S.A.K-Overlay

Swiss-Army-Knife Overlay

The Original Window Manager for Android

By: Louis Jenkins

Opening Notes

- It should be noted that just because the date shows that this report was created on the day it is due, does not mean that I was lazy with it. In fact, the reason is the opposite. As of now (and most likely when this is turned in today), there is almost 2,250 lines of code, with almost 1,000 lines of documentation, that's explicitly Java code (hence not including /res folder).
- The application is far from complete, however it is also far from unfinished. What I mean is, this application is my embodiment of hard work and, in my opinion, creativity. To give an estimate, the amount of time spent in from of the programming this one project, is at least 300 hours, and going on 400. That doesn't include the multitude amount of hours I spend researching and thinking up ideas/implementations for said ideas, which can easily rival 500 hours. Mainly the time spent not programming, was either doing homework, sleeping, or research.
- The application, while large, is very well documented, and should be very readable. It makes use of third party libraries, which are attached to the project, and hence you should not have any issues running it. I have tested it on the computers in class, and hence works perfectly.
- It requires API level 21, as I did not have time to add conditional compilation. Hence your Nexus tablet MUST be updated to API level 21. This application will not work well in an emulator. JSON serialization requires data storage, and google maps requires the API and of course Location. Also while the application has been optimized to run buttery smooth on the Nexus 7, I cannot guarantee that you will experience degraded performance on an Emulator.
    - If you do not want to upgrade your Nexus 7 tablet due to the slowness Lollipop brings for the Nexus 7, the issue is related to Google Now's indexing of applications. If you go through Google Settings -> Search and Now -> Tablet Search -> Uncheck everything, or at the very least Apps. This brings back the performance of KitKat.

Introduction

- **What is S.A.K-Overlay**
  - A Dynamic Tiling Window Manager
    - Session Manager
      - Restore and Create user sessions
    - Widget Manipulation
      - Snap
        - Allows user to snap widgets to both sides and corners
          - I.E Upper Right, Bottom Left, etc.
      - Maximize & Minimize
      - Move & Resize
        - Allows user to drag Widgets across the screen
        - Allows user to resize the Widget
          - Drag bottom left corner
        - Bounds checking
          - Ensures Widgets always remain in bounds
- **What does S.A.K-Overlay offer**
  - Widgets
    - Sticky-Note
      - Allows user to write notes to be recorded later.
        - Contents maintained across sessions
    - Google Maps
      - Allows user to retrieve their current location
        - Shows user their address
    - Web Browser
      - Allows the user to browse the web
        - Also allows navigation through current history
    - Screen Recorder
      - Allows user to video record their screen as well as external audio
        - Audio optional and Screen resolution adjustable
          - Default to audio enabled and max resolution
      - Allows user to control audio through a "Floating" Widget
        - Visible even when application sent to background.
  - InfoBar
    - Multi-purpose
      - Used for alert, status updates, contextual changes
        - In the future, it will also be used for menu options

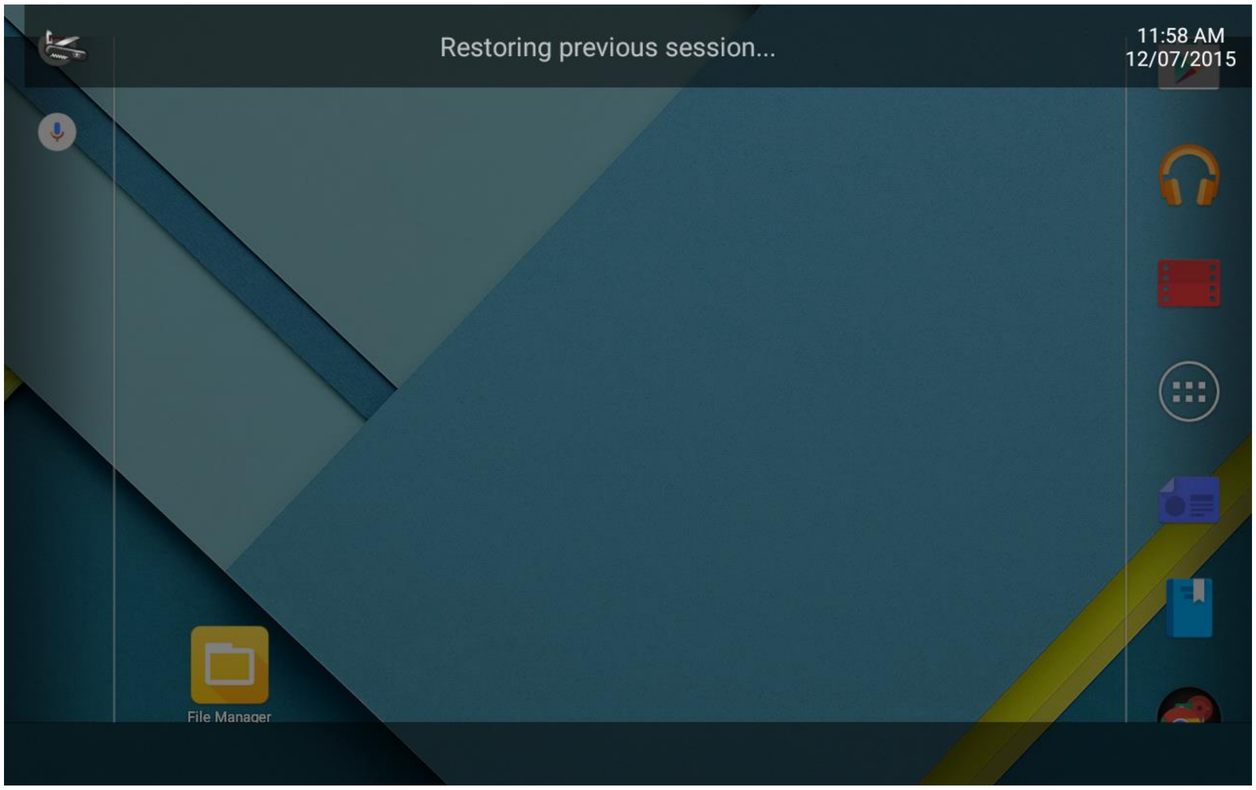
- **Practical uses of S.A.K-Overlay?**
    - Multi-tasking
        - Allows user to have multiple, persistent widgets, aligned however they want
    - Can be launched anywhere
        - A “Home Screen” away from the Home Screen.
    - Designed with gaming in mind
        - Seamlessly go between gaming and S.A.K-Overlay
        - Begin and end recording without ever leaving the game
        - Browse the web while you play
        - Lost? Not anymore with our Google Maps Widget.
        - Need to note something down but no paper nearby? Use Sticky Note
        - Trivia
            - This app was originally meant to emulate Steam’s overlay, allowing the user to answer Text Messages (Like Steam messages), browse the web, etc.
                - Now I plan to do that and more!

