

# SCORM TEST APPLICATION

## A GUIDE TO THE EXAMPLE INTEGRATION APPLICATION

*This guide explores the functionality of the example SCORM Test Application. This example application showcases the majority of the Unity-SCORM Integration Kit functions. We will explore the various functions as a way of helping you understand how you can implement the Integration Kit in your own project.*

### Getting Started With the prebuilt SCORM Package

You can see the example SCORM Test Application in action by downloading the [Unity-SCORM Integration Kit.zip](#) SCORM package and loading it into your LMS.

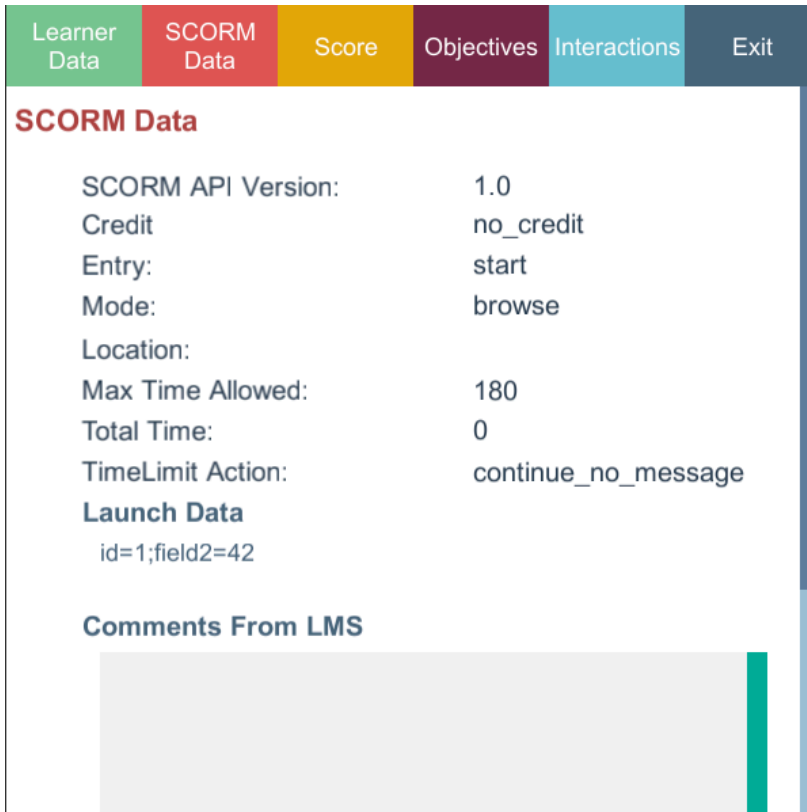
1. When you first open the Test Application, it will load all possible data from your LMS. You can see exactly which SCORM calls are made and the data returned in the *Log Window* at the right-hand side.

```
Log 00:03:18
Set cmi.objectives.0.id to urn:STALS:objective-id-0
Result true
Set cmi.objectives.0.score.scaled to 0
Result true
Set cmi.objectives.0.score.raw to 0
Result true
Set cmi.objectives.0.score.max to 100
Result true
Set cmi.objectives.0.score.min to 0
Result true
Set cmi.objectives.0.success_status to unknown
Result true
Set cmi.objectives.0.completion_status to not attempted
Result true
Set cmi.objectives.0.progress_measure to 0
Result true
Set cmi.objectives.0.description to 2323
Result true
Set cmi.interactions.0.id to urn:STALS:interaction-id-0
Result true
Set cmi.interactions.0.type to other
Result true
Set cmi.interactions.0.timestamp to 2015-06-30T16:23:27
Result true
Set cmi.interactions.0.weighting to 1
Result true
Set cmi.interactions.0.learner_response to 1
Result true
Set cmi.interactions.0.result to correct
Result true
Set cmi.interactions.0.latency to P0DT0H0M18.00S
Result true
Set cmi.interactions.0.description to 1
Result true
```

2. The **Learner Data** tab displays information about the learner that was returned from the LMS.

- a. You can set the Learner Preference data by changing a value in the input fields. You will notice that as soon as you finish editing a value, it is sent to the LMS.
- b. You can add a new Learner Comment using the form. The *Comment Location* is designed to be used to store the point in your learning object that the learner made the comment. Add a comment and you will see it being sent to the LMS (in the Log window) and the new comment will be added to the *Learner Comment List*.

