Несколько примеров

Игра «жизнь»

(по https://habrahabr.ru/company/mailru/blog/228379/)

```
from tkinter import Tk,
                  Canvas, Button, Frame, BOTH, NORMAL, HIDDEN
# функция получает координаты мыши
# в момент нажатия или передвижения с зажатой кнопкой
def draw_a(e):
     ii = (e.y-3) / cell_size
     jj = (e.x-3) / cell_size
     ii=int(ii);jj=int(jj)
     print("e=",e, ' ii=',ii, '  jj=',jj)
     canvas.itemconfig(cell_matrix[addr(ii, jj)], state=NORMAL,
tags='vis')

# эта функция преобразует двумерную
# координату в простой адрес одномерного массива
def addr(ii,jj):
     if(ii < 0 or jj < 0 or ii >= field_height or jj >=
field_width):
          return int( len(cell_matrix)-1 )
     else:
          return int( ii * (win_width / cell_size) + jj )


# функция обновления «картинки»
def refresh():
     for i in range(field_height):
          for j in range(field_width):
               k = 0
              # считаем число соседей
               for i_shift in range(-1, 2):
                    for j_shift in range(-1, 2):
                         if (canvas.gettags(
                                cell_matrix[addr(i + i_shift,
                                j + j_shift)])[0] == 'vis' and
                                 (i_shift != 0 or j_shift != 0)):
                              k += 1
               current_tag =
                    canvas.gettags(cell_matrix[addr(i, j)])[0]
               # в зависимости от их числа устанавливаем
               # состояние клетки
               # здесь можно экспериментировать
               # с самими "правилами" игры
```

```
                if(k == 3):
                     canvas.itemconfig(cell_matrix[addr(i, j)],
                                       tags=(current_tag, 'to_vis'))
                if(k <= 1 or k >= 4):
                     canvas.itemconfig(cell_matrix[addr(i, j)],
                                        tags=(current_tag, 'to_hid'))
                if(k == 2 and
                             canvas.gettags(cell_matrix[
                                     addr(i, j)])[0] == 'vis'):
                    canvas.itemconfig(cell_matrix[addr(i, j)],
                                        tags=(current_tag, 'to_vis'))

# сам шаг: обновляем состояние и рисуем
def step():
     refresh()
     repaint()

def clear():
     for i in range(field_height):
          for j in range(field_width):
               canvas.itemconfig(cell_matrix[addr(i, j)],
                                 state=HIDDEN, tags=('hid','0'))

# перерисовываем поле по буферу
# согласно второго тега элемента
def repaint():
     for i in range(field_height):
          for j in range(field_width):
               if (canvas.gettags(cell_matrix[addr(i, j)])[1] ==
                                                    'to_hid'):
                    canvas.itemconfig(cell_matrix[addr(i, j)],
                                      state=HIDDEN, tags=('hid','0'))
               if (canvas.gettags(cell_matrix[addr(i, j)])[1] ==
                                                    'to_vis'):
                    canvas.itemconfig(cell_matrix[addr(i, j)],
                                      state=NORMAL, tags=('vis','0'))

# «создание и поддержка выполнения»
#                                  автоматического режима
def loop():
    ##label['text'] = str(n)
    step()
    if (flag_run):
            root.after(500, loop) # call loop() in 0.5 seconds

# «выполнение» автоматического режима
def run():
        global flag_run, btn3
        #print('flag_run=',flag_run)
        if not flag_run:
               btn3.config(text="STOP")
               flag_run=True
```

```
                #print('stop')
                loop()
        else:
                btn3.config(text="RUN")
                flag_run=False
                #print('run')

