Abstract

Tutorial of how to transfer data from WebGl to google cloud. WebGl is made by using Unity game engine. First, data is serialized from .Net code to proto buffers using protobuf-net. Since WebGL is written in javascript, data is transferred using javascript library calling from C# in Unity.

Tool

Unity

Platform for creating WebGL. Download at Unity

Nuget

Now cannot directly download from the assets store. Can download the package at Nuget Gallery

ProtoBuf.Net

Serialized data from .Net code to protobuf buffer serialization format by Google. More info at protobuf-net github. Can install from Nuget package or import from github

Firebase CLI

Firebase App and firebase storage should be install in the google cloud shell. Firebase Hosting tutorial

Python Version 3

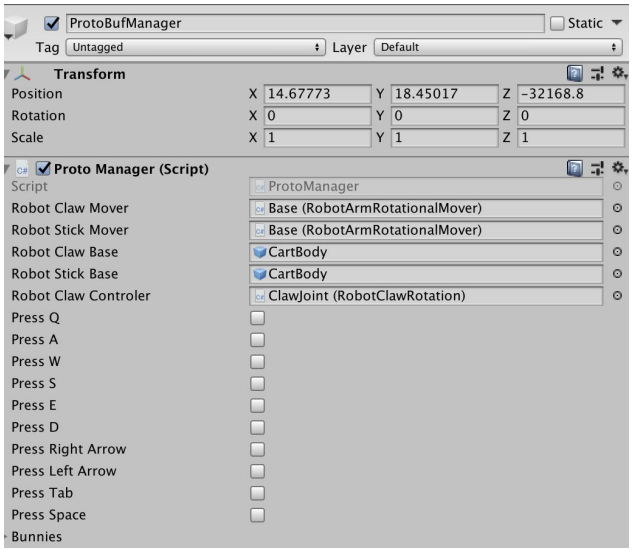
Main

GameDetail

```
[Serializable]
[ProtoContract]
public class ProtoGameDetail
{
    [ProtoMember(1)]
    public List<float> rotationClawArm;
    [ProtoMember(2)]
    public List<float> rotationStickArm;
    [ProtoMember(3)]
    public List<float> ClawBodyPosition;
    [ProtoMember(4)]
    public List<float> StickBodyPosition;
    [ProtoMember(5)]
    public List<bool> ClawController;
    [ProtoMember(6)]
    public List<byte[]> ImageByteBlock;
    [ProtoMember(7)]
    public List<byte[]> ImageByteBP;
    [ProtoMember(8)]
    public List<Int32> BlockCount;
    [ProtoMember(9)]
    public List<float> BlocksPosition;
    [ProtoMember(10)]
    public List<bool> PlayerControls;
}
```

Set class member to be serialize as above show. Protobuf-net can serialized class in a class as well. However, we want to easily pick our data, so we set each variable separately.

ProtoManager



ProtoManager is in the build scene. Every frame in the build scene, we will record all the gameobjects position, rotation and all necessary data. Every score changing will be recorded two images(BP and the current).

Since firebase SDK does not support WEBGL, javascript is needed to communicate WebGL with firebase. WebGL can interact with the browser scripting using jslib(javascript library function). We can call the javascript function inside the unity C# code with the syntax below

```
/// </summary>
public class ProtoFile : MonoBehaviour
{
    [DllImport("__Internal")] // file for syncFile in indexedDB as a tempory memory
    private static extern void SyncFiles();

    [DllImport("__Internal")] // Alert trigger function at the webgl content
    private static extern void WindowAlert(string message);

    [DllImport("__Internal")] // upload the file as a string to javascript function (data is string of the data,
                              // filename is the name of the file save as (ddMMhhmmssffffff)
    private static extern void uploadFile(string data, string filename);

    //public GameObject ProtoBufManager;
```

Basically import the javascript function on the top. Then create a .jslib file inside the Plugin/WebGL that contain a javascript object that merge to the LibraryManager.

```
var HandleIO = {
    WindowAlert : function(message)
    {
        window.alert(Pointer_stringify(message));
    },
};

mergeInto(LibraryManager.library, HandleIO);
```

More info the the [Unity page]

