廣東工業大學

本科毕业设计（论文）

外文参考文献译文及原文

学　　院	计算机学院
专　　业	软件工程
年级班别	2013 级(1)班
学　　号	3113006215
学生姓名	陈永坤
指导教师	杨　卓

2017 年　5 月

目录

使用 Unity3D 创建内容......1

1 **介绍**......2

1.1 游戏引擎（Unity 3D）......2

1.2 绑定......3

1.3 游戏对象上的动画......3

1.4 环境搭建......4

1.5 脚本......4

1.6 添加声音......5

1.7 着色器......6

1.8 地形引擎......6

2 **总结**......7

致 谢......8

参 考 文 献......9

Content Creation Using Unity3D......10

1 Introduction......10

1.1 Game Engine (Unity 3D)......10

1.2 Rigging......11

1.3 Animations on Game Object......12

1.4 Environmental setup......12

1.5 Scripting......12

1.6 Adding Sounds......13

1.7 Shaders......14

1.8 Terrain Engine......14

2 Conclusion......14

Acknowledgement......14

References **15**

Unity 3D 游戏引擎摄像机动作帧编辑工具应用 **16**

1 **介绍** **17**

2 **相关研究** **18**

2.1 现有技术 18

2.2 先进技术 18

3 **系统配置** **19**

3.1 摄像机运动图像生成与校正 19

3.2 摄像机运动图像数据保存与变化 19

4 **总结** **20**

致 谢 **21**

参 考 文 献 **22**

Application of Camera Motion Frame Editor Tool for UNITY 3D Game Engine **23**

1 Introduction **23**

2 Related Researches **23**

2.1 The Existing technologies 23

2.2 The Advanced technologies 24

3. System Configuration **24**

3.1 Camera Motion Graph Data Creation & Correction 24

3.2 Camera Motion Graph Data saving & variation 24

4 Conclusion **24**

Acknowledgments. **25**

References **25**

使用Unity3D创建内容

Ms.Mugdha Kolte

Student, IT, MITCOE, Pune,koltemugdha@gmail.com

摘　　要

游戏在我们的日常生活中扮演着重要的角色，特别是电子游戏从它的出现至今已经走过了很长的道路。从计算机到智能手机，电子游戏可以设计用于不同类型的平台。在游戏开发、游戏设计、游戏AI引擎、教育游戏等领域，游戏产业的研究已经取得了很大的进展。游戏引擎工具如Game Maker、Unity 3D的出现，使得更容易设计和开发更加复杂的游戏，并缩短了开发时间。从技术上讲，游戏是被设计和开发成一种通过特定的目标来提供娱乐性的软件。与软件产品相似，交付一个成功的游戏也需要一个开发生命周期，即游戏开发生命周期（GDLC）。对于一个高质量的游戏来说，设计与开发两者之间需要有所平衡。设计计算机游戏需要有足够的经验和对细节的高度关注来描述规则，游戏的美学构成了交互体验。因此，除了开发之外，我们还会回顾游戏设计的概念，比如设计过程、游戏世界和以玩家为中心的方法。游戏引擎是一种为游戏创作和开发而设计的系统，它包括一个渲染引擎，一个用于碰撞检测的物理引擎，以及一个高效的内存管理系统。

关键词： 游戏引擎，绑定，脚本，着色处理，地形引擎

1 介绍

电子游戏通常在Windows平台上使用unity3D API执行，需要高性能的CPU和显卡。对于在家庭、酒店或网吧等各种环境中普遍存在的游戏来说，在移动设备和一定性能的CE设备上运行游戏是很有优势的，因为它们不需要在客厅里放置一个嘈杂的工作站或者在酒店的每个房间里安装昂贵的计算机或控制台。现在，由于消费者总是要求游戏中需要具备更加真实的外观和感觉，游戏的质量和游戏的真实性都在不断提升。这意味着需要改进渲染，突出内容，提供更可信的动画和更真实的人工智能行为。

Unity是一个功能丰富的、用于创建交互式3D内容的跨平台游戏引擎。它提供了直观的界面，允许开发人员进行浅层访问。由于其直观的界面、设计良好的体系结构并且易于资源复用，由其他内容创建者提供的不计其数的资源可以被快速复用与浸入式体验的开发。与传统开发相比，3D软件可以在很短的时间内得到开发。

1.1 **游戏引擎（Unity 3D）**

游戏引擎是一种为游戏开发而设计的软件框架，电子游戏开发人员使用它们来为电子游戏机、移动设备和个人电脑开发游戏。游戏引擎通常提供的核心功能包括用于2D或3D图形的渲染引擎、物理引擎或碰撞检测、声音、脚本、动画、人工智能、网络、流媒体、内存管理、线程管理、本地化支持和场景图像。

在这个游戏引擎中有不同的导出选项，每一种都用于不同的平台（例如网页端、PC和Mac、ios、安卓、xbox360和PS3），这使得为不同的平台或设备进行开发变得简单。Unity是一个由Unity Technologies公司研发的跨平台游戏制作系统，它包括了游戏引擎和集成开发环境（IDE）。Unity特别吸引了小型和中型的开发工作室，他们几乎不必投资昂贵的高端渲染引擎。此外，由于WYSIWYG（“所见即所得”）编辑器允许即时更改和实时编辑，原型和游戏开发的速度都非常快。Unity还提供了多个内置的着色器和效果，以及物理引擎和碰撞检测。

