**Project Documentation**

Project Title: Memory Leak 2.0

Group Members: Daniel Li, Victor Hui, Julia Zeng, Jon Ngo, Samantha Au

**Challenges**

The group has encountered many challenges throughout the process. We are unfamiliar with Android Studio and java but our application is all based on them. Thus, overall learning and understanding the new interface is hard. At the beginning, it was frustrating to understand the connection between GitHub and Android Studio. It is challenging to understand the animation for the clouds. We had to use lots of trial and error to understand the timing and position for the clouds’ animation. Another challenge is understanding the constraining objects and how the mechanics work for those objects. We also have trouble with getting the fun fact from the internet because it is challenging to retrieve the data. Additionally, it is also difficult to conceptualize an efficient way to find tutorials and resources regarding the new interface.

**Front-end.**

*User Interface (rocket, shooting)*

|  |
| --- |
| The main way that the user interacts with the game is through clicking on buttons, moving the rocket, and shooting bullets at clouds which represent multiple choice answers (A, B, C, D). To move the rocket, the user taps an x position that the rocket does not currently occupy. To shoot bullets, the user taps within the rocket. The team implemented this function by spawning an image of a bullet at the tip of the rocket. With the help of the Handler and Time capabilities, the bullet is moved up at regular intervals of time. A similar property is given to the choice clouds where instead of moving up the screen, the positions are changed to move down. |

*User Experience (main page, background music)*

|  |
| --- |
| When the user opens the app, there will be the main page with buttons to navigate. There is also a programmed animation of clouds falling to fit the theme of the game. The start button will begin the game process, the settings button will open the settings menu, and the instructions button will open a screen with instructions to play the game. On the bottom of the main menu, there is a random fun fact about C++ that is retrieved from the internet. |

**Back-end.**

*Main Page*

|  |
| --- |
| Most of the front end was built using the XML file associated with the activity of the main screen. The button navigation uses an on-click listener to navigate to the associated activity. For the animation of the cloud, a timer and change position function is used to loop the clouds moving. For the fun fact, there is an online database of fun facts that are created using JSON. The app establishes a connection to the URL for that database by using an HTTP URL connection and opening that connection. Then an input stream is created and then a buffered reader reads that input stream. The data from that input stream is parsed to get a certain fun fact using a JSON array. |

*Utilizes objects or organize and control (main page, normal level, boss level)*

|  |
| --- |
| After clicking the start button, the user is brought to the main game screen where the game is played. To control the functionality of the screen we used Java, and to format the design we used XML code. The program uses object-oriented coding to control individual game functionalities like the firing of the bullet, motion of the bullet, collision of the bullet with the clouds, and changing of the score. Each object is either an ImageButton type or ImageView type, which allows us to use the accompanying built-in functions packages. |

*Point System Control (boss level)*

|  |
| --- |
| Through running our database of questions and the corresponding answers, the application indicates the user’s score. It also illustrates the HP level of both the boss and the user by using ProgressBar to control. |

*Detection of user’s motion (normal level)*

|  |
| --- |
| User-program interactions are controlled using event-driven code. To move the rocket, we used the OnTouchEvent method which detects the user’s touch motions and translates it into object motion. The actions of the bullet are controlled by the built-in visibility function and onClickListener function. |

*Collision Control (normal level)*

|  |
| --- |
| The program uses collision methods to detect the collision of the bullet and clouds. There is a separate method that holds the position of the cloud with the correct answer. When a bullet collides with any of the cloud objects the position of the bullet is compared to the position of the correct cloud answer. This action calls an update to the progress ImageView object and increments the score if the player is correct. |

**Final Thoughts**

There are lots of improvements that we can work on if there is more time for this project. If time allows it, we hope to add a couple changes to the game. In this game, we created a sample of database that we are using in the game. Thus, first, we would turn the questions into a bigger database in the cloud. This allows a bigger spectrum of questions that users can play around with. Secondly, we also hope to include more levels in the game. Thirdly, the code blocks for the cloud in the normal level games are generated by a set of data dedicated for each level. Therefore, we hope to improve this by adding a function that can randomly generates the code blocks. This way, user would not encounter the same question each time they repeat the game. Additionally, we also want to improve on the polymorphism of the mechanics side of the game. For instance, we hope to improve on the movement of the racket to be more flexible and realistic in the game. Afterall, we are happy with our application because we are able to put our ideas into live and we have gone further than we have expected with the application.