DOI:10.13657/j.cnki.gxkxyxb.2013.01.011

广西科学院学报
Journal of Guangxi Academy of Sciences

2013,29(1):34～36
Vol. 29, No. 1 February 2013

Unity3D 虚拟现实平台的重放功能设计
Design of Replay Function of Virtual Reality based on Unity3D

刘孟全
LIU Meng-quan

(空军空降兵学院,广西桂林 541003)
(Air Force Airborne Academy, Guilin, Guangxi, 541003, China)

摘要:利用 C＃脚本和序列化技术,提出一种在 Unity3D 平台上实现重放功能的设计方法,给出该方法涉及到的部分关键代码。

关键词:Unity3D 序列化 虚拟现实
中图法分类号:TP391 **文献标识码**:A **文章编号**:1002-7378(2013)01-0034-03

Abstract: Using C＃ script and object serialization techniques, a design of replay function of virtual reality was conducted based on Unity3D platform and some key programming codes were analyzed.

Key words: Unity3D, serialization, virtual reality

近年来虚拟现实技术在计算机领域引起了广泛关注,由于该技术能够生成逼真的视、听、触觉等多重感应的虚拟环境,而且用户可以同虚拟环境中的物体进行交互,能够产生二维动画所不具备的现场沉浸感,所以在军事、信息技术等领域的应用也越来越广泛。Unity3D 是一种涉及到多种不同领域的跨平台的三维虚拟现实游戏平台,是一款易于使用的产品,用户界面简单,编辑器上手容易,支持多脚本的语言,并具有强大的物理引擎。与其它虚拟现实开发引擎及工具相比,Unity3D 在跨平台开发方面有一定优势。它在同一个编辑器中就可以把同一款游戏发布到不同平台,适宜于开发仿真训练系统。由于在利用虚拟现实技术进行仿真训练时,通常需要记录训练数据,重放训练过程,以便评判训练效果,调整动作,提高训练效率。因此,在 Unity3D 平台中研究重放功能对于仿真训练有一定帮助。

把实现重放功能的所有资源保存在一个 Unity3D 虚拟现实平台的资源包中,资源导入后在工程资源列表中生成一个目录,其中包含预定义的游戏对象 ReplayManager 和所需的脚本。使用时只需将预定义的 ReplayManager 拖放到场景中,并将场景中关键的游戏对象拖放到 ReplayManager 记录游戏对象的变量中,启动游戏,点击 ReplayManage 的开始按钮即可录制动画,实现重放功能。Unity 3D 虚拟现实平台重放功能包括两个方面的内容:一是根据采样间隔时间对游戏状态进行采样,保存采样数据;二是根据采样数据重放动画。本文以 C＃为脚本语言,采用模块化方法、设计 Unity-3D 平台重放功能,先设计一个脚本部件 ReplayManager.cs,并附加到游戏对象 ReplayManager,在脚本中实现交互和定时采集等功能,再采用序列化技术保存采样数据,并通过反序列化实现动画重放。

1 脚本部件 ReplayManager 主要功能设计

1.1 交互功能设计

定义共有属性游戏对象列表 gameObjectsToRecord。该属性将暴露在检查器面板,用户可以将需要重放的游戏对象直接拖曳到此属性。

收稿日期:2012-12-15
修回日期:2012-12-25
作者简介:刘孟全(1968-),男,副教授,主要从事网络安全与应用研究。

```
public List<GameObject> gameObjectsToRecord = new List<GameObject>();
```

录制、停止和重放等按钮的实现在 OnGUI 事件中实现，主要代码如下：

```
void OnGUI (){
    if (GUI. Button (new Rect (158,72,40,23),playIcon)) {
        play ( speedSliderValue, true, false, false);}
    if (GUI. Button (new Rect(20,160,130,20),startRecordIcon)) {record (); }
    if (GUI. Button (new Rect (248,72,40,23),pauseIcon)) {pause();}
    if (GUI. Button (new Rect (293,72,40,23),stopIcon)) {stop();}
}
```

其中 speedSliderValue 代表播放速度。

1.2 单例模式设计

在动画运行过程中，ReplayManager 实例只能有一个，采用以下方式设计：

```
//定义私有静态变量
private static ReplayManager rm_Singleton;
public static ReplayManager singleton
  {
    get
    {
        if (rm _Singleton == null)
        {
            GameObject rm = GameObject. Find(" ReplayManager");
                Rm _Singleton = rm. GetComponent(typeof(ReplayManager)) as
                ReplayManager;
        }
    return rm_Singleton;
  }
}。
```

1.3 定时功能的设计

定时功能在记录和重放时都会用到，利用协程 StartCoroutine 和 yield return 实现。

```
private static IEnumerator waitForNewUpdate(float delay,float timeelapsed,int action) {
    yield return new WaitForSeconds(delay);
    其他操作
}。
```

