FLOATING IMAGE

Android Application

DEREYMEZ Maxime

FUSTES Raphaël

Projet 2016-2017

Index

Floating Image, the current application

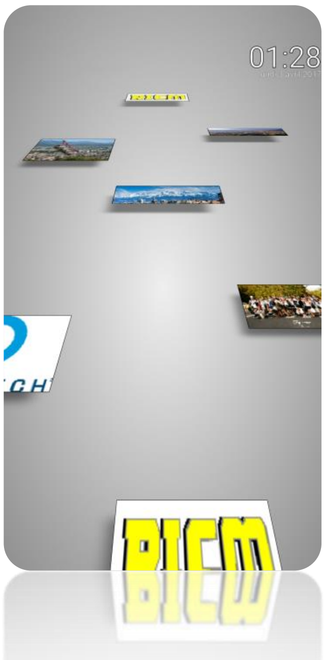
Floating Image is an Android application developed by Mark Gjøl that is available on the Google Play Store. The currently available version is the 3.4.27 version.

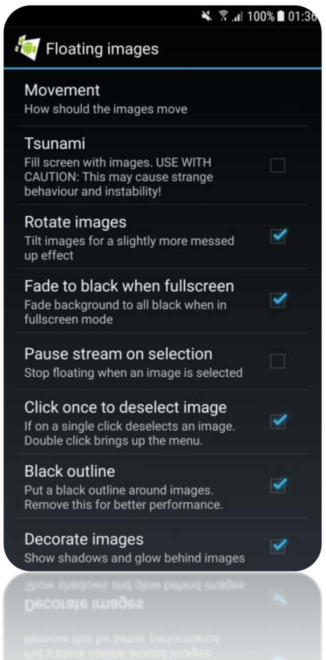
Floating Image is an image gallery application. With it, you can easily transform your Android device into a static or dynamic digital photo frame.

The user can choose which images to display by choosing among different types of sources where the application will retrieve them. The following feeds are available:

- Flickr
- 500px
- Photobucket
- Picasa
- Facebook
- RSS Stream
- Internal Storage

The application offers an in-depth customisation thanks to a versatile settings menu which allows the user to change the image display mode (floating images, slideshow, “hyperspeed” mode …), animations, speed among other settings.





Movement
How should the images move

Tsunami
Fill screen with images. USE WITH CAUTION: This may cause strange behaviour and instability!
☐

Rotate images
Tilt images for a slightly more messed up effect
✓

Fade to black when fullscreen
Fade background to all black when in fullscreen mode
✓

Pause stream on selection
Stop floating when an image is selected
☐

Click once to deselect image
If on a single click deselects an image. Double click brings up the menu.
✓

