

УДК 004.9

## ИСПОЛЬЗОВАНИЕ MYSQL ДЛЯ ХРАНЕНИЯ ДАННЫХ ИГР НА БАЗЕ UNITY

Сиводедова М.В., студент гр. ПИБ-202, II курс  
Научный руководитель: Киреева К.А., ассистент  
Кузбасский государственный технический университет  
имени Т.Ф. Горбачева  
г. Кемерово

При создании игры на любой платформе может возникнуть необходимость хранить данные. Это может быть счет, привязанный к аккаунту игрока, статистика, позиции игровых объектов на момент сохранения.

В данной статье будет приведен пример сохранения счета игрока в базе данных с использованием MySQL для игры, разработанной на движке Unity.

### Создание базы данных

База данных будет создаваться с помощью phpMyAdmin. Для примера будет использоваться таблица, в которой будут содержаться данные об имени игрока и его счете.

#	Имя	Тип	Сравнение	Атрибуты	Null	По умолчанию	Комментарии	Дополнительно	Действие
<input type="checkbox"/> 1	<b>Id</b>	int			Нет	Нет		AUTO_INCREMENT	
<input type="checkbox"/> 2	<b>Name</b>	varchar(30)	utf8mb4_0900_ai_ci		Нет	Нет			
<input type="checkbox"/> 3	<b>Score</b>	int			Нет	Нет			

☐ Отметить все
 С отмеченными:

Рисунок 1 – Таблица Players

На рисунке 1 представлена структура таблицы Players, которая будет использоваться для авторизации игрока.

### Серверные скрипты

Работу с данными можно проводить напрямую: Unity-MySQL-Unity. Но такой подход имеет несколько недостатков, главный из которых – если игра будет находиться в сети, то третье лицо сможет получить доступ к базе данных. Разработчикам понадобится два PHP-скрипта для того, чтобы это предотвратить. Первый скрипт представляет собой подключение к базе данных (рис. 2).

```
$hostname = 'localhost';
$username = 'root';
$password = '';
try
{
    $dbh = new PDO('mysql:host=' . $hostname . ';dbname=' . $database,
        $username, $password);
}
catch(PDOException $e)
{
    echo '<h1>An error has occurred.</h1><pre>', $e->getMessage()
        , '</pre>';
}
}
```

Рисунок 2 – Скрипт подключения к базе данных

Файл с этим кодом нужно назвать display.php и поместить в папку сервера, который будет использоваться для обработки запросов.

Второй PHP-скрипт (рис. 3) будет называться addscore.php. Данный скрипт будет отвечать за добавление данных в базу. Значение секретного ключа – фактически пароль, который будет использоваться для проверки полученных данных.

```
$hostname = 'localhost';
$username = 'root';
$password = '';
$database = 'highscores';
$secretKey = "mySecretKey";

try
{
    $dbh = new PDO('mysql:host=' . $hostname . ';dbname=' . $database,
        $username, $password);
}
catch(PDOException $e)
{
    echo '<h1>An error has occurred.</h1><pre>', $e->getMessage()
        , '</pre>';
}

$hash = $_GET['hash'];
$realHash = hash('sha256', $_GET['name'] . $_GET['score'] . $secretKey);

if($realHash == $hash)
{
    $sth = $dbh->prepare('INSERT INTO scores VALUES (null, :name
        , :score)');
    try
    {
        $sth->bindParam(':name', $_GET['name'],
            PDO::PARAM_STR);
        $sth->bindParam(':score', $_GET['score'],
            PDO::PARAM_INT);
        $sth->execute();
    }
    catch(Exception $e)
    {
        echo '<h1>An error has occurred.</h1><pre>',
            $e->getMessage() , '</pre>';
    }
}
}
```

Рисунок 3 – Добавление данных

## Создание сцен в Unity

Для примера будет достаточно двух сцен: сцена регистрации и сцена, на которой будет выводиться имя и счет игрока.

В меню будет достаточно поля ввода и кнопки. Найти их можно в меню добавления нового игрового объекта в иерархию (рис. 4).

