  win = Win(title='MESSAGE', font=font2, op=5, grow=False)
  # ------------------------------------------------------------------------
  # define multi-state values for 'text', 'bg' and 'fg' properties
  text = ('Try to find the correct button','RETRY','YOU WIN !','GAME OVER')
  fg, bg = ('#FFF','#FFF','#000','#FFF'), ('#00F','#F70','#0F0','#F00')
  Label(win, text=text, fg=fg, bg=bg, width=25, height=2, border=2)
  # ------------------------------------------------------------------------
  frame = Frame(win, font=font1, relief='groove', border=2, op=5)
  Button(frame, text='AAA', command=lambda: on_button(1)) # set state to 1
  Button(frame, text='BBB', command=lambda: on_button(2)) # set state to 2
  Button(frame, text='CCC', command=lambda: on_button(3)) # set state to 3
  # ------------------------------------------------------------------------
  win.label = win[0] # set friendly names for all widgets used in callbacks
  win.loop()
# ============================================================================
if __name__ == '__main__':
```

```
  main()
# ================================================================================
```

C4A_random.py

```
# ================================================================================
"""RANDOM : generate random values within a user-provided range"""
# ================================================================================
__author__  = "Christophe Schlick"
__version__ = "1.0" # use a Label widget to display single random value
__date__    = "2018-01-15"
# ================================================================================
from ezTK import *
from random import randrange
# --------------------------------------------------------------------------------
def on_scale(flag):
  """callback function for the 'MIN' and 'MAX' scales"""
  # get value of active scale, according to flag
  active = win.minscale.state if flag == 0 else win.maxscale.state
  win.minscale.state = min(win.minscale.state, active)
  win.maxscale.state = max(win.maxscale.state, active)
# --------------------------------------------------------------------------------
def on_random():
  """callback function for the 'RANDOM' button"""
  win.label['text'] = randrange(win.minscale.state, win.maxscale.state+1)
# --------------------------------------------------------------------------------
def main(minvalue=0, maxvalue=999):
  """create the main window and pack the widgets"""
  global win
  win = Win(title='RANDOM', op=5)
  # ------------------------------------------------------------------------------
  frame = Frame(win, grow=False, op=0)
  Label(frame, grow=False, text='MIN :', anchor='SE')
  Scale(frame, scale=(minvalue, maxvalue), state=minvalue,
        command=lambda: on_scale(0)) # flag = 0
  Label(frame, grow=False, text='MAX :', anchor='SE')
  Scale(frame, scale=(minvalue, maxvalue), state=maxvalue,
        command=lambda: on_scale(1)) # flag = 1
  Button(win, grow=False, text='RANDOM', command=on_random)
  Label(win, font='Arial 72 bold', width=3, border=2)
  # ------------------------------------------------------------------------------
  # set friendly names for all widgets used in callbacks
  win.label, win.minscale, win.maxscale = win[2], frame[1], frame[3]; win.loop()
# ================================================================================
if __name__ == '__main__':
  main()
# ================================================================================
```

C4B_random.py

```
# ================================================================================
"""RANDOM : generate random values within a user-provided range"""
# ================================================================================
__author__  = "Christophe Schlick"
__version__ = "2.0" # use 'Entry' widget to define range
__date__    = "2018-01-15"
# ================================================================================
from ezTK import *
from random import randrange
# --------------------------------------------------------------------------------
def on_entry():
  """callback function for the 'min,max' entry"""
  try: # try to parse the entry string as a couple of integer values
    minvalue, maxvalue = win.entry.state.split(',')
    minvalue, maxvalue = int(minvalue), int(maxvalue)
    win.min, win.max = min(minvalue, maxvalue), max(minvalue, maxvalue)
  except Exception:
    pass # keep previous values if the parsing fails
  win.entry.state = f"{win.min}, {win.max}"
# --------------------------------------------------------------------------------
def on_random():
  """callback function for the 'RANDOM' button"""
  on_entry() # reparse the entry string as user may forget to hit 'ENTER'
  win.label['text'] = randrange(win.min, win.max+1)
# --------------------------------------------------------------------------------
def main(minvalue=0, maxvalue=99):
  """create the main window and pack the widgets"""
  global win
  win = Win(title='RANDOM', op=5)
  # ------------------------------------------------------------------------------
  frame = Frame(win, grow=False, border=1, op=5)
  Label(frame, grow=False, text='Enter min,max :')
  Entry(frame, width=10, command=on_entry)
  Button(win, grow=False, text='RANDOM', command=on_random)
  Label(win, font='Arial 72 bold', width=3, border=1)
  # ------------------------------------------------------------------------------
  win.label, win.entry, win.min, win.max = win[2], frame[1], minvalue, maxvalue
  on_entry(); win.loop()
# ================================================================================
if __name__ == '__main__':
  main()
# ================================================================================
```

C4C_random.py