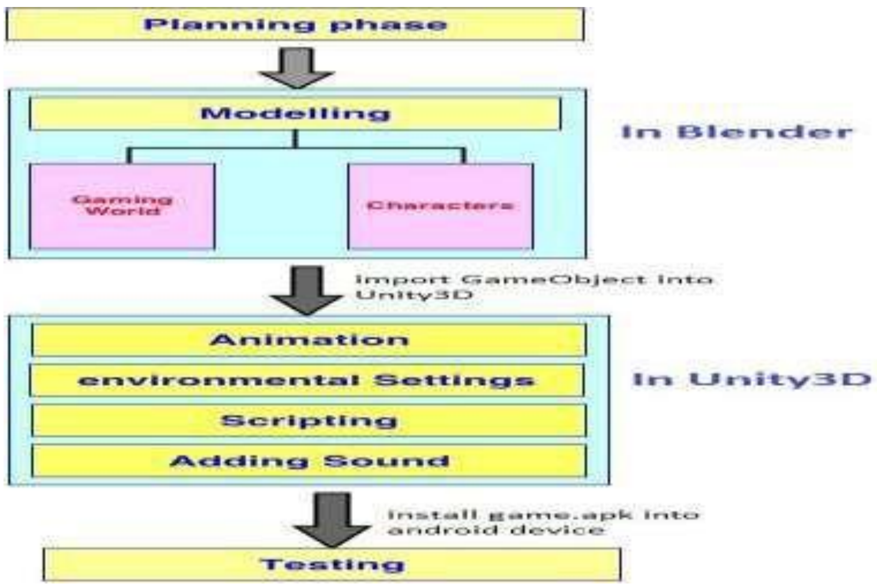

图1 游戏制作流程

1.2 绑定

骨胳动画是计算机动画中的一种技术，在计算机动画中，一个角色由两个部分来表示：用来绘制角色的外观（称为皮肤或网格），以及用于使网格动起来的相互连接的骨头（称为骨架或平台）的层次结构。这种技术被用来构造一系列的“骨骼”，有时也被称为绑定，绑定使我们的角色能够移动。绑定的过程是：先用数据建模并开始制作骨骼和肌肉，把皮肤和角色关联起来，再创造一套动画师用于推动和拉动身体的控制动画。每一根骨骼都有一个三维的变换属性（包括它的位置、大小和方向），以及一个可选的父骨骼。这样，这些骨骼便形成了一个层次结构。子节点的自身变换及其父变换形成了该节点的完整变换。因此，移动大腿骨头也会跟着移动小腿。而当角色在进行动作时，在动画控制器的影响下，骨头也会一直发生变换。

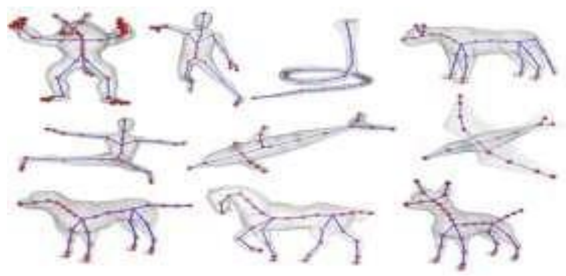

图2 绑定

1.3 游戏对象上的动画

将游戏对象导入 Unity 中后，动画便可完成。Unity 的动画系统是基于动画剪辑的概

念，它包含了一些关于特定对象如何随着时间改变位置、旋转或其他属性的信息。每个剪辑都可以被认为是单一的线性记录。来自外部源的动画剪辑是由艺术家或动画师使用了第三方工具如 Blender 或 Maya 创建的，或者是来自动作捕捉工作室或其他来源。动画片段将被组织成一个结构化的流程图，称为动画控制器。动画控制器充当了一个“状态机”，它可以跟踪当前播放的视频片段，以及动画何时应该改变或混合的情况。

Unity 的动画系统也包含很多特殊的功能用于处理人形角色，这让你有能力去重新定位来自任一来源的人形动画到你自己的角色模型，以及调整肌肉的定义。这些特殊的功能是由 Unity 的 Avatar 系统所支持的，在这个系统中，人形角色将被映射到一个通用的内部格式当中。

1.4 **环境搭建**

这一步涉及到设置或整个环境，包括世界、人物、玩家的动作、相机的动作以及整个游戏的样子。你需要以这样的方式来筹划这些事情，使它看起来具有视觉吸引力和挑战性。

1.5 **脚本**

在所有的游戏中，脚本都是必不可少的要素。即使是最简单的游戏也需要脚本对玩家的输入作出反应，并安排游戏中的事件发生。除此之外，脚本还可以用来制作画面效果，控制对象的物理行为，甚至可以为游戏中的角色实现自定义的 AI 系统。

图3 脚本

脚本序列化

对“事物”的序列化是Unity的核心。许多特性都是建立在序列化系统之上：

• **检视窗口**。检视窗口不与C#的API进行通信便能确定其检视的属性的值是什么。它要求对象能够序列化自己，以便能够显示出序列化的数据。

• **预制体**。在内部，预制体是一个（或多个）游戏对象和组件的序列化数据流。预制体实例是对该实例的序列化数据进行的一系列修改的表。这个概念实际上只存在于编辑器时期。当Unity进行发布时，预制体的修改会被嵌入到一个正常的序列化流中；而当它被实例化时，实例化的游戏对象并不知道它们在编辑器中是一个预制体的。

• **实例化**。当你对一个预制体、场景中的游戏对象，或其他任何对象（所有继承自UnityEngine.Object的对象都能被序列化）调用instantiate()方法时，对象被序列化，然后创建一个新的对象，以及数据被“反序列化”到新的对象上。（再次以不同的变体执行相同的序列化代码，用来汇报其他正在被引用的UnityEngine.Object对象，并检查所有被引用的UnityEngine.Object对象是否是被实例化的数据的一部分。如果引用指向的是“外部”对象（比如纹理），则保留那个引用，如果它指向的是“内部”对象（例如子游戏对象），则将引用指向对应的副本）。

