Python Project

Documentation

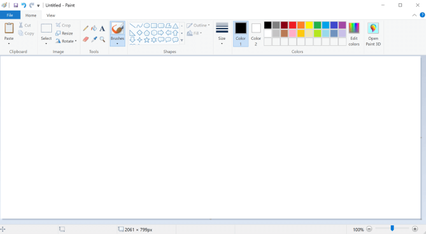
Description

The program will be a simple pygame project written entirely in Python. The functionality of the program should be primarily for drawing or painting images and shapes in a variety of different colours.

Analysis

The program will require nothing beyond a mouse, a keyboard is not necessarily as the only input will be a paint brush following the user’s cursor. The program will require a monitor and a computer capable of running Python 3.6, including pygame. The program will need to have a variety of different colours including the colours red, green, and blue and any secondary colours added on top. The program will have different tools for drawing, including a paint brush, an eraser, a pencil, a spray can, and various shapes primarily including a rectangle and a circle which will not be resizable. The paint brush should draw a circle where the user first clicks the mouse, and then draw a smooth, continuous line following the user’s cursor as they move around the screen. The pencil should do the same but rather than using a circle, it will instead draw a small square and continuously draw squares as the user moves their cursor. The eraser tool will function exactly the same as the pencil tool but will only be available in the colour white, in order to function as an eraser rather than another drawing tool. The spray can will create multiple small circles within a given radius in randomly generated positions using the random library, this will also be adjustable based on colour and the radius within which the tool can draw small circles. The shapes which will definitely be available to draw will be the circle and the square, which will be adjustable based on colour but not size or dimensions.

Additionally, each tool (with the exception of the shape tools) will be adjustable based on size, with three options: a thin line, a medium line, and a thick line. The colour should be retained when each option is selected, alongside which tool is currently in use, as this will result in the most simple and user friendly experience.

The program should have a clear and easy to use user interface so that the user can quickly and easily navigate to the option they want to use and select it. The UI’s colour scheme should be accessible to people who are vision impaired through the use of sensible colours, and the layout should be logical, for example: all colours grouped together, all tools grouped together, all line thickness adjustments grouped together, and all shapes grouped together. The program should also include a canvas section for the user to draw on, and this should be by default white in order to reflect paper, or an actual canvas. The user should be able to draw only on the canvas and never on the UI, including when selecting another tool or colour to use. The UI will be based on Microsoft Paint but not a direct replica, rather similar in the way that each tool is grouped together.

Success Criteria:

* Should run on virtually any desktop or laptop computer with a monitor and mouse
* Should have a basic paint brush tool, including:
  + A circle drawn following the user’s cursor
  + Colour change
  + Line thickness change
* Should have a basic pencil tool, including:
  + A square drawn following the user’s cursor
  + Colour change
  + Line thickness change
* Should have a basic eraser tool, including:
  + A white square drawn following the user’s cursor
  + Line thickness change
* Should have a spray can tool, including:
  + Several small circles drawn in randomly in a given radius
  + Colour change
  + Radius change
* Should have square and circle drawing tools, including:
  + Colour change
* Should have a sensible and well laid out UI.

Design

Necessary Functions

Smooth Drawing Function

This is a function which will calculate a line between the user’s current position while the mouse is held down and the user’s previous position, this will make the line appear to be drawn much smoother than it would have been without, as it would instead appear as a series of circles.

Spray Can Tool

The spray can tool will work using the following pseudocode:

For i in range(0, 10):

sprayPosition = (0, 0)

sprayPosition[0] = mousePositionX + random.randint(boundary1, boundary2)

sprayPosition[1] = mousePositionY + random.randint(boundary1, boundary2)

draw circle in desired colour using sprayPosition for position value

This will draw a series of very small circles within a given range, given by boundary1 and boundary2. As the user moves the mouse, more dots will be drawn following the user’s position.

Canvas Fill Tool

A simple tool to change the entire colour of the canvas. This function will not pay attention to lines already drawn and will not act as a fill can in the same way as MS Paint, instead simply filling the entire canvas with the desired colour.

