相关主题 RECOMMEND ARTICLE
- Introduction to 3d game engine design using directx-9 and c#(10)
- Introduction to 3d game engine design using directx-9 and c#(9)
- OGRE 3D 程序设计 (9)
- Introduction to 3d game engine design using directx-9 and c#(8)
- Introduction to 3d game engine design using directx-9 and c#(7)
- Introduction to 3d game engine design using directx-9 and c#(6)
- OGRE 3D 程序设计 (8)
- OGRE 3D 程序设计 (7)

MORE

推荐文章 RECOMMEND ARTICLE
- 数据广播方案的优化
- 网络游戏的位置同步
- 游戏音乐制作案例之《战火 红色警戒》音效制作揭秘
- 英雄连Online 原画
- 游戏音乐制作案例之《乱武天下》
- 游戏音乐制作案例之《诛仙》
- 《鹿鼎记》最新原画
- MIDP2.1规范的新特性

MORE

热门文章 HOT ARTICLE
- [电子书下载]游戏设计 — 原理与实践
- [电子书下载]网络游戏开发
- 游戏设计全过程
- [电子书下载]游戏设计技术
- [电子书下载]游戏设计理论
- CS游戏人物模型制作教程
- CG人物插画基本流程
- [转贴]MAX高级人头教程

MORE

您的位置： 游戏引擎

文章标题	一种2D游戏引擎的设计与实现		
来源：	[ogdev]	浏览：	[995]

摘 要 对现有游戏引擎开发技术进行改进，提出了一种有效的2D游戏引擎设计与实现方法，该方法以引擎架构为基础，以累积渲染技术作为主要渲染加速手段;结合精灵与动画技术进行具体实现，达到游戏引擎设计的目的。最后通过该方法实现了一个2D游戏引擎，并用实验验证了该方法的有效性。

1、引言

2D游戏开发是游戏开发领域中一个不可或缺的重要分支，绝大多数休闲游戏与动漫游戏都是使用2D游戏技术开发实现的。2D游戏技术已经成为了手机及一系列小型设备游戏开发的主导技术。游戏引擎是游戏开发的最基础部分，为此我们提出了一种有效的2D游戏引擎的设计与实现方法。通过该方法，我们设计并实现了一款2D游戏引擎。

2、2D游戏引擎核心构架

2.1 引擎构架

引擎构架是游戏引擎的骨架，它的任务就是把图像渲染、输入处理、音频播放、资源管理等游戏引擎的基本功能组合成一个有机的整体。目前已经有很多比较成熟的引擎架构方法包括：

(1)结构化构架。结构化构架采用的方法是将游戏引擎内部的模块以结构化的形式组合，并以API的形式提供接口，这种构架模式的优点是效率高，接口简单清晰，适合于做一些速度要求比较高的引擎，但是缺点是其模块之间耦合度高，不易修改。

(2)基类根形式构架。基类根形式构架是以基本的几个抽象服务提供接口类为基础，并以接口与实现相分离为原则，来进行引擎实现，提供的实现类在引擎内部创建，而将接口供给用户使用。这种构架模式的优点是结构清晰，实现灵活，可以适应比较大的变化，适合于做一些较大且有跨平台需求的引擎，但是该架构依赖于基本的抽象类的继承关系，导致引擎内部类继承了无用的功能而变得虚大，况且接口的通用性使得实现的效率不高，导致整体效率偏低。

(3)组件形式构架。组件形式构架是把不同功能的模块做成互相独立的系统，模块内部可以使用任何构架方式，只需要提供相应的接口即可。引擎以模块管理器为核心，支持插件形式组件增加方式，可以将新增的功能组件以插件的形式来插入系统工作，这种构架方式极其灵活，且模块内效率比较高，所以某些大型商用游戏引擎采用这种方式，但这种方式设计比较困难，实现比较复杂，需要大量的开发经验。

鉴于本文所提引擎开发是以研究为主，且2D游戏要求效率不是很高，我们决定采用结构最为清晰的基类根形式构架方法。以4个基本服务抽象类作为整个引擎底层提供服务的基础(如表1)。在基本类的基础上派生出引擎使用的基本类结构(见表2)。