```
# ================================================================================
"""RANDOM : generate random values within a user-provided range"""
# ================================================================================
__author__  = "Christophe Schlick"
__version__ = "3.0" # store random values in a scrollable listbox
__date__    = "2018-01-15"
# ================================================================================
from ezTK import *
from random import randrange
# --------------------------------------------------------------------------------
def on_entry():
  """callback function for the 'min,max' entry"""
  try: # try to parse the entry string as a couple of integer values
    minvalue, maxvalue = win.entry.state.split(',')
    minvalue, maxvalue = int(minvalue), int(maxvalue)
    win.min, win.max = min(minvalue, maxvalue), max(minvalue, maxvalue)
  except Exception:
    pass # keep previous values if the parsing fails
  win.entry.state = f"{win.min}, {win.max}"
# --------------------------------------------------------------------------------
def on_random():
  """callback function for the 'RANDOM' button"""
  on_entry() # reparse the entry string as user may forget to hit 'ENTER'
  minvalue, maxvalue, size = win.min, win.max, len(win.box)
  values = ["%2s" % randrange(minvalue,maxvalue+1) for loop in range(size+1)]
  win.box.append(' '.join(values)) # append new values as a single line
  #win.box(' '.join(values)) # replace box content with new values
# --------------------------------------------------------------------------------
def on_delete():
  """callback function for the 'DELETE' button"""
  del win.box[-1] # delete last line
  #del win.box[0:-1]  # delete all lines
# --------------------------------------------------------------------------------
def main(minvalue=0, maxvalue=99):
```

```
  """create the main window and pack the widgets"""
  global win
  win = Win(title='RANDOM', op=5)
  # ------------------------------------------------------------------------
  frame1 = Frame(win, grow=False, border=1, op=5)
  Label(frame1, grow=False, text='Enter min,max :')
  Entry(frame1, width=10, command=on_entry)
  frame2 = Frame(win, grow=False, op=0)
  Button(frame2, text='RANDOM', command=on_random)
  Button(frame2, text='DELETE', command=on_delete)
  Listbox(win, grow=True, scroll=True, width=30, height=15)
  # ------------------------------------------------------------------------
  win.box, win.entry, win.min, win.max = win[2], frame1[1], minvalue, maxvalue
  on_entry(); win.loop()
# ============================================================================
if __name__ == '__main__':
  main(0,99)
# ============================================================================
```

C5A_win.py

```
# ============================================================================
"""WIN : demo program for window manipulations"""
# ============================================================================
__author__  = "Christophe Schlick"
__version__ = "1.0" # dynamic creation and destruction of windows
__date__    = "2018-01-15"
# ============================================================================
from ezTK import *
# ----------------------------------------------------------------------------
# Note: three different kinds of windows may be created by calling Win(...):
# - a MASTER window, when the first argument of Win(...) is None
# - a SLAVE window, when the first argument represents an existing window
# - a MODAL window, is a special type of SLAVE window that blocks all events
#   of its MASTER window until the MODAL window is closed
# ----------------------------------------------------------------------------
def window(master=None, modal=False):
  """create a new window (either master, slave, modal)"""
  global counter
  counter +=1; win = Win(master, title=counter, op=2)
  # use specific 'text' and 'bg' according to window type (master, slave, modal)
  if master is None: text, bg = 'MASTER', '#0F0'
  elif modal: text, bg = 'MODAL of window %s' % master.title, '#F00'
  else: text, bg = 'SLAVE of window %s' % master.title, '#FF0'
  Label(win, text=text, bg=bg)
  Button(win, text='Create master window', command=lambda: window())
  Button(win, text='Create slave window',  command=lambda: window(win))
  Button(win, text='Create modal window',  command=lambda: window(win, True))
  Button(win, text='Kill me and all my slaves', command=win.exit)
  # modal window (= blocking window) requires win.wait() instead of win.loop()
  win.wait() if modal else win.loop()
# ============================================================================
if __name__ == "__main__":
  counter = 0 # 'counter' is a global variable used for windows numbering
  window() # create window with default arguments (= master window)
# ============================================================================
```

C5B_win.py

