计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2021,30(6):61−67 [doi: 10.15888/j.cnki.csa.007974]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于 Unity3D 的换装系统插件设计①

黄兆铭, 储　颖

(深圳大学 计算机与软件学院, 深圳 518060)
通讯作者: 储　颖, E-mail: chuying@szu.edu.cn

摘　要: 针对电子游戏领域缺乏国产、开源换装系统插件的现状, 设计并实现了基于 Unity3D 的换装系统插件: 方舟角色创建器 (Ark Avatar Customization, AAC). AAC 支持以可视化的方式更换游戏角色装扮、调整骨骼参数及外观颜色, 可满足开发者和游戏玩家的个性化换装需求. 论文以 Unity3D 为软件开发平台, 结合改进的贴图合并算法、编辑器扩展技术和预制件优化技术, 不仅实现了换装功能, 还大幅优化了系统性能. 性能比较实验的结果表明, AAC 的时间开销和空间开销较现有技术均有明显改善, 证实了所设计换装系统插件的实用性与有效性.

关键词: 换装系统插件; Unity3D; 贴图合并; 编辑器扩展; 预制件优化

引用格式: 黄兆铭,储颖.基于 Unity3D 的换装系统插件设计.计算机系统应用,2021,30(6):61–67. http://www.c-s-a.org.cn/1003-3254/7974.html

Design of Role-Customization System Plug-In Based on Unity3D

HUANG Zhao-Ming, CHU Ying

(College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518060, China)

Abstract: In response to the lack of domestic open-source role-customization system plug-ins in the field of computer game development, a Unity3D-based role-customization system plug-in, Ark Avatar Customization (AAC), is designed and implemented. AAC supports to change game characters' dress-up and adjust their bone parameters, appearances and colors in a visual way, enabling role-customization required by developers and players. This study takes Unity3D as a platform for software development to fulfill customization functions and optimize system performance with the improved texture merge algorithm, editor extension, and prefab optimization. Performance comparison reveals the time and space cost of AAC are lower than that of existing techniques, proving the practicability of the designed role-customization system plug-in.

Key words: role-customization system plug-in; Unity3D; texture merge; editor extension; prefab optimization

如今的电子游戏, 不仅仅是由代码编写的单纯软件, 更承载了开发者的内心世界[1]. 正因如此, 电子游戏的题材与玩法也愈加丰富. 相比传统电子游戏中固定的游戏角色, 如今的开发者和玩家对游戏角色的期望值越来越高, 希望在游戏中获得更多的自由感 (如: 自定义游戏角色模型). 换装系统, 正是游戏开发人员和游戏玩家追求自由的体现.

如果一款游戏搭载了优秀的换装系统, 对于开发者, 有可能大大降低开发难度; 对于玩家, 则有机会升级游戏体验. 举例来说, 著名的装扮类游戏《闪耀暖暖》[2]、模拟经营类游戏《模拟人生》[3]、大型多人在线角色扮演游戏《魔兽世界》[4] 等, 都内置优秀的换装系统, 且换装系统已成为这些游戏不可或缺的一部分.

然而, 独立游戏开发者想要实现具备换装功能的

① 基金项目: 广东省高等教育教学改革项目 (粤教高函 [2018]180); 深圳大学计算机与软件学院高水平大学二期建设本科教学建设项目 ([2020]02)
Foundation item: Teaching Reform Project of Higher Vocational Education of Guangdong Province ([2018]180); The Second Phase of the Undergraduate Teaching Construction Project for Constructing the High-level University of the School of Computer and Software, Shenzhen University ([2020]02)
收稿时间: 2020-10-04; 修改时间: 2020-10-28; 采用时间: 2020-12-01; csa 在线出版时间: 2021-06-01

游戏, 是相当有挑战性的. 原因在于, 大型商业游戏工作室基本自研换装系统, 一般不会公开相关技术细节, 而国内也没有成熟的文献资料供参考. 因此, 利用换装系统插件完成换装功能, 成为众多独立游戏开发者和小型游戏工作室的首选.

