Листинг кода:

```
from tkinter import *
import tkinter as tk
from PIL import Image, ImageDraw, ImageTk, ImageGrab
import PIL
from tkinter import colorchooser
import tkinter.ttk as ttk
from tkinter import filedialog
from tkinter import messagebox
from tkinter.simpledialog import askstring, askinteger
import time
from tkinter.messagebox import showinfo

root = Tk()
root.title('Paint')
root.geometry('800x800')
menubar = Menu(root)

w = 600
h = 400
x = w / 2
y = h / 2
num = 0
white = (255, 255, 255)
image1 = PIL.Image.new('RGB', (w, h), white)
draw = ImageDraw.Draw(image1)

my_canvas = Canvas(root, width=w, height=h, bg='#ffffff')
my_canvas.pack(pady=20)

brush_color = 'black'
bg_color = 'white'


def paint(e):

    brush_width = '%0.0f' % float(my_slider.get())

    brush_type2 = brush_type.get()

    x1 = e.x - 1
    y1 = e.y - 1

    x2 = e.x + 1
    y2 = e.y + 1

    my_canvas.create_line(x1, y1, x2, y2, fill=brush_color,
width=brush_width, capstyle=brush_type2, smooth=True)
    old_x = e.x
    old_y = e.y
def paint_bg(e):
```

```
    brush_width = '%0.0f' % float(my_slider.get())

    brush_type2 = brush_type.get()

    x1 = e.x - 1
    y1 = e.y - 1

    x2 = e.x + 1
    y2 = e.y + 1

    my_canvas.create_line(x1, y1, x2, y2, fill=bg_color,
width=brush_width, capstyle=brush_type2, smooth=True)
    old_x = e.x
    old_y = e.y

def reset(event):
    old_x = None
    old_y = None


def save_as_png():
    result =
filedialog.asksaveasfilename(defaultextension='.jpg',filetypes=(
('png files', '*.png'),('jpeg files', '*.jpeg'), ('all files',
'*.*')))
    if result.endswith('.png'):
        pass
    if result.endswith('.jpeg'):
        pass
    else:
        result = result + '.png'

    if result:
        x = root.winfo_rootx() + my_canvas.winfo_x()
        y = root.winfo_rooty() + my_canvas.winfo_y()
        x1 = x + my_canvas.winfo_width()
        y1 = y + my_canvas.winfo_height()
        ImageGrab.grab().crop((x, y, x1, y1)).save(result)

        messagebox.showinfo('Image Saved', 'Your Image Has Been
Saved!')


def save():
    x = root.winfo_rootx() + my_canvas.winfo_x()
    y = root.winfo_rooty() + my_canvas.winfo_y()
    x1 = x + my_canvas.winfo_width()
    y1 = y + my_canvas.winfo_height()

    converted_num = str(num)
    filename = converted_num + ".png"
    ImageGrab.grab().crop((x, y, x1, y1)).save(filename)
```

```
    messagebox.showinfo('Image Saved', 'Your Image Has Been
Saved!')


my_canvas.bind('<B1-Motion>', paint)
my_canvas.bind('<B3-Motion>', paint_bg)


def clear_screen():
    my_canvas.delete(ALL)
    my_canvas.config(bg='white')


def change_brush_color():
    global brush_color
    brush_color = 'black'
    brush_color = colorchooser.askcolor(color=brush_color)[1]




def change_canvas_color():
    global bg_color
    bg_color = 'white'
    bg_color = colorchooser.askcolor(color=bg_color)[1]
    my_canvas.config(bg=bg_color)


def change_brush_size(command_thing):
    slider_label.config(text='%0.0f' % float(my_slider.get()))


def shape():
    name = askstring('Shapes', 'What shape do you want to
draw?')