```
# ============================================================================
"""WIN : demo program for window manipulations"""
# ============================================================================
__author__  = "Christophe Schlick"
__version__ = "2.0" # dynamic creation and destruction of widgets
__date__    = "2018-01-15"
# ============================================================================
from ezTK import *
# ----------------------------------------------------------------------------
def grid():
  """create the grid within the main window and pack the widgets"""
  rows, cols = win.rowscale.state, win.colscale.state # get current scale values
  del win[1] # delete the frame containing the previous grid
  frame = Frame(win, fold=cols) # create a new frame to store the new grid
  for loop in range(rows*cols) :
    Brick(frame, height=64, width=64, bg='#00F')
# ----------------------------------------------------------------------------
def main():
  """create the main window and pack the widgets"""
  global win, rowscale, colscale # variables needed in auxiliary functions
  win = Win(title='CONFIG', grow=False, op=2)
  frame = Frame(win, fold=2)
  # ------------------------------------------------------------------------
  Label(frame, text='Number of rows :', grow=False, width=13, anchor='SW')
  Scale(frame, scale=(1,8), flow='W')
  Label(frame, text='Number of cols :', grow=False, width=13, anchor='SW')
  Scale(frame, scale=(1,8), flow='W')
  Button(frame, text='NEW GRID', command=grid)
  Frame(win) # create an empty frame to store the grid
  # ------------------------------------------------------------------------
  win.rowscale, win.colscale = frame[0][1], frame[1][1]; win.loop()
# ============================================================================
if __name__ == "__main__":
  main()
# ============================================================================
```

C5C_win.py

```
# ============================================================================
"""WIN : demo program for window manipulations"""
# ============================================================================
__author__  = "Christophe Schlick"
__version__ = "3.0" # toggle between two different master windows
__date__    = "2018-01-15"
# ============================================================================
from ezTK import *
# ----------------------------------------------------------------------------
def grid():
  """create the grid window and pack the widgets"""
  global win
  rows, cols = win.rowscale(), win.colscale() # get grid size from scales
  win.exit() # exit 'config' window before creating 'grid' window
  win = Win(title='GRID', grow=False, fold=cols, op=2) # create new grid window
  for loop in range(rows*cols): Brick(win, height=64, width=64, bg='#00F')
  # ------------------------------------------------------------------------
  win.loop(); config(rows,cols) # relaunch 'config' win when 'grid' is closed
# ----------------------------------------------------------------------------
def config(rows=8, cols=8):
  """create the config window and pack the widgets"""
  global win
  win = Win(title='CONFIG', grow=False, fold=2, op=2) # create new config window
  # ------------------------------------------------------------------------
  Label(win, text='Number of rows :', grow=False, width=13, anchor='SW')
  Scale(win, scale=(1,8), flow='W', state=rows)
  Label(win, text='Number of cols :', grow=False, width=13, anchor='SW')
  Scale(win, scale=(1,8), flow='W', state=cols)
  Button(win, text='NEW GRID', command=grid)
  # ------------------------------------------------------------------------
  win.rowscale, win.colscale = win[0][1], win[1][1]; win.loop()
# ============================================================================
```

```
if __name__ == "__main__":
  config()
# ================================================================================
```

C5D_win.py