设计换装系统插件给开发人员带来以下挑战:

(1) 由于插件的使用者多为美工, 所设计插件应使无编程基础的使用者也能快速上手;

(2) 设计换装系统插件涉及大量建模和图形学知识, 这就要求开发人员对美工职能有较为深刻的理解.

因此, 游戏开发者要设计并实现换装系统插件是有一定难度的. 国内学者的研究工作有涉及换装技术的, 例如: 杨静[5] 基于 HTC VIVE 设计实现了虚拟换装游戏, 李俊等[6] 基于人物替换技术实现了模拟试衣间, 徐康熙等[7] 基于物理方式实现了虚拟换衣系统. 然而, 这些换装、试衣系统的功能较为单一, 仅能实现特定环境下的基础装扮更换, 不能称为真正的换装系统插件.

目前, 市面上开源、开放使用的换装系统插件寥寥无几. 少量可供使用的换装系统插件中, 功能较为强大的是国外开发者基于 Unity3D[8] 设计的 Unity Multipurpose Avatar (UMA) 系统. UMA 系统的特点是开源、易用和功能丰富, 但也有缺点, 例如: 时间、空间复杂度较高; 更适用于制作美术模型, 而不适合运行时的角色动态合成.

为填补以上空白, 论文设计并实现了一个基于 Unity3D 平台的国产、开源换装系统插件, 命名为方舟角色创建器 (Ark Avatar Customization, AAC). AAC 插件采用改进的贴图集生成、编辑器扩展和预制件优化技术, 集成换装、捏脸和换色功能, 配合图形化扩展工具, 能够充分满足开发者游戏开发过程中的动态、个性化换装需求.

1 系统总体架构

首先, 从软件工程思想出发, AAC 系统设计时需兼顾功能性、可维护性与可扩展性. 同时, 依据设计模式所倡导的单一职责原则[9], 各个系统模块应尽量独立. 因此, AAC 插件系统定义了一系列新的、功能独立的概念, 包括: 装扮、种族、角色、基因和染料. 各概念具体含义如表 1 所示.

1.1 装扮

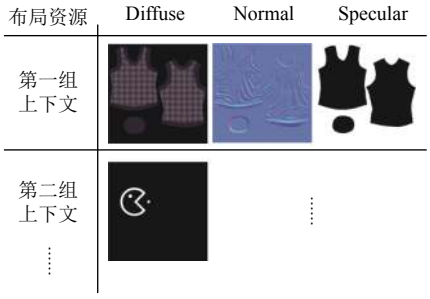
AAC 系统规定, 以装扮为基本单位进行角色的合成, 并使用字符串标识装扮所属的插槽类型. 装扮用以装备在角色身上, 每个插槽同一时间只能装备一个装扮. 当为角色装备插槽重复的装扮时, 旧的装扮将会被替换掉.

表 1 AAC 插件系统概念定义

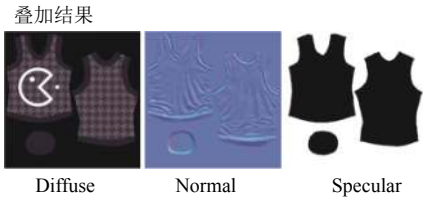
名称	含义
装扮	美术模型
种族	角色的生成模板, 描述不同角色的共同数据
角色	独立参与合成的个体
基因	影响角色体形的因素
染料	影响装扮颜色的因素

装扮代表了美术模型, 可以是上衣、饰品或身体部位 (取决于使用者的目的). 通常, 使用者会将完整模型按照不同部位切分为局部模型, 如: 头发、上衣、下衣、鞋子等, 再分别制作成装扮.

