Games for Mobile Devices – CA 2

In App Purchasing Steps (How it was done)

Jack Carroll (T00194823)

The in-app purchasing (IAP) service used for the testing of IAPs was the Unity Purchasing service within Unity itself. Before writing the code and testing the scene where IAPs were taken into consideration, the IAP service was switched on in the Unity application itself where this process can be seen in Figure 1.1 below.

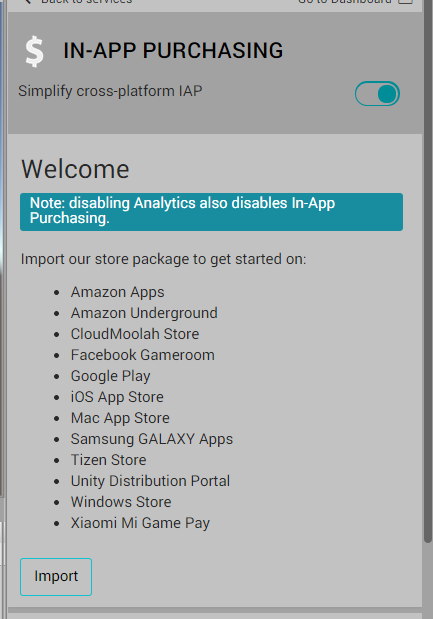


Figure 1.1 – In App Purchasing Service Switched on

Following this, the Unity Purchasing packages required the UDP and the Unity Purchasing package itself to write code for the testing of IAPs when different items were going to be tested for purchasing including the purchasing of 100 coins, 500 coins or to pay for no advertisements to show up on screen within the application. A demonstration of the installation and importing of the packages needed for the Unity IAP service is shown in Figure 1.2 following this sentence.

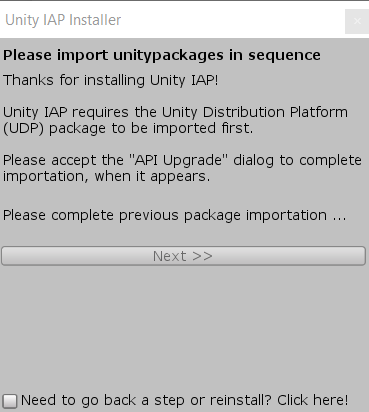


Figure 1.2 – Installation of Unity packages for IAP service.

Figure 1.3 displays all the packages which were installed when proceeding through the process of installing and importing the Unity packages. The Unity IAP packages were all installed to ensure that code could be written to activate prompts in the main application to potentially show the sample prices for the items including the removal of ads or getting either 100 coins or 500 coins for a particular price.

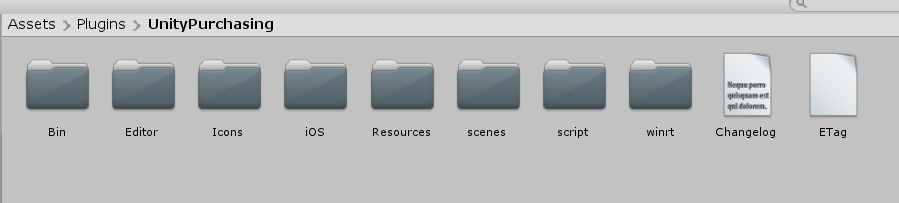


Figure 1.3 – Packages needed for Unity In-App Purchasing.

The establishment of the three different IAP purchases which included the removal of all advertisements within the application, getting the player 100 coins and getting the player 500 coins was carried out where the buttons indicated in Figure 1.4 show how the Unity Purchasing module would be implemented in the project after setting up the Unity Ads, Admob ads and the Google Play Services. The IAPs would be tested out in the three buttons for the three IAPs as seen in the image below this paragraph. Later, the Google Play Services pop-up for IAP testing would feature in the same scene where the three IAPs were located by using a tester link generated by the Google Play Console for testing out the IAPs. This process would eventually be carried out by performing the testing once the application was rolled out and published in the Alpha level of testing.

