```
import tkinter as tk
import tkinter.simpledialog as simpledialog
import tkinter.messagebox as
messagebox
import os
import time
import turtle
import random
import winsound

#
Constants for the screen width and height
SCREEN_WIDTH = 800
SCREEN_HEIGHT = 600
PLATE_WIDTH
= 20

# Define some bright colors for random plates
BRIGHT_COLORS = ["red",
"orange", "yellow", "cyan", "pink", "green",
"blue", "purple", "brown", "gray"]

# Background
music file path
music_file = r"C:\Users\aksha\Downloads\new_music.wav"  # Change
this to the file path of your background music (WAV format)

class Plate:
    def
__init__(self, color, size):
        self.color = color
        self.size = size

self.x = 0
        self.y = 0

def draw_plate(x, y, plate):
    turtle.penup()

turtle.goto(x, y)
    turtle.pendown()
    turtle.fillcolor(plate.color)

turtle.begin_fill()
    for _ in range(2):
        turtle.forward(plate.size)

turtle.left(90)
        turtle.forward(PLATE_WIDTH)
        turtle.left(90)

turtle.end_fill()

def draw_random_plates(plates):
    max_plate_width = max(plate.size for
plate in plates)
    vertical_spacing = PLATE_WIDTH + 10
    x = SCREEN_WIDTH // 4 -
max_plate_width // 2
    y = (SCREEN_HEIGHT - len(plates) * vertical_spacing) // 2
    for
plate in plates:
        plate.x = x
        plate.y = y
        draw_plate(x, y, plate)

     y += vertical_spacing

def animate_stack():
    global sorted_plates,
animation_completed
    sorted_plates = sorted(plates, key=lambda x: x.size, reverse=True)

 turtle.clear()
```

```
    stack_height = -200
    x = -SCREEN_WIDTH // 4 + 10
    for plate in
sorted_plates:
        y = stack_height
        draw_plate(x, y, plate)
        stack_height
+= PLATE_WIDTH + 10

    # Redraw the random plates to keep them visible
    for plate in
random_plates:
        draw_plate(plate.x, plate.y, plate)

    turtle.update()

def
is_sorted(plates):
    return all(plates[i].size >= plates[i + 1].size for i in
range(len(plates) - 1))

def on_plate_click(x, y):
    global plates, dragging_plate


for plate in plates:
        if plate.x - plate.size // 2 <= x <= plate.x + plate.size
// 2 and plate.y - 10 <= y <= plate.y + PLATE_WIDTH + 10:
            dragging_plate =
plate
            break


def stack_plates_animation(plates):  # Accept 'plates' as an
argument
    # Create the animation window
    anim_win = tk.Toplevel()

anim_win.title("Stacking Plates Animation")
    global canvas
    canvas =
tk.Canvas(anim_win, width=300, height=300)
    canvas.pack()
    x, y = 50, 250
    #
Initial coordinates for the first plate

    for plate in plates:
        # Calculate plate
width based on size (smaller plates are narrower)
        plate_width = 50 - 10 * (len(plates)
- plates.index(plate) - 1)

        # Draw the plate on the canvas
        plate_height =
10
        canvas.create_rectangle(x, y, x + plate_width, y - plate_height,
fill=plate.color)

        # Move the y-coordinate up for the next plate
        y -=
plate_height

        # Pause for a short time to create the animation effect

anim_win.update()
        time.sleep(0.5)

    anim_win.mainloop()

def
show_instructions():
    # Create the instructions window
    help_win = tk.Toplevel()
```

```
help_win.title("Instructions")

    # Instructions label
    instructions_label =
tk.Label(help_win, text="Instructions:\n\n"

       "1. Welcome to Plate Stacking!\n"

    "2. This fun program will showcase a cool animation of plates being stacked on top of
each other.\n"
                                                    "3. You will see
different-sized plates represented by rectangles of various colors.\n"

                               "4. The plates will stack from largest to smallest, starting
from the bottom.\n"
                                                    "5. The
animation will automatically start when you click the 'Let's go!' button below.\n"

                                           "6. Enjoy watching the plates stack up!\n"

                                                 "7. Feel free to close the animation window
anytime if you want to stop the animation.",

fg='white', bg='black', font=("Helvetica", 10))
    instructions_label.pack()

def
run_program():
    global animation_completed
    animation_completed = False

    # Play
the background music
    winsound.PlaySound(music_file, winsound.SND_ASYNC |
winsound.SND_LOOP)  # SND_ASYNC: Play in the background, SND_LOOP: Loop the music

    #
Prompt the user for the number of plates
    num_plates =
int(simpledialog.askinteger("Number of Plates", "Enter the number of
plates:",parent=root, minvalue=1, maxvalue=len(BRIGHT_COLORS)))

    # Create random
plates with unique colors
    random_plates = []
    random_colors =
random.sample(BRIGHT_COLORS, num_plates)
    for idx in range(num_plates):
        color =
random_colors[idx]
        size = random.randint(30, 100)
        plate = Plate(color,
size)
        random_plates.append(plate)

    # Create a copy of random_plates to preserve
the original random plates
    global plates
    plates = random_plates[:]

    # Draw
random plates
    draw_random_plates(plates)

    # Call the stack_plates_animation function
with 'plates' as an argument
    stack_plates_animation(plates)
    winsound.PlaySound(None,
winsound.SND_PURGE)  # Stop the background music

def main_window():
    global root

root = tk.Tk()
    root.title("WELCOME TO STACK PLATES")
```

```
    # Path to the
background image on the desktop
    desktop_path =
os.path.join(os.path.expanduser("~"), "Desktop")
    background_image_path
= os.path.join(desktop_path, r"C:\Users\aksha\OneDrive\Desktop\images\stacking
plates1.png")  # Use .gif format for PhotoImage

    # Check if the image file exists
before loading it
    if os.path.exists(background_image_path):
        # Load the background
image using PhotoImage
        bg_image = tk.PhotoImage(file=background_image_path)


# Create a label with the background image and place it at (0, 0)
        bg_label =
tk.Label(root, image=bg_image)
        bg_label.place(x=0, y=0, relwidth=1, relheight=1)  #
Cover the whole window with the image

    # Get screen width and height
    screen_width =
root.winfo_screenwidth()
    screen_height = root.winfo_screenheight()

    # Calculate x
and y positions for center alignment
    x = (screen_width // 2) - (300 // 2)  # Assuming a
window width of 300
    y = (screen_height // 2) - (150 // 2)  # Assuming a window height of
150

    root.geometry(f"300x150+{x}+{y}")

    # Create the heading label with
larger font size, white font color, and center alignment
    heading_label = tk.Label(root,
text="Welcome to Plate Stacking", fg='white', bg='black',
font=("Helvetica", 24, "bold"))
    heading_label.pack(pady=20)

    #
Add a button to start the program
    start_button = tk.Button(root, text="Start Plate
Stacking", bg="yellow", font=("Helvetica", 14, "bold"),
command=run_program)
    start_button.pack(pady=20)

    # Help button to show
instructions
    help_button = tk.Button(root, text="Help",
bg="lightblue", font=("Helvetica", 12, "bold"),
command=show_instructions)
    help_button.pack(pady=10)

    root.mainloop()

if __name__
== "__main__":
    main_window()
```