

GLS-Libraryの使い方

概要	2
開発環境	2
LTSとは？	2
導入方法	3
1. 開発に使用するプロジェクトのバージョンを確認する	3
2. プロジェクトのUnityバージョンのアップグレード(必要な場合)	4
Unity Hubを使用している場合	4
Unity Hubを使用していない場合	5
3. 開発プラットフォームをAndroidに設定する	6
PlatformからAndroidが選択できない場合	7
4. プロジェクトにパッケージをインポートする	8
使い方	10
準備	10
Advertisementのインストール	10
Prefabの配置	11
UnityProjectIDとのリンク	12
GameIDの設定	14
各種広告の表示	16
Placement IDについて	16
Placementの作成	16
インタースティシャル広告	18
Placementの設定	18
Placement IDの指定	20
インタースティシャル広告の表示	20
サンプルコード	20
動画リワード広告	21
Placementの設定	21
Placement IDの指定	23
サンプルコード	24
バナー広告	25
Placementの設定	25
PlacementIDの指定	27
バナー広告の表示	27
バナー広告の非表示	27
サンプルコード	28
カスタムイベントの発行	29
イベントの発行	29
サンプルコード	29
UIのセーフエリア対応	30
セーフエリア対応とは？	30
GLSSafeAreaCanvasの使い方	30

概要

このドキュメントは「GLS-Library-20200903.unitypackage」の導入方法や機能についての解説を記載しています。

開発環境

- Unity2019.4.x (LTS) <推奨>
 - Unity2019.4.0、Unity2019.4.9 など
- Unity2020.1 でも動作確認済み

Unity2018以前のバージョンでは動作しません。

LTSとは？

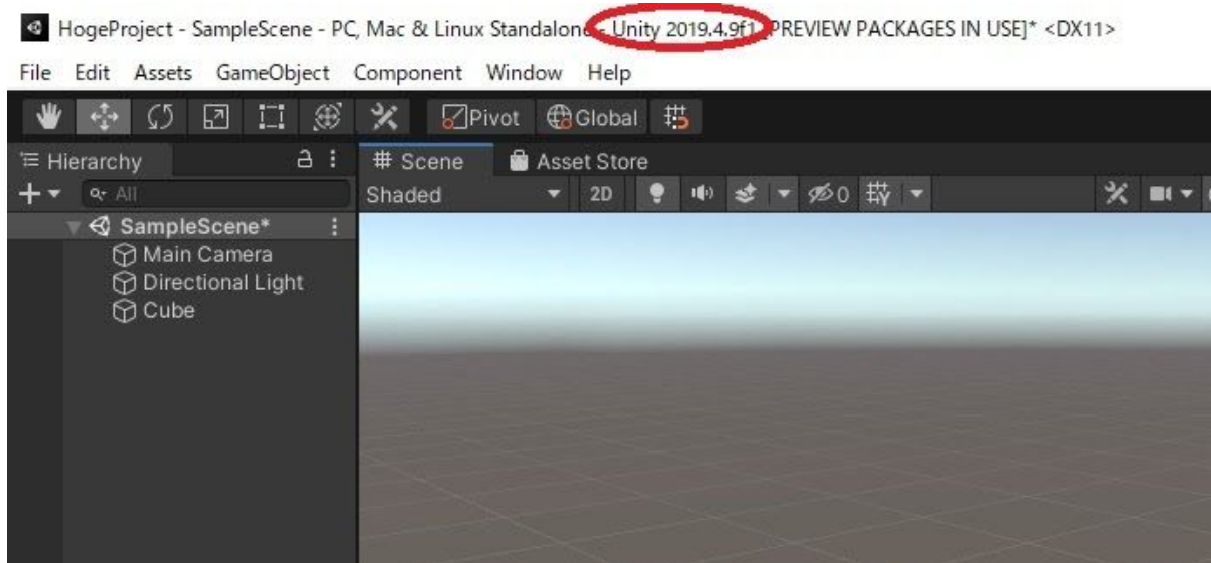
「Long-Term Support」の略。LTSがついたバージョンは2年間の長期サポート期間が設けられており、この期間内はUnityエディタ側に何か問題があった場合でもアップデートで解消される可能性が高い。要は**安定版**のこと。

導入方法

1. 開発に使用するプロジェクトのバージョンを確認する



Unity Hubを使用している場合は、プロジェクト名の右側の「Unityバージョン」から確認することが出来ます。

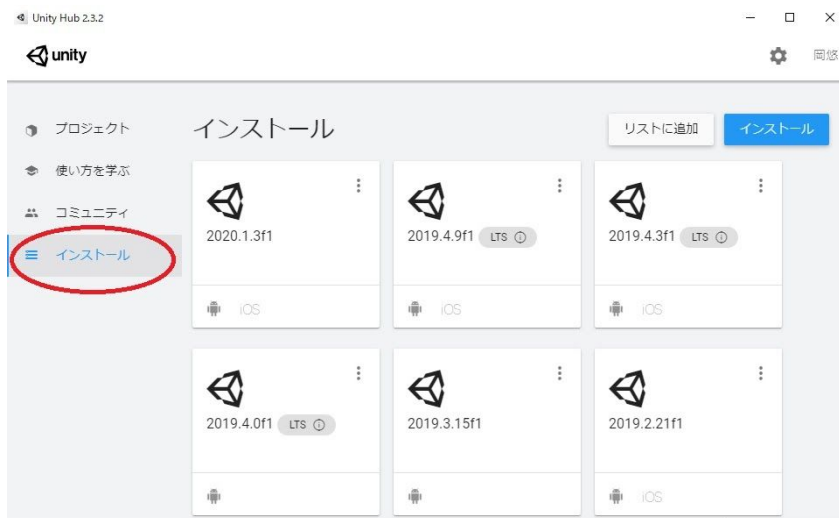


プロジェクトを開き、エディターの画面上部のバージョン表記を見ることでも確認することが可能です。

確認したUnityのバージョンが**2019.4.x (LTS)**より下のバージョンだった場合、次の手順で**プロジェクトのUnityバージョンをアップグレードしてください。**

2. プロジェクトのUnityバージョンのアップグレード(必要な場合)

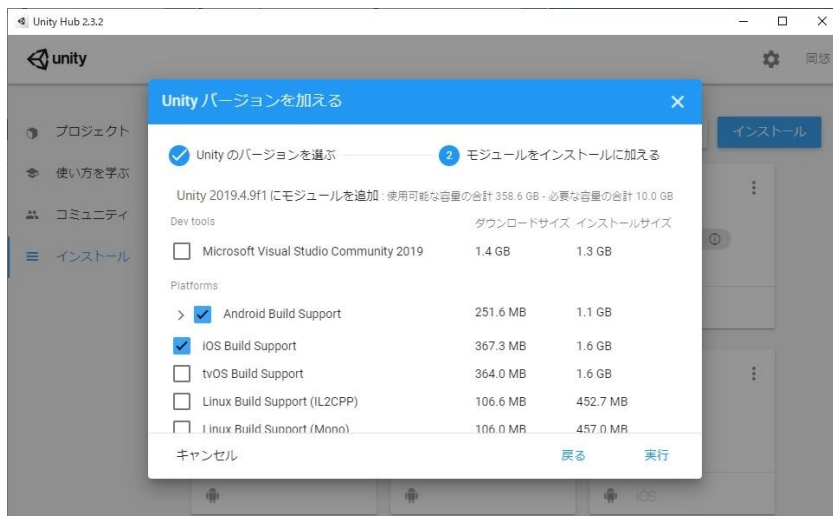
Unity Hubを使用している場合



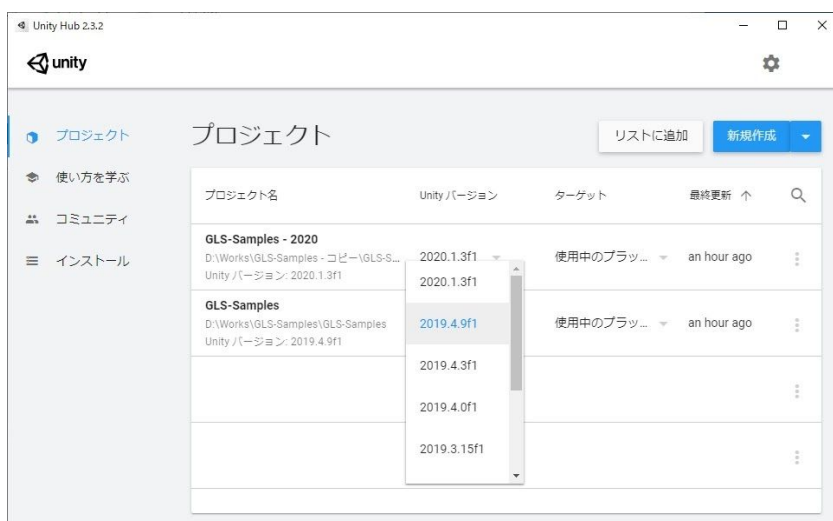
Unity Hubを開き、左側のリストから「インストール」を選択します。



右側の「インストール」をクリックし、「Unityのバージョンを加える」から **Unity2019.4.x (LTS)** を選択します。



必要なモジュールを選択した後、「実行」をクリックし、インストールします。

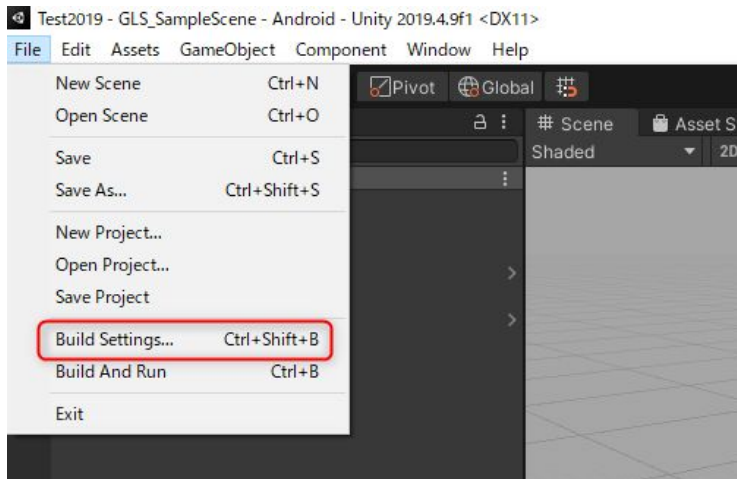


左側のリストから「プロジェクト」に戻り、開発に使うプロジェクトの「Unityバージョン」を開き、**Unity2019.4.x (LTS)**を選択し、プロジェクトを開くとアップグレードできます。