- **Significant Features Checklist**
    - Graphics and Animation ☑
        - Note, I count this as an animation, but it's debatable. Technically, we do not use any Animation objects, however we do have quite a few features which could count as animation. In fact, if I had used View.getAnimator().translateX() instead of View.setX(), it would have done the same. Up to you whether you count it.
    - Dynamic UI with Fragments ☑
        - Widgets are placed inside of a custom fragment, called FloatingFragment, which handles its own custom life cycle, deserialization/serialization, setup, cleanup, etc., as well as the manipulation of the Widgets discussed here.
    - Data Storage ☑
        - Sessions are preserved as best they can, whenever onPause() is called and restored in onCreate(), hence allowing persistent apps. Utilizing JSON makes it not only easy to write and parse back out, but also drives home just how powerful the language is.
    - Geolocation and Maps ☑
        - GoogleMapsFragment does use Google Maps, but as of now it is very proof of concept, like all other things, however to be fair, it does have some functionality. Like stated above, it does track the user's current location and show them name of the street/address they are currently at.

User Experience

UI – Sessions

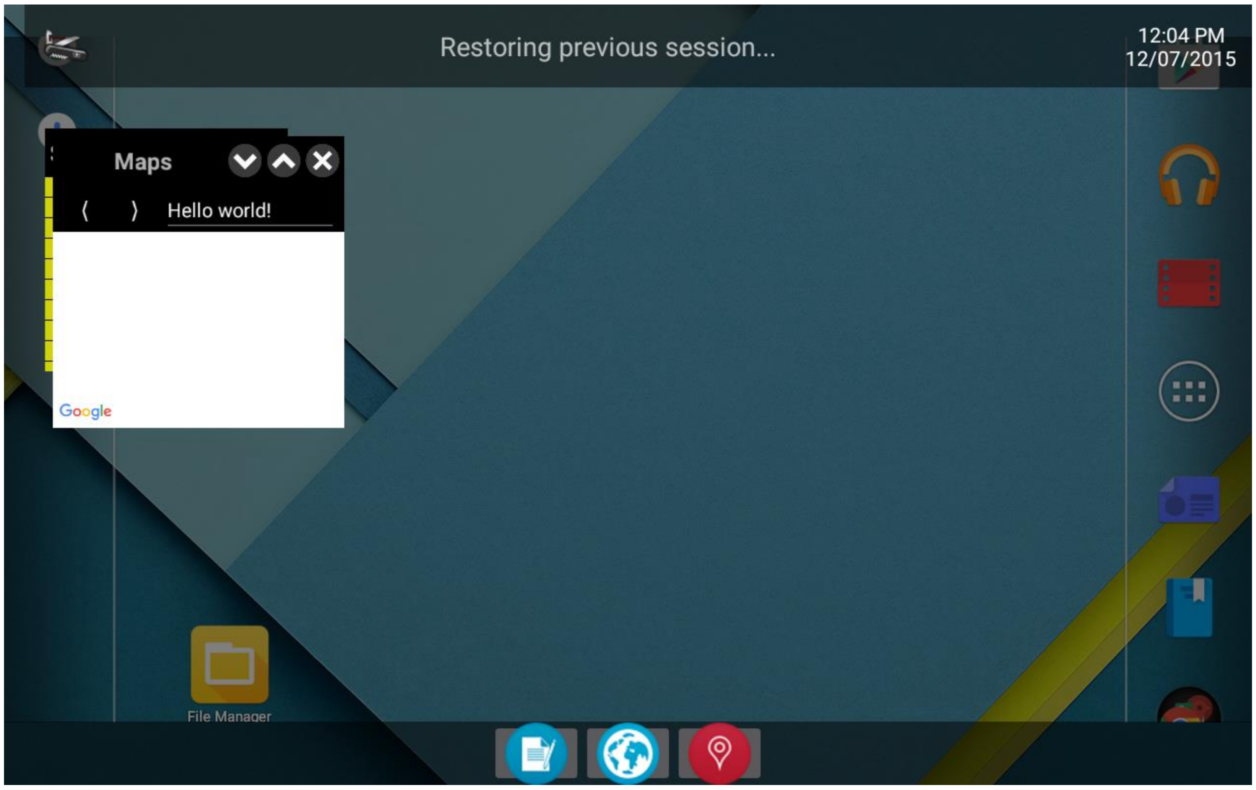
In Process of Restoring Previous Session



This is the main layout of S.A.K-Overlay. Depending on whether or not you have a previous session, it will either let you know it is "Restoring" or "Creating" a session.

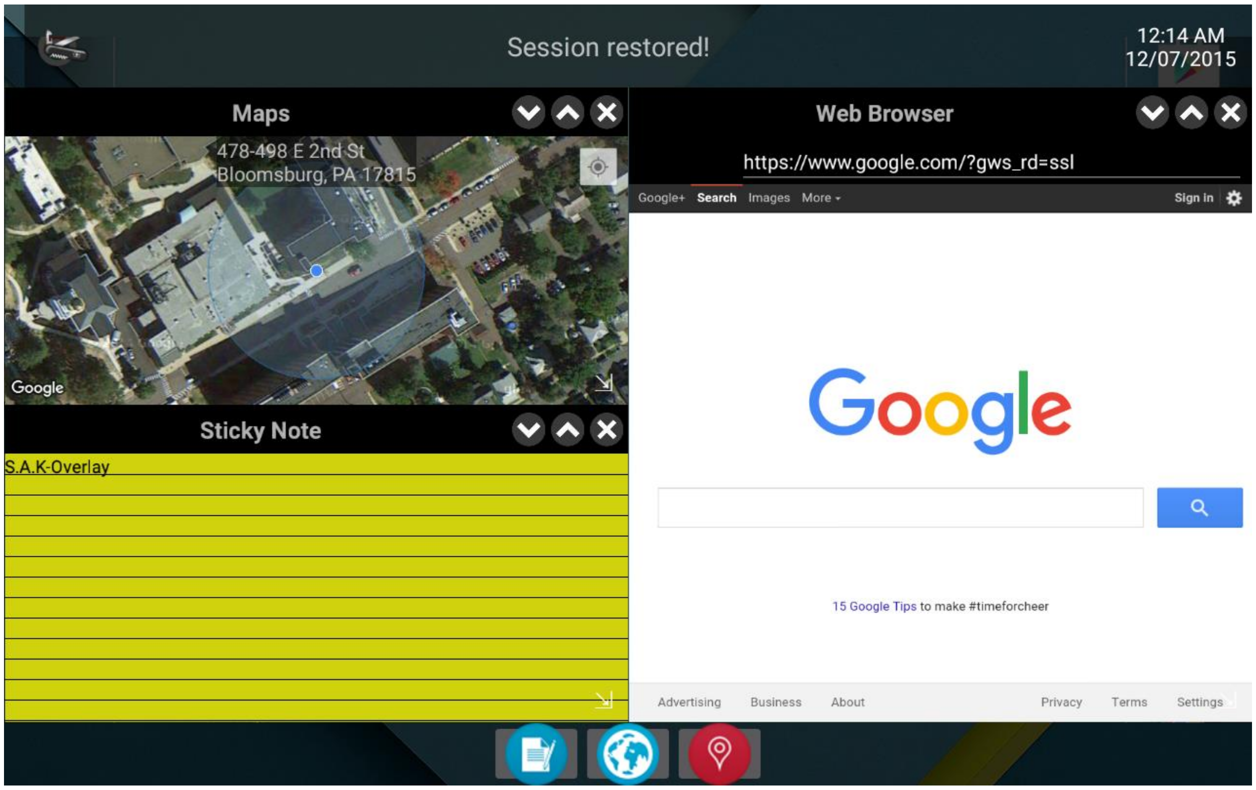
In Process of Restoring State

(This was actually very tricky to do because everything restores so fast, even on a slow device like Nexus 7… I suppose that's a good thing)



