C1A_frame.py

```
# ================================================================================
"""FRAME : demo program for the 'Win' and 'Frame' widgets"""
# ================================================================================
__author__  = "Christophe Schlick"
__version__ = "1.0" # one single 'Brick' widget
__date__    = "2021-03-15"
# ================================================================================
from ezTK import *
# --------------------------------------------------------------------------------
def main():
  """create the main window and pack the widgets"""
  win = Win(title='FRAME') # create main window
  Brick(win, width=400, height=200, bg='blue') # create Brick widget
  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 'Brick' widgets with 1D packing
__date__    = "2021-03-15"
# ================================================================================
from ezTK import *
# --------------------------------------------------------------------------------
def main():
  """create the main window and pack the widgets"""
  global win # always define 'win' as a global variable
  win = Win(title='FRAME') # use default 1D flow direction (= 'S')
  #win = Win(title='FRAME', flow='W') # change flow direction (= 'E','N' or 'W')
  #win = Win(title='FRAME', op=5) # add outer padding (in pixel units)
  # ------------------------------------------------------------------------------
  A, B, C = 'red', 'lime', 'blue'
  #A, B, C = '#FF0000', '#00FF00', '#0000FF'
  #A, B, C = '#F00', '#0F0', '#F0F'
  Brick(win, width=400, height=200, bg=A)
  Brick(win, width=400, height=200, bg=B)
  Brick(win, width=400, height=200, bg=C)
  # ------------------------------------------------------------------------------
  #properties()
  win.loop()
# --------------------------------------------------------------------------------
def properties():
  """view and edit some widget properties"""
  print(f"Number of widgets = {win.widgets}") # 1D flow -> int
  for n in (0,1,2): # loop over widgets and show their properties
    bg, width, height = win[n]['bg'], win[n]['width'], win[n]['height']
    print(f"* Properties for win[{n}] :")
    print(f"  bg={bg} width={width} height={height}")
  # ------------------------------------------------------------------------------
  #win[1]['bg'] = 'yellow' # edit widget property (use widget index)
# ================================================================================
if __name__ == "__main__":
  main()
# ================================================================================
```

C1C_frame.py

```
# ================================================================================
"""FRAME : demo program for the 'Win' and 'Frame' widgets"""
# ================================================================================
__author__  = "Christophe Schlick"
__version__ = "3.0" # 24 'Label' widgets with 2D packing
__date__    = "2021-03-15"
# ================================================================================
from ezTK import *
# --------------------------------------------------------------------------------
def main():
  """create the main window and pack the widgets"""
  global win
  # 2D packing is obtained by defining property 'fold' for window
  win = Win(title='FRAME', fold=6) # default 2D flow direction (= 'E' then 'S')
  #win = Win(title='FRAME', fold=3, flow='NE') # pack N then E, fold every 3
  #win = Win(title='FRAME', fold=12, flow='WN') # pack W then N, fold every 12
  # ------------------------------------------------------------------------------
  for loop in range(24): Label(win, text=loop, width=4, height=2, border=1)
  # ------------------------------------------------------------------------------
  #properties()
  win.loop()
# --------------------------------------------------------------------------------
def properties():
  """view and edit some widget properties"""
  print(f"Number of widgets = {win.widgets}") # 2D flow -> (int, int)
  rows, cols = win.widgets # number of rows, number of cols
  for row in range(rows): # loop over widgets and show 'text' properties
    for col in range(cols):
      widget = win[row][col] # get current widget
      text, bg, fg = widget['text'], widget['bg'], widget['fg']
      width, height = widget['width'], widget['height']
      print(f"* Properties for win[{row}][{col}] :")
      print(f"  text={text} bg={bg} fg={fg} width={width} height={height}")
  # ------------------------------------------------------------------------------
  #win[2][1]['bg'] = 'red' # edit widget properties (use widget coordinates)
  #win[1][4]['bg'] = 'blue'; win[1][4]['fg'] = 'white'
# ================================================================================
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__    = "2021-03-15"
# ================================================================================
from ezTK import *
# --------------------------------------------------------------------------------
def main():
  """create the main window and pack the widgets"""
  global win # always define 'win' as a global variable
  font1, font2 = 'Arial 14', 'Arial 32 bold' # define fonts used for widgets
  win = Win(title='FRAME', font=font1, op=2) # main window default flow='SE'
  # ------------------------------------------------------------------------------
  Button(win, text='OOOOOO', grow=False)
  # ------------------------------------------------------------------------------
  frame = Frame(win) # inner Frame with default flow='ES' (orthogonal flow)
  Button(frame, text='XXX\nXXX', grow=False)
  Button(frame, font=font2, text='YYY\nYYY') # use specific font for widget
  Button(frame, text='ZZZ\nZZZ', grow=False)
```

```
  # ------------------------------------------------------------------------
  Button(win, text='IIIIII', grow=False)
  # ------------------------------------------------------------------------
  #properties()
  win.loop()
# ----------------------------------------------------------------------------
def properties():
  """view and edit some widget properties"""
  print(f"Number of widgets for main window = {win.widgets}")
  print(f"Number of widgets for inner frame = {win[1].widgets}")
  text = win[0]['text']; print(f"Text for win[0] = {text!r}")
  text = win[2]['text']; print(f"Text for win[2] = {text!r}")
  text = win[1][0]['text']; print(f"Text for win[1][0] = {text!r}")
  text = win[1][2]['text']; print(f"Text for win[1][2] = {text!r}")
  # ------------------------------------------------------------------------
  win[1][1]['text'] = '\u2660\u2663\u2665\u2666' # edit widget property
# ============================================================================
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__    = "2021-03-15"
# ============================================================================
from ezTK import *
# ----------------------------------------------------------------------------
def main():
  """create the main window and pack the widgets"""
  global win # always define 'win' as a global variable
  win = Win(title='FRAME', font='Arial 14', op=2) # default flow='SE'
  # ------------------------------------------------------------------------
  fr1 = Frame(win, grow=False) # flow='ES' (orthogonal flow)
  Button(fr1, text='III', grow=False)
  Label(fr1, text='OOOOOO', border=2)
  Button(fr1, text='III', grow=False)
  # ------------------------------------------------------------------------
  fr2 = Frame(win) # flow='ES' (orthogonal flow)
  Label(fr2, text='A\nB\nC\nD\nE', grow=False)
  fr3 = Frame(fr2, border=4, op=5) # flow='SE' (orthogonal flow again)
  Label(fr3, text='ZZZ\nZZZ', border=2)
  fr4 = Frame(fr3, op=0, grow=False) # flow='ES' (orthogonal flow again)
  Button(fr4, text='XXX\nXXX')
  Button(fr4, text='YYY\nYYY')
  Label(fr2, text='A\nB\nC\nD\nE', grow=False)
  # ------------------------------------------------------------------------
  # Each widget can be accessed starting at any level of its parent hierarchy
  #win[0][2]['bg'], fr1[0]['bg'] = '#AFA','#FAA'
  #fr3[0]['bg'], fr3[1][0]['bg'], fr4[1]['bg'] = '#FFA','#FAA','#AFA'
  # ------------------------------------------------------------------------
  win.loop()
# ============================================================================
if __name__ == "__main__":
  main()
# ============================================================================
```