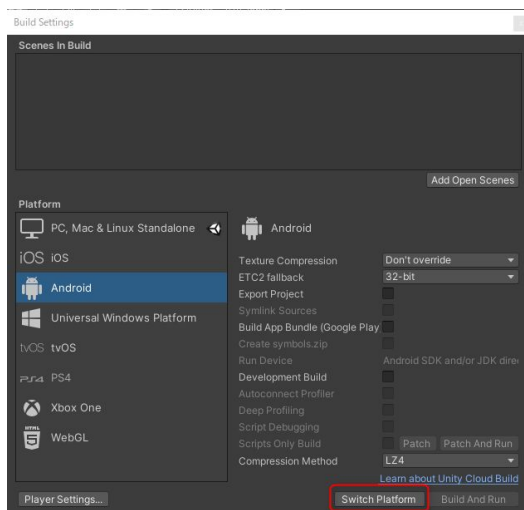
Unity Hubを使用していない場合

- <https://unity3d.com/jp/unity/ga/lts-releases>から**Unity 2019.4.x (LTS)**をダウンロードし、インストールします。
- インストールしたUnityを起動し、「Open」から開発に使うプロジェクトを開くとアップグレードできます。

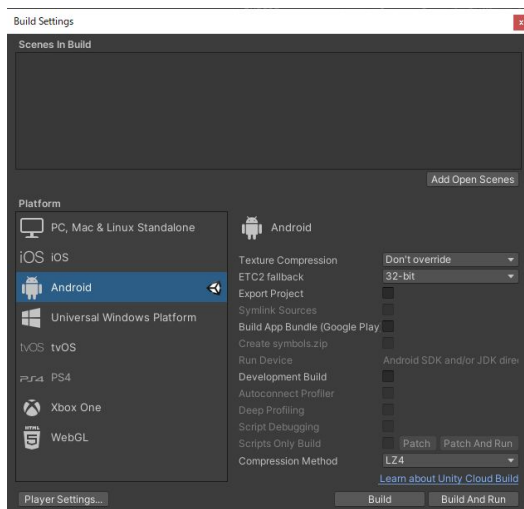
3. 開発プラットフォームをAndroidに設定する



画面上部の「Window」から「Build Settings」を選択します。

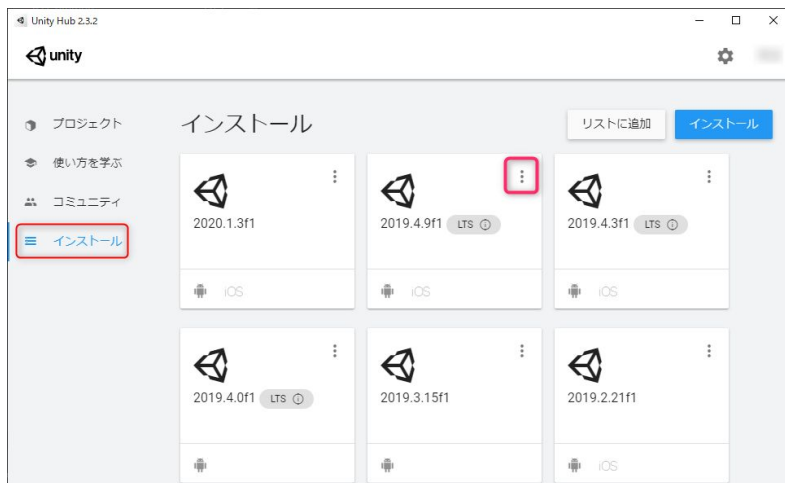


左側の「Platform」から**Android**を選択し、「Switch Platform」を押します。

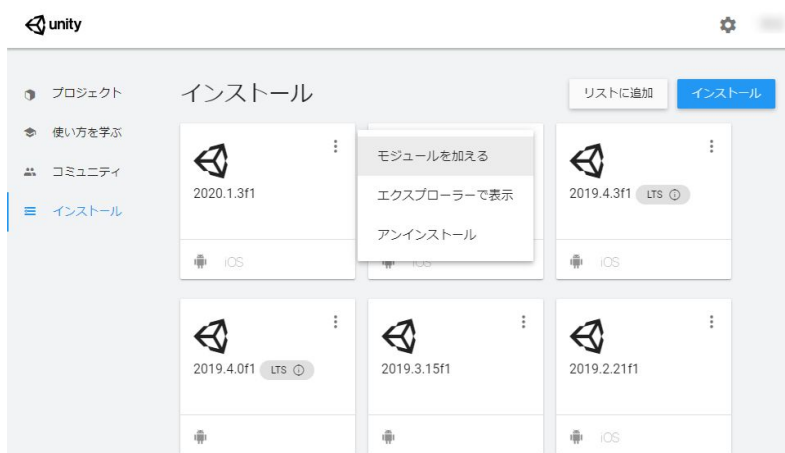


しばらく待った後、「Platform」のAndroidの欄にマークがついていれば完了です。

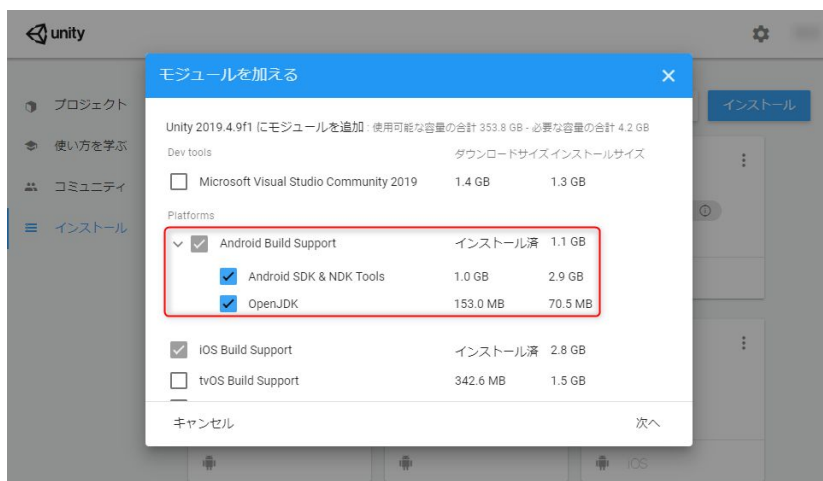
PlatformからAndroidが選択できない場合



UnityにAndroidのモジュールをインストールします。
まず、UnityHubを開き、左側のリストから「インストール」を選択します。

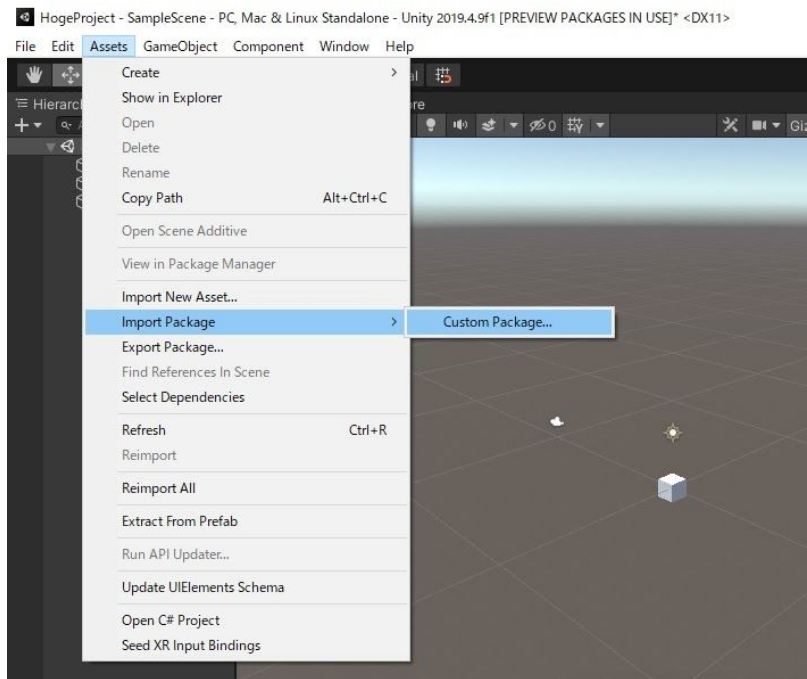


開発に使用しているバージョンで「モジュールを加える」をクリックします。

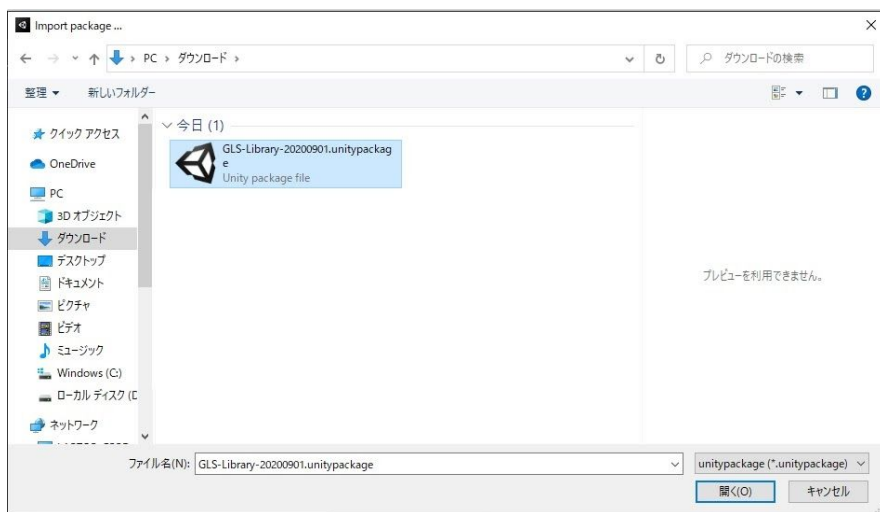


「Android Build Support」「Android SDK & NDK Tools」「OpenJDK」にチェックをつけて「次へ」をクリックします。後は画面の指示に従えばOKです。