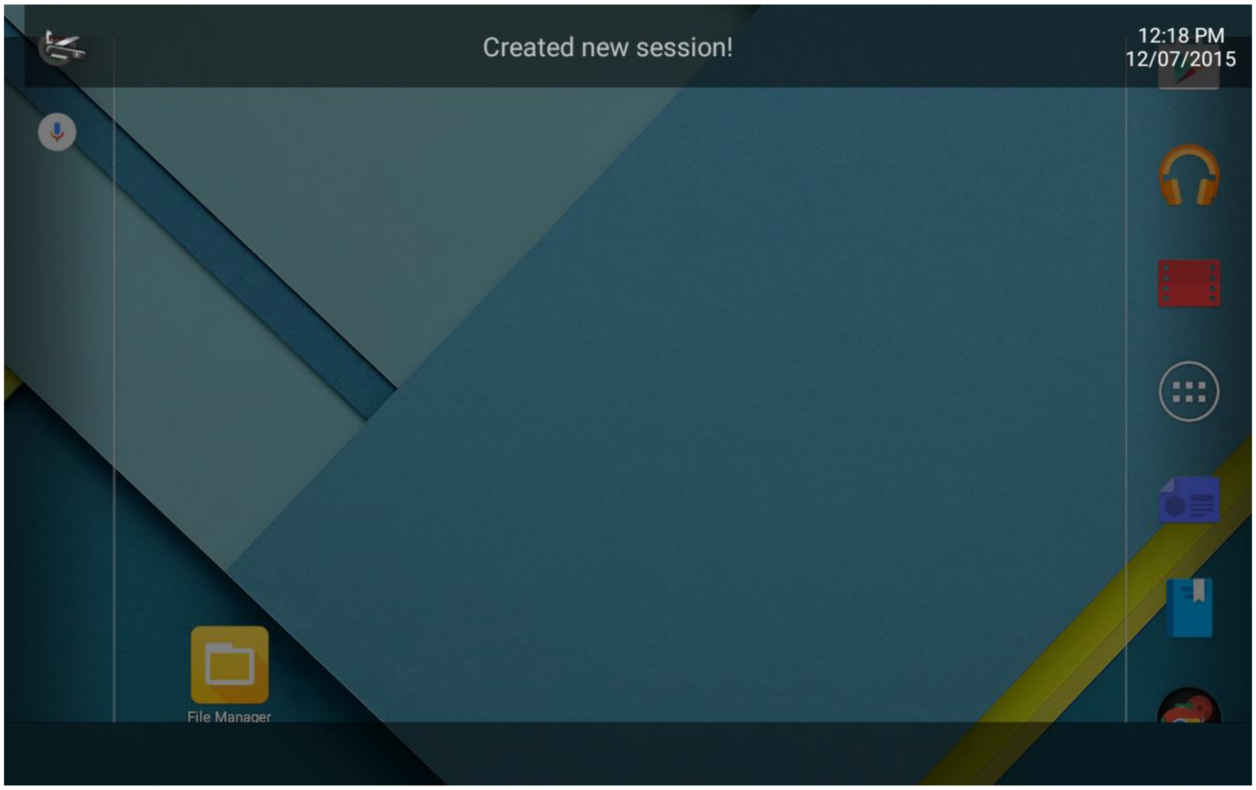
As can be seen here, the Widgets are created, and before they are "restored", they can be seen for a brief moment, like such here.

Finished Restoring Session & State (Extremely fast)


Session restored!
12:14 AM
12/07/2015
Maps
478-498 E 2nd St
Bloomsburg, PA 17815
Google
Web Browser
https://www.google.com/?gws_rd=ssl
Google+ Search Images More
Sign in
Google
15 Google Tips to make #timeforcheer
Advertising Business About Privacy Terms Settings
Sticky Note
S.A.K-Overlay

Notice the InfoBar's "Session restored!", indicating it has finished everything. Also notice the text on the Sticky Note, restored from previous session. As of yet the browser does not restore the web page nor its history, but it will in the future.

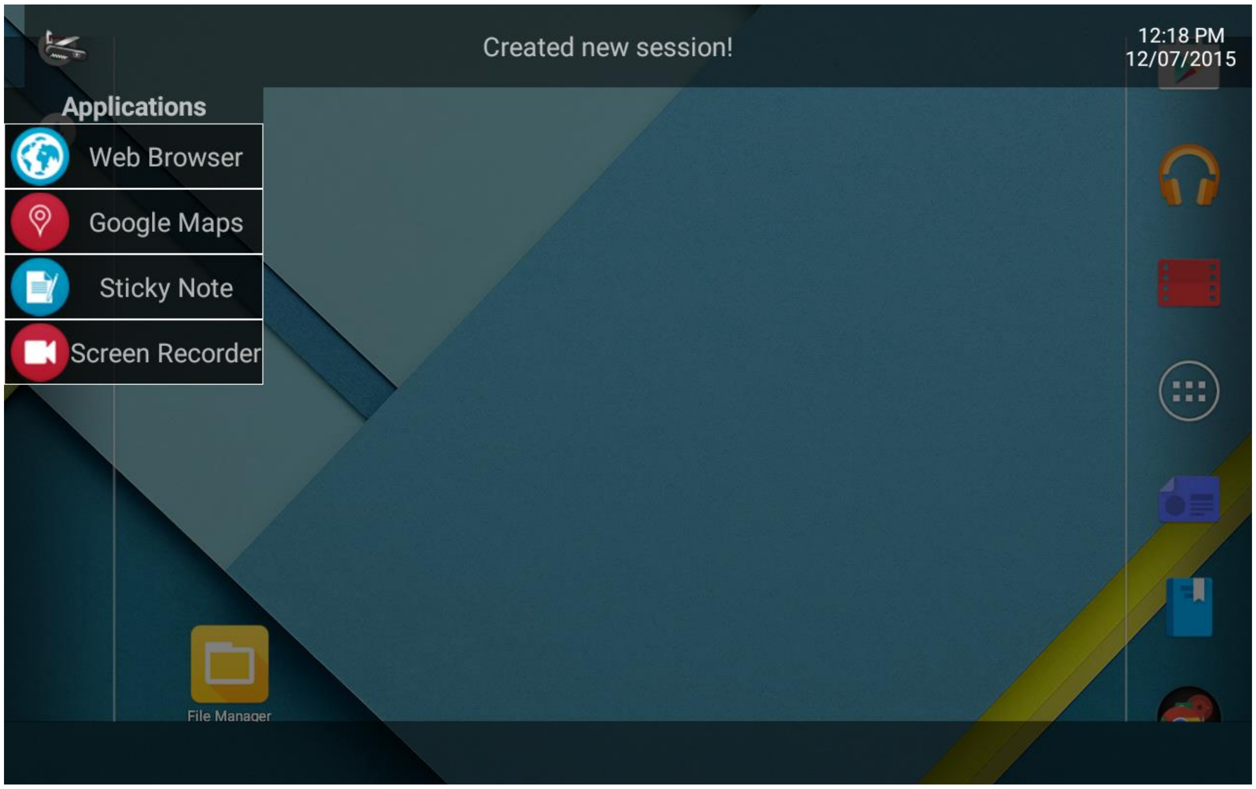
New sessions


Created new session!