Pseudocode

Set up pygame window

Set resolution to 800x600

Draw UI rectangle across top of screen

Draw colour element rectangles

Draw line thickness rectangles

Draw tool icon rectangles

Tool = paintBrush

brushColour = black

Thickness = medium

Def smoothStroke:

Draw circles joining distance from current position to previous position

While True:

If colour element is clicked:

brushColour = colour from colour element

If paintBrush tool is clicked:

Tool = paintBrush

If sprayCan tool is clicked:

Tool = sprayCan

If canvasFill tool is clicked:

Tool = canvasFill

If circle tool is clicked:

Tool = circle

If square tool is clicked:

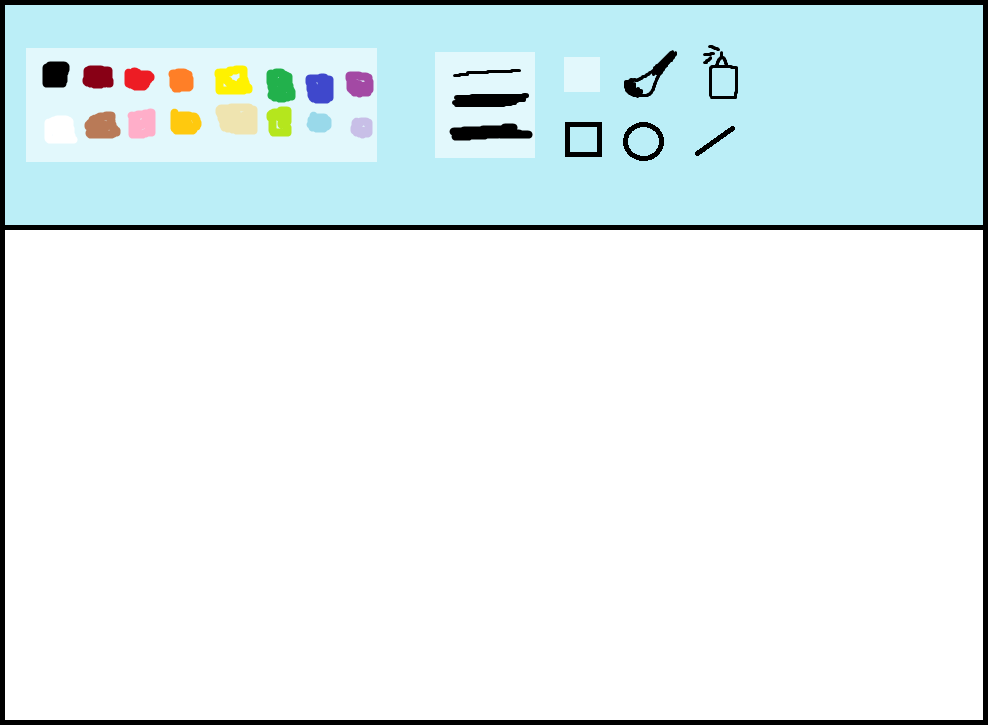
Tool = square

If mouse button is clicked on canvas, not on UI:  
 Draw from tool

If mouse is moved:

Call smoothStroke function

UI Design

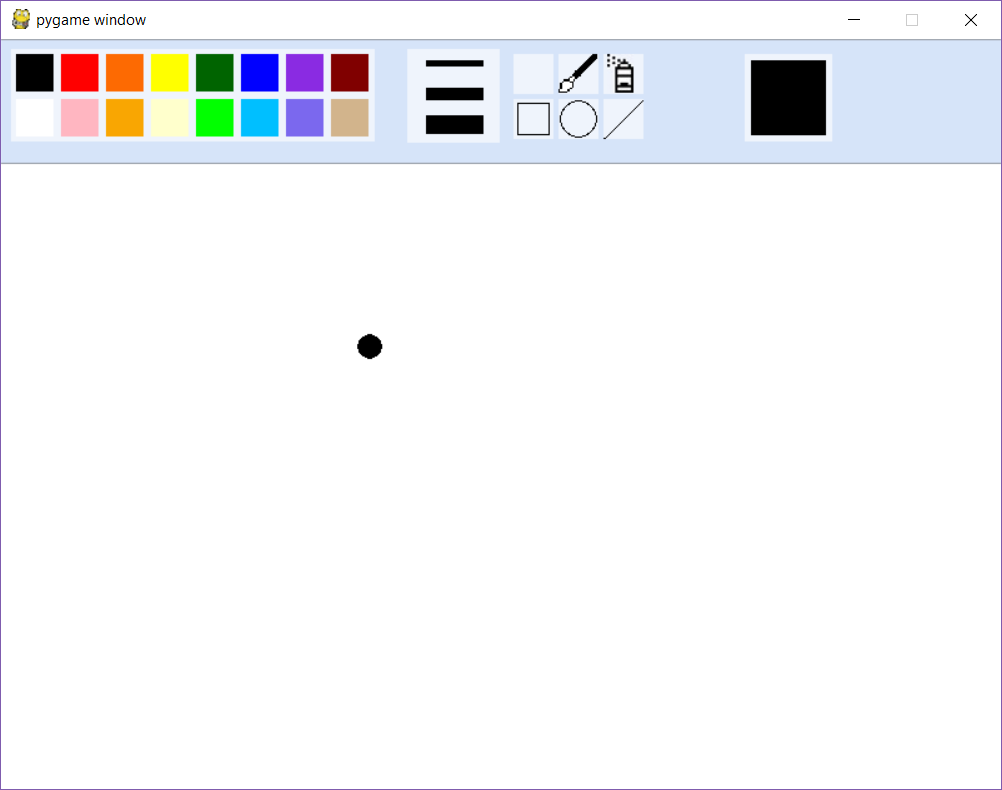


Testing

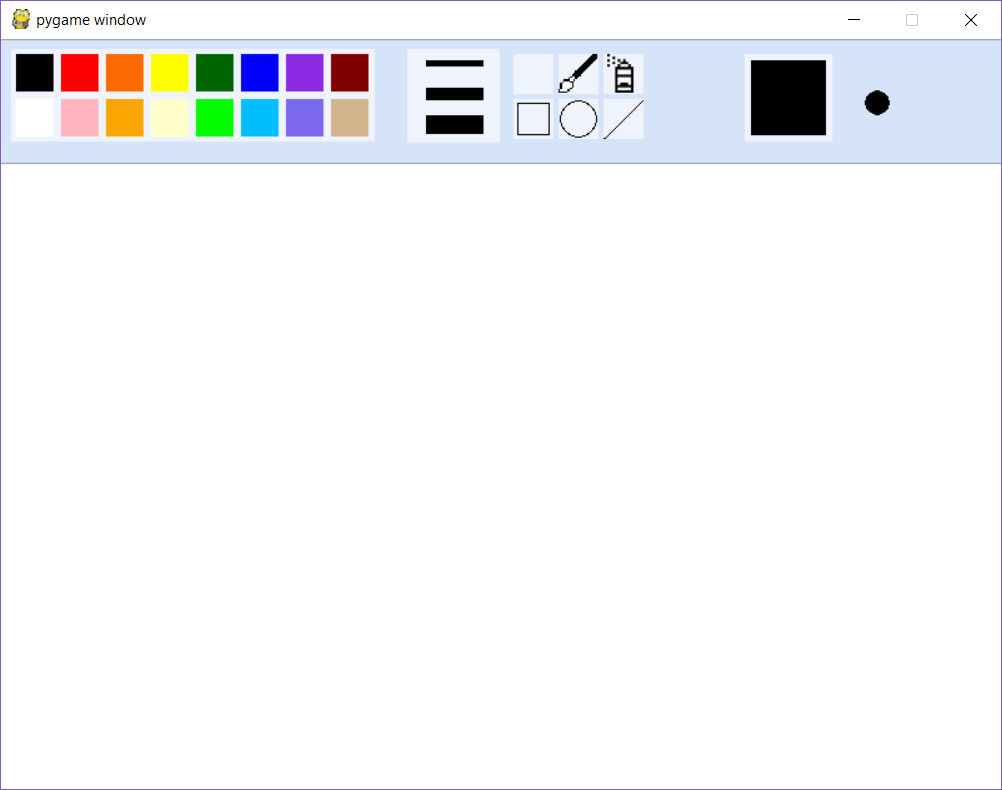
|  |  |  |  |
| --- | --- | --- | --- |
| **Test** | **Expected Outcome** | **Actual**  **Outcome** | **Action**  **Needed** |
| See if all UI elements correctly appear | All UI elements should load in the correct order, with no elements hidden behind the background | All UI elements loaded in the correct order.  (See evidence 1) | No action needed. |
| Drawing a small circle in the position of the cursor by clicking | When clicking on the canvas (white area) the program should draw a small circle in the position of the cursor. | The program drew a small circle in the position of the cursor when the user clicked.  (See evidence 1) | No action needed. |
| Drawing a small circle on the UI | When clicking on the UI, the program should NOT draw a small circle in the position of the cursor. | The program did draw a small circle on the UI and therefore needs to be resolved.  (See evidence 2) | Draw each UI element within the while True loop so that it’s drawn each frame. |
| Drawing a small circle on the UI again | When clicking on the UI, the program should NOT draw a small circle in the position of the cursor. | The program did not draw over the UI.  (See evidence 4) | No action needed. |
| Drawing a streak with left click on the canvas | When moving the mouse while holding left click, a streak of circles should be drawn to form a line. | The program drew a succession of lines, but the line seemed patchy.  (See evidence 3) | Use a function which calculates the distance between the current dot and the previous and joins them. |
| Drawing a smoother streak with left click on the canvas | When moving the mouse while holding left click, a streak of circles should be drawn to form a smoother line. | A smoother line was drawn.  (See evidence 4) | No action needed. |
| Change colour of drawn dot by clicking each UI element using a class | When each colour element is clicked the brush should change to the desired colour. | The UI crashed immediately. Various other solutions using classes had various bugs.  (See evidence 4) | Switch to using if statements. |