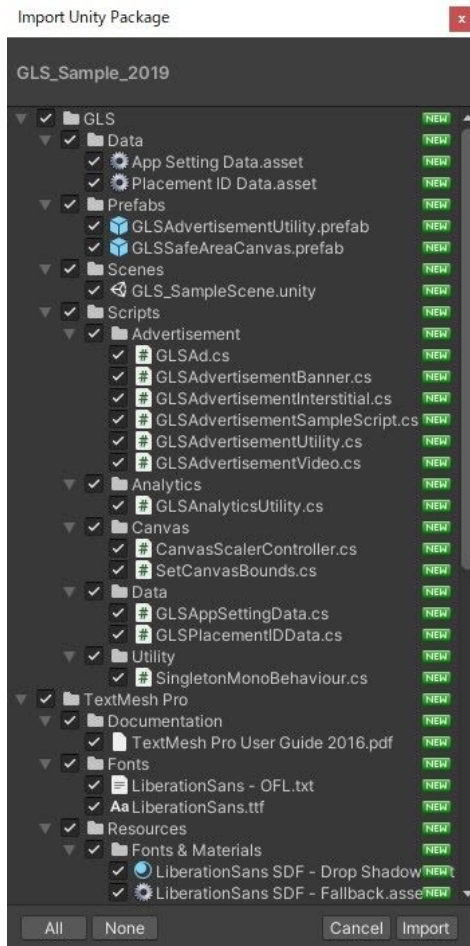
4. プロジェクトにパッケージをインポートする



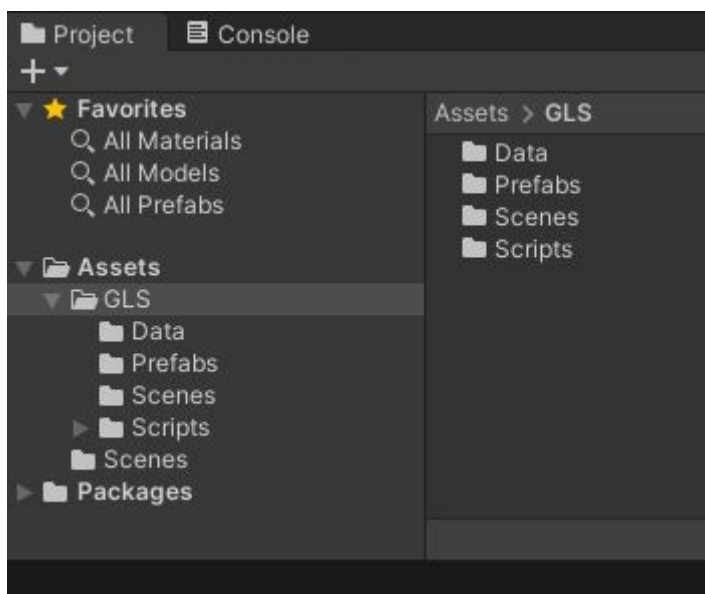
画面上部の「Assets」から「Import Package -> Custom Package」を選択します。



「GLS-Library-xxxxxxx.untypackage」を選択し、「開く」をクリックしてください。



全てのチェックマークにチェックがついていることを確認(「All」を押せばより確実です)し、右下の「Import」を押します。

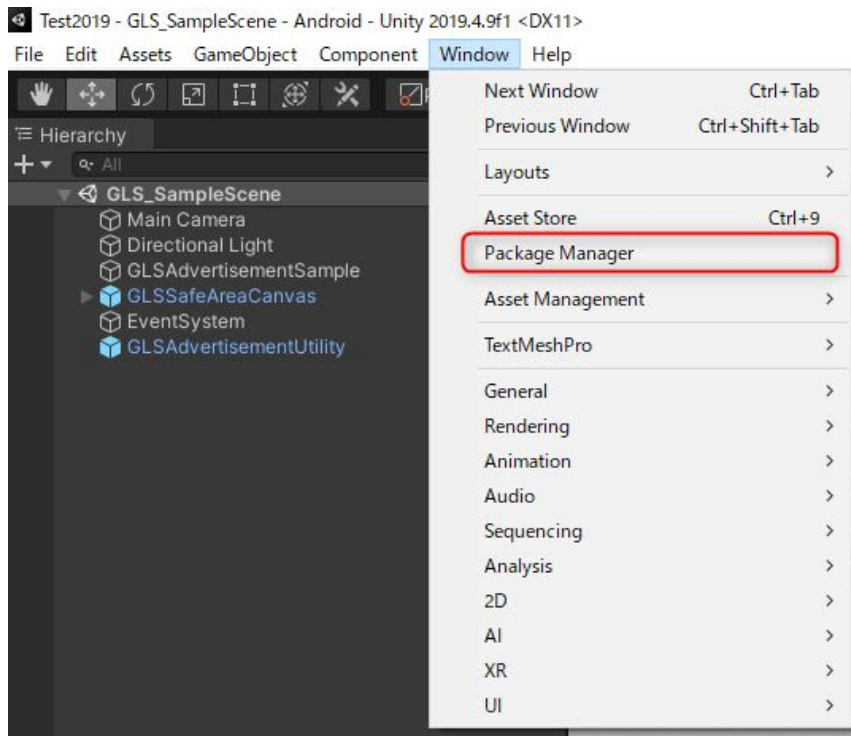


Projectウィンドウを開き、Assetsフォルダ直下に「GLS」フォルダが追加されていることが確認出来れば導入完了です。

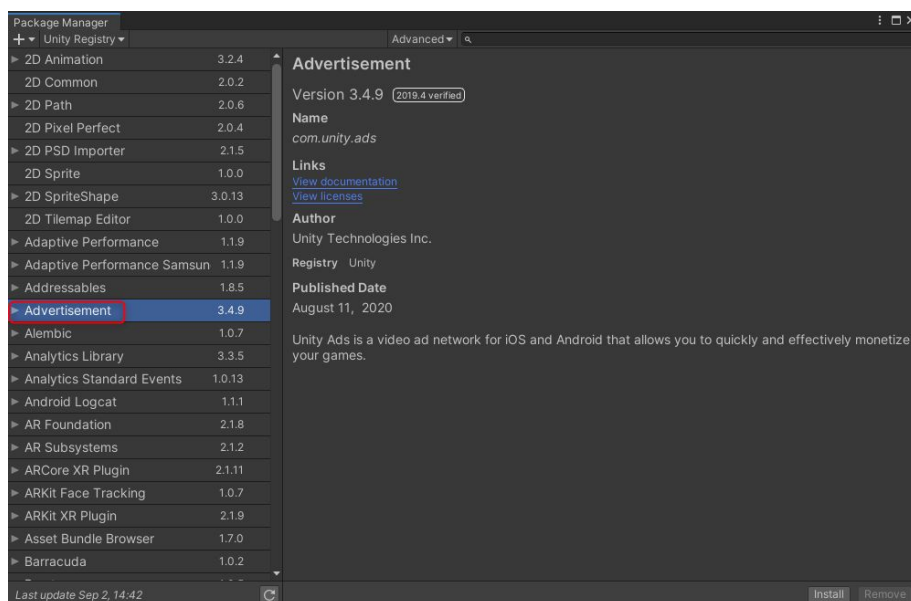
使い方

準備

Advertisementのインストール

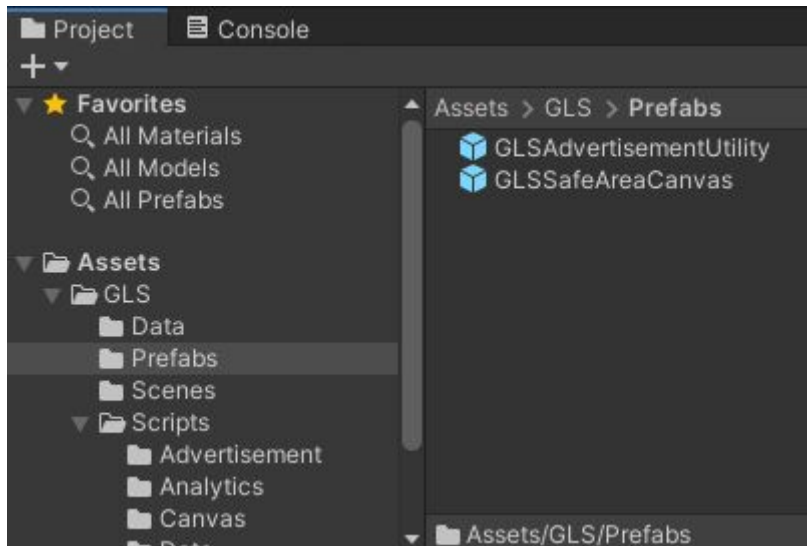


画面上部の「Window」から「PackageManager」を選択します。

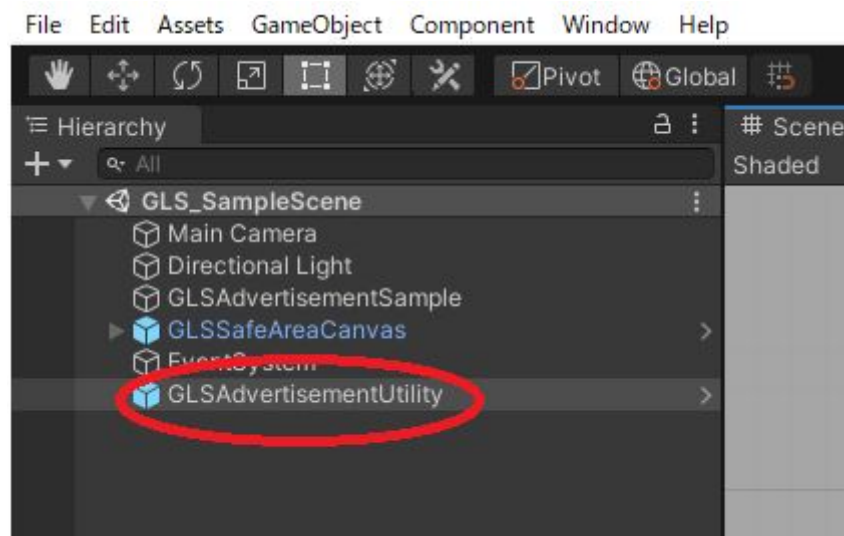


左側の一覧から「Advertisement」を探し、右下の「Install」を押せばインストール完了です。

Prefabの配置

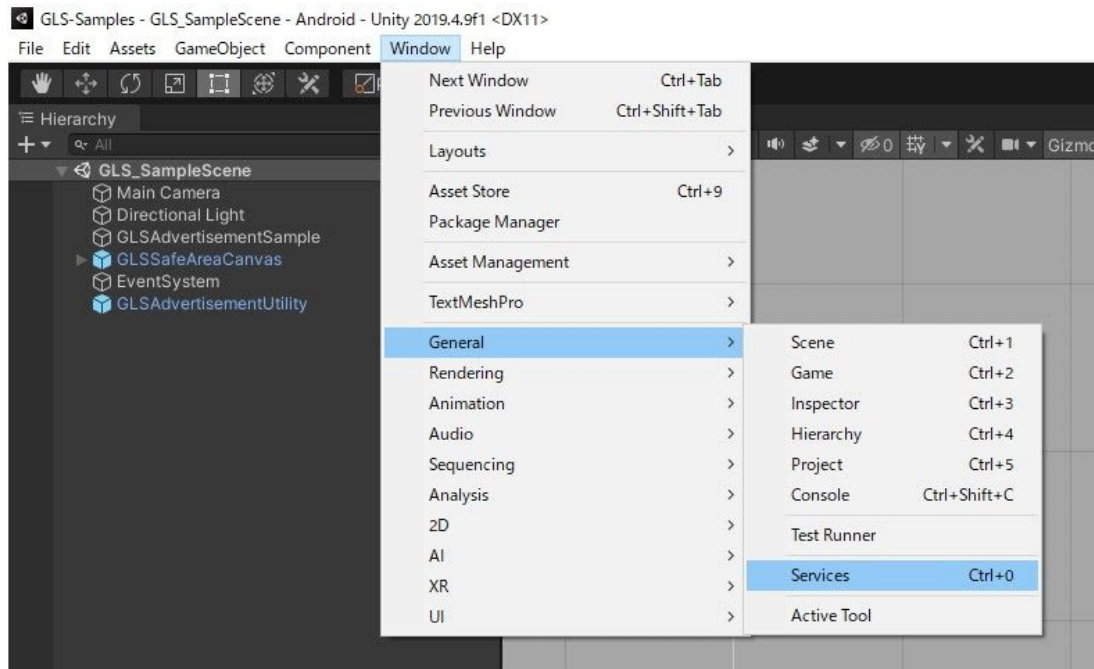


Projectウィンドウを開き、「**Asset -> GLS -> Prefabs**」内にある「**GLSAdvertisementUtility**」プレハブをHierarchy上にドラックアンドドロップします。