If there is no previous session on file or if an error occurs while parsing the file, a new session is created instead.

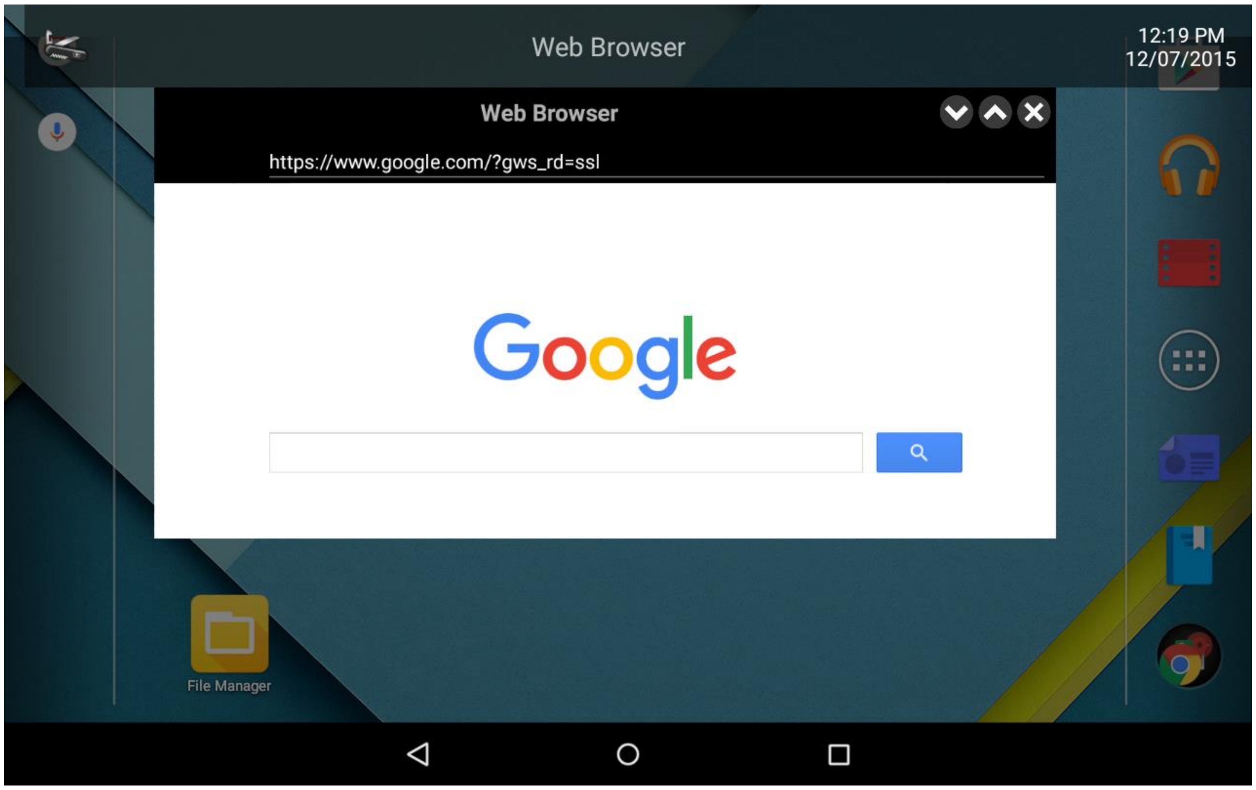
UI – Widgets

Application Menu Dropdown



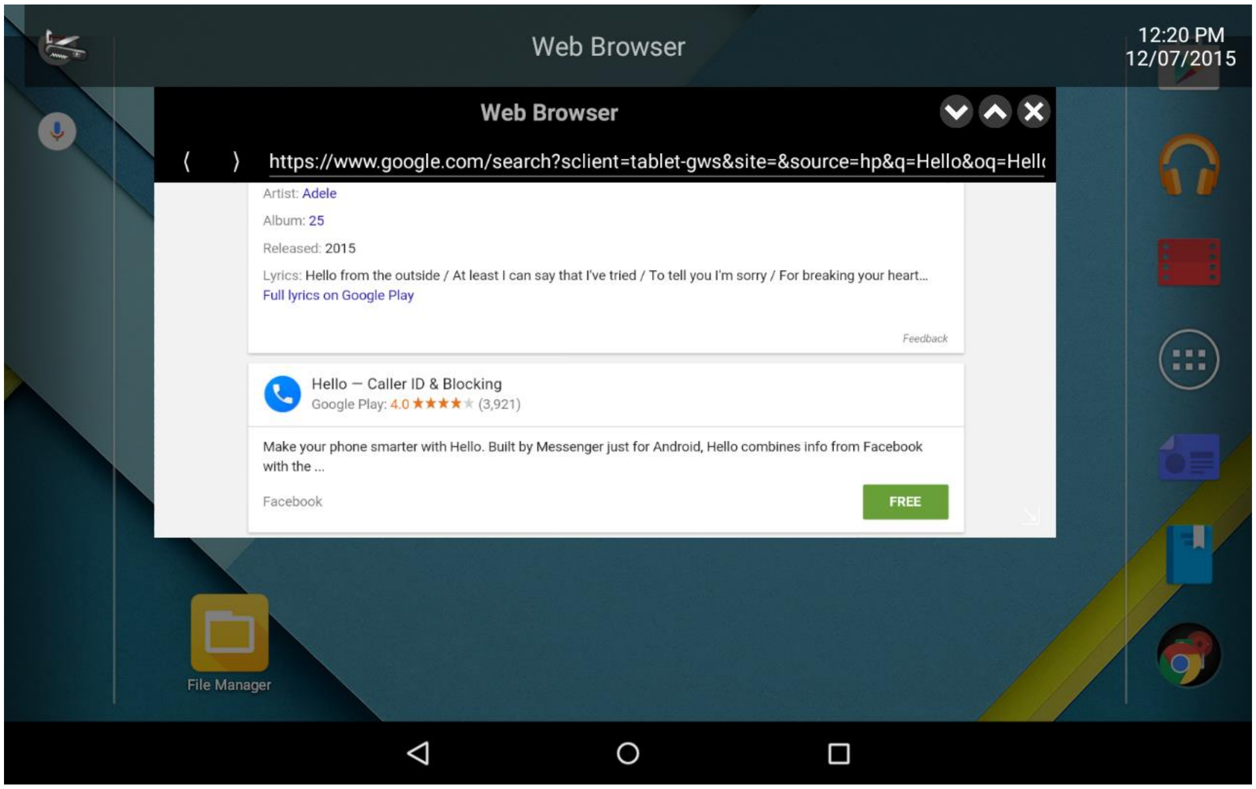
When the grayed app icon is clicked, a dropdown listing all applications is shown. In the future, other options will be added as well, such as “Close”, “New Session”, “Settings”, etc.

