基于Unity技术3D试衣间的设计与实现

计科142班 王家峰（10141353）

摘要

随着互联网的迅速发展，基于网络的商业项目的数量也越来越多，用户体验作为帮助企业赢得市场的重要手段，地位日趋重要。网络试衣间，顾名思义就是用户在互联网上，可以为自己的虚拟网络分身进行造型塑造、换装搭配的试衣系统。作为一种三维虚拟技术，其拥有实时交互、视觉展现、满足用户心理需求等方面的优点，因其构建用户体验时体现出的独特魅力，在网络商业项目中带来了巨大的商机。本项目的工作是基于三维虚拟应用技术的发展和电子商务应用的需求，开发一个使用三维虚拟技术实现的3D试衣间系统。

本项目研发的虚拟现实3D试衣间系统使用跨平台的游戏开发引擎 Unity3D，结合C#程序开发工具， iClone模型动作开发工具， 实现了3D实时试衣换装。本项目包括系统总体方案设计、3D试衣间需求分析、概要设计、详细设计、系统实现与测试等工作。系统设计与开发包括分别对模型骨骼绑定、蒙皮、动画制作、导入Unity3D引擎、渲染、换装等主要步骤。系统具有场景渲染、人物模特渲染、不同服 装发型的渲染、人物模特在场景中的行走以及换装动画、服装发型的更换等功能，通过GUI界面的控制，使用户可以通过选择，设置出自己想要选择的模特着装方案，并通过引擎渲染，通过Shader镜面反射渲染，得到模特背后的合理场景，从而实现360度三维效果的呈现。

关键词：三维虚拟技术，3D试衣间，Unity3D

Abstract

Asdads

**KeyWords:**Keyword1,keyword2,keyword3,keyword4

目录

[1 绪论 4](#_Toc512847115)

[1.1 虚拟试衣间的现状 4](#_Toc512847116)

[1.1.1 游戏领域的应用 4](#_Toc512847117)

[1.1.2 服装领域试衣间系统现状 4](#_Toc512847118)

[1.2 本项目与以往试衣间系统的不同之处 4](#_Toc512847119)

[1.3 主要研究内容 5](#_Toc512847120)

[2 需求分析 6](#_Toc512847121)

[2.1 开发需求分析 6](#_Toc512847122)

[2.1.1 模型要素 6](#_Toc512847123)

[2.1.2 开发平台 6](#_Toc512847124)

[2.1.3 系统工作流程 7](#_Toc512847125)

[2.2 试衣间功能分析 7](#_Toc512847126)

[2.2.1 换装功能 7](#_Toc512847127)

[2.2.2 界面交互 8](#_Toc512847128)

[2.2.3 动画功能 8](#_Toc512847129)

[2.2.4 试衣镜 8](#_Toc512847130)

[3 系统设计 9](#_Toc512847131)

[3.1 3D试衣间UI分析与设计 9](#_Toc512847132)

[3.1.1 3D试衣间服装部位UI控制的设计与分析 9](#_Toc512847133)

[3.1.2 3D试衣间UI换装功能设计 9](#_Toc512847134)

[3.2 3D试衣间换装核心的设计 9](#_Toc512847135)

[3.2.1 试衣间换装核心的实现 9](#_Toc512847136)

[3.2.2 试衣间文件管理功能设计 9](#_Toc512847137)

[3.2.3 数据字典模块设计 9](#_Toc512847138)

[4 系统实现 10](#_Toc512847139)

[4.1 3D试衣间实现 10](#_Toc512847140)

[4.1.1 3D试衣间服装部位UI控制实现 10](#_Toc512847141)

[4.1.2 3D试衣间UI换装功能实现 10](#_Toc512847142)

[4.2 3D试衣间换装核心的实现 10](#_Toc512847143)

[4.2.1 试衣间换装核心的实现 10](#_Toc512847144)

[4.2.2 试衣间文件管理功能实现 10](#_Toc512847145)

[4.2.3 数据字典模块 10](#_Toc512847146)

[5 测试工作 11](#_Toc512847147)

[5.1 测试元素与计划 11](#_Toc512847148)

[5.2 测试环境 11](#_Toc512847149)

[5.3 测试样例 11](#_Toc512847150)

[5.4 测试结果分析 11](#_Toc512847151)

[6 参考文献 12](#_Toc512847152)

# 绪论

网络试衣间是一种可供用户对自己的虚拟分身进行选择、设计，对服饰、装备进行自由搭配的展示系统。这种供用户自由换装的理念，最早出现于十几年前 网络刚开始流行的时候，当用户己不满足于仅靠文字ID或上传图片头像来代表自己的方式时，提供虚拟分身，自由挑选、任意更换造型的创意初现，并为大型多人在线角色扮演游戏提供了商机，网络使用者为了使自己在造型、穿着上与大 多数人不同，并为游戏的“2D纸娃娃”系统付费，购买服饰配件以使自己区别于其他玩家。

3D试衣间可以在网购服装或者游戏角色选装时，为用户提供服装、装备效果预览，为其设计网络分身、选择服饰装备提供服务，具有很强的互动性和实用性。在传统网络游戏以及页游、手游竞争越来越激烈的今天，用户体验决定了项目成败，谁能提供更优质的画面效果，给用户更全面、体贴的服务功能，谁就更具吸引力，占据竞争的优势，因此开发功能强大的3D试衣间，具有很强的实用 性和广泛的前景。

