Ads Framework

In-game advertising is a monetization strategy that game developers use to boost their game's revenue. Game developers earn money and get paid by showing mobile game ads to their users. There are several SDK available for implementing Ads in Application among them Google Mobile Ads SDK is commonly used for mobile platforms.
The Google Mobile Ads SDK is the latest generation in Google mobile advertising featuring refined ad formats and streamlined APIs for access to mobile ad networks and advertising solutions. This feature provides a way for you to earn more money by matching ads to your app based on the criteria you set. The ads are created and paid for by advertisers who want to promote their products and will pay different prices for different ads.
Google provides samples, source codes and sdk for users for free. You can check out the details on this link:
https://github.com/googleads/googleads-mobile-unity

Steps to setup Mobile Ads Unity plugin:

1. Download Google Mobile Ads Unity plugin form Website or Github.
   Website: https://developers.google.com/admob/unity/quick-start
   GitHub: https://github.com/googleads/googleads-mobile-unity

2. Import the Mobile Ads Unity plugin

   Open unity project and navigate to Assets > Import Package > Custom Package.
   You can see the downloaded file with name: “GoogleMobileAdsPlugin.unitypackage”
   Select this and import in the project.

3. Resolve the Dependency for the specific platform.
   For resolving dependency

   Android
   Assets > External Dependency Manager > Android Resolver > Resolve.

   iOS
   OS dependencies are identified using CocoaPods. CocoaPods is run as a post build process step.
   Open xcworkspace to it will include all the dependency libraries.

4. Setup your AdMob app ID

For setting up AdMob app ID
Assets > Google Mobile Ads > Settings.

Code Part (Unity Project)

*use **“using GoogleMobileAds.Api;”**

1. Raise ad events on the Unity main thread for synchronizing Mobile Ads SDK with unity Main thread.

```
MobileAds.RaiseAdEventsOnUnityMainThread = true;
```

2. Initialize the Mobile Ads SDK before loading any ads. This step will sett all the required parameters for call ads.

```
MobileAds.Initialize(initStatus => { });
```

3. Select Ads Format

There are several ads provided by Google but this framework support most basic ads

   a. Banner Ads
   b. Interstitial Ads
   c. Rewards Ads.

Banner Ads

For Banner Ads:
Example for banner ads code:

a. The below code is for Loading Banner ads using adUnitID.

```
  // These ad units are configured to always serve test ads.
#if UNITY_ANDROID
  private string _adUnitId = "ca-app-pub-3940256099942544/6300978111";
#elif UNITY_IPHONE
  private string _adUnitId = "ca-app-pub-3940256099942544/2934735716";
#else
  private string _adUnitId = "unused";
#endif

  BannerView _bannerView;

  /// <summary>
  /// Creates a 320x50 banner at top of the screen.
  /// </summary>
  public void CreateBannerView()
  {
      Debug.Log("Creating banner view");

      // If we already have a banner, destroy the old one.
      if (_bannerView != null)
      {
          DestroyAd();
      }

      // Create a 320x50 banner at top of the screen
      _bannerView = new BannerView(_adUnitId, AdSize.Banner, AdPosition.Top);
  }
```

b. The below code is for Showing Banner Ads

```
/// <summary>
/// Creates the banner view and loads a banner ad.
/// </summary>
public void LoadAd()
{
    // create an instance of a banner view first.
    if(_bannerView == null)
    {
        CreateBannerView();
    }
    // create our request used to load the ad.
    var adRequest = new AdRequest();
    adRequest.Keywords.Add("unity-admob-sample");

    // send the request to load the ad.
    Debug.Log("Loading banner ad.");
    _bannerView.LoadAd(adRequest);
}
```

c. The below code is for destroying banner ads.

```
/// <summary>
/// Destroys the ad.
/// </summary>
public void DestroyAd()
{
    if (_bannerView != null)
    {
        Debug.Log("Destroying banner ad.");
        _bannerView.Destroy();
        _bannerView = null;
    }
}
```

You can also add events for banner ads like when user click on ads, when ads is failed to load, when ads are closed, etc.

For more details go to this page - https://developers.google.com/admob/unity/banner.