系统实现方面, 由于模型包含骨骼、网格和贴图数据, 因此装扮需要存储它们作为角色合成时的数据依据. 为符合单一职责原则, AAC 系统定义了两个概念分别存储美术模型数据, 它们是: 槽位资源和布局资源. 槽位资源负责存储模型的骨骼和网格, 布局资源负责存储模型贴图, 如图 1 所示.



(a) 布局资源



(b) 上下文合成结果

图 1 AAC 布局资源和上下文示意图

需要指出的是, 槽位资源为装扮合成提供骨骼和网格依据, 而布局资源的功能不限于提供原模型贴图. 类似 Photoshop 的图层叠加原理, 假如将原模型贴图定义为第 1 组上下文, AAC 系统规定: 布局资源可以包含一组以上的上下文 (见图 1(a)). 进行装扮合成时, 从

第 2 组开始, 依次将各上下文叠加至第 1 组上下文, 最终得到目标贴图组 (见图 1(b)).

这样设计的好处在于: 能够实现类似“纹身”的装扮效果. 仅需更换“纹身贴图”, 就可以在不破坏、不冗余原模型贴图的情况下, 生成新的目标贴图. 此外, 由于“纹身贴图”一般来说精度不高, 系统可以使用较低的分辨率存储它们以节省空间. 结合染料的概念, “纹身贴图”还可以起到遮罩图层的作用, 即: 仅改变“纹身贴图”中非透明区域的颜色.

图 1 中, 第 2 组上下文的“吃豆人纹身”的有效区域只有很小一部分, 但为了与第一组上下文中左边背心的中心对齐, 贴图预留了很多“黑边”. 为进一步优化内存空间, 可以缩小“吃豆人纹身”的贴图大小, 采用记录偏移值的方式, 将“纹身”移动到背心中间.

1.2 种族

AAC 定义种族的目的, 是为创建角色提供依据. 与现实生活类似, 具有相同特征的个体被划分至同一种族. 换言之, 种族描述了这些特征的共性. 种族的基础数据定义如下:

(1) 骨架描述和人类描述;

(2) 装扮类型描述;

(3) 基因描述和基因作用器描述;

(4) 染料描述.

其中, 骨架描述用来重建角色的初始骨架, 人类描述用来创建 Unity3D 动画系统中的标准人类骨架. 它们都可以从导入至 Unity3D 的模型数据中获取.

装扮类型描述定义了某种族可接受的插槽类型. AAC 系统规定, 只有处于装扮类型描述中的装扮才是有效的、能够参与角色合成的装扮. 通过配置不同的装扮类型描述, 可以实现为游戏主角 (或者其种族) 装备更多部位装扮、而为 NPC (Non-Player Character) 装备较少装扮的效果.

基因描述和染料描述的含义, 将分别在 1.4 和 1.5 节中详细介绍.

1.3 角色

进一步地, 为了区分种族中每个单独作用的个体, AAC 系统提出角色的概念. 角色在种族中创建得到, 但每个角色可以有各自不同的表现 (如: 装备的装扮不同). 系统需要为每个角色单独存储以下数据:

(1) 装扮集合;

(2) 骨架和 Avatar;

(3) 基因数据和染料数据;

(4) 渲染数据.

其中, 装扮集合存储了角色当前装备的装扮. 骨架、Avatar、基因数据和染料数据分别从种族的骨架描述、基因描述、染料描述中创建得到.

1.4 基因

和现实生活中生物的特性由基因决定类似, AAC 系统中的基因特指那些能够影响角色参数的属性. 例如: 命名为身高的基因影响角色的身高, 命名为肥胖程度的基因影响角色的肚子、手臂和脸等部位. 基因作用器则负责产生这些影响.

