AugGraffiti Project Experience

SAURABH JAGDHANE
ASU ID: 1209572595
(sjagdhan@asu.edu)

SANDESH SHETTY
ASU ID: 1209395990
(ssshett3@asu.edu)

Description:

1. Design an Augmented Reality art application with a feature of authentication to store user's gameplay information.
2. All of the user's information is stored in central directory server and communication is managed with HTTP POST request to the pre-defined services and APIs using Volley module.
3. The user can click on his current location marker that opens a camera view where the user can draw and place his own tags.
4. The user can click on and see any of the near-by tags in 50m*50m but can collect only those which are within the range of 5m. To collect the tags, the user has to hold his camera in the same orientation as the tag was placed.
5. User can sign-out of the application, can view his total score depending upon the number of tags collected and tags collected by other users of the application which were placed by user.
6. User can view the tags collected by him/her using Gallery button.

Goals:

1. To design an augmented reality art application with simple UI but lag-free performance and intuitive experience.
2. Explore the existing Augmented Reality (AR) application and extract the best out of them for implementation in our very own AugGraffiti game.
3. Strategize and implement the efficient ways of handling threads and services while understanding their operating system level implementation.
4. Optimizing the network traffic and overhead due to the communication channel between application and PHP server. (Effective use of Volley module)

Design:

<u>Application components: -</u>

1. RequestQueueSingleton class ensures that a single instance of Request Queue is created for the entire lifetime of the application.
2. Threads created to handle continuous update (interval of 1 sec) of score and placing of near-by tags in 50m*50m to avoid any lag in user interface experience. (MapsActivity.java)
3. Only current user location tracked in onLocationChanged() which is also invoked in every interval of 1 second according to the interval set with location request. (MapsActivity.java)
4. MainActivity.java consists of integrating Google sign-in with the application. This provides the single sign-on across the devices to the application which makes use of only user's gmail address. This can also be referred to as a sign-In activity as it also handles second layer of authentication. This includes Google Sign-in API and /login.php. Once the valid result is obtained then only user will be allowed to move to the next activity. This activity also has features of cached signing in which is already explained with code-documentation.
5. GalleryScreen.java uses GridView Layout to display its images and has a GridViewAdapter (GridViewAdapter.java extends BaseAdapter) which uses ImageLoader to load all the collect tags. ImageLoader helps to avoid flickering when

switching between Gallery Activity and Maps Activity as it has an in-memory cache too in addition.

6. UserlocationService.java is a Service that keeps track of Location and Sensor data such that any activity can bind to and retrieve the required information when needed.
7. CameraActivity.java (Place Tag Screen) is the activity for place screen. It has a Fragment Layout in which two views: DrawingScreen.java (extends View) and CameraPreview.java (extends SurfaceView) are added. Also, on place button click it creates a new Volley request to placetag.php and sends the base64Encoded String format of the bitmap retrieved from DrawingScreen.java.
8. CollectActivity.java (Collect Tag Screen) is the activity for collecting tags placed by other users of the game in the near-by region of 50m*50m. It has a Fragment Layout in which two views: ImageView for tag (ImageView niv) and CameraPreview.java (extends SurfaceView) are added. Also, it continuously checks whether the tag and the current orientation details match and on-match it creates a new canvas with a new bitmap object on which a bitmap (cameraBitmap object), created from byte array data (representing compressed JPEG form) retrieved from CameraPreview using onPreviewFrame() callback method, is drawn and the ImageView object niv (representing Tag) is also drawn. This new bitmap is then used to generate a base64Encoded String format which is to be sent to collecttag.php using volley request.
9. Tag.java is an implementation of Parcelable interface which is passed to CollectActivity.java through intent from the MapsActivity.java whenever a user clicks on one of the near-by tags within 5m to collect it.
10. SingleTagView.java represents a single tag view in GalleryScreen.java when one of them is clicked to view it individually.