```
# ================================================================================
"""WIN : demo program for window manipulations"""
# ================================================================================
__author__  = "Christophe Schlick"
__version__ = "4.0" # test some standard dialog windows
__date__    = "2018-01-15"
# ================================================================================
from ezTK import *
# --------------------------------------------------------------------------------
def message(text):
  """change message shown on status bar"""
  win.label['text'] = text
# --------------------------------------------------------------------------------
def on_info():
  """callback for the "INFO" button"""
  message("INFO button has been pressed")
  MessageDialog.showinfo('INFO', message='This is an information message')
  message("INFO window has been closed")
# --------------------------------------------------------------------------------
def on_warn():
  """callback for the "WARN" button"""
  message("WARN button has been pressed")
  MessageDialog.showwarning('WARN', message='This is a warning message')
  message("WARN window has been closed")
# --------------------------------------------------------------------------------
def on_error():
  """callback for the "ERROR" button"""
  message("ERROR button has been pressed")
  MessageDialog.showerror('ERROR', message='This is an error message')
  message("ERROR window has been closed")
# --------------------------------------------------------------------------------
def on_yesno():
  """callback for the "YES-NO" button"""
  message("YES/NO button has been pressed")
  val = MessageDialog.askyesno('YES-NO', message='Select YES or NO')
  message("Value = %s" % val)
# --------------------------------------------------------------------------------
def on_color():
  """callback for the "COLOR" button"""
  message("COLOR button has been pressed")
  val = ColorDialog.askcolor()
  message("RGB = %s Color = %s" % val)
# --------------------------------------------------------------------------------
def on_open():
  """callback for the "OPEN FILE" button"""
  message("OPEN FILE button has been pressed")
  val = FileDialog.askopenfilename(title='OPEN FILE')
  message("File = %s" % val)
# --------------------------------------------------------------------------------
def on_save():
  """callback for the "SAVE FILE" button"""
  message("SAVE FILE button has been pressed")
  val = FileDialog.asksaveasfilename(title='SAVE FILE')
  message("File = %s" % val)
# --------------------------------------------------------------------------------
def on_popup():
  """callback for the "POPUP" button"""
  message("POPUP button has been pressed")
  popup = Win(win, title='POPUP', flow='E', op=10)
  Label(popup, text='This is a modal window\n\nPlease close it to continue',
    width=20, height=3, relief='flat')
  popup.wait() # wait until popup window has been closed
  message("POPUP window has been closed")
# --------------------------------------------------------------------------------
def main():
  """create the main window and pack the widgets"""
  global win
  win = Win(title='DIALOG', grow=False, fold=4, op=2)
  # ------------------------------------------------------------------------------
  Button(win, text='INFO', width=12, command=on_info)
  Button(win, text='WARN', width=12, command=on_warn)
  Button(win, text='ERROR', width=12, command=on_error)
  Button(win, text='YES-NO', width=12, command=on_yesno)
  Button(win, text='COLOR', width=12, command=on_color)
  Button(win, text='OPEN FILE', width=12, command=on_open)
  Button(win, text='SAVE FILE', width=12, command=on_save)
  Button(win, text='POPUP', width=12, command=on_popup)
  # ------------------------------------------------------------------------------
  win.label = Label(win, text='', relief='groove', anchor='W')
  win.loop()
# ================================================================================
if __name__ == '__main__':
  main()
# ================================================================================
```

C6A_event.py

```
# ================================================================================
"""EVENT : demo program for keyboard and mouse event handlers"""
# ================================================================================
__author__  = "Christophe Schlick"
__version__ = "1.0" # check mouse events (move)
__date__    = "2018-01-15"
# ================================================================================
from ezTK import *
# --------------------------------------------------------------------------------
def on_move(widget, code, mods):
  """callback function for all 'mouse move' events"""
  # display event parameters, only when 'win.brick' is the active widget
  if widget == win.brick: display('move', code, mods)
# --------------------------------------------------------------------------------
def display(event, code, mods):
  """display event parameters"""
  win.label['text'] = "Event = %r  Code = %r  Mods = %r" % (event, code, mods)
# --------------------------------------------------------------------------------
def main():
  """create the main window and pack the widgets"""
  global win
  win = Win(title='EVENT', grow=False, op=3, move=on_move)
  Label(win, font='Arial 14', height=2, border=2)
  Brick(win, width=640, height=480, bg='#00F')
  # ------------------------------------------------------------------------------
  win.label, win.brick = win[0], win[1]; win.loop()
# ================================================================================
if __name__ == '__main__':
  main()
# ================================================================================
```

C6B_event.py

```
# ================================================================================
"""EVENT : demo program for keyboard and mouse event handlers"""
# ================================================================================
```

