**Paint Program** Final Project Report CSE3013 Computer Lab I

20201658 GAN EUNICE NELL

1. Program Description

This program seeks to become a far duplicate of Microsoft’s ‘Paint’ with limited but various brush options. Brush options are selected through the Windows dialog menu.

The following are some supported brush options :

1. **Single Rectangle Brush** – a brush that draws strokes made up of solid rectangle shapes
2. **Single Rectangle Gradient Brush** – a brush that draws similar to the single brush, but each rectangle is drawn in a random color from the same color palette, thus creating a gradient effect
3. **Bursting/Scatter Rectangle Brush** – a brush that draws strokes of scattered solid rectangle shapes of randomized color
4. **Glowing Circle Brush** – a brush that draws strokes of faded and colorful glowing circles

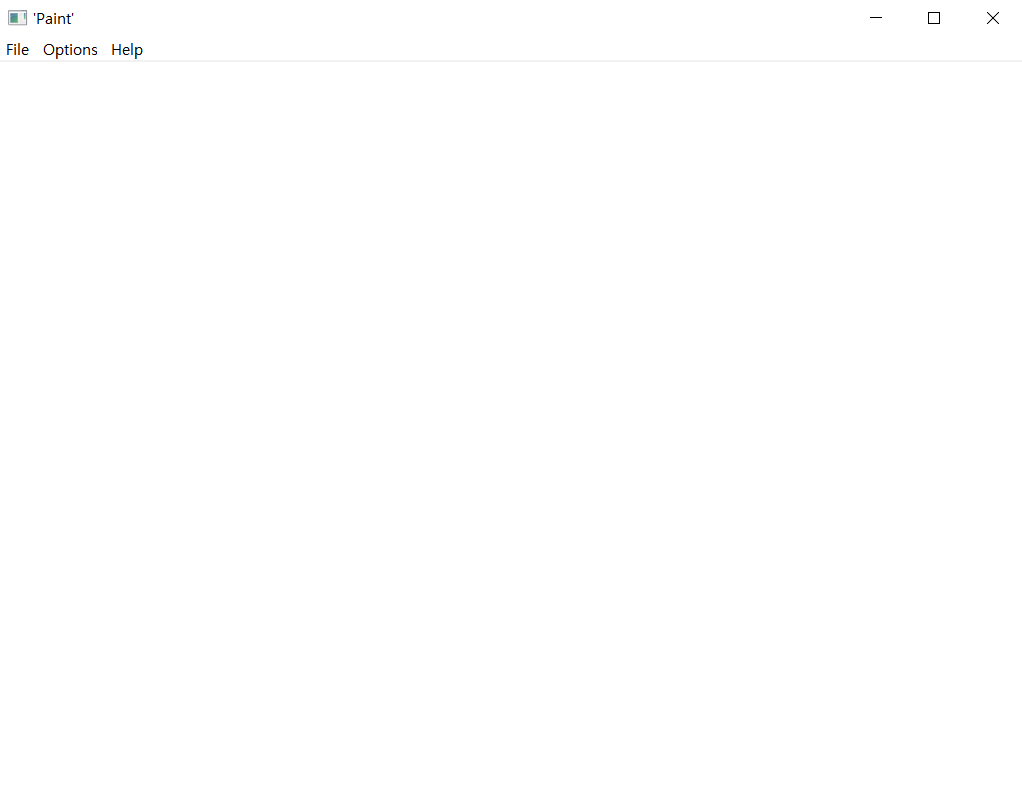
This program allows users to Save a screenshot of their drawings to bin\data directory or erase all their drawings by pressing the right-click button on their mouse. Keyboard shortcuts are also supported for ‘Save’ (Key S) and ‘Exit’ (Key ESC) options.

A separate ‘About’ section is also created to provide some basic info about the program.

As for the coding environment, this program was created through the OpenFrameworks Project Generator and coded using openFrameworks (version 0.11.2) and C++ on Visual Studio 2019 (version 16.8.6).

1. How to Use
   1. User Interface
      1. Screen / Canvas Layout

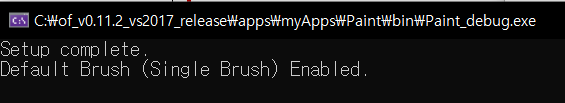
**Window Title**



**Windows Dialog Menu**

**CANVAS**

Console Preview

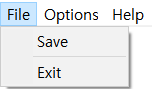


The program window consists of the Title, the Windows dialog menu, and the Canvas. The Window Title is set to ‘Paint’ hinting the program being a parody of MS Paint. The Windows dialog menu, created through OpenFramework’s ofxWinMenu functions, consists of three menus : File, Options, and Help. (Further information on each menu will be provided below.) The setup starts with a white background canvas where users can draw / paint on.

