# Image Warper - Documentation

## Contents

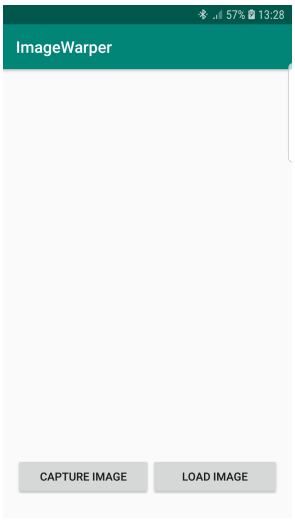
# 1 About

Image Warper - an android application created by our team - allows a simple image modification by a limited number of filters. Application was created and tested on devices with API level 23 and above. Although it should be able to work with the devices between levels 19 an 22, what - due to the lack of stable emulators/devices - was impossible to confirm. In this document, we will provide an in-detail user guide, listing of most important files and their content, as well as a rough guide on how to add another warping methods functionality.

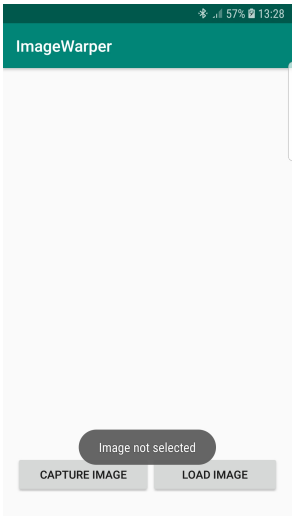
# 2 User Guide

In this part, we will describe briefly the possibilities of our application from the user's perspective.

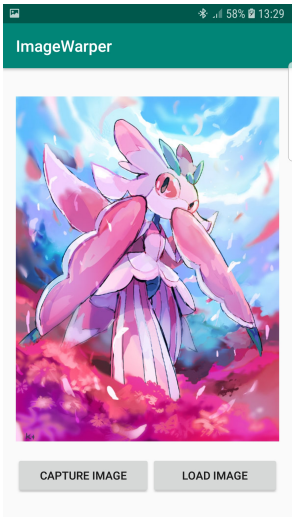
After launch, user is presented with a following view:




At that point. User is given three options. First option is to launch a proper image editor. Which is done by tapping on the center of the screen. Although trying to do that at this point would result in a single error message popping up, as seen below:

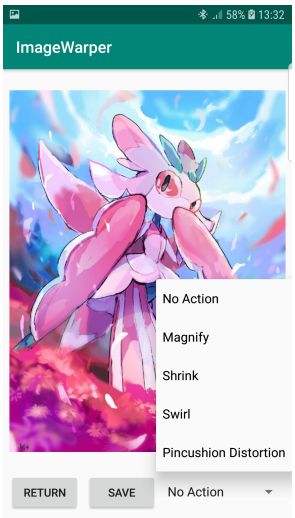


As seen, editor is unable to launch before the picture to edit is selected. For that, user is given two of the options we mentioned above. Those are to click one of the buttons on the bottom of the screen, and either take a picture using a camera (left button), or pick an image from a gallery(right button). If the app is run for a first time. User will get prompted by a message box asking for a proper permission, which is required to complete the following action. Once the file is selected or created, an image will appear on the area above:



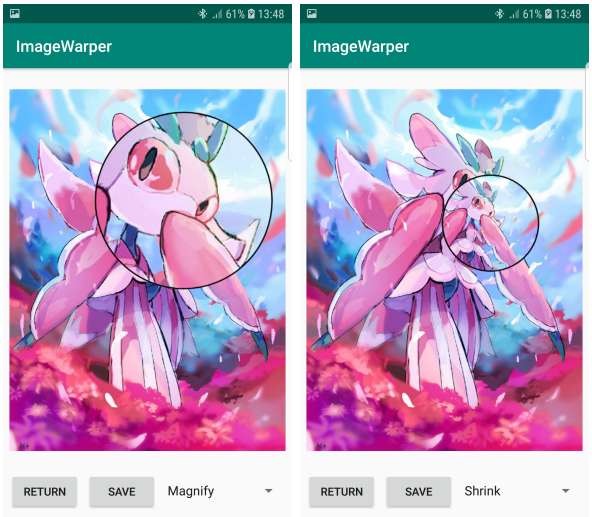

Tapping on the image will allow user to finally start the proper editor. Once it launches, user will see a following view:

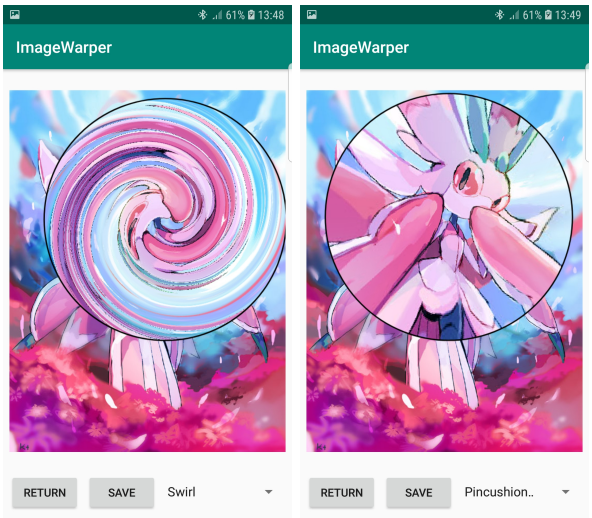

At this point. User is able to do four actions. One is able to either return to the previous screen by tapping on the ”return” button, save currently displayed image to external storage(”save” button) or select the desired warping meethod. If the fird option is chosen. User will be given a list of options, as seen below:




Selecting any option other than ”No Action” will unlock the fourth action available to user. Selecting the ”No Action” option will clear any changes made to a picture instead.

After the proper warping method is selected. User is able to draw a circular area on the screen, controlling its size using two fingers, which will determinge the diameter of the circle centered between them. Selected effect is applied in real time, but depending on the selected method, dimensions of the image and displayed area of the circle changes applied can have a small - to noticably long - delay. Despite that, they will be updated to resemble the current transformation defined by the warping area. Options given to the user will give results simillar to the ones seen below:






Once user is satisfied with the result, one can select the ”save” option we mentioned earlier, by tapping the middle button on the bottom of a screen. That would result in a new file being created in a specific folder on the external storage. User will be notified about the resulting task by a pop-up stating that a certain file was created. Example of which can be seen here


## 3 Application structure

Along the standard gradle build, our application consists of six new files. Two pairs of *.java* and *.xml* files related to the application layout and two *.xml* additional files used to specify field values. These files are as follows:

- ImageEditorActivity.java - consists of all the logic behind editor activity
- MainActivity.java - consists of all the logic behind main activity
- activity_image_editor.xml - specifies layout of an editor activity
- activity_main.xml - specifies layout of a main activity

- colors.xml - resources file for color macros
- strings.xml - resources file for string values

Each file contains variables and/or classes important for the app to work. In sections below we will focus mostly on the ones important for the reader in terms of the next chapter of this documentation.

### 3.1 ImageEditorActivity.java

This file contains the content of a class "ImageEditorActivity". UI related fields:

- returnButton - a reference to a return to main activity button
- saveButton - a referencee to a button allowing to save edited image
- imageDisplay a reference to a view field of the image
- storedPictureUri - Uri value of the image. Required to load
- originalPicture bitmap data of the image before transformations
- displayedPicture bitmap data of the image after most recent transformation
- methodSpinner - a reference to a drop-down menu of the transformation options

Below is the list of fields that are required for the application to work properly, but are not directly related to the UI.

- WRITE_PERMISSION_CODE - id value for writing into storage permissions.
- mShader - shader applied to a transformation matrix
- drawingBoard - canvas object used to apply transformations onto an image
- paint - transformation paint
- cursorOrigin - center point of the applied transformation
- warpingRadius - area of effect
- fileName - holds the name of the original file
- adapter - Implements adapter used by methodSpinner
- cursorScalingFactor - scale factor based on image to screen proportions
- bgTask - background task object
- maskMap bitmap storing applied transformation of a part of an image
- maskMapOrigin - origin point of the applied transformation
- warpingMethodId - Determines the active warping method used by the user.