C1F_frame.py

```
# ============================================================================
"""FRAME : demo program for the 'Win' and 'Frame' widgets"""
# ============================================================================
__author__  = "Christophe Schlick"
__version__ = "6.0" # add callback function with 'command' option
__date__    = "2021-03-15"
# ============================================================================
from ezTK import *
# ----------------------------------------------------------------------------
def main():
  """create the main window and pack the widgets"""
  global win # always define 'win' as a global variable
  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, text='XXX\nXXX', grow=False)
  Button(fr1, text='IIIIII')
  # ------------------------------------------------------------------------
  fr2 = Frame(win)
  fr3 = Frame(fr2, grow=False)
  Button(fr3, text='OOOOOO', grow=False)
  Button(fr3, text='XXX\nXXX')
  Button(fr3, text='OOOOOO', grow=False)
  Button(fr2, text='EXIT', font=font2, fg='#FFF', bg='#00F', command=win.exit)
  Button(fr2, text='OOOOOO', grow=False)
  # ------------------------------------------------------------------------
  fr4 = Frame(win, grow=False)
  Button(fr4, text='XXX\nXXX', grow=False)
  Button(fr4, text='IIIIII')
  Button(fr4, text='XXX\nXXX', grow=False)
  # ------------------------------------------------------------------------
  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__    = "2021-03-15"
# ============================================================================
from ezTK import *
# ----------------------------------------------------------------------------
def main():
  """create the main window and pack the widgets"""
  global win # always define 'win' as a global variable
  win = Win(title='TOGGLE', border=2, fold=5) # fold every 5 widgets
  # ------------------------------------------------------------------------
  for loop in range(25) : # loop over grid cells (5x5)
    Brick(win, height=64, width=64, bg='#00F', border=2) # create a blue Brick
  # ------------------------------------------------------------------------
  win.after(2000, tick) # launch 'tick' to start animation after 2000 ms
  win.loop()
# ----------------------------------------------------------------------------
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(500, tick) # launch 'tick' again after 500 ms
```

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

C2B_toggle.py

```
# ============================================================================
"""TOGGLE : demo program for simple animation of multi-state widgets"""
# ============================================================================
__author__  = "Christophe Schlick"
__version__ = "2.0" # use multi-state Brick widgets as grid cells
__date__    = "2021-03-15"
# ============================================================================
from ezTK import *
from random import randrange as rr
# ----------------------------------------------------------------------------
def main(rows=8, cols=16):
  """create the main window and pack the widgets"""
  global win # always define 'win' as a global variable
  win = Win(title='TOGGLE', op=4, fold=cols) # fold every 'cols' widgets
  # --------------------------------------------------------------------------
  colors = ('#F00','#0F0','#00F','#0FF','#F0F','#FF0') # define color set
  for loop in range(rows*cols): # loop over grid cells (nb of cells = rows*cols)
    # create Brick with 6 different states : one for each background color
    Brick(win, height=64, width=64, border=2, bg=colors, state=2) # 2 <=> blue
  # --------------------------------------------------------------------------
  win.after(3000, tick); win.loop() # wait 3000 ms then start animation
# ----------------------------------------------------------------------------
def tick():
  """update function for widget animation"""
  rows, cols = win.widgets # get number of rows/cols for the grid of widgets
  states = win[0][0].states # get number of states for each grid cell
  row, col = rr(rows), rr(cols) # select random position
  win[row][col].state = rr(states) # set selected widget to random state
  win.after(20, tick) # launch 'tick' again after 20 ms
# ============================================================================
if __name__ == "__main__":
  main()
# ============================================================================
```

