Database for Leaderboard and NetworkSceneManagement

There are a few ways to implement leaderboards like using Steam API (<https://partner.steamgames.com/doc/features/leaderboards/guide>) or Firebase (<https://firebase.google.com/docs/database/unity/start>). However for this lab we will do the leaderboards using a database and server-side scripting which would are skills transferable to other type of programming. This lab material is based on videos from <https://www.youtube.com/watch?v=SKbY-0zt2VE>.

# Save data to database

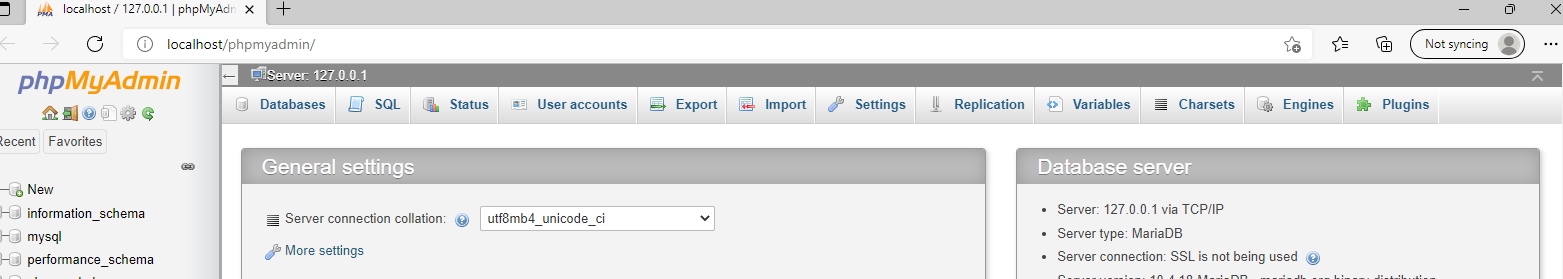
1. Install XAMPP which is a package of webserver and MySQL database : <https://www.apachefriends.org/index.html>
2. Run XAMPP and Start Apache and MySQL from XAMPP’s control panel
3. Table, calendar

   Description automatically generatedMySQL is a relational database system which consist of Database and Table consisting Rows and Columns. In MySQL there can be many database. Inside a database the data is stored in a form of tables which the columns will have headers and the data will be stored in a form of rows (in a sense it is similar to Excel sheet)

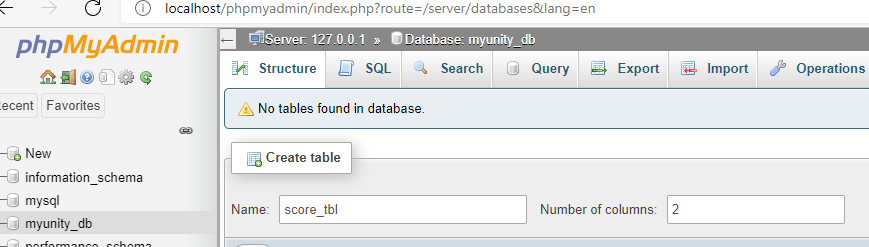
Table, calendar

Description automatically generated

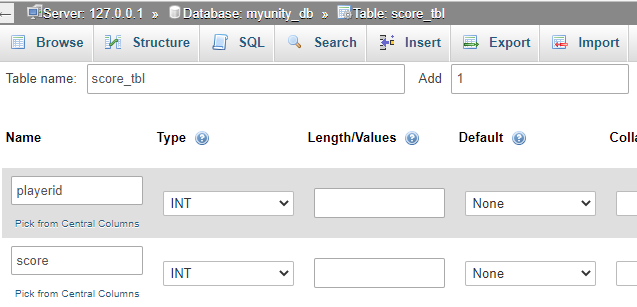
1. From XAMPP’s control panel, click on the Admin button of the MySQL. This will open a web browser which will open MySQL Admin page. From there you can create your database and view the data



1. Click on the Database tab and create a database called “myUnity\_db”. When the database is created, the new database will be displayed on the list of the databases on the left side of the MySQL admin
2. The next page to automatically appear after the database is created is the Create Table page. Here we will specify the table to score\_tbl with 2 columns



1. The columns will be “playerid” and “score” with the type of Integer for both



1. Apart from database, XAMPP also acts as a local webserver which mimic real commercial webserver. XAMPP is normally used for local development before deployment to the live server. Upon installing XAMPP, it will automatically create a htdocs folder inside the installation folder (which by default is: C:\xampp\htdocs). Inside the htdocs folder, you can create subfolder specific to the project in this case create a new folder called unityMultiplayerLeaderboard which we will use to store the php server-side script on.