Interstitial Ads

For Banner Ads:
Example for Interstitial ads code:

a. The below code shows how to Load Interstitial ads.

```
private InterstitialAd interstitialAd;

/// <summary>
/// Loads the interstitial ad.
/// </summary>
public void LoadInterstitialAd()
{
    // Clean up the old ad before loading a new one.
    if (interstitialAd != null)
    {
          interstitialAd.Destroy();
          interstitialAd = null;
    }

    Debug.Log("Loading the interstitial ad.");

    // create our request used to load the ad.
    var adRequest = new AdRequest();
    adRequest.Keywords.Add("unity-admob-sample");

    // send the request to load the ad.
    InterstitialAd.Load(_adUnitId, adRequest,
        (InterstitialAd ad, LoadAdError error) =>
        {
            // if error is not null, the load request failed.
            if (error != null || ad == null)
            {
                Debug.LogError("interstitial ad failed to load an ad " +
                               "with error : " + error);
                return;
            }

            Debug.Log("Interstitial ad loaded with response : "
                      + ad.GetResponseInfo());

            interstitialAd = ad;
        });
}
```

b. Below code shows how to show Interstitial ads.

```
/// <summary>
/// Shows the interstitial ad.
/// </summary>
public void ShowAd()
{
    if (interstitialAd != null && interstitialAd.CanShowAd())
    {
        Debug.Log("Showing interstitial ad.");
        interstitialAd.Show();
    }
    else
    {
        Debug.LogError("Interstitial ad is not ready yet.");
    }
}
```

c. You can destroy these ads the same as you destroy Banner ads just need to take reference of Interstitial ads object.

You can also add events for Interstitial ads like when users click on ads, when ads fail to load, when ads are closed, etc.

For more details go to this page - https://developers.google.com/admob/unity/interstitial.

Reward Ads

For Reward Ads:
Example for Rewards ads code:

a. The below code shows how to Load Reward ads.

```
private RewardedAd rewardedAd;

/// <summary>
/// Loads the rewarded ad.
/// </summary>
public void LoadRewardedAd()
{
    // Clean up the old ad before loading a new one.
    if (rewardedAd != null)
    {
          rewardedAd.Destroy();
          rewardedAd = null;
    }

    Debug.Log("Loading the rewarded ad.");

    // create our request used to load the ad.
    var adRequest = new AdRequest();
    adRequest.Keywords.Add("unity-admob-sample");

    // send the request to load the ad.
    RewardedAd.Load(_adUnitId, adRequest,
        (RewardedAd ad, LoadAdError error) =>
        {
            // if error is not null, the load request failed.
            if (error != null || ad == null)
            {
                Debug.LogError("Rewarded ad failed to load an ad " +
                               "with error : " + error);
                return;
            }

            Debug.Log("Rewarded ad loaded with response : "
                      + ad.GetResponseInfo());

            rewardedAd = ad;
        });
}
```

b. Below Code shows how to show Reward Ads.

```
public void ShowRewardedAd()
{
    const string rewardMsg =
        "Rewarded ad rewarded the user. Type: {0}, amount: {1}.";

    if (rewardedAd != null && rewardedAd.CanShowAd())
    {
        rewardedAd.Show((Reward reward) =>
        {
            // TODO: Reward the user.
            Debug.Log(String.Format(rewardMsg, reward.Type, reward.Amount));
        });
    }
}
```

c. You can destroy these ads the same as you destroy Banner ads just need to take reference of the Reward ads object.

You can also add events for Reward ads like when users click on ads, when ads fail to load, when ads are closed, etc.

For more details go to this page - https://developers.google.com/admob/unity/rewarded.

Using Ads Framework

“AdsConfigData.cs” and “TestDeviceConfigData.cs”

These are scriptable scripts to config for ID’s.
“AdsConfigData.cs” is for configuring App Id’s, and AdsUnit ID’s for production and test ads.
“TestDeviceConfigData.cs” is for configuring Test Device ID’s.
You have to create a Resource folder and add scriptable objects in it.
For adding scriptable objects, right click inside of "Resources" folder, then click on "Create" option on top.
You will see two options on top "AdsData" and "DeviceIdsData".
Click on both to create scriptable objects and set config data.
Set test and live Ids according to requirements.

"AdsInitializer.cs"

This script is to initialize the mobile ads sdk.
Attach "AdsInitializer.cs" to gameobject.
You can turn on Debug on this script.

“BannerAdsManager.cs” , “InterstitialAdsManager.cs” and “RewardAdsManager.cs”

These are the scripts for loading and showing respective ads.
There are methods to load, show, destroy and add listeners for events.

Sample

This framework is also attached with samples that have implementations for all three ads.
The “AdsManager” game object has all the required scripts and one sample script “SampleAdManager.cs” is also attached.
Scriptable objects are created and test AdsUnitIds are also configured.
All basic methods are called and some events are also invoked according to the requirements.
Please use this sample to understand the usability.

References

Google Mobile Ads - https://developers.google.com/admob.