C2C_toggle.py

```
# ============================================================================
"""TOGGLE : demo program for simple animation of multi-state widgets"""
# ============================================================================
__author__  = "Christophe Schlick"
__version__ = "3.0" # use multi-state Label widgets for grid cells
__date__    = "2021-03-15"
# ============================================================================
from ezTK import *
from random import randrange as rr
# ----------------------------------------------------------------------------
def main(rows=8, cols=16):
  """create the main window and pack the widgets"""
  global win # always define 'win' as a global variable
  win = Win(title='TOGGLE', bg='#000', op=4, fold=cols)
  # --------------------------------------------------------------------------
  images = ImageGrid('balls.gif') # load grid of 6 color balls (R,G,B,C,M,Y)
  for loop in range(rows*cols): # loop over grid cells (nb of cells = rows*cols)
    # create Brick with 6 different states : one for each color ball
    Brick(win, image=images, state=loop) # initial state = color cycling
  # --------------------------------------------------------------------------
  win.after(2000, tick); win.loop() # wait 2000 ms then start animation
# ----------------------------------------------------------------------------
def tick():
  """update function for widget animation"""
  rows, cols = win.widgets # get number of rows/cols for the grid of widgets
  states = win[0][0].states # get number of states for each grid cell
  row, col = rr(rows), rr(cols) # select random position
  win[row][col].state = rr(states) # set selected widget to random state
  win.after(20, tick) # launch 'tick' again after 20 ms
# ============================================================================
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__    = "2021-03-15"
# ============================================================================
from ezTK import *
from random import randrange as rr
# ----------------------------------------------------------------------------
def main(rows=5, cols=7):
  """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','#FFF')
  fg = ('#FFF','#000') # number of states may be different for each property
  text = ('RED','GREEN','BLUE','CYAN','MAGENTA','YELLOW','BLACK','WHITE')
  win = Win(title='TOGGLE', font='Arial 16 bold', bg='#000', fold=cols, op=2)
  # --------------------------------------------------------------------------
  for loop in range(rows*cols): # loop over grid (number of cells = rows*cols)
    Label(win, text=text, height=3, width=9, bg=bg, fg=fg, state=loop)
  # --------------------------------------------------------------------------
  win.after(2000, tick); win.loop()
# ----------------------------------------------------------------------------
def tick():
  """update function for widget animation"""
  rows, cols = win.widgets # get number of rows/cols for the grid of widgets
  states = win[0][0].states # get number of states for each grid cell
  row, col = rr(rows), rr(cols) # select random position
  win[row][col].state = rr(states) # set random widget to random state
  win.after(20, tick) # launch 'tick' again after 20 ms
# ============================================================================
if __name__ == "__main__":
  main()
# ============================================================================
```

C3A_message.py

```
# ============================================================================
"""MESSAGE : demo program for simple callback functions"""
# ============================================================================
__author__  = "Christophe Schlick"
__version__ = "1.0" # use specific callback function for each button
__date__    = "2021-03-15"
# ============================================================================
from ezTK import *
# ----------------------------------------------------------------------------
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)
  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()
# ----------------------------------------------------------------------------
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'
# ============================================================================
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__    = "2021-03-15"
# ============================================================================
from ezTK import *
# ----------------------------------------------------------------------------
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)
  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()
# ----------------------------------------------------------------------------
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]
# ============================================================================
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__    = "2021-03-15"
# ============================================================================
from ezTK import *
# ----------------------------------------------------------------------------
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)
  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()
# ----------------------------------------------------------------------------
def on_button(state):
  """generic callback function for all three buttons"""
  win.label.state = state
# ============================================================================
if __name__ == '__main__':
  main()
# ============================================================================
```

C4A_random.py

```
# ============================================================================
"""RANDOM : generate random values within a user-provided range"""
# ============================================================================
__author__  = "Christophe Schlick"
__version__ = "1.0" # use Scale widgets to define range for random values
__date__    = "2021-03-15"
# ============================================================================
from ezTK import *
from random import randrange as rr
# ----------------------------------------------------------------------------
def main(minval=0, maxval=99):
  """create the main window and pack the widgets"""
  global win
  win = Win(title='RANDOM', op=5); scale = (minval, maxval)
  # ------------------------------------------------------------------------
  frame = Frame(win, op=0, grow=False)
  Label(frame, text='MIN :', anchor='SE', grow=False)
  Scale(frame, scale=scale, state=minval, command=on_scale)
  Label(frame, text='MAX :', anchor='SE', grow=False)
  Scale(frame, scale=scale, state=maxval, command=on_scale)
  Button(win, text='RANDOM', command=on_random, grow=False)
  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.min, win.max = minval, maxval; win.loop()
# ------------------------------------------------------------------------------
def on_scale():
  """callback function for both 'MIN' and 'MAX' scales"""
  minval, maxval = win.minscale.state, win.maxscale.state # get scale values
  if maxval < win.min: minval = win.minscale.state = maxval # copy max to min
  if minval > win.max: maxval = win.maxscale.state = minval # copy min to max
  win.min, win.max = minval, maxval # store new range for random generator
# ------------------------------------------------------------------------------
def on_random():
  """callback function for the 'RANDOM' button"""
  win.label['text'] = rr(win.min, win.max+1) # display new random value on label
# ==============================================================================
if __name__ == '__main__':
  main()
# ==============================================================================
```

C4B_random.py

```
# ==============================================================================
"""RANDOM : generate random values within a user-provided range"""
# ==============================================================================
__author__  = "Christophe Schlick"
__version__ = "2.0" # use Spinbox widgets to define range for random values
__date__    = "2021-03-15"
# ==============================================================================
from ezTK import *
from random import randrange as rr
# ------------------------------------------------------------------------------
def main(minval=0, maxval=99):
  """create the main window and pack the widgets"""
  global win
  win = Win(title='RANDOM', op=5)
  values = tuple(range(minval,maxval+1,10)) # define all values for spinboxes
  # ----------------------------------------------------------------------------
  frame = Frame(win, op=0, grow=False)
  Label(frame, text='MIN ', anchor='E', grow=False)
  Spinbox(frame, values=values, state=values[0], width=5, wrap=False)
  Label(frame, text='MAX ', anchor='E', grow=False)
  Spinbox(frame, values=values, state=values[-1], width=5, wrap=False)
  Button(win, text='RANDOM', command=on_random, grow=False)
  Label(win, font='Arial 72 bold', width=3, border=2)
  # ----------------------------------------------------------------------------
  # set friendly names for all widgets used in callbacks
  win.label, win.minspin, win.maxspin = win[2], frame[1], frame[3]
  win.min, win.max = minval, maxval; win.loop()
# ------------------------------------------------------------------------------
def on_random():
  """callback function for the 'RANDOM' button"""
  # spinbox values are strings, so the first step is to convert them to integers
  minval, maxval = int(win.minspin.state), int(win.maxspin.state)
  if maxval < win.min: minval = win.minspin.state = maxval # copy max to min
  if minval > win.max: maxval = win.maxspin.state = minval # copy min to max
  win.min, win.max = minval, maxval # store new range for random generator
  win.label['text'] = rr(win.min, win.max+1) # display new random value on label
# ==============================================================================
if __name__ == '__main__':
  main()
# ==============================================================================
```