• **保存**。如果你打开一个Unity场景文件与文本编辑器，并且设置了Unity“强制文本序列化”，则会在后台运行一个yaml序列化器。

• **加载**。这看起来并不奇怪，但是向后兼容加载是在序列化的基础上构建的一个系统。不仅yaml加载使用了序列化系统，场景、资源和assetbundle的运行时加载也使用了序列化系统。

• **编辑器代码热重载**。当你修改一个编辑器脚本时，将对所有编辑器窗口进行序列化(它们继承自UnityEngine.Object！)，然后销毁掉所有的窗口界面。接着卸载掉旧的C#代码，加载新的C#代码，并重新创建窗口，最后将数据流反序列化到新的窗口当中。

• **Resource.GarbageCollectSharedAssets()**。这是本地垃圾收集器，但与C#的垃圾回收器是不同的。它是在加载场景后运行的，用于找出之前的场景中哪些东西不再被引用，这样我们就可以卸载它们。本地垃圾收集器会在用于汇告所有外部UnityEngine.Object对象引用的变体中运行序列化器。

1.6 添加声音

音效被用来增强整体游戏体验，增加游戏的刺激感和时间感。音效在 Unity3D 中被当作游戏对象看待，其中包括了音频监听器和音频源组件。音频监听器是作为播放器组件添加的，而音频源是对声音的支持。

使用音频资源：

Unity可以导入AIFF、WAV、MP3或OGG格式的音频文件，如同其他资源一样，只需将文件简单拖放到项目面板中即可。导入一个音频文件可以创建出一个音频片段，之后便可以将其拖到音频源或在脚本中使用。

1.7 着色器

一个着色器基本上定义了游戏内着色外观效果的公式。在任一给定的着色器当中，都有着许多属性（通常是纹理属性）。着色器是通过直接依附在单个的游戏对象上的材质来实现的。在一个材质中，你将选择一个着色器，然后定义着色器所使用的属性（通常是材质和颜色，但是属性可以不同）。着色器可以很灵活地计算显卡的渲染效果。通过使用在着色器中定义的算法，可以动态地改变所有像素、饱和度、亮度、对比度，以及用于形成最终图像的纹理，并且可以通过程序调用着色器所引入的外部变量或纹理进行修改。

1.8 地形引擎

Unity 的地形系统允许你为游戏添加大量的地形。在运行时，地形渲染的效率得到了高度的优化；而在编辑器中，有一些可供选择的工具可使创建地形变得简单快捷。

2 总结

随着游戏引擎技术的成熟及变得对用户更加友好，游戏引擎的应用范围也在扩大。它们现在被用于更严肃的游戏：可视化、训练、医疗和军事模拟应用。为了方便这种可访问性，新的硬件平台现在正被游戏引擎所盯准，包括手机（如 Android 手机、iPhone、Windows phone）和网页浏览器。更多的游戏引擎是建立在诸如 Java 和 C#/.NET 这样的高级语言之上的。游戏引擎是问题的解决方案，因为它是可扩展的，并且可以作为许多不同游戏的基础，而不需要进行重大修改。引擎为开发游戏应用程序提供了一个灵活的、可重用的软件平台，包含了所需的所有核心功能，即开即用，从而降低成本。

致　谢

我借此机会表达我对我的导师Dhanashree Wategaonkar的深深的感激和敬意。我非常感谢她的慷慨、专业能力和指导；我从她那里得到了她在这个研讨会和整个学习过程中收集的数据。如果没有她的支持和及时的指导，我的研讨会的完成将会成为一个牵强的梦想。在这方面，她不仅指导了我的主题，还教会了我正确的工作方式、技巧和表现手法。

我很高兴能向那些参与我论文的人表达我的感激之情。我由衷地感激我们的HOD教授（博士）、A.S.Hiwale（IT部门）和校长（博士）M.S.Nagmode的指导。在我完成报告时所使用的各种信息和来源都在我的报告中可找到。我已经尽了最大的努力使这篇论文不出错。但是，如果有的话，我会懊悔的。

参 考 文 献

[1] Mariebeth Aquino, Florian Grash¨aftl, Stephanie Kohl,"Content creation using Maya and Unity3D"

[2] A. Jurgelionis, P. Fechteler, P. Eisert, F. Bellotti, H. David, J. P. Laulajainen, R. Carmichael, V. Poulopoulos, , A. Laikari, P. Per¨al¨a, A. De Gloria, and C. Bouras , "Platform for Distributed 3D Gaming", Volume 2009, Article ID 231863, 15 pages, doi:10.1155/2009/231863

[3] Bharti Verma, Anjali Vijayvargiya, Dr. Neeta Nain, "Development approach of 3D PACMAN for android platform", Volume 4, Issue 11, Year Nov 2014, ISSN 2277 128X

[4] Rita Costa Nascimento\ Ana Gil\ Teresa Heitor, Ana Tome, Helena Rua2,"Computer Game engine for Accessibility assessment", 978-1-4673-6165-1/13/$31.00 ? 2013 IEEE

[5] Unity 3D. Unity: Behind the scenes http://unity3d.com/support/documentation/Manual/ Behind%20the%20Scenes.html, 2014.

[6] https://www.google.com/patents/US6428413dq=game+engine&hl=en&sa=X&ei=q2vvVP3iJIq3uQThnIH4CA&sqi=2&pjf=1&ved=0CDkQ6AEwBA

[7] https://www.google.com/patents/CN102880464A?cl=en&dq=game+engine+for+windows&hl=en&sa=X&ei=5m3vVIGZF8KIuASc4YDYBQ&sqi=2&pjf=1&ved=0CB0Q6AEwAA.