Web Browser Home



This is the simple and minimal Web Browser implemented. I'll mention it once, but notice the InfoBar mimics the Widget's Title. This was implemented as a proof of concept for contextual dropdown menu based on the current selected Widget.

Web Browser – History

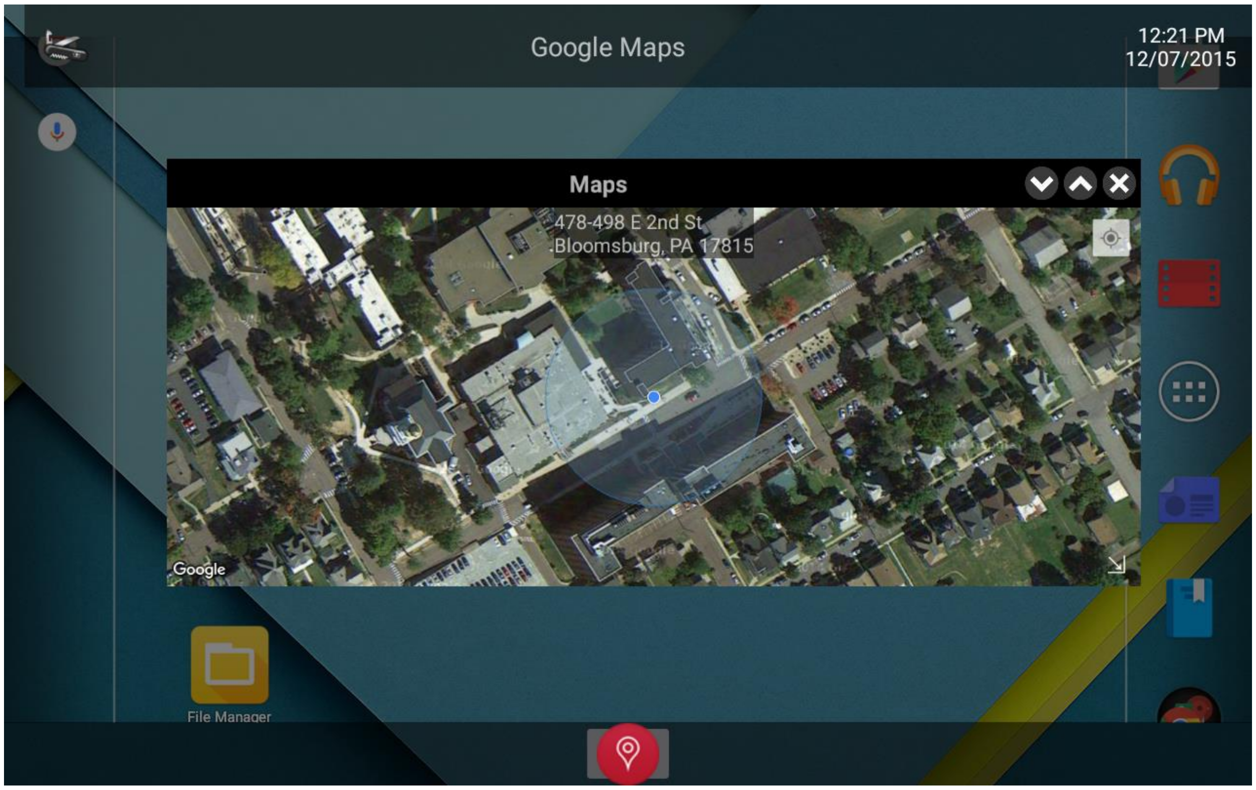

Web Browser
12:20 PM
12/07/2015
Web Browser
https://www.google.com/search?sclient=tablet-gws&site=&source=hp&q=Hello&oq=Hell
File Manager

Artist: Adele
Album: 25
Released: 2015
Lyrics: Hello from the outside / At least I can say that I've tried / To tell you I'm sorry / For breaking your heart...
Full lyrics on Google Play
Feedback

Hello – Caller ID & Blocking
Google Play: 4.0 ★★★★★ (3,921)
Make your phone smarter with Hello. Built by Messenger just for Android, Hello combines info from Facebook with the ...
Facebook
FREE

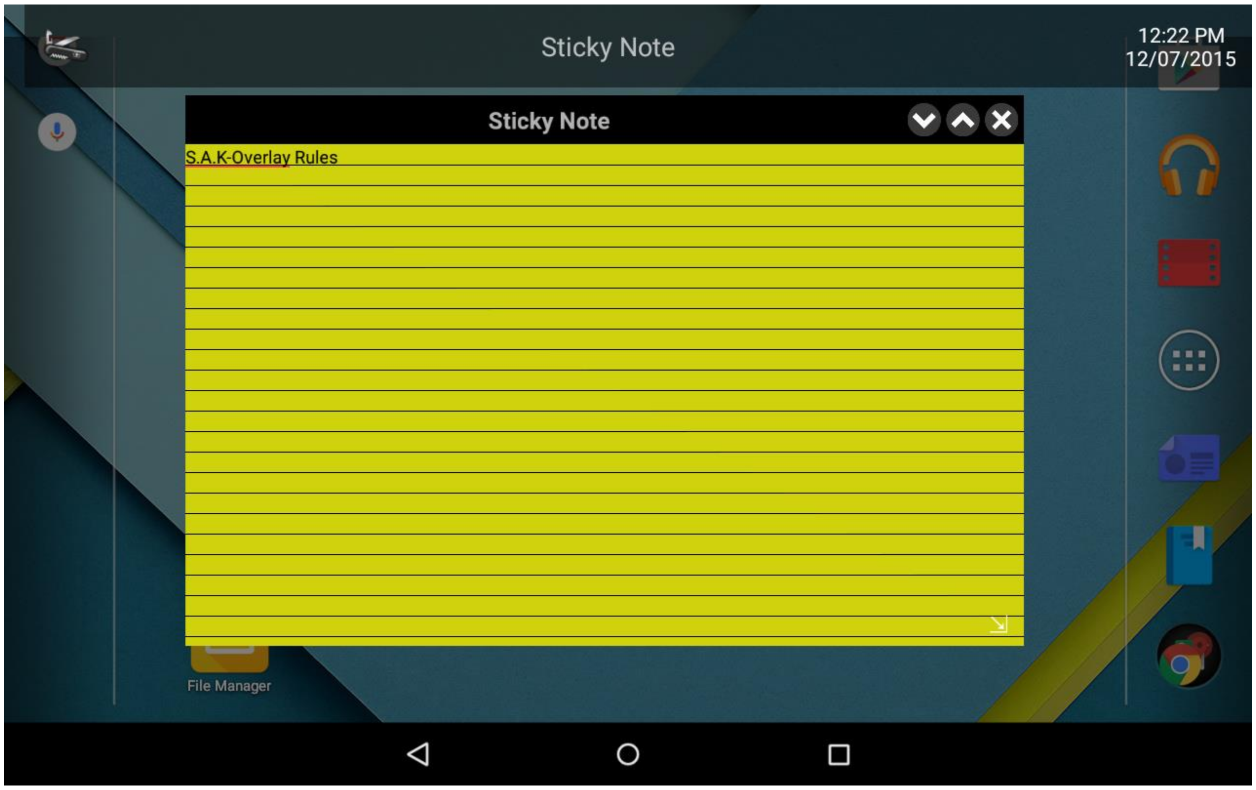
As can be seen, the back and forward buttons do appear based on user's current session history.

