**PhotoSynthesis Key Elements**

**Drawing**

When the user draws on the canvas the onTouchEvent is called **(DrawingView line:216)**. This checks to see which tool is currently active, zoom, draw, or extract:

If zoom is active it will only allow the user to scale the image by calling the zoom event.

If draw is active it will make all the buttons on the screen invisible for the duration of the stroke. It will then get the motion of the event and call the functions touch\_start, touch\_move and touch\_up when each is applicable. It is in those touch functions where the path is kept track of. On touch\_up the undo listarrays add the new path.

If extract is active it will get the pixel at the current xy location clicked. It then gets the RGB values of that pixel and sets the brushes color to that value.

**Updating Canvas**

The onDraw function **(DrawingView line:147)** is in charge of making sure the canvas is centered, all the previous strokes appear, and that the current stoke appears. It centers the bitmap by calculating the top left corner’s location using the screen size and bitmaps size. It then sets the bitmap there and redraws the canvas. To make sure the previous paths show while the user is drawing, they get redrawn during the process. Also to make sure the user can see the stroke they are making, it gets constantly drawn as they move across the canvas. (If this was not implemented the user would see no stroke appear until they let up their finger).

**Undo/Redo**

Undo **(DrawingView line:370)**: This function will only execute if the size of the paths listarray is greater than zero. It starts by adding the path, paint, and filter at the current index into the undone versions of each by removing it from their current listarray, then decreases the index by one. Next it checks to see if an image is being used, if so, it will copy the original of that image stored in imageBitmap over what is stored in the canvasBitmap then updated the canvas. Otherwise it will just paint over the canvas white. It then goes through a for loop drawing all the paths, paints stored in the listarrays up till the current index. Lastly it checks for what the filter is at the current index and applies it.

Redo **(DrawingView line:432)**: Same as undo but swaps the places of the listarrays and undone listarrays at the start for adding and removing.

**User Interface**

The user interface comprises of mainly the buttons the user is able to use.

Menu: Clicking the menu button will call the dropDown function **(MainActivity line:144)**. This simply drops the menu or retracts the menu depending on its current orientation.

Brush Tool: Clicking this button will call brushClicked **(MainActiviy line:239)** which creates a dialog that contains the brush options. It checks which current brush you have and displays this by making its image lighter. It then has listeners to see what the user clicks. If the user clicks a brush it will call the changeBrush function in the DrawingView and pass it the new brush number. If the user changes the size it will call changeBrushSize in the DrawingView, passing it the new size.

Color Tool: Clicking this button will call colorClicked **(MainActivity line:332)** which creates a dialog that contains all the color options. There are listeners to see where the user clicks. Clicking on the color wheel will update the current color to the pixel you are clicking. Clicking on any of the sliders (hue, sat., bri.) will get their progress and update the current color. This dialog also has info buttons for if the user does not know what hue, saturation, or brightness are in regards to color. When confirm is clicked the brushes color is updated by calling setColor in the DrawingView while passing it the new color.

Save: Clicking the save button will call saveClicked **(MainActivity line:505)** which creates an alert dialog. This dialog asks if the user wants to saver their edited photo. It then uses the built in MediaStore function to save the image to the user’s gallery.

New: Clicking the new button will call newClicked **(MainActivity line:540)** which creates a dialog which contains the four options the user has for a new image.

The first option Blank Canvas will call newDrawing in the DrawingView, this will just create a blank white canvas.

The second option Upload Photo will call choosePhotoFromGallary **(MainActivity line:953)**. This function creates a new intent that accesses the phones MediaStore.Images allowing the user to select an image form their phone. Upon returning from the gallery activity onActivityResult **(MainActivity line:1006)** is called. We are able to get the data from the gallery for the image we selected, make it a bitmap and pass this bitmap into uploadDrawing in the DrawingView.

The third option Take Photo will call dispatchTakePictureIntent **(MainActivity line:981)**. This function creates a new intent that accesses the camera. In order to get the full size of the image about to be taken a file has to be created, createImageFile **(MainActivity line:964)** is called here. This function simply creates a file name and the directory to store it in along with the suffix (.jpg, etc). It then sets this file into a string mCurrentPhotoPath as an absolute path. It then returns the file to the previous function. Now that we have the file we can get the Uri for that file, lastly we put that Uri into the intent and start the camera activity. Once the user has taken the picture and confirms it we return to the main activity and onActivityResult **(MainActivity line:1006)** is called. We now have to get the file from the string we saved it to. Then we get the bitmap from the Uri of the file and pass that bitmap into uploadDrawing of DrawingView. In order for this to work a file provider has to be added into the android manifest, this provider will have an authority which has to be renamed the same as the path of the project (com.example.photosynthesis). Also for this provider a file path xml has to be created containing a path to save the image.

The fourth option Stock Image will call sockImageGallery **(MainActivity line:590)**. This function creates a side scrolling dialog with the stock images. It then sets listeners for every stock photo that can be selected. When an image is selected creates a bitmap of the drawable resource for that image and saves it into stockbitmap. It then calls setStockImage **(MainActivity line:671)** which gets the screen size and checks to see if the stockbitmap is too large, if it isn’t it will call uploadDrawing in DrawingView. However if the image is to large it will create a bitmap called scaledBitmap and call resizeImage **(MainActivity line:1054)** on stockbitmap. This function will continuously scale down the image until it fits inside of 1920x1090 dimensions, once the image does it returns it as a newBitmap. ScaledBitmap now gets the new Bitmap from the resize function and calls uploadDrawing.

Effects/Filters

Clicking this button calls effectClicked **(MainActivity line:703)** which creates a dialog containing all the effects the user can select from. This dialog has listeners to see which effect gets clicked. When an effect is clicked it will call its corresponding function in the DrawingView (ex. Grayscale calls drawView.grayscale) as well as calling addFilter **(DrawingView line:742)**. This function adds the filter to the listarray so that it is able to be undone. The only effect that is called differently is the RGB filter which calls showPopup **(MainActivity line:811)**.This function just creates a popup menu that displays a list of colors the user can choose from for this filter. It has an on menu item click listener which call onMenuItemClick **(MainActivity line:820)**. The menu item click works the same as the other filters, it calls the RGB filter in DrawingView with the correct color value, then calls addFilter.