How to build a Paint application with Python

Programming a painting application is a basic exercise that will teach you a lot about GUI programming. Below are instructions on how to create a painting application in Python.

A simple painting tool is the most common application you can find on most computers. It allows artists to make mistakes without worrying, choose any color with a click of a button, and change the size of the brush instantly. You can use it to create brand logos, conceptualize user interfaces, and annotate diagrams.

Tkinter and Pillow Modules

To build a painting app, you need the Tkinter and Pillow modules. Tkinter is one of the top Python frameworks that you can use to customize your GUI. It is the standard Python GUI module for creating desktop applications. Tkinter offers a series of different widgets such as labels, entries, canvas and buttons.

Pillow, a fork of the Python image library (PIL), is an image processing module for Python. With Pillow, you can open, resize, flip, and crop photos. You can convert file formats, build recipe finder apps, and fetch random images.

To install these modules, run:

```
pip install tk pillow
```

Determine the structure of the painting application

Start by importing the required modules. Define a class, **DrawApp** . Set title, cursor color and eraser color. Make the application open in full screen. **Call the setup_widgets** method .

```
import tkinter as tk from tkinter.ttk import Scale from tkinter import colorchoos
```

Define a method named **setup_widgets** . Specifies a label that shows **heading** . Set the parent element, the content you want to display, font style, background color, text color. Defines a frame for the color palette. Set the parent element, the content it will display, the font style and border width. Set the border shape as desired and the background color to white.

```
 def setup_widgets(self): self.title_label = tk.Label( self.root, text="Kids' Pa
```

Specify a color group for the palette in a list. Repeat it and create buttons for each element. Set the parent element, its background color, border width, and shape. Additionally, set the width and command for each button to run when clicked. Arrange all components with appropriate **padding** and colors according to the set of

both parts.

```
colors = [ "blue", "red", "green", "orange", "violet", "black", "yellow", "purp
```

Similarly, define a button for the eraser, one to clear the screen, and one to save the image.

```
self.eraser_btn = tk.Button( self.root, text="Eraser", bd=4, bg="white", comman
```

Define a scale widget to increase or decrease the size of the cursor or eraser. Set the parent element, orientation, extent, and length in pixels. Define a canvas and set the parent element, background color, and border width. Also, set a shape that matches its width and height.

Locate the canvas with appropriate coordinates and set the anchor to the top left. Link **B1-Motion** to the drawing function. **B1** just holds the left mouse button and **Motion** just moves. Generally, you use it to track mouse movements, while pressing the left mouse button.

```
self.pointer_frame.place(x=10, y=580, height=150, width=70) self.pointer_size =
```

Identify drawing application features

Define a method, **paint** . To draw pictures, this application will continuously draw small ovals. **Subtract 2 from the x** and **y** coordinates of this mouse event to determine the upper left corner of the oval. Add 2 to define the lower right corner of the oval. Create an oval using these bounded coordinates.

Set the fill color, border color, and width for each cursor selection.

```
def paint(self, event): x1, y1 = (event.x - 2), (event.y - 2) x2, y2 = (event.x
```

Define 3 functions, **select_color** , **eraser** , and **clear_screen** . **The select_color** method takes a color and sets the cursor accordingly. **The eraser** method sets the cursor to an eraser-like effect and makes it draw transparent lines. **The clear_screen** method clears the entire item on the canvas.

```
def select_color(self, col): self.pointer = col def eraser(self): self.pointer =
```

Define a method, **canvas_color** . Open the color picker with all the different colors. Open the color picker containing all the different colors. Returns a tuple containing colors in RGB and hexadecimal format. **If the user chooses a color, use the configure** method to set the background color. Set the eraser color to the same as the background color.

```
def canvas_color(self): color = colorchooser.askcolor() if color: self.canvas.c
```

Define the **save_as** method . Opens a file dialog box asking the user to select a file name and path. If the user selects a link, use Pillow's **ImageGrab** class to capture the entire screen. Crop the image according to specific coordinates to get the canvas area. Experiment with coordinates to get the desired part.

Save the results to the desired file path. A box will appear notifying the user that the program has successfully saved the drawing as an image. In case an error occurs, it displays the corresponding error.

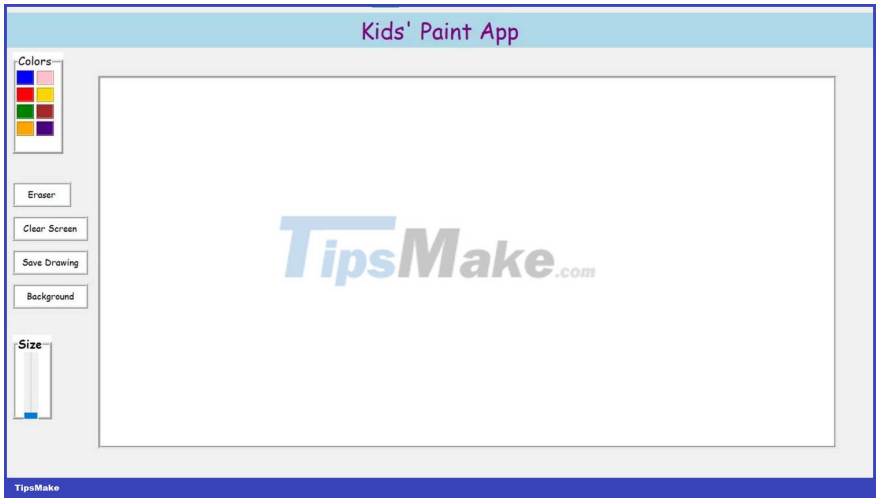
```
def save_as(self): file_path = filedialog.asksaveasfilename( defaultextension="
```

