brainCloud Space Shooter Tutorial

Thank you for trying the brainCloud Space Shooter Tutorial!

This example modifies the standard Unity Space Shooter example so that the player information persists in brainCloud.

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

Download the Space Shooter example package

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:

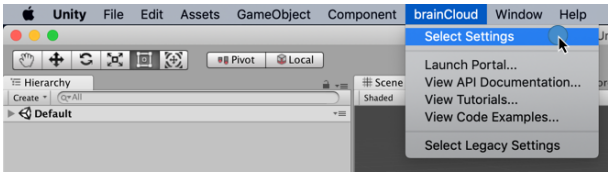
https://www.assetstore.unity3d.com/#!/content/50279

If your reading this, you probably have already downloaded the brainCloud Shooter Shooter example :)

Hook the example game up to your brainCloud account

For the game to connect to the brainCloud server, you need to first signup with an account. *brainCloud is free during development. See our* pricing page *for more information.* You can sign up to brainCloud directly from Unity.

In Unity, select **brainCloud | Select Settings**, to be brought to the brainCloud plugin interface.



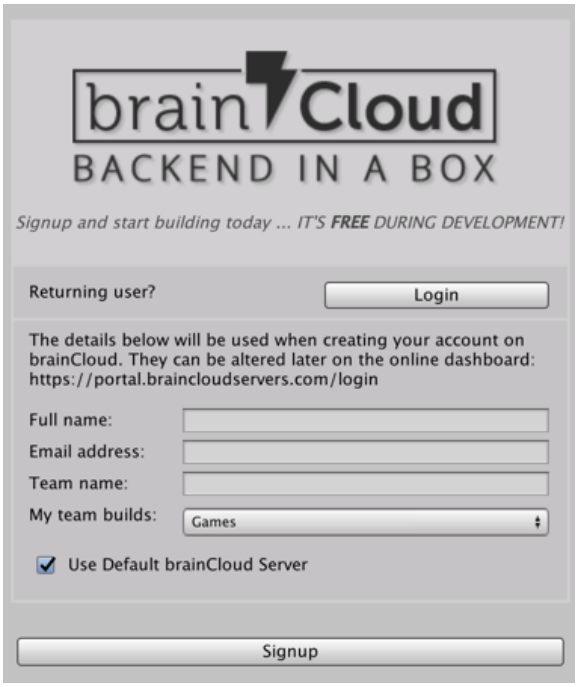
If brainCloud is not one of the available drop-downs, make sure you have properly imported the example package. *Latest development packages of our client SDK can be found here*

From the main plugin page, click **Signup** to create your new account.

*If you have already had a brainCloud account, you can use **Login** instead.*

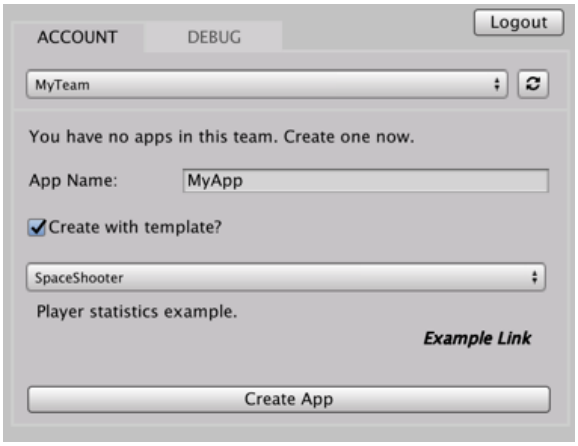
Once done registering your account, you will be sent an email to set the password to your account.

Note: Your email might be in your junk folder.



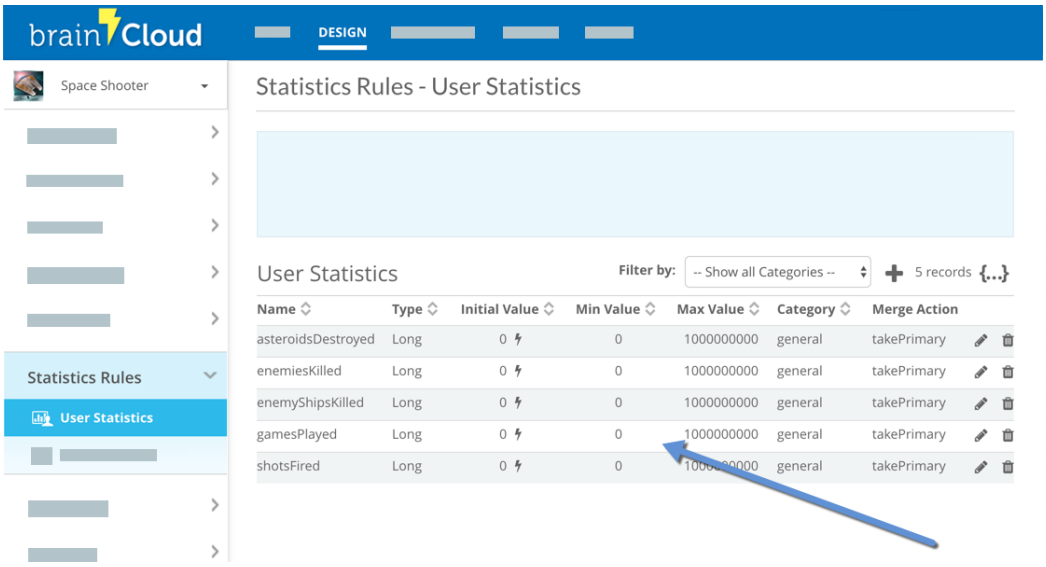
For creating your app, please select the **Create with template?** option, and choose the **SpaceShooter** example. The template will set your app up with the same configurations found in the SpaceShooter example.