## 虚拟试衣间的现状

### 游戏领域的应用

近年来随着大型3D网络游戏的盛行，3D试衣间功能也出现在众多火爆的网络游戏中，例如《QQ飞车》中，都出现了试衣间插件，这种插件链接了游戏内部的数据库，当用户点击装饰时，即可在弹出窗口中看到自己的虚拟角色更换装饰的形象，华丽的装备可以刺激玩家的购买欲望，使其花费更多的点券来获得该皮肤。这种3D试衣间需要有网游内部数据库作支持，每套换装部件都需要重新建模，用户做出不同选择时插件可以调用并可渲染各部件，整合到游戏角色身上。

### 服装领域试衣间系统现状

2007年年底，网络试衣间的雏形在中国出现并逐渐流行起来。但由于技术和资金上的瓶颈，还没有比较完美的作品形成，大多是Flash制作的半成品，使用平面的服装照片在平面的模特照片身上进行替换，没有立体感，制作较为粗糙，有的试衣间甚至让人感觉模特的头发像没有用PS处理好就硬贴上去一样。这一片领域上的空白与中国服装网购市场的巨大需求形成强烈对比。

## 本项目与以往试衣间系统的不同之处

本项目与多数3D游戏的试衣插件不同点在于：(1)以往的3D试衣系统是用没有骨骼动画的3D模特进行展示，而本项目研究 的网络3D试衣间的模特是带有骨骼动画的；(2)建模更加精细：以往的3D试衣系统中，模特头、身体、四肢的网格是一个整体，只有换装的装备、道具进行细分建模。而本项目研究的3D试衣间系统中，模特自身也进行了网格细分，头发、脸、身体、四肢分别建模。

## 主要研究内容

本选题研究的3D试衣间与现实服装展示网络试衣间的区别在于：服装展示 网络试衣间是2D平面的，无法进行三维展示，换装只是生硬的服装套用，用户体验较差。而本项目所研究的3D试衣间提供了制作三维服装展示的换装方法，使用这套方法只需在模型制作和贴图制作上多投入工作时间对服装展示类网络试衣间在三维领域的发展提供了技术支持。

# 需求分析

本章节主要讨论项目开发的预备工作，包括开发工具的选择、所需数据的分析，要实现的具体功能等等。本论文设计的项目需要多方面的知识，涉及到3D模型、骨骼动画、游戏引擎使用以及代码实现。所以需要对所有涉及方面进行统筹规划。

## 开发需求分析

### 模型要素

该系统的试衣间应该可以适用于男性与女性，所以首先需要有男性模型与女性模型，两套模型能够独立的进行换装，骨骼动画。

其次，本项目要求对于各个部位的衣物都有相应的模型可以使用。针对衣物主要包括上衣、下装、鞋子，但是考虑到用户的实际使用情况，对于整体形象影响较大的发型、瞳色、面部等部位也需要提供相应的模型以供其搭配整体风格。

在模型的数量方面，要求各个相同的衣物部位必须能够使用同一种骨骼信息，以便于在不同的衣物在同一份模型骨骼上进行切换。例如对于发型需要设置发型1、发型2，同时需要将发型的位置组件与整个身体的组件进行绑定，使之能够再身体运动时头发能够跟随运动。

### 开发平台

U3D C#VS2015阿萨达

本论文涉及到使用游戏引擎对3D模型进行渲染，所以在本项目中，以Unity3D游戏引擎与Visual Studio 2015作为开发平台。Unity引擎具有性价比高、先进易用性、兼容开发的跨平台性以及高质量的3D画面效果等优势[9]。

Unity室友Unity Technoligies开发的一个让用户轻松创建诸如三维视频游戏、建筑可视化、实时三维动画等类型互动内容的多平台综合型游戏开发工具，是一个全面整合的专业游戏引擎。[13]Unity类似于Director、Blender game engine、Virtools或Torque Game Builder等利用交互的图形化开发环境为首要方式的软件，其编辑器 运行在Windows 或Mac OS X下，可发布游戏至Windows、Mac、Wii、IPhone、Winnows Phone 8和Android平台。[]也可以利用UnityWebPlayer插件发布网页游戏 ，支持Mac和Windows的网页浏览，它的网页播放器也可以被MacWidgets所支持。[3]

Unity3D引擎在以下六个方面具有非常的劣势，为系统开发提供了优良的环境，非常适合本项目的开发：

123456.

### 系统工作流程

针对该系统要实现的功能，对该系统整个开发流程进行分析。首先由于本项目核心工作是3D换装，因此对模型有较大的需求，需要准备足够的可以使用的模型，并且要求对要求已经经过骨骼绑定蒙皮的原始的模型素材，以供直接能够在Unity3D中使用。准备好模型之后，需要在Unity3D引擎中搭建试衣间的场景。随后进行UI的设计。设计系统GUI以便用户通过GUI进行换装操作。最后进行统一的编码工作，包括人物模型生成模块、文件管理模块、程序配置模块并将其结合成完成的系统。整个开发流程图如下所示：