Create instances of classes **Tk** and **DrawApp** . **The mainloop()** function tells Python to loop Tkinter events and listen for events until you close the window.

```
if __name__ == "__main__": root = tk.Tk() app = DrawApp(root) root.mainloop()
```

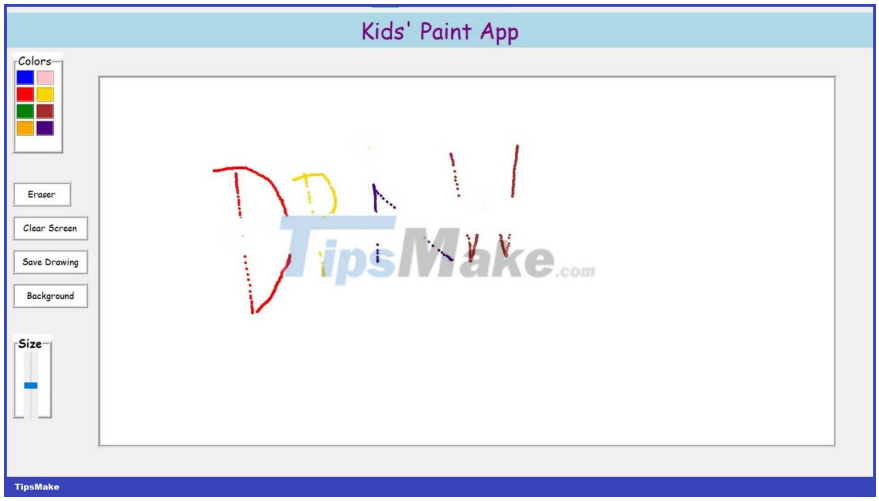
Test out different drawing features in Python

When you run the drawing program, you will see an application with a color palette, 4 buttons, a slider and a canvas to draw on:

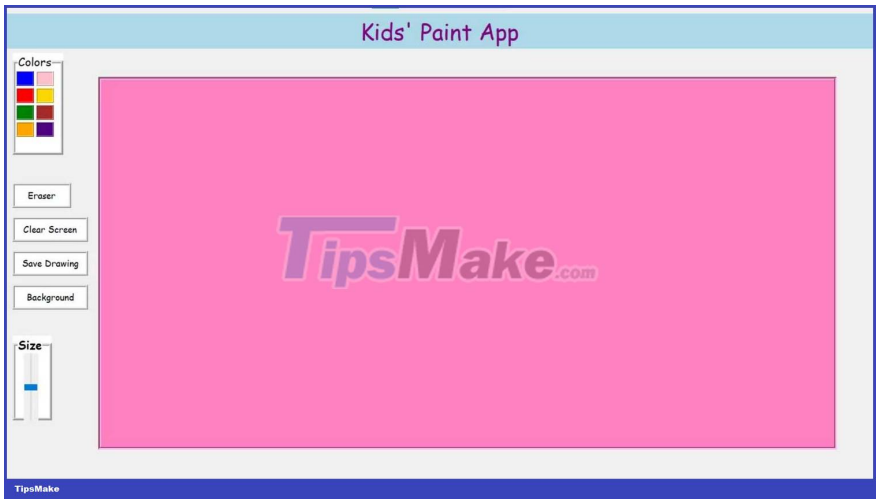


Click any color to select it. You can then draw on the canvas in that color with the left mouse button:

When you click the **Eraser** button and drag the slider vertically up, you will select the eraser and increase its size. Test the eraser by dragging it over the drawing to erase unnecessary strokes.



When clicking the **Clear Screen** button , the program deletes the previous drawing. Click the **Background** button to open the color palette and use it to change the background color.



When you click the **Save Drawing** button , a file dialog box will open. Select the path and name the file. This program will save it.

You can enhance the drawing application's features by adding additional shape options, choosing brush styles, opacity, applying stickers, etc. Add options to undo, redo, resize, flip images. This makes the drawing process smoother.

To create shapes, you can use methods like create_rectangle, create_oval, create_line, create_polygon. To add images, use create_text and create_image.

Above is **how to create a painting app with Python** . Hope the article is useful to you.

You finished reading the article "**How to build a Paint application with Python**" edited by the **TipsMake** team. We hope this article has provided you with many useful tech tips and tricks. You can search for similar articles on tips and guides. Thank you for reading and for following us regularly.