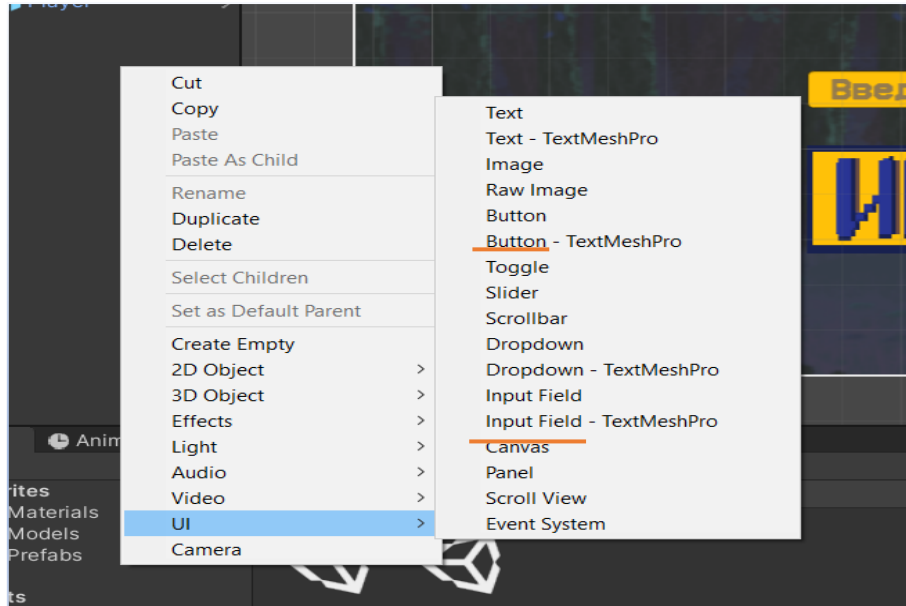


Рисунок 4 – Добавление поля ввода и кнопки

На рисунке 5 представлен пример сцены регистрации.

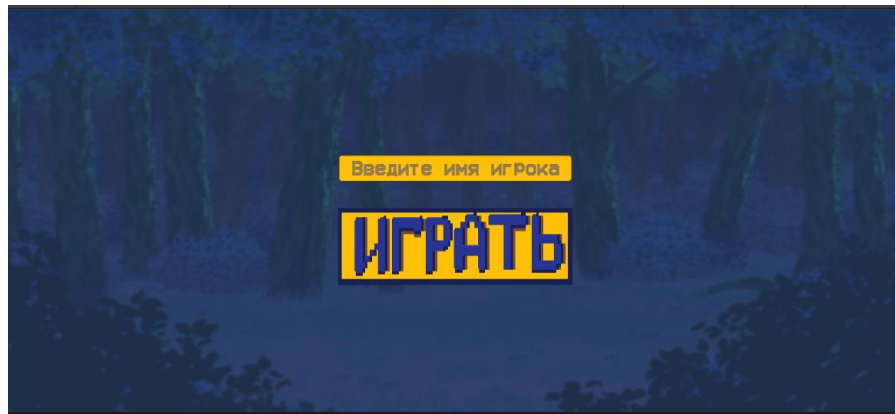


Рисунок 5 – Пример сцены регистрации

## Скрипт для записи и получения данных из Unity

Следующий скрипт отвечает, как за отправку, так и за извлечение данных. Получение счета реализуется при помощи этих двух действий, поскольку оно включает в себя создание UnityWebRequest для отправки HTTP-запроса. На этом этапе серверный скрипт будет выполнен и отправит данные обратно в Unity для отображения в пользовательском интерфейсе. Добавление нового счета выполняется по аналогичной схеме, но требует, использования метода POST для отправки данных в addscore.php. Прежде чем сделать это, вводимые пользователем данные также хэшируются с использованием алгоритма SHA256 в качестве базовой меры безопасности. Для повышения

безопасности потребуется сделать схему сложнее, чем просто хэширование значений, чтобы защитить их. Но в данной статье представлен более лаконичный пример, потому что цель – показать механику подключения базы данных к Unity приложению.