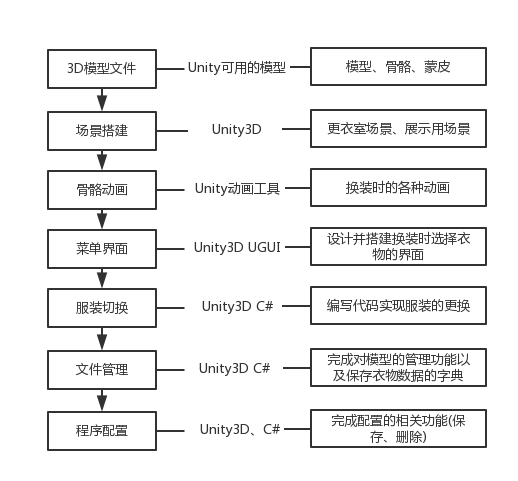


图2-1 系统开发流程

## 试衣间功能分析

针对本试衣间系统的功能分析，首先有可更换的衣物以及实时展示出更换的衣物功能；其次更换衣物都是在用户通过界面交互来控制的，所以需要界面交互；换装之后为了能够有一定的互动需要破防一些动画完成过渡。除此之外，试衣间场景中需要有镜面，所以需要进行镜面Shader的编写。

### 换装功能

换装按钮是本系统的核心功能。从用户角度来讲，进行换装是用户的目的，所以对于换装功能有以下要求：

（1）GUI：用户通过在GUI上互动，选择相应的衣物进而使模型换上相应的衣物模型选择性别来更换模型。

（2）数据字典：用于存储换装部件模型网格、骨骼、贴图、动画的名字与彼此之间的相关联系，组成逻辑上的数据字典，用于数据资源的分类和管理，共角色生成模块调用。

（3）角色生成：将数据字典中的数据进行进行整合，对模型资源进行整理与绑定，并控制相应的模型渲染在场景中。

### 界面交互

任何一个软件界面都是用户通过用户交互界面控制功能，在这里界面交互的需求必须能够满足其3D试衣间的各种功能。

（1）性别选择：由于需要男女两种模型，而且男女模型对应的衣物动画等等都是不同的，所以在用户进行换装之前，必须提供用于选择性别的功能。

（2）衣物选择：确定性别之后，用户即可选择衣物进行换装，针对具体部位的不同，衣物选择分为衣物面部、眼睛、上衣、裤子、鞋子等部位，每个部位下会有多种不同的衣物，当用户点击衣物时，场景中的模型就会换上相应的衣服。

（3）配置保存：用户选择好一套衣服之后，系统需要提供保存的功能，在该系统中，交互界面提供按钮，当用户点击这个按钮时，切换场景进入预览模式，同时将该套衣物的数据保存在配置文件中，下次打开场景时直接从配置文件中读取套装信息。

### 动画功能

人物模型的换装功能结束之后，直接显示换装后的模型而没有动作，会使用户换装体验较差，而现在的各种服务功能都强调以用户为核心[可用些其他论文]。所以本系统中，针对模型换装的各个部位分别制作相应的动画，如：更换发型之后抚摸头发的动作、更换鞋子之后看向鞋子等等。系统中所涉及的每个动画都需要将细分的模型所有部件与骨骼分别绑定并赋予动画。[王潇论文]

### 试衣镜

为了使用户更加有沉浸感，在试衣间场景以及其他展示场景中，本试衣间系统还需要在场景中添加试衣镜功能。试衣镜能够使用户看到模特背向镜头的一面，可以实现用户360度无死角的欣赏自己选择的衣物。在这里我们的选择使用自行编写shader来实现镜面的效果

# 系统设计

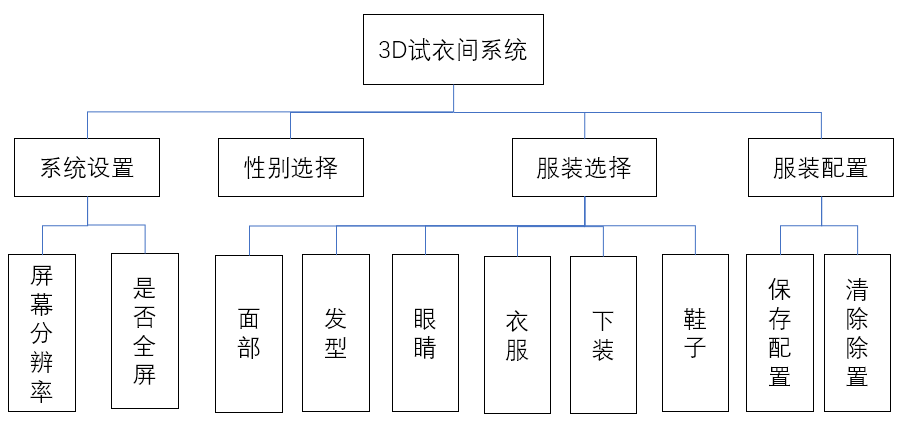
## 3D试衣间的总体设计

* + 1. 试衣间概要设计

该部分讲述根据3D试衣间的主要功能上来对对系统的功能框架设计。如下图3-1所示:

系统由四大功能模块组成，分别为系统设置、性别选择、服装选择和服装配置：

(1)系统设置：用户可以在此界面对3D试衣间系统的图像进行配置。(2)性别选择：用户可以根据实际情况选择模特的性别。(3) 服装选择：包括六种选择类型，用户可以通过对这六部分的点选，进行服装搭配。(4) 服装配置：用户可以选择配置保存来存储己选好的整套服装方案。