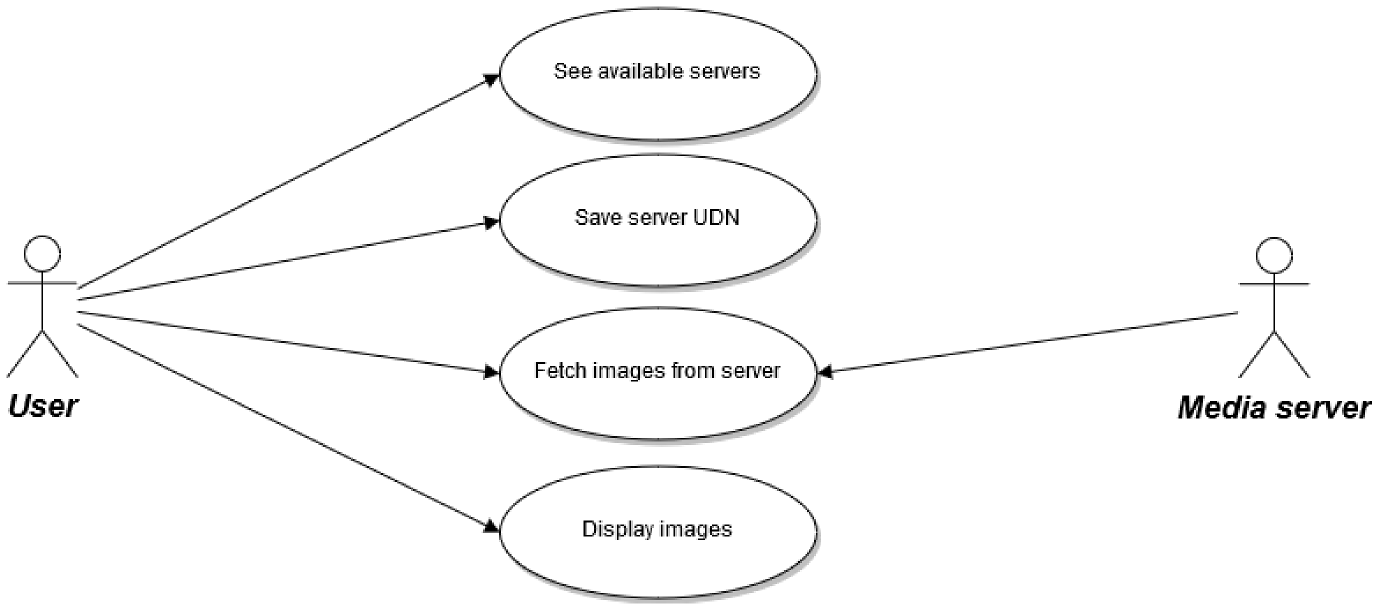
Black outline
Put a black outline around images. Remove this for better performance.
✓

Decorate images
Show shadows and glow behind images
✓

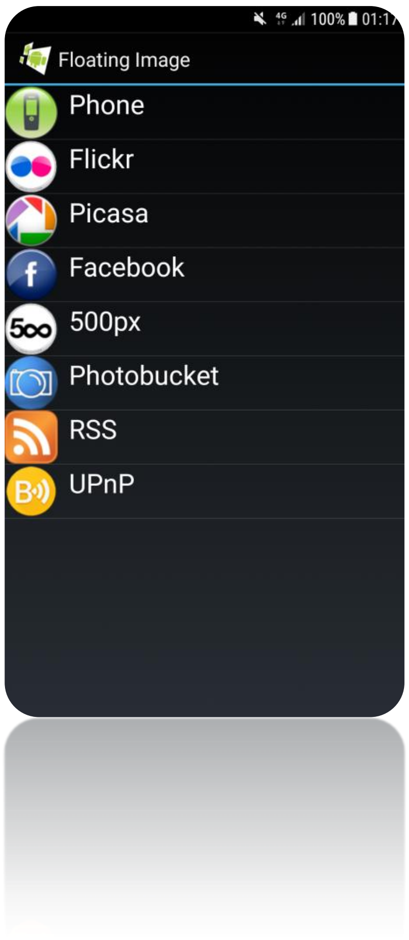
Aim of our project

The aim of our project is to contribute to the development of this application by adding a new function: using a new type of image feed: UPnP feeds.

This new type of image source will allow user to display images located on a remote media server by using the UPnP protocol.
This new feature will allow anyone to use a media server set up on a local network (for example at home), something that could not be done by using the previously available feeds which can only retrieve the phone's pictures or images from social medias/other websites.
In general, this feature works with easy-to-use media servers, like Kodi or even the default Windows media server, which makes it very accessible and require very little set-up time and computer know-how.



The project's realisation



Tools

Different tools were used to accomplish this project.

IDE

- Android Studio to develop the application
- Eclipse to make basic test application

Language / Library

- Android / Java
- Library Cling: *Cling is a library to create UPnP-compatible software stack in Java and Android.*

Hardware

As the UPnP protocol cannot be tested on a Virtual Device, UDP broadcast packets being unavailable; debug and tests were performed by building the application on a connected device through Android Studio.

