Title: Photo-Mosaic App

Introduction:

The Photo-Mosaic converts your favourite photos into an artistic mosaic of random shapes. It gives users the liberty to choose the number of pieces used to create the mosaic making it more user friendly and on the other hand the auto-mode is a hassle-free experience, which creates the mosaic using random pieces. It also allows the user to choose between a number of styles of filters making the image even more artistic and fun and the user can then save the image to his device in a desired space and a desired format. It not only allows the user to choose from the library of photos but also allows the user to take photos from the camera and directly convert it into a mosaic within the application. This makes the application user friendly and easy to use providing an artistic mosaic in a few moments.

Problem Statement:

Create an app which takes a coloured image as input and creates a mosaic as output.

Timeline

29-Jun-2020: Problem statement declared

1-Jul-2020: Brainstorming and discussion session on how to approach the project.

2-Jul-2020: Work on GUI and code for making random triangles using Subdiv2D begins

4-Jul-2020: Work on generating random triangles completed

5-Jul-2020: Work on filling mosaic pieces, merging 2 triangles, adding webcam and filters to the application begins

6-Jul-2020: Work on adding filters completed

7-Jul-2020: Work on merging 2 triangles completed. Work on filling mosaic pieces partially completed. Work on controlling the number of pieces in mosaic begins.

8-Jul-2020: Work on adding webcam to application finished.

9-Jul-2020: Work on controlling the number of pieces of webcam completed.

10-Jul-2020: Work on integrating the project begins

11-Jul-2020: Work on documentation begins.

13-Jul-2020: Integration and bug fixes go on simultaneously.

14-Jul-2020: Project and documentation completed

Implementation Details

Pre-installations

You need to have the following installed on you PC:

- Python 3.7: link to download
- OpenCV library: link to download
- PIL (Python Imaging Library)
- Tkinter Library

  You can install the library by using the command “pip install name’ in the Command Prompt For Windows and Terminal for Mac

  Start by downloading the Mosaic_Intergration.py from Github Repository:
  https://github.com/diamondgelato/MosaicCreation

  This file can run on code editors like Visual studio code,etc

Flow of Program

- The program starts up with a GUI window.
- Clicking on the ‘Browse’ button opens a file dialog from which the image to be made into a mosaic can be selected and opened.
- Instead of choosing an image, the user can use a picture which they take using the in-built webcam, by clicking on the ‘Webcam’ button and pressing the spacebar to click the image.
- Filters of choice can be applied to the selected image by clicking the ‘Filters’ button and clicking the option for filters in the following window which open.
- Mosaic can be created by clicking the ‘Create Mosaic’ Button. Clicking this will open a new window with two choices for the user.
    - The user can choose to get a random mosaic by clicking on the ‘Create Random Mosaic’ button.
    - The user can choose to get a mosaic with a predefined number of pieces defined by them by clicking on the ‘Create Mosaic Manually’ button and typing in the number of pieces they want in the new window.
- In either case, the mosaic created will be displayed in a new window.
- By clicking on the ‘Save’ button, the user can save the processed mosaic with or without filters. After clicking on the button, a file dialog box will open, asking the user to choose where to save the file and with what name.
- Clicking the ‘Info’ button will display basic information about the application.
- Clicking the ‘Close’ button will close the application.

Functions Used and Brief Descriptions.

def showtext ()

Displays information about the program

def choose_file ()

Opens a file dialog which allows the user to choose the photo they want to convert into a mosaic

def webcam ()

Allows the user to click a picture using the inbuilt webcam, which can be converted to a mosaic.

def filters_list ()

Opens a new window and allows the user to select which filter they want to apply to the image

def black_white (), def filter_hsv (), def filter_plasma (), def filter_ocean ()

Apply a colour filter to the image (still have to see whether mosaic or original)

def getTriangles ()

Takes the number of points (inside the image and on each edge) from def getMosaicPieces () and uses the class Subdiv2D from OpenCV library to get a set of triangles which can be made using those points. This set of triangles is converted to a mosaic in def getMosaicPieces ()

def getMosaicPieces ()

