**#Master Controll**

#region NativeAd

public object GetNativeAd()

{

return adsController.GetNativeAd();

}

public bool IsNativeAdAvailable()

{

return adsController.IsNativeAdAvailable();

}

#endregion

**#Master Ad Controll**

#region NativeAD

public bool IsNativeAdAvailable()

{

foreach (IAdsController adHandler in adsControllers)

{

if (adHandler.IsNativeAvailable())

{

return true;

}

}

return false;

}

public object GetNativeAd()

{

foreach (IAdsController adHandler in adsControllers)

{

if (adHandler.IsNativeAvailable())

{

return adHandler.GetCurrentNativeAd();

}

}

//LoadNativeAd();

return null;

}

public void LoadNativeAd()

{

Debug.Log("LOAD NATIVE AD");

foreach (IAdsController adHandler in adsControllers)

{

adHandler.LoadNativeAd();

}

}

public void ReloadAllNativeAdBanner()

{

Debug.Log("RELOAD NATIVE AD");

LoadNativeAd();

onNativeAdRefresh?.Invoke();

}

public delegate void OnNativeAdRefresh();

public OnNativeAdRefresh onNativeAdRefresh;

public void RemoveAd()

{

ReloadAllNativeAdBanner();

}

#endregion

**#IAdsController (IronsourceManager)**

#region NativeAds

public bool testNativeAds;

public string nativeAdID;

private Stack<NativeAd> nativeAds = new Stack<NativeAd>();

string NativeKey => testNativeAds ? "ca-app-pub-3940256099942544/2247696110" : nativeAdID;

AdLoader adsLoader;

private bool loadedFail;

//bool isLoading;

public void LoadNativeAd()

{

//if (isLoading) return;

Debug.Log("Native start loading");

adsLoader = new AdLoader.Builder(NativeKey)

.ForNativeAd().SetNumberOfAdsToLoad(1)

.Build();

adsLoader.OnNativeAdLoaded += HandleNativeAdLoaded;

adsLoader.OnNativeAdClicked += HandleNativeAdClicked;

adsLoader.OnAdFailedToLoad += HandleNativeAdFailedToLoad;

adsLoader.OnNativeAdImpression += HandleNativeAdImpression;

adsLoader.OnNativeAdOpening += HandleNativeOpening;

adsLoader.LoadAd(CreateAdRequest());

//isLoading = true;

}

private void HandleNativeAdLoaded(object sender, NativeAdEventArgs e)

{

MobileAdsEventExecutor.ExecuteInUpdate(() =>

{

//isLoading = false;

Debug.Log("HandleNativeAdLoaded event received");

FireBaseManager.Instance.LogEvent("ADS\_NATIVE");

NativeAd nativeAd = e.nativeAd;

nativeAd.OnPaidEvent -= NativeAdOnPaidEvent;

nativeAd.OnPaidEvent += NativeAdOnPaidEvent;

nativeAds.Push(nativeAd);

Debug.Log("Native ad loaded " + e.nativeAd.GetHeadlineText() + " " + nativeAd.GetResponseInfo().GetResponseId() + " count:" + nativeAds.Count);

});

}

private void NativeAdOnPaidEvent(object sender, AdValueEventArgs e)

{

Debug.Log("ADMOB natve ONPAID EVENT");

double revenue = e.AdValue.Value;

var impressionParameters = new[]

{

new Firebase.Analytics.Parameter("ad\_platform", "ADMOD"),

new Firebase.Analytics.Parameter("ad\_source", "admod"),

new Firebase.Analytics.Parameter("value", (revenue / 1000000f)),

new Firebase.Analytics.Parameter("currency", "USD"), // All Applovin revenue is sent in USD

};

Firebase.Analytics.FirebaseAnalytics.LogEvent("ad\_impression", impressionParameters);

AdjustAdRevenue adRevenue = new AdjustAdRevenue(AdjustConfig.AdjustAdRevenueSourceAdMob);

adRevenue.setRevenue(revenue / 1000000f, e.AdValue.CurrencyCode);

Adjust.trackAdRevenue(adRevenue);

//FirebaseAnalysticController.Instance.SendRevenueToAdjust(e.AdValue);

}

private void HandleNativeOpening(object sender, EventArgs e)

{

MobileAdsEventExecutor.ExecuteInUpdate(() =>

{

Debug.Log("HandleNativeAdOpening event received");

//Invoke("LoadNativeAd", 1);

});

}

private void HandleNativeAdImpression(object sender, EventArgs e)

{

MobileAdsEventExecutor.ExecuteInUpdate(() =>

{

Debug.Log("HandleNativeAdImpression event received");

});

}

private void HandleNativeAdClicked(object sender, EventArgs e)

{

MobileAdsEventExecutor.ExecuteInUpdate(() =>

{

Debug.Log("HandleNativeAdClicked event received");

});

//throw new NotImplementedException();

}

private void HandleNativeAdPaidEvent(object sender, AdValueEventArgs e)

{

MobileAdsEventExecutor.ExecuteInUpdate(() =>

{

Debug.LogWarning("HandleNativeAdPaidEvent event received");

OnPaidEvent(e.AdValue);

});

}

private void HandleNativeAdFailedToLoad(object sender, AdFailedToLoadEventArgs args)

{

MobileAdsEventExecutor.ExecuteInUpdate(() =>

{

//isLoading = false;

Debug.Log("HandleNativeAdFailedToLoad event received with message: " + args.ToString());

loadedFail = true;

});

}

public object GetCurrentNativeAd()

{

Debug.Log("GET NATIVE AD: " + nativeAds.Count);

if (nativeAds.Count > 0)

{

Debug.Log("+native>>>>> " + nativeAds.Peek().GetHeadlineText());

return nativeAds.Pop();

}

else

{

return null;

}

}

public bool IsNativeAvailable()

{

if (loadedFail)

{

loadedFail = false;

LoadNativeAd();

}

return nativeAds.Count > 0;

}

#endregion

**#Lưu ý**

* IAdsController (IronsourceManager): ***thêm LoadNativeAd()*** *cạnh load openAd trong lúc Init MobileAds*
* Trong **prefab xóa hết collider cũ**:

+ Để đúng **1 box collider** ở ngoài bao trọn native Ads

+ Xóa hết **raw Image** thay bằng Image thường

+ Thêm icon **AdChoices** sẵn vào (có trong package gửi)

+ Nếu dùng **remote** thì sửa trong #region Remote NativeAD

+ Cái này dùng **UniTask** (package trong file gửi), nếu không dùng dổi sang **IEnumerator(**bỏ/xóa ***cancellationToken,*** mấy cái ***await*** đổi thành ***return yield***, ***delay(ms)*** đổi thành ***waitforseconds(s)***, ***unitask.yield*** đổi thành ***return yield null***)