**深圳市科技创新委员会科技计划**

**技术创新项目**

**实**

**施**

**总**

**结**

**报**

**告**

项目名称：新一代国产移动3D游戏引擎关键技术研究

编制单位：深圳市创梦天地科技股份有限公司

**一、项目科技报告**

**公 开**

**新一代国产移动3D游戏引擎关键技术研究项目科技报告**

**实施单位:深圳市创梦天地科技有限公司**

**项目负责人：关嵩, 連冠榮**

**项目完成日期：2017-8-31**

**摘 要**

自从移动互联网和设备兴起和普及化，也带动了软件的爆炸式增长，游戏软件也是其中之一。游戏是一个涉及多个技术各领域的行业（如音效，物理模拟及碰撞检察，2D/3D 图形渲染，粒子系统，动画系统，资源管理，资源格式转换，输入，人工智能模块，服务器引擎，整合式游戏开发环境等），大部游戏开发者是基于中间件来加快开发速度，而＂游戏引擎＂则是这个中间件的行业专称。本来，在手机游戏产业链中，引擎只是并不起眼的底层技术体系，但随着手机游戏市场的井喷繁荣，引擎已经变得日益重要。

目前，占有率领先移动游戏引擎主要有Cocos2d-x、Unity3D、Unreal等。不同的统计方给出的数据各有差异，但总体来讲，行业目前首选的游戏引擎，主要都集中在Cocos2d-x 与Unity3D 之上。然而，目前国内游戏产业基本都是借用国外核心技术，一直都没有自主研发一套完善的整合式游戏开发环境及针对国内用户设备的3D 引擎。本项目的立项主要是为了开发出一款针对国内开发者和用户的游戏开发程序库及工具（或中间件：MIDDLEWARE）。在3D 图形，资源优化，支援将来移动设备有所突破，并透过开放式软件架构（非开源）让有志从事上述各技术领域，业界开发者用户们，去拓展自己独有的引擎功能，借此提高深圳乃至全国人才对核心技术的参与和共同进步，提高国内游戏各个技术领域的研究。

**目 录**

[（一）引言部分 7](#_Toc485220384)

[**1、背景** 7](#_Toc485220385)

[**2、意义** 8](#_Toc485220386)

[**3、范围、对象及目的** 8](#_Toc485220387)

[**4、已有工作情况** 9](#_Toc485220388)

[（二）主体部分 10](#_Toc485220389)

[**1、研究内容** 10](#_Toc485220390)

[**2、工作原理** 11](#_Toc485220391)

[**3、研究方法** 17](#_Toc485220392)

[（三）结论部分 30](#_Toc485220393)

[**1、实现的功能** 30](#_Toc485220394)

[**2、技术创新点** 31](#_Toc485220395)

[（四）建议部分 31](#_Toc485220396)

[**1、总结** 31](#_Toc485220397)

[**2、展望** 31](#_Toc485220398)

[（五）参考文献 32](#_Toc485220399)

**（一）引言部分**

**1、背景**

3D引擎作为一个名词已经存在了很多年，但即使是一些专业的引擎设计师，也很难就它的定义达成一个共识。通常来说，3D引擎作为一种底层工具支持着高层的图形软件开发。你可以把它看成是对3D API的封装，对一些图形通用算法的封装，对一些底层工具的封装。

3D引擎最基本的功能包括：①数据管理：这里的数据管理是一个比较广泛的定义，不同的3D引擎也许会拥有其中一个或多个功能。这些功能包括：场景管理，对象系统，序列化，数据与外部工具的交互，底层3维数据的组织和表示。②渲染器：之所以要说是合理的渲染器，是因为一个引擎的渲染能力是由多方面决定的。比如一款以实时游戏作为目标的游戏，会选择基于光栅化的渲染算法。在这种设计前提下，几何体一级的数据不会过于详细，例如物体表面的BRDF，折射率，纹理坐标空间的变化率，切线空间的变化率（当然随着硬件能力的提升和Shader能力的发展，这些数据也会出现在一些比较高级的游戏引擎中），这时候即使你在设计初期就考虑到这些数据需求，并将它们表现在了Render中，最后也不会有任何意义。③交互能力：简单的说，就是开发工具。任何一款3D引擎如果没有开发工具都不能称为是完整的。这些开发工具可能是一些文件转换器，场景编辑器，脚本编辑器，粒子编辑器。

随着3D技术的发展，特别是游戏产业的广泛应用，使得游戏3D引擎在游戏产业发展中的地位越来越显得重要而突出，许多大公司都将引擎视为公司的独门武器，是公司强大开发实力的有力印证，而随着用于对手游画面及质感要求的不断提升，3D引擎技术极可能引爆手游发展的另一个高潮。

本文将主要针对游戏3D引擎技术介绍和描述我们的3D引擎架构及实现方案。

**2、意义**

随着手游产业的迅猛发展，目前我国游戏市场规模已超日本成为全球第二，其未来发展态势值得我们高度关注。报告显示，2016年，中国游戏市场实际销售收入达到1655.7亿元，同比增长17.7%；其中移动游戏市场实际销售收入为819.2亿元，同比增长59.2%，市场占比超过客户端游戏市场达到49.5%，成为份额最大、增速最快的细分市场。2016年，中国游戏用户规模达到5.66亿人，同比增长5.9%，增长率小幅上升。然而事实上我国手游产业研发技术水平还有待提升，特别是近年来高速发展的中重度手游，对3D引擎技术的要求越来越高，能否高速使用3D引擎技术开发高质量的手游产品已经成为检验手游企业能否在市场中快速高效取得市场份额的有效武器。

出于节约成本、缩短周期和降低风险三方面的考虑，游戏开发者对游戏引擎的需求却是越来越旺盛，但是目前我国市场上还缺乏一个国人自主研发，完善的整合式游戏开发环境；针对国内用户设备；低成本让开发者参与拓展及支援微端应用的3D引擎。本项目将通过该项目的自主研发提高国内3D游戏产业的发展，特别是缩短周期和减低成本，借此提高国内人才对核心技术的参与和共同进步，提高国内游戏各技术领域研究。

**3、范围、对象及目的**

本项目“新一代国产移动3D游戏引擎关键技术研究”通过整合游戏开发环境，使设计、技术、美术及各工种协作无缝一体化，同时对3D模型和动画档案大小及内存专项优化，设定专属的渲染语言来解决跨平台渲染语言问题，进行创新的3D预见性场景广利，透过开放式插件SDK方式，让用户新增引擎核心功能。总而言之，创梦天地移动游戏3D引擎完善的整合了开发者环境，针对国内用户设备，低成本让开发者参与拓展及支援微端应用。

**4、已有工作情况**

曾经有一段时期，游戏开发者关心的只是如何尽量多地开发出新的游戏并把它们推销给玩家。尽管那时的游戏大多简单粗糙，但每款游戏平均开发周期也要达到8到10个月以上，这一方面是由于技术的原因，另一方面则是因为几乎每款游戏都要从头编写代码，造成了大量的重复劳动。渐渐地，一些有经验的开发者摸索出了一种方法，他们借用上一款类似题材的游戏中的部分代码作为新游戏的基本框架，以节省开发时间和开发费用，引擎的概念就是在这种机器化作业的背景下诞生的。

游戏引擎技术在国外起步早发展也比较成熟，目前国外最有名的游戏引擎技术是UNITY3D，而国内的游戏引擎由于起步较晚目前发展的速度和质量还有待提升，目前国内常用的游戏引擎技术是COCOS2D-X，它们分别对应了3D和2D的市场，而它们都各有各优缺点。

国内移动数据收费比国外高，所以中国市场对应用包大小有特别要求；国内移动高端用户设备性能及得上国外，但相比国外起来仍还有一大部份中低端用户群。

国内移动数据收费比国外高,所以中国市场对应用包大小有特别要求;国内移动高端用户设备性能及得上国外,但相比国外起来仍还有一大部份中低端用户群。

国外有UNITY3D引擎,它是一个一次性付费非开源引擎。它占据全功能游戏引擎市场45%的份额,全球用户330万人,每月活跃用户数60万,它的功能强大,比如说完善的整合式游戏开发环境,3D图形,引擎光是本身(不计算游戏代码资源)占用太多资源(7-8MB右),这对开发者的市场复盖率有一定影响(因为中低端用户未有被触到)。UNITY3D也不支持开发者自定义插件。这使得开发者只能用UNITY3D自带功能,当他们想更深度开发自定义功能时,不得不花费巨额资金去买整体UNITY3D代码和投入人力去理解这个覆杂的引擎。

而国内的COCOS2D-X,它是一个免费开源引擎,它占全球市场份额中超过25%,是基于外国Ricardo Quesada的Cocos2D-iPhone,它有基本2D图形功能,没有3D图形功能,也没有整合式游戏开发环境,使各游戏开发各工种协作碎片分散,工作效率降低。而刚出来不久的COCOS3D,虽然有简单的3D功能,但跟UNITY3D比较,不论功能,模型档案大小及内存还有优化空间。

按照以往非移动互联网的经验,应用程序由大容量应用过度到微端应用,即核心代码一早就在客户端设备(或第一次下载时安装),之后用户只需下载小部份资源及代码便可以马上应用,而其余资料则在用户在用的时间,以串流方式引入。

**（二）主体部分**

**1、研究内容**

本项目开发出一款针对国内开发者和用户的游戏开发程序库及工具(或中间件:MIDDLEWARE)。并且在3D图形,资源优化有所突破,该项目的主要研究内容如下：

①面向实体(ENTITY)及组件(COMPONENTS)模式游戏引擎架构

②可扩展组件(COMPONENTS)开放式插件引擎架构技术及其SDK

③３Ｄ骨格模型渲染,３Ｄ骨格动画

⑤跨平台引擎专属渲染语言规格及编輯工具(支援HLSL, GLSL)