<u>Interface between application components:</u>

- **Activities: MainActvity -> MapsActivity**
  Email ID passed in the intent from sign-in to Maps so that any following post requests to the APIs can be made based on this email id.
- **Services:** UserLocationService**.**java generates location and sensor data such that any activity can bound to this service as required thus enabling modularity and code reuse.
- **Threads:**
  startDisplayTags() and getScore(). Threads initiated in these two functions invoke the handlers defined in the scope of Maps Activity to update the user interface by placing or removing overlays in case of updating near-by tags and by updating the textview for displaying the score.
- **RequestQueueSingleton class:**
  Every activity can request for an instance of request queue using this Singleton class as it is passed the context of Application instead of Activity such that it is active for the entire lifetime of the application.
- **CameraActivity, CameraPreview & DrawingScreen:**
  CameraPreview is used to enable camera view in Camera Activity and DrawingScreen enables the user to draw on the camera view. Tag, parcelable object and email ID is passed from MapsActivity to CollectActivity. It retrieves bitmap object from Drawing Screen to be sent to placetag.java in base64EncodedFormat.

- **CollectActivity & CameraPreview :**
  CameraPreview is used to enable camera view in Collect Activity. It retrieves byte array data from CameraPreview's onPreviewFrame() method and uses it capture the camera view when the collected tag is to be sent in base64EncodedFormat to collecttag.php.
- **GalleryScreen.java**
  This Activity sends a request to getgallery.php to retrieve all the image urls of the collected tags by the user and displays it efficiently on the GridView layout using ImageLoader.

Strategy:

As Sandesh Shetty did not have an experience in android application development, Saurabh Jagdhane started with the first task so that Sandesh Shetty can go through android documentations and tutorial to learn about the application development. We would also like to give credits to the YouTube channel: thenewboston for a quick start with the Android. Sandesh Shetty could quickly grab up as he has had an experience with Java development. He immediately started off working towards the 2nd task of Volley module integration and sample for PHP server communication with POST requests.

After 1st Checkpoint, plan was to work simultaneously but it didn't work out as the tasks were dependent on each other and one or the other facing an issue resulted in delaying the deadline.

We planned to finish the project 2-days prior to deadline and also our Time Chart shows the plan we were on but due to some issue with thread handling in MapsActivity and CollectActivity we had to delay the submission by a day and hence utilizing 1-slip day.

Interval checkpoints were scheduled and followed as mentioned:

No.	Task	Assignee	Deadline
1.	Google sign-in	Saurabh	09-02-2016
2.	Volley module integration	Sandesh	09-06-2016
3.	Maps Activity	Saurabh, Sandesh	09-10-2016
4.	RequestQueueSingleton class, PHP Communication	Sandesh	09-15-2016
5.	Final UI changes (Checkpoint-1)	Saurabh	09-17-2016
6.	Custom View drawing activity	Saurabh	09-21-2016
7.	Camera Activity	Saurabh	09-23-2016
8.	Overlay of Drawing custom view over camera activity	Sandesh	09-27-2016

9.	Place a tag	Sandesh	09-30-2016
10.	SensorManager for azimuth	Saurabh	09-30-2016
11.	Collection of tags	Sandesh	10-05-2016
12.	UserLocationService for placing a tag	Saurabh	10-07-2016
13.	Gallery for tags	Sandesh	10-07-2016
14.	Testing and Verification using URL and TraceView	Saurabh	10-09-2016
15.	Final UI changes	Saurabh, Sandesh	10-10-2016
16.	Report and documentation of code	Saurabh, Sandesh	10-11-2016

Challenges:

1. **Bounded Service** created initially to retrieve tags in 50m*50m continuously and place it on the map screen. The thread invokes the handler to place or remove the tag. But, the handler should be in the scope of the Map Activity as the tags had to be placed on the Map.
   Solution: Instead created a thread which runs continuously and invokes the handler which is defined in the scope of the Map Activity (MapsActivity.java). It also helped reduced complexity. A bounded service would have been a feasible implementation for this requirement if minimum interaction is required with the Map Activity.
2. User location (blue circle overlay), near-by tags (green circle overlay) and lines connecting user and other **tags duplicating and overlapping** due to update in every interval of 1 second.
   Solution: Keep a list of all tags and lines placed previously, remove them every time before placing new ones.
3. **Sign-in issue:** Initially we were able to sign-in using only one developer as it was dependent on the SHA-1 key of the debug keystore and that is associated with the Google API key created by the developer of sign-in page to be used for sign-in functionality.
   Solution: Created a new keystore, created a new signing config using this new keystore and changed the signing field option in Build column in File->Project Structure to this config. This keystore was then shared with all developers and the same procedure was followed by each of them. Now, default keystore file is overridden with new keystore so that the SHA-1 key is common among developers. The SHA-1 key of this new keystore was then added to the list of SHA-1 keys associated with the Google API key.

