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.

Interface between application components:

- **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: None as design changed.**
- **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.

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.

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	Saurabh	09-17-2016

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.

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.