换言之, 基因负责定义角色有哪些受影响因素, 基因作用器则定义了这些影响因素如何影响角色. 之所以分别定义基因和基因作用器, 原因在于:

(1) 基因影响的功能十分丰富, 如: 影响骨骼、影响染料颜色、给角色装卸装扮等. 为了支持这种灵活性, AAC 系统定义了不同类型的作用器以匹配相对应的功能;

(2) 某些基因的改变需要多个基因作用器共同作用才能实现. 最直观的例子, 肥胖程度基因需要修改肚子、手臂、脸等多个部位, 分别对应着多个不同的基因作用器.

图 2 展示了基因和基因作用器的工作原理. 当角色从种族中创建时, 会读取并包装基因, 成为基因数据. 当基因数据中的数值发生改变时, AAC 系统会将与此基因相关的基因作用器应用至角色身上.

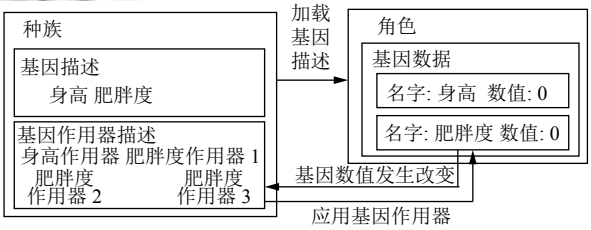


图 2 基因和基因作用器工作原理图

1.5 染料

AAC 系统定义了染料的概念以实现换色功能. 和基因的定义原理类似, 染料指那些能够改变装扮颜色的名称. 例如, 上衣的装扮中有上衣颜色染料, 通过修改染料, 可以修改上衣装扮的颜色.

换色的原理本质上与图 1 布局资源中多组上下文合并的原理是一样的. 区别在于, 此时第二套上下文中

的贴图是纯色的, 而非“纹身”.

具体实现方式为: 当角色从种族创建时, 会读取并包装其为染料数据. 增加两个颜色参数 (乘性因子 *Multiple* 和加性因子 *Additive*), 分别作为乘法因子乘以贴图颜色和作为加法因子加在贴图颜色上, 以获取换色后的目标颜色, 见式 (1).

$$Col_d = Col_s * Multiple + Additive \tag{1}$$

其中, Col_d为目标颜色, Col_s为原贴图颜色.

综上, AAC 系统概要设计图如图 3 所示. 由图 3 可见, 基于 Unity3D 的编辑器扩展技术, AAC 插件通过提供图形化界面和扩展工具, 使零编程基础的使用者也能轻松上手, 完成自定义游戏角色的换装功能.

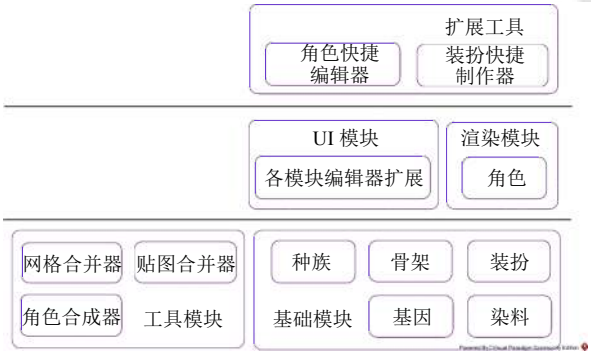


图 3 AAC 插件系统概要设计图

2 系统实现关键技术

2.1 基于切割扩展的贴图合并算法

AAC 插件以装扮为单位进行角色的合成, 包括骨骼合成、网格合成和贴图合成. 贴图的合成结果称为贴图集, 又名纹理集. 尽管 Unity3D 平台提供了纹理集合成的接口函数, AAC 系统却无法调用. 原因在于, 角色合成过程中需要用到染料, 而 Unity3D 并不支持这一 AAC 系统自定义概念. 因此, 需要自行设计并实现贴图合并算法.