C4C_random.py

```
# ==============================================================================
"""RANDOM : generate random vals within a user-provided range"""
# ==============================================================================
__author__  = "Christophe Schlick"
__version__ = "3.0" # use Entry widget to define range for random values
__date__    = "2021-03-15"
# ==============================================================================
from ezTK import *
from random import randrange as rr
# ------------------------------------------------------------------------------
def main(minval=0, maxval=99):
  """create the main window and pack the widgets"""
  global win
  win = Win(title='RANDOM', op=5)
  # ----------------------------------------------------------------------------
  frame = Frame(win, op=0, grow=False)
  Label(frame, text='Enter min,max : ', grow=False)
  Entry(frame, width=10, command=on_entry)
  Button(win, text='RANDOM', command=on_random, grow=False)
  Label(win, font='Arial 72 bold', width=3, border=2)
  # ----------------------------------------------------------------------------
  win.label, win.entry = win[2], frame[1] # friendly names
  win.min, win.max = minval, maxval; on_entry(); win.loop()
# ------------------------------------------------------------------------------
def on_entry():
  """callback function for the 'min,max' entry"""
  try: # try to parse the entry string as a couple of integer vals
    minval, maxval = win.entry.state.split(',') # get current min/max values
    minval, maxval = int(minval), int(maxval) # convert to integer
    win.min, win.max = min(minval,maxval), max(minval,maxval) # swap if needed
  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'] = rr(win.min, win.max+1) # display new random value on label
# ==============================================================================
if __name__ == '__main__':
  main()
# ==============================================================================
```

C4D_random.py

```
# ==============================================================================
"""RANDOM : generate random values within a user-provided range"""
# ==============================================================================
__author__  = "Christophe Schlick"
__version__ = "4.0" # use Menu widget to define range for random values
__date__    = "2021-03-15"
# ==============================================================================
from ezTK import *
from random import randrange as rr
# ------------------------------------------------------------------------------
def main(minval=0, maxval=99):
  """create the main window and pack the widgets"""
  global win
  win = Win(title='RANDOM', op=5)
  values = tuple(range(minval,maxval+1,10)) # define boundary values for range
  # ----------------------------------------------------------------------------
  win.master['menu'] = menu = Menu(win.master) # create master menu bar
  menu.add_command(label='EXIT', command=win.exit) # create command menu
  # ----------------------------------------------------------------------------
  minmenu = Menu(menu, tearoff=False) # create new menu to store min values
```

```
  menu.add_cascade(label='MIN', menu=minmenu) # add menu to menubar as cascade
  minmenu.state = StringVar(win, value=values[0]) # set initial min value
  for val in values: # insert all possible min values as radiobutton items
    minmenu.add_radiobutton(label=val, var=minmenu.state, command=on_menu)
  # ------------------------------------------------------------------------
  maxmenu = Menu(menu, tearoff=False) # create new menu to store max values
  menu.add_cascade(label='MAX', menu=maxmenu) # add menu to menubar as cascade
  maxmenu.state = StringVar(win, value=values[-1]) # set initial max value
  for val in values: # insert all possible max values as radiobutton items
    maxmenu.add_radiobutton(label=val, var=maxmenu.state, command=on_menu)
  # ------------------------------------------------------------------------
  Button(win, text='RANDOM', command=on_random, grow=False)
  win.label = Label(win, font='Arial 72 bold', width=3, border=2)
  # ------------------------------------------------------------------------
  win.minmenu, win.maxmenu = minmenu, maxmenu
  win.min, win.max = minval, maxval; win.loop()
# ----------------------------------------------------------------------------
def on_menu():
  """callback function for all menu radiobuttons"""
  # radiobutton items are strings, so first converted them to integers
  minval, maxval = int(win.minmenu.state.get()), int(win.maxmenu.state.get())
  if maxval < win.min: minval = maxval; win.minmenu.state.set(maxval)
  if minval > win.max: maxval = minval; win.maxmenu.state.set(minval)
  win.min, win.max = minval, maxval # store new range for random generator
# ----------------------------------------------------------------------------
def on_random():
  """callback function for the 'RANDOM' button"""
  win.label['text'] = rr(win.min, win.max+1) # display new random value on label
# ============================================================================
if __name__ == '__main__':
  main()
# ============================================================================
```

C4E_random.py

```
# ============================================================================
"""RANDOM : generate random values within a user-provided range"""
# ============================================================================
__author__  = "Christophe Schlick"
__version__ = "2.0" # store random values in a scrollable listbox
__date__    = "2021-03-15"
# ============================================================================
from ezTK import *
from random import randrange as rr
# ----------------------------------------------------------------------------
def main(minval=0, maxval=99):
  """create the main window and pack the widgets"""
  global win
  win = Win(title='RANDOM', op=5); scale = (minval, maxval)
  # ------------------------------------------------------------------------
  frame1 = Frame(win, op=0, grow=False)
  Label(frame1, text='Enter min,max : ', grow=False)
  Entry(frame1, width=10, command=on_entry)
  frame2 = Frame(win, op=0, grow=False)
  Button(frame2, text='RANDOM', command=on_random)
  Button(frame2, text='DELETE', command=on_delete)
  Listbox(win, width=30, height=15, scroll=True, grow=True)
  # ------------------------------------------------------------------------
  win.entry, win.box = frame1[1], win[2] # friendly names
  win.min, win.max = minval, maxval; on_entry(); win.loop()
# ----------------------------------------------------------------------------
def on_entry():
  """callback function for the 'min,max' entry"""
  try: # try to parse the entry string as a couple of integer vals
    minval, maxval = win.entry.state.split(',')
    minval, maxval = int(minval), int(maxval)
    win.min, win.max = min(minval, maxval), max(minval, maxval)
  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'
  values = [str(rr(win.min,win.max+1)) for loop in range(len(win.box)+1)]
  win.box.append(' '.join(values)) # append new values as a single line
  #win.box('\n'.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[:]  # delete all lines
# ============================================================================
if __name__ == '__main__':
  main()
# ============================================================================
```