4. **Capturing bitmap and Camera Preview simultaneously in CollectActivity:** Initially we were unable to capture both the views: tag and the camera view.
   Solution: Created a new canvas with a new bitmap object. Created a bitmap corresponding to the camera frame data retrieved from camera preview using onPreviewFrame() callback method and drew it on the new canvas. Then, the imageview (part of layout) which has the

tag image loaded is drawn on the canvas. Thus, the bitmap of the new canvas has successfully captured superimposed views of both the camera frame and tag image.

5. **Thread implementation issue in CollectActivity:** Initial implementation involved starting a thread in onStart() method which continuously updates the text view for current distance, tag azimuth and current azimuth and also monitors if the within 5m distance constraint and orientation matching. But, this caused the application to force close and the textview were getting updated slowly.

   Solution: Remove the thread and added function call to the function containing this thread in onLocationChanged() method. This implementation made by the whole UI very smooth and responsive. Tag can now be collected very easily.

6. **Drawing on CameraActivity screen for placing tag lagging:** After enabling setPreview callback() method in CameraPreview.java, responsiveness of drawing on the camera view was very low and was not working smoothly.

   Solution: We traced the issue in this new method added where we were keeping the quality of jpeg data as 100 which later on reduced to 10. As the corresponding byte data is saved by initializing one of the private variables of Camera Preview to this byte array, a large quality image will result in a large byte array being saved thus slowing the whole flow of execution.

Improvements/ Security Flaws:

1. Security lapse with the functioning of login.php as it does not authorize the user separately at the server. After Google sign-in any email id can be sent to login.php to be registered with the server and can access all details of the spoofed email ID thereafter.
2. Small enhancement in the Maps Activity done where if the user clicks on the near-by tags it shows a pop-up displaying how far is the user from the tag as the user can collect a tag only if the user is within 5m of the that tag.
3. If the user is changing his location by 500 m within an interval of specified time frame then he/she should be blocked by the application OR shouldn't be able to register with the server. This will avoid Geo-spoofing by the application developer.
4. The current backend server implementation does not verify the values of the location fields passed in the HTTP POST request, which can be exploited. Eg: using nearByTags.php, the user can pass the location parameters of the remote place and get the information about the tags present in that area.
5. As an application developer, the application user can be provided with a feature that he/she can collect same tag number of times and score will be increased accordingly as per PHP server script. This is definitely a flaw with the Server script and can be fixed so that user is allowed to collect any tag only once. This will make the gameplay interesting and more involving.
6. There should be a provision for earning points when someone else collects your tag which is definitely missing from the PHP server script logic.
7. As an application developer, it is very easy to maintain a distance further than 5 meters to allow user to collect tags approximately around 200 m area. This issue can be avoided if Server script takes user's location within a POST request and checks distance with the tag to be collected every time the request is made. This will avoid calculating distance from a collect

tag and the distance calculation can be offloaded to the server for smooth transition and uniform flow.

8. To emulate the tags placed in 3D space within collect activity along with azimuth angle, pitch and roll angle has to be considered. These are very easy to be associated with the tags placed and should be monitored while capturing a tag.
9. The authentication shouldn't be only bound to email ID. As one user can place a tag with @asu.edu email address and can collect a tag with @gmail.com email ID so that he gets top on leader board. This can be avoided if "Device authentication" is implemented.

Conclusion:

The development of AugGraffitti: real time art application game as an android application development was a great experience with lots of challenges within short period of time. It helped us to understand overall android activity life cycle, handling of multiple threads within Android OS, Volley module usage for POST requests, working with various Sensors, UI design and multiple APIs. All-in-all it was a challenging project to do within short period of time but very good learning experience is what matters.