Google Maps



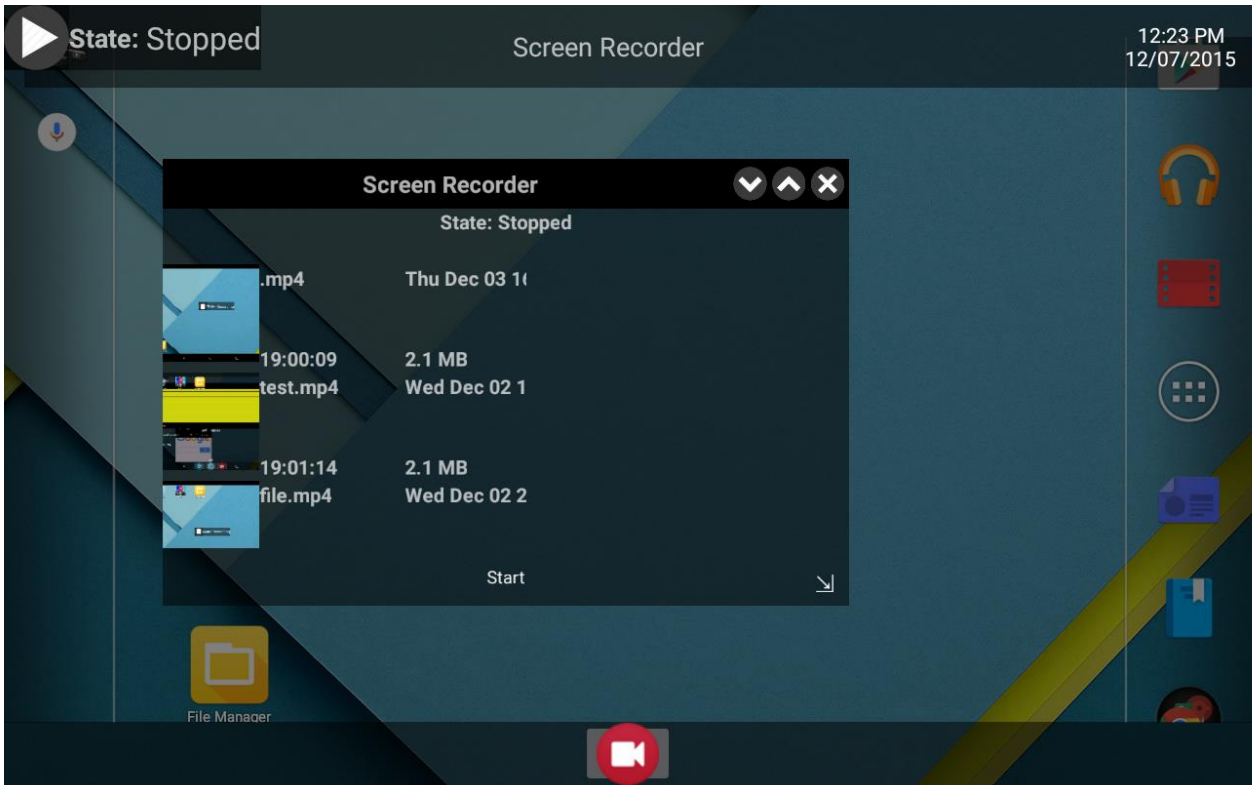
The Google Maps Widget currently only shows the user’s current location and address, but in the future will be able to show nearby stores, gas stations, and points of interest.

Sticky Note



The Sticky Note Widget allows users to jot down their ideas or notes on the go, and persists over sessions. As of yet, it cannot save or load its contents to/from disk, but it's not far off from doing so, especially when contextual menu options are implemented.

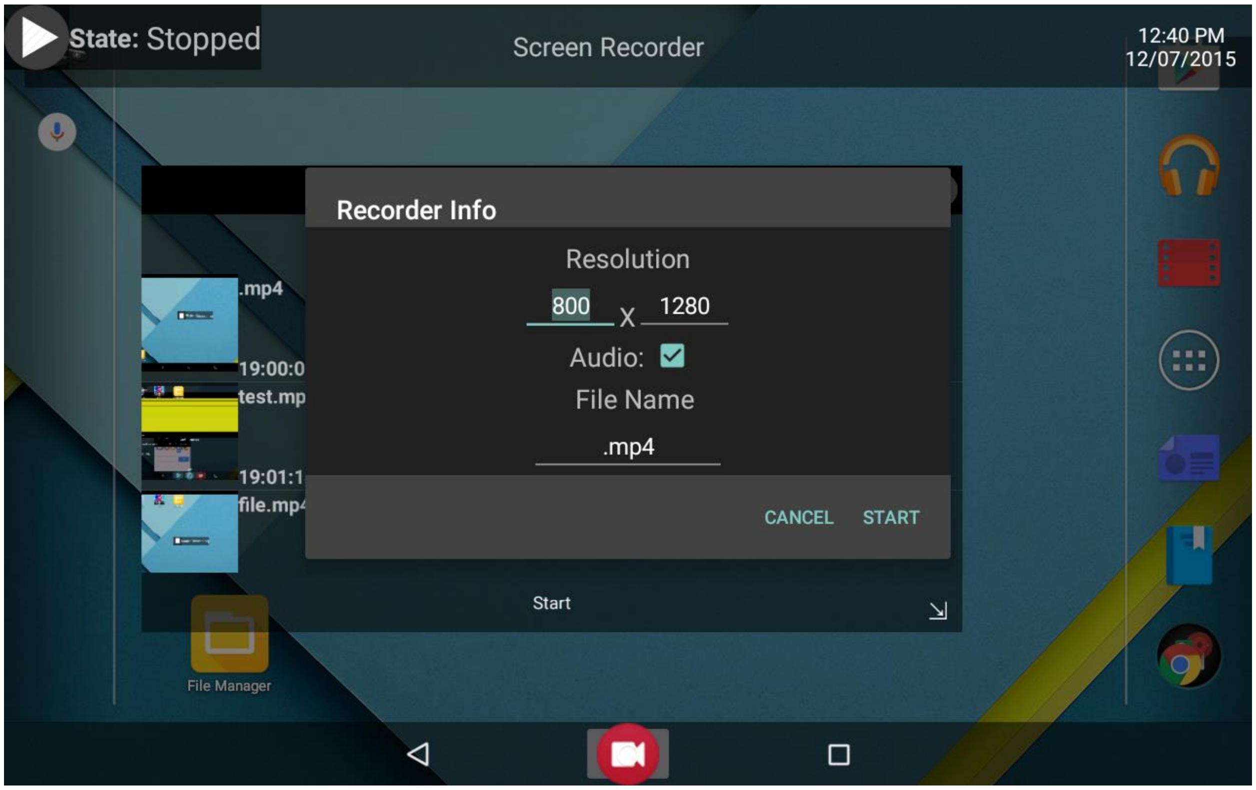
Screen Recorder



.mp4	Thu Dec 03 1(
19:00:09	2.1 MB
test.mp4	Wed Dec 02 1
19:01:14	2.1 MB
file.mp4	Wed Dec 02 2

The Screen Recorder Widget. Unfortunately, the Hour duration is off by 19 hours, however other than that, the minutes and seconds are 100% accurate.

