Agora WebGL Programming Guide

The WebGL plugin has almost the same applicable APIs as the original native SDK. This page explains the subtle differences between the native and WebGL target.

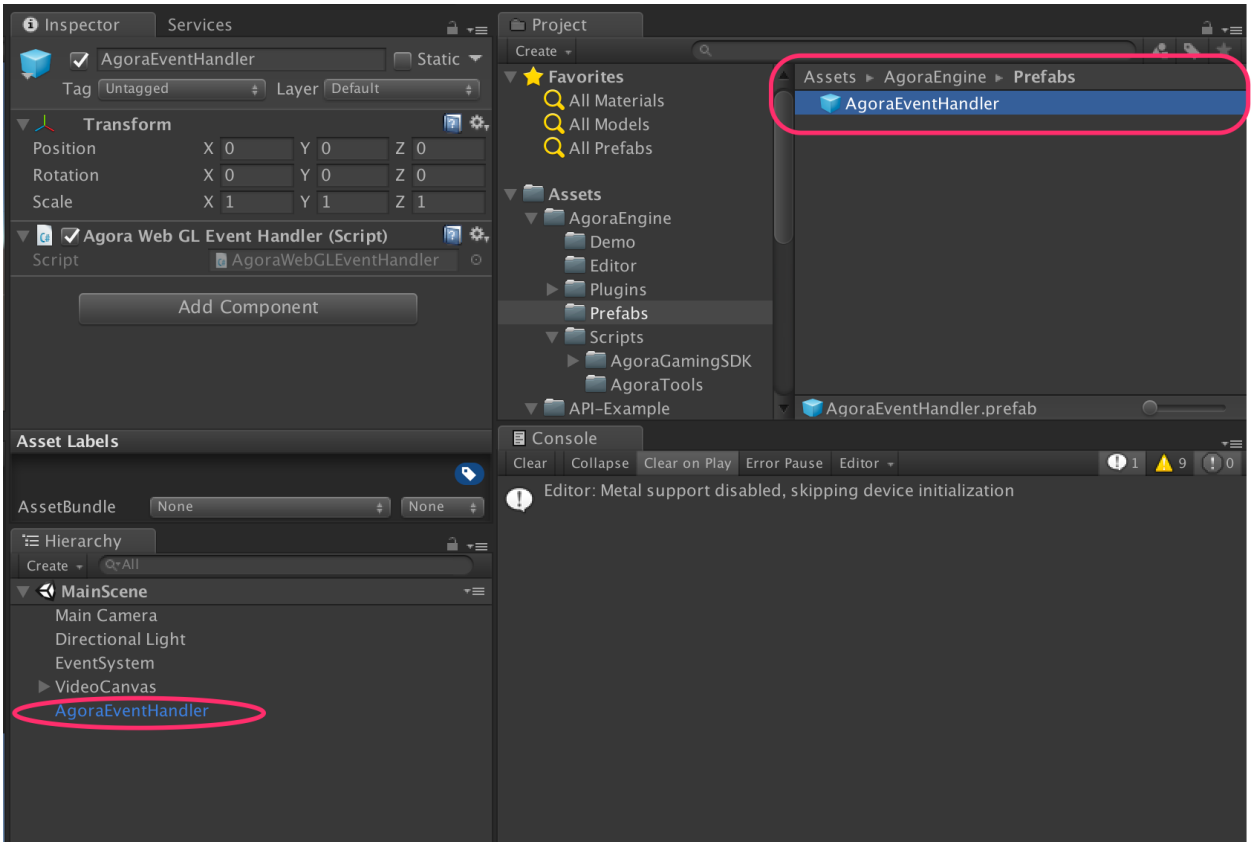
SDK Import

The WebGL plugin works by itself when the deployment target is WebGL only. However, in order to test your code in the Unity Editor and deploy to native platforms you should:

1. Download and import the official Unity SDK. (The current version is 3.7.0.3)
2. Download and import the separate Agora WebGL plugin package, overwrite any files that exist in the official Unity SDK.
3. You may need to restart Unity Editor after the import