Accepts the picture, number of points (to pass on to def getTriangles ()) and number of pieces required in the final mosaic (this parameter is zero if the user doesn't specify the number of pieces). This function merges the number of triangles (if specified) or arbitrarily merges one-third of the total number of triangles to get random shapes and draws those shapes of the pieces of the mosaic on the picture given to it. Returns the picture with mosaic pieces drawn on it

def getMosaic ()

Accepts the picture and number of pieces required in the mosaic (to pass on to def getMosaicPieces ()). This function sets the configuration of the number of points required for creating the triangles according to the number of pieces input. Passes on the picture returned by def getMosaicPieces.

def create_mosaic ()

Opens a window which allows the user to choose whether they want to make the mosaic with a predefined number of pieces or with a random number of pieces.

def manual ()

Accepts the number of pieces the user wants in their mosaic. Callback for the manual mosaic generation button. Calls getMosaic () to get the image with pieces drawn on and calls contours () which converts the drawn pieces on the image to a mosaic.

def auto_mosaic ()

Creates a mosaic with a randomly chosen number of pieces. Callback for the auto mosaic generation button. Calls getMosaic () to get the image with a random number of pieces drawn on and calls contours () which converts the drawn pieces on the image to a mosaic.

def contours ()

This function is called by the callback of the mosaic generation buttons, manual () and auto_mosaic (). The function draws contours around the individual pieces, finds the centroid of each individual contour/piece and fills that contour/piece with the colour of the centroid.

def save ()

Saves the processed mosaic in the location selected in the file dialog

Problems Faced:

The major hurdles that we faced during the complete development were:

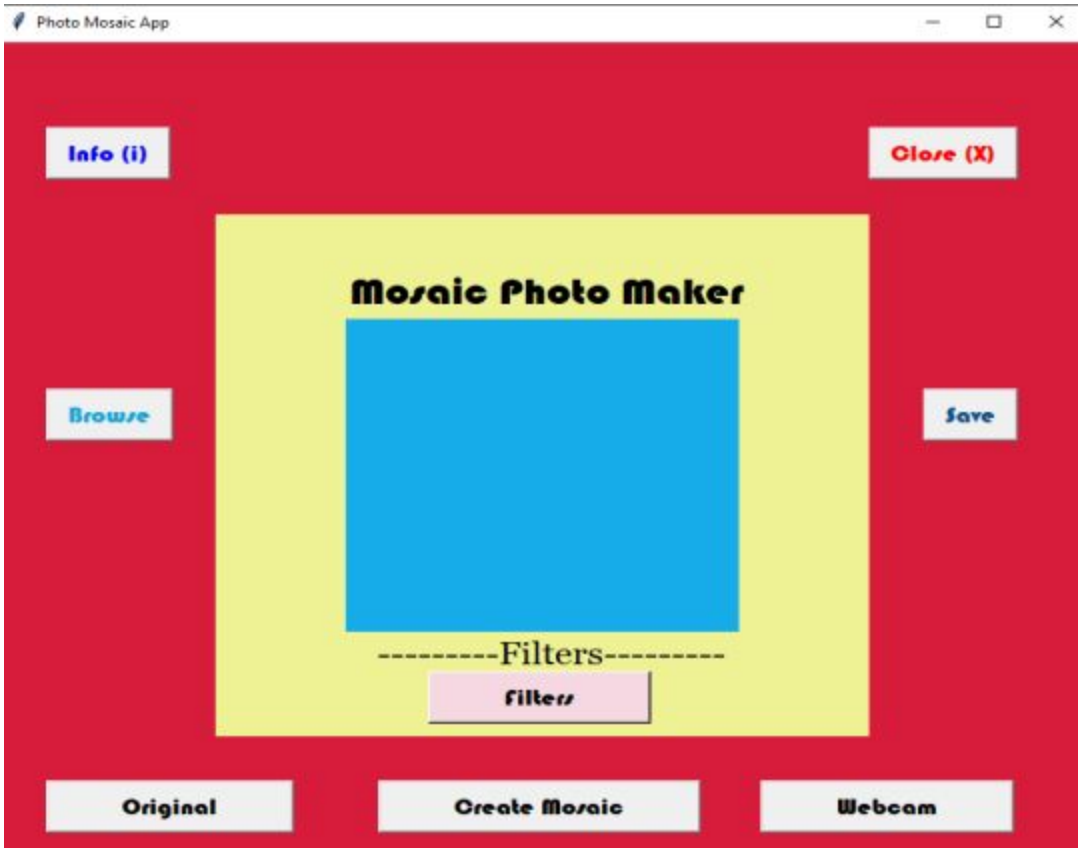
A. **Randomising the Pieces:** It was a tricky task to obtain random pieces throughout the image. In order to solve this problem we came up with the concept which was an extension of the triangulation function (that generated randomised triangles) of combining 2 adjacent triangles together. This merging of triangles came with the catch of we find the adjacent triangles correctly and each triangle doesn't get merged twice and more than 2 triangles do not get merged which was tackled appropriately.