* + 1. Windows Dialog Menu Layout

**File Menu** (Figure 1)

* **Save –** saves screenshot of canvas  
  **Keyboard Shortcut : ‘s’ or ‘S’**

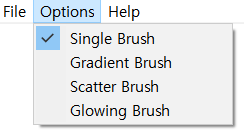


**Figure 1**

* **Exit –** exits program  
  **Keyboard Shortcut : ‘ESC’**

**‘Brush’ Options Menu** (Figure 2)

* **Menu where you can select brush options (each option will be checked when enabled or unchecked when disabled)**



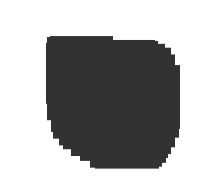
**Figure 2**

1. **Single Brush –** selects Single Rectangle Brush
2. **Gradient Brush –** selects Single Rectangle Gradient Brush
3. **Scatter Brush –** selects Bursting/Scatter Rectangle Brush
4. **Glowing Brush –** selects Glowing Circle Brush

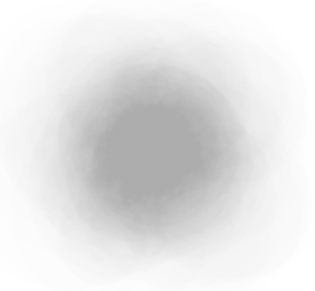
**(Keyboard Shortcuts for Brushes   
– Not Yet Supported)**



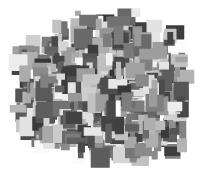
**Gradient Brush**



**Single Brush**



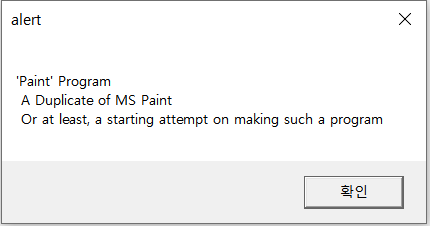
**Glowing Brush**



**Scatter Brush**

**Help Menu** (Figure 3)

1. **About –** shows ‘About Program’ info through dialog box (Figure 4)  
   **(Keyboard Shortcut for About Info – Not Yet Supported)**



**Figure 4**



**Figure 3**

* 1. Functions and Variables

There are three necessary source files for this project – main.cpp, ofApp.cpp, ofApp.h, along with the two source files needed for the Windows dialog menu (provided with openFrameworks) – ofxWinMenu.cpp, ofxWinMenu.h.

The functions and variables provided with the Windows Menu source files will not be discussed here.

The following is an explanation on the workings of the necessary variables and functions for this project of which are declared in ofApp.h (header file) and defined in ofApp.cpp (C++ source file).

**Functions Used and Defined**

void setup()

* + - Function which prepares the initial window setup

void draw()

* + - Function which draws strokes on the canvas depending on the brush option selected

void saveScreenshot()

* + - Function which saves the screenshot of the canvas

void keyPressed(int key)

* + - Input : int key
    - Determines which key was pressed on the keyboard

**Option Menu Variables**

bool bSingleBrush – variable to determine if Single Brush is selected

bool bGradientBrush – variable to determine if Gradient Brush is selected

bool bScatterBrush – variable to determine if Scatter Brush is selected

bool bGlowingBrush – variable to determine is Glowing Brush is selected

* 1. Thought Process
     1. Initial Algorithm

**START**

*FUNCTION setup*

Show Program Window

Show white background canvas

Show Windows dialog menu (

File > Save, Exit

Options > [Insert Brush options as popup menu here]

Help > About

)

Select Single Brush by default

*END FUNCTION*

*FUNCTION appMenuFunction*

If (Save is selected) {

Save screenshot using separate function

}

If (Exit is selected) {

Exit program

}

For every brush menu :

If (Brush Option is selected) {

Uncheck other brush menus

Check Brush Option menu

Call draw() function

}

*END FUNCTION*

*FUNCTION draw*

Verify which Brush option is selected (if-else)