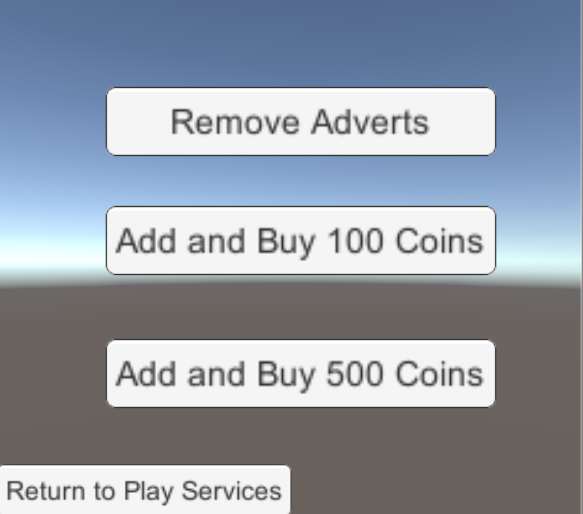


Figure 1.4 – Layout of IAP screen in “AdvertisingServices” Unity project

After the testing of the IAPs was successful by just using the Unity Purchasing package for seeing the purchasing being carried out with debug messages when the application was rolled out, an APK of the latest version of the project application was uploaded in the Alpha level of testing to allow managed products to be created in the Google Play Console. Three managed products were created where the removal of ads product, the 100 coins product and the 500 coins product were all created with set prices given to the products that were worth specific prices as established by the project developer. The list of three IAPs used for the testing of IAPs with the Google Play Services through a particular testing link to carry out the testing of the application can be seen following this sentence in Figure 1.5.

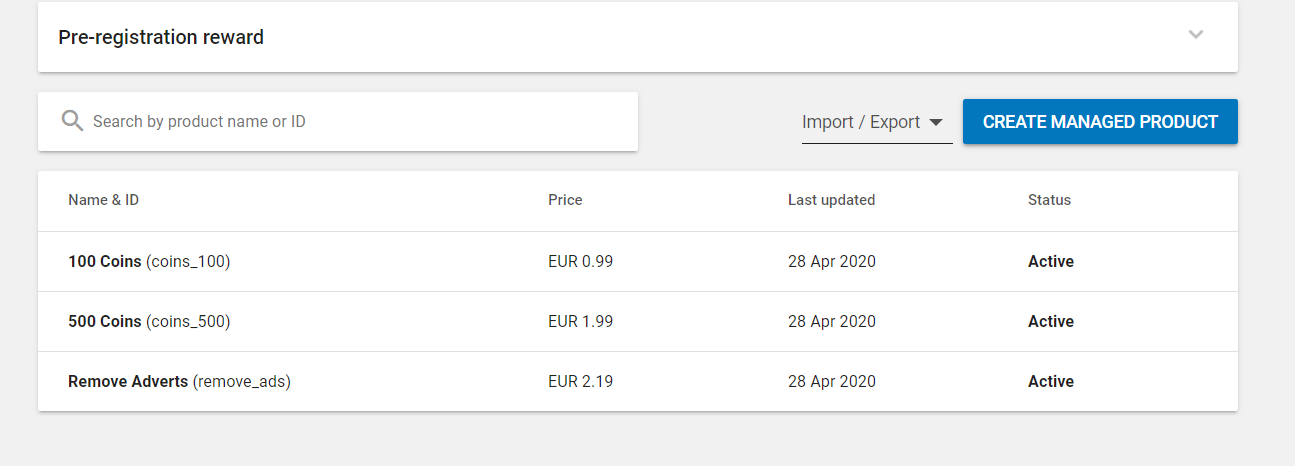


Figure 1.5 – List of In-App Purchases created by this user for Google Alpha Testing.

Once the IAPs were created in the Google Play Console, the list of testers to test out the application were setup in the Google Play Console to allow these testers to participate in the Internal Test Track testing once an email was sent to the testers where that link for the test application was copied and sent by email to an Android device for the developers to test out the application. In Figure 1.6, the demonstration of the Google Play Store in action to purchase the IAPs can be seen where once the user decides to choose an IAP, e.g. 100 Coins, the Google Pay pop-up appears to allow a user to add a debit or credit card or to add a PayPal account to officially buy an In App Purchase. Once the payment method was added, the IAP was officially purchased and the payment was successful.