表1 基本服务抽象类及功能说明

类名	功能说明
IRefCount	提供引用计数功能
ITypeInfomation	提供类型识别功能
IObjectInfomation	提供对象识别功能
IPersistence	提供对象持久化功能

表2 基本类结构继承关系

类名	父类
IBase	ITypeInfomation
IResource	IBase、IObjectInfomation、ITypeInfomation

其中，IBase类是引擎中绝大部分接口类的基类，它实现了的类型识别虚方法。IResource类是引擎中资源的基类，它在IBase基础上实现了引用计数和对象信息功能。

2.2 引擎模块实现

为了实现方便，我们把引擎划分成7个功能模块：core，math，video，system，input，audio，scene。

(1)core模块。除了定义了引擎基本类及功能以外，还提供了一些基本设施，例如日志系统，异常处理，以及资源管理等。

(2)math模块。提供引擎需要的数学库，包括向量，矩阵，基本图元的定义和实现功能。

(3)video模块。负责基本图元的渲染，对渲染环境的设置，对纹理的创建及加载，以及基本字体渲染。

(4)system模块。提供引擎在操作系统下的驱动以及对操作系统功能使用的封装，包括窗口管理，定时器等。

(5)input模块。负责处理输入设备产生的事件，及对输入设备的管理，如对键盘，鼠标，手柄等的管理。

(6)audio模块。处理音频文件的播放以及音效处理等。

(7)scene模块。负责引擎中的场景管理。

引擎的具体架构见图1。

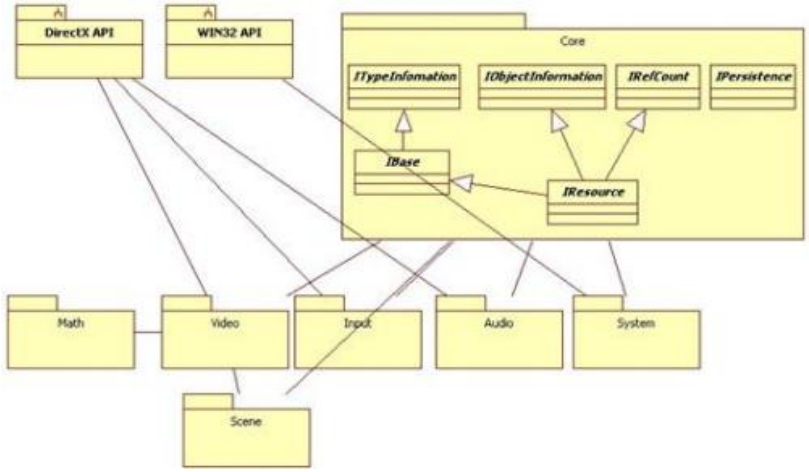

图1 引擎架构

3、2D游戏引擎核心渲染技术

3.1 累积渲染技术简介

在2D游戏中，常常将需要的游戏图像合并到一个大纹理中，渲染时使用该纹理进行大量图元渲染。如果一个一个渲染所需图元，引擎的效率必然会低，但如果一次性大量渲染许多图元，则效率会有较大提升。，鉴于2D图元的简单性，我们使用累积渲染技术就可以实现高效率的对大批量的图元进行渲染。

累积渲染技术的主要思想就是，在渲染2D图元时，不设置世界矩阵，而是创建一个相同的图元。利用原本的世界矩阵对此新图元的顶点直接进行变换，并且放入一个缓冲中，等适当的时机再一次性渲染。一次性渲染是在任何一个下列条件被满足之时进行的：