C5A_win.py

```
# ============================================================================
"""WIN : demo program for window manipulations"""
# ============================================================================
__author__  = "Christophe Schlick"
__version__ = "1.0" # dynamic creation and destruction of windows
__date__    = "2021-03-15"
# ============================================================================
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 = f"MODAL of window {master.title}", '#F00'
  else: text, bg = f"SLAVE of window {master.title}", '#FF0'
  Label(win, text=text, bg=bg, border=2, height=2)
  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" # creation and destruction of static frames
__date__    = "2021-03-15"
# ============================================================================
from ezTK import *
# ----------------------------------------------------------------------------
def main():
  """create the main window and pack the widgets"""
  global win
  win = Win(title='FRAMES', op=2)
  top = Frame(win) # create top frame to store the 3 Buttons
  Button(top, text='BRICK',  width=8, command=on_brick)
  Button(top, text='GRID',   width=8, command=on_grid)
  Button(top, text='SCALES', width=8, command=on_scales)
  Button(top, text='RESET',  width=8, command=on_reset)
  win.loop()
# ----------------------------------------------------------------------------
def on_reset():
  """on_reset window configuration by deleting bottom frame"""
  if win.widgets > 1: del win[1] # clear bottom frame if it exists
# ----------------------------------------------------------------------------
def on_brick():
  """create the brick configuration on the bottom frame"""
  on_reset(); Brick(win, width=500, height=300, bg='blue') # single Brick
# ----------------------------------------------------------------------------
def on_grid():
  """create the grid configuration on the bottom frame"""
  on_reset(); frame = Frame(win, fold=10) # create new frame to store the grid
  colors = ('#F00','#0F0','#00F','#0FF','#F0F','#FF0')
  for loop in range(100): # create a 10x10 grid of Brick widgets
    Brick(frame, height=40, width=40, border=2, bg=colors, state=loop)
# ----------------------------------------------------------------------------
def on_scales():
  """create the scales configuration on the bottom frame"""
  on_reset(); frame = Frame(win, fold=2) # create new frame to store the scales
  for char in 'ABCDEF': # loop over chars to create specific Label/Scale pairs
    Label(frame, text=f"Value of {char} :", width=10, anchor='SE', grow=False)
    Scale(frame, scale=(0,99))
# ============================================================================
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__    = "2021-03-15"
# ============================================================================
from ezTK import *
# ----------------------------------------------------------------------------
def config(rows=8, cols=8):
  """create the config window and pack the widgets"""
  global win
  win = Win(title='CONFIG', fold=2, op=2, grow=False) # config window
  # --------------------------------------------------------------------------
  Label(win, text='Number of rows :', width=13, anchor='SW', grow=False)
  win.rowscale = Scale(win, scale=(1,12), length=200, flow='W', state=rows)
  Label(win, text='Number of cols :', width=13, anchor='SW', grow=False)
  win.colscale = Scale(win, scale=(1,12), length=200, flow='W', state=cols)
  Button(win, text='NEW GRID', command=grid)
  # --------------------------------------------------------------------------
  win.loop()
# ----------------------------------------------------------------------------
def grid():
  """create the grid window and pack the widgets"""
  global win
  rows, cols = win.rowscale.state, win.colscale.state # get grid size
  win.exit() # exit config window (only after having read state of scales)
  win = Win(title='GRID', bg='#000', fold=cols, op=2, grow=False) # grid window
  images = ImageGrid('balls.gif') # load grid of 6 color balls (R,G,B,C,M,Y)
  for loop in range(rows*cols): Brick(win, image=images, state=loop)
  # --------------------------------------------------------------------------
  win.loop(); config(rows, cols) # relaunch 'config' when grid window is closed
# ============================================================================
if __name__ == "__main__":
  config()
# ============================================================================
```