**图3-1 3D试衣间的系统功能图**

* + 1. 系统层次结构图

本试衣间系统中，用户为模特选择不同的衣、裤、鞋子，从而产生不同颜色、不同款式进行搭配。为了达到以上的目的，试衣间程序实例中必须使用大量预设的模型基本组成元素，如不同款式的上衣与鞋裤。因此，程序实例需要与一个模型基本元素的文件管理器进行交互。根据实际情况的需要，这个文件管理器可能是存在于本地的文件系统中，也可能存在于互联网的远程数据库中。[引用王潇]整个系统的交互关系，如下图3-2所示:

**图3-2系统层次结构图**

用户

模型数据处理

人物模型的生成与加载

点击鼠标

渲染模型

请求模型数据

返回模型数据

数据库

程序实例

## 3D试衣间UI分析与设计

### 3D试衣间服装部位UI控制的设计与分析

依据第二部分的需求分析，该系统的UI主要分为三部分：

（1） 性别更换：该部分主要的功能是点击之后为用户提供性别更换，将场景中的男性模型更换为女性模型或相反操作。再次要介绍一下toggle，Toggle组合按钮（单选框），可以将多个Toggle按钮加入一个组，则他们之间只能有一个处于选中状态（Toggle组合不允许关闭的话）。Toggle大部分属性等同于Button组件，同为按钮，不同的只是他自带了组合切换功能，当然这些用Button也是可以实现的。在实现方式上，将该功能集中在一组toggle上，为该组toggle都绑定脚本，设定为当该toggle被点击时，执行脚本内的代码进而更换模型以及其他的UI。

（2）保存配置：该部分的功能是点击之后能够将已经搭配好的套装保存在本地的配置文件中。该功能的实现通过在Button上绑定脚本，当button被点击时执行脚本中的代码，保存数据并且切换至另一场景中。

（3）衣物选择：该部分为用户操作最多的部分，用户在这里选择不同部位的各种衣物，所以先针对同一部位的不同衣物进行分组，每一组中的每个衣物都是toggle，确保toggle group中一个toggle被选中时，其他的不被选中。然后对于每一个部位设置为toggle，并且将部位集中在一个toggle group中，确保按照部位选择衣物时，同一时间只能更换一个部位。

### 3D试衣间UI换装功能设计

UI换装功能设计主要是设计实现用户鼠标点击到衣物的UI时，模型能够替换成相应的衣物。具体更换衣物的设计在下一部分换装核心设计中会说明，该部分会分析如何从点击UI中获得相应的数据传送给核心功能。按照3.2.1中的UI设计，每个衣物都是上一部分部件的子节点，所以在UI上绑定脚本，在脚本中获得游戏物体的名称以及其父节点的名称，即可得到更换衣物部位的数据信息。

## 3D试衣间换装核心的设计

核心代码的分析。

### 试衣间换装核心的实现

换装时本系统的核心功能，分析Unity3D的模型结构，人物模型的构成：除去动画以外，名称中带有hip字样的节点即为骨骼，其余数据就是我们能看的到的皮肤。骨骼就是一些Transfrom，没有其他的组件，它们会根据动画的要求进行运动。而皮肤上关键的一个组件是SkinnedMeshRenderer，它关联了3样东西：1.网格SharedMesh；2.骨骼Bones；3.材质Material。所以只要在代码中改变以上三种数据，就能替换模型上的部件。

### 试衣间文件管理功能设计

本系统中的文件管理，主要是指管理本系统中的配置文件的管理。根据需求分析，配置文件中的主要内容是用户已经搭配好的套装。而在该系统中，文件管理主要作为本地数据库的存在，在技术实现上，本系统采用XML文件存储当前衣物的配置。当用户搭配好衣物之后，在UI界面点击保存配置对应的按钮时，代码中会执行文件管理模块的代码，将配置依照规则保存在XML文件中，系统下一次启动时，即可从文件管理模块中读取XML文件，取出保存的衣物配置并将其加载如场景中。

### 数据字典模块设计

数据字典模块在该系统中保存模特以及衣物的各种信息，主要的数据字典分析如下：

（1）衣物信息：该数据存储所有可更换的部位的数据，在这里存储的主要内容即为SkinnedMeshRenderer，而作为数据词典，为了方便查找，本系统选择了使用C#中的Dictionnary来存储该数据。Dicitionary <T1,T2>是一个泛型，他本身有集合的功能有时候可以把它看成数组，他的结构是这样的：Dictionary<[key], [value]>，他的特点是存入对象是需要与[key]值一一对应的存入该泛型，通过某一个一定的[key]去找到对应的值。例如定义Dictionary<string,int> dict,添加数据是为dict.add(“first”,1)，查找时dict[“first”]。使用Dictionnary可以快速的通过键值查找数据。根据3.2.1的UI设计分析，本系统需要根据部位和衣物编号来获取相应数据，所以Dictionnary设计为Dictionary<string, Dictionary<string, SkinnedMeshRenderer>>结构。

（2）模特信息：首先需要考虑的模特的骨骼信息，因为在模特的使用过程中所需要的数据主要hips，存储当前模型的骨骼信息，用于与更换的衣物进行匹配；另一组数据即为SkinnedMeshRenderer，用于存储当前模特正在使用的SkinnedMeshRenderer。