3. The **SCORM Data** tab displays the various read-only data sent from the LMS. The LMS does not enforce any of the options displayed here. For example, it is up to you to implement the workflow. (For example, if the Max Time Allowed is set, it is up to you to do something with it, the LMS will not prevent the student using the learning object past this time).

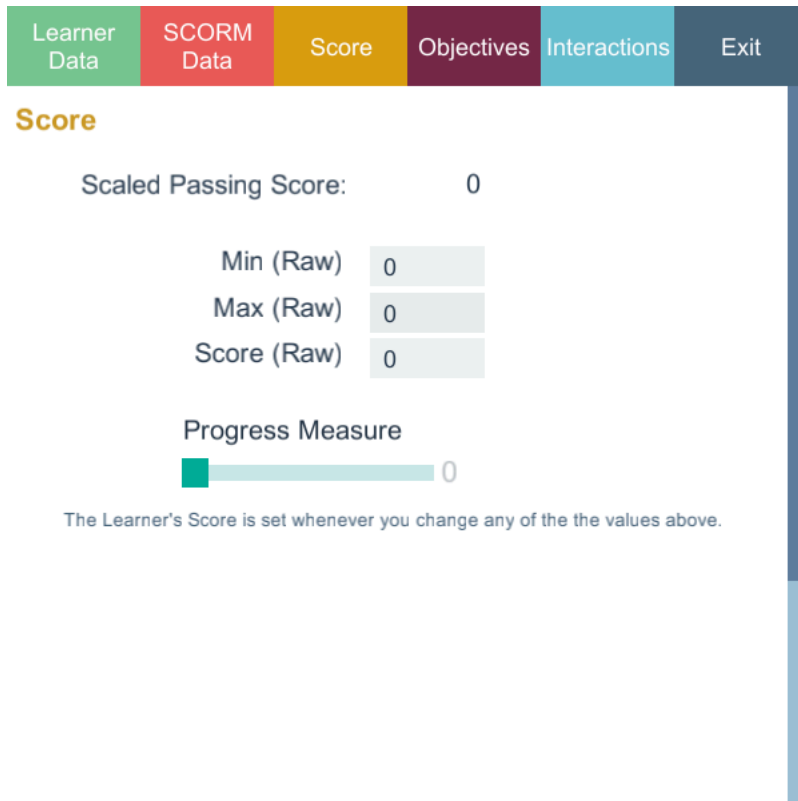


Learner Data	SCORM Data	Score	Objectives	Interactions	Exit
<b>SCORM Data</b>					
SCORM API Version:	1.0				
Credit	no_credit				
Entry:	start				
Mode:	browse				
Location:					
Max Time Allowed:	180				
Total Time:	0				
TimeLimit Action:	continue_no_message				
<b>Launch Data</b>					
id=1;field2=42					
<b>Comments From LMS</b>					
[Empty comment box]					

- SCORM API Version – not terribly useful, it may indicate a different version of the SCORM 2004 API at some later date.
- Credit – Indicates whether the learner will be credited for performance in the SCO
- Entry – Asserts whether the learner has previously accessed the SCO
- Mode – Identifies one of three possible modes in which the SCO may be presented to the learner (browse, normal, review).
- Location – Any previously set Bookmark.
- Max Time Allowed – in seconds (this is set in the imsmanifest.xml file).
- Total Time – the total time spent on this learning object.
- Time Limit Action - Indicates what the SCO should do when Max Time Allowed is exceeded.

- i. Launch Data – any parameters passed to the SCO on launch (this is set in the imsmanifest.xml file).
- j. Comments from the LMS – not implemented in many LMSs.

