Erik Barns

11/24/15

CSC372

Bubble Chat Report

**API Features**

The primary function of Bubble Chat is that it utilizes the phone's Gyroscope. This API feature gives the app its uniqueness by allowing a newsfeed of content to be represented as physical objects that float and react to the orientation of the phone. Bubble chat uses a canvas in which the Bubbles are custom class shapes that exist on the canvas. Each of these shapes incorporates a “move()” method which determines the distance at which they will move on the screen. In the “move” method the shape object’s coordinate position is influenced by the gyroscope sensor data. As the phone orientation shifts the sensor data will positively or negatively influence the value of the bubbles direction and velocity. The movement of the object is determined by the sensor data multiplied by our predetermined gravity value and then divided by two. This formula created the optimal movement for a bubble on screen.

Another feature included in Bubble Chat is the ability to produce 2D drawings. The user can draw a picture using their finger and then submit the drawing to be added to the Bubble Room. Similar to the Bubble Room the Drawing functionality relies on a custom canvas class. The Canvas class utilizes the Motion Event and Paint libraries to create an environment that allows users to draw free flowing lines, mimicking a marker on paper. When the canvas recognizes an OnTouchEvent the coordinate positions are collected. If the touch action was an initial press then a path is established from that point, if the user's finger is moving then a line is drawn to that path. Utilizing the invalidate method after each event call the canvas is redrawn. This gives the user the ability to draw free flowing lines and pictures with expected responsiveness.

The Bubble Room also utilizes 2D drawing but instead of drawing lines, the bubbles are constantly moved and redrawn. The Canvas View class utilizes a SurfaceView. The advantage to the SurfaceView class in this functionality of the app is it’s ability to produce smoother frames per second. The “doDraw()” method is constantly being called while the rendering thread is active, each iteration the “move()” method is called to evaluate the position each shape will move to and then the canvas is redrawn. This functionality creates the animation, it gives the illusion of movement to the user when in reality each object's position is being recalculated and redrawn constantly.

A feature that Bubble Chat incorporates but was not included in our classwork is the Camera. Bubble Chat utilizes the camera to take pictures and convert them into Bitmaps which are then transformed to shapes inside the Bubble Room. When using the camera I discovered that there is less work involved than expected. The camera is started up using an intent call to the phone’s Media Store library. From there the default action takes care of loading the camera, and taking a picture. All that is needed is to expect a result from that default activity. Using the intents extras bundle the picture can be retrieved as a Bitmap.

**Conclusions**

The biggest challenge that was faced when creating Bubble Chat was running out of memory. With no server in use all of the bubbles must exist on the phone. This requires the data to be stored in a centralized place that exists throughout the lifetime of the application. All of the bubbles are collected and stored on the main activity of the application, but they also need to be passed to the Bubble Room upon opening the activity. Passing multiple bitmaps across an intent was too strenuous and always resulted in an out of memory error. This lead me to research how I could reduce the size of the Bitmaps. Ultimately I didn’t want to lose the quality of the pictures as it would be a huge drawback to the application. Another solution was to put a cap on the number of bubbles the room could hold in order to prevent the error, although this quickly proved inefficient as it seriously handicaps the functionality of the app. Ultimately I discovered that the bitmaps could be temporarily converted into a byte array and then be expanded back into a bitmap when the bubbles are being initialized in the new activity. The byte array removes a lot of the overhead data that is passed along with the bitmap and instead of passing data it stores the data in memory and is then accessed from the new activity.

Bubble Chat is in a state that I’m content with for my first application but still has many limitations and room for growth. One limitation I’ve encountered is the quality of the image as the Bitmap is condensed and cropped into a circular bitmap. The problem with this is the image is shrunk to be placed into a bubble, when the user wishes to enlarge the photo it is of considerably lesser quality than the original. This could be greatly improved with more research and manipulation of the composition of a bubble. I think this could also be improved with the use of a server. The full image could be stored on the server so when the user clicks the bubble is links them to the full sized image opposed from enlarging the current bubble. I also hope to expand on the Bubble Room by having the bubbles react to each other and not only the bounds of the canvas. I have ideas for the implementation of this but could not successfully develop the functionality. Bubbles that are overlapping and following the same trajectory in the Bubble Room are an annoyance as you want to see each bubble individually and not struggle to separate the two moving images. This overlap also affects the function of focusing a bubble to view it enlarged. If two bubbles are overlapping then the current code will load both for view. I initially thought to remove this so that only one image pops up but the current functionality seemed to be optimal if two bubbles could overlap.

Another limitation of Bubble Chat is the selection of colors. I could not effectively create a way for users to choose colors based on hex value. Users can currently choose from preset colors. To improve this I would need to develop a color wheel or progress bar associated with hex values. This would improve the detail of the application, creating a more personalization for the users experience in Bubble Chat.

When developing this project I came across a few limitations or flaws in the Android SDK that affected my project. I don’t own an Android device and had been developing my homework on the Android emulator that Android Studio provides. When it came to the final project I discovered that the emulator cannot handle many of the key API features that were needed for the project. I also discovered that the emulator had memory issues with a few graphics, this proved challenging because it lead me to believe there was a problem with my code. Another limitation I discovered in the Android SDK was the design view. I didn’t make a habit of using the design view to build the layout but I often checked it to see my progress. I learned that the design view is not always an accurate representation of the applications layout architecture. It caused me to fiddle with XML for extended periods of times to realize the issue only existed in the design view.

Developing an Android application has been on my mind for the past few years, so this was a thrilling experience to finally produce an idea. In the past I had always been overwhelmed by the API syntax. This class taught me that Android Studio will do a lot of the work for you in regards to remember syntax specifics. This process has definitely motivated me to continue to pursue Android development when the moment arises. It has been a challenging but positive experience. Application development intertwines creativity, technicality and innovation. These are three of the characteristics that lead me into Computer Science and I thoroughly enjoy being able to apply them in some form.