## 试衣镜

试衣镜功能主要是用于展示模特背向摄像机的一面，主要利用到的技术是Shader的编写。下面开始分析局势的实现方法。

基于简单的物理知识，由于本系统的试衣镜是平面镜，我们知道镜面上反射的虚像与实际的实像是关于镜面对称的，虚像与实像各个对应顶点连线与镜面垂直且到镜面的距离相等。所以我们知道实像的信息与镜面的信息即可通过一系列的计算得到虚像。再次我们将其转换为数学公式解决这一问题。

假设现在有一点Q(x,y,z)，有平面,为平面上的点，为法向量，=(nx,ny,nz)，d为原点到平面的距离，P0(x0,y0,z0)为平面上一点。如下图所示：

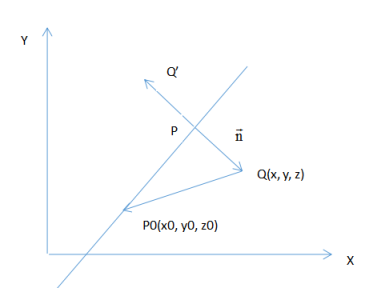


图4.1 示例

于是我们可以用下面的等式表示平面 d=-x0xn-y0yn-z0zn。

则：Q’=Q -2\***N**\*|QP|(QP 为点Q到镜面的距离)

|QP|为QP0在**N**上的投影|QP|=**QP0**·**N**/|**N**|

则带入各点的坐标

同理，y’与z’：

所以可得反射矩阵R：

考虑基本的渲染管线中的坐标变换，一般我们使用MVP来表示将一个点从物体坐标系转换到裁剪坐标系，其中M(model\_matrix)表示将点从物体坐标系转换到世界坐标系；V(view\_matrix)表示将点从世界坐标系变换到视点坐标系(摄像机坐标系);P(project\_matrix)表示将点从视点坐标系转换到裁剪坐标系。如果我们获取到了在世界坐标系下平面的法线向量和d（d可以通过平面上任意一点求得），那就可以求出在世界坐标系下将顶点转换到反射点的反射矩阵了。于是我们就可以在完成了M矩阵变换后，进行反射矩阵的变换，然后再接着完成V和P矩阵的变换。接着思考，在unity3d中，摄像机有个worldToCameraMatrix变量，这个变量就是V，那我们可以这样做：V=V\*R，这样经过MVP矩阵变换的顶点不就自然而然的经过了反射矩阵的变换了么？所以我们考虑复制一个当前摄像机（这可以通过Camera.CopyFrom来实现），将这个摄像机的worldToCameraMatrix乘以反射矩阵R，那么这个摄像机渲染出来的物体就是虚像。

# 系统实现

## 3D试衣间实现

### 3D试衣间服装部位UI控制实现

本项目中Unity3D中的UI系统为UGUI，可以通过简单的拖动组合等操作组合出UI界面。在这里针对各个部分进行分析

(1)衣物：针对每一个衣物设置UI，其主体为toggle，调整大小之后为其添加衣物的图片以及表示选择状态的图片，如下图选择发型时的UI界面：



图4-1 衣物选择UI

实现方式通过在toggle的属性中选择对应的图片，在toggle处于被选中状态时，图片就会显示，否则图片就会处于不可见状态。

(2)部位：部位主要有两个部分组成，第一部分是包含具体衣物的Panel组件，所有具体衣物的UI都按照部位依附于对应Panel上，同时每个Panel上还要添加togggle group组件，以控制每一个部位中同一时间只能选择一件衣物；另一部分为Toggle组件，这部分也分成了各个部位，其主要功能是控制对应的Panel的可见性，进而保证选择某一部位时，UI界面弹出相应部位的衣服UI。完成效果图如下：



图4.2 部件选择UI效果图

### 其他UI功能

除换装UI之外的，其余的重要UI有性别选择、配置保存、以及旋转查看等功能。

（1）性别选择：该功能基础组件也是toggle，搭建方式与部件选择类似，因为性别选择包含逻辑功能，所以要为其添加脚本SexMoudle，主要代码如下：

public void changeSex(bool isOn)

{

if (!isOn) return;

int isMale = AvatarSys.\_instance.nowCount;//isMale判断是否为男性，nowCount为核心换装中的标志位。nowCount==0表示为目前模型为女性

if (gameObject.name == "boy" && (isMale >0))

{

return;//如果选择了男性且当前也是男性，不作处理直接返回

}

if (gameObject.name == "girl" && (isMale ==0))

{

return; //如果选择了女性且当前也是女性，不作处理直接返回

}

if (isMale == 0)

{

AvatarSys.\_instance.clearGirl();//调用后核心换装模块的相应功能，隐藏女性模型数据，显示男性模型

}

else

{

AvatarSys.\_instance.clearBoy();

}

AvatarSys.\_instance.nowCount = 1 - isMale;//更改在核心模块中存储的标志位

}

以女性模型为例为例，clearBoy()的代码

public void clearBoy()

{

boyTarget.SetActive(false);

girlTarget.SetActive(true);

}

将该脚本添加到UI上并在toggle属性中设置当toggle被点击时执行该段代码即可。

（2）配置保存：该组件为Button组件，在添加AvatarButton脚本组件并设置为当被点击时执行LoadScenes()方法：

public void LoadScenes()

{

DateManager.GetInstance().SaveCloth();//通过单例模式调用保存衣物配置的方法

SceneManager.LoadScene(1);//调用Unity3D的方法切换场景

}

### 3D试衣间UI换装功能实现

由需求分析与具体UI实现，这里给出相应的代码：

public void OnValueChanged(bool isOn)