(1)2D场景结束时。

(2)图元类型变化。

(3)纹理变化。

(4)渲染状态变化。

(5)缓冲满。

(6)更换渲染目标(RenderTarget)。

3.2 与一般渲染技术的对比

在一般渲染技术和累积渲染技术中，处理世界矩阵和渲染方式都有较大的差别。一般渲染技术在将一个物体从模型坐标转换成世界坐标时，需要先设置世界矩阵，然后对这个物体进行渲染，这种渲染方式适用于每一个物体都有比较多的图元数据的3D场景。但是对于2D场景来说，这种技术效率太低。而累积渲染技术的世界矩阵一直为单位矩阵，在渲染2D图元时创建一个相同的新图元。利用此图元所需的世界矩阵对此新图元的顶点直接进行变换，并且放入一个缓冲中，需要时再一次性渲染缓冲。

3.3 累积渲染技术实现核心算法

算法1：渲染图元(render)

功能：使用累积渲染技术对2D图元进行批量渲染

输入：2D图元obj，纹理tex，世界矩阵world，渲染缓冲buffer，渲染设备videoDriver

输出：无

算法描述：

```
if (一次性渲染条件满足)
// 实现累积渲染
videoDriver->renderBuffer(buffer);
```

```
buffer->clear();
//设置当前图元类型
videoDriver->setType(obj->type);
//设置当前纹理
videoDriver->setTexture(tex)
end if

//从当前图元复制产生新图元，并对新图元进行变换，并压入缓冲

newObj = obj
//此变换改变的是newObj的顶点数据
newObj.transform(world);
//将定点数据放入buffer
buffer->push_back(newObj);
```

4、2D精灵与动画处理

在2D游戏中，精灵和动画处理是实现游戏效果的最基本方法，下面就对它们进行简要的分析。

4.1 精灵的实现分析

(1)精灵是一个矩形图元，其坐标分布为：以矩阵中心为原点，四个顶点分别位于(w/2，h/2)，(-w/2，h/2)，(w/2，-h/2)，(-w/2，-h/2)。

(2)精灵通过使用颜色键(colorkey)或纹理中的alpha通道来实现图像透明化。

(3)精灵通过指定纹理坐标来实现从组合大纹理中寻址小纹理。

(4)精灵使用各种变换(平移，旋转，缩放等)在2D屏幕坐标上面定位渲染，常用平移T(x，y)来将图元中心定位于屏幕的(x，y)处。

4.2 2D动画的实现分析

(1)动画是精灵的扩展，静止的动画与精灵无异，实现上也大致按照精灵的实现方法。

(2)动画需要一个动作序列，可将纹理按照矩形分块，然后把分块编号与每一个动作需要渲染的帧数储存至动画对象中的动作序列。

(3)动画的渲染与更新分开成两个过程。渲染同精灵一致，即对当前动作进行渲染，更新即为对当前动作与其所剩渲染帧数进行更新。

(4)如有必要，可以在一个动画中定义多个状态，每个状态对应一个动作序列，在游戏中可随时变换动画的状态。

5、引擎测试

通过以上方法，我们具体实现了一个2D游戏引擎，并对引擎的性能进行了测试，测试表明，引擎的性能非常令人满意。

测试环境：Athlon 2500 + 512M RAM+ Geforce 5200 FX

在全屏下旋转3100个精灵来进行alpha混合渲染时，使用累积渲染技术可以达到60.0fps

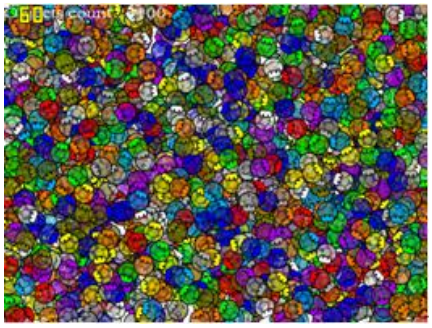

图2 测试截图

6、小结

本文提出了一种2D游戏引擎的设计与实现的方法，通过对引擎架构的分析，明确了2D游戏引擎的设计思路与方法。通过与3D引擎的渲染技术对比下，本文提出了专用于2D游戏引擎的高效渲染技术及其核心算法，即累积渲染技术，此方法在2D图元的渲染方面效率方面远比3D引擎使用的渲染技术高。最后阐述了2D游戏引擎中精灵与动画的简要实现方法。

游研发中心有权在网站内转载或引用
参与本留言即表明您已经阅读并接受上述条款