Content Creation Using Unity3D

Ms.Mugdha Kolte

Student, IT, MITCOE, Pune,koltemugdha@gmail.com

ABSTRACT

Games play an important role in our daily life especially video games has come a long way since its origin. Video games could be designed for different types of platforms ranging from computer to smart phones. A tremendous research has been going in the gaming industry mainly in the area s like game development, game designing, game AI engines, educational games. The emergence of game engine tools like Game Maker, Unity 3D made life easier to design and develop more sophisticated games with the reduced amount of time. Technically game is software that is designed and developed with a certain characteristic goal to provide entertainment. Similar to software product there is a development life cycle for delivering a successful game namely game development life cycle (GDLC). For a good quality game both development and design has to be balanced. Designing computer games requires adequate experience and great attention to detail to describe the rules, play aesthetics that compose the interactive experience. So, in addition to the development we also review designing a game and game design concepts like stages of design process, game world, and player-centric approach. A game engine is a system designed for the creation and development of video games. It includes a rendering engine, a physics engine for collision detection, and an efficient memory management system.

Keywords*:* Game Engine, Rigging, Scripting, shading, Terrain Engine.

1 Introduction

Video games are typically executed on Windows platforms with unity3D API and require high performance CPUs and graphics hardware. For pervasive gaming in various environments like at home, hotels, or internet cafes, it is beneficial to run games also on mobile devices and modest performance CE devices avoiding the necessity of placing a noisy workstation in the living room or costly computers/consoles in each room of a hotel. Nowadays, the quality of graphics and realism of games are constantly increasing, since consumers are always demanding a more realistic look and feel in their games. This means that improvement of renderings, outstanding content, more believable animations and more authentic behaviour of artificial intelligence are needed.

Unity is feature rich multi-platform game engine for the creation of interactive 3D content. It includes an intuitive interface while at the same time allowing low level access for developers. Thousand s of assets provided by other content creators can be reused to quickly develop immersive experiences because of its intuitive interface, well designed architecture, and ability to easily reuse assets, 3D Software can be developed in a fraction of time compared to traditional development.

1.1 Game Engine (Unity 3D)

A game engine is a software framework designed for the creation and development of video games. Video game developers use them to create games for video game consoles, mobile devices and personal computers. The core functionality typically provided by a game engine includes a rendering engine for 2D or 3D graphics, a physics engine or collision detection, sound, scripting, animation, artificial intelligence, networking, streaming, memory management, threading, localization support, and a scene graph.

There are different export options in this game engine, each one dedicated to another platform (e.g.: Web Player, PC and Mac Standalone, iOS, Android, Xbox 360, PS3), which eases the development for different consoles or devices. Unity is a cross-platform game creation system developed by Unity Technologies, including a game engine and integrated development environment (IDE). Unity attracts especially small and middle-sized development studios who hardly invest in expensive high end rendering engines. Furthermore prototyping and game development is very quick due to the WYSIWYG (”what you see is what you get”) editor which allows instant changes and live editing. In addition, Unity provides multiple built-in shaders and effects as well as physics engine and collision detection.

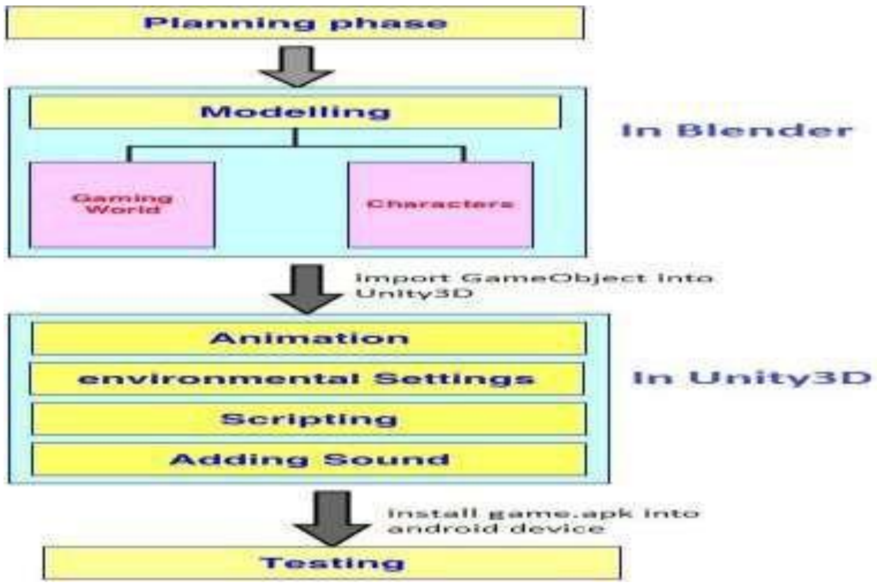

Fig-1: Game Production Pipeline

1.2 Rigging

Skeletal animation is a technique in computer animation in which a character is represented in two parts: a surface representation used to draw the character (called skin or mesh) and a hierarchical set of interconnected bones (called the skeleton or rig) used to animate the mesh. This technique is used by constructing a series of 'bones,' sometimes referred to as rigging. Rigging is making our characters able to move. The process of rigging is we take that digital sculpture and we start building the skeleton, the muscles, and we attach the skin to the character, and we also create a set of animation controls, which our animators use to push and pull the body around. Each bone has a three-dimensional transformation (which includes its position, scale and orientation), and an optional parent bone. The bones therefore form a hierarchy. The full transform of a child node is the product of its parent transform and its own transform. So moving a thigh-bone will move the lower leg too. As the character is animated, the bones change their transformation over time, under the influence of some animation controller.

