

# Android CapStone\_Stage1 Proposal

**Project:** Random-dot-Stereogram Messenger

**GitHub Username:** yeuchi

**Date:** July 15, 2018

**Author:** Chi T. Yeung

<b>Description</b>	<b>4</b>
<b>Introduction</b>	<b>4</b>
Problem(s) Solved	4
Intended User	4
<b>Features</b>	<b>5</b>
<b>User Interface Mocks</b>	<b>5</b>
Screen 1 Splash + Main page	5
Screen 2 View files	6
Screen 3 Settings	6
Screen 4 - Share options	7
Screen 5 Recipient/Settings Tab page	7
Screen 6 Message text Tab page	8
Screen 7 Message Shape Tab page	8
Screen 8 Preview Tab page	9
Screen 9 Widget	9
<b>Data</b>	<b>10</b>
<b>Key Considerations</b>	<b>10</b>
How will your app handle data persistence?	10
Describe any edge or corner cases in the UX.	11
Describe any libraries you'll be using and share your reasoning for including them.	11
Describe how you will implement Google Play Services or other external services.	12
<b>Next Steps: Required Tasks</b>	<b>12</b>
Task 0: Libraries	12
Task 1: Project Setup	12
Task 2: Implement UI for Each Activity and Fragment	13
Task 3: Physical device	14
Task 4: Toasts	14
<b>References</b>	<b>14</b>

# Description

Compose your 1-2 words (or simple shape) message in a random-dot-stereogram and share it with others via email, facebook or other social media. Make it extra challenging by encoding it in selective colors and use it as a color blindness tester.

## Introduction

Random-dot-stereogram technique was first published by Dr. Julesz<sup>[1]</sup> in 1970s. Since then, Random dot stereogram has experienced surges of popularity. The algorithm is available from GPU Gems - Chapter 41 Real-Time Stereogram<sup>[2]</sup> and I have already tested it in javascript (available in appendix)<sup>[3]</sup>.

In this design proposal, I propose a messenger application with its content, a random-dot-stereogram generated by user key entry and/or image upload (Alpha channel only).

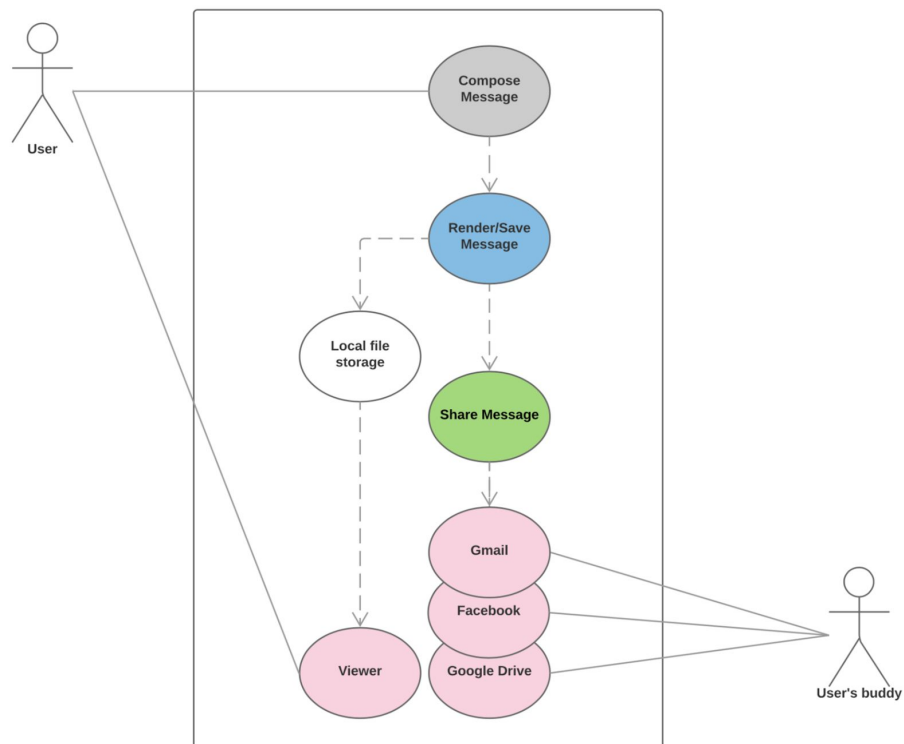


Figure 1. Use case diagram

## Problem(s) Solved

This is a fun messaging application for sending a 1-2 words encoded message in Text + shape graphics. The message is encoded in a random stereogram (auto-stereo-pair VR-View or Single-interlaced image).