```
__author__  = "Christophe Schlick"
__version__ = "2.0" # check mouse events (inout and click)
__date__    = "2018-01-15"
# ============================================================================
from ezTK import *
# ----------------------------------------------------------------------------
def on_click(widget, code, mods):
  """callback function for all 'mouse click' events"""
  if widget.master != win.grid or widget.index is None:
    return # nothing to do if the widget is not a grid cell
  display('click', widget.index, code, mods)
  if   code == 'LMB': widget.state += 1 # increment state
  elif code == 'RMB': widget.state -= 1 # decrement state
  elif code == 'MMB': reset() # reset grid state
# ----------------------------------------------------------------------------
def on_inout(widget, code, mods):
  """callback function for all 'mouse in' or 'mouse out' events"""
  if widget.master != win.grid or widget.index is None:
    return # nothing to do if the widget is not a grid cell
  display('inout', widget.index, code, mods)
  if code: widget['bg'] = '#FFF' # code = 1 for mouse in --> white background
  else: widget.state += 0 # code = 0 for mouse out --> restore background
# ----------------------------------------------------------------------------
def reset():
  """reset initial windows state"""
  win.label['text'] = '' # clear label widget
  rows, cols = win.grid.size
  for loop in range(rows*cols): # loop over grid cells
    row, col = loop // cols, loop % cols # get coordinates by Euclidian division
    win.grid[row][col].state = 0 # reset each cell
# ----------------------------------------------------------------------------
def display(event, index, code, mods):
  """display event parameters"""
  win.label['text'] = "Event = %r  Index = %r  Code = %r  Mods = %r"\
    % (event, index, code, mods)
# ----------------------------------------------------------------------------
def main(rows=3, cols=12):
  """create the main window and pack the widgets"""
  global win
  colors = ('#00F','#0F0','#F00','#0FF','#F0F','#FF0')
  win = Win(title='EVENT', op=3, grow=False, click=on_click, inout=on_inout)
  Label(win, font='Arial 14', height=2, border=2)
  grid = Frame(win, grow=False, fold=cols)
  for loop in range(rows*cols):
    Brick(grid, height=64, width=64, bg=colors)
  # ----------------------------------------------------------------------------
  win.label, win.grid = win[0], win[1]; win.loop()
# ============================================================================
if __name__ == '__main__':
  main()
# ============================================================================
```

C6C_event.py

```
# ============================================================================
"""EVENT : demo program for keyboard and mouse event handlers"""
# ============================================================================
__author__  = "Christophe Schlick"
__version__ = "3.0" # check keyboard events (key)
__date__    = "2018-01-15"
# ============================================================================
from ezTK import *
# ----------------------------------------------------------------------------
def on_key(widget, code, mods):
  """callback function for all 'key' events"""
  # Hint: len(code) == 1 means printable character
  win.char['text'] = code if len(code) == 1 else ''; display('key', code, mods)
# ----------------------------------------------------------------------------
def display(event, code, mods):
  """display event parameters"""
  win.label['text'] = "Event = %r  Code = %r  Mods = %r" % (event, code, mods)
# ----------------------------------------------------------------------------
def main():
  """create the main window and pack the widgets"""
  global win
  font1, font2 = 'Arial 14', 'Arial 48 bold'
  win = Win(title='EVENT', grow=False, op=3, key=on_key)
  Label(win, font='Arial 14', height=2, width=50, border=2)
  Label(win, font='Arial 48 bold', bg='#00F', fg='#FFF', border=2)
  # ----------------------------------------------------------------------------
  win.label, win.char = win[0], win[1]; win.loop()
# ============================================================================
if __name__ == '__main__':
  main()
# ============================================================================
```

C6D_event.py