4. The **Score** tab displays the students score for this attempt.



**Score**

Scaled Passing Score: 0

Min (Raw)	0
Max (Raw)	0
Score (Raw)	0

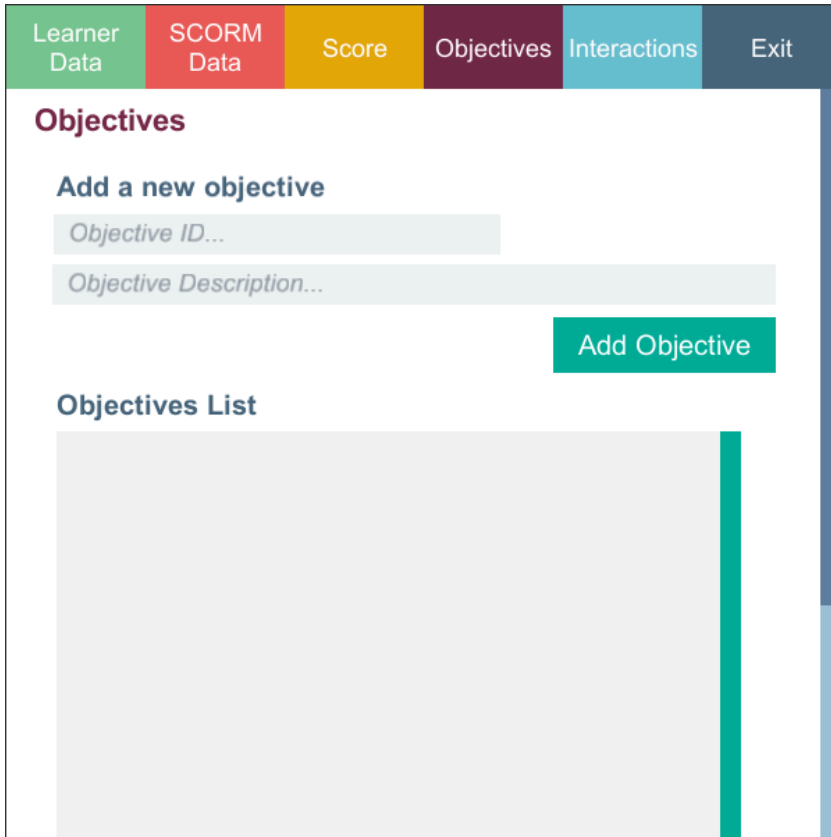
Progress Measure

0

The Learner's Score is set whenever you change any of the the values above.

- a. Scaled Passing Score – if set.
- b. Min – the minimum possible raw score (normally 0 is a sensible value).
- c. Max – the maximum possible raw score.
- d. Score – the earned raw score (between min and max). Whenever you change the Max or Score values, the Scaled score is re-calculated and also sent to the LMS.
- e. Progress Measure – indicates how much of the learning object has been completed.

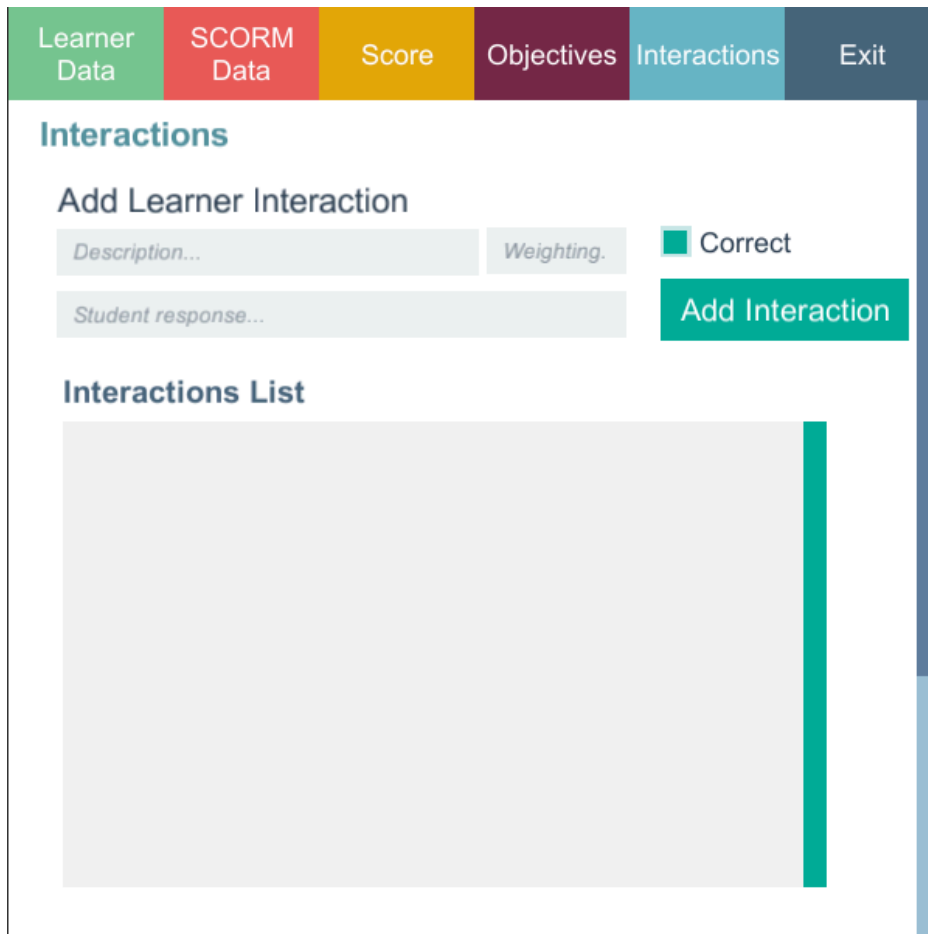
5. The **Objectives** tab displays objectives set for this learning object.