- Main device used: *Samsung Galaxy S7. Android version: 7.0 Nougat*
- Device also used to test the final app: *Samsung Grand Prime. Android version: 5.1.1 Kitkat*

Software / Server

- Cling Workbench: *A desktop application for browsing UPnP devices and interacting with their services.*

This simple Java application is bundled with the Cling libraries. We used it to discover nearby UPnP device, and display their services. It can also be used to call services' actions remotely. It helped us understand the basic behaviour of UPnP and to debug our application's ability to discover devices.

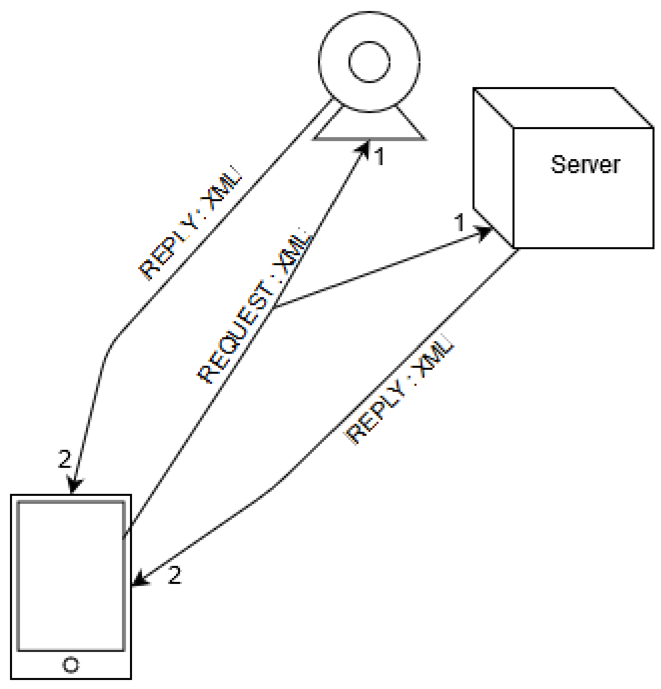
- SimpleDLNA : *A simple, zero-config DLNA media server.*

We mostly used simpleDLNA when we were developing our application. It's an extremely easy-to-use media server which allows to quickly create multiple servers to share folders (containing files like images, videos, musics) on a network. To test our application we shared images folders.
However, towards the end of the project, we had to stop using it as it had general instability problems, and the created servers id were different with every reboot, which meant that they could not be recognised if they were shut down once.

- Kodi (XBMC) : *A free and open-source media center. It can be* used as media player as well as media server