Firebase hosting

Can follow the [Tutorial] at firebase website.
Create firebase config and init it in the javascript script

```
// Set the configuration for your app
// TODO: Replace with your app's config object
var firebaseConfig = {
  apiKey: '<your-api-key>',
  authDomain: '<your-auth-domain>',
  databaseURL: '<your-database-url>',
  storageBucket: '<your-storage-bucket-url>'
};
firebase.initializeApp(firebaseConfig);

// Get a reference to the storage service, which is used to create references in your storage
var storage = firebase.storage();
```

Make sure to include the dependencies (JS SDKs) in the index.html inside <bod> </body>
Need firebase app which have to be at the top.
Firebase storage and firebase auth

```
<body>
  <script src="/__/firebase/7.2.0/firebase-app.js"></script>
  <script src="/__/firebase/7.2.0/firebase-storage.js"></script>
  <script src="/__/firebase/7.2.0/firebase-auth.js"></script>
  <script src="https://cdn.firebase.com/libs/firebaseui/4.2.0/firebaseui.js"></script>
```

More dependencies(JS SDKs) can be found at [firebase hosting]

Firebase product	Library reference (reserved URL)
Firebase core (required)	`<script src="/__/firebase/7.1.0/firebase-app.js"></script>`
Analytics	`<script src="/__/firebase/7.1.0/firebase-analytics.js"></script>`
Authentication	`<script src="/__/firebase/7.1.0/firebase-auth.js"></script>`
Cloud Firestore	`<script src="/__/firebase/7.1.0/firebase-firestore.js"></script>`
Cloud Functions for Firebase Client SDK	`<script src="/__/firebase/7.1.0/firebase-functions.js"></script>`
Cloud Messaging	`<script src="/__/firebase/7.1.0/firebase-messaging.js"></script>` For an optimal experience using Cloud Messaging, also add the Firebase SDK for Analytics.
Cloud Storage	`<script src="/__/firebase/7.1.0/firebase-storage.js"></script>`
Performance Monitoring *(**beta** release)*	`<script src="/__/firebase/7.1.0/firebase-performance.js"></script>`
Realtime Database	`<script src="/__/firebase/7.1.0/firebase-database.js"></script>`
Remote Config *(**beta** release)*	`<script src="/__/firebase/7.1.0/firebase-remote-config.js"></script>` For an optimal experience using Remote Config, also add the Firebase SDK for Analytics.
Firebase JavaScript SDK (entire SDK)	`<script src="/__/firebase/7.1.0/firebase.js"></script>`

Authentication UI

Firebase authentication set up. Functions are in the jslib
function: createAuthenticationUI : Using google account and email for authentication. Can add more different options as well.

```
createAuthenticationUI : function()
{
    window.authenticated = 0;
    var uiConfig = {
        callbacks: {
            signInSuccessWithAuthResult: function(authResult, redirectUrl) {
                var user = authResult.user;
                var credential = authResult.credential;
                var isNewUser = authResult.additionalUserInfo.isNewUser;
                var providerId = authResult.additionalUserInfo.providerId;
                var operationType = authResult.operationType;
                window.authenticated = 1;
                window.user = authResult.user;
                unityInstance.SendMessage('Authentication', 'RemoveLoadingText');
                unityInstance.SendMessage('Authentication', 'LoadLandingScene');
                unityInstance.Module.asmLibraryArg._getBucketFiles();

                // Do something with the returned AuthResult.
                // Return type determines whether we continue the redirect automatically
                // or whether we leave that to developer to handle.
                return false;
            },
```

verifyAuthentication: return bool => is the authentication success or not

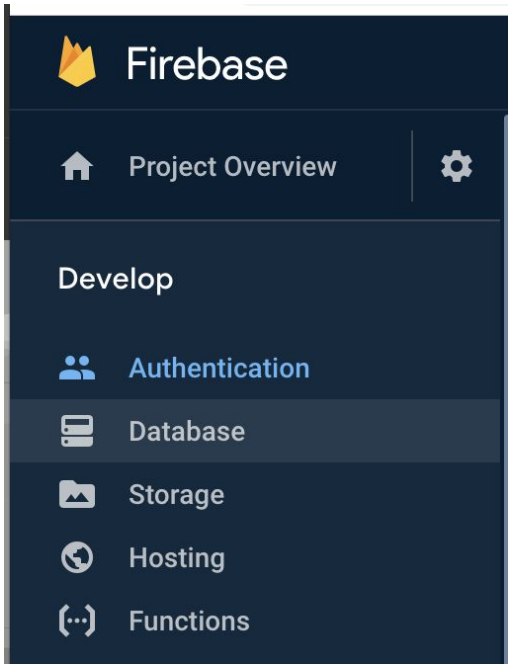
```
    },
    verifyAuthentication : function()
    {
        var user = firebase.auth().currentUser;
        if (user)
            return true;
        else
            unityInstance.SendMessage('Authentication', 'DisplayLoadingText');
        return false;
    },
```