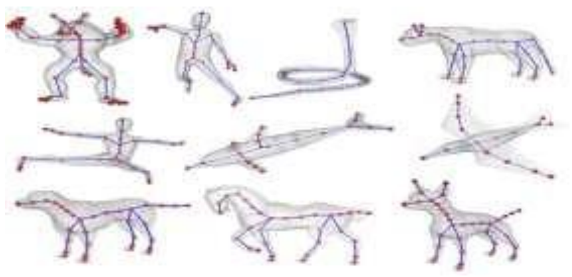

Fig- 2: Rigging

1.3 Animations on Game Object

After importing Game Object into unity animation can be done. Unity’s animation system is based on the concept of Animation Clips, which contain information about how certain objects should change their position, rotation, or other properties over time. Each clip can be thought of as a single linear recording. Animation clips from external sources are created by artists or animators with 3rd party tools such as Blender or Maya, or come from motion capture studios or other sources. Animation Clips are then organized into a structured flowchart-like system called an Animator Controller. The Animator Controller acts as a “State Machine” which keeps track of which clip should currently be playing, and when the animations should change or blend together.

Unity’s Animation system also has numerous special features for handling humanoid characters which give you the ability to retarget humanoid animation from any source to your own character model, as well as adjusting muscle definitions. These special features are enabled by Unity’s Avatar system, where humanoid characters are mapped to a common internal format.

1.4 Environmental setup

This step involves the setup or the overall environment including world, characters, player motions, camera motions and how the overall game will look like. One has to arrange these things in such a way that it looks visually attractive and challenging to the player.

1.5 Scripting

Scripting is an essential ingredient in all games. Even the simplest game will need scripts to respond to input from the player and arrange for events in the gameplay to happen when they should. Beyond that, scripts can be used to create graphical effects, control the physical behaviour of objects or even implement a custom AI system for characters in the game.

Fig-3: Scripting

Script serialization

Serialization of “things” is at the very core of Unity. Many of our features build on top of the serialization system:

• **Inspector window**. The inspector window doesn’t talk to the C# API to figure out what the values of the properties of whatever it is inspecting is. It asks the object to serialize itself, and then displays the serialized data.

• **Prefabs**. Internally, a prefab is the serialized data stream of one (or more) game objects and components. A prefab instance is a list of modifications that should be made on the serialized data for this instance. The concept prefab actually only exists at editor time. The prefab modifications get baked into a normal serialization stream when Unity makes a build, and when that gets instantiated, the instantiated game objects have no idea they were a prefab when they lived in the editor.

• **Instantiation.** When you call instantiate() on either a prefab, or a gameobject that lives in the scene, or on anything else for that matter (everything that derives from UnityEngine.Object can be serialized), we serialize the object, then create a new object, and then we “deserialize” the data onto the new object. (We then run the same serialization code again in a different variant, where we use it to report which other UnityEngine.Objects are being referenced. Then we check for all referenced UnityEngine.Objects, if they are part of the data being instantiated(). If the reference is pointing to something “external” (like a texture) we keep that reference as it is, if it is pointing to something “internal” (like a child gameobject), we patch the reference to the corresponding copy).

• **Saving.** If you open a .unity scene file with a text editor, and have set unity to “force text serialization”, we run the serializer with a yaml backend.

• **Loading.** Might not seem surprising, but backwards compatible loading is a system that is built on top of serialization as well. In-editor yaml loading uses the serialization system, but also the runtime loading of scenes, assets and assetbundles uses the serialization system.

• **Hot reloading of editor code.** When you change an editor script, we serialize all editor windows (they derive from UnityEngine.Object!). Then we destroy all the windows. We unload the old c# code, we load the new c# code, we recreate the windows, and then we deserialize the datastreams of the windows back onto the new windows.

• **Resource.GarbageCollectSharedAssets().** This is our native garbage collector. It’s a different thing than the c# garbage collector. It is the thing that we run after you load a scene, to figure out which things from the previous scene are no longer referenced, so we can unload them. The native garbage collector runs the serializer in a variation where we use it to have objects report all references to external UnityEngineObjects.

1.6 Adding Sounds

Sound effects are used to enhance the overall gaming experience and increase thrill and the real time feel of the game. It has been used as a game object in Unity3D. This includes audio listener and audio source components. Audio Listener has been added as player component and Audio source enables the sound.

Working with audio assets:

Unity can import audio files in AIFF, WAV, MP3 and OGG formats in the same way as other assets, simply by dragging the files into the Project panel. Importing an audio file creates an Audio Clip which can then be dragged to an Audio Source or used from a script.

1.7 Shaders

A Shader basically defines a formula for how the in-game shading should look. Within any given Shader is a number of properties (typically textures). Shaders are implemented through Materials, which are attached directly to individual GameObjects. Within a Material, you will choose a Shader, then define the properties (usually textures and colors, but properties can vary) that are used by the Shader. Shaders calculate rendering effects on graphics hardware with a high degree of flexibility. The position, hue, saturation, brightness, and contrast of all pixels, vertices, or textures used to construct a final image can be altered on the fly, using algorithms defined in the shader, and can be modified by external variables or textures introduced by the program calling the shader.

1.8 Terrain Engine

Unity’s Terrain system allows you to add vast landscapes to your games. At runtime, terrain rendering is highly optimized for rendering efficiency while in the editor, a selection of tools is available to make terrains easy and quick to create.