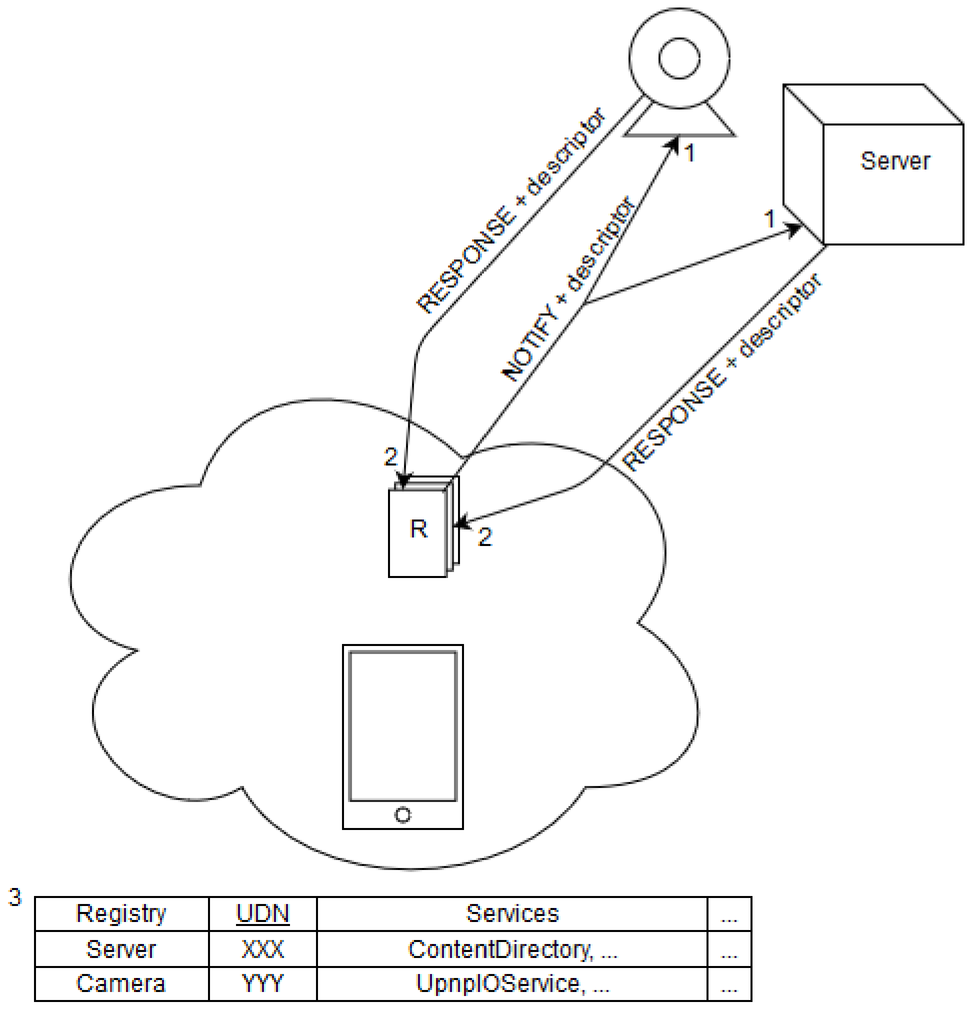
We resolved the problems encountered with simpleDLNA by using another UPnP server software: Kodi. This software is generally used as a media center, but what we used was its ability to share video and music through the UPnP protocol. Sharing video/music works with all types of files, and thus images.

UPnP Protocol



The diagram above shows in a simplified way, how the UPnP protocol works.
The camera represents a camera compatible with UPnP, and the server represents a media center (working with Kodi for example) which is currently sharing files.
The application sends a broadcast message containing its a descriptor in an XML file on the network, and the server and camera will reply by sending descriptors of their own to the phone.
Those descriptors give devices information about what the described device is: its UDN (Unique Device Name), the services it implements.
Every further request will be done in the same way (request in XML file, reply in XML file).

How the UPnP feature works in the application



Registry	UDN	Services	...
Server	XXX	ContentDirectory, ...	...
Camera	YYY	UpnpIOService, ...	...

When the user starts the application, we use the Cling API to create a UPnP service that will be used throughout the rest of the app, and request the descriptor of every connected UPnP device. The main feature of this service is the Registry: whenever the phone will receive the descriptor of a device from the network, it will remember the device and the services it implements.

To add an UPnP feed, the program will search through every device in the Registry for devices implementing the ContentDirectory service. This service is used to share files over UPnP. It will also add a listener to the Registry that will check for every new device, if that device implements ContentDirectory.

The list of those devices is displayed to the user, which can then select the one he wanted. The UDN of this device is then added to the application feed database: it will allow the program to search for that exact same device later.
In the list of every selected feed, the feed's colour helps the user see if the UPnP server is currently available or not: it is checked by searching for that feed's UDN in the devices in the Registry, and in every new device. Green means that it is available, while red means it is not.

