유니티 애드몹 탑재 가이드

1. 들어가기 전에

본 가이드는 **유니티 프로젝트에 대한 가이드**로, **애드몹을 탑재하는 과정 및 방법을 설명하기 위한 샘플 프로젝트 가이드**입니다. 이 가이드에서는 **안드로이드 앱에 대한 내용만 다루며**, iOS에 대한 내용은 다루지 않습니다. 모든 내용은 Mac 환경에서 작성되었습니다.

본 유니티 애드몹 샘플에서는 다음과 같은 스펙을 기준으로 사용합니다.

스펙은 개발환경에 따라 달라질 수 있습니다.

- Unity 버전: 2022.1
- Java 버전: 11
- Minimum Android Target 버전: 21 이상
- Target Android 버전: 33 이상

---

2. 애드몹 회원가입, APP ID 발급 및 AD Unit ID 발급

애드몹 배너 광고를 붙이기 위해서는 애드몹 회원가입부터 진행을 하여야 합니다. 회원가입을 한 후에는 애드몹 UI에서 APP 프로젝트를 만들어 APP ID 를 발급 받고, 이어서 AD Unit ID 를 발급 받아야 합니다. 여기에서는 **배너광고에 대한 AD Unity ID를 발급 받고 사용하는 방법에 대해서만 다룹니다.** 기타 다른 광고 유형에 대한 사용 방법은 애드몹 공식 가이드(광고 단위, 광고 형식, 광고 유형 - Google AdMob 고객센터) 를 참고해주시기 바랍니다.

2-1. 애드몹 회원가입

1) Google AdMob: 모바일 앱 수익 창출 접속
2) 가입 클릭
3) (가입 진행 후) 로그인

2-2. APP ID 발급

1) 왼쪽 메뉴 중 앱 > 앱 추가
2) 플랫폼 Android 선택 > "지원되는 앱 스토어에 앱이 등록되어 있나요?" 에 "예" 혹은 "아니요" 선택 > 계속
3) 앱 이름 입력 > 사용자 측정항목 설정 > 앱 추가 클릭
4) 완료 클릭

2-3. APP ID 확인

1) 왼쪽 메뉴 중 앱 > 모든 앱 보기 클릭
2) ca-app-pub으로 시작하는 APP ID 메모(이후 유니티 프로젝트 설정 단계에서 사용 되므로 따로 메모)

2-4. AD Unit ID 발급

주의사항1) 테스트(혹은 디버그) 환경에서는 아래와 같이 **테스트 AD Unit ID를 사용**하거나, 혹은 **실제 AD Unity ID를 사용하되, 테스트 디바이스를 등록**하여 사용하여야 합니다. 실제 AD Unit ID를 사용하는 방법은 **주의사항2)** 에서, 테스트 디바이스를 등록하는 방법은 다음 단계인 **주의사항3)** 에서 다룹니다.

<안드로이드>

광고 형식	샘플 광고 단위 ID
앱 오프닝 광고	ca-app-pub-3940256099942544/9257395921

적응형 배너	ca-app-pub-3940256099942544/9214589741
배너	ca-app-pub-3940256099942544/6300978111
전면 광고	ca-app-pub-3940256099942544/1033173712
동영상 전면 광고	ca-app-pub-3940256099942544/8691691433
보상형 광고	ca-app-pub-3940256099942544/5224354917
보상형 전면 광고	ca-app-pub-3940256099942544/5354046379
네이티브 광고 고급형	ca-app-pub-3940256099942544/2247696110
네이티브 동영상 광고 고급형	ca-app-pub-3940256099942544/1044960115

<iOS>

광고 형식	샘플 광고 단위 ID
App Open	ca-app-pub-3940256099942544/5575463023
Adaptive Banner	ca-app-pub-3940256099942544/2435281174
Banner	ca-app-pub-3940256099942544/2934735716
Interstitial	ca-app-pub-3940256099942544/4411468910
Interstitial Video	ca-app-pub-3940256099942544/5135589807
Rewarded	ca-app-pub-3940256099942544/1712485313
Rewarded Interstitial	ca-app-pub-3940256099942544/6978759866
Native Advanced	ca-app-pub-3940256099942544/3986624511
Native Advanced Video	ca-app-pub-3940256099942544/2521693316

