**Android Project Notes**

So in CS 454 one of our projects was to develop an android application based on the game boggle. The first piece I needed to do was create a dynamically scalable android layout that could fit any type of android device. I didn’t this by using android’s match parent and wrapped content settings for most of the layouts height and width along with creating my own custom layout and text view to create the boggle grid. These custom objects would always draw themselves with their height equal to their width creating a square layout and square text views that I can arrange into a grid.

Once I got the dynamic android layout created the next piece that I needed to work on was the touch inputs for the boggle board. In this application the user should be able to touch the screen on the local board and trace out a word using the letters displayed in the grid what I needed to do was to create a on touch event function in the android layout’s activity that would listen for whatever the user touched the screen and track the position of the user’s finger. When the user placed his or her finger down onto the screen the touch event function would get those X and Y coordinates and compared them to the positions of the grids on the screen. If the person’s finger was inside one of those grids then the function would register that as a hit and add the letter in the grid the user was touching to the word. Then if the user moves his or her finger to another grid the touch event would trigger again and find the next location the user moved his or her finger. Once the user was done highlighting a word he or she would lift their finger off the touchscreen and this would trigger the final touch event. The function would check to see if the word highlighted was a real and if it was it would add it to a list containing all of the users highlighted words.

The next major piece of the android app that I created was the word search function. This function requires several key pieces to work such as a dictionary to find a word or tell if a string is part of a word and a function that takes a position in the grid and the previous path already taken and finds all possible positions that a user can take at that position. The dictionary piece needs to be able to read in all the words from a text file contained in the razz/raw folder and add them to a data structure that will be used throughout the entire life of the game activity and then quickly search through the data structure and find if a string exists or if that string is a sub string of a word in the dictionary that begins at the first character of the string. The data structure I used to hold all the words in the dictionary file is a hash map that uses a string as the key and stores an array list for each key. This key represents the first two letters of each word in the dictionary and the array list stores all words that contain those first two letters. This means that the dictionary has been split up into a number of smaller easier to search pieces and can filter out many dud words based upon the first two characters. Once I had the dictionary and its two functions working along with the get all possible movements functions working all that was left was for me to create the word search algorithm. I decided to use a recursive algorithm that would start at one of the 16 possible positions in the 4 x 4 grid and find all possible words that could be highlighted starting at that position. The recursive algorithm would do that by passing the algorithm the word so far the X and Y positions of the grid that the algorithm is that and a 4 x 4 double array representing all of the other grid positions that have already been traversed. The algorithm would then mark its position off as being already traversed in the double array and then get the letter located at that position and add it to the string representing a possible word that was passed to the algorithm. It would then check to see if this new word was a word in the dictionary if it was than the algorithm would add that word to an array list of strings representing all words found in the word search algorithm. The algorithm would then find a list of all possible grid positions that can be traversed to at this position it would then call itself using the new word the new double array of traversed positions and each of the possible positions that are found to be traversable from this position. If the string passed to the algorithm wasn’t a word then the algorithm would check to see if that string represented the first part of a word. If the string was part of a possible word then that string would not be added to the words found array list but you would do everything else that you would do if the string was the word. This algorithm would find all the possible words that could be found if the user started at disposition so in order to find all the possible words in a grid than this algorithm needs to be called for all 16 positions on the grid and passed each one of those positions along with an empty string for the word and a double array with no positions traversed yet.

The final piece that I implemented for the project was the ability to communicate and play the boggle game with the second player. I did this by using Bluetooth to establish a peer to peer local connection based off of a Bluetooth chat program provided by android SDK. What this program did was to first search for all available Bluetooth devices and display them along with their information. Once the user had decided on a device to connect to they would then select that device an attempt to establish a connection. My program would attempt to connect with another device by utilizing a Bluetooth manager that would run different threads depending on states. When the app starts up the Bluetooth manager runs two threads at once, one that listens for an attempted connection by a secure device and the other listens for an attempted connection buying insecure device. When one of the apps tries to connect its state changes to the connecting state and it sends a connection message to the other device that’s listening for a connection. If the devices make a successful connection then the listening device should send its information back to the other device for display and then both devices should stop running their active threads and move to the connected thread. In this state both devices have sockets to the other device and can send string messages to each other. Once the connected thread is running our app can start the game by first determining who is the master and who was the slave. The master is the device that creates the boggle board and word list that is shared between the two devices and should be determined by whoever started the connection. To send over the boggle board and word list my program should combine all the elements in a boggle board or word list in to a single string then add an identifier to the front of the string before sending it. When the other device receives that message in the handler it first removes the identifier from the message and then uses that identifier to determine if the message is a boggle board, word list, or word from player two, it then parses the message based on the identifier.