Scene Update

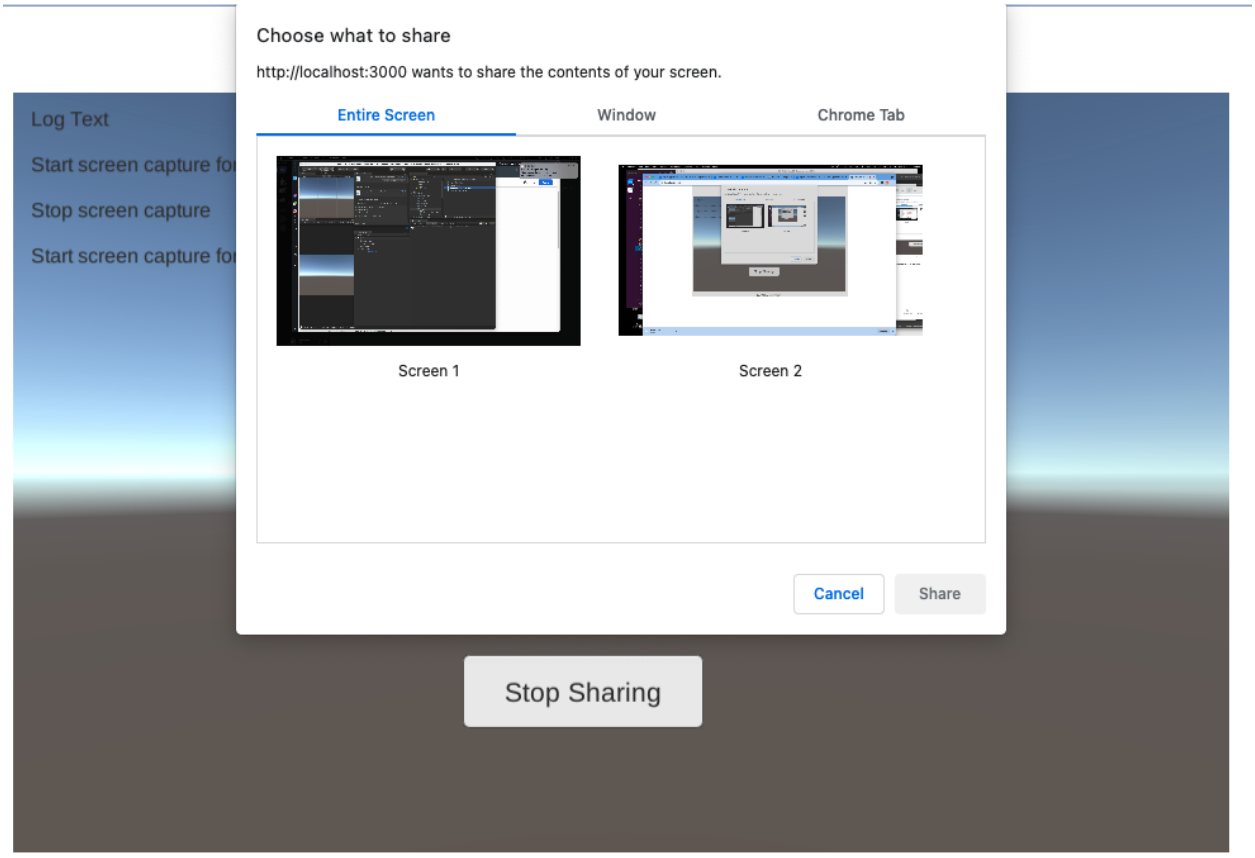
The application needs an AgoraEventHandler instance to communicate with the WebGL layer of the engine. To do this, drag the AgoraEventHandler prefab from the Prefabs folder to the scene. This object is marked DontDestroyOnLoad. Therefore, you only need to add it to your entry level scene once.



ScreenSharing

Clearly, ScreenSharing is on the top of the popular features list. The WebGL plugin covers this for:

- **App ScreenSharing**: shares a particular texture within the Unity Application. This texture could be the entire visible screen through the main camera or a static texture. See the included *app-screenshare-sample* demo for this feature. The core API in use is *PushVideoFrame*(). The order of SetExternalVideoSource() and JoinChannel() calls are different from the native behavior.
- **Desktop ScreenSharing**: shares a window captured by the desktop OS. This can be a whole display or a particular window on MacOS or Windows. Extended to the web browsers, this would also include tabs inside the browser. The core API is new, use *StartScreenCaptureForWeb()* for this call.
- Starting in Refactor3 release, new APIs are added:
    - Start**New**ScreenCaptureForWeb(uid, **audioEnabled**) can start a separate client that doesn't take over the webcam stream. With audioEnabled, the shared content's audio is sent instead of the mic on the screenshare track. With an assigned uid, the remote client can identify this stream is screensharing stream.
    - The similar APIs applies to multichannel (i.e. AgoraChannel class)



localVideo(163592208)

SetVideoEncoderConfiguration

This API is not a 100% match to the native SDK. The main difference is that the mirror mode parameter does not exist for the Web. If your original code sets the mirror-mode true and expects the video stream to become left-right flipped for the remote viewer, this won't be the case for the Web user. Please leverage other ways to achieve the same effect. The following documents explain the difference:

- Native SDK reference
- Web SDK reference

String user id (a.k.a userAccount name) vs integer user id (i.e. uid)

In the original Unity SDK and the Web SDK, both string user id and unsigned integer user id are accepted. However, Unity always gets a uint uid for remote users from the *OnUserJoin*() call. The Web gets a string id if the remote user joins with a string id. Therefore, the guideline is that

not to use string user id if the applications are talking across the native (Unity) SDK and the Web SDK. Here is the summary on correspondence:

- If native SDK uses string UID, web SDK uses int UID, there is no issue with audio or video call between native user and web user. Web will return int UID for the remote user.
- If native SDK uses int UID, web SDK uses string UID, native can receive audio and video stream from web side, but web cannot receive audio or video streams from native side, there is no call back for user joined or subscribe event on web side.
- If both native SDK and web SDK uses string UID, the web will return string UID for the remote users.