2 Conclusion

As game engine technology matures and becomes more user-friendly, the application of game engines has broadened in scope. They are now being used for serious games: visualization, training, medical, and military simulation applications. To facilitate this accessibility, new hardware platforms are now being targeted by game engines, including mobile phones (e.g. Android phones, iPhone, Windows phone) and web browsers. More game engines are being built upon higher level languages such as Java and C#/.NET. Game Engine is the solution to the problem since it is extensible and can be used as the foundation of many different games without major modification. It provides a flexible and reusable software platform which provides all the core functionality needed, right out of the box, to develop a game application while reducing costs.

Acknowledgement

I take this opportunity to express my deep sense of gratitude and respect towards my guide Prof. Dhanashree Wategaonkar, Dept. of IT, MITCOE, Pune. I am very thankful to her for the generosity, expertise and guidance; I have received from her while collecting data on this seminar and throughout my study. Without her support and timely guidance, the completion of my seminar would have seemed a farfetched dream. In this respect I, She has guided me not only with the subject matter, but also taught me the proper style and technique of working and presentation.

It is a great pleasure for me to express my gratitude towards those who are involved in the completion of my paper. I whole-heartedly thank to our HOD Prof. (Dr.) A. S. Hiwale (Dept. of IT) and Principal Prof. (Dr.) M. S. Nagmode for their guidance. The various information and sources I used during my report completion find place in my report. I have tried my best level to make this paper error free. But, I regret for errors, if any.

References

[1] Mariebeth Aquino, Florian Grash¨aftl, Stephanie Kohl,"Content creation using Maya and Unity3D"
[2] A. Jurgelionis, P. Fechteler, P. Eisert, F. Bellotti, H. David, J. P. Laulajainen, R. Carmichael, V. Poulopoulos, , A. Laikari, P. Per¨al¨a, A. De Gloria, and C. Bouras , "Platform for Distributed 3D Gaming", Volume 2009, Article ID 231863, 15 pages, doi:10.1155/2009/231863
[3] Bharti Verma, Anjali Vijayvargiya, Dr. Neeta Nain, "Development approach of 3D PACMAN for android platform", Volume 4, Issue 11, Year Nov 2014, ISSN 2277 128X
[4] Rita Costa Nascimento\ Ana Gil\ Teresa Heitor, Ana Tome, Helena Rua2,"Computer Game engine for Accessibility assessment", 978-1-4673-6165-1/13/$31.00 ? 2013 IEEE
[5] Unity 3D. Unity: Behind the scenes http://unity3d.com/support/documentation/Manual/Behind%20the%20Scenes.html, 2014.
[6] https://www.google.com/patents/US6428413dq=game+engine&hl=en&sa=X&ei=q2vvVP3iJIq3uQThnIH4CA&sqi=2&pjf=1&ved=0CDkQ6AEwBA
[7] https://www.google.com/patents/CN102880464A?cl=en&dq=game+engine+for+windows&hl=en&sa=X&ei=5m3vVIGZF8KIuASc4YDYBQ&sqi=2&pjf=1&ved=0CB0Q6AEwAA.

Unity 3D游戏引擎摄像机动作帧编辑工具应用

Hyungjin Jeon[1], Eeljin Chae[2], Hongsik Pak[3]

Department of Visual Contents, Graduate School, Dongseo University,

47 Jurye-ro, Sasang-gu, Busan 617-716, South Korea

H01086090095@gmail.com[1],{cinetree[2], hspak[3]}@gdsu.dongseo.ac.kr

摘　　要

多媒体互动与游戏相似，它通过一种可以将视觉体验和运动平台的运动关联起来的方法，在以数字屏幕传播的虚拟现实环境中构成游戏内容。本论文介绍了一种摄像机运动帧编辑工具及其操作方法，可用于开发期间直接对交互式游戏内容进行修改。UNITY三维摄像机运动帧编辑器能够实时识别运动平台运动的每一个X、Y、Z值，因此，我们期望这个编辑工具不仅能提高生产质量，而且能大大减少开发时间和费用。

关键词: UNITY 3D，游戏引擎，互动游戏，摄像机运动帧编辑工具，摄像机路径，运动框架

1 介绍

当今时代，我们早已习惯了数字技术和虚拟体验，因而主题公园的发展和所有的游乐设施都与数字化的虚拟现实相关[1]。以互动内容为主题的主题公园呈现出从 3D/4D 互动影院和城市主题公园的业务到互动游戏的扩张。互动游戏通过三维图像实现虚拟现实，结合软件和硬件创造感官体验，以给予用户强大的交互体验[2]。

在设计过程中，让互动游戏内容变得有趣的关键一点是，对各种车辆的动态运动进行视觉化处理，并对摄像机运动轨迹的 X、Y、Z 值进行详细的设计和修正[3]。本论文研究了一种系统，它通过能够对运动帧进行编辑的工具检查用户界面的实时移动来创建和修正任何数据，同时也可以通过鼠标绘图来完成这些过程。

2 相关研究

2.1 现有技术

互动游戏的特点采用了将用户界面上的摄像机运动数据与运动座椅进行同步，并进行显示的方法。然而，在现有的摄像机输入方式，例如 UNITY3D 中的，每一个 X、Y 和 Z 值都必须将坐标作为一个数字来输入，而这样做会造成不便。此外，在进行修正时，特定摄像机路径的具体修改将会影响到其他的方方面面，并且很难在屏幕上检查修正的结果，从而导致效率低下和成本开销[4]。

