**Unity3D教程：保存或读取数据组的方法**

Posted on 2013年02月05日 by U3d / [Unity3D 基础教程](http://www.unitymanual.com/category/manual/unity3d-%e5%9f%ba%e7%a1%80%e6%95%99%e7%a8%8b)/被围观 195 次

[**Unity**](http://www.unitymanual.com/)本身有[**PlayerPrefs**](http://www.unitymanual.com/category/manual)来做一些数据的保存和读取,也可以通过循环来做批量的读取或者保存,下面这个脚本可以方便的调用用来做上面批量的工作,比如读取一组文本数组数据和保存一组文本数组数据。[**Unity3D教程：Unity3D实用小功能**](http://www.unitymanual.com/1318.html)

建议把这个脚本放在Standard Assets目录下,这样可以按照下面的方法方便的调用它.现在包含了下面这些命令:

PlayerPrefsX.SetVector3

PlayerPrefsX.GetVector3

PlayerPrefsX.SetIntArray

PlayerPrefsX.GetIntArray

PlayerPrefsX.SetFloatArray

PlayerPrefsX.GetFloatArray

PlayerPrefsX.SetStringArray

PlayerPrefsX.GetStringArray

**保存一个向量**：

static function SetVector3 (key : string, value : Vector3) : boolean

//尝试保存一个物体位置

var player : GameObject;

if (!PlayerPrefsX.SetVector3("PlayerPosition", player.transform.position))

print("不能保存物体位置!");

成功返回真,否则假(例如用Webplayer保存超过1M数据的时候)。

**获得一个向量：**

var player : GameObject;

player.transform.position = PlayerPrefsX.GetVector3("PlayerPosition");

如果读取的向量存在的话将会返回这个向量值。

**保存一组整型数据：**

//当保存Scores命名的分数时候创建一个10成员数组

var myScores = new int[10];

for (i = 0; i < myScores.Length; i++)

myScores = i+1;

if (!PlayerPrefsX.SetIntArray("Scores", myScores))

print("不能保存分数!");

**获得一组整型数据**：

static function GetIntArray (key : string) : int[]

如果存在将返回这组数据,否则将返回int[0];

var scores = PlayerPrefsX.GetIntArray("Scores");

static function GetIntArray (key : string, defaultValue : int, defaultSize : int) : int[]

如果不存在这组数据,将返回指定长度的数组以及每个成员都会赋予默认值.

**其他函数的使用方法:**

static function SetFloatArray (key : string, value : float[]) : boolean

static function GetFloatArray (key : string) : float[]

static function GetFloatArray (key : string, defaultValue : float, defaultSize : int) : float[]

static function SetStringArray (key : string, value : String[]) : boolean

static function SetStringArray (key : string, value : String[], separator : char) : boolean

static function GetStringArray (key : string) : string[]

static function GetStringArray (key : string, separator : char) : string[]

static function GetStringArray (key : string, defaultValue : String, defaultSize : int) : string[]

static function GetStringArray (key : string, separator : char, defaultValue : String, defaultSize : int) : string[]