⑥低流量贴图格式及转换工具

⑦低流量３Ｄ骨格模型格式,３Ｄ骨格动画格式

⑨基本的２Ｄ骨格模型,２Ｄ骨格动画

**2、工作原理**

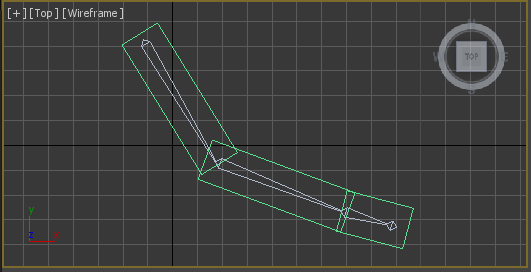
**2.1) 2D動骨骼動畫壓縮**

**技術方案的详细阐述**

**2.1.1** 2D骨骼動畫壓縮格式**技術**, 它可以以較少的內存及存儲空間來存儲2D骨骼動畫數據。因為本技術不需使用硬件解壓, 所以它是通用的性的方案。

2.1.2關鍵字

**2D骨骼動畫**: 一種由骨骼組成的2D動畫數據結構.



**骨骼**: 代表了動畫某一節骨骼的狀態, 包括位移,旋轉,縮放數據

**位移**: 一個2維向量, 表示骨骼的位置

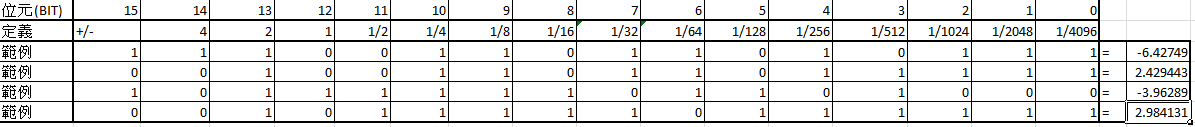
**旋轉**: 一個小數, 表示骨骼的旋轉角度

**縮放**: 一個2維向量, 表示骨骼的縮放比倍

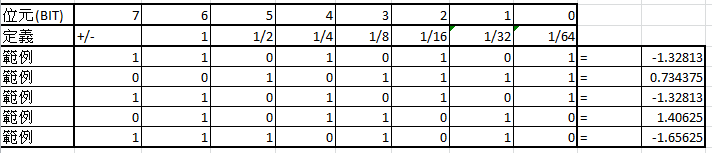
**父節點**: 因為骨骼是一個有層次的結構, 每一個骨骼都有他的父節點骨骼.

父節點的位移, 旋轉, 縮放數據都會继承到子節點

* + 1. **定點數**（FIX POINT）：計算機一般都由浮點數(比如說ANSI C 的float)來表達小數，浮點數數值範圍大,精度高，佔32bit。在某些應用場合，小數也可以由精度較小的定點數（FIX POINT）來表示。常見有fix8, fix16, 他們定義如下：  
       fix16

****

fix8

****

* + 1. 此檔案格式技術原理

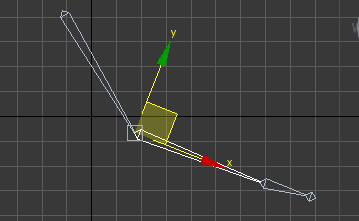
2D骨骼動畫是由關鍵幀組成, 每一個關鍵幀（Key frame），都由位移（Translation）, 旋轉（Rotation）,縮放（Scale）組成。當以下條件成立, 我們可以利用數據對關鍵幀壓縮 (此條件在動物骨骼動畫經常發生)

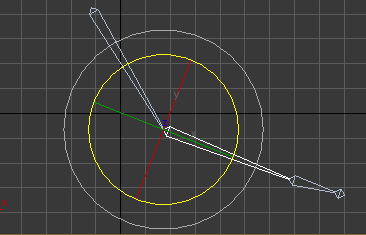
**位移（Translation）：**如子節點沒有相對於父節點有位移或子節點的子節點數值不跟時間變化， 只需它只須存放第一幀的數據即可, 其余幀的數據不必保存。

**旋轉（Rotation）：**因為是一個小數，而且數值範圍在(0, 360)之間，本來1個float的數據可以用1個fix16來表示，本來是1\*4bytes=4Bytes,壓縮後是1 \* 2 = 2bytes

**縮放（Scale）** ：如果縮放少於7, 跟旋轉（Rotation）原理一樣, 本來3個float的數據可以用3個fix16來表示. 本來是3\*4bytes=12Bytes, 壓縮後是3 \* 2 =6bytes

**縮放（Scale）：**另外, 如果縮放比不跟時間變化，只須保留第一幀數據, 其余幀的數據不必保存。







* + 1. 此檔案格式有３大部份
* 頭文檔(Header)
* 骨骼層次結構(Bones Hierarchy)
* 骨骼關鍵幀(KEYFRAMES)
  + 1. 以下是檔案格式示意圖：



其中，頭文檔(Header)各數據的定義如下：

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **中文名稱** | **Bytes(位元數)** | **Notes(備注)** |
| ID | 文件格式標示 | 4 | 0x49 0x44 0x53 0x55 |
| Copyright | 文件著作權 | 32 |  |
| Author | 文件作者 | 16 |  |
| Format | 格式 | 4 | Position Compressed：0x01  Rotation Compressed：0x02  Scale Compressed：0x04 |
| NumofBones | 骨骼數目 | 4 |  |
| FPS | 幀率 | 4 |  |

其中，骨骼層次結構(Bones Hierarchy)各數據的定義如下：

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **中文名稱** | **Bytes(位元數)** | **Notes(備注)** |
| Bone0: Name | Bone0的名字 | 20 |  |
| Bone1: Name | Bone1的名字 | 20 |  |
| … | … | … | … |
| BoneN: Name | BoneN的名字 | 20 |  |
| Bone0: ParentIndex | Bone0的父節點索引 | 2 |  |
| Bone1: ParentIndex | Bone1的父節點索引 | 2 |  |
| … | … | … | … |
| BoneN: ParentIndex | BoneN的父節點索引 | 2 |  |

其中，關鍵幀(KEYFRAMES)各數據的定義如下：

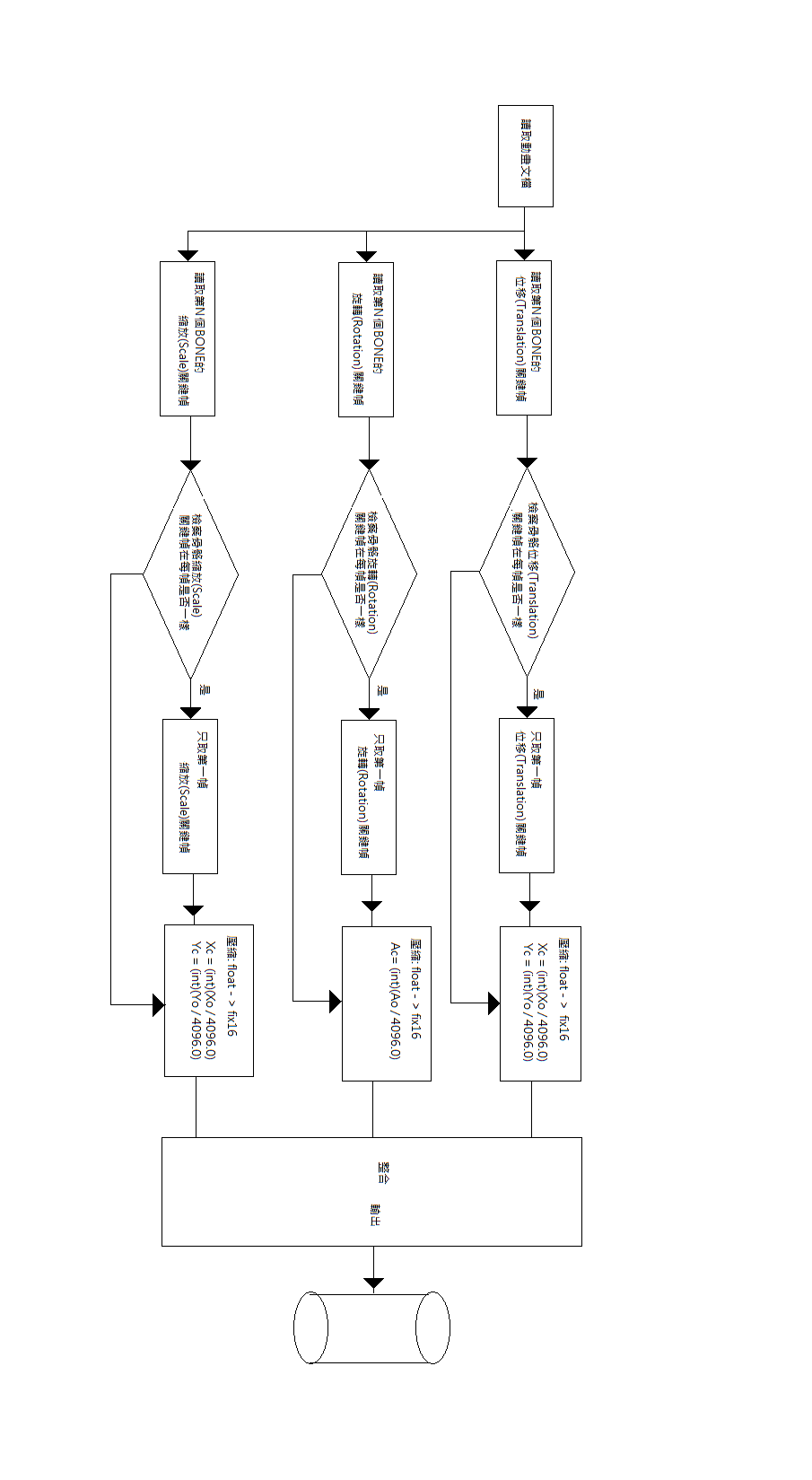
|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **中文名稱** | **Bytes(位元數)** | **Notes(備注)** |
| NumOfPositionKey | 位移關鍵幀數 | 2 |  |
| NumOfRotationKey | 旋轉關鍵幀數 | 2 |  |
| NumOfScaleKey | 縮放關鍵幀數 | 2 |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Bone0: Time = 0  Translation | Bone0在Time=0的位移關鍵幀 | 未壓縮:4\*2  壓縮:2\*2 |  |
| Bone0: Time = 0  Rotation | Bone0在Time=0的旋轉關鍵幀 | 未壓縮:4\*1  壓縮:2\*1 |  |
| Bone0: Time = 0 Scale | Bone0在Time=0的縮放關鍵幀 | 未壓縮:4\*2  壓縮:2\*2 |  |
| Bone0: Time = T  Translation | Bone0在Time=T的位移關鍵幀 | 未壓縮:4\*2  壓縮:2\*2 |  |
| Bone0: Time = T  Rotation | Bone0在Time=T的旋轉關鍵幀 | 未壓縮:4\*1  壓縮:2\*1 |  |
| Bone0: Time = T Scale | Bone0在Time=T的縮放關鍵幀 | 未壓縮:4\*2  壓縮:2\*2 |  |
| … | … | … | … |
| Bone1: Time = 0  Translation | Bone1在Time=0的位移關鍵幀 | 未壓縮:4\*2  壓縮:2\*2 |  |
| Bone1: Time = 0  Rotation | Bone1在Time=0的旋轉關鍵幀 | 未壓縮:4\*1  壓縮:2\*1 |  |
| Bone1: Time = 0 Scale | Bone1在Time=0的縮放關鍵幀 | 未壓縮:4\*2  壓縮:2\*2 |  |
| Bone1: Time = T  Translation | Bone1在Time=T的位移關鍵幀 | 未壓縮:4\*2  壓縮:2\*2 |  |
| Bone1: Time = T  Rotation | Bone1在Time=T的旋轉關鍵幀 | 未壓縮:4\*1  壓縮:2\*1 |  |
| Bone1: Time = T Scale | Bone1在Time=T的縮放關鍵幀 | 未壓縮:4\*2  壓縮:2\*2 |  |
| … | … | … |  |
| BoneN: Time = 0  Translation | BoneN在Time=0的位移關鍵幀 | 未壓縮:4\*2  壓縮:2\*2 |  |
| BoneN: Time = 0  Rotation | BoneN在Time=0的旋轉關鍵幀 | 未壓縮:4\*1  壓縮:2\*1 |  |
| BoneN: Time = 0 Scale | BoneN在Time=0的縮放關鍵幀 | 未壓縮:4\*2  壓縮:2\*2 |  |
| BoneN: Time = T  Translation | BoneN在Time=T的位移關鍵幀 | 未壓縮:4\*2  壓縮:2\*2 |  |
| BoneN: Time = T  Rotation | BoneN在Time=T的旋轉關鍵幀 | 未壓縮:4\*1  壓縮:2\*1 |  |
| BoneN: Time = T Scale | BoneN在Time=T的縮放關鍵幀 | 未壓縮:4\*2  壓縮:2\*2 |  |
| … | … | … |  |