2.2 先进技术

此处需要用图像来显示 X、Y 和 Z 摄像机的任何运动数据，并使其能够通过鼠标操作来纠正已呈现的或新的图像[5]。此外，还需要研究振动、滚动、角度变化、旋转、摄像机运动、屏幕缩放、在每一帧中设定的速度，以及每个轴特定部分进行的修正和所增加的额外效果。当然，还需要采取更有效的方法——通过工具进行运动帧编辑来在屏幕上查看每一个修正数据，并通过改进摄像机的修正和合成方式来缩短开发时间和成本。

3 系统配置

3.1 摄像机运动图像生成与校正

摄像机运动图像数据在生成环节中收集了运动帧的旋转值——一组关于摄像机 X、Y 和 Z 值的相对值，以便在播放时能够编辑 X、Y 和 Z 轴的旋转值。在这一过程中，获取摄像机相对旋转值的方法就是从当前的角度减去原来的角度，如同 “Vector3 angle = Vector3(CurmainCamera-OriginMainCamera);”，这便得到了摄像机运动图像数据。

用摄像机 Y 轴来换算 Z 轴的旋转角，它的值是通过比较当前摄像机 Y 轴与原摄像机 Y 轴的夹角得到的，即通过函数式“float Zangle = Quaternion.Angle(CurMainCamera.Y, OriginMainCamera.Y);”，接着再用四元数来表示。每个轴的值都会以自动方式保存在 CMF 文件结构中。

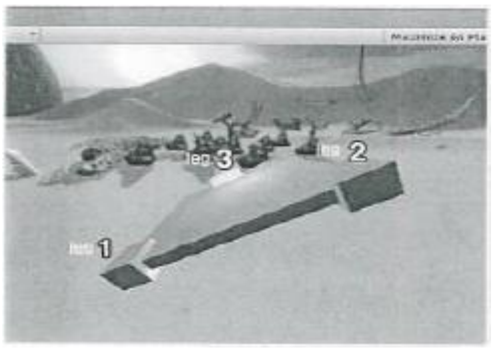

图1 在运动中收集运动数据

3.2 摄像机运动图像数据保存与变化

<图 1>是存储区在运动帧中收集运动数据时的屏幕截图。通过旋转运动帧，可以将 Leg1、Leg2 和 Leg3 箱子的高度作为运动椅上的每一个轴的数据，并将此值保存为 MOT 文件数据。首先，将 Y 轴上收集的值应用到作为运动帧感知对象的每条腿上，然后再在运动帧中获取 Y 轴的初始值以及它在每一帧中的相对位置。

当运动者需要实时微动时，摄像机运动图像数据变化区会从摄像机指针中调用微动的函数，然后从每个轴的当前值中添加或减去微动值。该函数分为两种情况，包括微动的起始与暂停。

4 总结

本研究表明，该系统有一个组合，使得在设计诸如交互式骑行游戏等内容时，可以将摄像机 X、Y 和 Z 轴上的所有移动都视为坐标，并且可以轻松地使用鼠标进行编辑。这意味着，由于简化了开发过程、缩短了时间消耗并加快了摄像机方向的修正过程，使得要求高参与度和娱乐性的多媒体内容在整体质量上能够得到提高[6]。在城市主题公园和互动影院中需要更新游戏内容时，这将会在减少时间和开销上做出重要贡献。

致 谢

本项工作得到了 2015 年“Brain Busan 21”项目的支持。作者们很感激东西大学通过 DSU 项目对这项研究的支持。

参 考 文 献

1. Robinett, J.: 2013 Theme Park Index Museum Index. Themed Entertainment Association(TEA) pp. 24--27 (2013)

2. Eeljin, C., Cheolyoung, C., Kyudon, C., Kihong, K.: Developing a Sensory Ride Film ‘Dragon Dungeon Racing’. Journal of the Korea Contents Association. Vol.11. No.2, pp. 178--185 (2011)

3. AECOM.: China Theme Park Pipeline Report 2013. Inpark Magazine pp.7--8 (2013)

4. Hyungjin, J., Eeljin, C., Hongsik, P., Taesoo, Y.: Study of Camera Motion Frame Creation Tool and CMF File Creation for Interactive Ride Game Content. International Journal of Applied Engineering Research. Vol.9. No 24, pp. 25057--25071 (2014)

5. Hyungjin, J., Eeljin, C., Hongsik, P.: Study of Camera Path and Motion Data Creation for UNITY 3D Game Engine. Advanced Science and Technology Letters. Vol.65 (Games and Graphics 2014), pp. 13--16 (2014)

6. Hawkins, D. G.: “Virtual Reality and Passive Simulators: the Future of Fun.” Communicaion in the Age of Virtual reality, New Jersey (1995)

Application of Camera Motion Frame Editor Tool for UNITY 3D Game Engine

Hyungjin Jeon[1], Eeljin Chae[2], Hongsik Pak[3]

Department of Visual Contents, Graduate School, Dongseo University,

47 Jurye-ro, Sasang-gu, Busan 617-716, South Korea

H01086090095@gmail.com[1],{cinetree[2], hspak[3]}@gdsu.dongseo.ac.kr

Abstract. Multimedia interactive content which is similar with interactive ride game content, produce game content through a method that can concatenate visual experience and physical movement of motion platform within a virtual reality environment that spread through digital screen. In this paper, we introduced a camera motion frame editor tool and operation method, which can directly modify interactive ride game contents during developing period. As UNITY 3D camera motion frame editor tool can identify every X, Y, Z value of motion platform movement in real-time, we are expecting this editor tool can not only improve production quality but reduce the production time and spending drastically.