{

if (!isOn)//判断对应组件的toggle组件是否为isON(即被选中)状态

{

return;

}

string[] name = gameObject.name.Split('-');//划分部位与衣物编号

AvatarSys.\_instance.onChagePeople(name[0],name[1]);//将部位信息与衣物编号信息传给核心换装功能部分

switch (name[0])//根据部位名称进行播放对应的动画

{

case "pants":

PlayAnimation("item\_pants");

break;

case "shoes":

PlayAnimation("item\_boots");

break;

case "top":

PlayAnimation("item\_shirt");

break;

default:

break;

}

}

public void PlayAnimation(string animName) //播放换装动画

{

Animation anim = GameObject.FindWithTag("Player").GetComponent<Animation>();

if (!anim.IsPlaying(animName))

{

anim.Play(animName);

anim.PlayQueued("idle1");

}

}

## 3D试衣间换装核心的实现

本系统中，换装分为男女角色，进行不同的数据操作，在代码中表现为按照标志nowCount的至判断调用男性角色或是女性角色的方法。由于两者的代码基本相同，所以这这里以女性模型为代表进行分析。

### 试衣间换装核心的实现

如3.2.1分析，换装的核心主要是对模型的SkinnedMeshRenderer进行更改，所以首先分析主要的换装部分代码：

void changeMesh(string part, string num,

Dictionary<string, Dictionary<string, SkinnedMeshRenderer>> data,

Transform[] hips,

Dictionary<string, SkinnedMeshRenderer> smr)//部位与编号

{//data中存储衣物数据

SkinnedMeshRenderer skm = data[part][num];//要更换的部位

List<Transform> bones = new List<Transform>();//要更换的谷歌部分

foreach (var trans in skm.bones)//如果一个骨骼既存在于要更换的部位中，有存在于target的骨骼中，name他就是要被更换的骨骼

{

foreach (var bone in hips)

{

if (bone.name == trans.name)

{

bones.Add(bone);

break;

}

}

}

//huanzhuang shixian guge caizhi mesh

smr[part].bones = bones.ToArray();//更换骨骼

smr[part].material = skm.material;//更换材质

smr[part].sharedMesh = skm.sharedMesh;//更换网格

if (nowCount ==0)

SaveData(part, num, girlstr);//将更换厚的衣服保存

else

SaveData(part, num, boystr);

}

### 试衣间文件管理功能实现

文件管理功能在这里主要是能够将用户使用的衣物数据保存到本地的XML文件中，为了确保不被重复读写本地文件，所以在这里将文件管理功能设计为单利模式，确保只有一个对象对文件进行操作。

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using System.Xml;

using System.IO;

public class DateManager : MonoBehaviour {

private static DateManager \_instance;

//XmlAttribute xmla;

public XmlDocument DateSource;

public XmlElement date;

public XmlElement date1;

public XmlNode node1;

string m\_XmlName = "/Configs/config.xml";

string m\_XmlPath;

public Dictionary<string, string> ClothConfig = new Dictionary<string, string>();

private void Awake()

{

\_instance = this;

m\_XmlPath = Application.dataPath + m\_XmlName;

if (!File.Exists(m\_XmlPath))

{//当不存在配置文件是，新建文件的内容

DateSource = new XmlDocument();

date = DateSource.CreateElement("Cloth");//新建cloth节点

DateSource.AppendChild(date);//cloth节点设置为XML文件的子节点

XmlElement elment = DateSource.CreateElement("Sex");

elment.InnerText = "0";

date.AppendChild(elment);

elment = DateSource.CreateElement("eyes");

elment.InnerText = "1";

date.AppendChild(elment);

elment = DateSource.CreateElement("hair");

elment.InnerText = "1";

date.AppendChild(elment);

elment = DateSource.CreateElement("top");

elment.InnerText = "1";

date.AppendChild(elment);

elment = DateSource.CreateElement("pants");

elment.InnerText = "1";

date.AppendChild(elment);

elment = DateSource.CreateElement("shoes");

elment.InnerText = "1";

date.AppendChild(elment);

elment = DateSource.CreateElement("face");

elment.InnerText = "1";

date.AppendChild(elment);

DateSource.Save(m\_XmlPath);

}

else

{

DateSource = new XmlDocument();

DateSource.Load(m\_XmlPath);

}

initdate();

}

private void initdate()//从XML读取衣物套装的数据并将其保存在ClothConfig中，ClothConfig 将别传送给核心数据字典中用来生成模型

{

XmlNodeList nodes = DateSource.SelectSingleNode("Cloth").ChildNodes;//获取名为Cloth的节点的所有子节点

foreach(XmlElement ele in nodes)

{

ClothConfig.Add(ele.Name,ele.InnerText);

}

}

public void changenode(string nodename,string nodevalue)//将XML文件中名为nodename节点中的值改为nodevalue

{

node1 = DateSource.SelectSingleNode(nodename);

node1.InnerText = nodevalue;

DateSource.Save(m\_XmlPath);

}

public static DateManager GetInstance()//获得对象的实例

{

return \_instance;