The screenshot shows a web interface with a top navigation bar containing six tabs: 'Learner Data' (green), 'SCORM Data' (red), 'Score' (yellow), 'Objectives' (dark purple, currently selected), 'Interactions' (light blue), and 'Exit' (dark blue). Below the navigation bar, the 'Objectives' section is titled 'Objectives' in dark purple. Underneath, there is a sub-section 'Add a new objective' which includes two text input fields: 'Objective ID...' and 'Objective Description...'. To the right of these fields is a green button labeled 'Add Objective'. Below this form is an 'Objectives List' section, which is currently empty and represented by a large grey rectangular area.

- Use the add objective form to create a new objective. The Objective ID must be unique for this attempt.
- When you create a new objective, you can then set the additional values such as the Score, Progress Measure and Success and Completion statuses.
- Objectives are also intended to be able to be associated with Student Interactions. However, this version of the Test Application does not facilitate this.

6. The **Interactions** tab displays the interaction items set for this student. Interactions are typically answers to quiz questions or tracked activity in the learning object.



- a. Use the Add Interaction form to create a new interaction.
- b. Once created, you will see the interaction listed. For the purposes of this Test Application, the interaction type is set to *other* and the latency (time spent on this interaction) is set to 18.4 seconds. Please see the Guide to the Unity-SCORM Integration Kit Functions for detail on what is possible here.

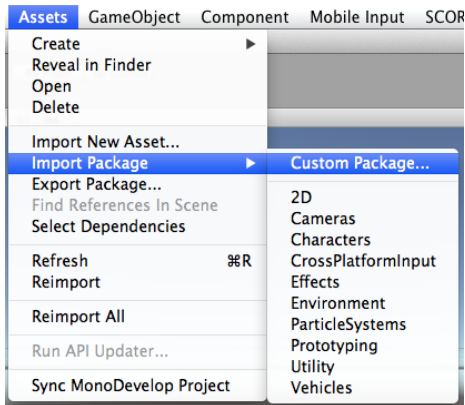
7. The **Exit** tab displays the various options that should be set when exiting the learning object.

- a. The Completion Threshold – what progress measure indicates that the learning object has been completed?
  - b. Success status – did they pass or fail this learning object?
  - c. Completion status – setting the completion status to complete will mean that the student will not be able to add any more interaction data to this attempt.
  - d. Click the Exit SCORM button to exit. This will generally prevent any further interaction in this attempt. Any subsequent access of the learning object will open a new attempt (if allowed by the LMS). Or,
  - e. Set a Location (bookmark) and suspend the attempt. This will reopen this attempt on subsequent access of the learning object by the same student. The student will be able to add to the interactions of this attempt (continue).
8. Closing the window will force a call to the API.Commit() function saving any data set along the way.

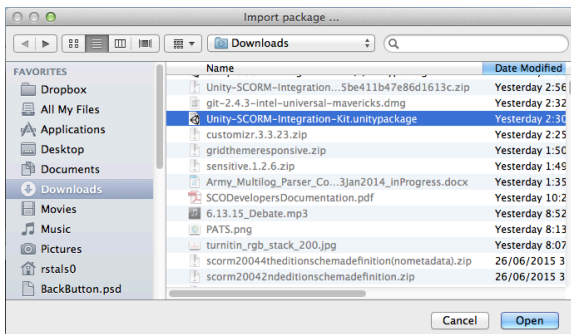
## Getting Started in Unity3D

### Import the Unity-SCORM Integration Kit

1. Download [Unity-SCORM-Integration-Kit.unitypackage](#).
2. Open your Unity3D project (or create a new project). On the top menu, select **Assets > Import Package > Custom Package...**