| Change colour of drawn dot by clicking each UI element using if statements | When each colour element is clicked the brush should change to the desired colour. | The colours changed successfully.  (See evidence 4) | No action needed. |
| Change line thickness between thin, medium, and thick | When each line thickness is selected, the drawn line should change accordingly with all colours. | The thickness changed successfully (See evidence 4) | No action needed. |
| Change tool to spray can | When the spray can icon is clicked, the tool should instead use a spray like pen. | The tool changed successfully. The drawn line appeared using a spray effect.  (See evidence 4) | No action needed. |
| Change tool back to paint brush | When the paint brush icon is clicked, the tool should revert back to being a smooth paint brush. | The tool successfully reverted back to a paint brush and drew a smooth line.  (See evidence 4) | No action needed. |
| Change tool to canvas fill | When the canvas fill button is clicked, the tool should change to canvas fill and fill the screen when a colour is picked. | The tool successfully changed to the canvas fill tool. When a colour was selected, the screen successfully filled with that colour | No action needed. |
| Change tool to square tool | When the square tool is clicked, the tool should switch to the square tool and when clicking on the canvas, a square should be drawn. | The tool successfully changed to the square tool. When clicking the canvas, a square was successfully drawn (See evidence 4) | No action needed. |
| Change tool to circle tool | When the circle tool is clicked, the tool should switch to the circle tool and when clicking on the canvas, a circle should be drawn. | The tool successfully changed to the circle tool. When clicking the canvas, a circle was successfully drawn (See evidence 4) | No action needed. |
| Change tool to line tool | When the line tool is clicked, the tool should switch to the line tool and when clicking on the canvas, a line should be drawn. | The tool successfully changed to the line tool. When clicking the canvas, a line was successfully drawn (See evidence 4) | No action needed. |