Screen Recorder – Dialog



Recorder Info

Resolution

800 X 1280

Audio: ✓

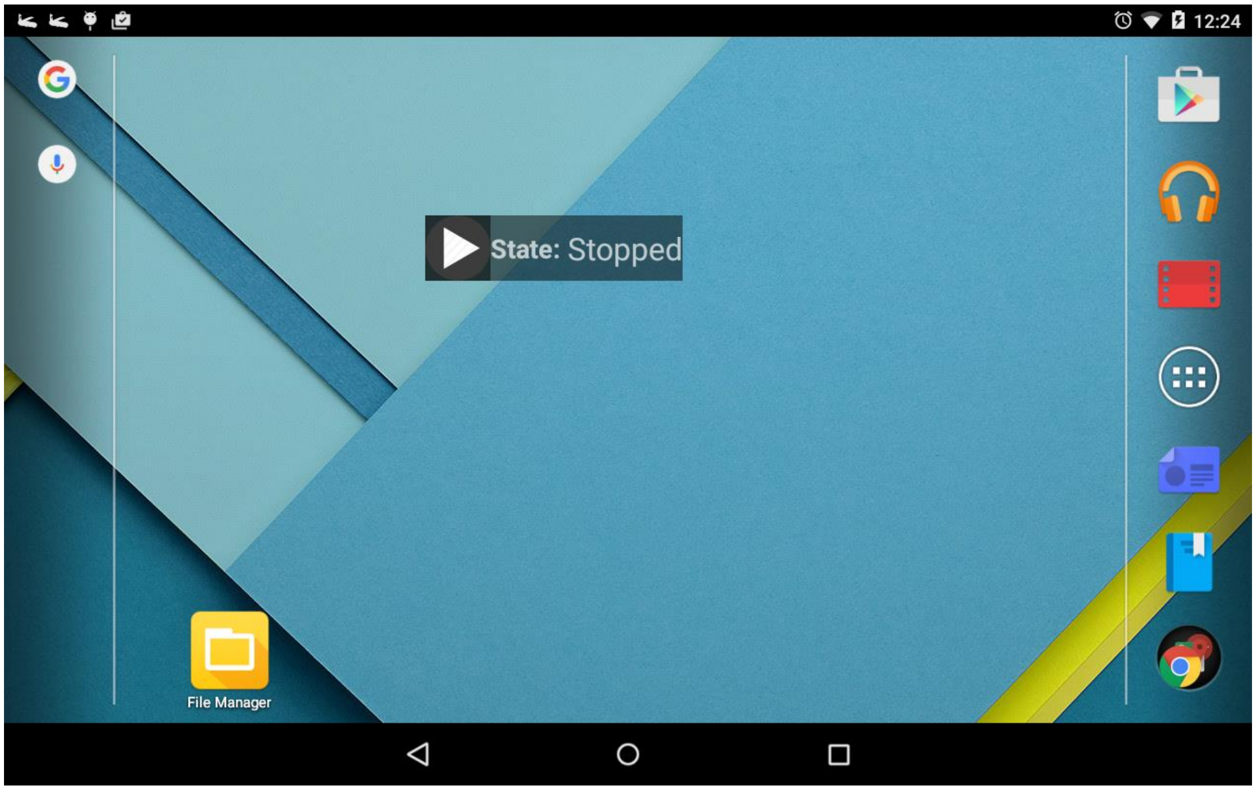
File Name

.mp4

CANCEL START

The dialog that pops up when the user selects start. By default, the max resolution is chosen, audio is enabled, and .mp4 is given as the extension.

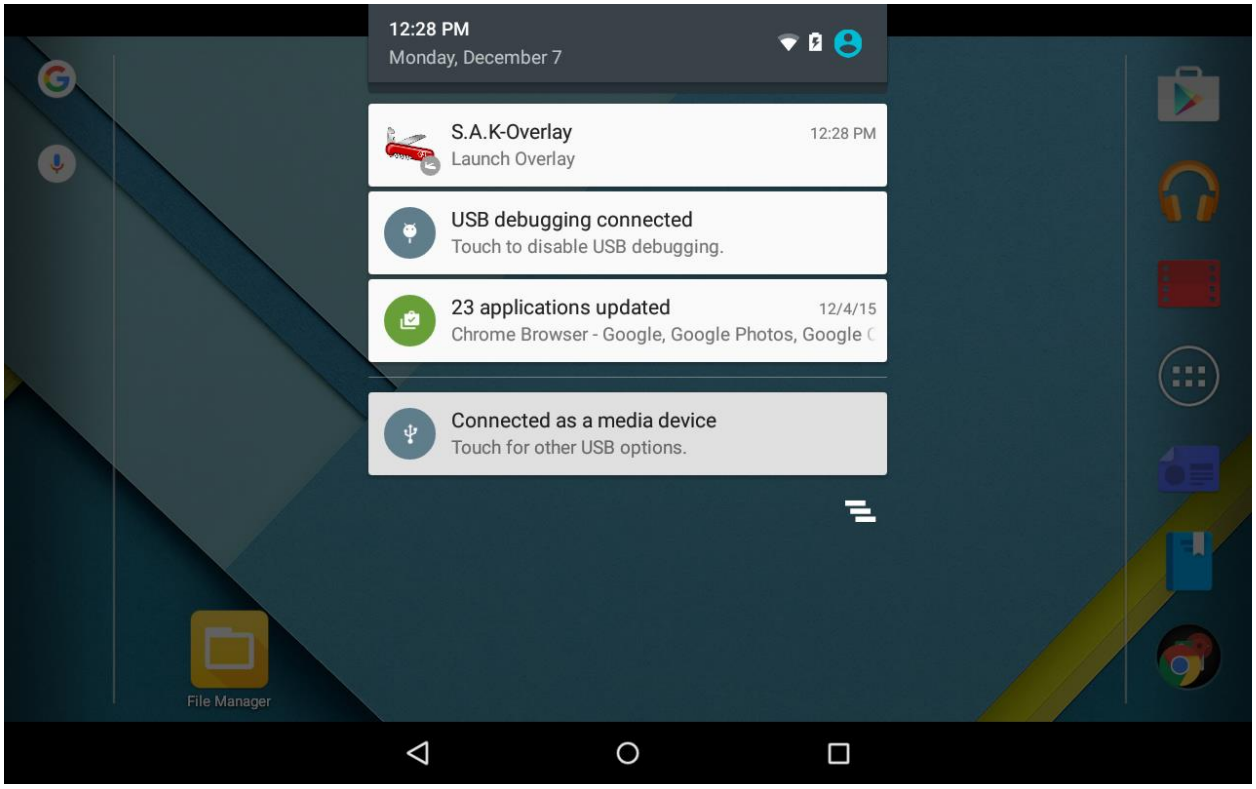
Screen Recorder – Controller



The minimal Screen Recorder Widget's background controller. Its lifetime is tied to the Screen Recorder Widget's, but that doesn't mean it has to be tied to its container as well.