```
# ============================================================================
"""EVENT : demo program for keyboard and mouse event handlers"""
# ============================================================================
__author__  = "Christophe Schlick"
__version__ = "4.0" # use arrow keys to control cursor movement
__date__    = "2018-01-15"
# ============================================================================
from ezTK import *
# ----------------------------------------------------------------------------
def on_key(widget, code, mods):
  """callback function for all 'key' events"""
  moves = {'Up':(1,0), 'Down':(-1,0), 'Right':(0,1), 'Left':(0,-1)}
  if code not in moves: return # nothing to do if key is not an arrow key
  # compute new cursor position by using modulo to get automatic cycling
  row = (win.cursor.index[0] + moves[code][0]) % win.rows
  col = (win.cursor.index[1] + moves[code][1]) % win.cols
  if win[row][col].state == 2: return # cursor blocked by red square
  win.cursor.state = 0; win.cursor = win[row][col]; win.cursor.state = 1 # move
# ----------------------------------------------------------------------------
def main(rows=5, cols=5, size=64):
  """create the main window and pack the widgets"""
  global win
  # use flow='EN' to get standard Cartesian system (X = right, Y = up)
  win = Win(title='EVENT', flow='EN', fold=cols, key=on_key, grow=False)
  # ----------------------------------------------------------------------------
  for loop in range(rows*cols):
    Brick(win, height=size, width=size, bg=('#00F','#0F0','#F00'))
  # ----------------------------------------------------------------------------
  win.rows, win.cols = rows, cols
  # put cursor (= green cell) at the center of the grid
  win.cursor = win[rows//2][cols//2]; win.cursor.state = 1
  # put some walls (= red cells) near the corners of the grid
  walls = ((0,0),(1,0),(0,1),(-1,-1),(-2,-1),(-1,-2),(-1,0),(0,-1))
  for row,col in walls: win[row][col].state = 2
  # ----------------------------------------------------------------------------
  win.loop()
# ============================================================================
if __name__ == '__main__':
  main(7,9)
  #main(32,32,20)
```

C6E_event.py

```
# ================================================================================
"""EVENT : demo program for key press and mouse click event handlers"""
# ================================================================================
__author__  = "Christophe Schlick"
__version__ = "5.0" # combine time events and user events
__date__    = "2018-01-15"
# ================================================================================
from ezTK import *
from random import choice
# --------------------------------------------------------------------------------
def move(code):
  """move the green square according to 'code'"""
  moves = {'Up':(1,0), 'Down':(-1,0), 'Right':(0,1), 'Left':(0,-1)}
  # compute new cursor position by using modulo to get automatic cycling
  row = (win.cursor.index[0] + moves[code][0]) % win.rows
  col = (win.cursor.index[1] + moves[code][1]) % win.cols
  if win[row][col].state == 2: return # cursor blocked by red square
  win.cursor.state = 0; win.cursor = win[row][col]; win.cursor.state = 1 # move
# --------------------------------------------------------------------------------
def tick():
  """time-controlled window update"""
  move(choice(('Up','Down','Right','Left'))) # pick a random direction
  win.after(500, tick)
# --------------------------------------------------------------------------------
def on_key(widget, code, mods):
  """callback function for each key press in the window"""
  if code not in ('Up','Down','Right','Left'): return # not a direction key
  move(code)
# --------------------------------------------------------------------------------
def main(rows=5, cols=5, size=64):
  """create the main window and pack the widgets"""
  global win
  win = Win(title='EVENT', flow='EN', fold=cols, key=on_key, grow=False)
  # ------------------------------------------------------------------------------
  for loop in range(rows*cols):
    Brick(win, height=size, width=size, bg=('#00F','#0F0','#F00'))
  # ------------------------------------------------------------------------------
  win.rows, win.cols = rows, cols
  # put cursor (= green cell) at the center of the grid
  win.cursor = win[rows//2][cols//2]; win.cursor.state = 1
  # put some walls (= red cells) near the corners of the grid
  walls = ((0,0),(1,0),(0,1),(-1,-1),(-2,-1),(-1,-2),(-1,0),(0,-1))
  for row,col in walls: win[row][col].state = 2
  # ------------------------------------------------------------------------------
  win.after(1000, tick); win.loop()
# ================================================================================
if __name__ == '__main__':
  main(7,9)
  #main(32,32,20)
# ================================================================================
```

C7A_canvas.py