주의사항2) Release 환경에서는 아래의 절차에 따라 **실제 AD Unit ID를 발급하여 사용**해야 합니다.
1) 왼쪽 메뉴 중 앱 > 앱 클릭
2) 왼쪽 메뉴 중 광고 단위 > 시작하기 클릭
3) 광고 형식에서 배너 선택
4) 광고 단위 이름 입력 > 광고 단위 만들기 클릭
5) 광고 단위 생성 완료
6) **ca-app-pub으로 시작하는 광고 단위 ID**를 복사(혹은 메모할 것. 이후 유니티 프로젝트 설정 단계에서 사용됨) > 완료

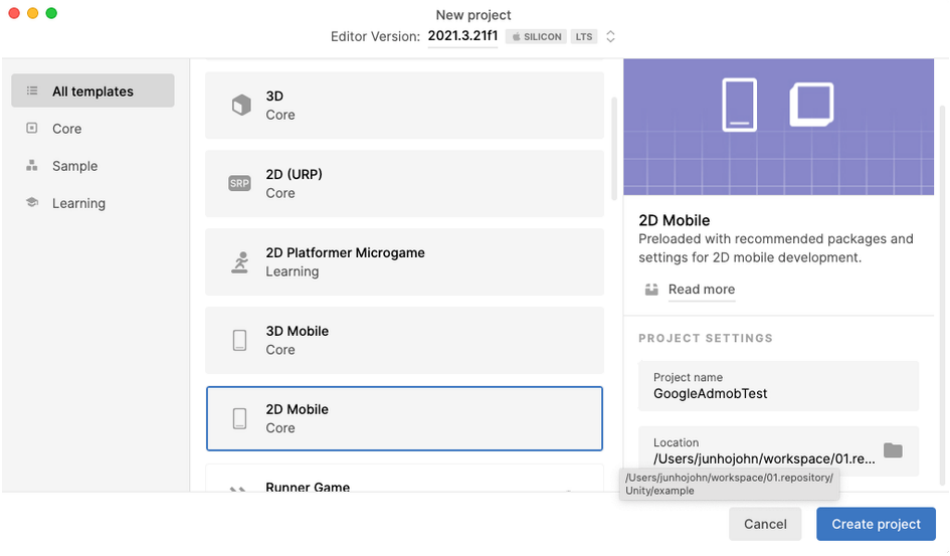
주의사항3) 테스트 디바이스 등록 방법
주의사항2) 에서 확인한 실제 AD Unity ID를 테스트(혹은 디버그) 환경에서 사용하려면, 반드시 테스트 디바이스를 등록하여 사용하여야 합니다. 등록 방법은 2가지가 있습니다. AdMob UI에서 추가하거나 Google Mobile Ads SDK를 사용하여 프로그래매틱 방식으로 추가 가능합니다. **여기서는 AdMob UI에서 추가하는 방법을 다룹니다.**
1) Google AdMob: 모바일 앱 수익 창출 접속
2) 왼쪽 설정 메뉴 클릭 > 기기 테스트 > 테스트 기기 추가
3) 기기 이름 입력 > 플랫폼 Android 선택 > Device 광고 ID 입력 > 광고 검사기 > 흔들기 선택 > 저장
4) 테스트 디바이스 등록 완료

3. 유니티 프로젝트 생성

1) Unity Hub 열기 > New Project

2) Unity 버전 선택

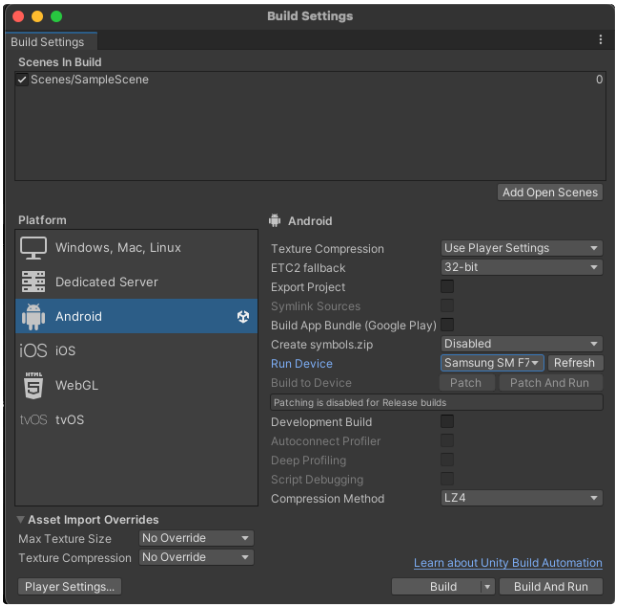
3) 2D Mobile > 프로젝트 명 입력 > 프로젝트 루트 경로 지정 > Create