贴图集生成是计算机图形学中提高计算机生成贴图利用率的技术[9]. 在贴图集生成、优化方面, 国内外已有大量的研究工作. 例如, Maillot 等[10] 提出交互式纹理映射方法, Lévy 等[11] 提出基于最小二乘保角映射的方法, Purnomo 等 [12] 提出无缝纹理图集的概念, 宋歌等[13] 提出基于动态空间合并的算法, 詹勇[14] 提出针对重复纹理的合并算法, 戴雪峰等[15] 提出基于贪心算法和退火算法的混合算法.论文在现有研究成果的基础上, 针对 AAC 系统对时间空间敏感、游戏模型贴图尺寸通常等于 2 的整数次幂的特点, 实现了一种基于切割扩展的贴图集合并算法, 主要包含两个过程:

(1) 切割过程: 以空闲空间 (Free Room, FR) 表示贴图集中的空白区域; 向其插入一张贴图将会切割得到若干个更小的 FR; 不断地将贴图插入这些 FR 并进行合并 (见图 4(a) 至图 4(c));

(2) 扩展过程: 当贴图较大, 无法插入任一现有 FR 时, 放入更大的 FR 进行扩展 (见图 4(d)).

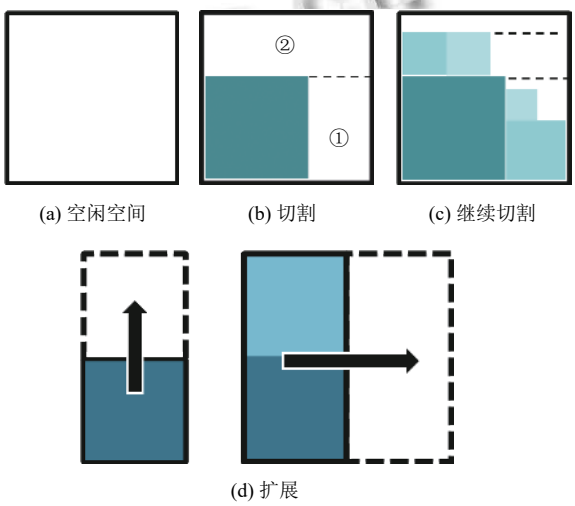


图 4 AAC 贴图集生成、切割与扩展示意图

AAC 系统贴图合并算法的关键代码如代码 1 所示. 其中, FRL 代表空闲空间链表, S_{tex}代表待合并的贴图集合, S_{rect}代表贴图集上的矩形集合, CAW 和 CAH 分别代表当前图集的宽度和高度.

2.2 编辑器扩展

为了使美工和程序开发人员都能方便地使用, AAC 系统设计了图形化操作界面, 使用者无需编写代码即可调用相关 API. 通过 Unity3D 平台提供的编辑器扩展功能, 用户甚至能制作出无需编写代码就能开发游戏的开发套件[16]. 此外, Unity3D 开源了 C#引擎和编辑器代码[17], 二者均为实现编辑器扩展的利器.

具体来说, 与图形界面软件开发原理类似, AAC 系统通过 Unity3D 提供的绘制组件接口, 在窗口内逐个绘制组件, 并响应用户输入, 以及执行不同的逻辑操作, 包括:

(1) 装扮制作: 提供装扮快捷制作扩展工具;

(2) 种族制作: 提供装扮类型编辑器、基因编辑器、基因作用编辑器、染料编辑器;

(3) 角色快速编辑扩展工具: 包括修改角色种族、装卸装扮、修改基因、修改染料、修改纹理集大小、序列化和反序列化、保存为预制件等.