其中特別注明一下, 本檔案格式的生成方法

* 當格式(format) 標示出有使用位移關鍵幀壓縮(Position Compressed)時, 位移關鍵幀數據使用2個fix16來儲存, 否則使用2個float格式來儲存.
* 當格式(format) 標示出有使用旋轉關鍵幀壓縮(Rotation Compressed)時, 旋轉關鍵幀數據使用1個fix16來儲存, 否則使用1個float格式來儲存.
* 當格式(format) 標示出有使用縮放關鍵幀壓縮(Scale Compressed)時, 縮放關鍵幀數據使用2個fix16來儲存, 否則使用2個32bits格式來儲存.
* 因為每個骨骼(Bone) 位移關鍵幀, 旋轉關鍵幀, 縮放關鍵幀, 都分別有標明

位移關鍵幀數(NumOfPositionKey), 旋轉關鍵幀數(NumOfRotationKey), 縮放關鍵幀數(NumOfScaleKey), 所以, 每個骨骼(Bone) 的位移關鍵幀數目,旋轉關鍵幀數目, 縮放關鍵幀數目, 可以應用戶的壓縮要求而有所不同.比如說某一個骨骼只有1個位移關鍵幀, 10個旋轉關鍵幀, 1個縮放關鍵幀.



* + 1. **技术方案所产生的有益效果**

　　本技术方案描述一種實用新型，所产生的有益效果有：

* 讓2D骨骼動畫,在內存佔用及文件大小上都得到有效的壓縮，(即是說在無損質量的前題下，一般來說一個動畫100（Key frame）來說，本來4000bytes，壓縮後只要1420bytes，壓縮率30％)
* 在所有32BIT或以上的CPU加載此格式時不用對它進行解壓即可使用, 只需在運行時實時轉換。
* 骨骼(Bone) 的位移關鍵幀數目,旋轉關鍵幀數目, 縮放關鍵幀數目, 可以應用戶的壓縮要求而有所不同.比如說某一個骨骼只有1個位移關鍵幀, 10個旋轉關鍵幀, 1個縮放關鍵幀, 可針對位移
  + 1. **本方案已向國家知識產權局申請發明專利保护**
* 位移關鍵幀使用2個fix16 (比如說ANSI C的short [3])來儲存,
* 位移關鍵幀也可選擇性地使用3個float (比如說ANSI C的float [3])來儲存.
* 旋轉關鍵幀使用1個fix16格式(比如說ANSI C的short [1])來儲存,
* 旋轉關鍵幀也可選擇性地使用1個float格式(比如說ANSI C的float [1])來儲存.
* 縮放關鍵幀數據使用2個fix16格式(比如說ANSI C的short [3])來儲存,
* 縮放關鍵幀數也可使用2個float格式(比如說ANSI C的float [3])來儲存.
* 每個骨骼(Bone) 位移關鍵幀, 旋轉關鍵幀, 縮放關鍵幀, 都分別有標明

位移關鍵幀數(NumOfPositionKey), 旋轉關鍵幀數(NumOfRotationKey), 縮放關鍵幀數(NumOfScaleKey). 每個骨骼(Bone) 的位移關鍵幀數目,旋轉關鍵幀數目, 縮放關鍵幀數目, 可以應用戶的壓縮要求而有所不同.(比如說某一個骨骼只有1個位移關鍵幀, 10個旋轉關鍵幀, 1個縮放關鍵幀).

* 在所有32BIT或以上的CPU加載此格式時不用對它進行解壓即可使用, 只需在運行時實時轉換。

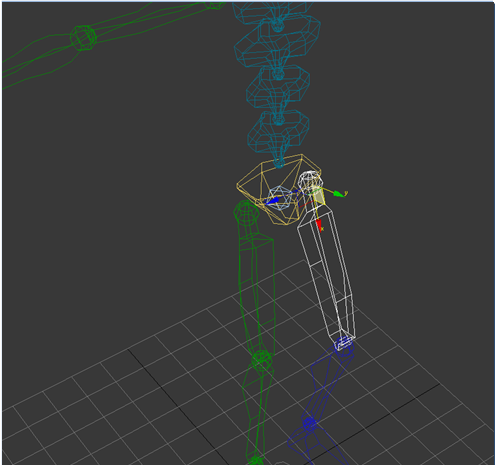
**2.2) 3D動骨骼動畫壓縮**

**2.2.1本技术方案的详细阐述**

本技术方案描述一種實用新型，它是一種3D骨骼動畫壓縮格式。 它可以以較少的內存及存儲空間來存儲3D骨骼動畫數據。因為本技術不需使用硬件解壓, 所以它是通用的性的方案。

* + 1. 關鍵字

**3D骨骼動畫**: 一種由骨骼組成的3D動畫數據結構.



**骨骼**: 代表了動畫某一節骨骼的狀態, 包括位移,旋轉,縮放數據

**位移**: 一個3維向量, 表示骨骼的位置

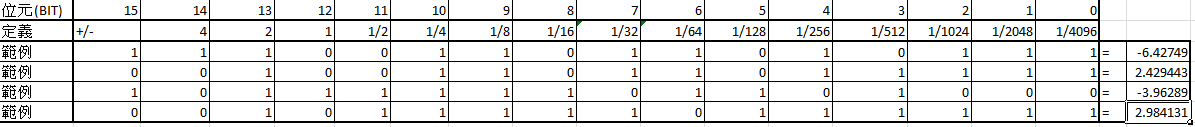
**旋轉**: 一個四元數(Quaternion), 表示骨骼的旋轉角度

**縮放**: 一個3維向量, 表示骨骼的縮放比倍

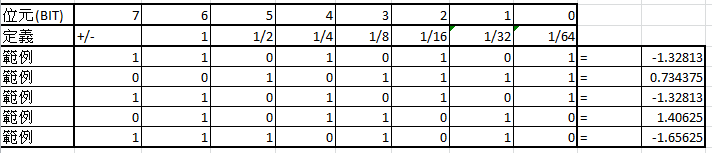
**父節點**: 因為骨骼是一個有層次的結構, 每一個骨骼都有他的父節點骨骼.

父節點的位移, 旋轉, 縮放數據都會继承到子節點

* + 1. **定點數**（FIX POINT）：計算機一般都由浮點數(比如說ANSI C 的float)來表達小數，浮點數數值範圍大,精度高，佔32bit。在某些應用場合，小數也可以由精度較小的定點數（FIX POINT）來表示。常見有fix8, fix16, 他們定義如下：  
       fix16

****

fix8

****

* + 1. 此檔案格式技術原理

3D骨骼動畫是由關鍵幀組成, 每一個關鍵幀（Key frame），都由位移（Translation）, 旋轉（Rotation）,縮放（Scale）組成。

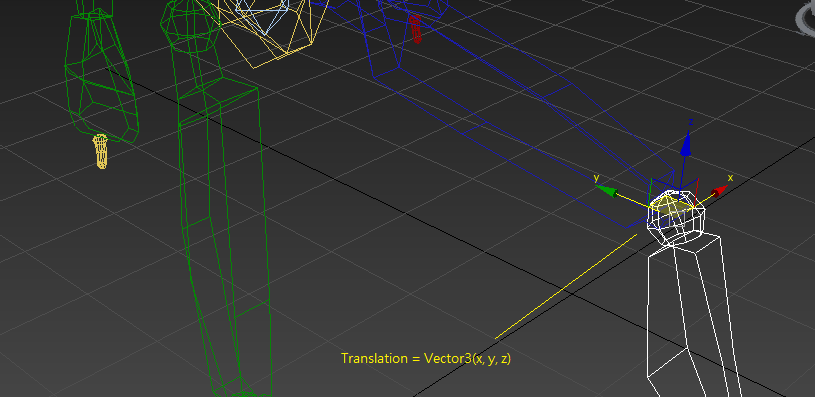
當以下條件成立, 我們可以利用數據對關鍵幀壓縮 (此條件在動物骨骼動畫經常發生)

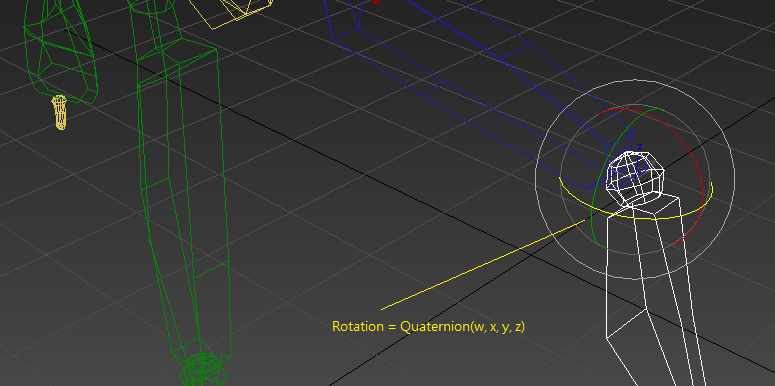
**位移（Translation）：**如子節點沒有相對於父節點有位移或子節點的子節點數值不跟時間變化， 只需它只須存放第一幀的數據即可, 其余幀的數據不必保存。