```
# ================================================================================
"""CANVAS : demo program for the Canvas widget"""
# ================================================================================
__author__  = "Christophe Schlick"
__version__ = "1.0" # draw lines, rectangles, ovals, strings and images
__date__    = "2018-01-15"
# ================================================================================
from ezTK import *
from random import randrange as rr
# --------------------------------------------------------------------------------
def draw_rect():
  """draw a set of color rectangles"""
  steps, colors = 20, ('#F00','#0F0','#00F','#FF0','#000')
  w, h = win.width, win.height; dw, dh = w/steps/2, h/steps/2
  for n in range(steps):
    win.canvas.create_rectangle(n*dw, n*dh, w-n*dw, h-n*dh, width=3,
      outline=colors[4], fill=colors[n%4])
# --------------------------------------------------------------------------------
def draw_oval():
  """draw a set of color ovals"""
  steps, colors = 20, ('#F00','#0F0','#00F','#FF0','#000')
  w, h = win.width, win.height; dw, dh = w/steps/2, h/steps/2
  for n in range(steps):
    win.canvas.create_oval(n*dw, n*dh, w-n*dw, h-n*dh, width=3,
      outline=colors[4], fill=colors[n%4])
# --------------------------------------------------------------------------------
def draw_line():
  """draw a set of color lines"""
  steps, colors = 20, ('#F00','#0F0','#00F','#FF0')
  w, h = win.width, win.height; dw, dh = w/steps, h/steps
  for n in range(steps):
    win.canvas.create_line(n*dw, 0, w,   n*dh, width=2, fill=colors[0])
    win.canvas.create_line(n*dw, 0, 0, h-n*dh, width=2, fill=colors[1])
    win.canvas.create_line(n*dw, h, 0,   n*dh, width=2, fill=colors[2])
    win.canvas.create_line(n*dw, h, w, h-n*dh, width=2, fill=colors[3])
# --------------------------------------------------------------------------------
def draw_curve():
  """draw a set of curves by sampling mathematical functions"""
  from math import cos, exp
  w, h, colors = win.width//2, win.height//2, ('#000','#0F0','#F00','#00F')
  for n in range(4): # loop over curves
    xa, ya, xb, yb = 0, h, 0, h # initial position for each curve
    for x in range(w+1): # loop over horizontal axis
      t = 2*x/w - 1 # parameter t moves over range [-1,1]
      if n == 0: xa, ya, xb, yb = xb, yb, 2*x, h
      elif n == 1: xa, ya, xb, yb = xb, yb, 2*x, h - h*exp(-5*t*t)
      elif n == 2: xa, ya, xb, yb = xb, yb, 2*x, h + h*exp(-5*t*t)
      else: xa, ya, xb, yb = xb, yb, 2*x, h + h*exp(-5*t*t)*cos(25*t)
      win.canvas.create_line(xa, ya, xb, yb, width=2, fill=colors[n])
# --------------------------------------------------------------------------------
def draw_text():
  """draw a set of strings in a grid"""
  steps, colors, font = 12, ('#F00','#0F0','#00F','#000'), 'Arial 12 bold'
  w, h = win.width, win.height; dw, dh = w/steps, h/steps
  for row in range(steps):
    for col in range(steps):
      x, y, n = dw/2 + dw*col, dh/2 + dh*row, (col+1)*(row+1)
      win.canvas.create_text((x,y), text=n, font=font, fill=colors[(col+row)%3])
  # ------------------------------------------------------------------------------
  for n in range(1, steps):
    win.canvas.create_line(0, n*dh, w, n*dh, width=2, fill=colors[3])
    win.canvas.create_line(n*dw, 0, n*dw, h, width=2, fill=colors[3])
# --------------------------------------------------------------------------------
def draw_image():
  """draw a set of images at random position"""
  x, y, n = rr(win.width), rr(win.height), rr(len(win.images))
  win.canvas.create_image(x, y, image=win.images[n])
  win.after(20, draw_image) # recursive call of 'draw_image' after 20ms
# --------------------------------------------------------------------------------
def wait():
  """wait for user click, then clear canvas"""
```

```
  MessageDialog.showinfo('', message='Click to draw new shape')
  win.canvas.delete('all')
# ------------------------------------------------------------------------------
def main(width=480, height=480):
  """main program of the "canvas" module"""
  global win
  win = Win(title='CANVAS', grow=False)
  win.canvas = Canvas(win, width=width, height=height)
  win.images = [Image(file="smiley%s.png" % name) for name in '123456']
  win.width, win.height = width, height
  # ------------------------------------------------------------------------------
  #draw_rect(); wait(); draw_oval(); wait(); draw_line(); wait()
  draw_curve(); wait(); draw_text(); wait(); draw_image(); win.loop()
# ==============================================================================
if __name__ == '__main__':
  main(720,360)
# ==============================================================================
```

C7B_canvas.py