代码 1. AAC 系统贴图合并算法关键代码

```
开始
//初始化 FRL, 放入适当大小的 FR
Init FRL with a appropriate_size_FR
// 设置 CAW 和 CAH 与此 FR 的宽高一致
CAW = appropriate_size_FR.w
CAH = appropriate_size_FR.h
// 对S_tex按照面积从大到小排序
Sort S_tex by area from largest to smallest
For tex in S_tex
// 循环直到从 FRL 中找到合适的 FR
While( !FindFirstBestFit( tex, out FR) )
// 若没找到, 则进行扩展
Expand()
// 向 S_r 中添加此贴图所处的矩形
S_r.Add( Rect(FR.x,FR.y,tex.w tex.h))
// 切割此 FR, 并添加进 FRL 中
FRL.Add( FR.Sqlit( tex ) )
return S_r, CAW, CAH
结束
```

图 5 展示了 AAC 系统编辑器扩展示例图, 包括: 槽位资源编辑器、布局资源编辑器、装扮资源编辑器、基因作用器编辑器、燃料编辑器、装扮类型编辑器、基因描述编辑器、种族编辑器、装扮制作扩展工具和角色快速编辑扩展工具.

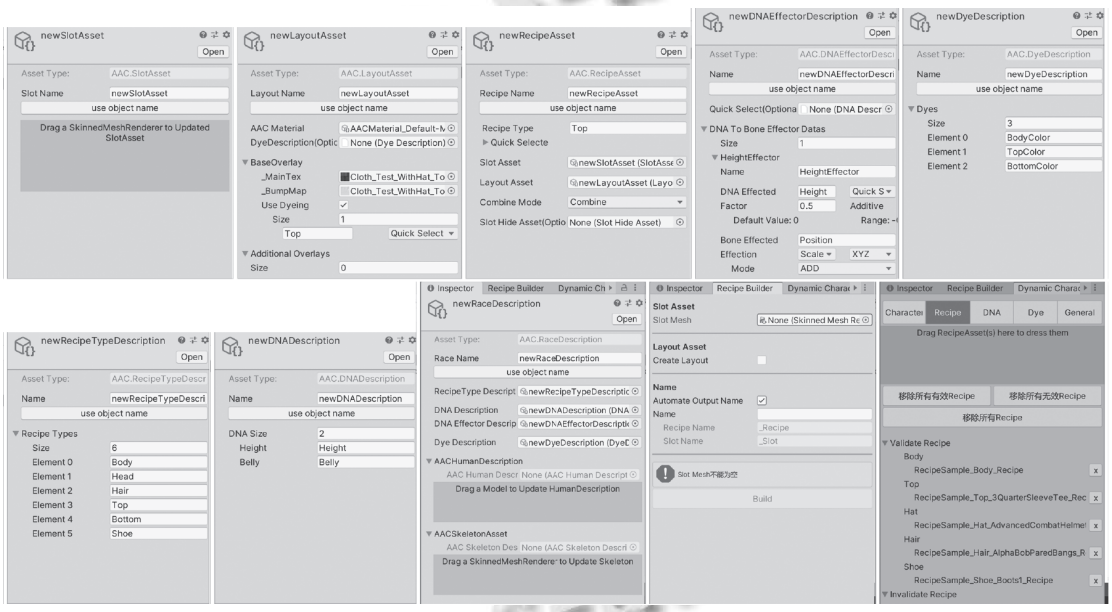


图 5 AAC 编辑器扩展图

2.3 预制件优化

AAC 换装系统中, 每个游戏角色是独立的. 因此, 系统会为每个角色分配独立的内存. 但是, 假如两个角色的数据完全相同, 例如: 拥有相同的装扮、基因和染料, 系统仍然得为它们维护两份独立内存, 从而产生数据冗余.

传统方法利用 Unity3D 提供的预制件技术来减少这部分数据冗余. 预制件是一种可重复利用的游戏对象资源类型[18], 所有从预制件实例化得到的游戏对象都有与预制件相同的对象数据. 换言之, 若将游戏角色制作为预制件, 通过预制件实例化, 这些对象实例将共享相同的网格和纹理集, 从而减少冗余数据 (原理见图 6).