Now when you play the game, you will be sending the data to your copy of the app!

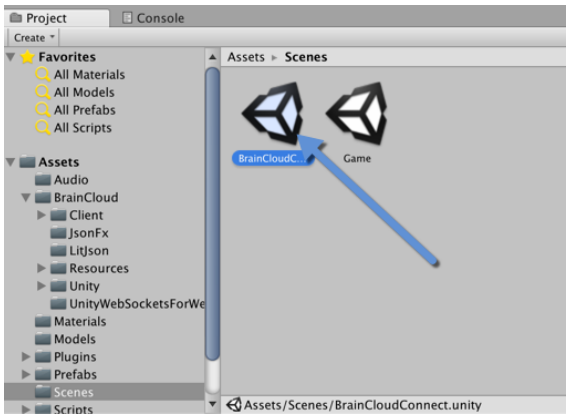


You can go to the brainCloud dashboard with your login details at this link, and check out the data on the dashboard — *login with the account login details you just made earlier*

You'll see the User Statistics from the SpaceShooter template under **Design | Statistics Rule | User Statistics**



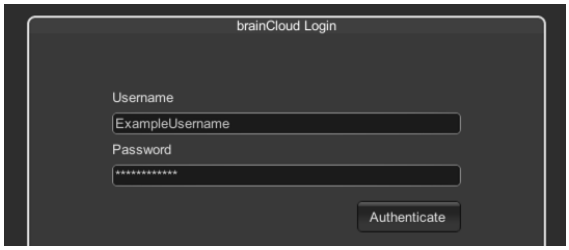
Try the game



Now that the game is hooked up with brainCloud, you should try it out!

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

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



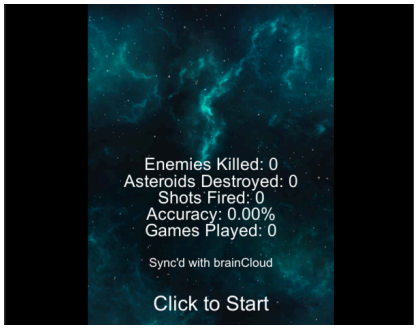
Enter a *username* and *password*. You can use any values you'd like as the system will automatically create a user account if one does not already exist.

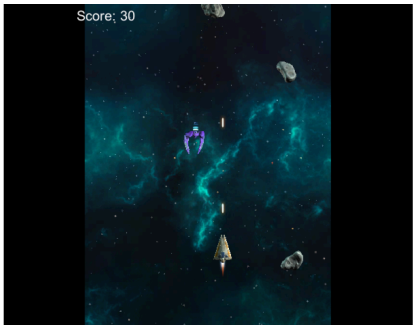
On successful authentication, you should see the space shooter background and a list of your statistics. If you've created a brand new player, the statistics should all be zero. Click a button and give the game a whirl.

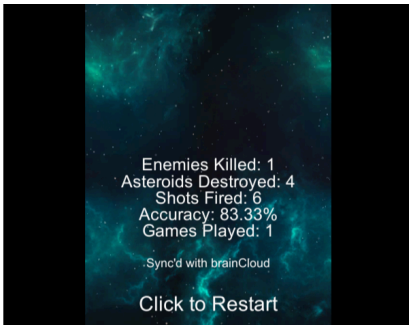
Note the controls for the space shooter are:

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

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







You can check this out the saved player statistics on the brainCloud dashboard, on the **Monitoring | User Monitoring | Statistics** page.

Initialization

In the code, we initialize the brainCloud Client in **Assets | Scripts | App.cs.** We store an instance of it to use throughout our game.

```
private void Awake()
    {
        Bc = gameObject.AddComponent<BrainCloudWrapper>(); // Create the
brainCloud Wrapper
        DontDestroyOnLoad(this); // on an Object that won't be destroyed on
Scene Changes

        Bc.WrapperName = WrapperName; // Optional: Add a WrapperName
        Bc.Init(); // Required: Initialize the Wrapper

        Bc.Client.EnableLogging(true);
    }
```

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

```
App.Bc.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
  App.Bc.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
  App.Bc.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 shown 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!