Adze

> Read the code in the Examples Folder.

Introduction

Advertisements are the life-blood for many games. It is difficult to attract players if they have to pay upfront. There are many different formats with the three most common being:

- **Banner** - displays above or below game-play
- **Interstitial** - displays between game segments
- **Reward** - Player chooses to see ad for gain

With an extensive range of networks to choose from:

- **AdMob** - Google-backed and the most popular
- **Chartboost** - specifically for games
- **Appodeal** - Aggregator of many of the others
- and many more

Now you could use an aggregator and manually select the network. There are some problems with this:

- **Dependency** - If the aggregator is offline or shut down you will not have any advertisements served.
- **Size** - An aggregator loads libraries for all the networks it supports. It is a problem for Android with the 64k limit on entry points.
- **Versioning** - You have to rely on the version of a network API supplied by the aggregator.
- **Complexity** - The aggregator system to set up priorities on networks can be hard to program and painful to maintain.

- **Reliability** - Networks cannot always serve an advertisement. It is useful to have more than one source to cover this situation.
- **Platforms** - Different providers support different platforms, with Android and iOS being the most popular.

Adze provides a decoupled interface to the systems of your choice. If you have more than one source, you can choose different strategies for using them.

Adze also provides a rewarded video prefab that once reskinned can ease the process for you.

Installation

You will need to install ***Askowl-Lib*** and ***Askowl-Adze*** packages. It is all you will need for development. The external interfaces are stubbed out and only show a message if used.

Install the API packages when you are ready to test on active devices. The stub message will give you a URL.

Implementation

All components are Custom Assets. Instances of Custom Assets reside in a ***Resources*** somewhere in your project.

View the assets in the sample project for examples.

1. Working in the Unity editor project view
2. Create a ***Resources*** directory if needed
3. Select this directory to open it
4. Open the right-click context menu
5. Select *Create // Adze // nameOfNetwork*
    1. Select the newly created asset
    2. Fill in the application keys for each platform you are supporting
    3. Select the type of advert (banner, interstitial, reward, more)
    4. Select the priority (or leave all at 1) to set the order of network processing
6. Repeat for each network that may be needed
7. Select *Create // Adze // Distributor
    1. Deselect round-robin option for fixed order
    2. Drag each network asset into the ***Servers*** field

Application Keys

The advertising network will provide a key unique to your application. For each network asset you have created, add entries in the ***App Keys*** field for each platform you intend to support.

The platform dropdown lists all the platforms that Unity3D supports. Each network only supports a small subset.

ChartBoost requires two keys. Enter both keys into the same field separated by a semi-colon.

Mode

Select the display format. It can be banner, interstitial or reward. Create separate assets for the network and the distributor if your application uses more than one mode.

Priority

The order that networks are accessed is based on the priority, with one being used first.

Usage Balance

When in round-robin mode, each network entry will be used ***Usage Balance*** times in a row before going to the next on the list. Using this fields some networks can be used more heavily than others.

If a network fails to fill a request the next network is used without waiting for the balance to turn to zero.

Round Robbin

Round Robin mode is on by default. Each network is used ***Usage Balance*** times before moving on to the next. Once the last network has been used, it is back to the first.

Current Networks

Appodeal

For an Appodeal asset you can disable any number of networks you don't want used. Just add a new row to the ***Disabled Networks*** field and select the network to be ignored. In Appodeal, *location* is called *placement*. Placements can be created on the dashboard under a relevent segment. Give it a name and use that in `Show(string placement)`.

Chartboost

Chartboost is a games specific advertising network. Locations in `Show(string location)` has a special meaning. The pre-specified locations are:

> Default, Startup, Home Screen, Main Menu, Game Screen, Achievements, Quests, Pause, Level Start, Level Complete, Turn Complete, IAP Store, Item Store, Game Over, Leaderboard, Settings, Quit

You can also add your own, but it is then unlikely to improved future eCPM.

Google AdMob

Admob is the granddaddy of them all thanks to Google support. It can also mediate between many networks. In this case it is all online and requires you to set the technical data required. Location is set as a keyword in an add request. Options in the dashboard can change targeting based on keywords.

Adding a new advertising Network

At last count, there are over 40 networks, not including special and regional ones. Here is a partial list:

> AdColony, Adcash, Admob, Adsterra, Airpush, Applovin, Appnext, AppsUnion, Avazu, Billy, Mobile, Black6Adv, CPAlead, Chartboost, Clickdealer, Epom, Fyber, GOWIDE, InMobi, Kimia, Leadbolt, Minimob, Mobads,MobAir, Mobidea, MobileCore, Mobobeat, MoBrain, Mobfox, Mobusi, Mobvista, Mpire, Network, Msales, PassionTeck, Persona.ly, Phunware, Propeller, Ads, RevMob, Smaato, Startapp, Tapjoy, Unity Ads, Vungle and Yeahmobi

`Adze` only supports a few of them. If you want to add a new one, it is a relatively simple process. Inherit from `AdzeServer` and implement `showNow(string location)`. Implementing `Initialise(string key)` and `Destroy()` are optional.

`showNow` is a coroutine. It should continue after the advert or an error occurs.

There are also three public booleans that can be updated:

1. `adActionTaken` - if a user clicks on an advert or earns a reward.
2. `error` - If there is an error in loading or showing the ad.
3. `loaded` - Set when an advert is loaded and ready to play.

An offer

If you write a new network interface and donate it back to Adze, I will provide you with free access to all my packages on the Unity3D store in perpetuity.