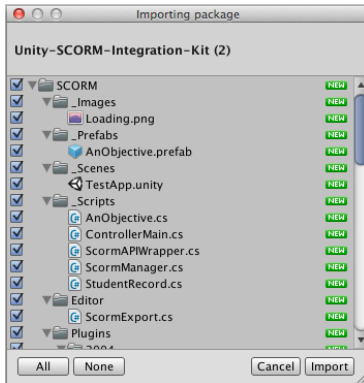


3. Select the Unity-SCORM-Integration-Kit.unitypackage file you downloaded in step 1 and click the **Open** button.

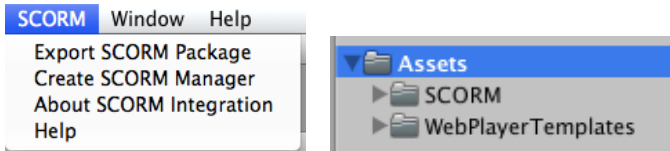




4. On the Importing package window, click the **Import** button.

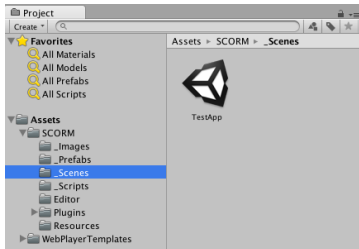


5. The Integration Kit is imported into your project. To refresh the menu, click on any of the menu items. You will then see a new SCORM menu item and two new folders: *SCORM* and *WebPlayer Templates*.



## Set up your Unity project

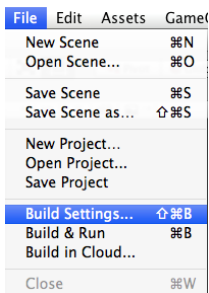
1. Expand the **SCORM** folder, click on the **\_Scenes** folder and double click the **TestApp** Unity3D scene to open it.



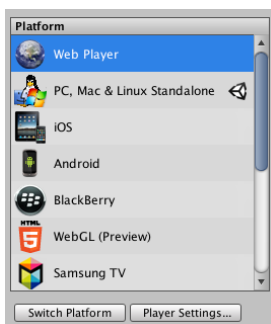
2. Set your *Scene window* option to display in 2D mode. Click the **2D** button at the top of the *Scene window*.



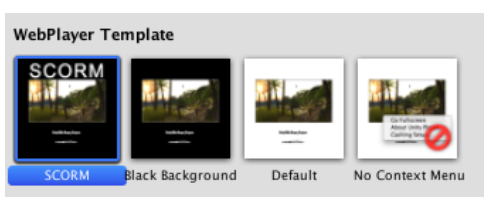
3. Access the *Build Settings*. On the menu select **File > Build Settings...**



4. In the *Platform* section, select **Web Player** and then click the **Switch Platform** button.

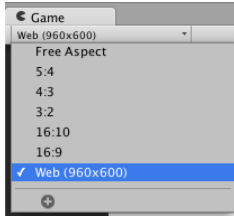


5. Now click the **Player Settings** button. In the *Inspector window*, change the *WebPlayer Template* to **SCORM**.

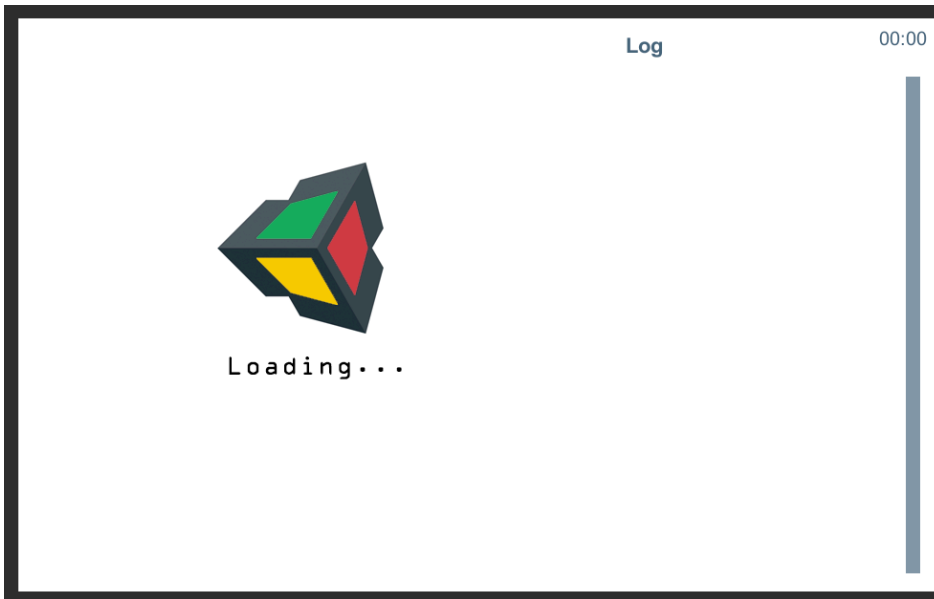


You can close the *Build Settings* window.