C5D_win.py

```
# ============================================================================
"""WIN : demo program for window manipulations"""
# ============================================================================
__author__  = "Christophe Schlick"
__version__ = "4.0" # demo for some standard dialog windows
__date__    = "2021-03-15"
# ============================================================================
from ezTK import *
# ----------------------------------------------------------------------------
def main():
  """create the main window and pack the widgets"""
  global win
  win = Win(title='DIALOG', fold=4, op=2, grow=False)
  # --------------------------------------------------------------------------
  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='CHOICE', width=12, command=on_choice)
  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='', anchor='W', border=1) # create status line
  win.loop()
# ----------------------------------------------------------------------------
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")
  val = Dialog(mode='info', message='Information message', title='INFO')
  message(f"Dialog return : {val}")
# ----------------------------------------------------------------------------
def on_warn():
  """callback for the "WARN" button"""
  message("WARN button has been pressed")
  val = Dialog(mode='warning', message='Warning message', title='WARNING')
  message(f"Dialog return : {val}")
# ----------------------------------------------------------------------------
def on_error():
```

```
  """callback for the "ERROR" button"""
  message("ERROR button has been pressed")
  val = Dialog(mode='error', message='Error message', title='ERROR')
  message(f"Dialog return : {val}")
# ------------------------------------------------------------------------------
def on_choice():
  """callback for the "CHOICE" button"""
  message("CHOICE button has been pressed")
  val = Dialog(mode='choice', message='Select YES or NO', title='CHOICE')
  message(f"Dialog return : {val}")
# ------------------------------------------------------------------------------
def on_color():
  """callback for the "COLOR" button"""
  message("COLOR button has been pressed")
  val = Dialog(mode='color', title='COLOR')
  message(f"Dialog return : RGB = {val[0]} Color = {val[1]}")
# ------------------------------------------------------------------------------
def on_open():
  """callback for the "OPEN FILE" button"""
  message("OPEN FILE button has been pressed")
  val = Dialog(mode='open', title='OPEN FILE')
  message(f"Dialog return : File = {val}")
# ------------------------------------------------------------------------------
def on_save():
  """callback for the "SAVE FILE" button"""
  message("SAVE FILE button has been pressed")
  val = Dialog(mode='save', title='SAVE FILE')
  message(f"Dialog return : File = {val}")
# ------------------------------------------------------------------------------
def on_popup():
  """callback for the "POPUP" button"""
  message("POPUP button has been pressed")
  popup = Win(win, title='POPUP', op=10) # create popup window
  text = 'This is a modal window\n\nPlease close it to continue'
  Label(popup, text=text); popup.wait() # wait for popup window to be closed
  message("POPUP window has been closed")
# ==============================================================================
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__    = "2021-03-15"
# ==============================================================================
from ezTK import *
# ------------------------------------------------------------------------------
def main():
  """create the main window and pack the widgets"""
  # create window and set global callback for 'move' event
  global win; win = Win(title='EVENT', op=5, move=on_move)
  Label(win, height=2, border=2, grow=False)
  Brick(win, width=1000, height=500, bg='#00F', border=2)
  # ----------------------------------------------------------------------------
  win.label, win.brick = win[0], win[1]; win.loop()
# ------------------------------------------------------------------------------
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"""
  text = f"Event = '{event}'  Code = {code}  Mods = {mods}"
  win.label['text'] = text # show event parameters on label widget
# ==============================================================================
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__    = "2021-03-15"
# ==============================================================================
from ezTK import *
# ------------------------------------------------------------------------------
def main(rows=4, cols=12):
  """create the main window and pack the widgets"""
  # create window and set global callbacks for 'click' and 'inout' events
  global win; win = Win(title='EVENT', click=on_click, inout=on_inout, op=5)
  Label(win, height=2, border=2, grow=False); grid = Frame(win, fold=cols)
  colors = ('#00F','#0F0','#F00','#0FF','#F0F','#FF0') # colors for grid cells
  for loop in range(rows*cols):
    Brick(grid, width=64, height=64, border=3, bg=colors)
  # ----------------------------------------------------------------------------
  win.label, win.grid = win[0], win[1]; win.loop()
# ------------------------------------------------------------------------------
def on_click(widget, code, mods):
  """callback function for all 'mouse click' events"""
  #print(widget, code, mods, widget.master, widget.index)
  if widget.master != win.grid or widget.index is None:
    return # nothing to do (mouse click is not on a grid cell)
  display('click', widget.index, code, mods)
  if   code == 'LMB': widget.state += 1 # increment state for left click
  elif code == 'RMB': widget.state -= 1 # decrement state for right click
  elif code == 'MMB': reset() # reset grid state for middle click
# ------------------------------------------------------------------------------
def on_inout(widget, code, mods):
  """callback function for all 'mouse in' or 'mouse out' events"""
  #print(widget, code, mods, widget.master, widget.index)
  if widget.master != win.grid or widget.index is None:
    return # nothing to do (mouse in/out is not on a grid cell)
  display('inout', widget.index, code, mods) # display event parameters
  if code == 1: widget['bg'] = '#FFF' # 'mouse in' event --> white background
  else: widget.state += 0 # 'mouse out' event --> restore background
# ------------------------------------------------------------------------------
def reset():
  """reset initial windows state"""
  rows, cols = win.grid.widgets # get size for grid of widgets
  for loop in range(rows*cols): # loop over grid cells
    row, col = loop // cols, loop % cols # get coords by Euclidian division
    win.grid[row][col].state = 0 # reset state for each cell
# ------------------------------------------------------------------------------
def display(event, index, code, mods):
  """display event parameters"""
  text = f"Event = '{event}'  Index = {index}  Code = {code}  Mods = {mods}"
  win.label['text'] = text # show event parameters on label widget
# ==============================================================================
```

```
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__    = "2021-03-15"
# ============================================================================
from ezTK import *
# ----------------------------------------------------------------------------
def main():
  """create the main window and pack the widgets"""
  # create window and set global callback for 'key' event
  global win; win = Win(title='EVENT', op=3, key=on_key)
  Label(win, font='Arial 14', height=2, width=50, border=2, grow=False)
  Label(win, font='Arial 48 bold', bg='#00F', fg='#FFF', border=2)
  # --------------------------------------------------------------------------
  win.label, win.char = win[0], win[1]; win.loop()
# ----------------------------------------------------------------------------
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"""
  text = f"Event = '{event}'  Code = '{code}'  Mods = {mods}"
  win.label['text'] = text # show event parameters on label widget
# ============================================================================
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__    = "2021-03-15"
# ============================================================================
from ezTK import *
# ----------------------------------------------------------------------------
def main(rows=9, cols=9, size=64):
  """create the main window and pack the widgets"""
  global win; win = Win(title='EVENT', fold=cols, key=on_key, grow=False)
  colors = ('#00F','#0F0','#F00') # define color set for board cells
  # --------------------------------------------------------------------------
  for loop in range(rows*cols): # create all grid cells
    Brick(win, height=size, width=size, border=2, bg=colors)
  # --------------------------------------------------------------------------
  # 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()
# ----------------------------------------------------------------------------
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
  rows, cols = win.widgets # get total number of rows and cols
  row, col = win.cursor.index # get current cursor position
  drow, dcol = moves[code] # get displacement vector
  # compute new position for cursor by using modulo to get automatic cycling
  row, col = (row + drow) % rows, (col + dcol) % 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
# ============================================================================
if __name__ == '__main__':
  main() # create window with default parameters
  #main(32,32,24) # try alternative set of parameters
# ============================================================================
```