OnVolumeIndication

- On native, this includes local user's callback; the WebGL version supports remote users only.
- The native version doesn't combine more than one user in one callback; the WebGL version implements the design with multiple users in the list when possible.

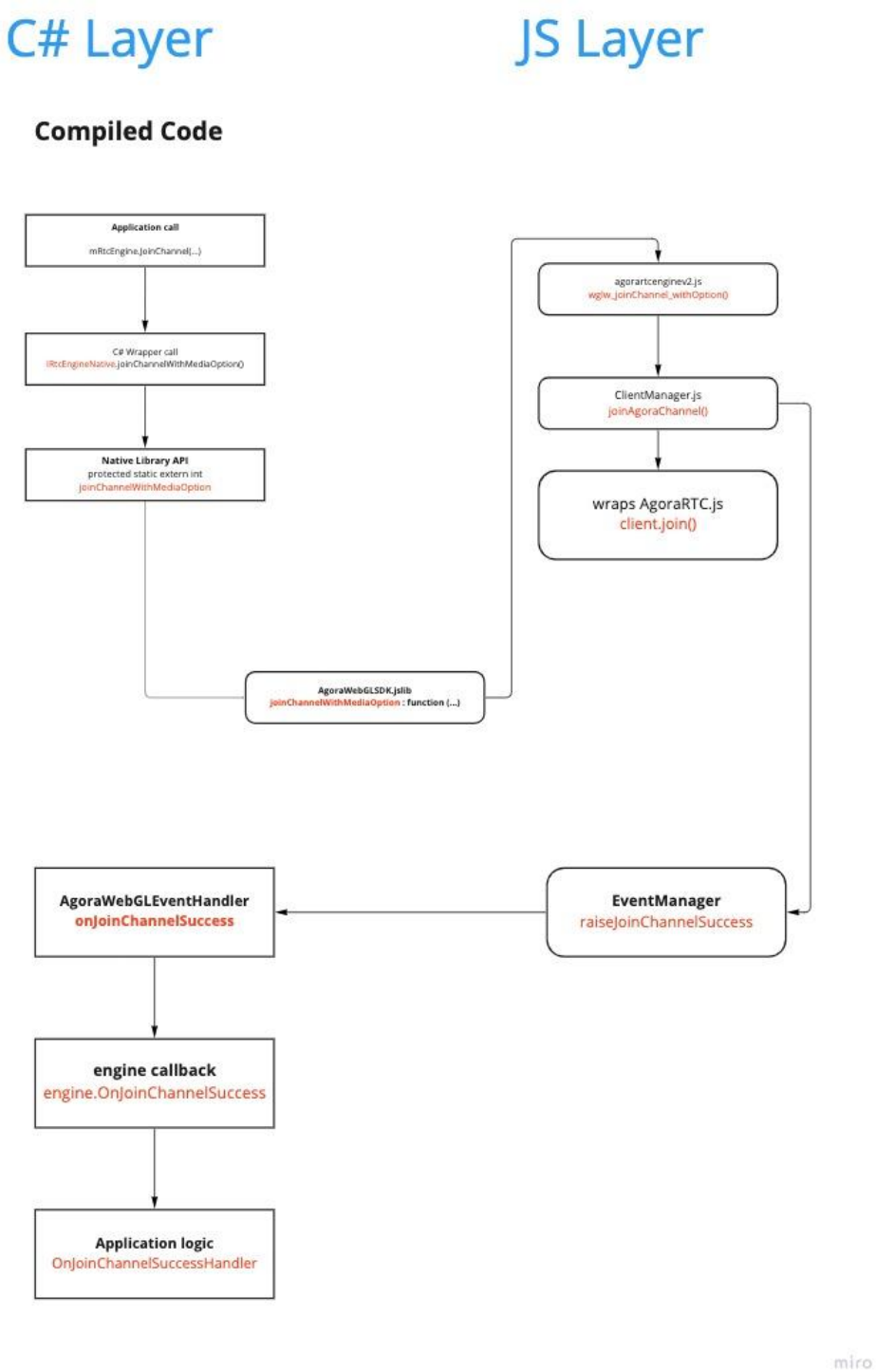
DataStream Signaling

Subtle differences in implementing the APIs are as described below:

Int streamID	StreamID is not used in WebGL. It will default to 0. This does not affect interoperation between native and WebGL.
public int CreateDataStream(bool reliable, bool ordered) (Note this is marked deprecated)	Parameter *reliable* is useless. Ordered means needsRetry, provided some certainty that the message is delivered.
public int CreateDataStream(DataStreamConfig config)	Parameter syncWithAudio inside the config is not supported. Ordered means needsRetry,
OnStreamMessage(uint userId, int streamId, byte[] data, int length)	Works as normal.

