**Mobile Application Development with Android: Project II**

**Participants: Connor Hargus, Jimmy Ye, Tahmid Munat, Nikiander Pelari**

**Project Name: Barrel Dodger**

**Project Type: Interactive Android Game**

**Project Description:**

We created an interactive android game called Barrel Dodger. This game uses similar ideas and functionalities to the common game “Flappy Bird”, and we built it as an adaptation of a game created in the online tutorial at <https://www.simplifiedcoding.net/android-game-development-tutorial-1/>. The main activity acts as the entry point to the application where the user can either start the game or play/pause the background music. The ‘start game’ button takes the user to the actual game screen where a donkey can be controlled to avoid incoming barrels.

The user can touch the screen which will ‘activate’ the jetpack carried by the donkey, and it can fly vertically in order to avoid the barrels to stay alive. The user has 10 available lives and gain points based on time. The incoming barrels arrive at different speeds and also speed up based on the user’s use of the boosters. The game ends when all 10 lives are lost, and the user is prompted with their score and asked to play again if they so choose. The user can also go back to the main menu to change their music settings.

We monitored several aspects of the energy consumption. Below is the result of our observation:

1. What aspects affect energy consumption?

We found the resources to consume the most amount of energy. The background music files and the touch.mp3 file are accessed frequently. The android device (and the emulator) dropped a few frames due to high usage. Battery usage was also higher on the main menu while the music was playing vs when it was muted. Depending on the number of touches and collisions, we saw a spike/dip in CPU and memory usage which also affected the energy consumption.

1. What kind of app costs most energy?

We found out that apps that use media resources and often rely on user interactions are the most consumption-heavy. Multiple services are run and background tasks are executed in order for them to perform successfully. Framerates, resolution and internet usage also play a role in determining battery consumption.

1. Is there any way to save the energy from the view of user?

While running an app, the user may choose to turn off a few options to save energy. This may be higher graphics options, turning off music, lower the brightness etc. The user should also close unnecessary apps to avoid draining more energy. Additionally, the user could play the app in “airplane” mode since it does not require internet or Bluetooth connectivity and these things use energy running in the background.

1. Is there any way to save the energy from the view of app developer?

A few possible ways to save energy from the view of an app developer is to run a proper number of methods in an activity to avoid dropping frames. The developer should create options for the user to turn off unwanted features, whether or not they are implemented. While coding, all edge cases should be considered so that processes that need to be terminated are taken care of. We could also sacrifice some of the smoothness of the gameplay by lowering the frame rate, which would mean that the canvas would not have to be updated so frequently and the app would run fewer computations. This would save energy.

1. Describe all methods you can propose to save energy.

Here are some of the ideas we discussed above:

* Using a lower graphics resolution
* Turn on airplane mode
* Turning off music
* Lower brightness
* Lowering frame rate
* Minimize unnecessary/unwanted features (done in development)

1. Try to use the “Monitor” to evaluate your approaches.

We used the Monitor app to evaluate energy consumption. Most of the time the CPU and battery usage hovered around 10-15% and memory usage was normal (about 200 MB). The battery always remained at 100% due to the nature of the emulator. We tested the approach of using airplane mode, and found as we expected that the relative percentage of battery usage by our app actually increased to about 20%, but this was merely because other apps using internet in the background were now using less power. Next, we tried lowering the brightness, which didn’t seem to drop the battery usage of our app as much as we expected (only a couple percent). However, there was still some improvement. Turning off the music seemed to make a bigger difference, dropping the battery usage by around 3% in our tests. Lowering the frame rate dropped the cpu usage substantially, but we felt that we did not want to sacrifice smooth gameplay for these decreases in power consumption. Similarly, if our game was more complicated we could have simplified our graphics a bit to decrease CPU usage in manipulating and displaying the images, but we felt that the resolutions of the images as they are is important to the look and feel of the app. Overall, we found that media makes the biggest difference for our particular app.