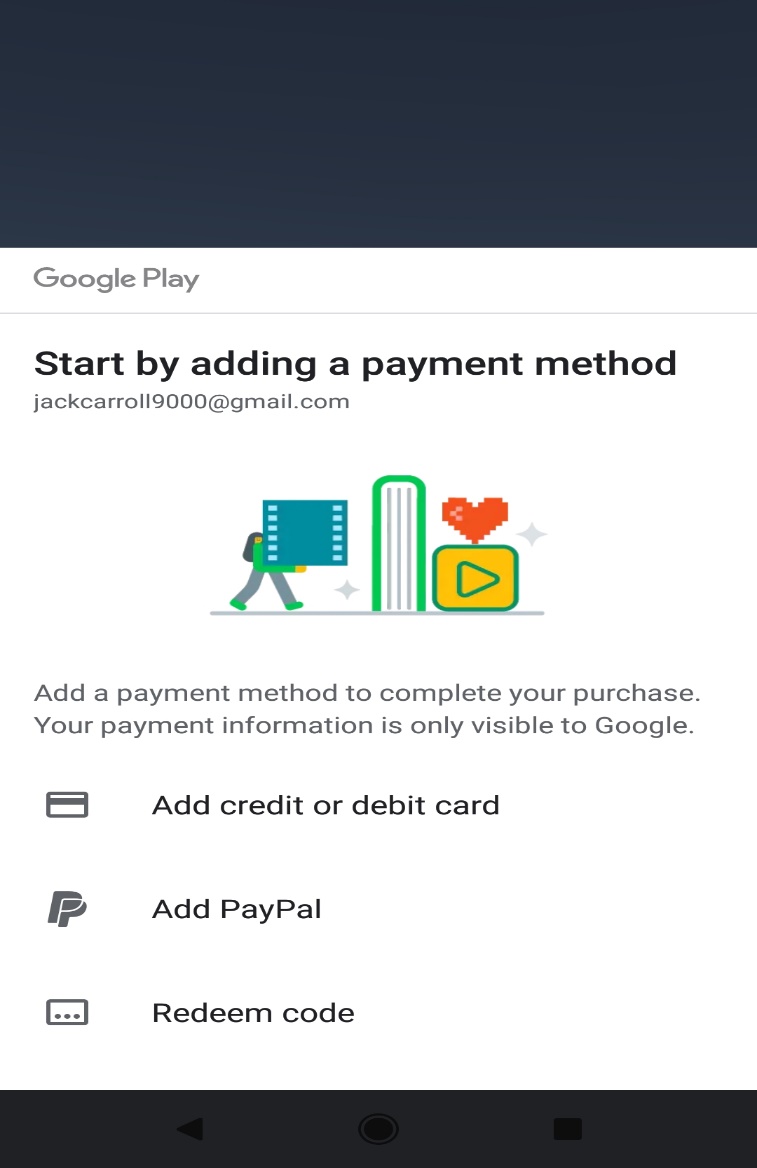


Figure 1.6 – Google Pay pop-up showing that In App Purchase would work when credit / debit card added.

Once a copy of the web link to access the Internal Test track or Alpha Testing program was established online in the Google Play Console, the testers could test out the application to see how the application functions and to discover any flaws present within the application that was tested. Figure 1.7 displays the official email and message to join the Internal test programme to test out the application which was carried out by this user.

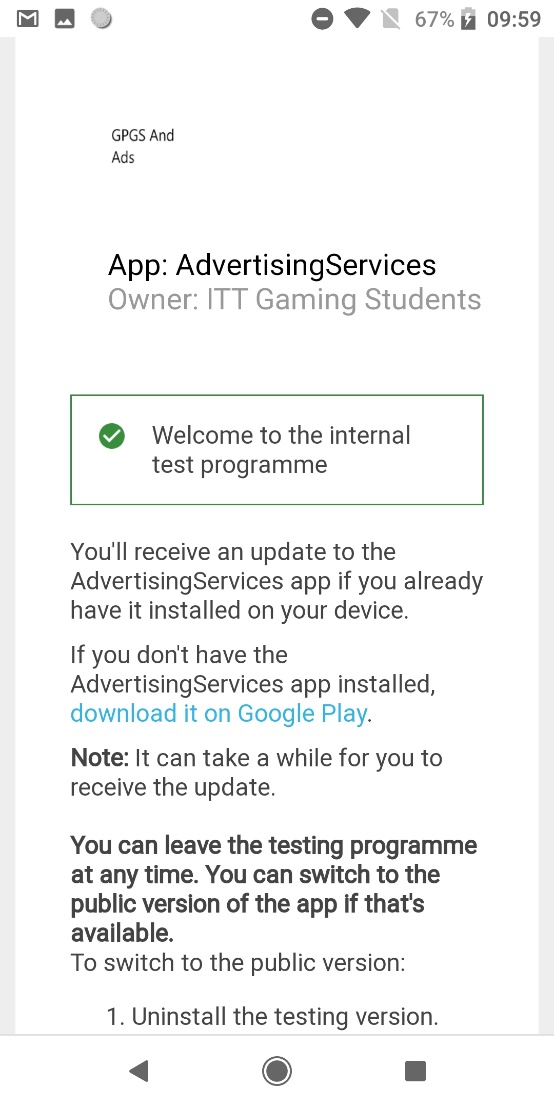


Figure 1.7 – Accessing Internal Test program.

