# Toon网页游戏SDK接入相关

1.ToonSDK是用于挂载在漫画堂App服务器的网页游戏，因此Unity中使用WebGL打包。

2.调用ToonSDK需要在Index文件的Head中添加如下JS代码，添加方式有指向网页或下载本地的方法二选一。

<!-- 无需任何操作的添加ToonSDK方式 -->  
 <script src="https://cn-e-pic.itoon.org/cp-resource/mtjssdk2.js"></script>  
   
<!-- 需要本地存储一份JSSDK的添加ToonSDK方式 -->  
 <script src="mtjssdk.js"></script>

3.需要与JS代码进行交互需要在Unity的Plugin文件夹下，建立一个文件，文件名为mergeInto.jslib。大部分逻辑都写在Index上，相比FB小游戏SDK更简单一些，注意JsLib里的名字和C#里的名字需要一致，否则找不到方法会导致打包报错。内容例子如下：

<!-- 找个html内的空白处添加即可 -->  
<script>  
 function loadRewardAD() {  
 //加载激励广告  
 if (window.mangatoon && window.mangatoon.loadAd) {  
 getPlacementId('biz\_reward', function (placementId) {  
 window.mangatoon.loadAd({  
 placementId: placementId, //广告位编号  
 gameId: mGameid, //每一个游戏的唯一ID，由MangaToon提供  
 success: function (result) {  
 //加载成功  
 myGameInstance.SendMessage('ToonSDK', 'LoadAdsCallback', 1);  
 },  
 fail: function (result) {  
 //加载失败  
 myGameInstance.SendMessage('ToonSDK', 'LoadAdsCallback', result.errorCode);  
 },  
 })  
 })  
 }  
 }  
</script>

var func = {  
 //加载广告  
 ToonLoadAds: function () {  
 loadRewardAD();  
 },  
}  
  
mergeInto(LibraryManager.library, func);

public class ToonSDK : MonoBehaviour  
{  
 /// <summary>  
 /// 加载广告  
 /// </summary>  
 [DllImport("\_\_Internal")]  
 private static extern void ToonLoadAds();  
 /// <summary>  
 /// 加载广告的回调  
 /// </summary>  
 private Action \_loadSuccessAdsCallback;  
 private Action<int> \_loadFailedAdsCallback;  
 /// <summary>  
 /// 暴露给外部调用的加载广告  
 /// </summary>  
 public void LoadAds(Action successCallback, Action<int> failedCallback)  
 {  
 \_loadSuccessAdsCallback = successCallback;  
 \_loadFailedAdsCallback = failedCallback;  
 ToonLoadAds();  
 }  
 /// <summary>  
 /// 加载广告JS执行回调 0=失败 1=成功  
 /// </summary>  
 [MonoPInvokeCallback(typeof(int))]  
 private void LoadAdsCallback(int code)  
 {  
 if (code == 1)  
 {  
 \_loadedRewardCount++;  
 \_loadSuccessAdsCallback?.Invoke();  
 }  
 else  
 {  
 \_loadFailedAdsCallback?.Invoke(code);  
 }  
 \_loadSuccessAdsCallback = null;  
 \_loadFailedAdsCallback = null;  
 }  
}

4.因为有些方法需要调用到UnityWebGL的内容，所以需要在Index文件中新建一个变量，将UnityWebGL客户端赋值给它。

//暴露给mergeInto.jslib使用的UnityWebGL客户端  
 var myGameInstance = null;  
 // 把加载Unity游戏代码和素材的代码段提取到一个function中，便于后续调用  
 function startUnity(){  
 var script = document.createElement("script");  
 script.src = loaderUrl;  
 script.onload = () => {  
 createUnityInstance(canvas, config, (progress) => {  
 progressBarFull.style.width = 100 \* progress + "%";  
 }).then((unityInstance) => {  
 loadingBar.style.display = "none";  
 fullscreenButton.onclick = () => {  
 unityInstance.SetFullscreen(1);  
 };  
 //因为是异步方法，所以需要在外部声明 内部赋值   
 myGameInstance = unityInstance;  
 }).catch((message) => {  
 alert(message);  
 });  
 };  
 document.body.appendChild(script);  
 }

5.相关Demo：<https://github.com/ToxicStar8/ToonToUnityWebGL/>

6.如此就大功告成了，可以联系Toon运营进行测试