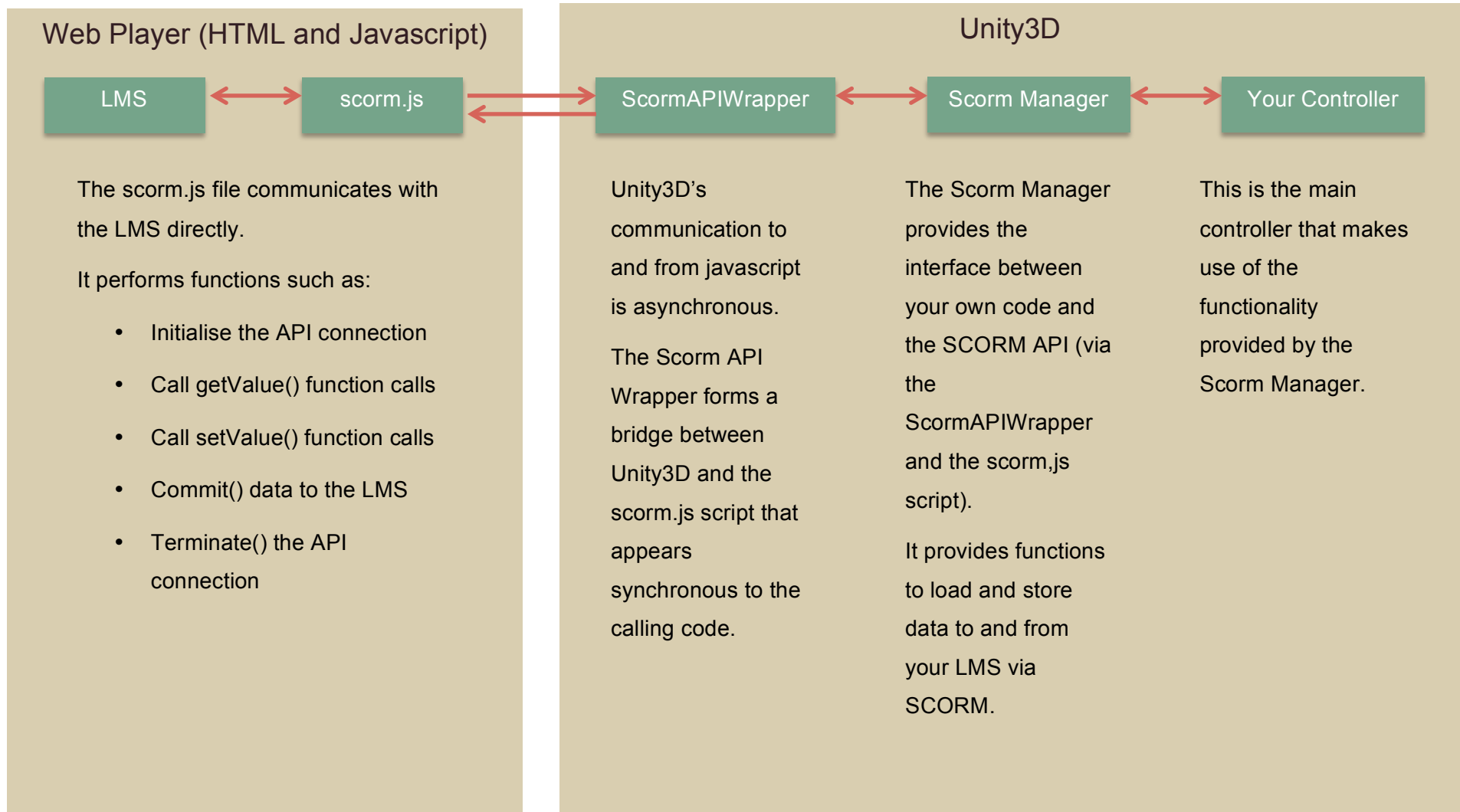
6. Finally, change the *Game window* size to **Web (960x600)**.