```
# ==============================================================================
"""CANVAS : demo program for the Canvas widget"""
# ==============================================================================
__author__  = "Christophe Schlick"
__version__ = "2.0" # create 3 moving circles
__date__    = "2018-01-15"
# ==============================================================================
from ezTK import *
# ------------------------------------------------------------------------------
def tick():
  """move all canvas items and make recursive function call after 10ms"""
  for item in win.items:
    xa, ya, xb, yb = win.canvas.coords(item[0]) # get item coordinates
    vx, vy = item[1:3] # get item velocity
    xa, ya = xa + vx, ya + vy # edit item coordinates by adding velocity
    xb, yb = xb + vx, yb + vy
    if xa < 0 or xb > win.width:  vx = -vx # horizontal bounce (reverse vx)
    if ya < 0 or yb > win.height: vy = -vy # vertical bounce (reverse vy)
    win.canvas.coords(item[0], xa, ya, xb, yb) # update position
    item[1:3] = vx, vy # update velocity
  win.canvas.after(10, tick) # recursive call of 'tick' after 10ms
# ------------------------------------------------------------------------------
def main(width=640, height=480):
  """main program of the "canvas" module"""
  global win
  win = Win(title='CANVAS', grow=False)
  win.canvas = Canvas(win, width=width, height=height)
  win.width, win.height, win.items = width, height, []
  # ------------------------------------------------------------------------------
  colors = ('#F00','#0F0','#00F','#000')
  for n in range(3):
    # define dimension, position and velocity for each item
    dim, dx, dy = min(width,height)/8, width/4, height/4
    x, y, vx, vy = dx + n*dx, dy + n*dy, 3-n, n+1
    win.items.append([win.canvas.create_oval(x-dim, y-dim, x+dim, y+dim,
      width=5, outline=colors[3], fill=colors[n]), vx, vy])
  # ------------------------------------------------------------------------------
  win.after(1000, tick) # start animation after 1000ms
  win.loop()
# ==============================================================================
if __name__ == '__main__':
  main()
# ==============================================================================
```

C7C_canvas.py

```
# ==============================================================================
"""CANVAS : demo program for the Canvas widget"""
# ==============================================================================
__author__  = "Christophe Schlick"
__version__ = "3.0" # create 6 moving sprites
__date__    = "2018-01-15"
# ==============================================================================
from ezTK import *
# ------------------------------------------------------------------------------
def tick():
  """move all canvas items and make recursive function call after 10ms"""
  for item in win.items:
    xa, ya, xb, yb = win.canvas.bbox(item[0]) # get item coordinates
    vx, vy = item[1:3] # get item velocity
    xa, ya = xa + vx, ya + vy # edit item coordinates by adding velocity
    xb, yb = xb + vx, yb + vy
    if xa < -6 or xb > win.width+6:  vx = -vx # horizontal bounce (reverse vx)
    if ya < -6 or yb > win.height+6: vy = -vy # vertical bounce (reverse vy)
    win.canvas.coords(item[0], (xa+xb)//2, (ya+yb)//2) # update position
    item[1:3] = vx, vy # update velocity
  win.canvas.after(10, tick) # recursive call of 'tick' after 10ms
# ------------------------------------------------------------------------------
def main(width=640, height=480):
  """main program of the "canvas" module"""
  global win
  win = Win(title='CANVAS', grow=False)
  win.canvas = Canvas(win, width=width, height=height, bg='black')
  win.width, win.height, win.items = width, height, []
  # ------------------------------------------------------------------------------
  images = [Image(file="smiley%s.png" % name) for name in '123456']
  for n in range(6):
    # define dimension, position and velocity for each item
    dx, dy = width/7, height/7
    x, y, vx, vy = dx + n*dx, dy + n*dy, 3-n, n-2
    win.items.append([win.canvas.create_image(x, y, image=images[n]), vx, vy])
  # ------------------------------------------------------------------------------
  win.after(1000, tick) # start animation after 1000ms
  win.loop()
# ==============================================================================
if __name__ == '__main__':
  main()
# ==============================================================================
```