1. Inside the folder unityMultiplayerLeaderboard, lets create a php file called leaderboard.php. This php script will handle the transaction between Unity and the Database. We will fill the code in this script later.
2. Back to Unity, technically the leaderboard will only store and retrieve data after the game has ended. In this case just to make it easier, we will just capture a button press (in this case button “X” which will end the game, store the data to database and then retrieve and display the data in the leaderboards). So, lets create a script called *saveDB.*cs inside a new empty object and make sure to add a Network Object component to the object
   1. Since we are going to use UnityWebRquest we need to include:

using UnityEngine.Networking;

* 1. Adds a bool variable to the class to make sure the key capture only happen once

bool XIsPressed;

* 1. Create a Dictionary which consist of ulong and score. You would later can change this accordingly to match the data structure you use.

Dictionary<ulong, int> playersScoresDictionary = new Dictionary<ulong, int>();

* 1. Set default value of the bool variable and we add some values to the dictionary as an example

void Start()

{

XIsPressed = false;

playersScoresDictionary.Add(1, 15);

playersScoresDictionary.Add(2, 25);

}

* 1. In the Update method, check the input (in this case a button X) and make sure the code only runs once by changing the bool variable created earlier

if (Input.GetKey(KeyCode.X) && !XIsPressed)

{

//to make sure X is only pressed once

XIsPressed = true;

sendDataToServerScript();

}

* 1. On the sendDataToServerScript method, we are going to send the dictionary to the store it to the server. To do this first we will pack each ClientID with its score into WWWForm and then use WWW to send the data to server-script. We also need to capture the return text from the server script to indicate the status of the process

void sendDataToServerScript()

{

StartCoroutine(toDatabase());

}

IEnumerator toDatabase()

{

int xcount = 0;

foreach (KeyValuePair<ulong, int> entry in playersScoresDictionary)

{

WWWForm form = new WWWForm();

form.AddField("clientid", (int)entry.Key);

form.AddField("score", entry.Value);

UnityWebRequest www = UnityWebRequest.Post("http://localhost/unityMultiplayerLeaderboard/leaderboard.php", form);

yield return www.SendWebRequest();

if(www.result != UnityWebRequest.Result.Success){

Debug.Log(www.error);

} else

{

xcount++;

Debug.Log(www.downloadHandler.text);

}

www.Dispose();

}

}

* 1. Now we are going to fill in the php file leaderboard.php which will handle the data sent from Unity, perform a database and store the data to the database. The code in PHP Script should be wrap in **<?php** as the start tag and **?>** as the end tag

<?php

//set up connection to database

$con = mysqli\_connect("localhost", "root", "", "myunity\_db");

if(mysqli\_connect\_error()){

echo "Connection Failed";

exit();

}

//handle the data sent from Unity. Here the data passed on from Unity is handled through a POST request (note that the variable name inside the bracket has to be identical to the variable name you specified on Unity

$clientID = $\_POST["clientid"];

$score = $\_POST["score"];

//Create a SQL statement which then executed along with the connection. Check this out for reference and list of SQL statements: https://www.w3schools.com/php/php\_mysql\_insert.asp

$sqlQuery = "INSERT into score\_tbl (playerid, score) values (".$clientID.", ".$score.")";

$executeSQLSquery = mysqli\_query($con, $sqlQuery);

//close the connection

mysqli\_close($con);

?>

# Multiplayer SceneManagement

We are going to call a scene to show the data saved in the leaderboard.

1. Click on the NetworkManager object we created earlier and make sure the NetworkManager component has the config as followed:
   1. “Don’t Destroy“ is ticked
   2. Add the scene to transition to in the Registered Scene Names
   3. “Enable SceneManagement” is ticked
   4. “Allow Runtime Scene Changes” is ticked
2. After you create a new scene called “leaderboard”, add this code to IEnumerator toDatabase. This code will call a serverrpc since networkscenemanagement can only be executed in the server:

if (xcount == playersScoresDictionary.Count)

{

changeSceneServerRpc();

}

1. The serverrpc:

[ServerRpc(RequireOwnership = false)]

public void changeSceneServerRpc()

{

var status = NetworkManager.SceneManager.LoadScene("leaderboard", LoadSceneMode.Single);

if (status != SceneEventProgressStatus.Started)

{

Debug.LogWarning($"Failed to load {"leaderboard"} " +

$"with a {nameof(SceneEventProgressStatus)}: {status}");

}

}

1. To allow RPC, add the lines below to the top of the code:

using Unity.Netcode;

using UnityEngine.SceneManagement;

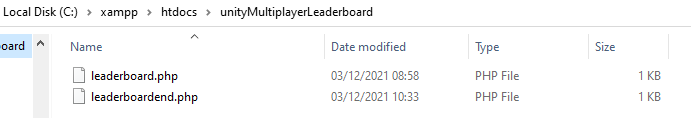
1. Instead of inherit from Monobehaviour, the class should be inherited from NetworkBehaviour

public class saveDB : NetworkBehaviour

# Show leaderboards based on data from database

Now we are going to show the information retrieved in the database to the player. The functionalities will be the same which again Unity will send a request to get the data and PHP will bridge the request to database.