WebGL Plugin Code Modification

Here is a diagram showing the Unity Layer to JS Layer architecture, using JoinChannel as an example.



(Or click here to visit the original whiteboard)

Developer’s note:

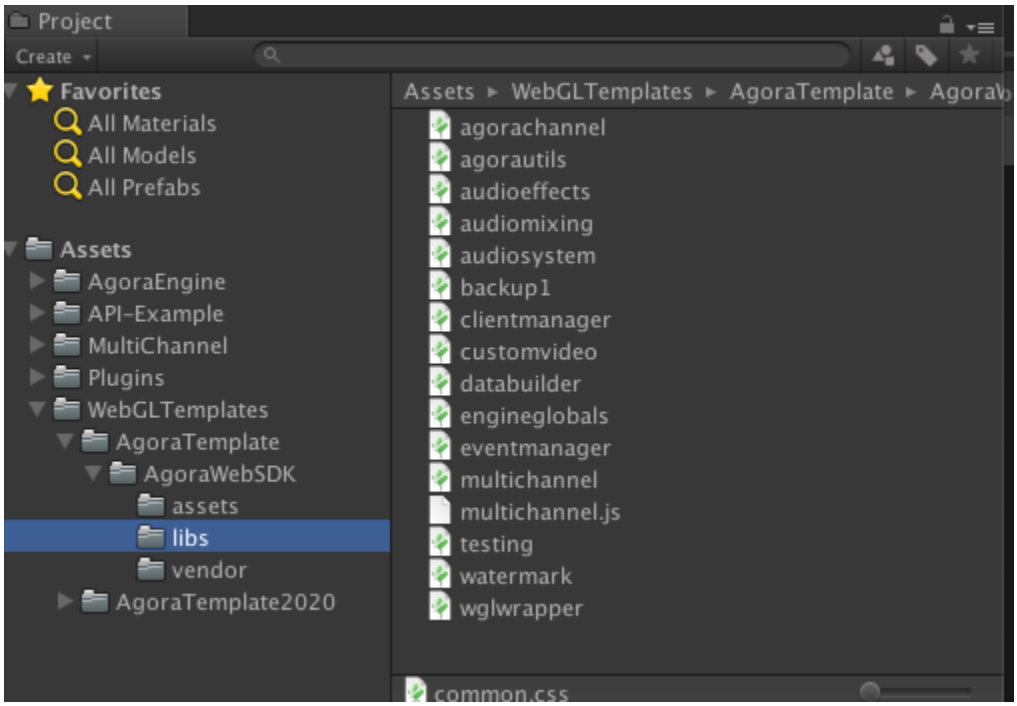
In order to add new APIs or fix bugs, you can extend and modify the following components in the WebGL plugin:

Jslib file

Best practice for using browser JavaScript in your project is to add your JavaScript sources to your project, and then call those functions directly from your script code. To do so, place files with JavaScript code using the .jslib extension under a “Plugins” subfolder in your Assets folder.

You can find more information about browser scripting in [the official Unity documentation](#).

The feature wrapper scripts are under Template/AgoraSDK/libs folder:



Template

WebGL template is a configuration setting that allows you to control what this HTML page looks like. This enables you to test, demonstrate, and preview your WebGL application in the HTML page.

Template has been customized from Unity template and differs because it has to load our javascript SDK and other essentials. You can customize most of the template as long as you keep the SDK included and do not change the following:

```
// REQUIRED - PLEASE DONT DELETE
var mainCanvas = document.getElementById("myCanvas");
var mainContext = mainCanvas.getContext("2d");
var inMemCanvas = document.getElementById("inMem_Canvas");
var inMemContext = inMemCanvas.getContext("2d");
var canvasWidth = mainCanvas.width;
var canvasHeight = mainCanvas.height;
```

You can find more information in the official Unity documentation.

AgoraEventHandler

Use the SendMessage method to deliver events to the Unity container. AgoraEventHandler catches all the events from Web scripts. Some events such as media device listing receive a lot of data. Hence, there are other helper classes to store this data.

Other References

- Agora WebGL Build And Run Guide
- Agora WebGL Plugin README