getUserIDString: return string => return authentication user ID as a UTF8 string

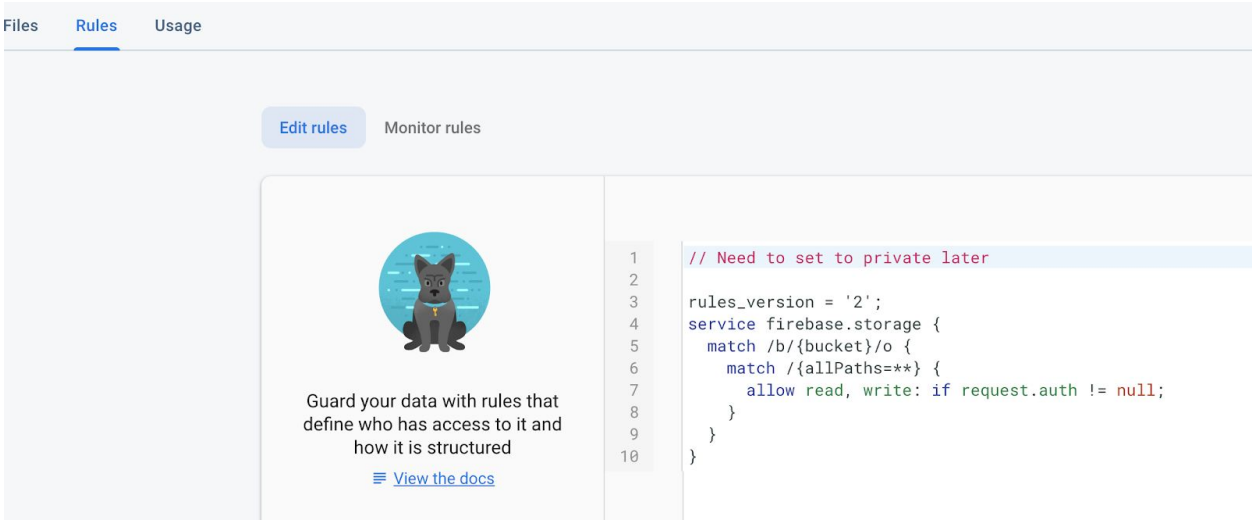
```
// return the authentication user ID
getUserIDString : function()
{
    if (window.user !== "undefined")
    {
        var bufferSize = lengthBytesUTF8(window.user.displayName) + 1;
        var buffer = _malloc(bufferSize);
        stringToUTF8(window.user.displayName, buffer, bufferSize);
        return buffer;
    }
    return "undefined";
},
```

Firebase storage & authentication

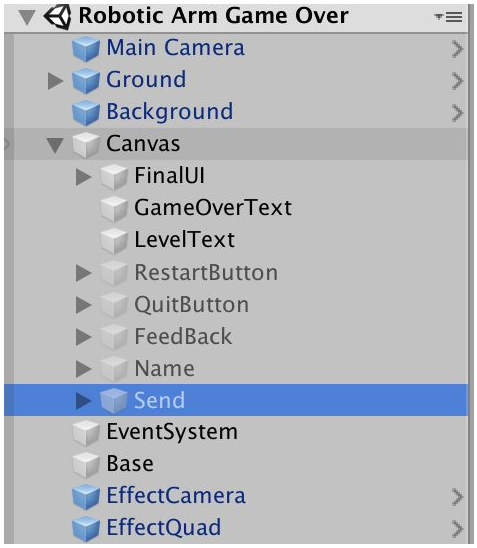
Go to [Firebase Console](). Inside the project make sure the authentication is working.



