brainCloud Space Shooter Tutorial

Thank you for downloading the brainCloud Space Shooter Tutorial! This example modifies the standard Unity Space Shooter example so that the player information is persisted to brainCloud.

This document along with the *tutorial video* will teach you how to:

- Create a game in the brainCloud portal
- Define a set of player statistics
- Write code to authenticate with brainCloud
- Write code to read and modify player statistics

Note that the brainCloud BaaS client SDK is already included in the brainCloud Space Shooter Asset Store package so you do not need to download or import it into your project.

Step 1: Download the Unity Asset Store package

Since you are reading this document, the assumption is that you have already done this step. However you can locate the brainCloud Space Shooter tutorial package by searching on the keyword "braincloud" in the Unity Asset Store, or by navigating directly to this link:

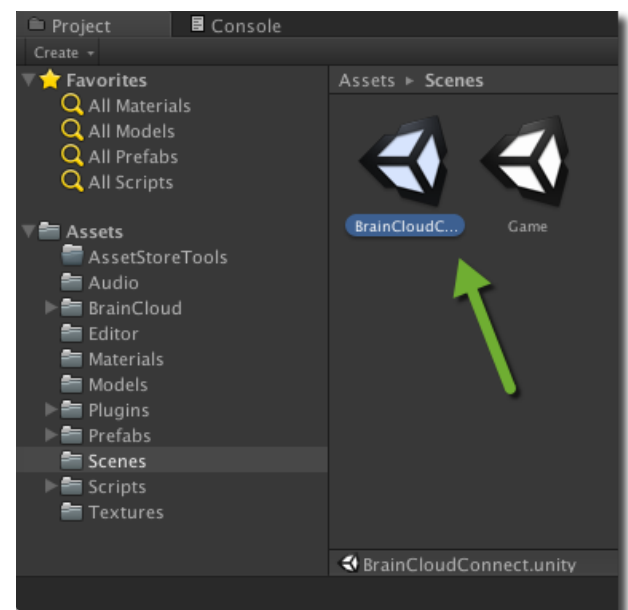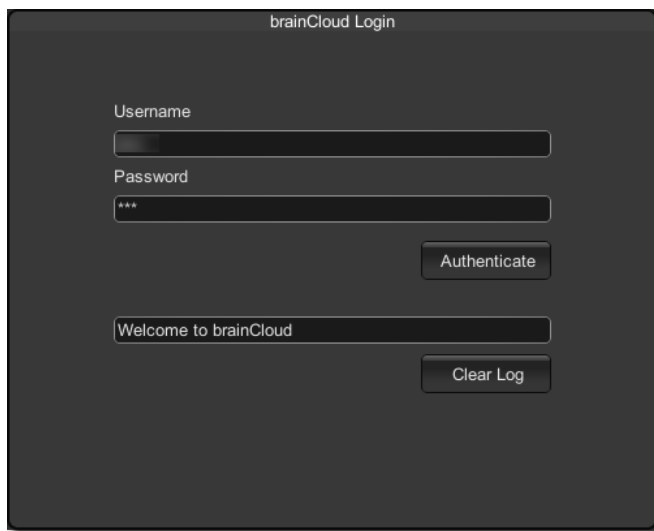https://www.assetstore.unity3d.com/#!/content/50220

Step 2: Try the game

Once you've downloaded and imported the brainCloud Space Shooter Unity project, the game should be ready to run.

Open the **Assets | Scenes | BrainCloudConnect** scene.