B. **Getting Uniform Color:** We tested various approaches to average the color of the vertices of the shapes and color any one vertex of the triangle but that gave inappropriate results. Finally we settled on the logic of the centroid of the shape to color the entire shape which gave us desirable results.

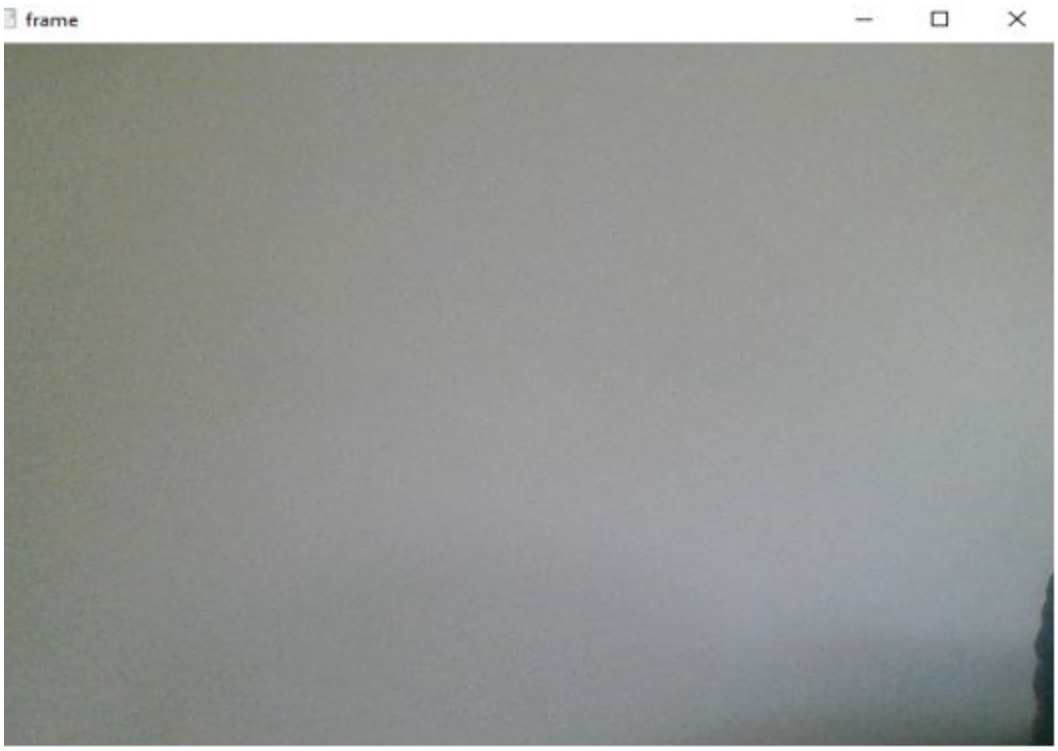
C. **Integration:** Since there were various elements to the code integrating all of these elements together in order to work accurately was a tedious task. But ultimately we successfully integrated all the elements for an accurately functioning application.