**该脚本的Javascript版:**

// Site of this script: <a href="http://www.unifycommunity.com/wiki/index.php?title=ArrayPrefs" target="\_blank" rel="external">http://www.unifycommunity.com/wiki/index.php?title=ArrayPrefs</a>  
// Created by: Eric Haines (Eric5h5)  
// Contribution (Set/Get Vector3) 03/2010: Mario Madureira Fontes (fontmaster)  
static function SetVector3 (key : String, vector : Vector3) : boolean {  
return SetFloatArray(key, [vector.x, vector.y, vector.z]);  
}  
static function GetVector3 (key : String) : Vector3 {  
var floatArray = GetFloatArray(key);  
if (floatArray.Length < 3) {  
return Vector3.zero;  
}  
return Vector3(floatArray[0], floatArray[1], floatArray[2]);  
}  
static function SetIntArray (key : String, intArray : int[]) : boolean {  
if (intArray.Length == 0) return false;  
var sb = new System.Text.StringBuilder();  
for (i = 0; i < intArray.Length-1; i++) {  
sb.Append(intArray[i]).Append("|");  
}  
sb.Append(intArray[i]);  
try {  
PlayerPrefs.SetString(key, sb.ToString());  
}  
catch (err) {  
return false;  
}  
return true;  
}  
static function GetIntArray (key : String) : int[] {  
if (PlayerPrefs.HasKey(key)) {  
var stringArray = PlayerPrefs.GetString(key).Split("|"[0]);  
var intArray = new int[stringArray.Length];  
for (i = 0; i < stringArray.Length; i++) {  
intArray[i] = parseInt(stringArray[i]);  
}  
return intArray;  
}  
return new int[0];  
}  
static function GetIntArray (key : String, defaultValue : int, defaultSize : int) : int[] {  
if (PlayerPrefs.HasKey(key)) {  
return GetIntArray(key);  
}  
var intArray = new int[defaultSize];  
for (i = 0; i < defaultSize; i++) {  
intArray[i] = defaultValue;  
}  
return intArray;  
}  
static function SetFloatArray (key : String, floatArray : float[]) : boolean {  
if (floatArray.Length == 0) return false;  
var sb = new System.Text.StringBuilder();  
for (i = 0; i < floatArray.Length-1; i++) {  
sb.Append(floatArray[i]).Append("|");  
}  
sb.Append(floatArray[i]);  
try {  
PlayerPrefs.SetString(key, sb.ToString());  
}  
catch (err) {  
return false;  
}  
return true;  
}  
static function GetFloatArray (key : String) : float[] {  
if (PlayerPrefs.HasKey(key)) {  
var stringArray = PlayerPrefs.GetString(key).Split("|"[0]);  
var floatArray = new float[stringArray.Length];  
for (i = 0; i < stringArray.Length; i++) {  
floatArray[i] = parseFloat(stringArray[i]);  
}  
return floatArray;  
}  
return new float[0];  
}  
static function GetFloatArray (key : String, defaultValue : float, defaultSize : int) : float[] {  
if (PlayerPrefs.HasKey(key)) {  
return GetFloatArray(key);  
}  
var floatArray = new float[defaultSize];  
for (i = 0; i < defaultSize; i++) {  
floatArray[i] = defaultValue;  
}  
return floatArray;  
}  
static function SetStringArray (key : String, stringArray : String[], separator : char) : boolean {  
if (stringArray.Length == 0) return false;  
try {  
PlayerPrefs.SetString(key, String.Join(separator.ToString(), stringArray));  
}  
catch (err) {  
return false;  
}  
return true;  
}  
static function SetStringArray (key : String, stringArray : String[]) : boolean {  
if (!SetStringArray(key, stringArray, "\n"[0])) {  
return false;  
}  
return true;  
}  
static function GetStringArray (key : String, separator : char) : String[] {  
if (PlayerPrefs.HasKey(key)) {  
return PlayerPrefs.GetString(key).Split(separator);  
}  
return new String[0];  
}  
static function GetStringArray (key : String) : String[] {  
if (PlayerPrefs.HasKey(key)) {  
return PlayerPrefs.GetString(key).Split("\n"[0]);  
}  
return new String[0];  
}  
static function GetStringArray (key : String, separator : char, defaultValue : String, defaultSize : int) : String[] {  
if (PlayerPrefs.HasKey(key)) {  
return PlayerPrefs.GetString(key).Split(separator);  
}  
var stringArray = new String[defaultSize];  
for (i = 0; i < defaultSize; i++) {  
stringArray[i] = defaultValue;  
}  
return stringArray;  
}  
static function GetStringArray (key : String, defaultValue : String, defaultSize : int) : String[] {  
return GetStringArray(key, "\n"[0], defaultValue, defaultSize);  
}