Keywords: UNITY 3D, Game Engine, Interactive Ride Game, Camera Motion Frame Editor Tool, Camera Path, Motion Frame

1 Introduction

In these days, our lives are used to digital technology and virtual experience so that the development of theme park industry and all the rides goes with such digitalized virtual reality [1]. The theme park based on interactive content has shown the expansion from 3D/4D interactive theatre and urban theme park business to the interactive ride game content. The interactive content makes the virtual reality through the three-dimensional image and delivers it to the users with software and hardware creating sensory experience for the purpose of strong engagement [2].

As the critical point of making the Interactive ride game content amusing is the dynamic movement visualized with various kinds of vehicles on screen, it is important to create and correct the camera movement path of X, Y, and Z in detail while in design process [3]. This study researches the system that makes it possible to create and correct any correction data by checking the real-time movement of UI screen with the tool available to edit motion frame and also to work those process only with mouse drawing.

2 Related Researches

2.1 The Existing technologies

The Interactive ride game content features a displaying method by sync the camera motion data from the movement of UI screen with the movement of motion chair. However, in case of the existing camera input approach such UNITY3D, every value of X, Y, and Z has to be input as a figure by each coordinate and it causes inconvenience [4]. Moreover, whenever correcting, the detailed correction in specific camera path affects the others and it is hard to check the result of correction on screen, causing inefficiency in terms of time and cost.

2.2 The Advanced technologies

There is need to present any motion data from X, Y, and Z camera in the form of graphic image and make it able to correct the presented graph or the new one by mouse operation [5]. Furthermore, it is also necessary to study vibration, rolling, angle variation, rotation, camera movement, zooming screen in/out, setting the given speed in each frame and correcting and extra effects in specific section of each axis. Also, there is need for more efficient way to see every data with correction on screen through the tool for motion frame editing and to cut the production time and cost by improving the camera modification and composition.

3. System Configuration

3.1 Camera Motion Graph Data Creation & Correction

The camera motion graph data creation section collects the rotation value of motion frame referring to the relative value of camera X, Y, and Z so as to edit the rotation value for the axis of camera X, Y, and Z while playing. In this process, the way to collect the relative rotation value of camera is to subtract the original angle from the current angle and it becomes the data for camera motion graph just like “Vector3 angle = Vector3(CurmainCamera-OriginMainCamera);”.
Converting the axis Z rotation against the camera axis Y, its value is found by comparing angles between the current camera axis Y and the original one with the function “float Zangle=Quaternion.Angle(CurMainCamera.Y, OriginMainCamera.Y);” and then presented in quaternion. The value from each axis in automatic method is saved as CMF file structure.

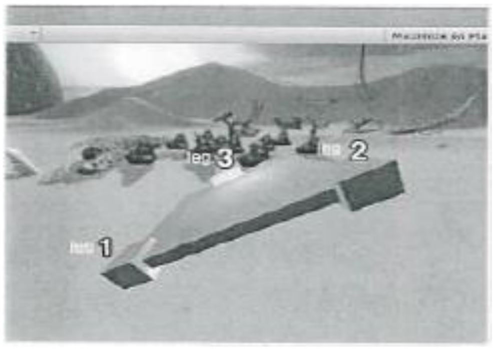

Fig. 1. Collecting motion ride data within motion

3.2 Camera Motion Graph Data saving & variation

<Fig. 1> is the screen shot when the saving section is collecting motion ride data within motion frame. As the height of Leg1, Leg2 and Leg3 box by rotating of motion frame is used for data of each axis in motion chair, this value is saved as MOT file data. The saving section, at first, collects the axis Y value to each Leg, the perceived object by motion frame, and then gets the initial value of Y in motion frame and the relative location value of every frame.
As the Motion Rider needs the real-time micro vibration, the camera motion graph data variation section finds the function for micro vibration from the camera pointer and then adds or subtracts micro vibration value from the current value of each axis in Rider. The function is divided into two cases, the beginning and the pause of micro vibration.

4 Conclusion

This study suggest the system has a composition that makes it possible to see every movement of camera X, Y, and Z as coordinate when designing content such as the Interactive ride game and also edit by mouse drawing with ease. It suggests an advantage of improvement in overall quality of multi-media content that required the high level of engagement and amusement due to the simplified developing procedure, the shorten time consumption and quicken correction process for camera direction [6]. It would be significant contribution to cut time and cost when updating game content having demand in the urban theme park and the interactive theatre.

Acknowledgments. This work was supported by the Brain Busan 21 Project in 2015. The authors would like to acknowledge Dongseo Univ. for supporting this research through DSU project.

References

1. Robinett, J.: 2013 Theme Park Index Museum Index. Themed Entertainment Association(TEA) pp. 24--27 (2013)

2. Eeljin, C., Cheolyoung, C., Kyudon, C., Kihong, K.: Developing a Sensory Ride Film ‘Dragon Dungeon Racing’. Journal of the Korea Contents Association. Vol.11. No.2, pp. 178--185 (2011)

3. AECOM.: China Theme Park Pipeline Report 2013. Inpark Magazine pp.7--8 (2013)

4. Hyungjin, J., Eeljin, C., Hongsik, P., Taesoo, Y.: Study of Camera Motion Frame Creation Tool and CMF File Creation for Interactive Ride Game Content. International Journal of Applied Engineering Research. Vol.9. No 24, pp. 25057--25071 (2014)

5. Hyungjin, J., Eeljin, C., Hongsik, P.: Study of Camera Path and Motion Data Creation for UNITY 3D Game Engine. Advanced Science and Technology Letters. Vol.65 (Games and Graphics 2014), pp. 13--16 (2014)

6. Hawkins, D. G.: “Virtual Reality and Passive Simulators: the Future of Fun.” Communicaion in the Age of Virtual reality, New Jersey (1995)