    if (name == "oval"):

        x = True
        topx = askstring('Shapes', 'Enter top right x point')
        topy = askstring('Shapes', 'Enter top right y point')
        bottomx = askstring('Shapes', 'Enter top right x point')
        bottomy = askstring('Shapes', 'Enter top right x point')
        oval = my_canvas.create_oval(topx, topy, bottomx,
bottomy, fill="orange")

        def oval_anim(event):
            my_canvas.delete(oval)
            my_canvas.create_oval(topx, topy, bottomx, bottomy,
fill="blue")

        def animate_stop(event):
            animate_ball(False)
```

```
        def animate_ball(event):
            xinc = 5
            yinc = 5

            while True:
                my_canvas.move(oval, xinc, yinc)
                root.update()
                time.sleep(0.05)
                ball_pos = my_canvas.coords(oval)

                al, bl, ar, br = ball_pos
                if al < abs(xinc) or ar > 600 - abs(xinc):
                    xinc = -xinc
                if bl < abs(yinc) or br > 400 - abs(yinc):
                    yinc = -yinc


        my_canvas.tag_bind(oval, '<Button-1>', animate_ball)


    if (name == "circle"):
        topx = askinteger('Shapes', 'Enter top right x point')
        topy = askinteger('Shapes', 'Enter top right y point')
        r = askinteger('Shapes', 'Enter radius')
        circle=my_canvas.create_oval(topx - r, topy - r, topx +
r, topy + r,fill="blue")

        def animate_circle(event):
            xinc = 5
            yinc = 5

            while True:
                my_canvas.move(circle, xinc, yinc)
                root.update()
                time.sleep(0.05)
                ball_pos = my_canvas.coords(circle)

                al, bl, ar, br = ball_pos
                if al < abs(xinc) or ar > 600 - abs(xinc):
                    xinc = -xinc
                if bl < abs(yinc) or br > 400 - abs(yinc):
                    yinc = -yinc

        my_canvas.tag_bind(circle, '<Button-1>', animate_circle)
    if (name == "line"):
        topx = askstring('Shapes', 'Enter top right x point')
        topy = askstring('Shapes', 'Enter top right y point')
        bottomx = askstring('Shapes', 'Enter top right x point')
        bottomy = askstring('Shapes', 'Enter top right x point')

        my_canvas.create_line(topx, topy, bottomx, bottomy,
fill="#476042", width=3)
    if (name == "triangle"):
```

```
        x1 = askstring('Shapes', 'Enter  x1 point')
        y1 = askstring('Shapes', 'Enter  y1 point')
        x2 = askstring('Shapes', 'Enter x2 point')
        y2 = askstring('Shapes', 'Enter  y2 point')
        x3 = askstring('Shapes', 'Enter x3 point')
        y3 = askstring('Shapes', 'Enter  y3 point')
        points = [x1, y1, x2, y2, x3, y3]

        triangle=my_canvas.create_polygon(points,
outline='#476042', fill='yellow', width=3)

        def animate_triangle(event):
            xinc = 5
            yinc = 5
            while True:
                my_canvas.move(triangle, xinc, yinc)
                root.update()
                time.sleep(0.05)
                ball_pos = my_canvas.coords(triangle)

                al, bl, ar, br,ca,cb = ball_pos
                if al < abs(xinc) or ar > 600 - abs(xinc):
                    xinc = -xinc
                if bl < abs(yinc) or br > 400 - abs(yinc):
                    yinc = -yinc

        my_canvas.tag_bind(triangle, '<Button-1>',
animate_triangle)
    if (name == "text"):
        topx = askstring('Shapes', 'Enter some string')
        xx = askstring('Shapes', 'Enter  x point')
        yy = askstring('Shapes', 'Enter  y point')
        my_canvas.create_text(xx, yy, text=topx)
    if (name == "rectangle"):
        topx = askstring('Shapes', 'Enter top right x point')
        topy = askstring('Shapes', 'Enter top right y point')
        bottomx = askstring('Shapes', 'Enter top right x point')
        bottomy = askstring('Shapes', 'Enter top right x point')
        rectangle=my_canvas.create_rectangle(topx, topy,
bottomx, bottomy, fill="#476042")

        def animate_rectangle(event):
            xinc = 5
            yinc = 5

            while True:
                my_canvas.move(rectangle, xinc, yinc)
                root.update()
                time.sleep(0.05)
                ball_pos = my_canvas.coords(rectangle)

                al, bl, ar, br = ball_pos
                if al < abs(xinc) or ar > 600 - abs(xinc):
```

```
                    xinc = -xinc
                if bl < abs(yinc) or br > 400 - abs(yinc):
                    yinc = -yinc

        my_canvas.tag_bind(rectangle, '<Button-1>',
animate_rectangle)
    else:
        messagebox.showinfo('Error!', 'No shape like that!')