## Intended User

- User must have good eyesight and patience.
- User must already have knowledge to cross eyes when viewing single-interlaced image.
- For the selective color feature, users with color-blindness may find some content prohibitive to decode.

## Features

With this application user will be able to use the following features.

- Create 1-2 words (and/or shape) message.
- View random-dot-stereogram with VR Viewer
- View random-dot-stereogram as single-interlaced image.
- Share message via email, facebook, google-drive.

From the perspective as a development requirement list, below list granular capabilities in this application.

- Provide a view with text entry and upload shapes.
- Support Right-To-Left (RTL) for the Arabic language.
- Provide sizing capability on text and shapes.
- Render a random-dot-stereogram into 2 auto-stereo-pairs for VR Viewer.
- Render a random-dot-stereogram into a single-interlaced image for unaided viewing.
- Archive above mentioned rastered files to local or Google drive service.
- Share above mentioned files with 3rd party services such as email, facebook, etc.
- Load/render previously archived files.
- Accessibility with Google voice access service<sup>[6]</sup> or speech recognition service<sup>[5]</sup>.
- Provide a widget for archive listing and quick application launch to viewer.
- Persist application states and configuration between uses as well as phone reboots.
- Support (defeatured) offline usage.

## User Interface Mocks

## Screen 1 Splash + Main page



### Splash

Animating transition of graphic and text introduces user into the main page.

### Menu selections

On the AppToolBar, user can select general settings and shared options. Shared options allow user to compose and send new random-dot-stereogram for destinations such as facebook, email or file. User may also select 'Settings' from menu at anytime.

### View button

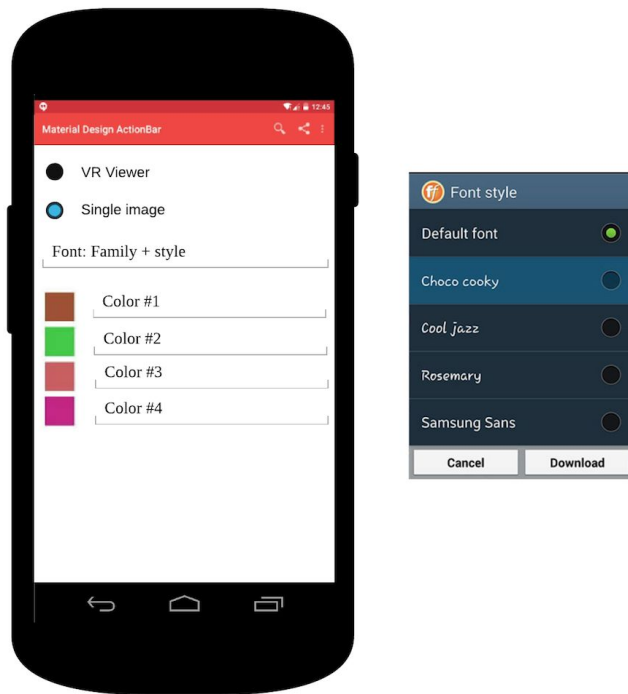
navigate to random-dot-stereogram view.

## Screen 2 View files



User may view rendered random-dot-stereogram on file system that were shared previously. This particular mock-up image depicts an auto-stereo pair for Virtual reality viewer.

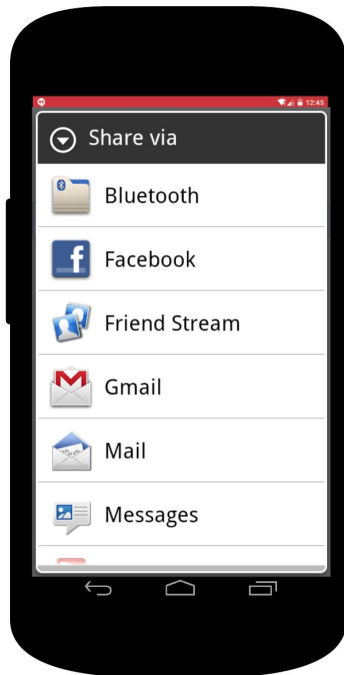
## Screen 3 Settings



At anytime, user can change application settings. These settings include the following:

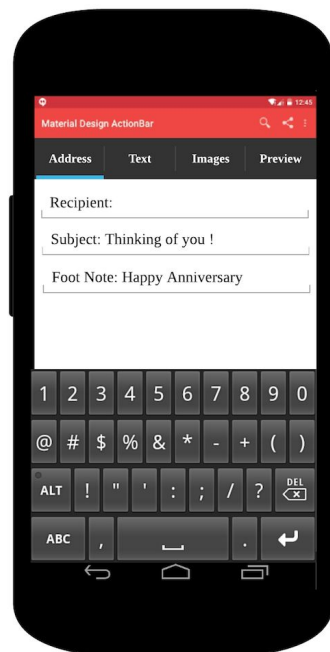
1. Target rendered image type
  - a. auto-stereo image pair for VR viewer.
  - b. single interlaced image.
2. Font type
3. Colors used for the random dot dithering.

## Screen 4 - Share options



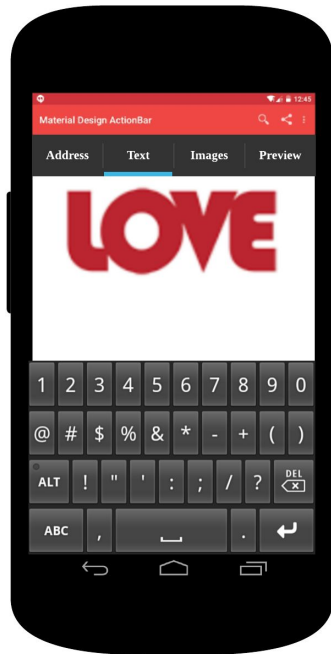
First implementation will include three options: Facebook feed, Email message (HTML body) embedded content and File saved on phone.

## Screen 5 Recipient/Settings Tab page



For the email scenario, user will input standard email header along with a footer text message and viewing image type (VR View - 2 images or Single interlaced stereogram).

## Screen 6 Message text Tab page



Use keyboard entry to develop your 1-2 word(s) message.

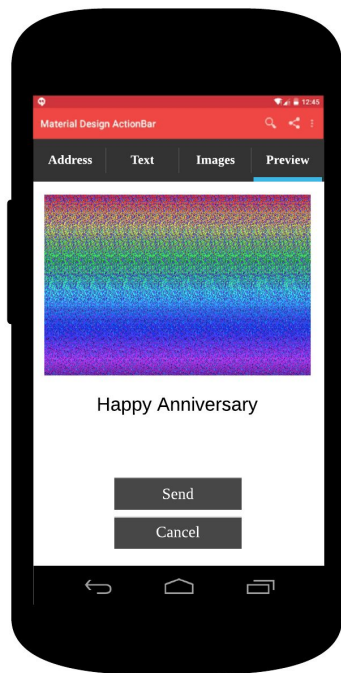
## Screen 7 Message Shape Tab page



Upload your custom shape from file or select existing to accent your expression. Image shape should be simple and binary in nature, 1 bit per pixel alpha channel PNG image.

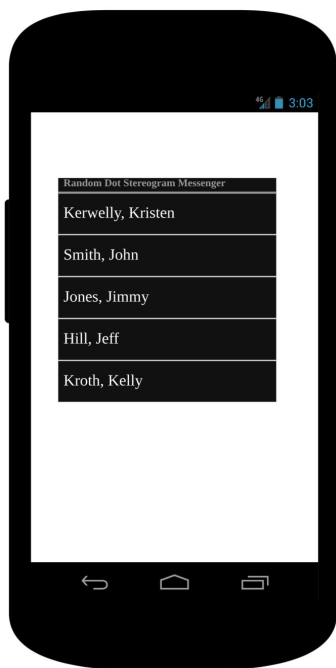


## Screen 8 Preview Tab page



Preview your message in an encoded Random-dot-stereogram. Depress Send button to commit.

## Screen 9 Widget



Send list available on widget. Click selection to launch full application viewer for content.

# Data

Below diagram characterizes the entities to be persisted in a logically separation.