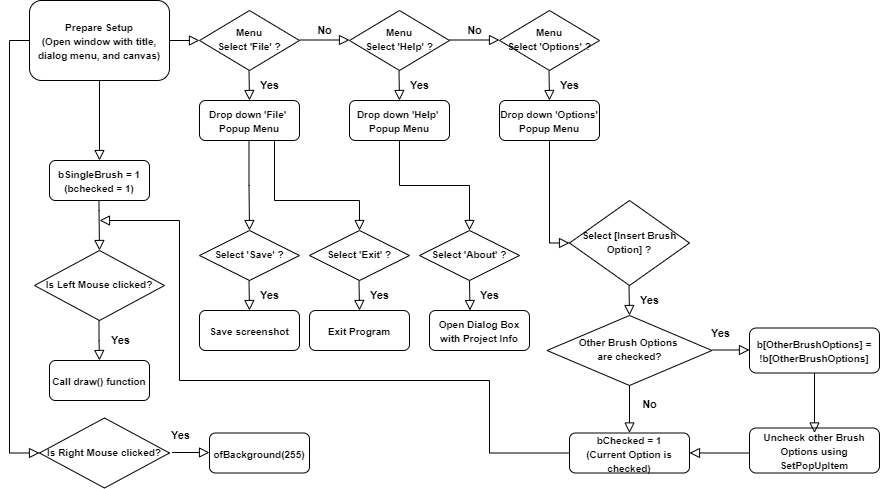
If MOUSE\_LEFT\_CLICK, then draw according to Brush Option

If MOUSE\_RIGHT\_CLICK, then erase canvas

*END FUNCTION*

**END**

* + 1. Flowchart



To explain,

First, the setup() function prepares the window setup consisting of the title, the dialog menu and the white canvas. Then, the program enables the Single Rectangular Brush by default.

If the user hovers over the dialog menu and selects the File Menu, then the user will have the choice to select Save or Exit. By clicking ‘Save’, the user can save a screenshot of their canvas (which can also be done by pressing ‘s’ or ‘S’ on the keyboard). By clicking ‘Exit, the user can exit the program (which can also be done by pressing ESC key on the keyboard or clicking the Close Button on the upper-right corner of the window). If the user hovers over the dialog menu and selects the Help Menu, the user can select ‘About’ and a dialog box will appear giving some brief info about the program. If the user hovers over the dialog menu and selects the Options Menu, the user will see the Brush options (i.e. Single Brush, Gradient Brush, Scatter Brush, and Glowing Brush). The user may choose one of these options. Each time the user chooses an option, the other brush options are unchecked on the menu and their Boolean variables are switched to 0 (!b[BrushOption]).

Whenever the left mouse button is clicked on the canvas, the program will call the draw() function which will draw strokes depending on the enabled brush option. If the right mouse button is clicked, all the drawings on the canvas are erased.

Each brush option was implemented through the draw() function. The unique characteristics of each brush were implemented as in the following code :

// SINGLE RECTANGLE BRUSH

if (bSingleBrush) {

if (ofGetMousePressed(OF\_MOUSE\_BUTTON\_LEFT)) {

// If the left mouse button is pressed...

ofSetColor(50);

ofSetRectMode(OF\_RECTMODE\_CENTER); ofDrawRectangle(ofGetMouseX(), ofGetMouseY(), 50, 50);

// Draw a 50 x 50 rect centered over the mouse

}

}

// GRADIENT RECTANGLE BRUSH

else if (bGradientBrush) {

if (ofGetMousePressed(OF\_MOUSE\_BUTTON\_LEFT)) {

float randomColor = ofRandom(50, 255);

ofSetColor(randomColor);

ofSetRectMode(OF\_RECTMODE\_CENTER); ofDrawRectangle(ofGetMouseX(), ofGetMouseY(), 50, 50);

}

}

// BURST/SCATTER RECTANGLE BRUSH

else if (bScatterBrush) {

if (ofGetMousePressed(OF\_MOUSE\_BUTTON\_LEFT)) {

ofSetRectMode(OF\_RECTMODE\_CENTER);

int numRects = 10;

for (int r = 0; r < numRects; r++) {

ofSetColor(ofRandom(50, 255));

float width = ofRandom(5, 20);

float height = ofRandom(5, 20);

float xOffset = ofRandom(-40, 40);

float yOffset = ofRandom(-40, 40);

ofDrawRectangle(ofGetMouseX() + xOffset, ofGetMouseY() + yOffset, width, height);

}

}

}

// GLOWING CIRCLE BRUSH

else if (bGlowingBrush) {

int maxRadius = 100; // Increase for a wider brush

int radiusStepSize = 5; // Decrease for more circles (i.e. a more opaque brush)

int alpha = 3; // Increase for a more opaque brush

int maxOffsetDistance = 100; // Increase for a larger spread of circles

// draw smaller and smaller circles and layering (increasing) opaqueness

if (ofGetMousePressed(OF\_MOUSE\_BUTTON\_LEFT)) {

for (int radius = maxRadius; radius > 0; radius -= radiusStepSize) {

float angle = ofRandom(ofDegToRad(360.0));

float distance = ofRandom(maxOffsetDistance);

float xOffset = cos(angle) \* distance;

float yOffset = sin(angle) \* distance;

ofSetColor(132, alpha);

ofDrawCircle(ofGetMouseX() + xOffset, ofGetMouseY() + yOffset, radius);

}

}

}

As for erasing the drawings on the canvas, the right-click feature makes the program turn the background into white again.