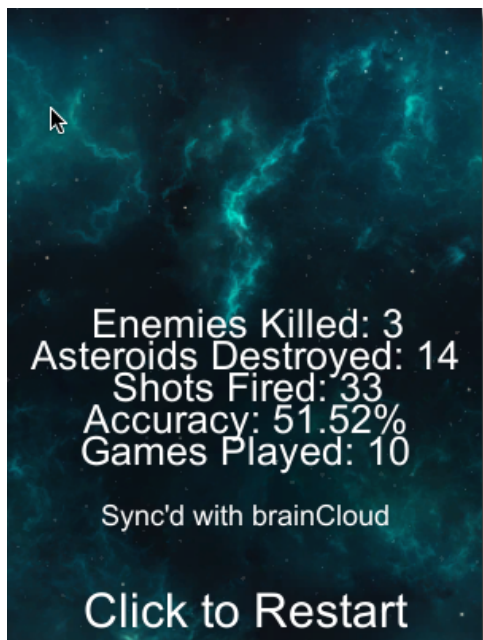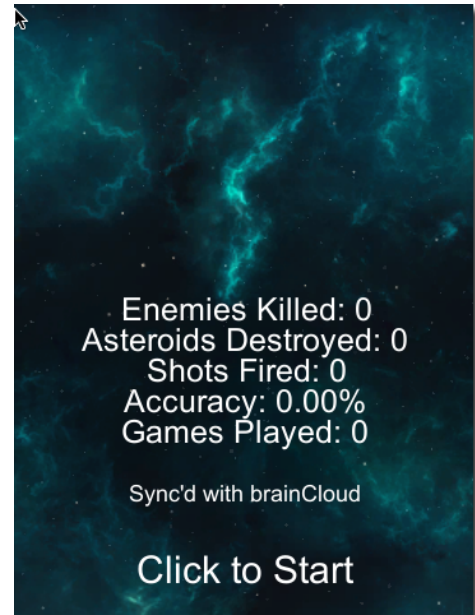Hit the **Play** button and you should be presented with a login dialog.

Enter a *username* and *password*. You can use any values you'd like as the system will automatically create a user account if one does not already exist.

On successful authentication you should see the space shooter background and a list of your statistics. If you've created a brand new player, the statistics should all be set to 0. Click a button and give the game a whirl.

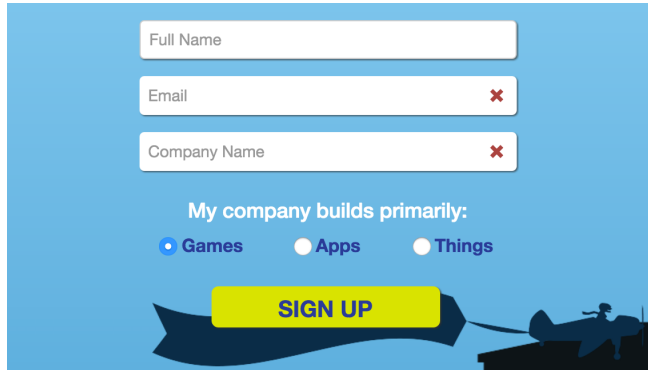Note the controls for the space shooter are:

- WASD or Arrow keys to move the ship
- Mouse button 1 to fire

Enemies Killed: 0
Asteroids Destroyed: 0
Shots Fired: 0
Accuracy: 0.00%
Games Played: 0

Sync'd with brainCloud

Click to Start

Enemies Killed: 3
Asteroids Destroyed: 14
Shots Fired: 33
Accuracy: 51.52%
Games Played: 10

Sync'd with brainCloud

Click to Restart

Once you've been destroyed, you will get a summary of your statistics. These statistics are being persisted to brainCloud.

Step 3: Hook it up to your own account

The game you've just played is hooked up to one of our demo accounts. To better understand how things work, we'll take you through the steps of duplicating this setup in your own brainCloud account.

If you haven't already done so, you'll want to create a free account on the brainCloud servers.

Navigate to:
http://getbraincloud.com/signup/

and follow the steps to register an account.

Account setup? Great - now watch the video and follow along:

Tutorial video - https://www.youtube.com/watch?v=hvgaJ74fFLo

Digging into the code

The following sections give a bit more information about what's happening under the covers.