---

4. 프로젝트 실행 설정

File > Build Settings > Android 선택 > Switch Platform



---

5. Admob Unity Package Import 설정

5-1. Package 다운로드 및 Import

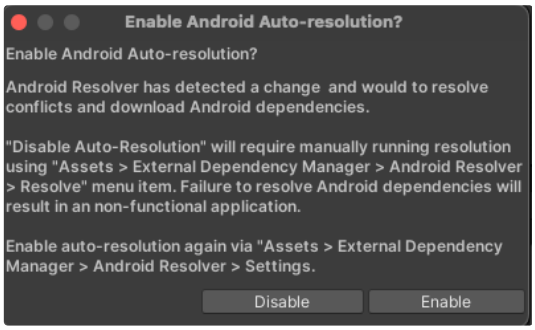
1) 아래 링크에서 Admob Package 다운로드

Releases · googleads/googleads-mobile-unity

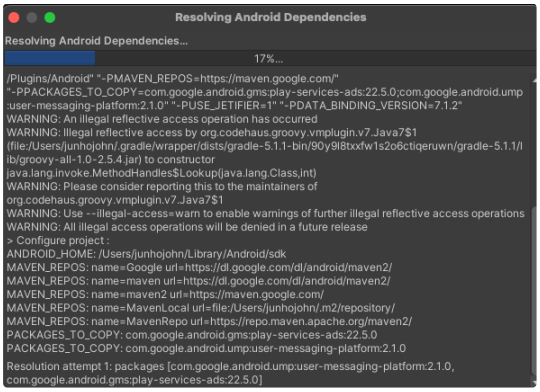
2) Assets > Import Package > Custom Package > 다운받은 Admob Package 선택

3) All 선택 > Import

4) Enable Android Auto-resolution? 창에서 Enable 선택



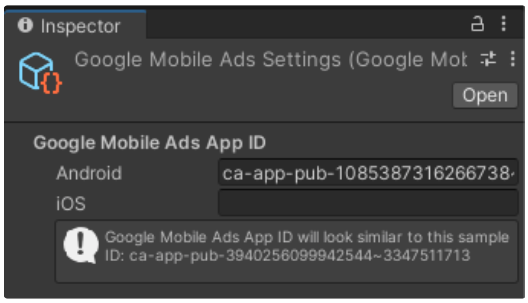
5) Android Dependencies 자동으로 다운로드 진행 중



5-2. Admob App ID 설정

1) Assets > Google Mobiles Ads > Settings

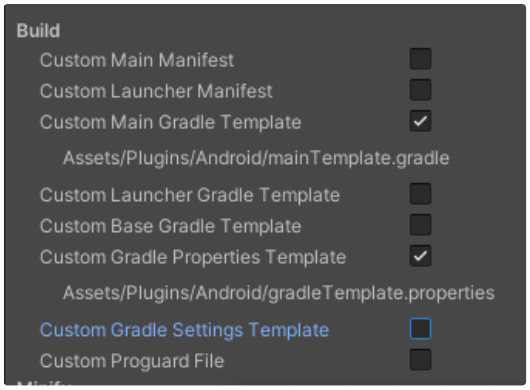
2) Android에 App ID 입력. App ID는 **2-3. APP ID 확인** 단계에서 메모한 App ID를 입력



5-3. custom gradle 파일 설정

1) Player Settings… > Player > 안드로이드 아이콘 탭 > Publishing Settings > Build 에서 Custom Main Gradle Template 체크

2) Custom Gradle Properties Template 체크



---

6. 예제 파일 구현

1) Project 뷰 > Assets 선택 > 빈 공간 우 클릭 > Create > C # Script > 파일 명을 AdmobTest 로 입력 후 Enter

2) AdmobTest.cs 파일 우 클릭 > Open(External Editor로 열림)