2 采样数据保存

采样数据保存采用对象序列化技术。先将对象的公共字段和私有字段以及类的名称（包括类所在的程序集）转换为字节流，然后再把字节流写入数据流。在对对象进行反序列化时，将创建出与原对象完全相同的副本。序列化的目的主要有两点：一是以某种存储形式使自定义对象持久化；二是将对象从一个地方传递到另一个地方。由于采样数据涉及到每次采样时场景中指定游戏对象的状态，采用序列化技术是较好的选择。保存重放状态数据的类层次结构主要分为以下 4 种。

2.1 保存表示位置 position 的类 SerVector3 和表示旋转 rotation 的类 SerQuaternion

```
[Serializable()]
public class SerVector3: ISerializable {
  public float x;
  public float y;
  public float z;
  方法省略…
}
[Serializable()]
public class SerQuaternion: ISerializable {
  public float x;
  public float y;
  public float z;
  public float w;
  方法省略…
}。
```

2.2 保存单个游戏对象某次采样状态的类 SavedState

```
[Serializable()]
public class SavedState : ISerializable {
  public SerVector3 position;
  public SerVector3 localPosition;
  public SerQuaternion rotation;
  public SerQuaternion localRotation;
  方法省略…
}。
```

2.3 保存单个游戏对象全部状态的类 Object2PropertiesMapping

由于每次采样都对应动画的某一帧，游戏对象的全部状态可以采用 Dictionary 来实现记录，其中

键值为帧号，值为 SavedState。泛型 Dictionary 类不支持 XML 序列化，需要先定义一个支持 XML 序列化的泛型 Dictionary 类：

```
[Serializable]
public class SerializableDictionary < TKey, TVal> : Dictionary<TKey, TVal>, IXmlSerializable, Iserializable
{内容省略}
```

Object2PropertiesMapping 类的定义为：

```
[Serializable()]
public class Object2PropertiesMapping : ISerializable{
    //保存属于某个游戏对象的状态
    public SerializableDictionary<int, SavedState> savedStates = new SerializableDictionary<int,SavedState>();
    //对应的游戏对象
    private GameObject gameObject;
    //对应游戏对象的克隆
    private GameObject gameObjectClone;
    //父游戏对象标志
    private bool isParentObj = false;
    //保存父游戏对象状态的 Object2PropertiesMapping
    private Object2PropertiesMapping parentMapping;
    //子游戏对象编号，如果是父对象则为 0
    public int childNo;
    其他省略
}。
```

2.4 保存所有要记录游戏对象的类 Object2PropertiesMappings

```
[Serializable()]
public class Object2PropertiesMappingListWrapper : ISerializable {
    public List<Object2PropertiesMapping> object2PropertiesMappings = new List < Object2PropertiesMapping>();
    //采样间隔时间
    public float recordingInterval;
    方法省略…
}。
```

通过定义以上类，所有重放时需要的状态信息都可通过 Object2PropertiesMappings 对象保存，将其序列化到数据流中，通过反序列化可重现游戏对象。例如，将其序列化到磁盘文件中，即实现记录数据保存到文件，可随时读取文件内容反序列化后实现重放。

3 动画重放设计

重放通过反序列化设计，使用的是游戏对象的克隆来完成。主要方法有 6 个：Object2PropertiesMapping 类中的 prepareObjectForReplay、synchronizeProperties 方法；脚本部件 ReplayManager 中的 loadFromFile，与数据保存共用的 execRecorderAction、updateRecording 和 waitForNewUpdate 方法。

以 Object2PropertiesMapping 类中的属性 gameObjectClone 为例阐述动画重放设计的步骤：

首先切换到重放模式，调用 prepareObjectForReplay 方法为 gameObjectClone 属性赋值，父游戏对象从预置对象生成克隆，子游戏对象的克隆则从对应的父游戏对象根据编号查找生成，之后调用 execRecorderAction()方法；

再在 execRecorderAction 方法中，调用 synchronizeProperties 方法，将游戏对象的状态值设置到 gameObjectClone，然后与保存数据过程相同，生成循环不断为 gameObjectClone 赋状态值，实现动画重放；

在文件回放时，定义一个序列化时用到的类 Object2PropertiesMappings 实例进行反序列化，其余操作与上两步相同。

4 结束语

目前，Unity3D 开发人员众多，在网页 3D 游戏方面表现尤为突出。由于该平台用于仿真训练时的重放功能可以提高训练效果，所以在军事、信息技术等领域的应用也越来越多。本文简述了如何利用序列化技术在 Unity3D 平台上设计动画重放，给出了部分关键代码。已有的一些应该表明，利用序列化技术在 Unity3D 虚拟现实平台实现重放功能，有较好的应用前景。

（责任编辑：尹 闯）