7. The SCORM Test Application is now ready in your Unity3D project.



## Architecture Overview



## Basic SCORM Application Workflow

### Setup

To set up a new Unity3D scene ready for SCORM integration, create a new scene and then click the **SCORM** item on the menu, then select **Create SCORM Manager**.

This creates a new GameObject on your scene hierarchy called **ScormManager**. This has the ScormManager.cs script attached and provides the connection to the SCORM API.

**IMPORTANT:** Your main controller script that communicates with the ScormManager must be attached to a GameObject that is a child of the ScormManager GameObject. For example, in the Test Application see the ControllerMain GameObject. This is because the ScormManager communicates key events to your code by sending broadcast messages to any GameObjects that are its children.

### Workflow

1. The ScormManager launches a thread that ensures the SCORM API is available in the Web Player and then loads any data from the LMS into the StudentRecord.
2. When this is complete, the ScormManager broadcasts the Scorm\_Initialize\_Complete message to your code. Your code should not use the inbuilt Unity3D Start() function. Instead, put your initialisation code in a public function called Scorm\_Initialize\_Complete().
3. It is good practice to set the Completion Status to Incomplete if this is the first attempt (i.e. StudentRecord.CompletionStatusType.incomplete). This is particularly important when using your project in Blackboard Learn otherwise Blackboard will set your attempt to complete by default.
4. Your code then runs. Call the various Get...() functions to read the stored LMS data and call the various SET...() functions to set values to be sent back to the LMS.
5. When complete, set the session time (ScormManager.SetSessionTime()), the exit type (ScormManager.SetExit()) and then call the Commit function (ScormManager.Commit()).
6. The ScormManager will commit your attempt data to the LMS. When this is finished, it will send the Scorm\_Commit\_Complete message. Do not quit or close the application until your code receives the Scorm\_Commit\_Complete message as you may stop the data being written to the LMS completely.

7. Typically, your Scorm\_Commit\_Complete function will simply call the Terminate function in the ScormManager (ScormManager.Terminate()). This in turn sends a message to close the browser window.

There is one additional (optional) function that the ScormManager will call in your code. Any logging information can be manipulated by creating a public function called Log (i.e. public void Log(string data)). In the example Test Application, this functions displays the log data in a text field on the right-hand side of the screen.

## An Overview of the Assets

### 1. SCORM

- a. `_Images`
  - i. Loading - The Loading Screen sprite image
- b. `_Prefabs`
  - i. AnObjective – The Unity3D prefab used whenever a new objective is loaded or created via SCORM.
- c. `_Scenes`
  - i. TestApp – The SCORM Test Application
- d. `_Scripts`
  - i. AnObjective.cs – The functionality required to display a SCORM objective on the AnObjective prefab.
  - ii. ControllerMain.cs – Drives the Test Application. This script demonstrates how you can use the Integration Kit in your own project.
  - iii. ScormAPIWrapper.cs – This acts as the bridge between the SCORM API (served via the scorm.js file that is packaged in the WebPlayer) and Unity3D code.
  - iv. ScormManager.cs – The Unity3D side of the SCORM bridge. This contains all the functions needed to load, access and store SCORM data.
  - v. StudentRecord.cs – A utility class that simply stores the structure of the data loaded via SCORM.
- e. Editor
  - i. ScormExport.cs – The file that drives the SCORM menu item. The most important function is that of exporting and packaging your Unity3D project ready for loading into your LMS.
- f. Plugins
  - i. 2004 (folder) – contains the set of files required to create a SCORM 2004 Compliant package. Used by the ScormExport.cs script when creating your package.

- ii. Ionic.Zip.Reduced.dll – A library for creating Zip files. Used by the ScormExport.cs script when creating your package.

g. Resources

- i. ScormManager – A prefab used to set up a new scene for SCORM integration. Used by the ScormExport.cs script when you select *Create SCORM Manager* from the *SCORM* menu.

2. WebPlayer Templates

- a. SCORM – The SCORM WebPlayer Template to select in the *Build Settings*.

- i. Index.html – the template for the index.html file used when you build this project. Contains the Unity3D Web Player as well as a reference to the scorm.js API file.
- ii. thumbnail – A thumbnail for the Player Settings window.
- iii. scripts

1. scorm.js – the file that communicates with the LMS via SCORM.
2. ScormSimulator.js – this can be used to test your own project without having to load it into an LMS first.