Authentication

The code to authenticate with brainCloud can be found in **Assets | Scripts | BrainCloudConnectScene.cs.** This class handles drawing the Login dialog as well as authenticating with brainCloud. In the `Start()` method of this class you will find a call to initialize brainCloud:

```
BrainCloudWrapper.Initialize();
```

Later on in the OnWindow() method, you will find the code to Authenticate with brainCloud with the username and password of the user:

```
BrainCloudWrapper.GetInstance().AuthenticateUniversal(
  m_username, m_password, true,
  OnSuccess_Authenticate, OnError_Authenticate);
```

The "true" flag indicates that we want a new account to be created if the user does not already exist. Note also that success and error callback methods are registered in this function. We can find their definitions in this script file as well:

```
public void OnSuccess_Authenticate(string responseData, object cbObject)
{
  AppendLog("Authenticate successful!");
  Application.LoadLevel("Game");
}

public void OnError_Authenticate(int statusCode, int reasonCode,
  string statusMessage, object cbObject)
{
  AppendLog("Authenticate failed: " + statusMessage);
}
```

The success callback loads the main Game scene.

Player statistics

The code for reading and writing player statistics to brainCloud is located in **Assets | Scripts | SpaceShooterTutorial | GameController.cs**. You will find two functions at the beginning of the file:

```
private void ReadStatistics()
{
  // Ask brainCloud for statistics
  BrainCloudWrapper.GetBC().PlayerStatisticsService.ReadAllPlayerStats(
    StatsSuccess_Callback, StatsFailure_Callback, null);

  brainCloudStatusText.text = "Reading statistics from brainCloud...";
  brainCloudStatusText.gameObject.SetActive(true);
}


private void SaveStatisticsToBrainCloud()
{
  // Build the statistics name/inc value dictionary
  Dictionary<string, object> stats = new Dictionary<string, object> {
    {"enemiesKilled", m_enemiesKilledThisRound},
    {"asteroidsDestroyed", m_asteroidsDestroyedThisRound},
    {"shotsFired", m_shotsFiredThisRound},
    {"gamesPlayed", 1}
  };

  // Send to the cloud
  BrainCloudWrapper.GetBC().PlayerStatisticsService.IncrementPlayerStats(
    stats, StatsSuccess_Callback, StatsFailure_Callback, null);

  brainCloudStatusText.text = "Incrementing statistics on brainCloud...";
  brainCloudStatusText.gameObject.SetActive(true);
}
```

As expected, the `ReadStatistics()` method reads the player statistics from brainCloud. The `StatsSuccess_Callback` method handles the return JSON from this method. Similarly, the `SaveStatisticsToBrainCloud()` method increments the current statistic values on brainCloud.

Note that the same callback is used for this method to update the current values of the statistics within the game. The callback is show below:

```
private void StatsSuccess_Callback(string responseData, object cbObject)
{
  // Read the json and update our values
  JsonData jsonData = JsonMapper.ToObject (responseData);
  JsonData entries = jsonData["data"]["statistics"];

  m_statEnemiesKilled = int.Parse(entries["enemiesKilled"].ToString());
  m_statAsteroidsDestroyed =
    int.Parse(entries["asteroidsDestroyed"].ToString());
  m_statShotsFired = int.Parse(entries["shotsFired"].ToString());
  m_statGamesPlayed = int.Parse(entries["gamesPlayed"].ToString());

  ShowStatistics();

  if (brainCloudStatusText)
  {
    brainCloudStatusText.text = "Sync'd with brainCloud";
  }
}
```

The callback is responsible for parsing the JSON string and updating the local copy of the statistics.

brainCloud API Reference

For the complete reference of available APIs refer to the brainCloud APIDocs at:

http://getbraincloud.com/apidocs

For more Unity tutorials, go to:

http://getbraincloud.com/apidocs/tutorials/unity-tutorials/

Happy Coding!