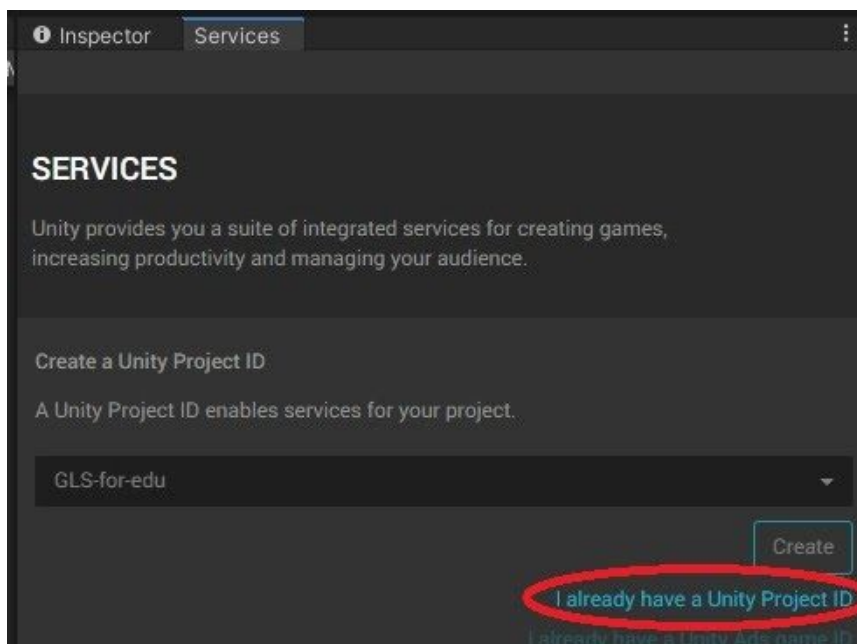


Scene内にGLSAdvertisementUtilityプレハブが配置されていればOKです。

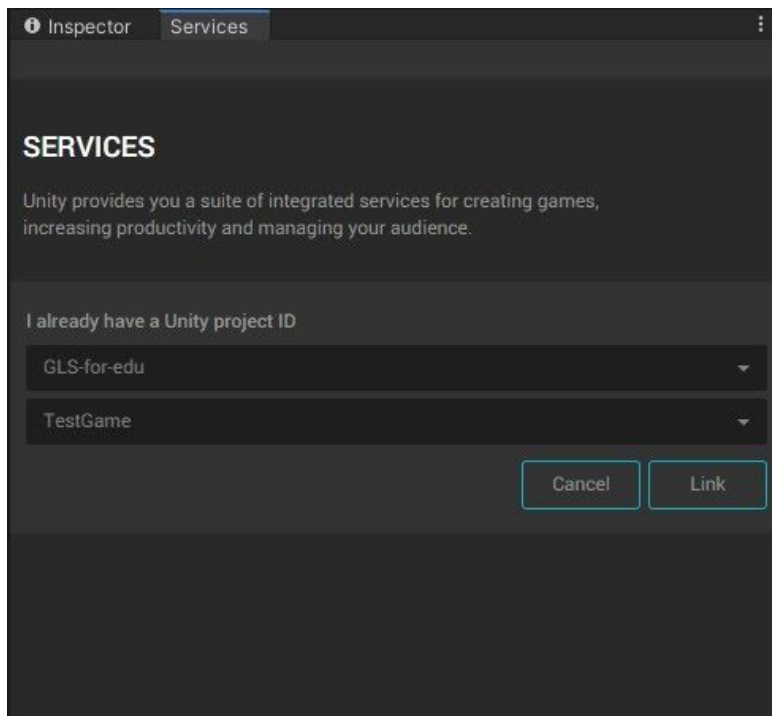
UnityProjectIDとのリンク



画面上部の「**Window**」から「**General -> Services**」と選びServiceタブを開きます。



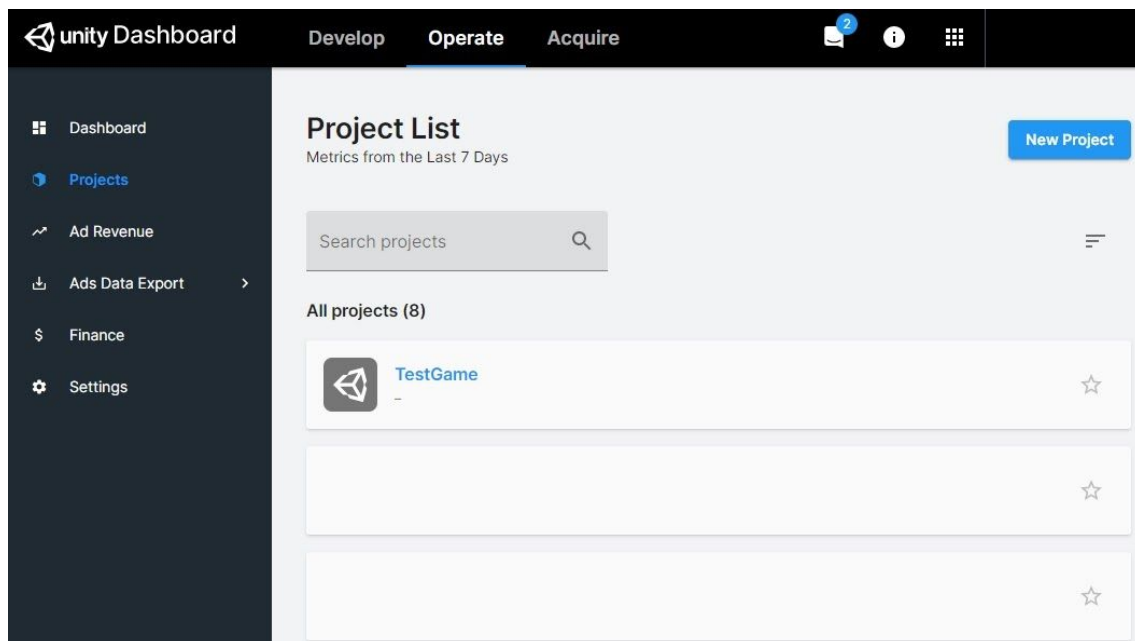
Serviceタブ内の右下にある「**I already have a Unity Project ID**」をクリックします。



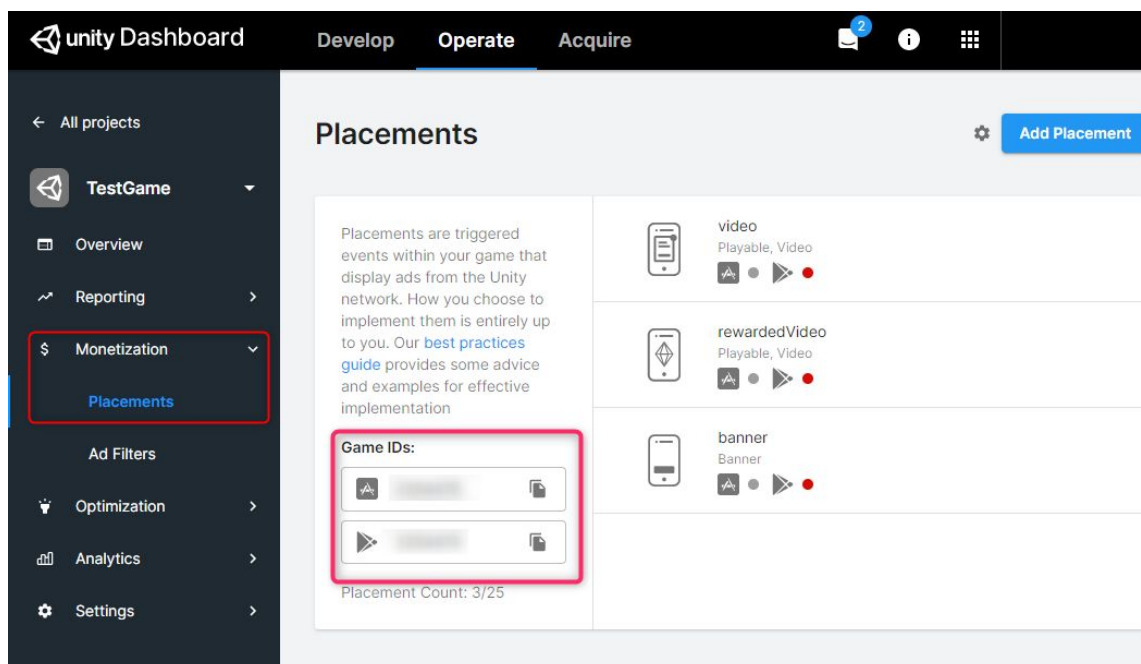
上の欄には「**GLS-for-edu**」、下の欄にはUnity**自分のゲームのプロジェクト名**を選択し、「**Link**」を押してください。