The second part of the application is to fetch the images from the remotes devices. For every selected UPnP feed, the app checks for a device with the same UDN in the Registry, and in every new device. When the device has been found, a request is sent to it for the content of the root folder that is shared. The folder is identified by the id "0": it's always the same.
The server responds with an XML containing every file and subfolder in this folder. By parsing it, the application registers the id of every subfolder and the title of every image file (identified by an attribute of value "object.item.imageItem.photo") and the URL where it can be accessed.
The program then sends additional requests for every subfolder, until the whole file tree has been inspected. The list of every image is then returned, and the application can display them.

GlobalUPnPService: manages the Registry and every listener on it, contains a reference to the UPnPService used by the application

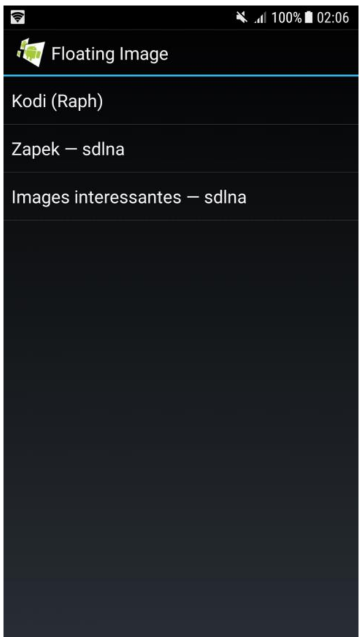
UPnPHandler extends Handler: used by the listeners to change UI elements, changes the color of the feed depending on its availability

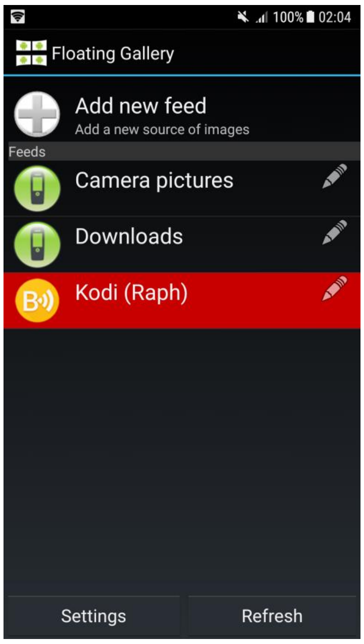
UPnPBrowser extends SourceFragment: displays the list of compatible UPnP servers that are available, allows the selection of one feed to add to the list of selected feeds

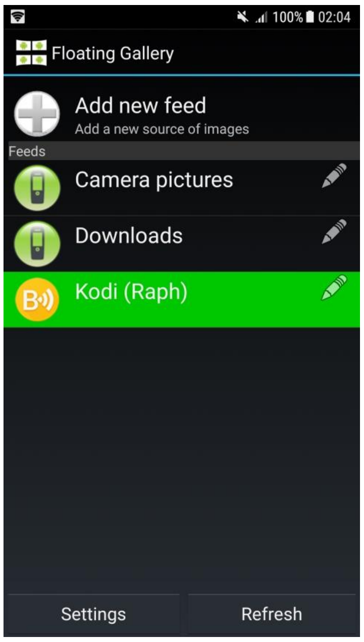
UPnPImage extends ImageReference: similar to other types of ImageReference specific to types of feed

UPnPParser implements FeedParser: adds a listener to the Registry that will trigger for a specific device (using a UDN), and store the device's images in a UPnPImage list
PicturesBatch: called to retrieve images from a UPnP server by sending requests for every folder, and store them in a list

UPnPXMLParserSAX: parses an XML from a UPnP server containing the content of a folder, creates a list of subfolders' id and UPnPImage


100% 02:06
Floating Image
Kodi (Raph)
Zapek – sdlna
Images interessantes – sdlna


100% 02:04
Floating Gallery
Add new feed
Add a new source of images
Feeds
Camera pictures
Downloads
Kodi (Raph)
Settings
Refresh


100% 02:04
Floating Gallery
Add new feed
Add a new source of images
Feeds
Camera pictures
Downloads
Kodi (Raph)
Settings
Refresh