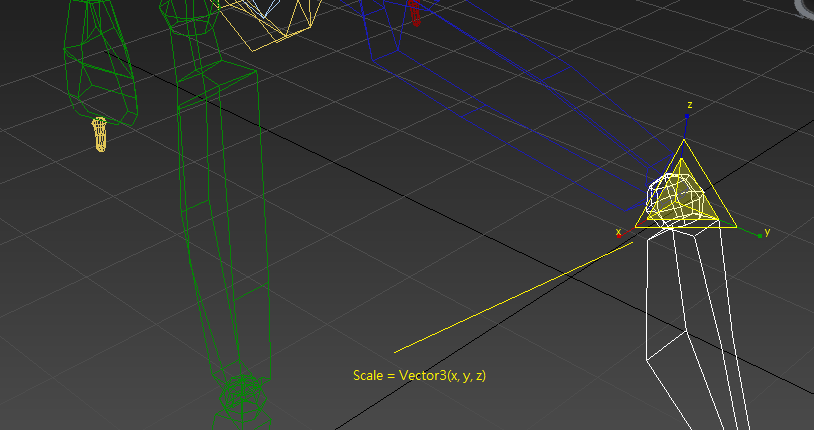
**旋轉（Rotation）：**因為是一個四元數(Quaternion)，而且數值範圍在(-1,1)之間，本來4個float的數據可以用4個fix8來表示，本來是4\*4bytes=16Bytes,壓縮後是4 \* 1 = 4bytes

**縮放（Scale）** ：如果縮放少於7, 跟旋轉（Rotation）原理一樣, 本來3個float的數據可以用3個fix16來表示. 本來是3\*4bytes=12Bytes, 壓縮後是3 \* 2 =6bytes

**縮放（Scale）：**另外, 如果縮放比不跟時間變化，只須保留第一幀數據, 其余幀的數據不必保存。







* + 1. 此檔案格式有３大部份
* 頭文檔(Header)
* 骨骼層次結構(Bones Hierarchy)
* 骨骼關鍵幀(KEYFRAMES)
  + 1. 以下是檔案格式示意圖：



其中，頭文檔(Header)各數據的定義如下：

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **中文名稱** | **Bytes(位元數)** | **Notes(備注)** |
| ID | 文件格式標示 | 4 | 0x49 0x44 0x53 0x55 |
| Copyright | 文件著作權 | 32 |  |
| Author | 文件作者 | 16 |  |
| Format | 格式 | 4 | Position Compressed：0x01  Rotation Compressed：0x02  Scale Compressed：0x04 |
| NumofBones | 骨骼數目 | 4 |  |
| FPS | 幀率 | 4 |  |

其中，骨骼層次結構(Bones Hierarchy)各數據的定義如下：

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **中文名稱** | **Bytes(位元數)** | **Notes(備注)** |
| Bone0: Name | Bone0的名字 | 20 |  |
| Bone1: Name | Bone1的名字 | 20 |  |
| Bone2: Name | Bone2的名字 | 20 |  |
| BoneN: Name | BoneN的名字 | 20 |  |
| Bone0: ParentIndex | Bone0的父節點索引 | 2 |  |
| Bone1: ParentIndex | Bone1的父節點索引 | 2 |  |
| Bone2: ParentIndex | Bone2的父節點索引 | 2 |  |
| BoneN: ParentIndex | BoneN的父節點索引 | 2 |  |

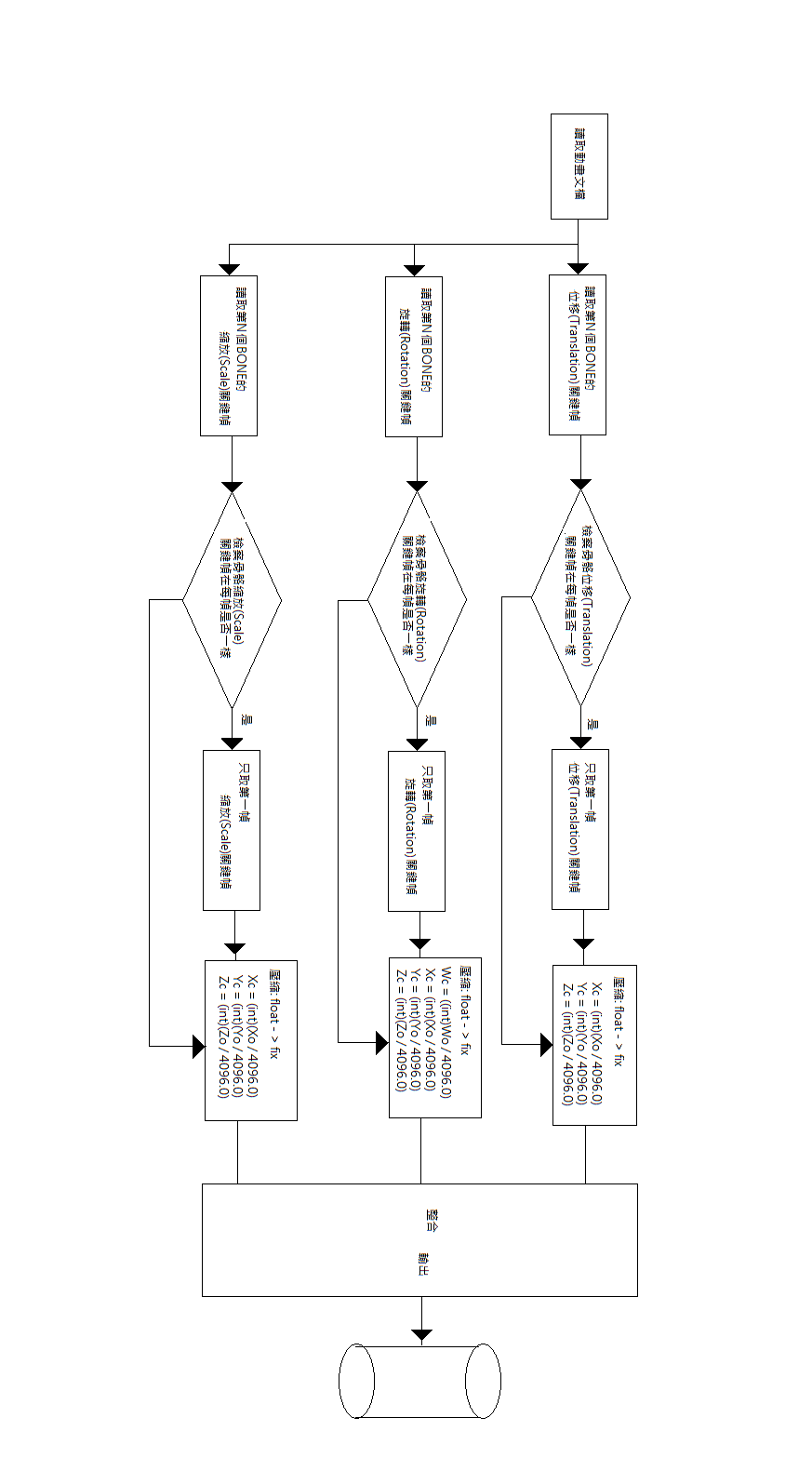
其中，關鍵幀(KEYFRAMES)各數據的定義如下：

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **中文名稱** | **Bytes(位元數)** | **Notes(備注)** |
| NumOfPositionKey | 位移關鍵幀數 | 2 |  |
| NumOfRotationKey | 旋轉關鍵幀數 | 2 |  |
| NumOfScaleKey | 縮放關鍵幀數 | 2 |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Bone0: Time = 0  Translation | Bone0在Time=0的位移關鍵幀 | 未壓縮:4\*3  壓縮:2\*3 |  |
| Bone0: Time = 0  Rotation | Bone0在Time=0的旋轉關鍵幀 | 未壓縮:4\*4  壓縮:1\*4 |  |
| Bone0: Time = 0 Scale | Bone0在Time=0的縮放關鍵幀 | 未壓縮:4\*3  壓縮:2\*3 |  |
| Bone0: Time = T  Translation | Bone0在Time=T的位移關鍵幀 | 未壓縮:4\*3  壓縮:2\*3 |  |
| Bone0: Time = T  Rotation | Bone0在Time=T的旋轉關鍵幀 | 未壓縮:4\*4  壓縮:1\*4 |  |
| Bone0: Time = T Scale | Bone0在Time=T的縮放關鍵幀 | 未壓縮:4\*3  壓縮:2\*3 |  |
| … | … | … |  |
| Bone1: Time = 0  Translation | Bone1在Time=0的位移關鍵幀 | 未壓縮:4\*3  壓縮:2\*3 |  |
| Bone1: Time = 0  Rotation | Bone1在Time=0的旋轉關鍵幀 | 未壓縮:4\*4  壓縮:1\*4 |  |
| Bone1: Time = 0 Scale | Bone1在Time=0的縮放關鍵幀 | 未壓縮:4\*3  壓縮:2\*3 |  |
| Bone1: Time = T  Translation | Bone1在Time=T的位移關鍵幀 | 未壓縮:4\*3  壓縮:2\*3 |  |
| Bone1: Time = T  Rotation | Bone1在Time=T的旋轉關鍵幀 | 未壓縮:4\*4  壓縮:1\*4 |  |
| Bone1: Time = T Scale | Bone1在Time=T的縮放關鍵幀 | 未壓縮:4\*3  壓縮:2\*3 |  |
| … | … | … |  |
| Bone2: Time = 0  Translation | Bone2在Time=0的位移關鍵幀 | 未壓縮:4\*3  壓縮:2\*3 |  |
| Bone2: Time = 0  Rotation | Bone2在Time=0的旋轉關鍵幀 | 未壓縮:4\*4  壓縮:1\*4 |  |
| Bone2: Time = 0 Scale | Bone2在Time=0的縮放關鍵幀 | 未壓縮:4\*3  壓縮:2\*3 |  |
| Bone2: Time = T  Translation | Bone2在Time=T的位移關鍵幀 | 未壓縮:4\*3  壓縮:2\*3 |  |
| Bone2: Time = T  Rotation | Bone2在Time=T的旋轉關鍵幀 | 未壓縮:4\*4  壓縮:1\*4 |  |
| Bone2: Time = T Scale | Bone2在Time=T的縮放關鍵幀 | 未壓縮:4\*3  壓縮:2\*3 |  |
| … | … | … |  |
| BoneN: Time = 0  Translation | BoneN在Time=0的位移關鍵幀 | 未壓縮:4\*3  壓縮:2\*3 |  |
| BoneN: Time = 0  Rotation | BoneN在Time=0的旋轉關鍵幀 | 未壓縮:4\*4  壓縮:1\*4 |  |
| BoneN: Time = 0 Scale | BoneN在Time=0的縮放關鍵幀 | 未壓縮:4\*3  壓縮:2\*3 |  |
| BoneN: Time = T  Translation | BoneN在Time=T的位移關鍵幀 | 未壓縮:4\*3  壓縮:2\*3 |  |
| BoneN: Time = T  Rotation | BoneN在Time=T的旋轉關鍵幀 | 未壓縮:4\*4  壓縮:1\*4 |  |
| BoneN: Time = T Scale | BoneN在Time=T的縮放關鍵幀 | 未壓縮:4\*3  壓縮:2\*3 |  |
| … | … | … |  |