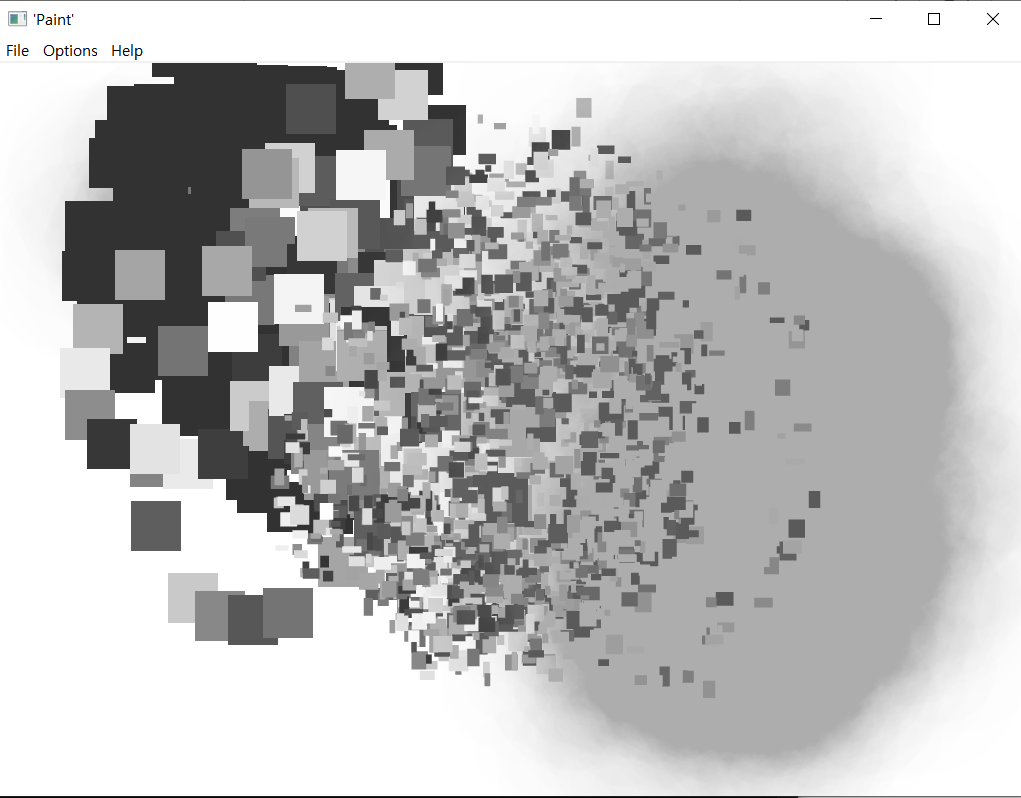
// If the right mouse button is pressed...

if (ofGetMousePressed(OF\_MOUSE\_BUTTON\_RIGHT)) {

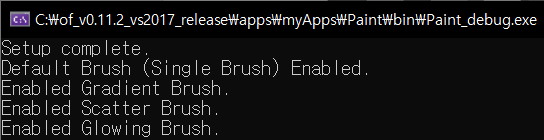
ofBackground(255); // Erase the screen with a white background

}

**Capture of Program Run**



**Console Preview**



1. Additional Remarks

Seeing that OpenFrameworks has a vast library of functions, I have realized that there is so much more to explore and utilize in it to make creative code. While building this Duplicate ‘Paint’ program, I found myself researching more about creating my own drawing tools as well as graphics and how I can feature this in my Paint program. Sadly, however, I am not fully adept in utilizing OpenFrameworks and its vast library of functions to apply more features to this program. So, for the meantime, I shall list some possible extensions or things to improve for this program.

1. Add an option to make the brushes draw different shapes apart from circles and rectangles
2. Add an option to change brush colors and background colors
3. In addition to using Windows Dialog Menu to select brushes, keyboard shortcuts can also be added (e.g. press ‘1’ to select single brush, press ‘2’ to select single gradient brush)
4. Instead of using Windows dialog Menu, use on-screen buttons.   
   Note : requires boundary for painting canvas (need to code separately, perhaps use mouseExited(int x, int y) OF function)
5. Add ‘Open’ and ‘Save As…’ options to import and export the drawings created in the program
6. (Advanced – needs tree or queue data structures) Add ‘Undo’ and ‘Redo’ features

These extensions shall be added in the near future as further knowledge and expertise on the usage of the OpenFrameworks is accumulated.