}

// Use this for initialization

void Start () {

}

// Update is called once per frame

void Update () {

}

public void SaveCloth()//对衣物进行保存

{

changenode("Cloth/eyes",AvatarSys.getInstance().getClothConfig("eyes"));

changenode("Cloth/hair", AvatarSys.getInstance().getClothConfig("hair"));

changenode("Cloth/top", AvatarSys.getInstance().getClothConfig("top"));

changenode("Cloth/pants", AvatarSys.getInstance().getClothConfig("pants"));

changenode("Cloth/shoes", AvatarSys.getInstance().getClothConfig("shoes"));

changenode("Cloth/face", AvatarSys.getInstance().getClothConfig("face"));

}

}

### 数据字典模块

数据字典中存储了在项目运行中的大部分数据，这里同样以女性模型的数据为例进行分析：

1. 数据字典定义：如下代码所示：

private Dictionary<string, Dictionary<string, SkinnedMeshRenderer>>

girlData = new

Dictionary<string, Dictionary<string, SkinnedMeshRenderer>>();//存储所有的衣物资源

private Dictionary<string, SkinnedMeshRenderer> girlSmr= new Dictionary<string, SkinnedMeshRenderer>(); //字典存储信息当前模型正在使用的SkinnedMeshRenderer

Transform[] girlHips;//当前模型正在使用的骨骼信息

private string[,] girlstr = new string[,] { { "eyes","1"},

{"hair","1" },

{ "top","1"},

{ "pants","1"},

{ "shoes","1"},

{"face","1" } };

//当前模型正在穿着的衣物

1. 数据字典的初始化：

public void initGirl()

{

instaniateGirlAvatar();

saveDate(girlSourceTransform,girlData,girlTarget,girlSmr);

initAvatar();

}

void instaniateGirlAvatar()

{

GameObject go = Instantiate(Resources.Load("FemaleModel")) as GameObject;//加载资源，此处的资源是包含有所有的衣物的资源

girlSourceTransform = go.transform;//

go.SetActive(false);

girlTarget = Instantiate(Resources.Load("FemaleTarget")) as GameObject;//实例化展示用的模型的资源，此时模型只有骨骼和动画信息

girlHips = girlTarget.GetComponentsInChildren<Transform>();//获取骨骼信息

}

//saveDate从资源文件中获取所有的衣物模型数据的过程

void saveDate(Transform sourceTransform,

Dictionary<string, Dictionary<string, SkinnedMeshRenderer>> data,

GameObject target,

Dictionary<string, SkinnedMeshRenderer> smr)

{

if (sourceTransform== null)

{

return;

}

data.Clear();

smr.Clear();

SkinnedMeshRenderer[] parts = sourceTransform.GetComponentsInChildren<SkinnedMeshRenderer>();

foreach (var part in parts)

{

string[] names = part.name.Split('-');

//if (names[0] = )

if (! data.ContainsKey(names[0]))//确保一个部位之声层一次

{

//生成对应的部位

GameObject partGo = new GameObject();

partGo.name = names[0];

partGo.transform.parent = target.transform;

smr.Add(names[0],partGo.AddComponent<SkinnedMeshRenderer>());//把骨骼target的信息存储到skm信息

data.Add(names[0],new Dictionary<string, SkinnedMeshRenderer>());

}

data[names[0]].Add(names[1], part);//存储所有的skinedmeshrennder信息

}

}

## 试衣镜

脚本：

using UnityEngine;

using System.Collections;

public class Reflection : MonoBehaviour {

public Transform Panel;

public Camera RefCamera;

public Material RefMat;

// Use this for initialization

void Start () {

if(null == RefCamera)

{

GameObject go = new GameObject();

go.name = "refCamera";

RefCamera = go.AddComponent<Camera>();

RefCamera.CopyFrom(Camera.main);

RefCamera.enabled = false;

RefCamera.cullingMask = ~(1 << LayerMask.NameToLayer("Water"));

}

if(null == RefMat)

{

RefMat = this.GetComponent<Renderer>().sharedMaterial;

}

RenderTexture refTexture = new RenderTexture(Mathf.FloorToInt(Camera.main.pixelWidth),

Mathf.FloorToInt(Camera.main.pixelHeight), 24);

refTexture.hideFlags = HideFlags.DontSave;

RefCamera.targetTexture = refTexture;

}

// Update is called once per frame

void Update () {

}

public void OnWillRenderObject()

{

RenderRefection();

}

void RenderRefection()

{

Vector3 normal = Panel.up;

float d = -Vector3.Dot(normal, Panel.position);

Matrix4x4 refMatrix = new Matrix4x4();

refMatrix.m00 = 1 - 2 \* normal.x \* normal.x;

refMatrix.m01 = -2 \* normal.x \* normal.y;

refMatrix.m02 = -2 \* normal.x \* normal.z;

refMatrix.m03 = -2 \* d \* normal.x;

refMatrix.m10 = -2 \* normal.x \* normal.y;

refMatrix.m11 = 1 - 2 \* normal.y \* normal.y;

refMatrix.m12 = -2 \* normal.y \* normal.z;

refMatrix.m13 = -2 \* d \* normal.y;

refMatrix.m20 = -2 \* normal.x \* normal.z;

refMatrix.m21 = -2 \* normal.y \* normal.z;

refMatrix.m22 = 1 - 2 \* normal.z \* normal.z;

refMatrix.m23 = -2 \* d \* normal.z;

refMatrix.m30 = 0;

refMatrix.m31 = 0;

refMatrix.m32 = 0;

refMatrix.m33 = 1;

RefCamera.worldToCameraMatrix = Camera.main.worldToCameraMatrix \* refMatrix;