（もし、下の選択欄に自分のゲームのプロジェクト名が表示されていない場合はメンターにご連絡ください）

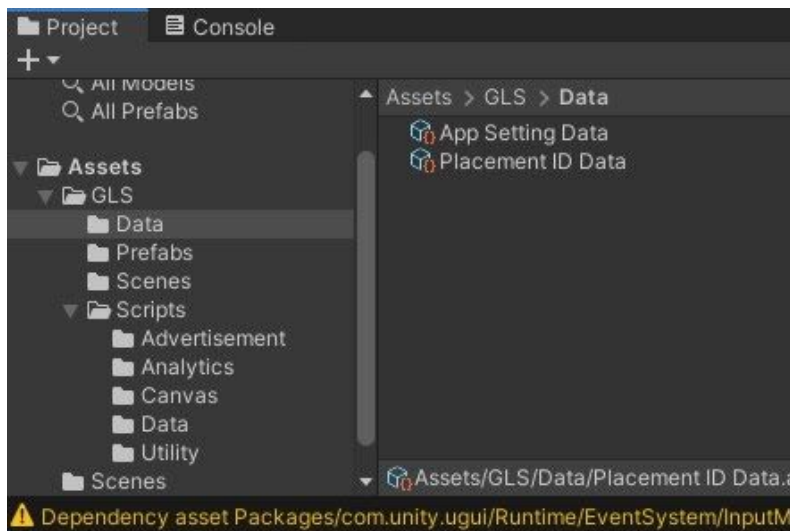
GameIDの設定



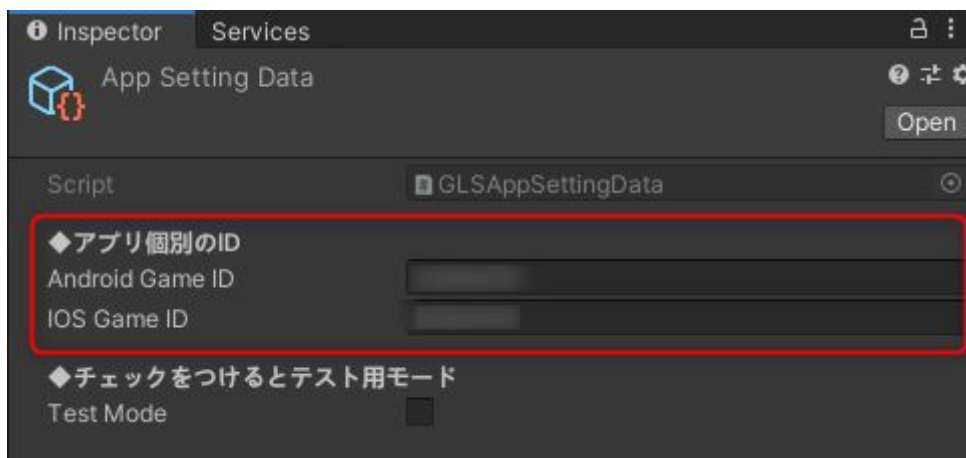
Unityダッシュボードを開き、自分のゲームのプロジェクトを選択します。



左側のリストから「**Monetization -> Placements**」を選択し、**GameIDs**からプラットフォームごとのIDを確認してUnityエディターに戻ります。



プロジェクトウィンドウから「**Assets -> GLS -> Data -> App Setting Data**」を開きます。



Android GameIDと**iOS GameID**に先ほど確認したIDを設定します。
(初期値には動作テスト用プロジェクトのIDが指定されています)

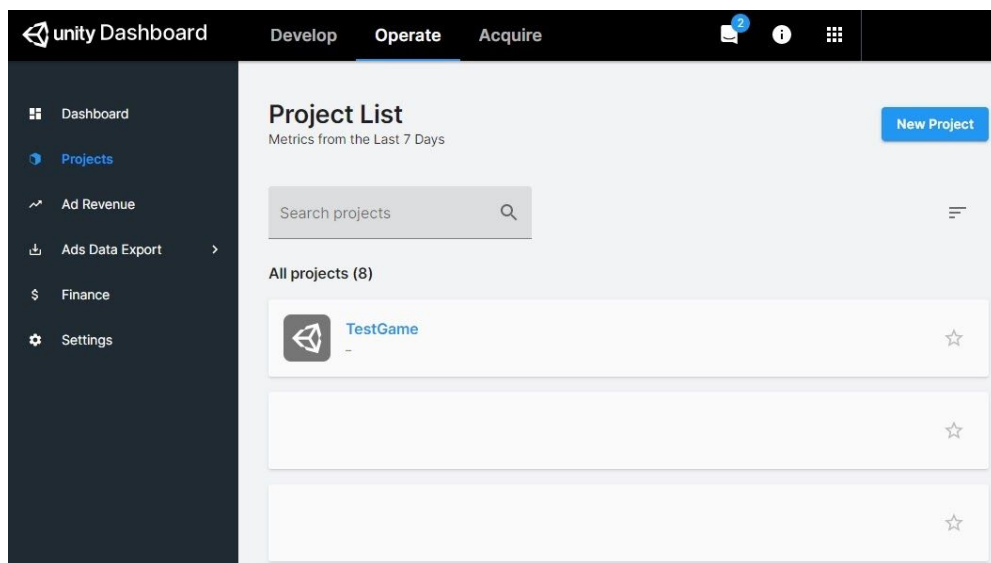
各種広告の表示

Placement IDについて

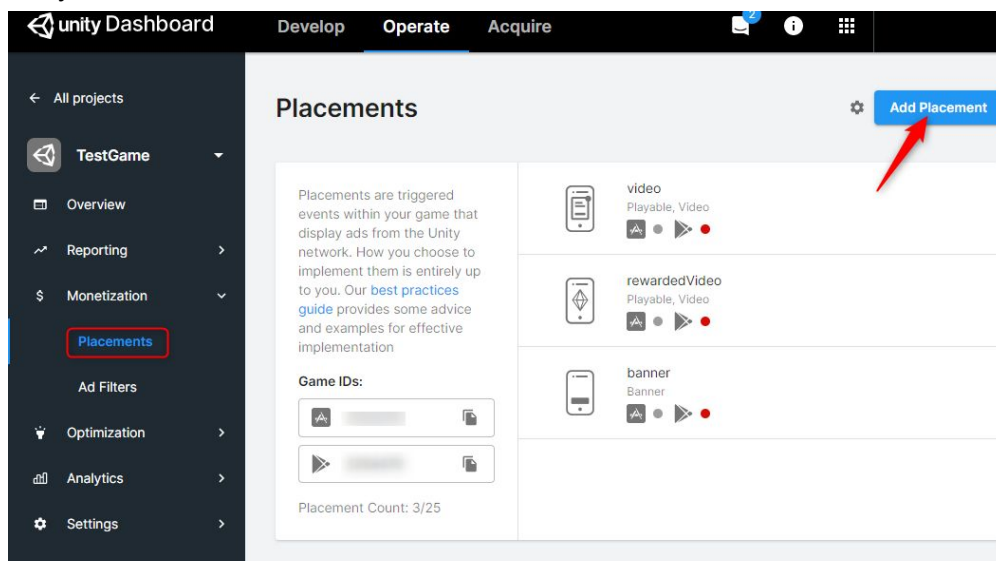
Unityのスクリプト上からリワード広告の実装を行う際、Placementの**Placement ID**が必要になります。

プロジェクトにはすでにそれぞれの種類の広告のPlacementが作成されているので、基本的にはそれらのPlacement IDを使用し、必要に応じてPlacementを新規作成してください。

Placementの作成



Unityダッシュボードを開き、自分のゲームのプロジェクトを選択します。




左側のリストから「**Monetization -> Placements**」を開き、「**Add Placement**」を選択します。


by Cookies


Add new placement (3/25)

Placement ID*
e.g. LevelComplete

Placement type

**Rewarded video**
Full screen ads that offer the user a reward, typically in the form of in game currency, for completing the video in full.

**Interstitial video**
Full screen ads appearing at natural breaks in the game. Users can choose to skip these ads.

**Banner**
Banner ads are a form of smaller sized graphical ads, typically including static images & text to convey a marketing message.

Cancel [Create placement](#)

作成するPlacementのパラメータを設定します(後で変更不可)。

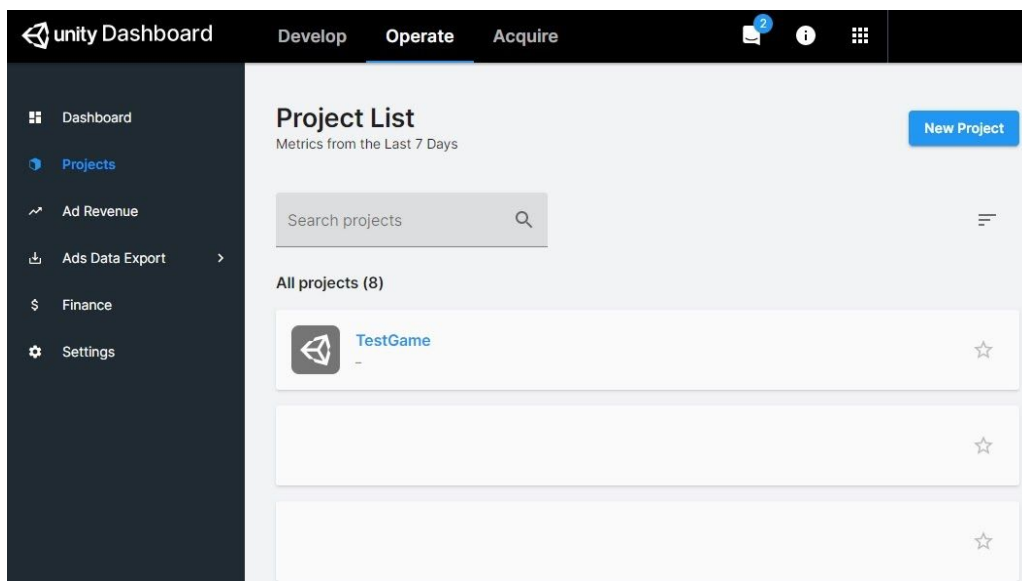
- **Placement ID** : Placement IDの名前を入力します。
- **Placement type** : Placement IDに設定したい広告の種類を入力します。