1. Lets create a PHP file called leaderboardend.php in the same folder as the php script we created earlier



1. The PHP code for leaderboardend.php script to retrieve the data from database:

<?php

//setup connection to database

$con = mysqli\_connect("localhost", "root", "", "myunity\_db");

if(mysqli\_connect\_error()){

echo "Connection Failed";

exit();

}

//the SQL statement to retrieve all data from database and execute it via the connection

$sqlQuery = "select \* from score\_tbl";

$result = mysqli\_query($con, $sqlQuery);

//retrieve all data from database. Note here that we only need to “echo” it which basically in PHP means to print it to screen and Unity C# will be able to get what are being printed by PHP

if (mysqli\_num\_rows($result) > 0) {

// output data of each row. Note here that we are going to use “&” to split the record / row retrieved from the database

while($row = mysqli\_fetch\_assoc($result)) {

echo "Player ID : " . $row["playerid"]. " - Score: " . $row["score"] . "&";

}

} else {

echo "0 results";

}

//close the connection

mysqli\_close($con);

?>

1. In Unity, create an empty gameobject on *leaderboard* scene and create a C# script (lets call it: calldatafromleaderboard) to interact with the PHP script
2. We then need to capture what text being “echoed” by PHP

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.Networking;

public class calldatafromleaderboard : MonoBehaviour

{

string[] theResultString;

bool getDataSuccess;

IEnumerator Start()

{

//the delimiter basically to parse the data which in this case we are going to use ‘&’ to distinguish new row / data record

char[] delimiterChars = {'&'};

//connect to the php script

UnityWebRequest www = UnityWebRequest.Get("http://localhost/unityMultiplayerLeaderboard/leaderboardend.php");

yield return www.SendWebRequest();

//check if the result is correct. If the result is back, split / parse the information by the delimiter

if (www.result != UnityWebRequest.Result.Success)

{

Debug.Log(www.error);

getDataSuccess = false;

}

else

{

theResultString = www.downloadHandler.text.Split(delimiterChars);

getDataSuccess = true;

}

www.Dispose();

}

void OnGUI()

{

//show the parsed information to screen

if (getDataSuccess) {

for(int i=0;i< theResultString.Length-1;i++)

{

GUI.Label(new Rect(10, 60 + (15 \* i+1), 300, 20), theResultString[i]);

}

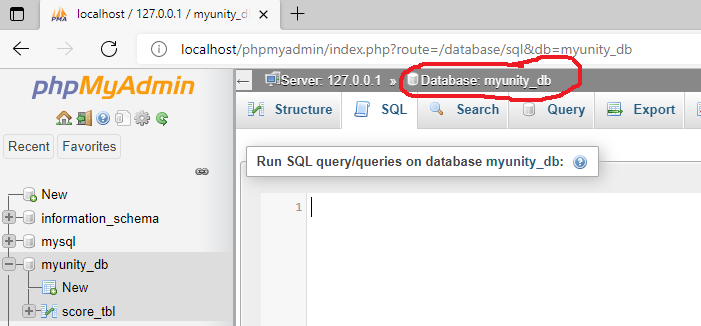
}

}

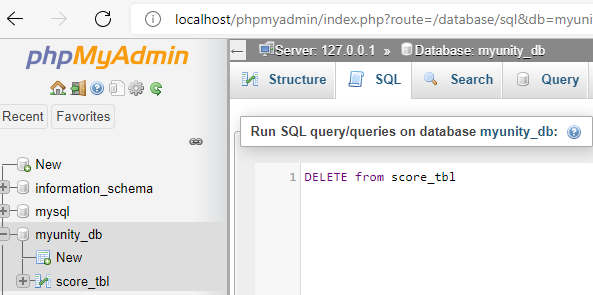
}

NOTE:

* You can also serialise the dictionary into json before passing it to the php server script
* The name of the selected database can be found at the top (if there is no database indicated, that means there is no database selected hence no operations will be available)



* You can directly try out SQL statement to the database directly. To do so: click on the database containing the table and data you want to manipulate -> click on the “SQL” tab. In this screenshot, the SQL statement will delete all data in table



Task:

* Use UPDATE SQL statement to update the score IF the userID has indeed exist in the database