Как только значения будут отправлены через HTTP-запрос, addscore обработает процесс добавления оценки в базу данных.

Для работы скрипта нужно добавить директивы using в C# – код (рис. 6).

```
using System;  
using System.Collections;  
using System.Security.Cryptography;  
using System.Text.RegularExpressions;  
using UnityEngine;  
using UnityEngine.Networking;  
using UnityEngine.UI;
```

Рисунок 6 – перечень необходимых using

Затем понадобятся переменные, которые будут использоваться для поиска игрока в базе данных и вывода его имени и счета в сцене игры (рис. 7).

```
private string secretKey = "mySecretKey";  
public string addScoreURL =  
    "http://localhost/ScoreGame/addscore.php?";  
public string highscoreURL =  
    "http://localhost/ScoreGame/display.php";  
public Text nameTextInput;  
public Text nameResultText;  
public Text scoreResultText;
```

Рисунок 7 – переменные C# – скрипта

Значение ключа безопасности должно совпадать со значением из addscore.php, иначе данные не будут записаны в базу.

Далее следуют два метода: один для получения баллов, другой для их отправки (рис. 8). Это будут методы, привязанные к кнопкам в пользовательском интерфейсе: в сцене регистрации и сцене самой игры.

```
public void GetScoreBtn()  
{  
    nameResultText.text = "Player: \n \n";  
    StartCoroutine(GetScores());  
}  
public void SendScoreBtn()  
{  
    StartCoroutine(PostScores(nameTextInput.text,  
        Convert.ToInt32(scoreTextInput.text)));  
    nameTextInput.gameObject.transform.parent.GetComponent<InputField>().text = "";  
}
```

Рисунок 8 – Метод поиска игрока

Далее нужно создать сопрограмму GetScores, как следует из названия, будет отвечать за связь с базой данных MySQL, чтобы получить результат, который в настоящее время хранится в базе данных.

## Завершающая настройка в Unity

А теперь нужно завершить настройку в Unity. Для этого сначала необходимо добавить скрипт HighScoreControl к Main Camera (рис. 9).