Prior to testing out the application with the IAPs established in the Google Play Console, the Unity Purchasing module on its own was tested out in the Unity application itself before it was built when any button was pressed on the screen to purchase any of the three IAPs. Figure 1.8 shows the execution of the removal of the advertisements which was successful once the “Remove Adverts” button was pressed within the application.

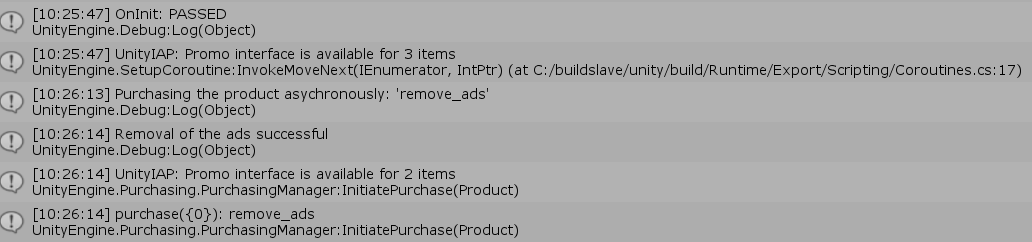


Figure 1.8 – Removal of ads shown in debug messages once the “Remove Adverts” button pressed.

Part of the code shown in Figure 1.9 based on the Unity tutorial given on the Unity IAP service shows how the IAPs were added to the Unity Purchasing builder for these to be initialized. This allowed for the execution of the purchasing of any of the products after the initialization of all the IAPs occurred. The creation of the IAP and PurchaseButtons classes was based on a tutorial on how to initiate Unity Purchasing for IAPs where the names of some of the methods and the names of the classes themselves were changed for the benefit of this user (MacNeal, 2020).

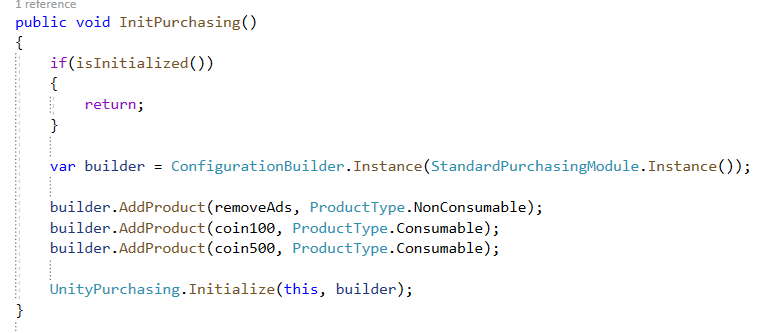


Figure 1.9 – Code to initiate the Unity Purchasing module for the addition of the non-consumable and consumable products.

To officially buy the removal of adverts IAP when the button to purchase the IAP for “Remove Adverts” was pressed, the code shown in Figure 1.10 below displays the method used to execute the code where each button was represented by an IAP where the buttons class determined the actions of a particular IAP which was pressed. For example, the removal of adverts would show that the purchasing of the removal of ads was either successful or failed. The code shown below was placed in the IAP class where this was implemented in the “PurchaseButtons” class for the pressing of the buttons which were placed into a switch statement.

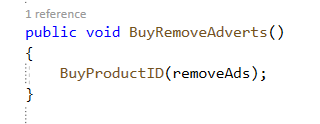


Figure 1.10 – Code for initiating the purchase of the removal of the ads once the button was pressed.

Each method was implemented in the functionality within Unity to use the switch statement to allow for the activation of a chosen IAP where the user could choose which IAP or IAPS were purchased. The switch statement code placed within the PurchasingButtonClick() class was implemented in each of the three buttons in Unity depending on the type of IAP that the user chose to buy. Figure 1.11 refers to the code written to allow the user to choose from the different IAPs when one of these buttons was pressed in the Android application.