**该脚本的C#版：**

// Contribution (Created CSharp Version) 10/2010: Daniel P. Rossi (DR9885)  
// Contribution (Created Bool Array) 10/2010: Daniel P. Rossi (DR9885)  
// Contribution (Made functions public) 01/2011: Bren  
using UnityEngine;  
using System;  
public static class PlayerPrefsX  
{  
#region Vector 3  
/// <summary>  
/// Stores a Vector3 value into a Key  
/// </summary>  
public static bool SetVector3(string key, Vector3 vector)  
{  
return SetFloatArray(key, new float[3] { vector.x, vector.y, vector.z });  
}  
/// <summary>  
/// Finds a Vector3 value from a Key  
/// </summary>  
public static Vector3 GetVector3(string key)  
{  
float[] floatArray = GetFloatArray(key);  
if (floatArray.Length < 3)  
return Vector3.zero;  
return new Vector3(floatArray[0], floatArray[1], floatArray[2]);  
}  
#endregion  
#region Bool Array  
/// <summary>  
/// Stores a Bool Array or Multiple Parameters into a Key  
/// </summary>  
public static bool SetBoolArray(string key, params bool[] boolArray)  
{  
if (boolArray.Length == 0) return false;  
System.Text.StringBuilder sb = new System.Text.StringBuilder();  
for (int i = 0; i < boolArray.Length - 1; i++)  
sb.Append(boolArray[i]).Append("|");  
sb.Append(boolArray[boolArray.Length - 1]);  
try { PlayerPrefs.SetString(key, sb.ToString()); }  
catch (Exception e) { return false; }  
return true;  
}  
/// <summary>  
/// Returns a Bool Array from a Key  
/// </summary>  
public static bool[] GetBoolArray(string key)  
{  
if (PlayerPrefs.HasKey(key))  
{  
string[] stringArray = PlayerPrefs.GetString(key).Split("|"[0]);  
bool[] boolArray = new bool[stringArray.Length];  
for (int i = 0; i < stringArray.Length; i++)  
boolArray[i] = Convert.ToBoolean(stringArray[i]);  
return boolArray;  
}  
return new bool[0];  
}  
/// <summary>  
/// Returns a Bool Array from a Key  
/// Note: Uses default values to initialize if no key was found  
/// </summary>  
public static bool[] GetBoolArray(string key, bool defaultValue, int defaultSize)  
{  
if (PlayerPrefs.HasKey(key))  
return GetBoolArray(key);  
bool[] boolArray = new bool[defaultSize];  
for (int i = 0; i < defaultSize; i++)  
boolArray[i] = defaultValue;  
return boolArray;  
}  
#endregion  
#region Int Array  
/// <summary>  
/// Stores a Int Array or Multiple Parameters into a Key  
/// </summary>  
public static bool SetIntArray(string key, params int[] intArray)  
{  
if (intArray.Length == 0) return false;  
System.Text.StringBuilder sb = new System.Text.StringBuilder();  
for (int i = 0; i < intArray.Length - 1; i++)  
sb.Append(intArray[i]).Append("|");  
sb.Append(intArray[intArray.Length - 1]);  
try { PlayerPrefs.SetString(key, sb.ToString()); }  
catch (Exception e) { return false; }  
return true;  
}  
/// <summary>  
/// Returns a Int Array from a Key  
/// </summary>  
public static int[] GetIntArray(string key)  
{  
if (PlayerPrefs.HasKey(key))  
{  
string[] stringArray = PlayerPrefs.GetString(key).Split("|"[0]);  
int[] intArray = new int[stringArray.Length];  
for (int i = 0; i < stringArray.Length; i++)  
intArray[i] = Convert.ToInt32(stringArray[i]);  
return intArray;  
}  
return new int[0];  
}  
/// <summary>  
/// Returns a Int Array from a Key  
/// Note: Uses default values to initialize if no key was found  
/// </summary>  
public static int[] GetIntArray(string key, int defaultValue, int defaultSize)  
{  
if (PlayerPrefs.HasKey(key))  
return GetIntArray(key);  
int[] intArray = new int[defaultSize];  
for (int i = 0; i < defaultSize; i++)  