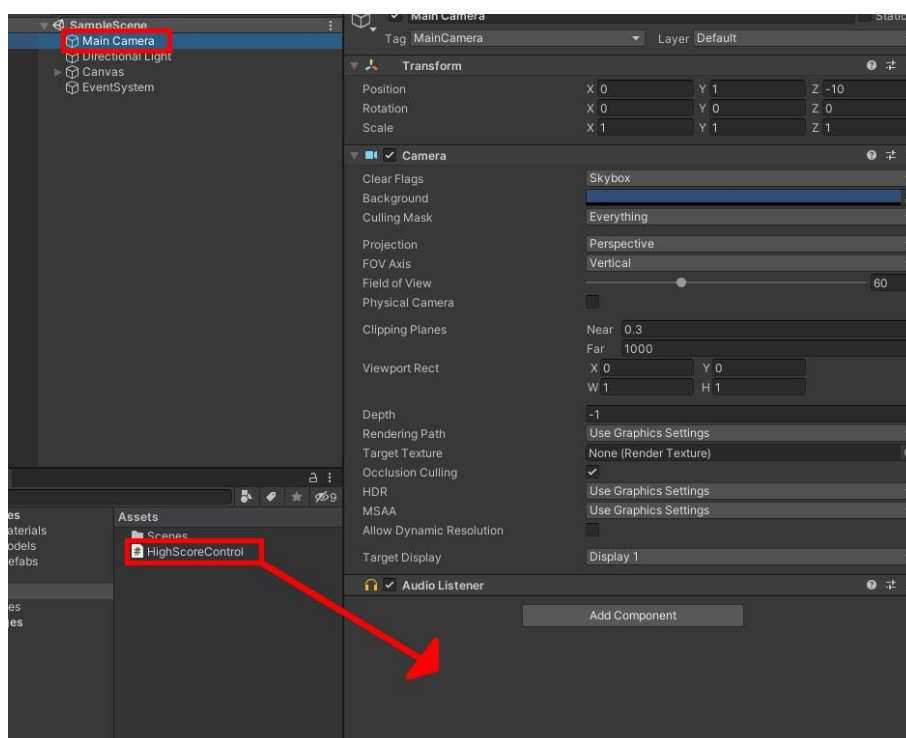


Рисунок 9 – Добавление скрипта HighScoreControl

Затем в иерархии нужно создать игровые объекты в соответствии со скриптом и связать их со скриптом, перетаскив их (рис. 10) [1].

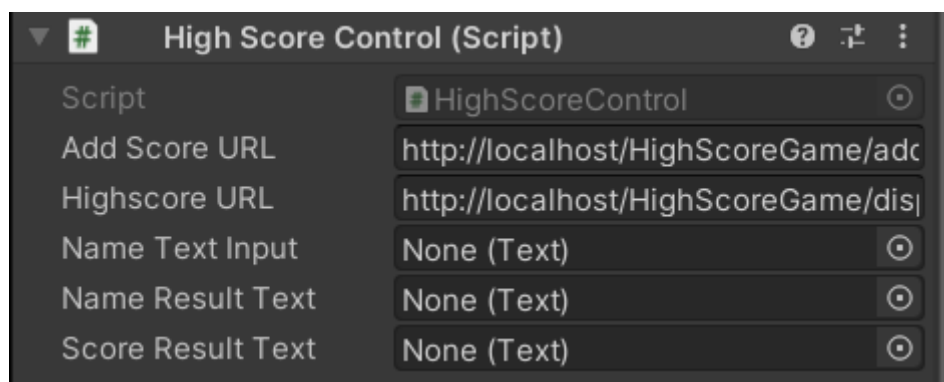


Рисунок 10 – Поля для связывания игровых объектов с переменными скрипта

Далее следует создать новое событие для кнопки, переместить туда игровой объект Main Camera и выбрать необходимый метод из скрипта в контекстном меню (рис. 11) [1].

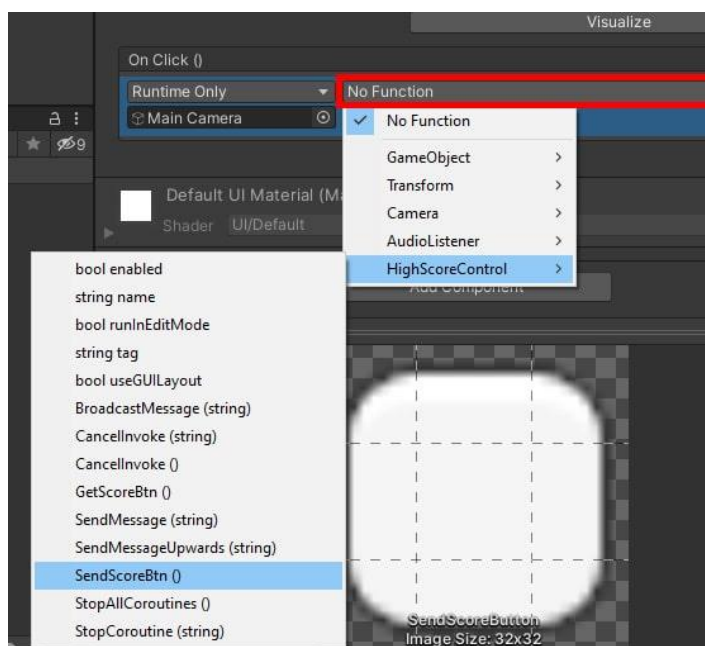


Рисунок 11 – Добавление события на нажатие кнопки

### Список литературы:

1. Джозеф Хокинг. Unity в действии. Мультиплатформенная разработка на Си Шарп / Пер. с англ. – М.: Издательство «Питер СПб», 2019. –352 с.