入力が終了した後、「[Create placement](#)」をクリックするとPlacement IDが作成されます。

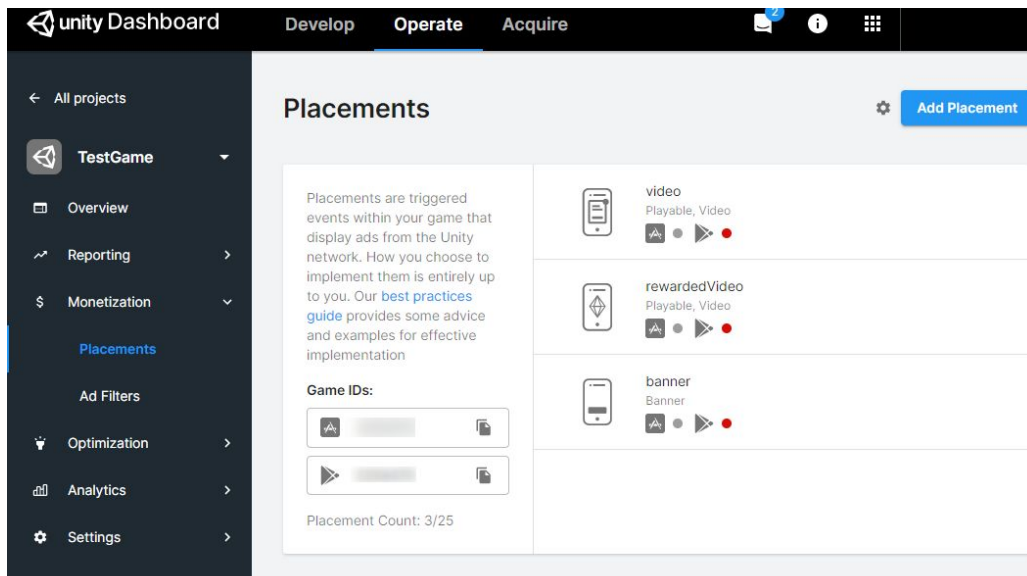
インタースティシアル広告



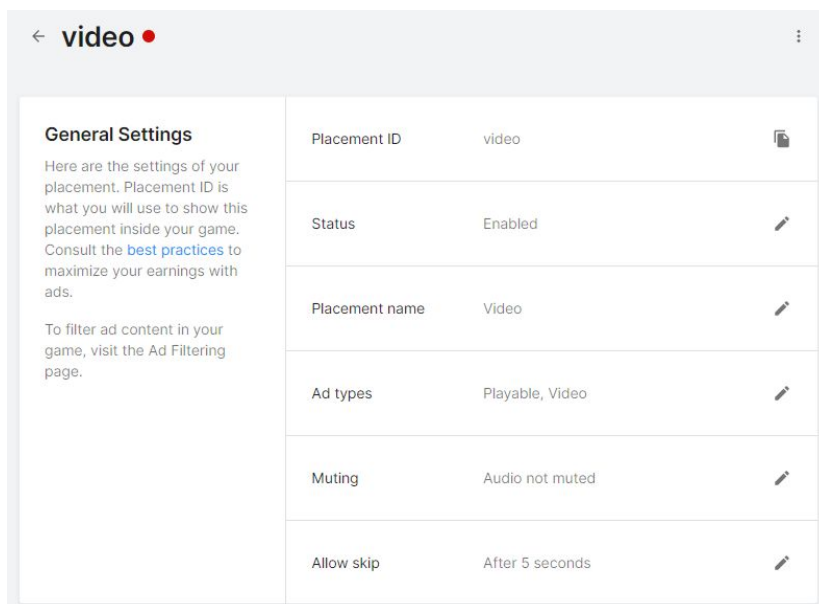
Placementの設定



Unityダッシュボードを開き、自分のゲームのプロジェクトを選択します。



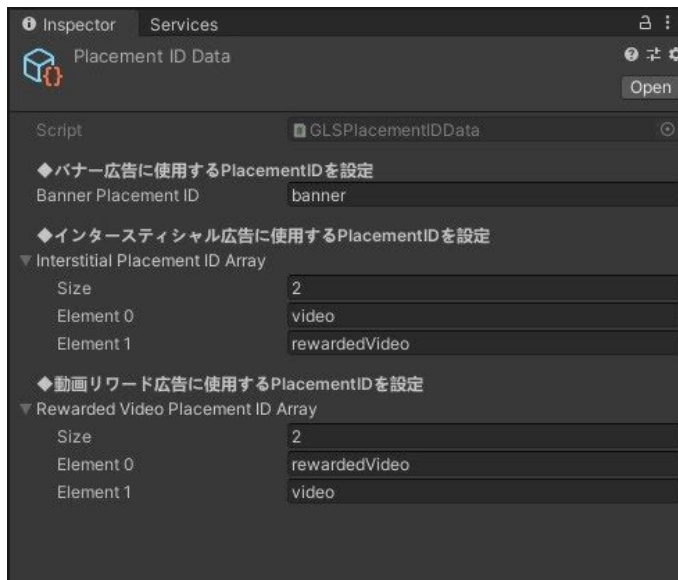
左側のリストから「**Monetization -> Placements**」を開き、Placement typeが「**Interstitial Video**」のPlacementを選択します。



必要に応じた箇所のパラメータを設定します。

- **Placement ID** : PlacementのIDです。右側のアイコンからクリップボードにコピーできます。(IDの変更は出来ません)
- **Status** : Enable placementのチェックを外すことで、広告を非表示にできます。
- **Placement name** : Placement 名を変更できます。この設定を変更しても、Placement IDには影響しません。
- **Ad types** : 表示する広告の種類を選べます。
- **Muting** : 「Mute audio」にチェックをつけると広告の音声をミュートにできます。
- **Allow skip** : チェックをつけると、広告が再生されてから何秒目から広告をスキップできるようにするか指定できます。

Placement IDの指定



インタースティシャル広告に使用するPlacementIDは「**Asset -> GLS -> Data**」内にある「**Placement ID Data**」の「**Interstitial Placement ID**」から指定できます。
「Size」の数を変更することで項目数の増減ができます。

インタースティシャル広告の表示

int Index :読み込む広告の番号

```
GLS.Ad.ShowInterstitial(int index)
```

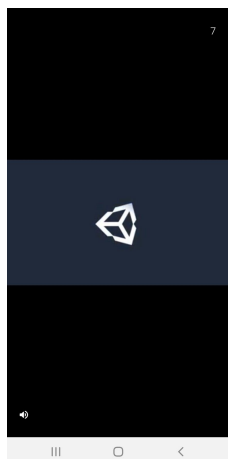
サンプルコード

```
using UnityEngine;
using UnityEngine.UI;

public class AdInterstitialSampleScript : MonoBehaviour
{
    [SerializeField]
    private Button interstitialButton = null;

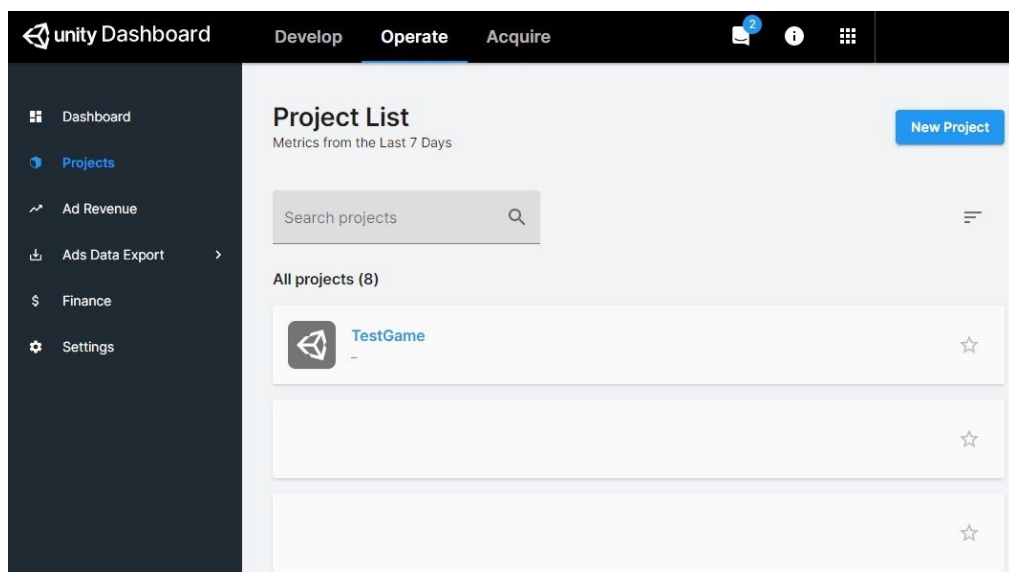
    private void Start()
    {
        // ◆Interstitialボタンが押された時の処理
        interstitialButton.onClick.AddListener(() =>
        {
            // ◆0番のインタースティシャル広告を表示
            GLS.Ad.ShowInterstitial(0);
        });
    }
}
```