Select the storage. Make sure to set the bucket's rule to not public. More info for [storage rule]



Send Protofile




```
using (MemoryStream ms = new MemoryStream())
{
    ProtoBuf.Serializer.Serialize(ms, gameDetail);
    string stringBase64 = Convert.ToBase64String(ms.GetBuffer(), 0, (int)ms.Length);
    string fileName = DateTime.Now.ToString("MMddhhmmssffffff");
    uploadFile(stringBase64, fileName);
}
```

SendProto is inside Gameover scene GameDetail object will be serialized as a memory stream. Then the byte stream will be converted to base64 encoded so we can pass it as a base64 string to the javascript function: uploadFile

Filename is User ID + Timestamp with b64 suffix to indicate the file is encoded as base64.
In the uploadFile function, firebase storage is initiated and a storage ref is created to upload the file. More info [firebase storage web]

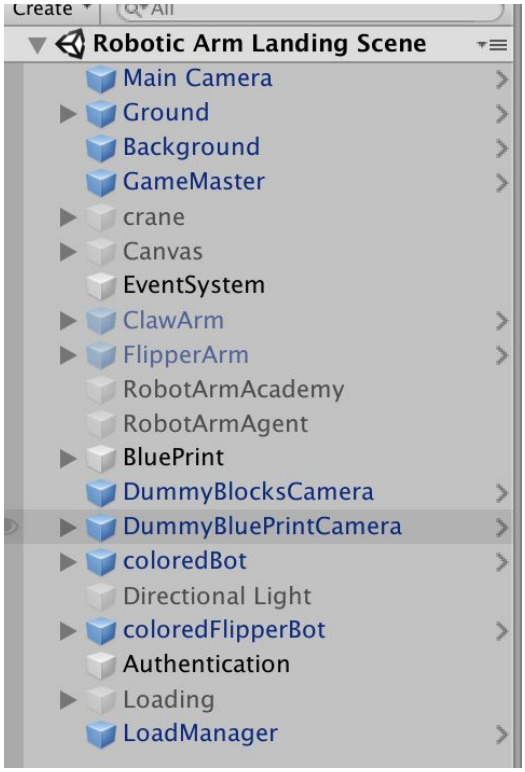
```
    // Upload the file to the bucket
    uploadFile : function(data, filename){
        console.log("JS Filename: " + filename);
        var jsData = Pointer_stringify(data);
        var jsFileName = Pointer_stringify(filename);

//      Get a reference to the storage service, which is used to create references in your storage bucket
        var storage = firebase.storage();

        //Create a storage reference from our storage service
        var storageRef = storage.ref();
        storageRef.child('alldatas/' + jsFileName + '.b64').putString(jsData).then(function(snapshot) {
        console.log('Uploaded a base64 string!');
       });
```

Load ProtoData

LoadManager: In the Landing Scene control the loading process



Loading Functions in jslib
getBucketFiles: function => populate the list with the timestamp for replay table view display
Use firebase item ref list to get all the file name from the bucket. More info of [List items]
unityInstance.SendMessage('LoadManager', 'LoadFilePath', path) : Call the C# function LoadFilePath to populate the LoadManager.fileName list

```
    getBucketFiles: function(){

//      Get a reference to the storage service, which is used to create references in your storage bucket
        var storage = firebase.storage();

        //Create a storage reference from our storage service
        var storageRef = storage.ref();
        // Create a reference under which you want to list
        var listRef = storageRef.child('alldatas');

        // Find all the prefixes and items.
        listRef.listAll().then(function(res) {
        res.prefixes.forEach(function(folderRef) {
            // All the prefixes under listRef.
            // You may call listAll() recursively on them.
        });
        res.items.forEach(function(itemRef) {
            // All the items under listRef.
            var path = itemRef.name;
            var bufferSize = lengthBytesUTF8(itemRef.name) + 1;
            var buffer = _malloc(bufferSize);
            stringToUTF8(path, buffer, bufferSize);
            // Call C# function LoadFilePaths in LoadManager to load the file list for the table view
            unityInstance.SendMessage('LoadManager', 'LoadFilePaths', path);
        });
    }).catch(function(error) {
        // Uh-oh, an error occurred!
```

getBucketFiles was called after the authentication is success in jslib function => createAuthenticationUI

```
    createAuthenticationUI : function()
    {
        window.authenticated = 0;
        var uiConfig = {
            callbacks: {
                signInSuccessWithAuthResult: function(authResult, redirectUrl) {
                    var user = authResult.user;
                    var credential = authResult.credential;
                    var isNewUser = authResult.additionalUserInfo.isNewUser;
                    var providerId = authResult.additionalUserInfo.providerId;
                    var operationType = authResult.operationType;
                    window.authenticated = 1;
                    window.user = authResult.user;
                    unityInstance.SendMessage('Authentication', 'RemoveLoadingText');
                    unityInstance.SendMessage('Authentication', 'LoadLandingScene');
                    // after authentication populate the fiilename list in LoadManager.fileNames
                    unityInstance.Module.asmLibraryArg._getBucketFiles();
```