intArray[i] = defaultValue;  
return intArray;  
}  
#endregion  
#region Float Array  
/// <summary>  
/// Stores a Float Array or Multiple Parameters into a Key  
/// </summary>  
public static bool SetFloatArray(string key, params float[] floatArray)  
{  
if (floatArray.Length == 0) return false;  
System.Text.StringBuilder sb = new System.Text.StringBuilder();  
for (int i = 0; i < floatArray.Length - 1; i++)  
sb.Append(floatArray[i]).Append("|");  
sb.Append(floatArray[floatArray.Length - 1]);  
try  
{  
PlayerPrefs.SetString(key, sb.ToString());  
}  
catch (Exception e)  
{  
return false;  
}  
return true;  
}  
/// <summary>  
/// Returns a Float Array from a Key  
/// </summary>  
public static float[] GetFloatArray(string key)  
{  
if (PlayerPrefs.HasKey(key))  
{  
string[] stringArray = PlayerPrefs.GetString(key).Split("|"[0]);  
float[] floatArray = new float[stringArray.Length];  
for (int i = 0; i < stringArray.Length; i++)  
floatArray[i] = Convert.ToSingle(stringArray[i]);  
return floatArray;  
}  
return new float[0];  
}  
/// <summary>  
/// Returns a String Array from a Key  
/// Note: Uses default values to initialize if no key was found  
/// </summary>  
public static float[] GetFloatArray(string key, float defaultValue, int defaultSize)  
{  
if (PlayerPrefs.HasKey(key))  
return GetFloatArray(key);  
float[] floatArray = new float[defaultSize];  
for (int i = 0; i < defaultSize; i++)  
floatArray[i] = defaultValue;  
return floatArray;  
}  
#endregion  
#region String Array  
/// <summary>  
/// Stores a String Array or Multiple Parameters into a Key w/ specific char seperator  
/// </summary>  
public static bool SetStringArray(string key, char separator, params string[] stringArray)  
{  
if (stringArray.Length == 0) return false;  
try  
{ PlayerPrefs.SetString(key, String.Join(separator.ToString(), stringArray)); }  
catch (Exception e)  
{ return false; }  
return true;  
}  
/// <summary>  
/// Stores a Bool Array or Multiple Parameters into a Key  
/// </summary>  
public static bool SetStringArray(string key, params string[] stringArray)  
{  
if (!SetStringArray(key, "\n"[0], stringArray))  
return false;  
return true;  
}  
/// <summary>  
/// Returns a String Array from a key & char seperator  
/// </summary>  
public static string[] GetStringArray(string key, char separator)  
{  
if (PlayerPrefs.HasKey(key))  
return PlayerPrefs.GetString(key).Split(separator);  
return new string[0];  
}  
/// <summary>  
/// Returns a Bool Array from a key  
/// </summary>  
public static string[] GetStringArray(string key)  
{  
if (PlayerPrefs.HasKey(key))  
return PlayerPrefs.GetString(key).Split("\n"[0]);  
return new string[0];  
}  
/// <summary>  
/// Returns a String Array from a key & char seperator  
/// Note: Uses default values to initialize if no key was found  
/// </summary>  
public static string[] GetStringArray(string key, char separator, string defaultValue, int defaultSize)  
{  
if (PlayerPrefs.HasKey(key))  
return PlayerPrefs.GetString(key).Split(separator);  
string[] stringArray = new string[defaultSize];  
for (int i = 0; i < defaultSize; i++)  
stringArray[i] = defaultValue;  
return stringArray;  
}  
/// <summary>  
/// Returns a String Array from a key  
/// Note: Uses default values to initialize if no key was found  
/// </summary>  
public static String[] GetStringArray(string key, string defaultValue, int defaultSize)  
{  
return GetStringArray(key, "\n"[0], defaultValue, defaultSize);  
}  
#endregion  
}