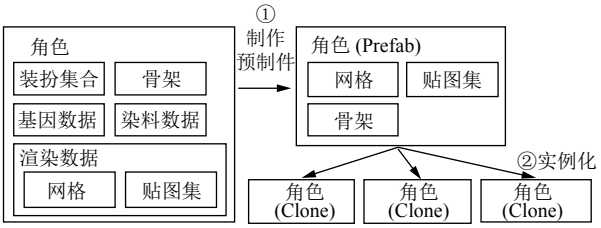


图 6 Unity 3D 预制件技术原理图

然而, 由图 6 可见, 基于 Unity3D 的预制件方法在制作过程中仅保留了渲染数据中的网格、贴图集和骨架信息, 丢失了装扮、基因数据、染料数据和种族信息. 也就是说, 该方法不具备换装功能.

为了既减少空间开销又保留换装功能, AAC 系统

设计了保留换装功能的预制件, 具体实现方式如下:

(1) 生成预制件的同时, 将角色的种族及附属资源一并写入预制件中;

(2) 当从此预制件实例化时, 将会加载这些资源, 从而获得所有的系统功能.

实际应用时, 建议开发者通过以上方式制作一个类似“素体”且保留换装功能的预制件. 所谓“素体”, 是指只有基本装扮 (如躯干), 没有修改过基因和染料的角色. 在此开发模式下, 只有额外的装扮才会产生渲染数据, “素体”部分的空间占用是共享的. 因此, 与不使用 AAC 预制件优化技术的情形相比, 此方法将大幅度减少冗余空间, 提高换装速度.

3 示例程序与性能分析

3.1 换装示例程序

我们设计和开发了一个能调用 AAC 插件的换装示例程序以展示换装效果. 程序建模工作流包括: (1) 制作模型; (2) 在线绑定骨骼和蒙皮; (3) 对模型进行加工处理.

由图 7 可见, 添加 AAC 插件后, 换装示例程序具备了捏脸、换装、换色等功能.其中, 捏脸功能不局限于脸部基因的改变, 也包括身体各部位的变化; 换色功能既包括装扮 (如帽子、上衣等) 颜色的改变, 也包括基因 (如肤色、眼球) 颜色的变化.

3.2 性能比较与分析

3.2.1 时间开销

本节对不同装扮数量下 AAC 与 UMA 系统的时间开销进行比较 (见表 2), 结论如下:

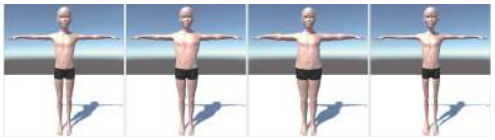
(1) 对于不同的装扮数量, 游戏角色进行换装合成所消耗的时间是不同的. 无论哪种系统, 时间开销均随装扮数量的增长呈正比例增长.

(2) 大部分情况下 (更换装扮数量少于 7 件), AAC 系统比 UMA 系统的时间开销小; 当装扮数量接近上限, AAC 系统性能与 UMA 系统性能持平.

3.2.2 空间开销

本节比较的是随机装扮数量下 AAC 与 UMA 系统的空间性能 (见表 3). 实验条件: 创建相同数量、随机装扮的角色; 观察对象: 创建前后系统内存的使用情况.

需要说明的是, 由于网格和贴图集占用内存的比例最高, 因此仅以网格和贴图的内存空间占用量作为代表进行比较. 本实验中, 网格顶点数不少于 25 000 个, 贴图集尺寸约为 2–4 KB.



(a) 捏脸

(b) 换装

(c) 换色

图 7 AAC 插件换装功能展示

表 2 不同换装系统、不同装扮数量情况下时间开销比较 (单位: ms)

名称	0	1	2	3	4	5	6	7
UMA	3.78	4.36	6.37	5.06	6.92	7.03	7.83	**9.35**
AAC	**0.26**	**2.64**	**4.06**	**4.34**	**5.76**	**6.29**	**7.55**	10.01

表 3 不同换装系统空间开销比较 (单位: MB)

名称	网格	贴图	合计
UMA	5.17	85.86	91.03
AAC	**2.11**	**22.27**	**24.38**

实验结论如下:

(1) 相同内存占用下, AAC 系统容纳角色的数量接近 UMA 的 4 倍.