其中特別注明一下, 本檔案格式的生成方法

* 當格式(format) 標示出有使用位移關鍵幀壓縮(Position Compressed)時, 位移關鍵幀數據使用3個fix16來儲存, 否則使用3個float格式來儲存.
* 當格式(format) 標示出有使用旋轉關鍵幀壓縮(Rotation Compressed)時, 旋轉關鍵幀數據使用4個fix8來儲存, 否則使用4個float格式來儲存.
* 當格式(format) 標示出有使用縮放關鍵幀壓縮(Scale Compressed)時, 縮放關鍵幀數據使用3個fix16來儲存, 否則使用3個float格式來儲存.
* 因為每個骨骼(Bone) 位移關鍵幀, 旋轉關鍵幀, 縮放關鍵幀, 都分別有標明位移關鍵幀數(NumOfPositionKey), 旋轉關鍵幀數(NumOfRotationKey), 縮放關鍵幀數(NumOfScaleKey), 所以, 每個骨骼(Bone) 的位移關鍵幀數目,旋轉關鍵幀數目, 縮放關鍵幀數目, 可以應用戶的壓縮要求而有所不同.比如說某一個骨骼只有1個位移關鍵幀, 10個旋轉關鍵幀, 1個縮放關鍵幀.



* + 1. 技术方案所产生的有益效果

**本方案**描述一種實用新型，所产生的有益效果有：

* 讓2D骨骼動畫,在內存佔用及文件大小上都得到有效的壓縮，(即是說在無損質量的前題下，一般來說一個動畫100（Key frame）來說，本來4000bytes，壓縮後只要824bytes，壓縮率20％)
* 在所有32BIT或以上的CPU加載此格式時不用對它進行解壓即可使用, 只需在運行時實時轉換。
* 骨骼(Bone) 的位移關鍵幀數目,旋轉關鍵幀數目, 縮放關鍵幀數目, 可以應用戶的壓縮要求而有所不同.比如說某一個骨骼只有1個位移關鍵幀, 10個旋轉關鍵幀, 1個縮放關鍵幀, 可針對位移
  + 1. **本方案已向國家知識產權局申請發明專利保护**
* 位移關鍵幀使用3個fix16 (比如說ANSI C的short[3])來儲存,
* 位移關鍵幀也可選擇性地使用3個float (比如說ANSI C的float[3])來儲存.
* 旋轉關鍵幀使用4個fix8或fix16格式(比如說ANSI C的short[3] 或ANSI C的char[3] )來儲存,
* 旋轉關鍵幀也可選擇性地使用4個float格式(比如說ANSI C的float[3])來儲存.
* 縮放關鍵幀數據使用3個fix16格式(比如說ANSI C的short[3])來儲存,
* 縮放關鍵幀數也可使用3個float格式(比如說ANSI C的float[3])來儲存.
* 每個骨骼(Bone) 位移關鍵幀, 旋轉關鍵幀, 縮放關鍵幀, 都分別有標明

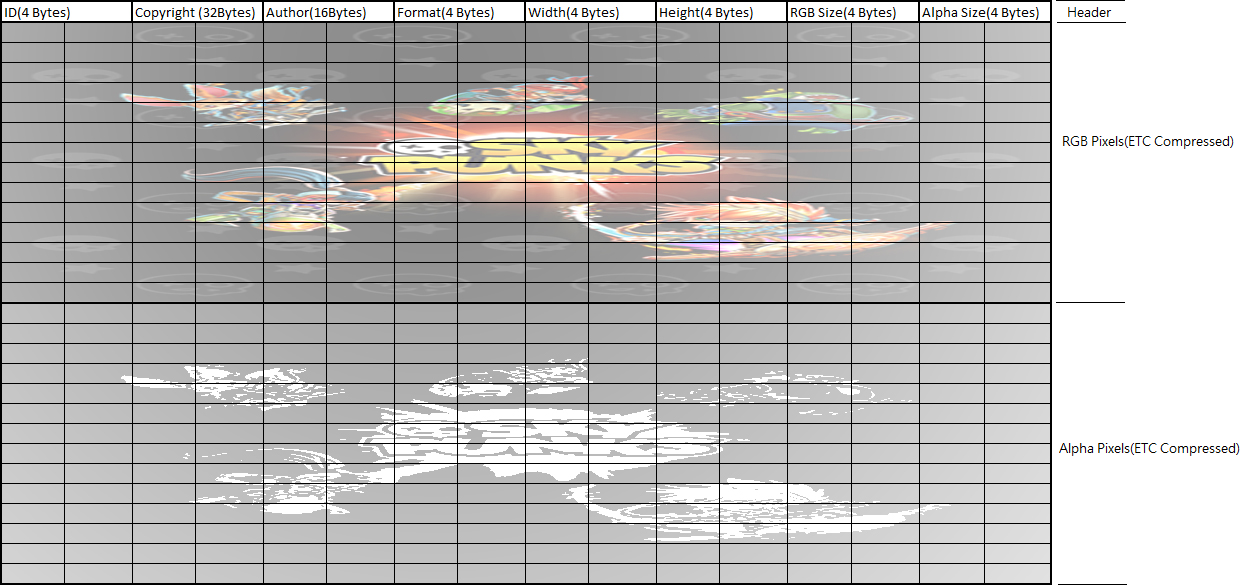
位移關鍵幀數(NumOfPositionKey), 旋轉關鍵幀數(NumOfRotationKey), 縮放關鍵幀數(NumOfScaleKey). 每個骨骼(Bone) 的位移關鍵幀數目,旋轉關鍵幀數目, 縮放關鍵幀數目, 可以應用戶的壓縮要求而有所不同.(比如說某一個骨骼只有1個位移關鍵幀, 10個旋轉關鍵幀, 1個縮放關鍵幀).

* 在所有32BIT或以上的CPU加載此格式時不用對它進行解壓即可使用, 只需在運行時實時轉換。
  1. **一種廣泛适用於OpenGL ES 2.0標準的RGBA紋理壓縮格式**
     1. **与本方案最相近的现有技术**

**2.3.1.1** PVRTC：它是一種支持透明通道(ALPHA)的紋理壓縮（Texture Compression）和紋理解壓(Texture Decompression)技術及檔案格式。這技術主要應用在減少紋理的內存佔用。

這技術存在的問題是：現今大多數移動設備只支援到OpenGL ES 2.0。而PVRTC並未被收入OpenGL ES 2.0的標準之內，另大多數GPU品牌類型都不支援PVRTC技術。這導致PVRTC技術只能應用在IMAGINE TECHNOLOGIES公司授權的GPU芯片上（如POWER VR系列），所以PVRTC技術在實際應用中不能廣泛地應用在大多數OPENGL ES 2.0的GPU硬件上。

* + 1. **本技术方案的详细阐述（发明内容）**
       1. 本技术方案描述一種實用新型，它是一種基於OpenGL ES 2.0的，並支持透明通道(ALPHA)的紋理壓縮檔案格式。跟PVRTC類似，此技術跟也是應用在GPU渲染時紋理的實時解壓上，減少紋理的內存佔用。不同的是：因為是基於OpenGL ES 2.0標準中的ETC紋理壓縮技術，所以它能廣泛應用在其他OpenGL ES 2.0的GPU硬件上。
    2. 此檔案格式有３大部份
* 頭文件（Header）
* 顏色(RGB) 象素緩衝(ETC Compressed)
* 透明通道(ALPHA)象素緩衝(ETC Compressed)
  + 1. 以下是檔案格式示意圖：



圖一, 檔案格式示意圖

其中，各數據的定義如下：

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **中文名稱** | **Bytes(位元數)** | **Notes(備注)** |
| ID | 文件格式標示 | 4 | 0x49 0x44 0x53 0x54 |
| Copyright | 文件著作權 | 32 |  |
| Author | 文件作者 | 16 |  |
| Format | 格式 | 4 | Alpha 1:1分辨率：0x00  Alpha 1:4分辨率：0x01  Alpha 1:16分辨率：0x02  Alpha 1:64分辨率：0x03 |
| Width | 圖片寬度 | 4 |  |
| Height | 圖片高度 | 4 |  |
| RGB Size | RGB部份長度 | 4 |  |
| Alpha Size | ALPHA部份長度 | 4 |  |
| RGB Texels (ETC Compressed) | RGB 象素數據 (已ETC壓縮) | =RGBSize |  |
| Alpha Texels (ETC Compressed) | ALPHA 象素數據 (已ETC壓縮) | =Alpha Size |  |

其中特別注明一下:

* 圖片的R,G,B象素是存放到”RGB Texels (ETC Compressed)”區域中，而圖片的透明通道(ALPHA)象素，是存放”Alpha Texels (ETC Compressed)”區域中。
* ”RGB Texels (ETC Compressed)”和Alpha Texels (ETC Compressed)是緊接相連的。
* 圖片的透明通道(ALPHA)象素可以1:1分辨率，1:4分辨率，1:16分辨率存儲，1:64分辨率存儲。
* Header中有指定圖片寬度Width, 圖片高度Height:

所謂1:1分辨率，圖片的透明通道(ALPHA)象素的寬等於Width，高等於Height

1:4分辨率，圖片的透明通道(ALPHA)象素的寬等於Width/2，高等於Height/2

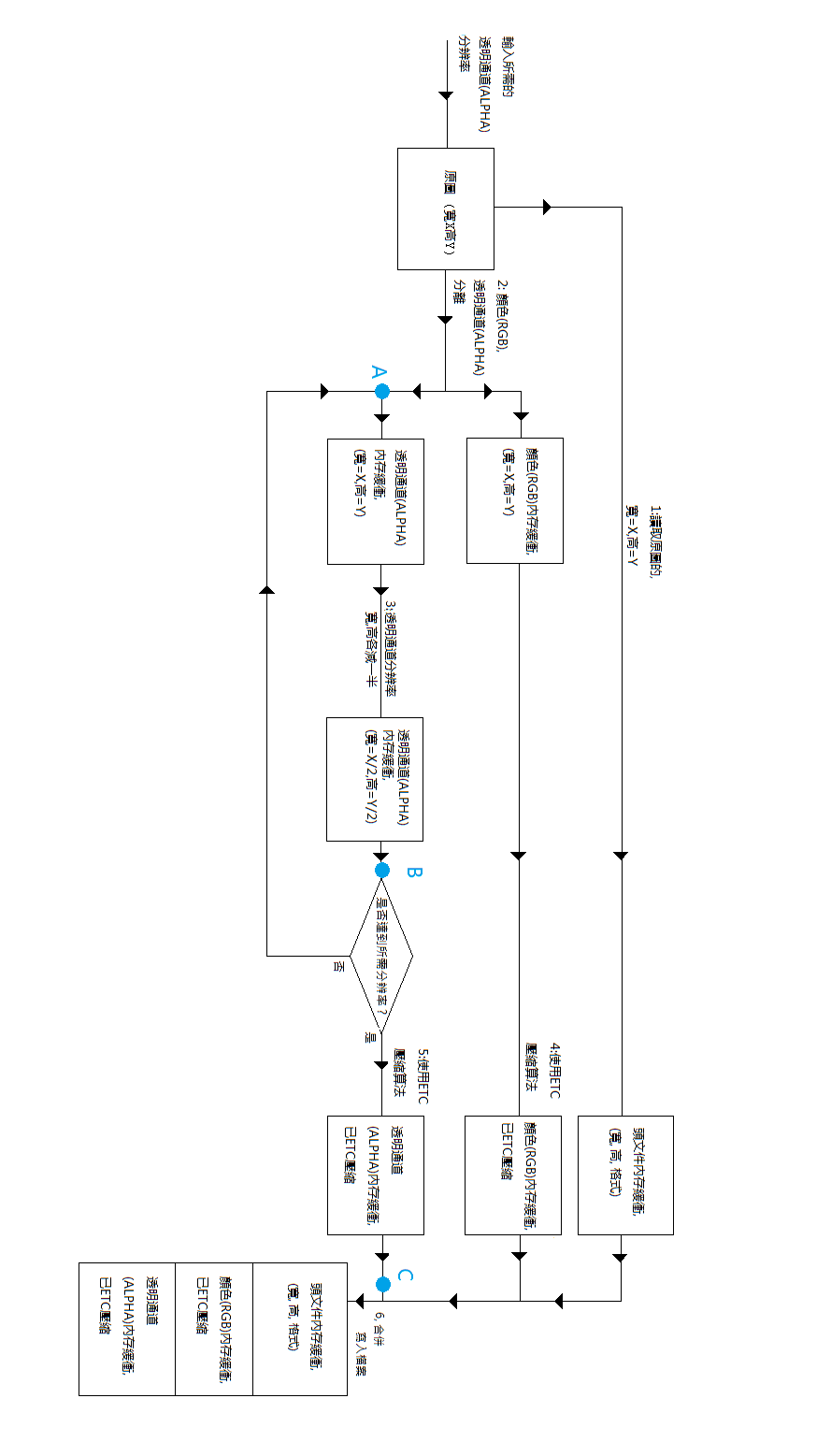
1:16分辨率，圖片的透明通道(ALPHA)象素的寬等於Width/4，高等於Height/4

1:64分辨率，圖片的透明通道(ALPHA)象素的寬等於Width/8，高等於Height/8

* + 1. 本檔案格式的生成流程圖

本檔案格式需要由轉換工具來生成，按圖二,此轉換工具的具體流程如下：

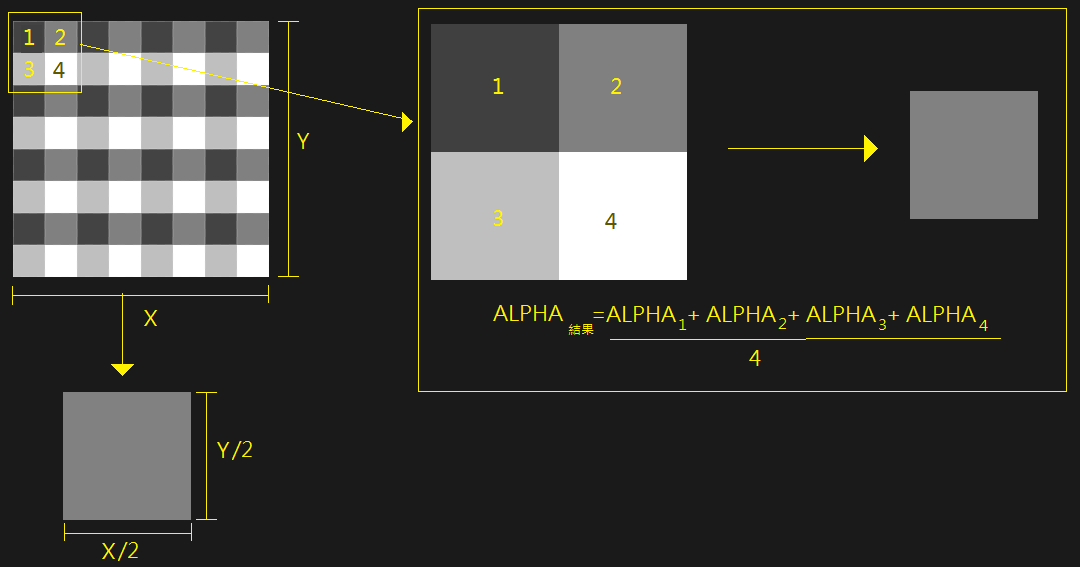
1. 讀取原圖的寬高和輸入所需的格式(Format)，並將其寫入頭文件內存緩衝（HEADER BUFFER）中。
2. 讀取原圖片的所有的RGBA像素，並把圖片的顏色(RGB)像素透明通道(ALPHA)像素分離出來。分別把顏色(RGB)像素存入顏色(RGB)緩衝，而透明通道(ALPHA)像素存入透明通道(ALPHA) 緩衝,
3. 按照輸入的格式(Format), 對透明通道(ALPHA)進行1:1, 1:4, 1:16或1:64的減低分辨率。
4. 用ETC算法把顏色(RGB)緩衝壓縮，並存入顏色(RGB) ETC緩衝
5. 用ETC算法把透明通道(ALPHA) 緩衝壓縮，並存入透明通道(ALPHA)ETC緩衝
6. 將頭文件內存緩衝（HEADER BUFFER）, 顏色(RGB) ETC緩衝, 透明通道(ALPHA)ETC緩衝寫入檔案.



圖二) 本檔案格式的生成方法

* + 1. 透明通道(ALPHA)減分辨率算法

　　圖片的透明通道(ALPHA)是一個8位(Bits)的象素數組。0代表完全透明，255代表完全不透明，中間不同大小的數值代表不同的半透明象素。因為在一些應用中（如遊戲，3D模型等），對透明通道(ALPHA) 象素的分辨率的要求不如顏色(RGB) 象素要求高，所以此文檔支援了1:1, 1:4, 1:16或1:64的透明通道(ALPHA)分辨率。如圖一我們在A，B兩個節點中，透明通道(ALPHA)的寬和高都會被減少1/2，這個步驟會一直循環到透明通道(ALPHA)分辨率降至用戶所輸入的需要的分辨率為止。圖三是透明通道(ALPHA) 降分辨率的示意圖。



圖三是透明通道(ALPHA) 降分辨率的示意圖

* + 1. **技术方案所产生的有益效果**

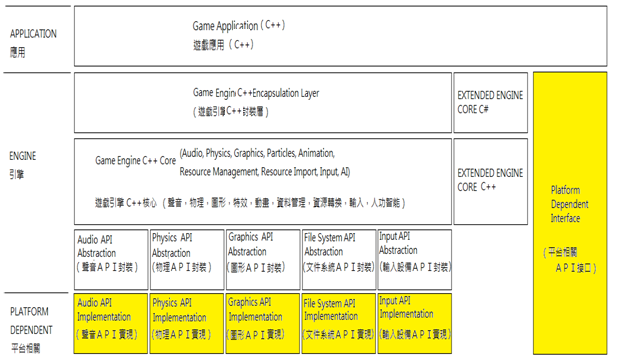
　　本技术方案描述一種實用新型，所产生的有益效果有：

* 讓紋理在內存佔用及文件大小上都得到有效的壓縮（跟32位（Bits）的RGBA格式比較，壓縮率25%），而且可以普及應用到所有支援OpenGL ES 2.0的設備上。
* 圖片的透明通道(ALPHA)象素可以1:1分辨率，1:4分辨率，1:16分辨率存儲，1:64分辨率存儲，增加圖片的壓縮率。
  + 1. **本方案已向國家知識產權局申請發明專利保护**
* 圖片的R,G,B象素是存放到”RGB Texels (ETC Compressed)”區域中，而圖片的透明通道(ALPHA)象素，是存放”Alpha Texels (ETC Compressed)”區域中。
* 而”RGB Texels (ETC Compressed)”和Alpha Texels (ETC Compressed)是緊接相連的。
* 圖片的透明通道(ALPHA)象素可以，以1:1分辨率，1:4分辨率，1:16分辨率存儲，1:64分辨率存儲。

**3、研究方法**

项目作为基础性平台，需要解决的关键技术问题比较复杂，涉及到多学科之间的交叉使用。3D引擎需要结合动画、语言等相互应用。具体解决的技术性问题如下：

**（1）跨平台移动设备游戏引擎**



图一）引擎的架构方块图

本引擎是目标是能够让开发者能够在单一平台上开发调试之后便能在各平台上运行发布(WRITE ONCE DEPLOY ANYWHERE)。

本引擎将采用ANSI C++为核心。因为C++是支援现行主流的移动,主机,台式平设备,所以游戏应用是可以移植到在各个支援ANSI C++平台上运行,如IOS,ANDROID,PC

图一是游戏应用本引擎的架构方块图,其中包括

* **GAME APPLICATION**

游戏应用本身是用C++来编写。

* **GAME ENGINE C++ CORE**

这是游戏引擎C++核心代码层,整个引擎的大部份的功能都在这一层实现,用C++的原因是它能在各平台生成其原生代码,确保运行速度,它主要包括了音效,物理模拟,２Ｄ/3D图形渲染,粒子系统,动画系统,资源管理,资源格式转换,输入,AI等。