Results

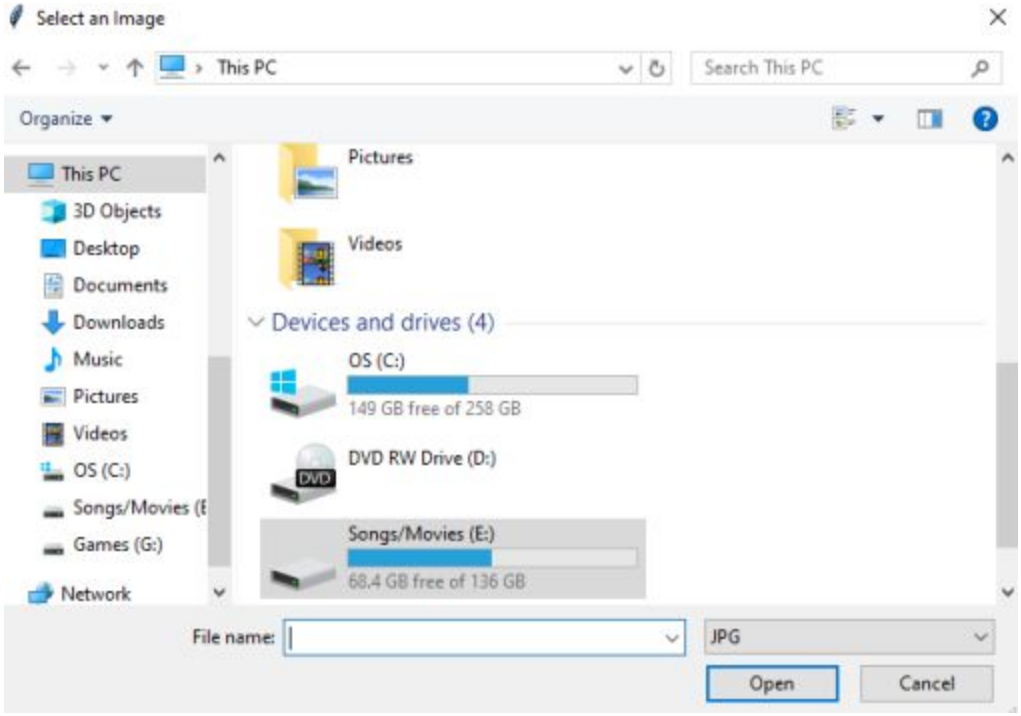
Main Window(Tkinter):



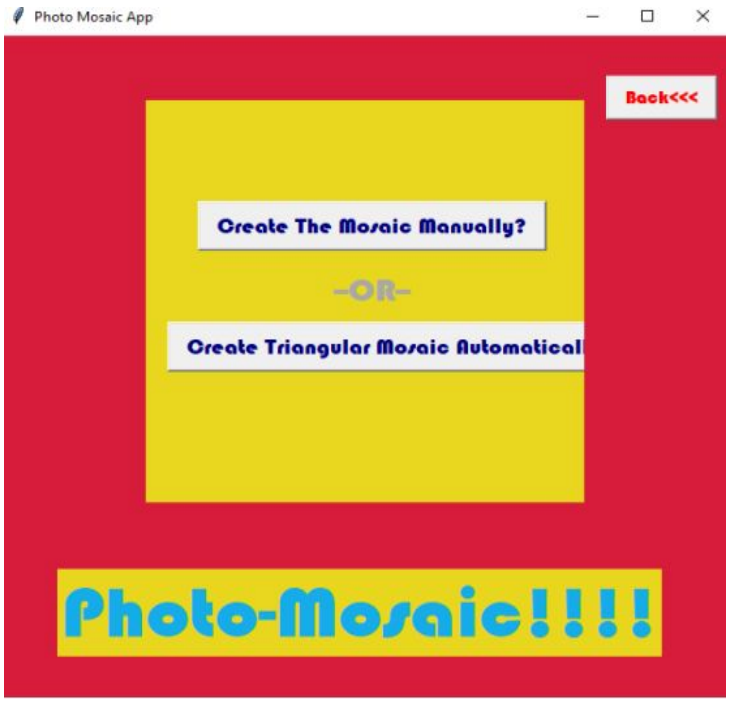
Webcam Button(To open live feed Webcam and take a screenshot):