C7A_canvas.py

```
# ============================================================================
"""CANVAS : demo program for the Canvas widget"""
# ============================================================================
__author__  = "Christophe Schlick"
__version__ = "1.0" # draw lines, rectangles, ovals, strings and images
__date__    = "2021-03-15"
# ============================================================================
from ezTK import *
from random import randrange as rr
# ----------------------------------------------------------------------------
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) # create canvas widget
  win.images = ImageGrid('smileys.png') # load grid of 6 smiley images (RGBA)
  win.width, win.height = width, height # store canvas size
  # --------------------------------------------------------------------------
  # show several patterns on canvas and wait for user validation in between
  draw_rect(); wait(); draw_oval(); wait(); draw_line(); wait()
  draw_curve(); wait(); draw_text(); wait(); draw_image(); win.loop()
# ----------------------------------------------------------------------------
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): # draw one rectangle for each loop step
    xa, ya, xb, yb, bg, fg = n*dw, n*dh, w-n*dw, h-n*dh, colors[n%4], colors[4]
    win.canvas.create_rectangle(xa, ya, xb, yb, fill=bg, outline=fg, width=3)
# ----------------------------------------------------------------------------
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): # draw one rectangle for each loop step
    xa, ya, xb, yb, bg, fg = n*dw, n*dh, w-n*dw, h-n*dh, colors[n%4], colors[4]
    win.canvas.create_oval(xa, ya, xb, yb, fill=bg, outline=fg, width=3)
# ----------------------------------------------------------------------------
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): # draw four lines for each loop step
    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) # call 'draw_image' again after 20ms
# ---------------------------------------------------------------------------
def wait():
  """wait for user click, then clear canvas"""
  Dialog(mode='info', message='Click to draw next shape')
  win.canvas.delete('all')
# ===========================================================================
if __name__ == '__main__':
  main()
# ===========================================================================
```

C7B_canvas.py

```
# ===========================================================================
"""CANVAS : demo program for the Canvas widget"""
# ===========================================================================
__author__  = "Christophe Schlick"
__version__ = "2.0" # create 3 color circles as moving sprites
__date__    = "2021-03-15"
# ===========================================================================
from ezTK import *
# ---------------------------------------------------------------------------
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.sprites = width, height, []
  # -------------------------------------------------------------------------
  colors = ('#F00','#0F0','#00F','#000')
  dx, dy = width/4, height/4 # set initial distance between neighboring sprites
  w = h = min(width, height)/8 # set dimension for all sprites
  for n in range(3): # create 3 sprites (= color disks) on canvas
    x, y, vx, vy = dx+n*dx, dy+n*dy, 3-n, n+1 # initial position and velocity
    item = win.canvas.create_oval(x-w, y-h, x+w, y+h, width=5,
      outline=colors[3], fill=colors[n]) # add item (= color circle) to canvas
    win.sprites.append([item, x, y, w, h, vx, vy]) # store sprite parameters
  # -------------------------------------------------------------------------
  win.after(1000, tick) # start animation after 1000ms
  win.loop()
# ---------------------------------------------------------------------------
def tick():
  """move all canvas items and make recursive function call after 10ms"""
  for sprite in win.sprites: # loop over sprites
    item, x, y, w, h, vx, vy = sprite # get sprite parameters
    x, y = x+vx, y+vy # compute new position for sprite (add current velocity)
    if x-w < 0 or x+w > win.width:  vx = -vx # horizontal bounce (reverse vx)
    if y-h < 0 or y+h > win.height: vy = -vy # vertical bounce (reverse vy)
    win.canvas.coords(item, x-w, y-h, x+w, y+h) # update item coords on canvas
    sprite[1:] = x, y, w, h, vx, vy # update sprite parameters
  win.canvas.after(10, tick) # call 'tick' again after 10ms
# ===========================================================================
if __name__ == '__main__':
  main()
# ===========================================================================
```

C7C_canvas.py

```
# ===========================================================================
"""CANVAS : demo program for the Canvas widget"""
# ===========================================================================
__author__  = "Christophe Schlick"
__version__ = "3.0" # create 6 images as moving sprites
__date__    = "2021-03-15"
# ===========================================================================
from ezTK import *
# ---------------------------------------------------------------------------
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='#000')
  win.width, win.height, win.sprites = width, height, [] # store attributes
  win.images = ImageGrid('smileys.png') # extract images from image grid
  # -------------------------------------------------------------------------
  dx, dy = width/7, height/7 # initial distance between neighboring sprites
  for n in range(6): # create 6 sprites (= images) on canvas
    # define dimension, initial position and initial velocity for each sprite
    x, y, vx, vy = dx+n*dx, dy+n*dy, 3-n, n-2 # initial position and velocity
    image = win.images[n]; w, h = image.width()//2, image.height()//2
    item = win.canvas.create_image(x, y, image=image) # add sprite to canvas
    win.sprites.append([item, x, y, w, h, vx, vy]) # store sprite parameters
  # -------------------------------------------------------------------------
  win.after(1000, tick) # start animation after 1000ms
  win.loop()
# ---------------------------------------------------------------------------
def tick():
  """move all canvas items and make recursive function call after 10ms"""
  for sprite in win.sprites:
    item, x, y, w, h, vx, vy = sprite # get sprite parameters
    x, y = x+vx, y+vy # compute new position for sprite (add current velocity)
    if x-w+6 < 0 or x+w-6 > win.width: vx = -vx # horizontal bounce (reverse vx)
    if y-h+6 < 0 or y+h-6 > win.height: vy = -vy # vertical bounce (reverse vy)
```

```
    win.canvas.coords(item, x, y) # update item coordinates on canvas
    sprite[1:] = x, y, w, h, vx, vy # update sprite parameters
  win.canvas.after(10, tick) # recursive call of 'tick' after 10ms
# ============================================================================
if __name__ == '__main__':
  main()
# ============================================================================
```