* **FUNCTIONAL API ABSTRACTION**

这是一个相对薄的硬件平台API抽象层(估计占整个引擎代码量<10%),为了方便移植到各硬件平台及保证引擎C++核心代码的高移植性,引擎C++核心代码不会直接调用特定硬件平台的功能,所有这些功能都先由这硬件平台API抽象层所定义的API封装,而这些功能的具体实现则交由下一层按不同的平台实现。

* **FUNCTIONAL API IMPLEMENTATION**

这是一个相对薄的硬件平台API实现层(估计占整个引擎代码量<10%),每当引擎要移植时,总有不同的硬件适配或平台特有的函数库。因为上面各个模块都是设计成硬件平台独立的,所以我们只有要重写这个硬件平台API实现层就能把引擎移植。

* **PLATFORM DEPENDENT INTERFACE**

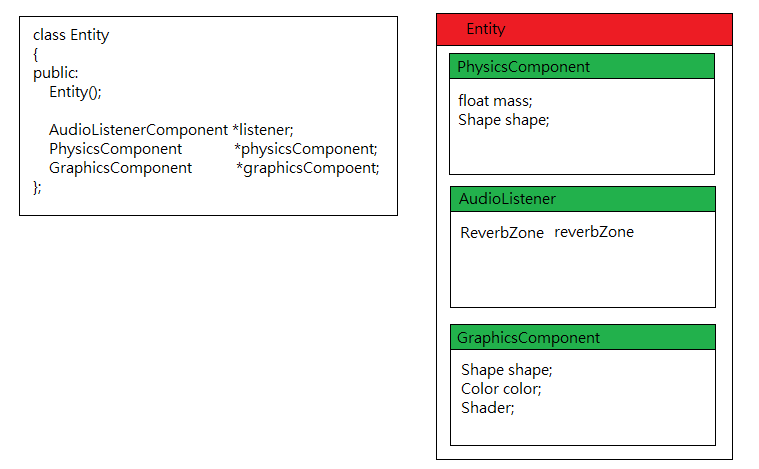
在一些例外情况(如支付,广告,交叉推广,调用第三方SDK等),开发者需要直接调用平台相关代码,引擎也会各平台提供专用统一接口来提供这种需求。

**（2）面向实体（ENTITY）及组件（COMPONENTS）模式游戏引擎架构**

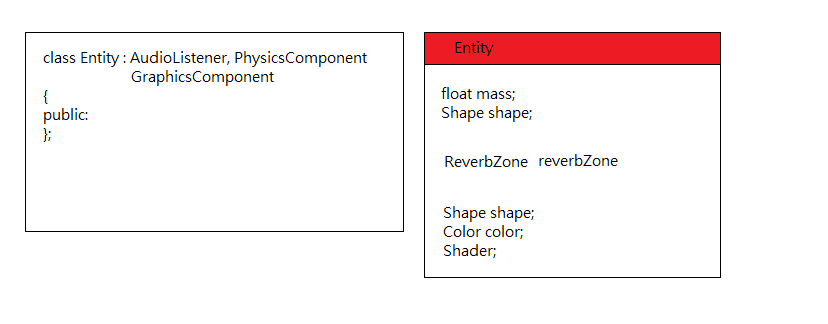
为了做到简单易用,可视化编程和整合式游戏开发环境,我们采用了实体组件架构(ENTITY-COMPONENTS-ARCHITECTURE)。

实体组件架构(ECA)是软件架构的一种。实体组件架构中有实体(ENTITY)和组件(COMPONENTS),每个实体包含一个或更多个组件。每个组件都可为实体添加额外的行为(BEHAVIOUR)和属性(PROPERTIES)。在游戏的场景中的每个对象是一个实体(如敌人,子弹,车辆等),而不同的实体是通过不同的组件来组合。

ECA架构思想是基于组成(COMPOSITION)优于继承(INHERITANCE)的原则。它另开发者在制作游戏中的实体的时候有更大的灵活性。因为组成(COMPOSITION)是可以在运行时执行的,因此实体的行为属性可以在运行时通过添加或移除组件而被改变。相反继承(INHERITANCE)是编译时执行的,因此实体的行为属性必须把程序停止了才可被修改。



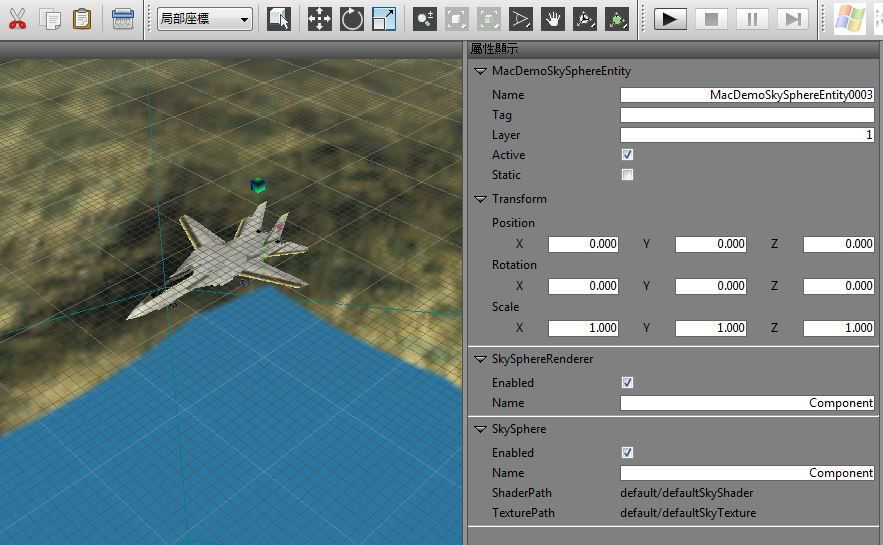
图二)以组成来开发(EXTEND BY COMPOSITION)



图三)以继承来开发(EXTEND BY INHERITANCE)

这意味着ECA架构可以让我们透个工具程式及一个简单单一动作（如鼠标拖动）就可以在运行时添加一个实体，并在运行时为实体添加属性和行为。这也意味着ECA架构能让非技术人员都可以新增，移除，修改，游戏功能等操作，简单易用。而透过工具程式我们可以做到编程可视化。因为工具程式不一定要求技术人员来操作，游戏开发流水线上各工种都可以在整合式游戏开发环境一同协作，无缝一体化。

以上提及的工具称为场景编辑器，我们用了ECA架构和Ｃ＃反射机制来实现，这两个技术另编辑器可以处理各种类的实体和组件及在运行时读取实体和组件的属性类型及数值，继而生成相对应的介面给用户修改。如图四，我们有一个叫MacDemoSkySphereEntity的实体，它包括了一个SkySphereRenderer及SkySphere的组件。其SkySphereRenderer有一个属性Enabled，我们可以在用户介面把这个值设为false，那背景就不会显示。又例如SkySphere, 我们可以把贴图拖到TexturePath那一个属性上，那背景的图就可以简单的更改。

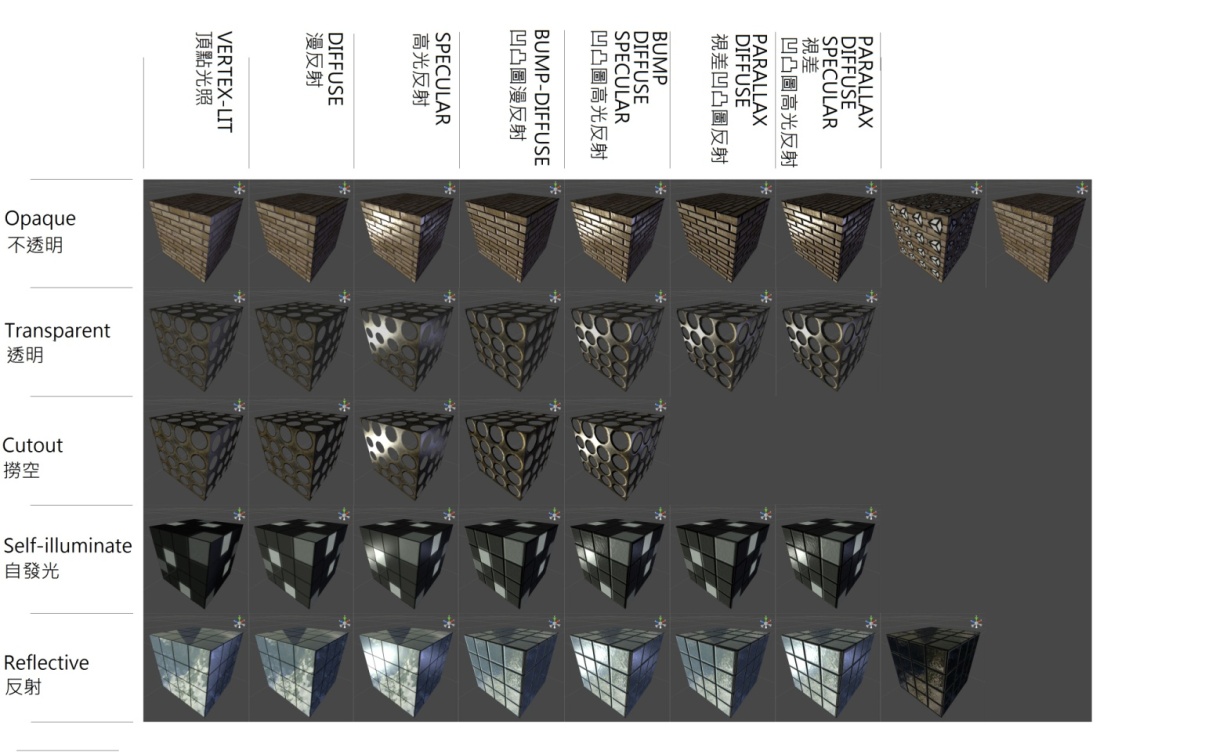


图四）可视化编辑器

**（3）跨平台引擎专属渲染语言规格**

现代移动设备的图形硬件及API(如OPENGLES2, HLSL)已经支援可编程的著色器,技术人员可以为图形硬件自定义渲染算法做出不同的画面效果。

图五是一此渲染效果的一些例子,其中包括(图五中的列)顶点光照,漫反射,高光反射,凹凸漫反射,凹凸高光漫反射,视差凹凸漫反射,视差凹凸高光漫反射。在这些技术上,我们又有几组变种(图五中的行),包不透明,透明,捞空,自发光,镜反射等。