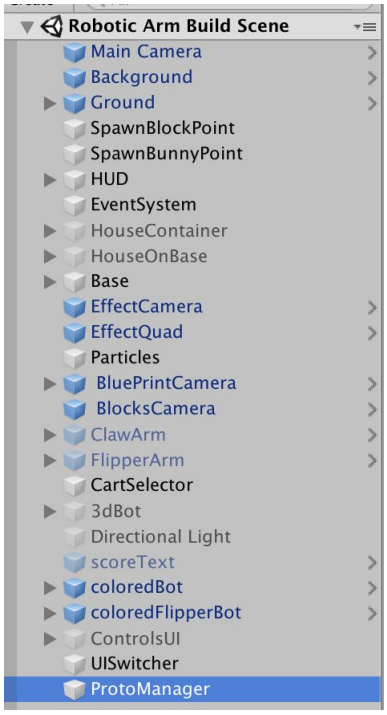
getProtoData: function(fileName) => download the file using getDownloadURL() from firebase ref. More info
Since the download URL is at different origin, have to set up the CORS Configuration
More info of Cross-origin resource sharing (CORS)

```
// Download the Protobut file
getProtoData: function(fileName){

    var jsFileName = Pointer_stringify(fileName);
    var path = "alldatas/".concat(jsFileName);
    //      Get a reference to the storage service, which is used to create references in your storage bucket
    console.log("path of the file: " + path);
    var storage = firebase.storage();

    //Create a storage reference from our storage service
    var storageRef = storage.ref()
    storageRef.child(path).getDownloadURL().then(function(url) {
        console.log("JS->Proto url: " + url);
        unityInstance.SendMessage('LoadManager', 'LoadProtoUrl', url);
       // var bufferSize = lengthBytesUTF8(url) + 1;
       // var buffer = _malloc(bufferSize);
       // stringToUTF8(url, buffer, bufferSize);
       // return buffer;
    }).catch(function(error) {
        // Handle any errors from Storage
        console.log("Request getProtoData Fail");
    });
```

LoadProtoData is in the Build Scene of ProtoManager gameobject
LoadProtoData control the replay for every frame



Python Deserialize

Protobuf-net have built in function can directly get the protobuf class structure

```
syntax = 'proto3';
message ProtoGameDetail {
    repeated float rotationClawArm = 1;
    repeated float rotationStickArm = 2;
    repeated float ClawBodyPosition = 3;
    repeated float StickBodyPosition = 4;
    repeated bool ClawController = 5;
    repeated bytes ImageByteBlock = 6;
    repeated bytes ImageByteBP = 7;
    repeated int32 CurrentBPScore = 8;
    repeated int32 CurrentScore = 9;
    repeated bool PlayerControls = 10;
    repeated int32 BlockCount = 11;
    repeated float BlocksPosition = 12;
    repeated int32 BunnyCount = 13;
    repeated float BunniesPosition = 14;
}
```

The .proto extension file can generate the proto buffer file
Run ```protoc --python_out=./ message.proto``` in the terminal
More info of [Protobuf of python]
After that we can directly import the Class and read the data

```python
import numpy as np
from message_pb2 import ProtoGameDetail

import binascii
import base64

print("START")

#with open("GameDetail", mode='rb') as file:
#    msg_bytes = file.read()

with open("fileName") as file:
    msg_file = file.read()
    msg_bytes = base64.b64decode(msg_file)

print("MIDDLE")

msg = ProtoGameDetail()
msg.ParseFromString(msg_bytes)

#for x in msg.ImageByteBlock:
for x in msg.CurrentScore:
  print(x)
# print (binascii.hexlify(bytearray(x)))

print("END")
```