3) 아래와 같이 구현 후 저장

```
using GoogleMobileAds.Api;
using System;
using UnityEngine;

public class AdmobTest : MonoBehaviour
{

      // These ad units are configured to always serve test ads.
    #if UNITY_ANDROID
      private string _adUnitId = "ca-app-pub-0000000000000000/0000000000"; // 2-4. AD Unit ID 발급 에서 발급 받은
    #elif UNITY_IPHONE
      private string _adUnitId = "ca-app-pub-0000000000000000/0000000000";
    #else
      private string _adUnitId = "unused";
    #endif

      BannerView _bannerView;


    /// <summary>
    /// Creates a 320x50 banner view 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)
        {
            DestroyBannerView();
        }

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

    /// <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();

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


    /// <summary>
    /// listen to events the banner view may raise.
    /// </summary>
    private void ListenToAdEvents()
    {
        // Raised when an ad is loaded into the banner view.
        _bannerView.OnBannerAdLoaded += () =>
        {
            Debug.Log("Banner view loaded an ad with response : "
                + _bannerView.GetResponseInfo());
        };
        // Raised when an ad fails to load into the banner view.
        _bannerView.OnBannerAdLoadFailed += (LoadAdError error) =>
        {
            Debug.LogError("Banner view failed to load an ad with error : "
                + error);
        };
        // Raised when the ad is estimated to have earned money.
        _bannerView.OnAdPaid += (AdValue adValue) =>
        {
            Debug.Log(String.Format("Banner view paid {0} {1}.",
                adValue.Value,
                adValue.CurrencyCode));
        };
        // Raised when an impression is recorded for an ad.
        _bannerView.OnAdImpressionRecorded += () =>
        {
            Debug.Log("Banner view recorded an impression.");
        };
        // Raised when a click is recorded for an ad.
        _bannerView.OnAdClicked += () =>
        {
            Debug.Log("Banner view was clicked.");
        };
        // Raised when an ad opened full screen content.
        _bannerView.OnAdFullScreenContentOpened += () =>
        {
            Debug.Log("Banner view full screen content opened.");
        };
```

```
97          // Raised when the ad closed full screen content.
98          _bannerView.OnAdFullScreenContentClosed += () =>
99          {
100             Debug.Log("Banner view full screen content closed.");
101         };
102     }
103
104     /// <summary>
105     /// Destroys the banner view.
106     /// </summary>
107     public void DestroyBannerView()
108     {
109         if (_bannerView != null)
110         {
111             Debug.Log("Destroying banner view.");
112             _bannerView.Destroy();
113             _bannerView = null;
114         }
115     }
116
117
118     public void Start()
119     {
120         // Initialize the Google Mobile Ads SDK.
121         MobileAds.Initialize((InitializationStatus initStatus) =>
122         {
123             // This callback is called once the MobileAds SDK is initialized.
124             LoadAd();
125         });
126     }
127
128 }
129
```