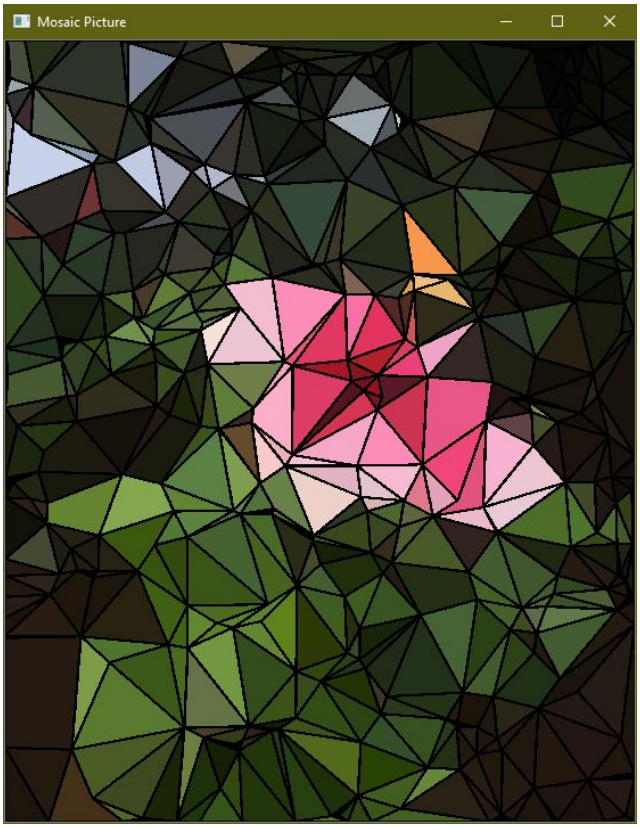
Browse Button(To select file):



Create Mosaic Button(To create mosaic) :



Create The Mosaic Manually Button:

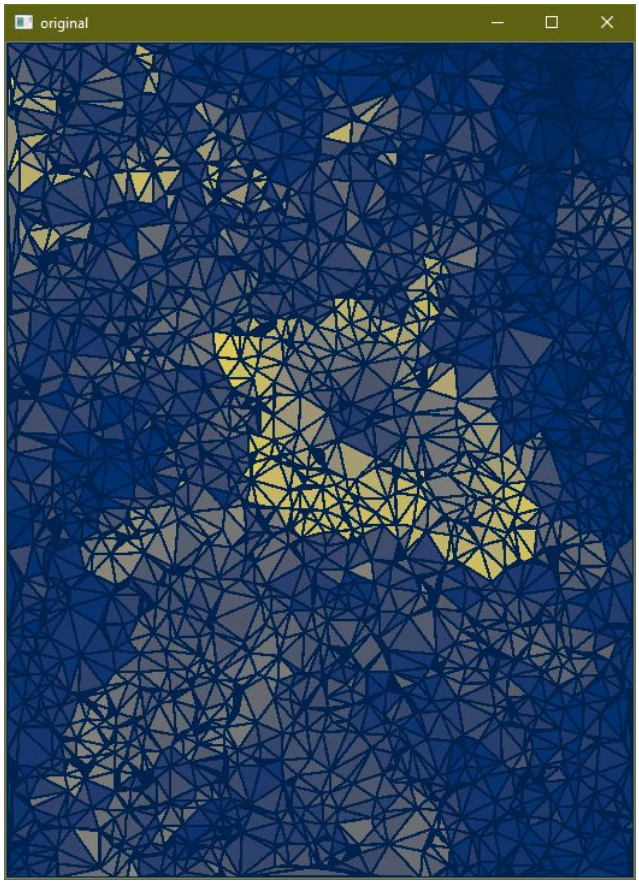


Create The Mosaic Automatically Button:

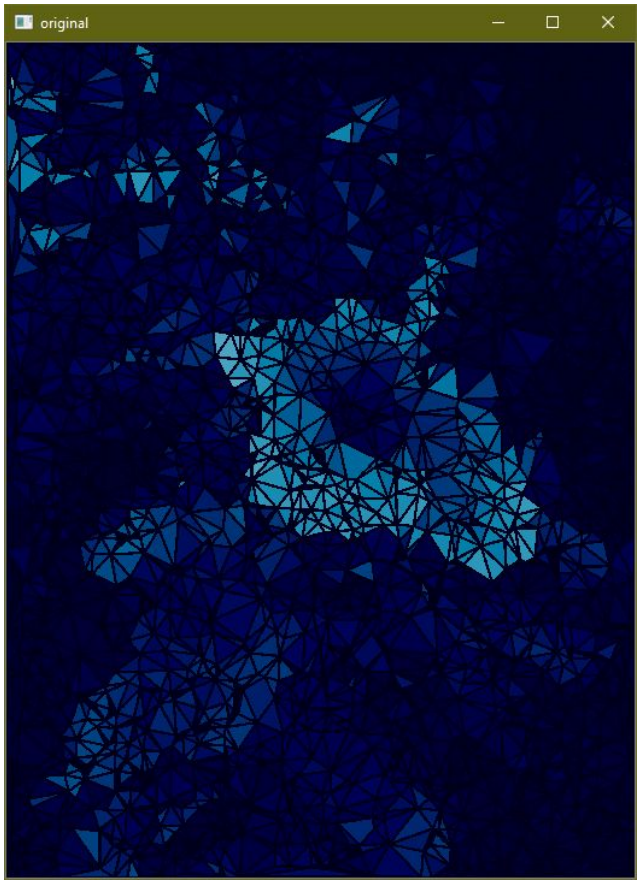
Filters(To apply Filters):

Vintage:

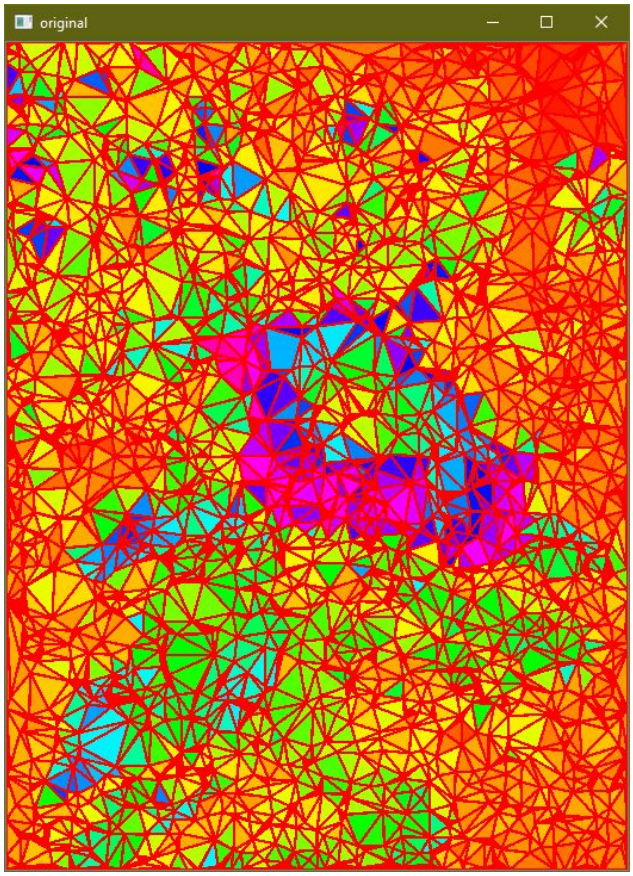
Dim:



Ocean:



Negative:



Roles Played

Mugdha: To create the Triangles, merging them to give irregular shapes and controlling the number of pieces in the mosaic.
Abhishek: To create the Tkinter GUI and apply filters to the image.
Mrunmayee: To draw the contours, apply the webcam feature and integrate the application to work seamlessly.
Dhairya: Finding the centroid and applying colour to each piece of the mosaic.

Conclusion

We have successfully create a Python GUI(Tkinter) App which gives us the mosaic of the given picture(either a selected picture or from the webcam) and it also gives the options to save the image and also to apply various filters