(2) 相比 UMA 系统, AAC 系统的时间空间复杂度更低, 更适合用于运行时的角色动态生成任务.

4 结论与展望

本文设计并实现了一个基于 Unity3D 平台的换装系统插件: 方舟角色创建器, 简称 AAC 系统. 该系统具备游戏角色换装、捏脸、换色等功能. 通过编辑器扩

展技术, 该插件可提供图形化操作界面, 令使用者无需具备编程基础也能轻松掌握. 再结合预制件优化技术, AAC 系统在完整保留系统换装功能的同时, 还可进一步提高系统的时间和空间利用率, 使换装系统适用于运行时角色动态生成的应用场景.

需要指出的是, AAC 系统也存在一些不足, 如: (1) 与 UMA 系统相比, 缺少高级功能; (2) 未设计资源管理组件. 进一步完善 AAC 插件功能, 将是未来的研究工作重点.

参考文献

1 马红亮. 电子游戏的教育价值: 来自美国研究的新观点. 开放教育研究, 2009, 15(1): 105–109. [doi: 10.3969/j.issn.1007-2179.2009.01.018]

2 叠纸网络. 闪耀暖暖官网. https://nikki4.papegames.cn/home?utm_source=official&utm_medium=home_nav.[2020-08-06].

3 百度百科. 模拟人生. https://baike.baidu.com/item/%E6%A8%A1%E6%8B%9F%E4%BA%BA%E7%94%9F/276335?fr=aladdin. [2020-08-06].

4 暴雪娱乐. 魔兽世界. https://wow.blizzard.cn/landing. [2020-08-06].

5 杨静. 基于 HCT VIVE 的虚拟换装游戏的设计与实现. 信息技术与信息化, 2018, 43(10): 58–59, 62.

6 李俊, 张明敏, 潘志庚. 人物替换模式的虚拟试衣. 计算机辅助设计与图形学学报, 2015, 27(9): 1694–1700. [doi: 10.3969/j.issn.1003-9775.2015.09.013]

7 徐康熙, 郝泳涛. 基于物理引擎 PhysX 的 3D 试衣系统的设计与实现. 电脑知识与技术, 2014, 10(8): 1826–1829, 1837.

8 引擎鉴赏小组. 关于 Unity. https://unity.cn/projects/about-unity. [2020-08-08].

9 程杰. 大话设计模式. 北京: 清华大学出版社, 2007. 30–33.

10 Maillot J, Yahia H, Verroust A. Interactive texture mapping. Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques. Anaheim, CA, USA. 1993. 27–34.

11 Lévy B, Petitjean S, Ray N, *et al*. Least squares conformal maps for automatic texture atlas generation. ACM Transactions on Graphics, 2002, 21(3): 362–371. [doi: 10.1145/566654.566590]

12 Purnomo B, Cohen JD, Kumar S. Seamless texture atlases. Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing. Nice, France. 2004. 65–74.

13 宋歌, 杨红雨. 基于纹理集的大规模场景模型优化算法. 计算机工程与设计, 2011, 32(9): 3119–3122.

14 詹勇. 三维城市模型重复纹理合并方法研究. 城市勘测, 2015, 30(6): 17–20.

15 戴雪峰, 熊汉江, 龚健雅. 一种三维城市模型多纹理自动合并方法. 武汉大学学报 (信息科学版), 2015, 40(3): 347–352, 411.

16 hutong games. PlayMarker. https://hutonggames.com/index.html. [2020-08-10].

17 Unity-Technologies. UnityCsReference. https://github.com/Unity-Technologies/UnityCsReference. [2020-08-10].

18 陈嘉栋. Unity3D 脚本编程: 使用 C#语言开发跨平台游戏. 北京: 电子工业出版社, 2016. 309–310.