flag_run=False # работает «автоматически» или по шагам

# создаем само окно
root = Tk()
root.title('4 cuba')
# это ширина и высота окна
win_width = 360 #350
win_height = 380 #370
# «строим» конфигурационную строку – размер окна
config_string = "{0}x{1}".format(win_width, win_height + 32)
#  Задаем цвет клетки
fill_color = "green"
# методом geometry() задаем размеры, можно написать и
# просто строчку вида '360x380'
root.geometry(config_string)
# это ширина самой клетки, с учетом «просвета»
cell_size = 20
# создается объект типа Canvas, на котором будет
# происходить непосредственно рисование,
# он делается дочерним по отношению к самому окну root
canvas = Canvas(root, height=win_height)
# пакуем его, аналог show() в других системах
canvas.pack(fill=BOTH)
# определяем размеры поля в клетках
field_height = int( win_height / cell_size )
field_width = int( win_width / cell_size )
# создаем массив для клеток, он одномерный,
# но используя вспомогательную фунукцию addr –
# будем преобразовавать индексы 2-х мерный к 1-мерному
cell_matrix = []
# здесь в цикле создаем экземпляры клеток
#  и делаем их скрытыми
for i in range(field_height):
     for j in range(field_width):
                # здесь создаем экземпляр клетки
                # и делаем его скрытыми
          square = canvas.create_rectangle(2 + cell_size*j,
                                           2 + cell_size*i,
                                cell_size + cell_size*j - 2,
                                cell_size + cell_size*i - 2,
                                            fill=fill_color)
          canvas.itemconfig(square, state=HIDDEN,
                                               tags=('hid','0'))
```

```
            # «добавляем» в массив
            cell_matrix.append(square)
#  создадим  фиктивный элемент,
#  он как бы «повсюду» вне поля
fict_square =
                canvas.create_rectangle(0,0,0,0, state=HIDDEN,
                                                tags=('hid','0'))
cell_matrix.append(fict_square)
# задаем «клетки» которые находятся на поле
                                    #       «по умолчанию»
canvas.itemconfig(cell_matrix[addr(8, 8)], state=NORMAL,
                                                tags='vis')
canvas.itemconfig(cell_matrix[addr(10, 9)], state=NORMAL,
                                                tags='vis')
canvas.itemconfig(cell_matrix[addr(9, 9)], state=NORMAL,
                                                tags='vis')
canvas.itemconfig(cell_matrix[addr(9, 8)], state=NORMAL,
                                                tags='vis')
canvas.itemconfig(cell_matrix[addr(9, 7)], state=NORMAL,
                                                tags='vis')
canvas.itemconfig(cell_matrix[addr(10, 7)], state=NORMAL,
                                                tags='vis')

# создаем фрейм для хранения кнопок и
# аналогичным образом, как с Canvas,
# устанавливаем кнопки дочерные фрейму

frame = Frame(root)
# выполнить «шаг»
btn1 = Button(frame, text='Step', command = step)  # Eval
# очистить поле
btn2 = Button(frame, text='Clear', command = clear)
# смена режимов «автоматически»  -  «по шагам»
btn3 = Button(frame, text='Run', command = run)
# пакуем кнопки

btn1.pack(side='left')
btn2.pack(side='right')
btn3.pack(side='right')
# пакуем фрейм
frame.pack(side='bottom')

# привязываем события клика и
# движения мыши над canvas к функции draw_a
canvas.bind('<B1-Motion>', draw_a)
canvas.bind('<ButtonPress>', draw_a)
# через 500 мкс передать управление функции loop
root.after(500, loop)
# стандартный цикл, организующий события
# и общую работу оконного приложения
```

```
root.mainloop()
```

Разное

```
import sys
from math import *
from tkinter import *
from tkinter.filedialog import *
from tkinter.messagebox import *
#### http://younglinux.info/tkinter/canvas.php

def f1():
    print('f1 - run.........')