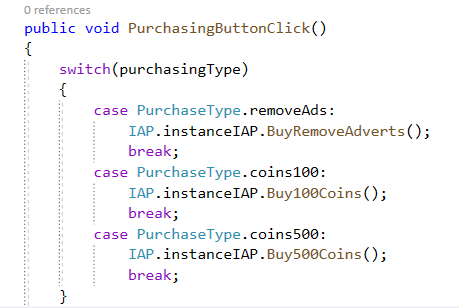


Figure 1.11 – Code that shows the switch statement in action which initiates an action based on what the user chooses to buy from the IAPs that is implemented in the Unity inspector for every button.

With the methods needed for initiating the purchasing of any IAP, the methods shown below in Figure 1.12 indicate how the BuyProductID(string purchase) which returns an ID for a purchase, e.g. buying 100 coins plays a significant role in testing out the IAPs to see how the code is executed with the various messages including the type of purchase being made and the initialization of the purchase. This was all carried out through the Unity Purchasing module which was installed and updated to the latest version for the testing of the IAP.

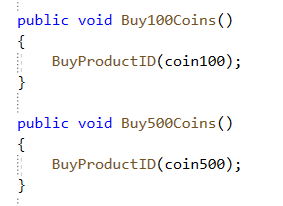


Figure 1.12 – Methods for implementing IAPS to purchase 100 coins and to buy 500 coins once those buttons pressed.

With the method shown in Figure 1.13, the PurchaseProcessingResult method was an important part of the Unity Purchasing test prior to placing the managed IAP purchases in the Google Play Console once the latest APK file was uploaded to the app releases for either Internal Track testing, Beta / Alpha testing or the Production phase. This was because the method displayed below in code represented the outcome of which IAP was purchased based on which IAP button was pressed for purchasing. Otherwise, the purchase that was attempted to be made by the user would fail due to no Internet connection when the IAP process was online-only.

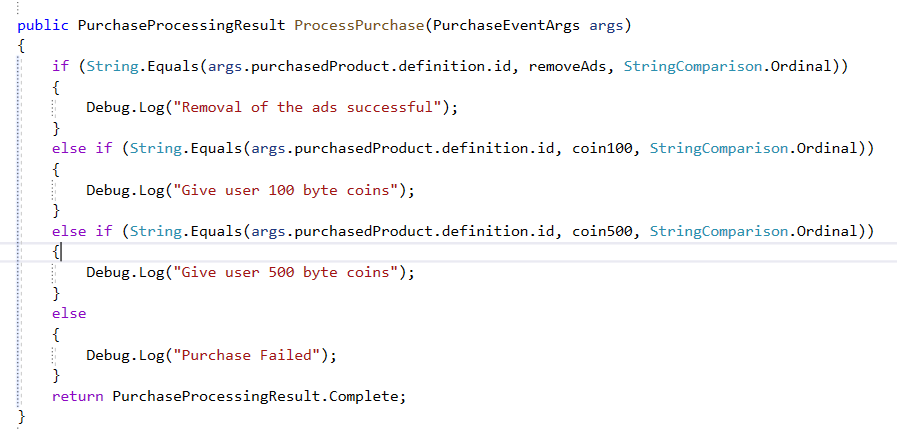


Figure 1.13 – PurchaseProcessingResult method which processes the result of the payment that the user decided to make depending on which IAP is purchased.

With the OnPurchaseFailed method, the failure of the IAP purchase can be seen in the code shown in Figure 1.14 where if the user was disconnected from the Internet and pressed one of the three IAP buttons, the debug log showing the failure of the purchasing of a particular item along with the ID of that product and the reason for the failure is displayed in the Unity debug console when the Unity Purchasing module was active before implementing the Google IAP system in the Play Console.

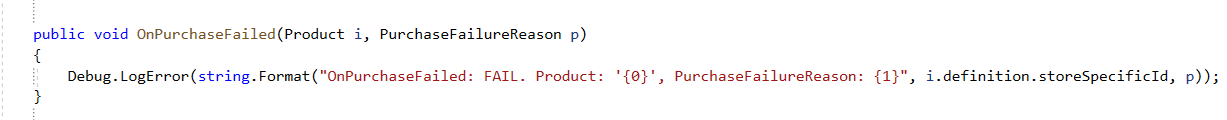


Figure 1.14 – OnPurchaseFailed method that indicates the failure of the purchase.

# References

In-App Purchases 2020 | Unity Tutorial – Jake McNeal – 2020 - <https://www.youtube.com/watch?v=G2peSMPGffI>

Unity IAP Tutorial – Unity – 2020 - <https://learn.unity.com/tutorial/unity-iap>