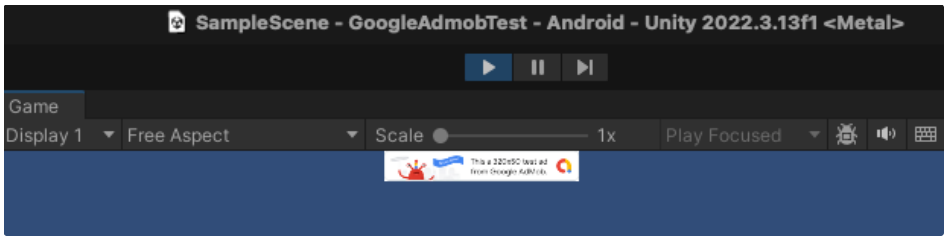
---

7. 실행 및 결과 확인

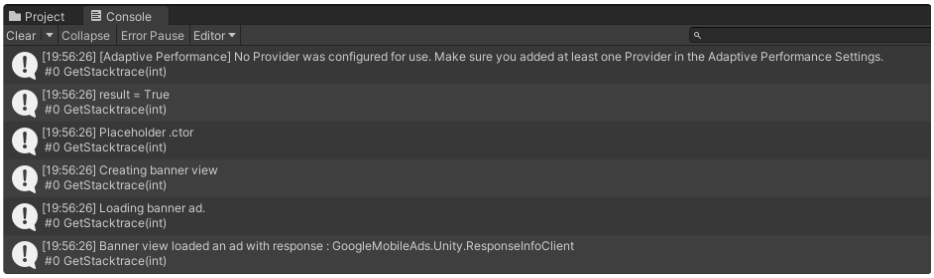
7-1. 유니티 기본 게임으로 실행할 경우

1) 맨 위 플레이 아이콘 클릭

2) 테스트 리워드 광고가 출력되는 것이 확인됨



3) 또한 Console 창에 다음과 같이 Debug.Log() 내용이 출력되는 것이 확인됨



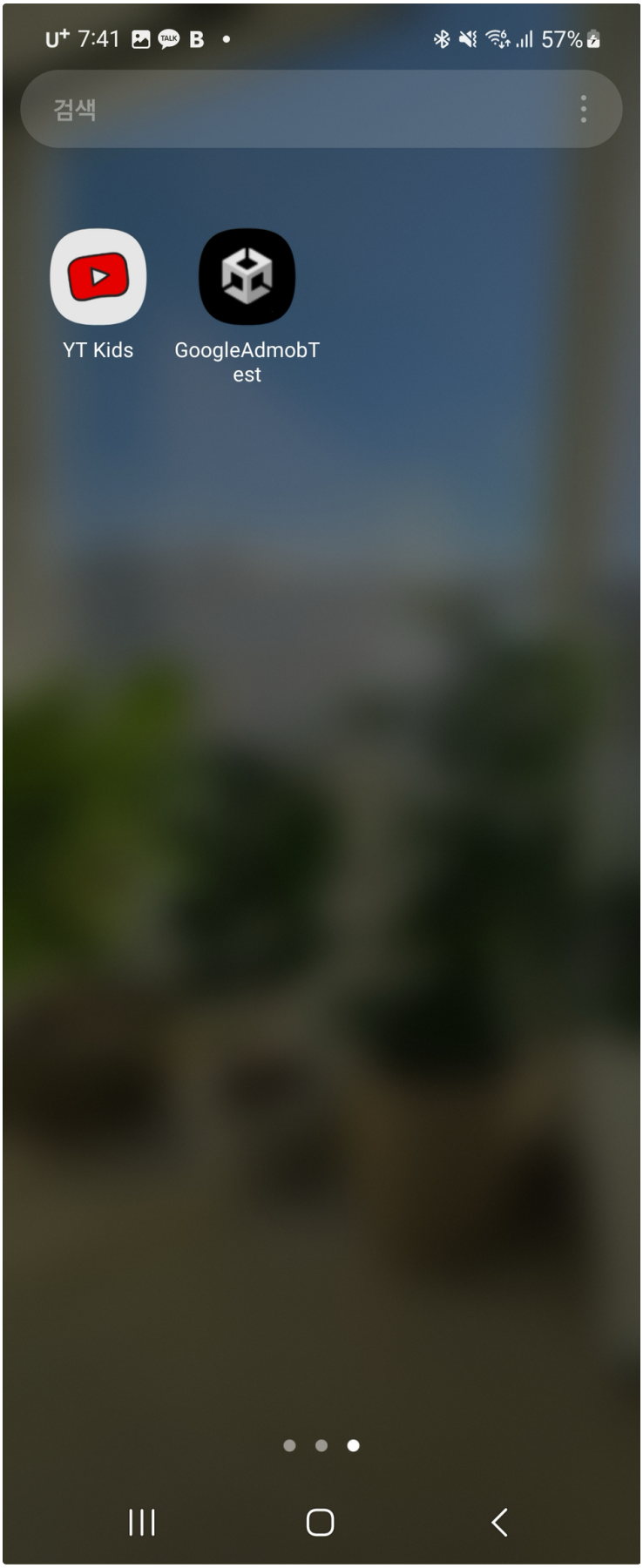
7-2. 안드로이드 실 디바이스로 실행할 경우

1) 디바이스 USB 연결 후 > Build Settings

2) Run Device 에서 실 디바이스 선택 > Build And Run

3) APK 명 입력 > Save

4) 테스트 디바이스에 apk가 자동설치됨



5) 테스트 리워드 광고가 출력되는 것 확인