Evidence

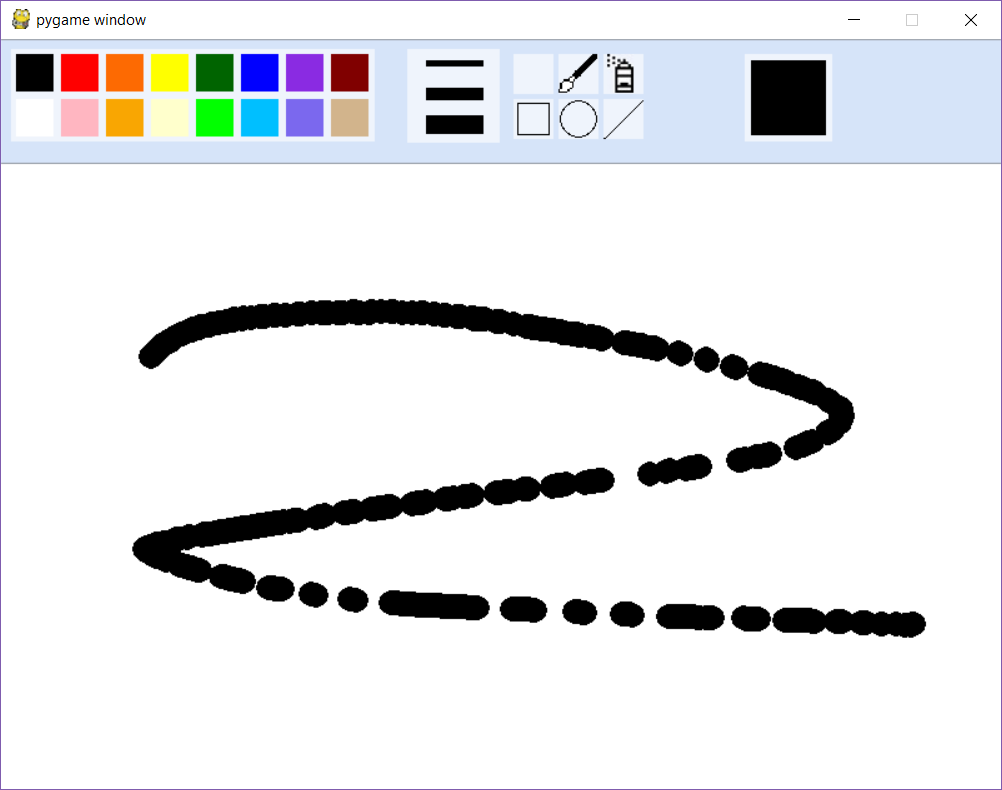
Evidence 1:



Evidence 2:



Evidence 3:



Evidence 4:



Evaluation

Success Criteria Revisited:

* Should run on virtually any desktop or laptop computer with a monitor and mouse **[1]**
* Should have a basic paint brush tool, including: **[2]**
  + A circle drawn following the user’s cursor
  + Colour change
  + Line thickness change
* Should have a basic pencil tool, including: **[3]**
  + A square drawn following the user’s cursor
  + Colour change
  + Line thickness change
* Should have a basic eraser tool, including: **[4]**
  + A white square drawn following the user’s cursor
  + Line thickness change
* Should have a spray can tool, including: **[5]**
  + Several small circles drawn in randomly in a given radius
  + Colour change
  + Radius change
* Should have square and circle drawing tools, including: **[6]**
  + Colour change
* Should have a sensible and well laid out UI. **[7]**

**[1]** - The program has yet to encounter a computer on which will not run the program, therefore this aspect is successful.

**[2]**  - The program does have a basic paint brush tool which will follow the user’s cursor, change colour, and change line thickness.

**[3]**  - The program does not have a basic pencil tool as this was not necessary and would function virtually the same as the paintbrush tool. Given more time, and more developers working on the program, this could be implemented easily in the future.

**[4]**  - The program does not have a basic eraser tool as this was not necessary, since the user can just use the paintbrush tool to match the background colour.

**[5]**  - The program successfully includes a spray can tool which incorporates a colour change and a radius change.

**[6]** - The program includes both a square tool and a circle tool, which can change colour.

**[7]** - The program has a reasonably visible and well laid out UI.

Additional Functionality

* The program includes a canvas fill button which was not in the original design.
* The program includes a current colour indicator, which was not in the original design.
* The program expands upon the square and circle tool by allowing the user to choose a radius.

What Could Be Improved

Rather than using global variables, the program could instead contain all functionality within functions. This would be better for larger programming teams as there would be fewer conflicts in terms of variable naming. Therefore, the amount of global variables should be minimised.

Instead of selecting the colour through a series of ‘if’ statements, it would be more efficient for the programmer to simply have that functionality within a class. This approach was attempted multiple times, however was found to be very difficult due to inexperience of the developer with classes.

The program could allow the user to resize the circle and square tools rather than simply placing them down. It should also allow the user to choose whether to have the circles and squares filled in or not. If the project is to be continued, it could be built upon to perform this, however due to limited time and only having one developer work on the program, this is not entirely feasible.

References

Smoothstroke function from:

<https://stackoverflow.com/questions/597369/how-to-create-ms-paint-clone-with-python-and-pygame> accessed on [10/11/2018]