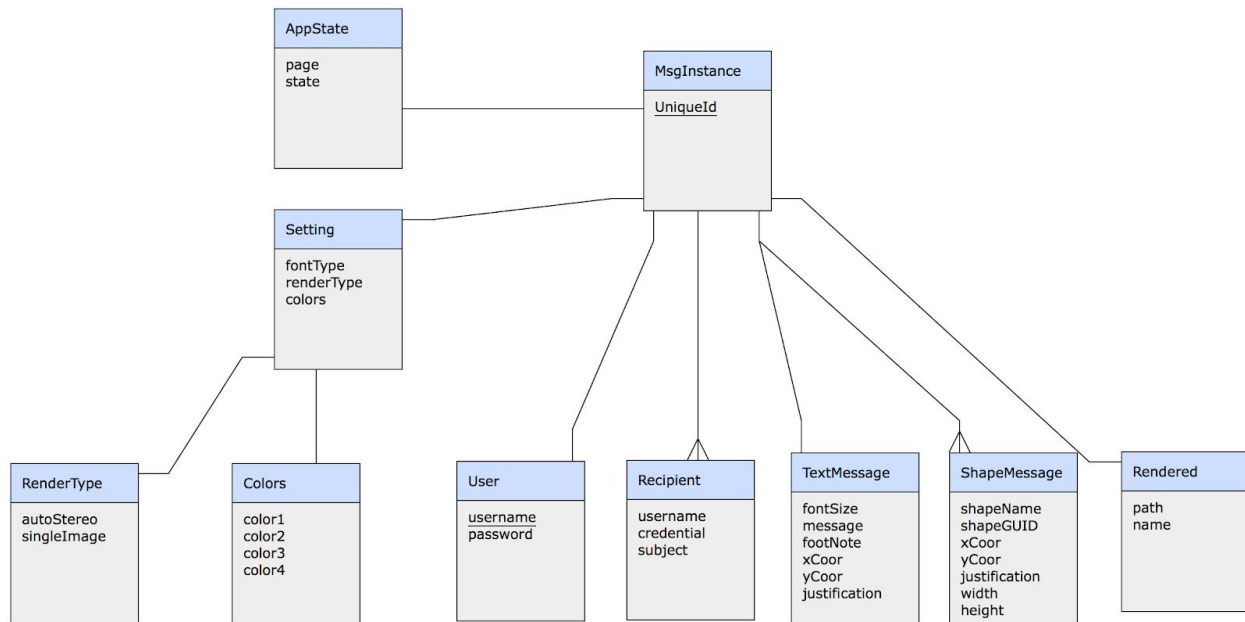


Figure 1 Application Entity diagram <sup>[4]</sup>

## Key Considerations

### How will your app handle data persistence?

#### Random-dot-stereogram (rendered) image files

For this application, I like to store them on local file system.

But in a real production environment, storing these images on Google storage or through a RESTful service would make more sense.

#### Message Instance

A local SQL-Lite with Content Provider make sense to manage all application state, resources and variables for a current message instance. Each instance can be uniquely identified by recipient entity + GUID.

#### Application settings

Font, color and rendering types are specific to the message instance above mentioned. This data is also store in the SQL-Lite database.

#### Credentials

User passwords will be maintained with CredentialOptions object.

## **Describe any edge or corner cases in the UX.**

### Blank message

'Send' button will only be enabled if message content exists.

### Large message

Pending on the viewing device's physical dimension, a long message may have to be clipped in a small window. Scaling down the image may reduce acuity of the text. One possible solution is an animating scroll window.

### Complex Upload shape

Random-dot-stereogram is best when viewing simply and commonly recognizable shapes. Complicated and unfamiliar shapes are not recommended.

Only PNG image format is supported for the upload feature. PNG 1 bit per pixel is preferred and will be treated as alpha channel data. If alpha channel is available, all the pixel color information is ignored. Highly compressed JPG images contain undesirable block artifacts.

### Page transitions - Tab pages

Users are expected to move between tab pages. Application will maintain states and variables in SQL-Lite database for storage. When user completes the workflow, application will return to main view. If user leaves the application for any reason otherwise, the application will restore last known state.

To return main view from message composition, user can select 'cancel' button in preview.

## **Describe any libraries you'll be using and share your reasoning for including them.**

1. Picasso will be used to load and cache images (version 2.71828 or later).
2. Random-dot-stereogram auto-stereo-pair renderer, to be written by me.
3. Random-dot-stereogram single-interlaced renderer, to be written by me.
4. PNG parser to extract alpha channel for shape, to be written by me.