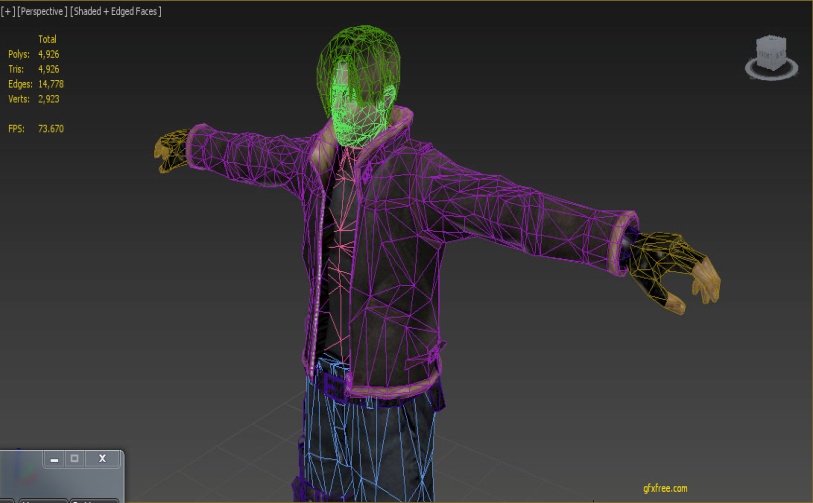


图五）着色器效果

**（4）2D /3D骨格模型渲染，骨格动画**

本引擎将支援2D/3D模型渲染，2D/3D骨格模型动画渲染及低流量2D/3D骨格模型格式。游戏3D模型一般由三角形组成，其中包括顶点包括，位置，法线，切线，纹理坐标，与骨格的索引比重等属性（图九左）。其中用到的2D/3D模型技术原理包括Skinning（蒙皮），Vertex Blending（顶点混合），Morphing（变形）等基本功能。本引擎也可以用着色树为2D/3D模型提供自定义的材质特效。

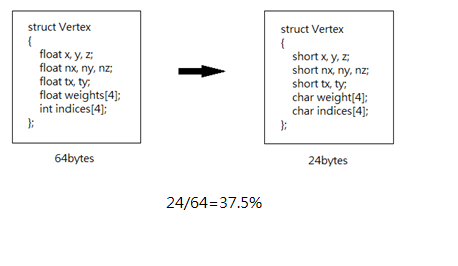




图八）3D模型，位置，法线，切线，纹理坐标

**（5）低流量2D/3D骨格模型格式，2D/3D骨格动画格式**

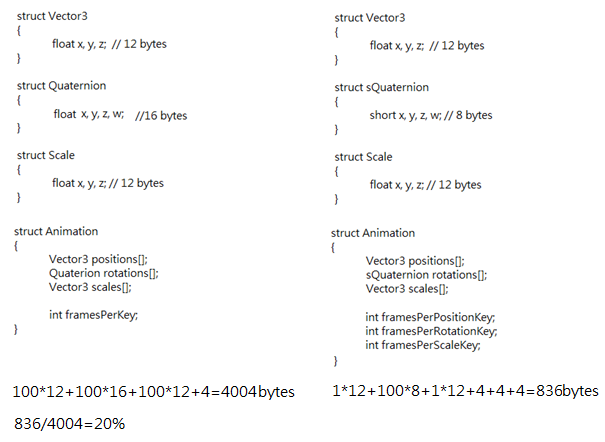
除了以上基本的功能,因为2D/3D模型及动画占用了游戏包体不小空间,本引擎在移动设备上的2D/3D模型及动画格式也有作出优化。因为顶点数据间有相对强的空间一致性(SPATIAL COHERENCE),而且顶点数据在游戏应用中的数值范围是有限的,所以我们可以利用这特性做压缩(图九:压缩率37。5%)内存及数据量,使得游戏包可以变得更少,可以适配更多机型。



图九）３Ｄ模型顶点数据优化

在2D/3D骨格动画格式方式,美工会在图形软件中定义出动作的关键帧,关键帧是定义了该动作在这个帧中的每个关节的位置,转动及缩放讯息。然而美工不会在所有帧上都定义关键帧,一般都是某个时间间隔(如每3帧一个关键帧),其余的则是实时用插值算法算出来。因为关键帧之间一般有很强的空间一致性,而且关键帧在游戏应用中的数值范围是有限的,加上关键帧之间时间一致性(TIME CHERENCE),所以位置(POSITION)和缩放(SCALE)关键帧都可以被优化。所以我们可以利用这特性来做压缩内存及数据量,我们初步估计可以做到最低每帧2.3bytes（2.3Bytes/Frame），平均每帧6.3bytes（图十：压缩率20%),使得游戏包可以变得更少,可以适配更多机型。

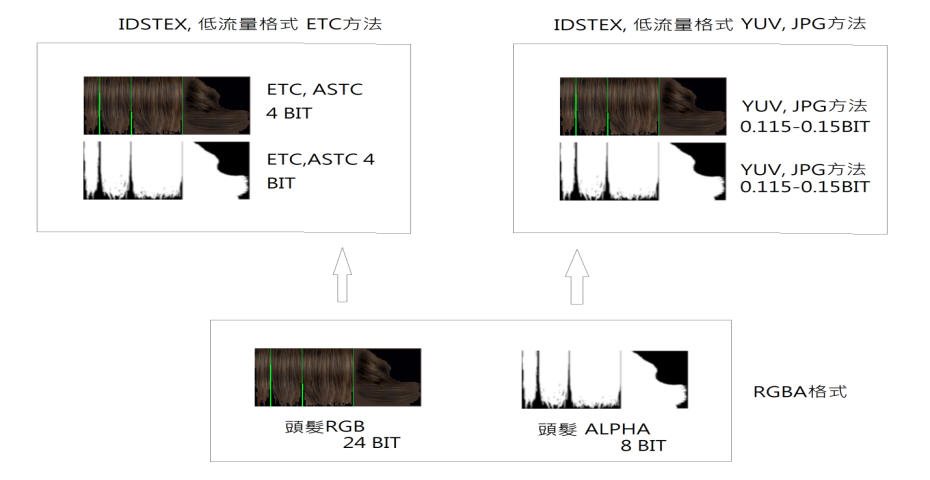
本引擎希望提供低流量格式技术降低游戏对内存等设备规格上的要求,使开发者可以开发更多的内容和加大游戏的可适配移动设备数。

****

图十）3D模型动画数据优化

**（7）低流量贴图格式及转换工具**

本项目研发了更适合移动设备所要求的小文件及内存用量的低流量贴图格式及转换工具。一般带ALPHA通道贴图RGBA每像素占32BIT。而我们的方案初步估计可以做最低每像素0.23069BIT，平均每像素0.3BIT。其原理是把RGBA,分为RGB及A两个部份, 然后跟据它们数据特性来用针对性的算法来压缩。RGB部份我们可以利用它的颜色的空间一致性来针对性地(如ETC,ASTC或JPG等)压缩,ALPHA部份我们则视乎贴图的用途。如文字,我们可以用GZIP来压缩;或如人物头发,我们可用ETC,ASTC或JPG来压缩。图十三是以上压缩方法的示意图。



图十三）低流量贴图格式

**（三）结论部分**

**1、实现的功能**

本项目基于ETC+A，JPG+A贴图技术，以ENTITY-COMP ONENT MODEL为技术框架，使开发者基于同一套模式来组合整个游戏，在特定用户可选条件下，利用了３Ｄ动画关键帧的时间与空间同调的特性（SPATIAL AND TEMP ORAL COHERENCE）来实现压缩。同时按开发者选择，实现渲染语言编译器来转换器来转换成平台渲染语言。本项目实现了以下技术内容

（1）设计出3D游戏引擎专属的渲染语言，支持GLSL100，GLSL300，HLSL3.0

（2）支持低流量3D/2D骨骼动画格式，现实3D/2D骨骼动画导出工具，最低每帧2.3bytes，平均每帧6.3bytes

（3）开发出一套开放式3D游戏引擎技术及SDK

**2、技术创新点**

本项目的技术创新点及突破点主要体现在以下几点：

1. 减少3D游戏应用包大小或内存要求,降低用户下载门槛
2. 透过开放式插件SDK方式,让用户新增引擎核心功能
3. 整合式游戏开发环境,使设计,技术,美术,各工种协作无缝一体化
4. 对3D模型和动画档案大小及内存专向优化
5. 专属渲染语言来解决跨平台渲染语言问题

**（四）建议部分**

**1、总结**

本项目是一款针对国内开发者和用户的游戏开发程序库及工具（或中间件：MIDDLEWARE），以解决3D图形渲染、动画系统、资源管理、资源格式转换等问题，并透过开放式软件架构（非开源）让开发者去拓展自己独有的引擎功能。

本文实现的是一个具备独立引擎功能的系统，游戏开发人员可以在这个引擎平台上开发各种不同的游戏，而不用在底层处理上花费过多的时间。2016年，中国游戏市场实际销售收入达到1655.7亿元，同比增长17.7%；其中移动游戏市场实际销售收入为819.2亿元，同比增长59.2%，市场占比超过客户端游戏市场达到49.5%，成为份额最大、增速最快的细分市场。2016年，中国游戏用户规模达到5.66亿人，同比增长5.9%，增长率小幅上升。移动游戏市场持续爆发已经成为行业增长的主要驱动力。目前移动游戏已经为中国社会和经济的发展发挥了积极作用，中国移动游戏产业已经发展成为一个具备较大规模的产业。而且，作为文化创意产业的重要组成部分，移动游戏产业正在和电影、电视、音乐等传统文化娱乐产业不断融合，进一步扩大它对中国社会和经济发展的影响力。

尽管游戏引擎技术在国内发展还是刚刚起步，技术还不够成熟，其完善也需要一段时间，但是随着中国游戏产业的迅速发展，游戏引擎市场将越来越受到广泛的关注和应用，也将成为一个新兴的产业增值亮点。

**2、展望**

在现在工作的基础上，根据产业的发展需求，引擎还将新增一下功能：

1、64位色高精度动态渲染管道

2、支持当前所有的基于像素的光照和渲染技术，包括使用法线贴图技术的参数化phong光照、虚拟位移贴图、光线衰减函数，采用预计算的阴影遮罩技术以及使用球形harmonic贴图的预计算的凹凸自阴影。

3、扩展美术、设计、开发等一体化引擎办公桌面，让引擎使用更加人性化。

**（五）参考文献**

[1]潘云鹤编著：CAD系统与方法.浙江大学出版社，2000-1

[2]（美）（John P.Chismar:3D Studio MAX3 Media Animation.电子工业出版社，2000-4）

[3]基于3D结构和图像的复杂场景建模绘制.高月芳，张准声，张佑生.计算机工程与应用，2003.10