This file contains various methods. These are as follows:

- onCreate - serves as a constructor of this class

- onRequestPermissionsResult - makes sure the proper permissions are granted by user before executing the action
- saveImage - saves resulting image to external storage
- draw - updates the content of the displayed image

There are also two separate classes. One of which serves as an implementation of an OnTouchListener for the imageView. Other implementing the background thread logic. For the purpose of the guide, that class will be analized in upcoming chapter of this document Those classes are:

- DrawingBoardListener - implements onTouch and getRelativePosition methods for the imageView listener
- BGTask - implements background task

### 3.2 MainActivity.java

This file consists of a single class named MainActivity, which acts as an implementation of the logic behind activity_main.xml. It also implements an OnClickListener for the UI. Important fields of this class are as follows:

- captureImageButton - serves as a reference to a button launching a camera
- openImageButton - a reference to a button opening storage gallery
- imageEditorView - a reference to a imageView field that holds a preview of a selected image/captured photo
- storedPictureUri - file Uri passed by the activity to another to keep track of the selected image/photo
- cameraUri - same as above. But used as a temporary value only for camera intent
- imageSelected - boolean value used to check if user have selected a file to edit, before trying to launch the editor

There are also two static final int fields representing values for permission requests. These are as follows:

- CAMERA_REQUEST_CODE - holds id value for camera permission
- IMAGE_GALLERY_REQUEST_CODE - holds id value for reading from external storage

Lastly, there are nine methods in this class. Although they are self explanatory, we listed them below.

- onCreate - acts as a constructor of this class
- onClick - if any of the UI elements is clicked, proper action is taken within this function
- onRequestPermissionsResult - used in case the given action requires proper runtime permissions that are not yet granted.

- launchImageEditor - launches another activity, responsible for the proper image editor provided by the application
- loadImage - allows user to pick an image from his external storage
- captureImage launches camera intent allowing user to take a picture with it
- getImageFile - if user decided to take a picture using camera, this function creates a new file to store it in memory and provides a reference to it
- displayImage - when the picture is taken/selected, this method is run to update the imageView with a preview of selected image
- onActivityResult - Method responsible for proper reaction to user actions, like displaying selected image by running thee displayImage method, or giving user a proper message regarding the errors that occured.

### 3.3 strings.xml

This file contains every string resource used by the Toast.makeText method, but also string-array called warping_methods that holds the names of the morphing methods given for the user to play with.

## 4 Project's expansion guide

As stated. In this section we will take a look in the code, in effort to expand the functionality of our application. Almost entire task of adding a new warping method can be accomplished by editing a single file. This process can be split up into following steps:

1. Specifying a name for the new warping method - This is done by adding a new item to string-array called "warping_methods" located in *strings.xml* file. This is best to be done as a new line at the end of a list. Preserving alphabetical order would require rewriting a bulk of the app to contain usage of enum values, which were not used in original application due to the slower performance, or painstakingly fixing every usage of warpingMethodId field in the whole file to react properly for fixed values. Once the name is specified, an option to pick it would already be available. However, it would result in no action whatsoever. The rest of the process is done within *ImageEditorActivity.java* file.

2. First, one needs to add another switch case, marked by the new method's id in "doInBackground" method of a BGTask class, in which one would specify how the pixels of a matrix that applies the effect on a picture are defined.

3. With main part of a problem done, another change is required in "draw" method. There. In another switch segment, one would have to add an additional "case id:" statement. Either just above the line starting "case 4:", or if preserving logical order is sought after, one would need to copy this line and paste it above it, only to fix the original one to display the correct id. This is done so every warping method that uses the BGTask class to do its work would run thru this block of instructions, keeping track of the background task. With that, the main functionality of new warping method is preserved.

4. To maintain proper naming convention, a little alteration in ”saveImage” method is required. There, one needs to add an else if statement that checks for a proper warpingMethodId value and changes the string value to display right warping method’s name. Otherwise, pictures altered with it would simply be named with a suffix ”not viable modification”.