//在计算漫反射等光照效果时，需要使用顶点的normal和view向量，view跟摄像机位置有关，所以我们也对refcamera做反射变换

RefCamera.transform.position = refMatrix.MultiplyPoint(Camera.main.transform.position);

//以下部分是变换摄像机的方向向量，当然其实这里没有必要，你可以删掉它

Vector3 forward = Camera.main.transform.forward;

forward = refMatrix.MultiplyVector(forward);

RefCamera.transform.forward = forward;

GL.invertCulling = true;

RefCamera.Render();

GL.invertCulling = false;

//将贴图传递给shader

RefCamera.targetTexture.wrapMode = TextureWrapMode.Repeat;

RefMat.SetTexture("\_RefTexture", RefCamera.targetTexture);

}

}

Shader：

Shader "Custom/Mirror" {

Properties

{ \_MainTex("Base(RGB)",2D) = "white"{}

\_ReflectionTex("Reflection",2D) = "white"{TexGen ObjectLinear}

}

// two texture cards: full thing

Subshader

{ Pass

{ SetTexture[\_MainTex]{ combine texture }

SetTexture[\_ReflectionTex]{ matrix[\_ProjMatrix] combine texture\*previous }

}

}

// fallback: just main texture

Subshader

{ Pass

{ SetTexture[\_MainTex]{ combine texture }

}

}

# 测试工作

## 测试元素与计划

需要测试的功能：换装功能、性别更换功能、动画播放、镜面反射、配置。

测试计划：

1. 按照系统的核心功能进行测试，先进行换装、之后依次进行性别更改、动画播放、镜面反射以及文件配置功能。
2. 系统测试将会进行功能以及可用性方面的测试。
3. 本测试优先采用黑盒测试，在发现错误时进行白盒测试。

## 测试环境

本系统在内存4Gm，CPU I5-3230(M),主频2.6GHz，硬盘500G，Windows10环境下测试。

## 测试样例

|  |  |  |  |
| --- | --- | --- | --- |
| 测试元素 | 描述 | 期望结果 | 测试结果 |
| 菜单界面 | 选择相应的部位时，能够显示正确的对应下的衣物 | 选择相应的部位时，能够显示正确的对应下的衣物 |  |
| 模型渲染 | 模型能够在场景中正确的渲染 | 运行程序之后，人物模型出现在正确的位置且无明显其他错误 |  |
| 更换发型 | 点击界面上相应的UI，查看场景中模型情况 | 模型对应部位更华为选择的 |  |
| 更换眼球 |  |  |  |
| 更换上衣 |  |  |  |
| 更换裤子 |  |  |  |
| 更换鞋子 |  |  |  |
| 性别更换 |  |  |  |
| 动画播放 | 快速更换衣物，观察是否会出现动画播放混乱 | 一段动画未播放完时，新要求的动画能够打断未播放完的动画 |  |
| 镜面反射 | 观察镜面是否能够正确的反射模型背向摄像机的一面 |  |  |
| 配置 | 保存配置后，再次打开能够直接渲染出上一次保存的配置 |  |  |

## 测试结果分析

经测试，本3D试衣间系统没有明显的BUG，运行过程中没有明显的卡顿现象，可以正确的执行各种操作。

# 参考文献

1. 蔡升达.设计模式与游戏完美开发[M].清华大学出版社.2017.1.
2. 郭浩瑜.Unity 3D ShaderLab开发实战详解[M].人民邮电出版社.2014.4.
3. 刘晴.基于服装消费者的网络购物关键问题研究[D].北京：华北电力大学(保定).2014.
4. 李汉文.3D虚拟衣服动画系统关键技术的研究与实现[D].兰州大学.2012.
5. 陆永良,李汝勤,胡金莲.虚拟服装的发展历史和现状[J].纺织学报. 2005(01).
6. 徐冉,何琳.3D虚拟试衣技术在网络营销中的价值研究[J].现代装饰(理论). 2016(11).
7. 路朝龙.Unity 3D游戏开发从入门到精通[M].中国铁道出版社.2013.11.
8. 谌宝业，刘若海.游戏角色动画设计[M].清华大学出版社.2012.1.
9. 王潇.基于Unity技术3D试衣间的设计与实现[D]. 北京北京工业大学. 2014.
10. 王中州，李晓峰，张洋.网络游戏美工制作教程[M].电子工业出版社.2011.1.
11. 吴亚峰，于复兴.Unity 3D游戏开发技术详解与典型案例[M].人民邮电出版社.2012.11.
12. 张伊凡.基于unity3D开发引擎的虚拟试衣间的设计[J].科技风.2017,6:1-1.
13. （美）Steve Roberts （著），韩佳，刀海鹏（译）.角色动画基础2D和3D角色动画制作全解析[M].人民邮电出版社.2013.6.
14. （美）米歇尔.梅纳德（著），石晓明，李强（译）.Unity游戏开发实战[M].机械工业出版社.2012.4.
15. (US)Suejung B. Huh,(US)Dimitris N. Metaxas.A collision resolution algorithm for clump-free fast moving cloth[J]. Computer Graphics International,2005,22 (6) :51-58.
16. Oktar Ozgen,Marcelo Kallmann.Directional Constraint Enforcement for Fast Cloth Simulation[M]:Motion in Games: 4th International Conference, MIG 2011, Edinburgh, UK, November 13-15, 2011. Proceedings (pp.424-435).