動画リワード広告

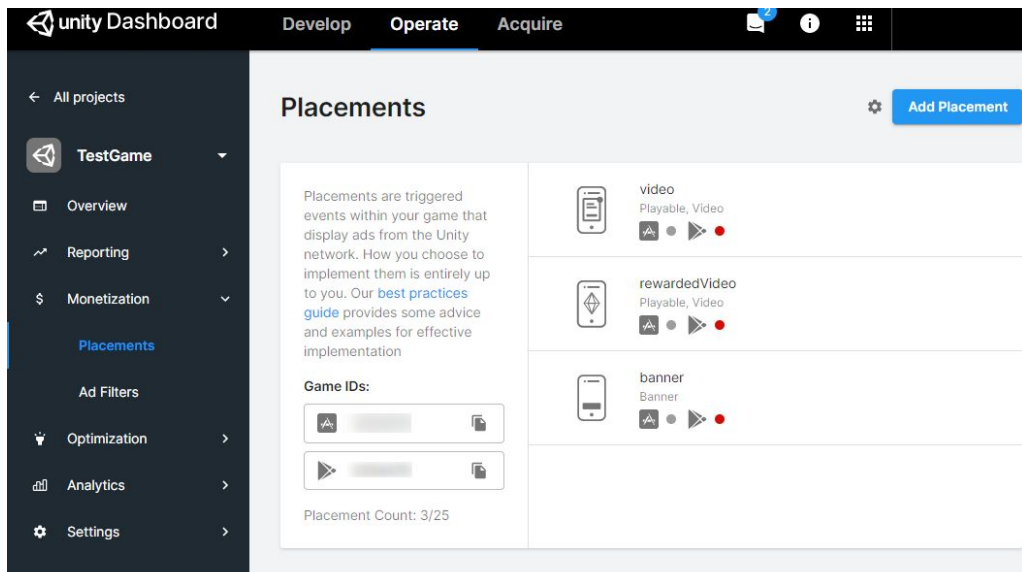


動画リワード広告を表示する際は、あらかじめ広告の表示結果別にコールバックを設定しておく必要があります。

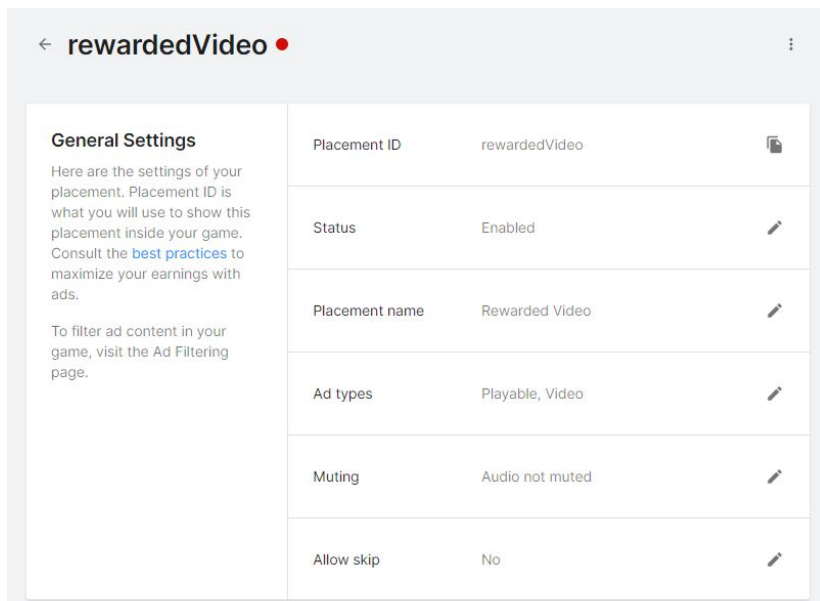
Placementの設定



Unityダッシュボードを開き、自分のゲームのプロジェクトを選択します。



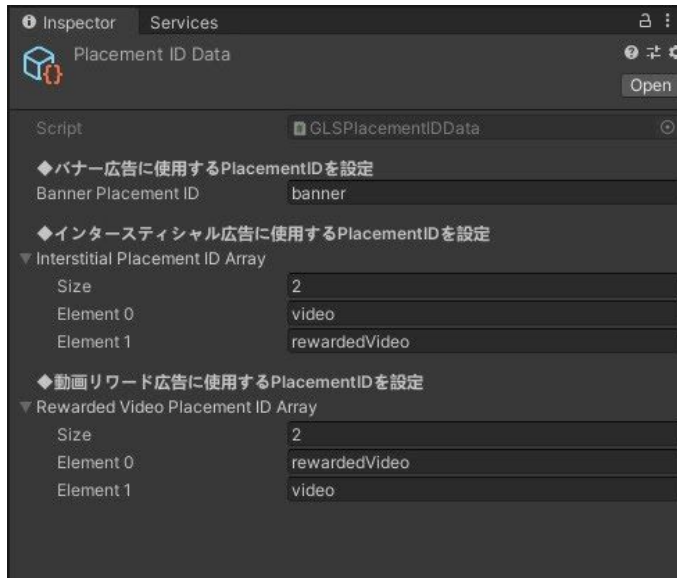
左側のリストから「**Monetization -> Placements**」を開き、Placement typeが「**Rewarded Video**」のPlacementを選択します。



必要に応じた箇所のパラメータを設定します。

- **Placement ID** : PlacementのIDです。右側のアイコンからクリップボードにコピーできます。(IDの変更は出来ません)
- **Status** : Enable placementのチェックを外すことで、広告を非表示にできます。
- **Placement name** : Placement 名を変更できます。この設定を変更しても、Placement IDには影響しません。
- **Ad types** : 表示する広告の種類を選べます。
- **Muting** : 「Mute audio」にチェックをつけると広告の音声をミュートにできます。
- **Allow skip** : チェックをつけると、広告が再生されてから何秒目から広告をスキップできるようにするか指定できます。

Placement IDの指定



動画リワード広告に使用するPlacementIDは「**Asset -> GLS -> Data**」内にある「**Placement ID Data**」の「**Rewarded Video Placement ID**」から指定できます。「Size」の数を変更することで項目数の増減ができます。

動画広告の表示

int Index :読み込む広告の番号

System.Action onFinished : 動画が最後まで視聴された時に呼ばれるコールバック

System.Action onSkipped : 動画がスキップされた時に呼ばれるコールバック(省略可)

System.Action onFailed : 動画が最後まで視聴された時に呼ばれるコールバック(省略可)

```
GLS.Ad.ShowRewardedVideo(  
    int index,  
    System.Action onFinished,  
    System.Action onSkipped = null,  
    System.Action onFailed = null  
)
```

サンプルコード

```
using UnityEngine;
using UnityEngine.UI;

public class AdRewardVideoSampleScript : MonoBehaviour
{
    [SerializeField]
    private Button rewardedVideoButton = null;

    private void Start()
    {
        // ◆ShowRewardedVideoの結果を受け取るコールバックを設定
        System.Action onFinished = () =>
        {
            // ◆動画リワード広告が最後まで視聴された際の処理
            Debug.Log("Finish");
        };

        System.Action onSkipped = () =>
        {
            // ◆動画リワード広告がスキップされた際の処理
            Debug.Log("Skip");
        };

        System.Action onFailed = () =>
        {
            // ◆動画リワード広告が再生失敗した際の処理
            Debug.Log("Failed");
        };

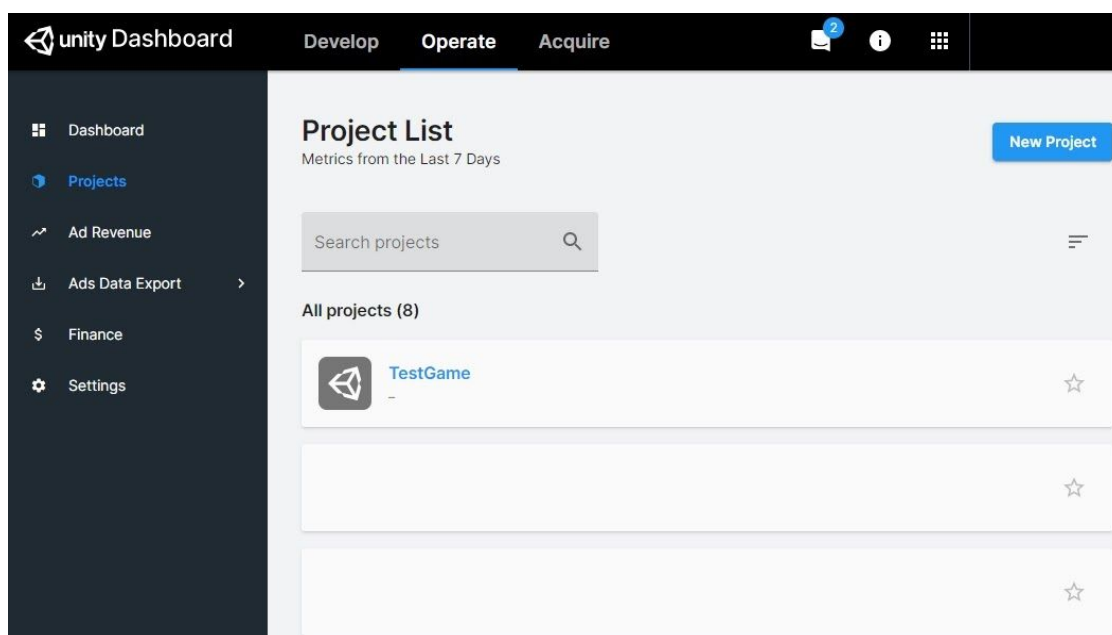
        // ◆RewardedVideoButtonボタンが押された時の処理
        rewardedVideoButton.onClick.AddListener(() =>
        {
            // ◆0番の動画リワード広告を表示
            GLS.Ad.ShowRewardedVideo(0, onFinished, onSkipped, onFailed);
        });
    }
}
```