file = Menu(menubar, tearoff=0)
menubar.add_cascade(label='File', menu=file)
file.add_command(label='New File', command=clear_screen)
file.add_command(label='Save', command=save)
file.add_command(label='Save as...', command=save_as_png)
file.add_separator()
file.add_command(label='Exit', command=root.destroy)
shapes = Menu(menubar, tearoff=0)
menubar.add_cascade(label='Shapes', menu=shapes)
shapes.add_command(label='Add one', command=shape)


brush_options_frame = Frame(root)
brush_options_frame.pack(pady=20)


brush_size_frame = LabelFrame(brush_options_frame, text='Brush
Size')
brush_size_frame.grid(row=0, column=0, padx=50)

my_slider = ttk.Scale(brush_size_frame, from_=1, to=100,
command=change_brush_size, orient=VERTICAL, value=10)
my_slider.pack(pady=10, padx=10)

slider_label = Label(brush_size_frame, text=my_slider.get())
slider_label.pack(pady=5)


brush_type_frame = LabelFrame(brush_options_frame, text='Brush
Type', height=400)
brush_type_frame.grid(row=0, column=1, padx=50)

brush_type = StringVar()
brush_type.set('round')

brush_type_radio1 = Radiobutton(brush_type_frame, text='Round',
variable=brush_type, value='round')
brush_type_radio2 = Radiobutton(brush_type_frame, text='Slash',
variable=brush_type, value='butt')
brush_type_radio3 = Radiobutton(brush_type_frame,
text='Diamond', variable=brush_type, value='projecting')

brush_type_radio1.pack(anchor=W)
brush_type_radio2.pack(anchor=W)
brush_type_radio3.pack(anchor=W)
```

```
change_colors_frame = LabelFrame(brush_options_frame,
text='Change Color')
change_colors_frame.grid(row=0, column=2)

brush_color_button = Button(change_colors_frame, text='Brush
Color', command=change_brush_color)
brush_color_button.pack(pady=10, padx=10)

canvas_color_button = Button(change_colors_frame, text='Canvas
Color', command=change_canvas_color)
canvas_color_button.pack(pady=10, padx=10)



root.config(menu=menubar)
root.mainloop()
```

Результаты работы:

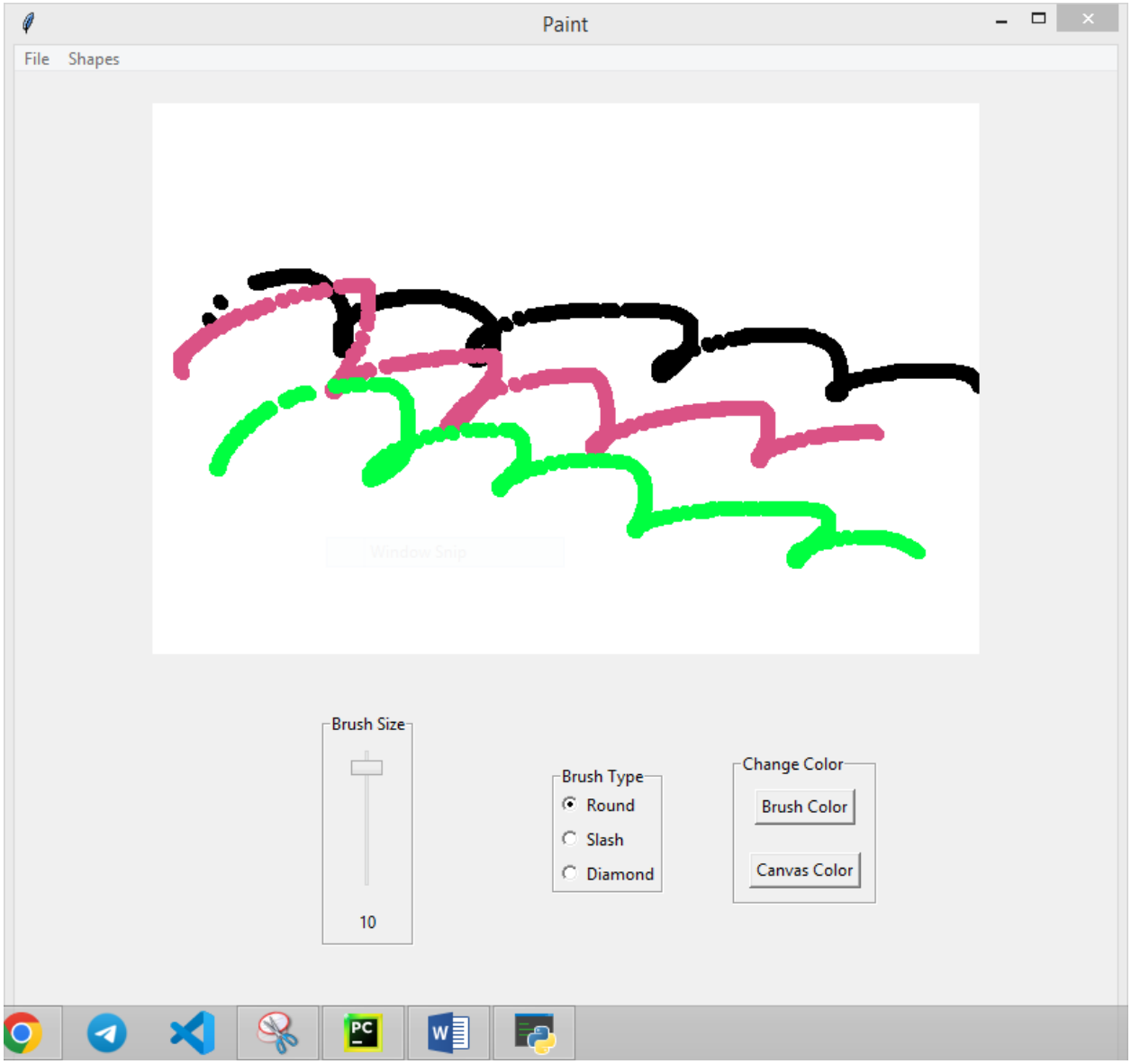

Рис.1 Рисование разными цветами

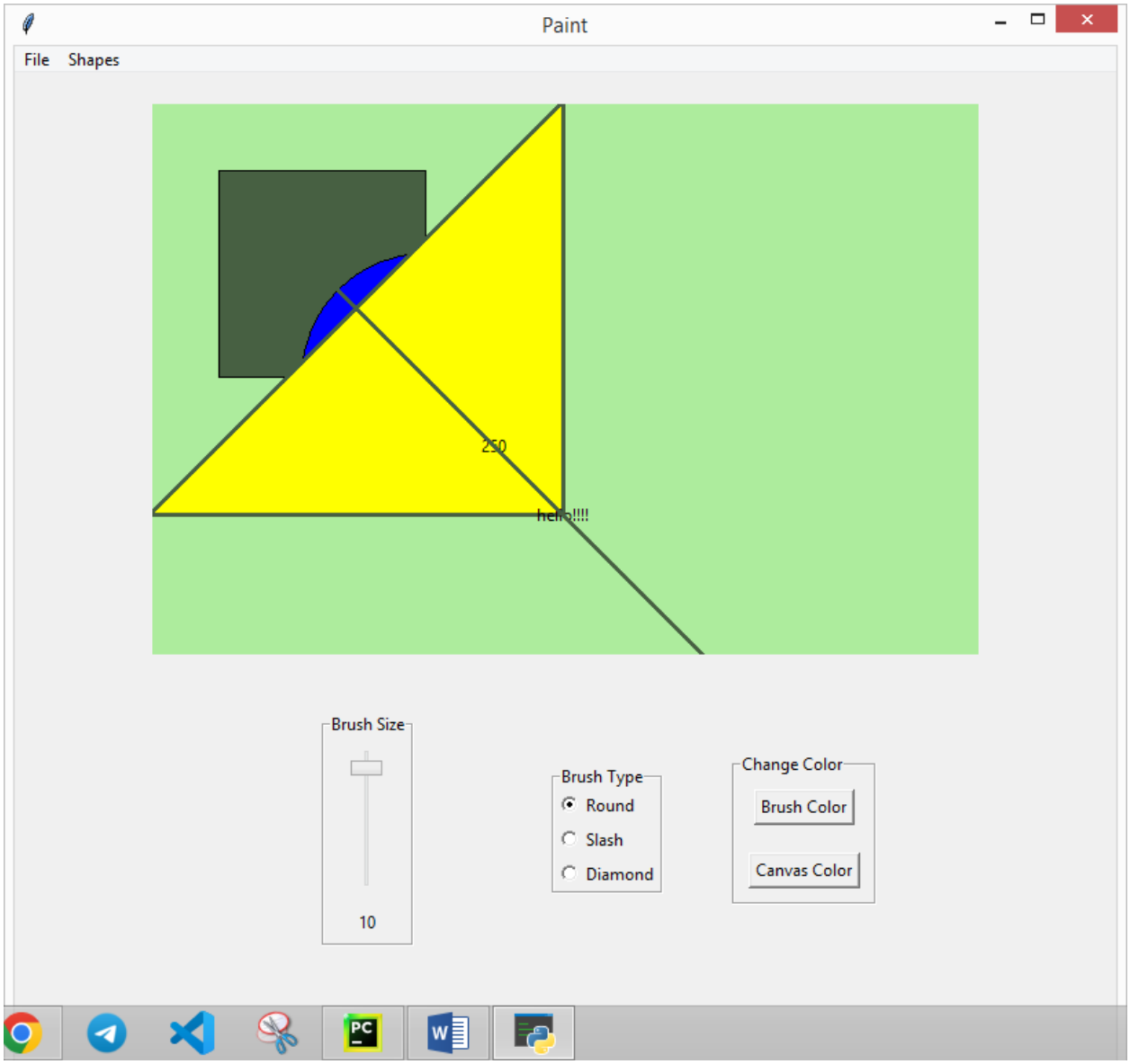

Рис. 2 Изменение цвета фона + построение различных фигур, добавление текста