Simple Scores - version 1.1

Simple Scores is a very easy to use highscore system for Unity.

The test scene included presents a very simple game showing the player accumulating a score and submitting it to the leaderboard upon completion.

The GUI that is included is a very simple layout and can be used, edited or even removed at the users free will. The scripts will need to be modified slightly if the GUI were to be replaced by the users custom UI or even by other plugins such as ngui. A brief description of the included GUI script (OnScreeNGUI.cs) will be added below.

Placing SimpleScores into your scene:
1) Import the HighscoreSystem prefab into your folder structure.
2) Drag the HighscoreSystem prefab into your scene where relevant.

Required components
To be able to use our entire asset as standard, there are specific scripts that need to be available on gameobjects being used in the game scene. The player object must have a script attached to it called "PlayerScore.cs", this is the script that holds the player's current score and is used as a reference within the Scoretracker script that is attached to the SimpleScore's object.

There is a public list within the inspector that needs to be filled with the current player's score class, so make sure you drag the player object onto the list before starting the game so the reference can be made.

The method of adding scores to the highscore system has been changed slightly due to different code arrangements. The new method can be seen within the example scene in the PlayerController class supplied. Upon "completing" the level (simply walk into each red cube and destroy each one) the code finds the SimpleScores object and passes through the new data by calling the PushScoreToBoard() function within the Highscore.cs class. In the example the supplied GUI is toggled on to render the scoreboard displaying the results of the example scene. The function will handle the loading and saving of data to the devices local storage. If the data is needed to be stored online, you can view the method of doing so below.

If you choose to not use our entire scoring asset to store your data then the code will need to be adapted slightly but feel free to do so, however editing the classes provided above will allow faster storing and easier usage of our scoring system and highscore board.

Team based scoring is also available, this works as usual with each player accumulating their own score, however there is a boolean called "Store Score As Team" within the ScoreTracker script that can be set to true to allow the team of players to store their total score as one submission.

<u>Local Storage</u>

Saving with SimpleScores:
Local data storage has been implemented into SimpleScores, allowing the game to store the current leaderboard state to the device for future use. The stored scores can be saved by calling the public function SaveValuesLocally(). Note that the amount of scores saved is subject to the public variable Number of Scores set at the beginning. It is also worth noting that this function could become memory intensive and may cause performance issues if it were called excessively, for example in an Update loop. To make sure this process doesn’t slow down your gameplay call it at important times, for example when completing a level or objective.

Loading with SimpleScores:
Local data can be loaded whenever is necessary. For example, if data is stored on the last play session, then that same data could be loaded the next time round for persistent use throughout playtime. This can be done by calling the public function LoadValuesLocally(). Note that the amount of scores loaded is subject to the public variable Number of Scores set at the beginning. It is also worth noting that this function could become memory intensive and may cause performance issues if it were called excessively, for example in an Update loop. To make sure this process doesn’t slow down your gameplay call it at important times, for example when loading or initialising your leaderboard state.

Deleting and Clearing Data:
Locally stored data can be removed by calling the public function DeleteAllLocallySavedData(), please not that this function is irreversible so once the data has been deleted it cannot be retrieved.

The scoreboard can also be cleared so no information is being shown on screen. This is accomplished by calling the public function ClearScoreBoard(). Calling this function will not delete any stored data like the above function, but it will clear the current scores within the game instance so make sure any data that needs to be used again is saved locally or online before calling this function.

<u>**Online Storage**</u>

PHP Scripts
We have included some PHP scripts to that can be used to store your data online using your own web server. If you have never used PHP or even heard of PHP before do not worry! We will walk you through adapting them for your personal use right now!

There are two scripts that will be needed to use our asset in its entirety. These are getscores.php and savescores.php. To get started, you will need to edit the some simple variables within the scripts before uploading them to your website.

The variables that need changing are listed below:

$db_name = ‘databasename’;	- the name of the database being accessed
$username = ‘username;	- the username used to login to the database
$password = ‘password’;	- the password used with the username above
$host = 'localhost';	- the host of the database, normally just localhost
$tbl_name = ‘tablename;	- the name of the table being used to store the data

If you do not have a database and table already created then web hosting tools, such as cpanel, can be used to do so. In our PHP script we access two different columns to store the scores, these are called ‘name’ and ‘score.’ For lack of confusion please create 3 columns within your table - the first one will be an auto incrementing column called ID, the second will be called name and be of type varchar, the third and final one will be called score and be of type integer.

Saving to an Online Database:
The public function SaveValuesToDatabase() can be called from outside of the class in order to store the values online. This function communicates with the web server by calling the PHP script “savescores.php” which sends the local game data (name and score) through to the PHP and tries to save it to the assigned database.

Loading from an Online Database:
The public function LoadValuesFromDatabase() can be called from outside of the class in order to load the values from an online database. This function communicates with the web server by calling the PHP script “getscores.php” which loads the data from the database and returns it to the C# environment. The data is then turned back into usable data for the C# and shown within the on screen scoreboard.

Clearing Online Database Scores:
The public function ClearValuesFromDatabase() can be called to clear the currently used table of all the information. This function does not delete the table itself, just deletes all the rows of information from the database. Warning, once this function has been called the process cannot be reversed. So make sure data is backed up or unwanted before proceeding.

Included GUI

The artwork that we've provided with SimpleScores is available from: http://bit.ly/1miRPaC

Changing the UI:
Included in this package is a very simple UI that runs within the script OnScreenGUI.cs to show the use of SimpleScores. Some aspects of the UI are hard placed within the editor, where as others (such as the buttons) are purely scripted.

There are multiple on screen buttons that should be taken into consideration, these are:
- Save Local: Saves the current scoreboard data locally.
- Load Local: Loads the currently saved scoreboard data.
- Delete Local: Deletes all the locally saved data (this could effect other saved data).
- Clear Board: Clears all the current scoreboard data, presenting a fresh board.
- Up: Moves up the leaderboard showing currently hidden values.
- Down: Moves down the leaderboard showing currently hidden values.

None of these buttons need to actually exist on an in game UI, but the functions called by them will need to be used elsewhere in the script. For example, when loading the scoreboard, the LoadValues() function may want to be called to show persistent data and upon leaving the scoreboard the save function may want to be called.

Within the Update function of the OnScreenGUI.cs is a loop that reads the specific values to show on screen. This is done by taking the section of values chosen with the up and down buttons and adds them to the scoreFieldText gameobject, which is a simple GUIText object in Unity. If this is to be changed to a custom UI, then certain changes would need to be made. Note, these scripts can be edited and changed in any way needed.

Within the OnGUI() function is the code to show and listen for button actions. A lot of the code is to help scale the on screen UI when changing the screen size. However, this would only be necessary for specific deployment as many game windows cannot be resized during play time, for example full screen applications are a set size and don't require rescaling calculations.