def b1(event):
     root.title("Левая кнопка мыши")
def b3(event):
     root.title("Правая кнопка мыши")
def move(event):
     root.title("Движение мышью")
def res(event):
    root.title("изменение размера окна")
    print("изменение размера окна")
    xm, ym = xymax(root.geometry())
    print("--------xm---ym----------> ", xm, "  ", ym)
    print("---------------------> ",canv. .geometry() )
    canv.config(width = xm, height = ym)

def xymax(str):
    """ получим новый размер окна """
    print("===> ", str )
    k1=str.index('x')
    xmax1=str[:k1]
    k2=str.index('+',k1+1)
    ymax1=str[k1+1:k2]
    xmax=int(xmax1)
    ymax=int(ymax1)
    print(xmax,'  ',ymax)
    return  xmax,ymax

def button_clicked():
    sss=root.geometry()
    print ("Клик!")
    print('root.maxsize()=<', root.maxsize(),'>'  )
    print('canv.height()=<', canv.height(),'>'  )
    print('canv.size()=<', canv.size(),'>'  )
    print(root.geometry() )
    print('---canv.winfo_height()=',canv.winfo_height() )
     print('---canv.winfo_width() =',canv.winfo_width()  )
     return
def window_deleted():
    print ('Окно закрыто')
    #canv.delete('all')
```

```
    if askyesno("Завершение программы",
            "Действительно Вы хотите завершить программу?"):
            #self.save_file()
            root.destroy()
    else:
            pass
   #root.destroy()
   # root.quit() # явное указание на выход из программы
   # exit(0)
f = input('f(x):')

root = Tk()

f_1=Frame(root,bg="Yellow",)
f_2=Frame(root,bg="Blue")
f_1.pack(side=BOTTOM)
f_2.pack(side=BOTTOM)
button3 = Button(f_1, text=u"__B3 привет", command=getF)
button3.pack(side=LEFT)

button4 = Button(f_2, text="_Б4 привет", command=getF)
button4.pack(side=RIGHT) #LEFT)

canv = Canvas(root, width = 500, height = 500, bg = "white")

canv.create_line(500,1000,500,0,width=2,arrow=LAST)
canv.create_line(0,500,1000,500,width=2,arrow=LAST)

canv.create_line(200,50,300,50,width=3,fill="blue")
canv.create_line(0,0,100,100,width=2,arrow=LAST)

print('root.resizable()=<',root.resizable(),'>'  )
print('root.root.maxsize()=<', root.maxsize(),'>'  )

# кнопка по умолчанию
button1 = Button(root, text=u"file read!", command=getF)
button1.pack(side=TOP) #LEFT) #'top') #place

# кнопка с указанием родительского виджета и несколькими
                                                    аргументами
button2 = Button(root, bg="red", text=u"Кликни меня!",
command=button_clicked)
button2.pack(side=TOP) #'top') #top')


root.protocol('WM_DELETE_WINDOW', window_deleted) # обработчик
                                                  закрытия окна
#sys.exit(0)

First_x = -500;

for i in range(16000):
     if (i % 800 == 0):
```

```
          k = First_x + (1 / 16) * i
          canv.create_line(k + 500, -3 + 500, k + 500, 3 + 500,
width = 0.5, fill = 'black')
          canv.create_text(k + 515, -10 + 500, text = str(k),
                           fill="purple", font=("Helvectica", "10"))
          if (k != 0):
               canv.create_line(-3 + 500, k + 500, 3 + 500,
                                k + 500, width = 0.5, fill = 'black')
               canv.create_text(20 + 500, k + 500, text = str(k),
                                fill="purple", font=("Helvectica", "10"))
     try:
          x = First_x + (1 / 16) * i
          new_f = f.replace('x', str(x))
          y = -eval(new_f) + 500
          x += 500
          # canv.create_oval(x, y, x + 1, y + 1, fill = 'black')
          canv.create_line(x, y, x + 1, y + 1, fill = 'black')
     except:
          pass
canv.pack()
################
root.bind('<Button-1>',b1)
root.bind('<Button-3>',b3)
root.bind('<Motion>',move)
root.bind('<Configure>',res)
#############
canv.focus_set()
root.mainloop()
print('@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@')
```