Assignment 2

ArtCeleration

Amit Rane, Manoj Babu

Project Experience Description – Checkpoint 1

Goals:

- The intent of ArtCeleration was to setup an android library framework and service environment for the provided android app and test their functionality using dummy data.
- The focus of this project is different from assignment 1. We have to implement the underlying service and library to an existing (provided) application.
- The practical aspects of how the principles of Computer System Design - Layering, Hierarchy, Modularity and Abstraction, form the basis of Android library framework and service development was also encountered during the development of this process.
- Since we didn't have the luxury of a complete Application program for testing, we had to think of some innovative approaches to test the functionality of our library and service,

Design:

FIFO QUEUE:

- In order to perform in-order returns, we decided to employ the concept of ticketing algorithms that are usually found in the Linux Kernel.
    - Each Request is serviced as a separate thread
    - Each Request gets its own unique ticket number.
    - We keep a global ticket_to_execute number which is incremented each time a request has been serviced.
    - The threads check whether their own ticket number is equal to the ticket_to_execute. If it is not, they sleep for 10ms and then check again.
    - If they are the same, this thread sends the data back to the Library and increments the global ticket_to_execute.
    - We use the synchronize() construct in Java to enforce mutual exclusion.

COMMUNICATION:

- We use Ashmem to transfer data across the library and the service.
- This involves creating 4 messengers to transport the required data.
    - We have 1 messenger (myMessenger) to send data from the Library to the Service. The data is bundled into a parcelable file descriptor and sent. It is of utmost importance to also embed in the message the messenger that will be listening to data that the Service wants to send back.

- The second messenger (FromServiceMessenger) in the Library is used as a listener. This only listens for messages sent from the Service. It has its own handler to handle incoming messages.
- There exists two messengers in the Service (The previous two were in the Library). We do listening on one of them (ServiceMessenger) for any messages that the Library might send the service.
- The last messenger (ResponseMessenger) is used to send data to the Library. This messenger is actually obtained from the message received by the ServiceMessenger. This ensures that we correctly send the data to the FromServiceMessenger.
- Although there are 4 variables, we observe that there are actually only 3 unique messengers.
- Note: The messenger names mentioned here are not directly the same as used in our implementation.

OUTPUT:

We report the transactions as log messages. The transactions involve sending the image from the library to the service. It also includes the index of the requested transform. We currently print the string equivalent of the byteArray form of the input image bitmap. Prints on the console help the user observe that the queue works as intended.

We have put a long sleep (10s) in the second transform (i.e Neon Edges) which simulates a long execution time compared to the other 2 transforms. We used this to determine if our Queue was working as a FIFO.

Example:

We requested transforms in the following order:

1. Color Filter (Sets the elements to 0)
2. Neon Filter (Increments every element by 1)
3. Color Filter (Decrements every element by 1)

The input image for each transform is:

D/Library Input: [-119, 80, 78, 71, 13, 10, 26, 10, 0, 0, 0, 13, 73 ...]

The log outputs that need to be verified are the ones with the TAG = Library Response. Messages with this TAG should always be IN ORDER. For the above case, the following is the output:

D/LibraryResponse: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...]

D/LibraryResponse: [-118, 81, 79, 72, 14, 11, 27, 11, 1, 1, 1, 14, 74,...]

D/LibraryResponse: [-120, 79, 77, 70, 12, 9, 25, 9, -1, -1, -1, 12, 72,...]

Strategy:

- The plan was to undertake a modular approach. Each member would work on a separate aspect and then integrate the two parts.
- The tasks we chose were:
    - Communication between Library and Service. This includes two way communication between them.
    - Implementation of non blocking FIFO Queue. This ensures that each request sent is committed (returned) in the order it was requested.
- Amit worked on the FIFO Queue and Manoj was responsible for the communication across the Library and Service.
- Integration took much longer than anticipated due to conflicts and issues with transfer of messages.
- Finally, once resolved, the threads were able to launch and return as required.
- We approached the problem with layered and abstracted coding ideology, where we focused on adding only one functionality to the app at a time and then moving forward towards the requirements. The kind of abstraction we did was to replace functions with placeholders to continue the layer. The functions were subsequently implemented to finish functionality.
- The report was written at the end of development.

Challenges:

- The android documentation for IPC from service to Library isn't concise and clear. The idea of Client-Server approach to IPC though intuitive, it's implementation only facilitates the documentation of transfer of data from the Library to the Service. How to transfer data from the Service to the Library was something we took quite some time to figure out. The key idea of sending the messenger on which we will expect a reply from was figured out very late in the design process.

Feedback:

- It was much easier to understand and implement this checkpoint, thanks to the extra Tutorial sessions that Prof. LiKamWa took for each of the core concepts necessary to complete the checkpoint.
- The checkpoint exposed us to new aspects of the android environment and as such was interesting to work on.