バナー広告

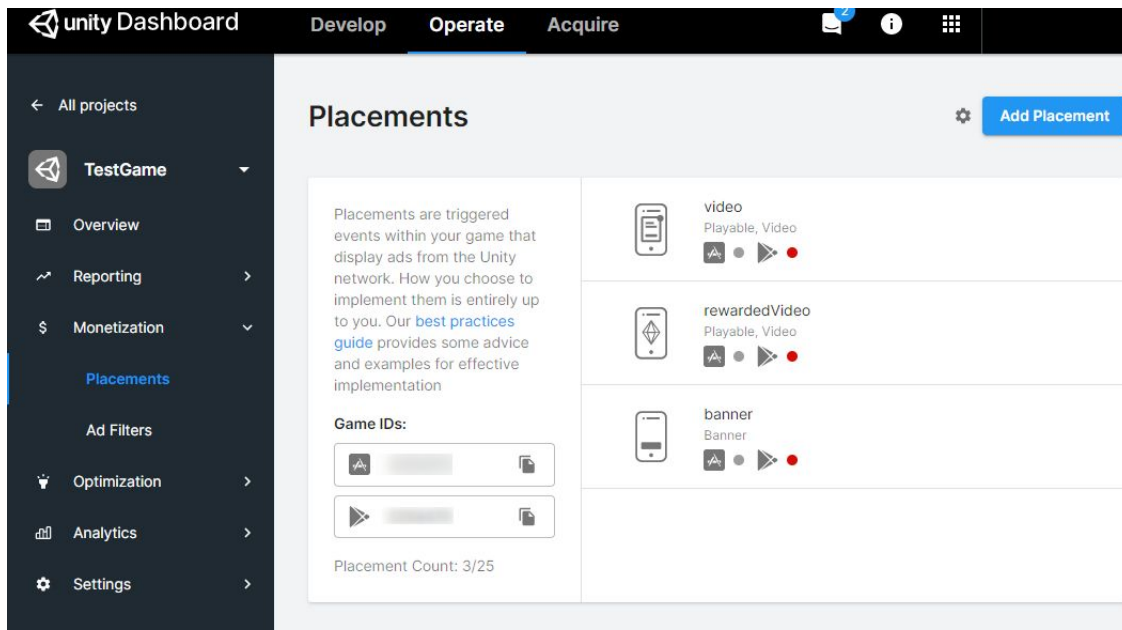


バナー広告は常に画面の最前面に表示され、描画優先度を変更することは出来ません。

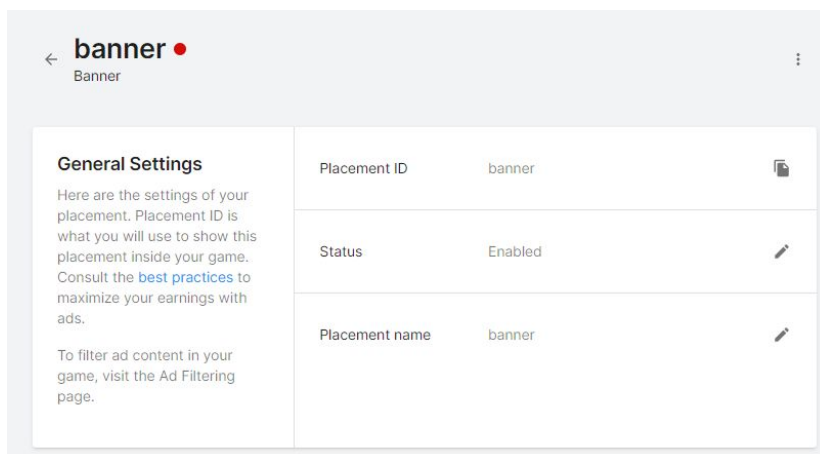
Placementの設定



Unityダッシュボードを開き、自分のゲームのプロジェクトを選択します。



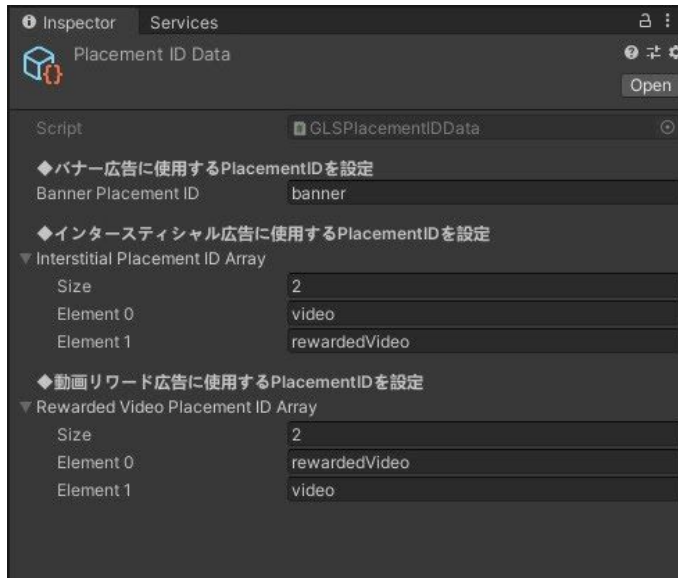
左側のリストから「**Monetization -> Placements**」を開き、Placement typeが「**Banner**」のPlacementを選択します。



必要に応じた箇所のパラメータを設定します。

- **Placement ID** : PlacementのIDです。右側のアイコンからクリップボードにコピーできます。(IDの変更は出来ません)
- **Status** : Enable placementのチェックを外すことで、広告を非表示にできます。
- **Placement name** : Placement 名を変更できます。この設定を変更しても、Placement IDには影響しません。

PlacementIDの指定



バナー広告に使用するPlacementIDは「**Asset -> GLS -> Data**」内にある「**Placement ID Data**」の「**Banner Placement ID**」から指定できます。

バナー広告の表示

```
GLS.Ad.ShowBanner();
```

バナー広告の非表示

```
GLS.Ad.HideBanner();
```

サンプルコード

```
using UnityEngine;
using UnityEngine.UI;

public class AdBannerSampleScript : MonoBehaviour
{
    [SerializeField]
    private Button showBannerButton = null;
    [SerializeField]
    private Button hideBannerButton = null;

    private void Start()
    {
        // ◆ShowBannerボタンが押された時の処理
        showBannerButton.onClick.AddListener(() =>
        {
            // ◆バナー広告を表示
            GLS.Ad.ShowBanner();
        });

        // ◆HideBannerボタンが押された時の処理
        hideBannerButton.onClick.AddListener(() =>
        {
            // ◆バナー広告を非表示
            GLS.Ad.HideBanner();
        });
    }
}
```

カスタムイベントの発行

イベントの発行

string customEventName : カスタムイベントの名前

string eventName : イベントの名前

object data : カスタムイベントがトリガーされるときに送られる追加のパラメーター

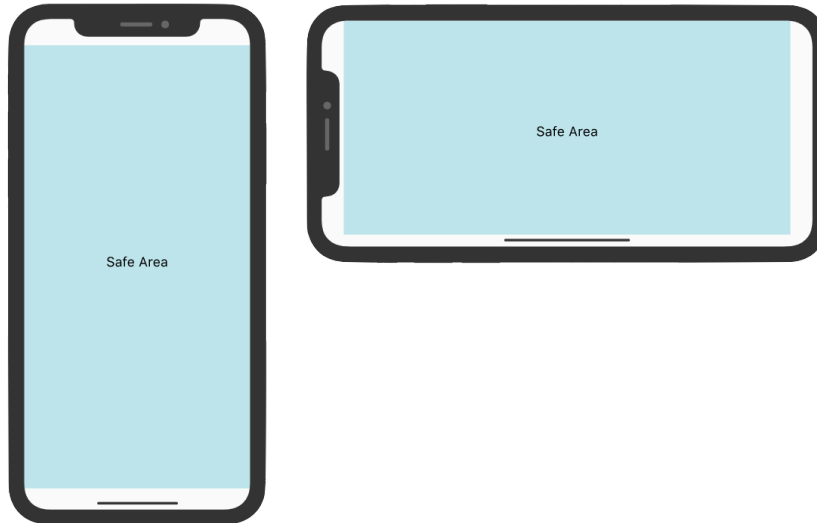
```
GLS.AnalyticsUtility.TrackEvent(  
    string customEventName,  
    string eventName,  
    object data  
)
```

サンプルコード

```
using UnityEngine;  
using UnityEngine.UI;  
  
public class TrackEventSampleScript : MonoBehaviour  
{  
    [SerializeField]  
    private Button eventButton = null;  
  
    private void Start()  
    {  
        // ◆EventButtonボタンが押された時の処理  
        eventButton.onClick.AddListener(() =>  
        {  
            // ◆カスタムイベントを発行  
            GLS.AnalyticsUtility.TrackEvent("CustomEventName", "EventName",  
            "Value");  
        });  
    }  
}
```

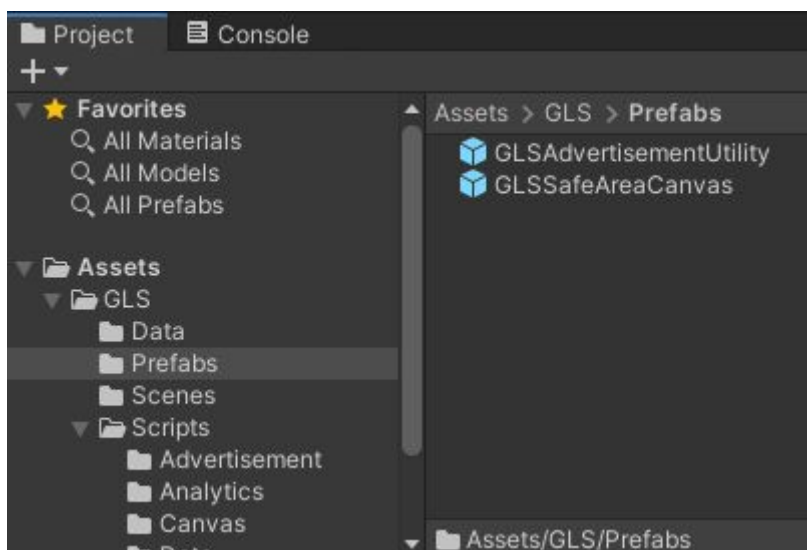
UIのセーフエリア対応

セーフエリア対応とは？

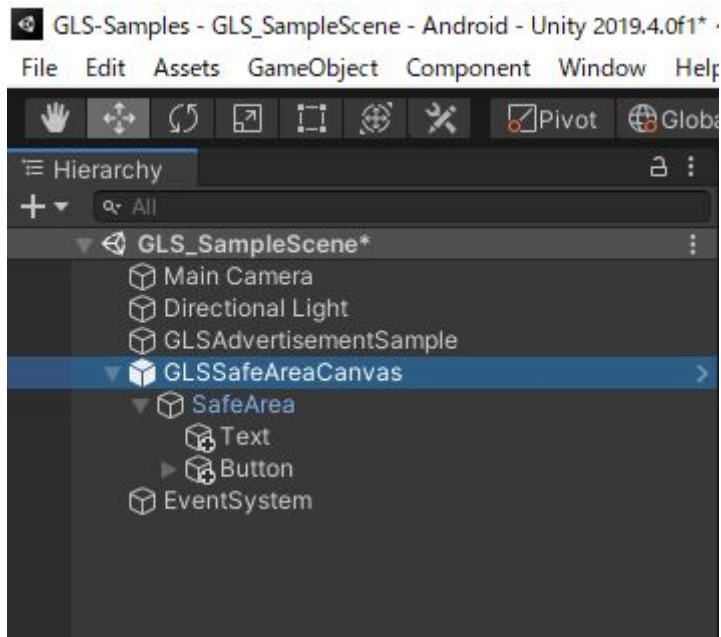


セーフエリアとは、ナビゲーションバーやステータスバーなどの外側のUIに被らない安全な領域のこと。このセーフエリア内にUIが収まるように調整することをセーフエリア対応と呼ぶ。

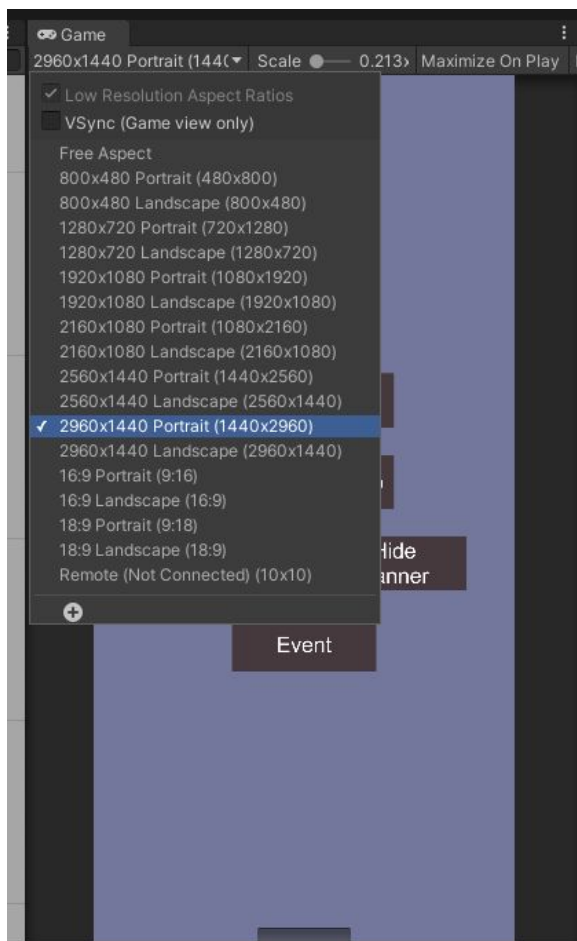
GLSSafeAreaCanvasの使い方



Projectウィンドウを開き、「**Asset -> GLS -> Prefabs**」内にある「**GLSSafeAreaCanvas**」プレハブをHierarchy上にドラックアンドドロップします。



GLSSafeAreaCanvasプレハブ内にある「**SafeArea**」の子として ButtonやTextなどのUIを配置するようにします。



端末にビルドしたり、ゲームビューで解像度設定を変えてプレイしてみたりしても UIの表示に問題が無ければ成功です。