UI – Misc

Notification


12:28 PM
Monday, December 7

S.A.K-Overlay 12:28 PM
Launch Overlay

USB debugging connected
Touch to disable USB debugging.

23 applications updated 12/4/15
Chrome Browser - Google, Google Photos, Google

Connected as a media device
Touch for other USB options.

Offers a simple notification that lets you launch the overlay from wherever you are.

Implementation

Note

The implementations are summarized below, however the details can easily be found in the documentation of the project.

- Screen Size issue
    - A big issue with Android, but also its greatest, is the fragmentation of devices.
        - According to OpenSignal, there are over 24,000 unique Android devices
            - Each easily able to vary in screen size and resolution
    - Attempted solutions
        - Use weighted views
            - Does not work since Widgets are dynamically resizable
                - I.E, the size of the title bar should remain consistent
                    - If the screen size is reduced in size enough, the title bar will be impossible to use
        - Directly scale the view based on the desired size
            - Means having to resize each child a second time whenever user resized the Widget
                - Too slow
        - Scale down views based on the "perfect" screen size with setScale*
            - Best solution so far
                - But has a lot of consequences
            - Each view is scaled inside of the original width and height
                - Means the reported height and width are wrong
                    - Hence touch events need to be adjusted accordingly
                - Leads to complicated math and code, but can be dealt with
- Persistence Problem
    - When the user shuts off their phone or ends the application themselves, onSaveInstance Bundle gets destroyed.
        - Very bad for UX
    - Attempted solutions
        - Use Parcelable or Serializable interface
            - They are not designed for persistence, merely for IPC
        - Directly serialize the view ourselves
            - JSON
            - Best solution so far, but has complications

            - Each element is serialized and deserialized to/from Stringified format
                - Not all things can be made to String
            - Future solution
                - Use a configuration file to describe how it should be parsed
                    - Using ClassLoaders and reflection

- Widget bounding
    - With Widgets being moved by the user, it is extremely easy for it to go out of bounds, even when an Orientation Change occurs.
    - Attempted Solutions
        - Check whenever user moves the view or resizes, that it does not go out of bounds
            - Scaled views make this difficult, as the actual width and height are inaccurate.
                - Still do-able with extra math and logic
            - Current solution so far, open to suggestions
- Screen Recording in background
    - While the screen recorder does continue in the background, the user would have to reopen the app. No matter how "Seamless" I make it, it is unnecessary
    - Attempted Solutions
        - Notification
            - Can allow me to add buttons
                - Not implemented yet, but having to pull down Notification bar to stop it may cause user to lose focus of what they are doing.
        - WindowManager controller
            - Add a view that acts as controller to the Window Manager
                - Drawn on top of any other apps
                    - Future issues
                        - Starting with Android Marshmallow, things like these will require a specific permission which cannot be asked for, and needs the user to go to Settings menu and enable. Not reliable and can annoy the user, spoiling UX.
                    - Current solution
- Coupling issue
    - Communicating through multiple difference classes can be difficult, especially when threading is involved. It goes without saying, when every object knows

each other, removing that object can cause problems, even with Android Studio's refactoring, this leads to overly complicated code, I'll give an example

- Intents
  - Impossible to send objects, even locally, which cannot be serialized or parceled.
  - No way of knowing if the message is received.
    - And sending/receiving requests/responses is extremely messy and complicated.
  - Need a key for each extra, meaning another global constant
    - Also need to instantiate another object just to send locally.
- LocalBroadcastManager
  - Requires a context to retrieve
    - Pretty much forced to pass the Context from an Activity or Fragment
      - Can leak the entire Activity if threaded and runs even after the Activity has died
        - Can be remedied with WeakReferencing
    - Overly complicated and generalized
  - Requires an Intent to send each time

- Solution
  - RxBus
    - Implemented event bus built on top of RxJava
    - Can register to receive events
      - Meaning objects of a certain class
        - I.E, registering for String.class, means you intercept any String objects sent
    - Can send objects directly as events
      - Even non serializable/parcelable ones
    - Anonymous events
      - Classes that sends events do not need to know who sent what and to where.
      - Static context
        - Can be called anywhere, from any class, any thread, etc.
        - Thread safe through RxJava

Bugs

Notes

While some of these bugs are my own fault and are being ironed out (and would have been given more time), there is one pressing bug which I have no control over, and definitely should be mentioned.

- Hardware/Native OS Bug
  - In ScreenRecorder, sometimes when calling MediaRecorder.reset() and/or MediaRecorder.stop(), a race condition for the FrameBuffer's mutex
    - "118-118/? E/gralloc : GetBufferLock timed out for thread 118 buffer 0x18 usage 0x200 LockState 1"
    - Causes device to become complete unresponsive unless restarted
- Software Bug
  - After moving from maximized or snapped state, the size restores to normal, however the because of this, the width and height are also changed so the original touch offset becomes inaccurate. I have made numerous attempts to fix it, but I have not been able to in time. Other than that, moving and resizing views work fine.
  - If the application is started and if the screen gets turned off, it throws an IllegalStateException because the AsyncTask is still executing and attempts to call FragmentTransaction.commit()
    - This is because you cannot add new fragments while the application is in the background, and unfortunately it does not queue up fragments to be done after app restores to the foreground.
      - Can be remedied in the future by not calling it while onPause is paused by keeping an atomic flag, however, FragmentTransaction.commit() is not an atomic operation, so it may throw the exception anyway.
    - Of course, since serialize() is called in onPause(), it fails to finish it's write and ends up deleting the old file, deleting the user's session data.
    - Can be prevented at least waiting until the FloatingFragments have been deserialized before changing applications or locking the device.

Misc

Notes

This is the only note here. I barely finished this in time, and it may not be as good as I expected it to be, however given the time constraints, I'd say it came out pretty well. Once again, the project is very well documented, and the specifics are explained in the documentations and javadocs. Of course I understand you have a LOT of applications to grade, and this app is most likely the biggest one in class, I just hope you have as much fun using/grading it as I had creating/writing it. It was a very fun semester.

Lastly, I've been updating it on GitHub the entire time as it is a personal project I plan on continuing, and can be seen here. https://github.com/theif519/S.A.K-Overlay