C7D_canvas.py

```
# ============================================================================
"""CANVAS : demo program for the Canvas widget"""
# ============================================================================
__author__  = "Christophe Schlick"
__version__ = "4.0" # animate scrolling sprites on Canvas
__date__    = "2021-03-15"
# ============================================================================
from ezTK import *
from random import randrange as rr
# ----------------------------------------------------------------------------
def main(width=900, height=500):
  """main program of the "canvas" module"""
  global win
  win = Win(title='CANVAS', grow=False)
  win.canvas = Canvas(win, width=width, height=height, bg='#000')
  win.width, win.height, win.sprites = width, height, [] # store attributes
  win.images = ImageGrid('smileys.png') # load grid of 6 smiley images (RGBA)
  win.step, win.counter = 0, 0 # initialize step counter and sprite counter
  win.after(1000, tick); win.loop() # wait 1000ms and launch animation
# ----------------------------------------------------------------------------
def tick():
  """move all canvas items and make recursive function call after 10ms"""
  if win.counter == 200: return win.exit() # close window after 200 sprites
  if win.step == 0: # create new sprite when step counter drops to zero
    win.step = max(5, 50-win.counter); win.counter += 1 # update both counters
    # select random image, position and velocity for new sprite
    image = win.images[rr(6)]; w, h = image.width()//2, image.height()//2
    x, y, vx, vy = rr(win.width), -h, 0.1*rr(10), 1.0
    if 2*x > win.width: vx = -vx # reverse vx when sprite starts on the right
    item = win.canvas.create_image(x, y, image=image) # add image to canvas
    win.sprites.append([item, x, y, w, h, vx, vy]) # store sprite parameters
  item, x, y, w, h, vx, vy = win.sprites[0] # get parameters for eldest sprite
  if y-h > win.height: # delete eldest sprite when it goes out of screen
    win.canvas.delete(item); win.sprites.pop(0)
  win.step -= 1; speed = 1+win.counter/5 # set speed according to sprite counter
  for sprite in win.sprites: # loop over remaining sprites
    item, x, y, w, h, vx, vy = sprite # get parameters for current sprite
    x, y = x + vx*speed, y + vy*speed # compute new position for sprite
    win.canvas.coords(item, x, y) # update item coordinates on canvas
    sprite[1:3] = x, y # update sprite parameters
  win.canvas.after(10, tick) # call 'tick' again after 10ms
# ============================================================================
if __name__ == '__main__':
  main()
# ============================================================================
```

C7E_canvas.py

```
# ============================================================================
"""CANVAS : demo program for the Canvas widget"""
# ============================================================================
__author__  = "Christophe Schlick"
__version__ = "5.0" # move ball with mouse and destroy canvas items
__date__    = "2021-03-15"
# ============================================================================
from ezTK import *
from random import randrange as rr
# ----------------------------------------------------------------------------
def main(dim=800):
  """create the main window and pack the widgets"""
  global win
  win = Win(title='CANVAS', move=on_move, op=2, grow=False)
  win.images = ImageGrid('balls.png'); win.image = Image('ball.png')
  win.canvas = Canvas(win, width=dim, height=dim, bg='#000', cursor='none')
  # create player ball with its score item (initial position = out of canvas)
  win.ball = win.canvas.create_image(-dim, 0, image=win.image)
  win.text = win.canvas.create_text(-dim, 0, text=0, font='Arial 16')
  # ----------------------------------------------------------------------------
  win.dim, win.score, win.balls = dim, 0, {}; win.after(1000, tick); win.loop()
# ----------------------------------------------------------------------------
def tick():
  """generate a new random ball on canvas and recurse after 1000ms"""
  x, y, n = rr(32,win.dim-32), rr(32,win.dim-32), rr(6) # random pos and color
  ball = win.canvas.create_image(x, y, image=win.images[n]) # create new ball
  win.balls[ball] = n; win.after(500, tick) # store ball color in dictionary
# ----------------------------------------------------------------------------
def on_move(widget, code, mods):
  """callback function for all 'mouse move' events"""
  if widget != win.canvas: return # mouse is not on canvas
  x, y = code; widget.coords(win.ball, x, y); widget.coords(win.text, x, y)
  # check if the player ball is currently overlapping any color ball on canvas
  balls = widget.find_overlapping(x-8,y-8,x+8,y+8)[2:]
  if not balls: return # no overlapping color ball has been found
  ball = balls[-1]; widget.delete(ball) # get color ball and delete it on canvas
  # update score on player ball, depending on the color of the deleted ball
  win.score += win.balls[ball]; widget.itemconfig(win.text, text=win.score)
# ============================================================================
if __name__ == '__main__':
  main()
# ============================================================================
```

C7F_canvas.py

```
# ============================================================================
"""CANVAS : demo program for the Canvas widget"""
# ============================================================================
__author__  = "Christophe Schlick"
__version__ = "6.0" # draw lines and disks according to mouse position
__date__    = "2021-03-15"
# ============================================================================
from ezTK import *
from random import choice
# ----------------------------------------------------------------------------
def main(dim=400):
  """create the main window and pack the widgets"""
  global win
  win = Win(title='CANVAS', click=on_click, op=2, grow=False)
  win.canvas = Canvas(win, width=2*dim, height=2*dim)
  win.canvas.create_oval(dim-20, dim-20, dim+20, dim+20, fill='#000')
  win.x = win.y = dim; win.loop() # set initial position and start event loop
# ----------------------------------------------------------------------------
def on_click(widget, code, mods):
  """callback function for all 'mouse click' events"""
  if widget != win.canvas: return # mouse click is not on canvas
  # get current mouse position, relative to top left corner of canvas
  x = widget.winfo_pointerx() - widget.winfo_rootx()
  y = widget.winfo_pointery() - widget.winfo_rooty()
  widths, colors = (1,3,5,7,9), ('#F00','#0F0','#00F','#0FF','#F0F','#FF0')
  # create line between previous and current mouse position (use random width)
```

```
  line = win.canvas.create_line(win.x, win.y, x, y, width=choice(widths))
  widget.tag_lower(line) # put new line UNDER all existing items
  # create oval centered at current mouse position (use random fill color)
  oval = win.canvas.create_oval(x-20, y-20, x+20, y+20, fill=choice(colors))
  widget.tag_raise(oval) # put new oval OVER all existing items
  win.x, win.y = x, y # store current coordinates for next click
# ============================================================================
if __name__ == '__main__':
  main()
# ============================================================================
```