### Google Services

1. Google Speech recognizer  
As an optional feature, user will be able to navigate through the application using basic voice commands such as 'next page', 'tab', 'click', etc. I have already tested this service via my android dictation application exercise <sup>[5]</sup>.

2. Google Service - com.google.android.gms.drive  
Source to retrieve file for viewer or shared option destination.

Other tools

1. Android studio IDE (version 3.1.3 or later)
2. Gradle (version 4.4-all or later)

**Describe how you will implement Google Play Services or other external services.**

Database

Content provider service will be used to communicate with SQL-Lite database.

ShareIntent

Message will be shared with destinations such as Gmail, Facebook.

## Next Steps: Required Tasks

Below is a break down of my application development. It consist of tangible technical tasks that I can complete one at a time until a finished app is realized.

### Task 0: Libraries

Digital image processing is an expensive task. Native development kit (NDK) in low level languages (C, C++) may offer more effective image rendering and filters. I will implement and compare with Java to offer the best performance.

1. Implement Random-dot-stereogram auto-stereo-pair library.
2. Implement Random-dot-stereogram interlace library.
3. Implement PNG filtering library (extract alpha channel).

### Task 1: Project Setup

Steps I will take to setup and/or configure this project.

4. Create Default android project with main view.
5. Add this proposal and and additional feedback from Udacity advisor(s).
6. Add ReadME page.
7. Add SQLite database to data directory.

- a. Implement Content Provider.
  - b. Implement Content Resolver.
  - c. Implement Cursor Loader.
8. Include above Task 0 libraries.

## **Task 2: Implement UI for Each Activity and Fragment**

User interface development include the following activities which are already described above in the mock-ups.

### **Common**

1. Read/Write from sharedPreferences and DB when application start/stop/pause.

### **MainActivity**

1. Animating transition for graphics, text.
2. Menu with Settings, shared options selectors.
3. Shared options dialog to select Facebook, email, etc.
4. View button to invoke ViewerActivityIntent.

### **SettingsActivity**

1. Font configuration dialog
2. Colors selection
3. Radio buttons - random-dot-stereogram type selection

### **Tab pages**

1. Address
  - a. Recipient text field
  - b. Subject text field
  - c. Footer text field
2. Text message
  - a. Key entry rendered in Vector graphics on display.
  - b. Scale to fit screen.
3. Shape
  - a. Upload custom PNG.
  - b. List selection of shapes in memory.
  - c. Scale to fit screen.
4. Preview
  - a. Execute render to generate raster image.

- b. Send button to execute `shareIntent`.
- c. Cancel button to dispose all and return to `MainActivity`.

#### Widget

1. Implement send list.
2. Add `SharedIntent` to handle click item event; launch full application.

### Task 3: Physical device

In addition to the basic user interface, I also need to consider device dimension (tablet vs phone) and orientation (landscape vs portrait).

### Task 4: Toasts

I need error and success handling with toasts.

---

#### Submission Instructions

- After you've completed all the sections, download this document as a PDF [ File → Download as PDF ]
  - Make sure the PDF is named "**Capstone\_Stage1.pdf**"
- Submit the PDF as a zip or in a GitHub project repo using the project submission portal

If using GitHub:

- Create a new GitHub repo for the capstone. Name it "**Capstone Project**"
- Add this document to your repo. Make sure it's named "**Capstone\_Stage1.pdf**"

## References

1. Julesz, Béla (1971). *Foundations of Cyclopean Perception*. Chicago: The University of Chicago Press. ISBN 0-226-41527-9.
2. GPU Gems Chapter41  
[http://developer.download.nvidia.com/books/HTML/gpugems/gpugems\\_ch41.html](http://developer.download.nvidia.com/books/HTML/gpugems/gpugems_ch41.html)
3. My Random-dot-stereogram implementation  
<http://www.ctyeung.com/wordpress/?p=630>
4. Application entity diagram  
<http://erdraw.com/graphs/257151464379/edit>

5. My Android dictation service exercise

<https://github.com/yeuchi/Dictation>

6. Google voice access service

<https://play.google.com/store/apps/details?id=com.google.android.apps.accessibility.voiceaccess>