基于 XNA 的飞行仿真模型可视化与控制①

刘 春, 杨宏亮, 魏流锋

(沈阳航空航天大学 航空航天工程学部, 沈阳 110136)

摘 要: 与其它第三方飞行仿真软件开发平台相比, XNA 是微软 Visual Studio 集成开发环境的一个应用扩展, 开发的软件产品拥有自主知识产权. 研究了基于 XNA 框架的飞行仿真模型的可视化与控制问题, 介绍了飞行仿真模型的加载和渲染的关键技术以及飞行仿真中常见的舵面类、副翼、起落架的控制原理和方法. 结果表明 XNA 框架同样可以简单高效地实现飞行仿真模型的可视化与控制, 为具有自主知识产权的飞行仿真软件开发提供了新的方法和基础实例.

关键词: 飞行仿真; 可视化; 模型; 控制

Visualization and Control of Flight Simulation Model Based on XNA

LIU Chun, YANG Hong-Liang, WEI Liu-Feng

(Faculty of Aerospace Engineering, Shenyang Aerospace University, Shenyang 110136, China)

Abstract: Compared with other third-party flight simulation software development platform, XNA framework is an application of Microsoft Visual Studio integrated development environment extensions, and the flight simulation software development which has independent intellectual property rights. This paper focuses on the visualization and control issues based on XNA flight simulation model; it briefly includes loading and rendering of the key technologies of flight simulation model, and the theory and way for the control of aircraft components like the rudder, aileron and landing gears. It proved that the XNA framework can also be simple to efficiently implement the visualization and control of the flight simulation model, with independent intellectual property rights of flight simulation software development and provide a new method based instance.

Key words: flight simulation;visualization; model; control

视景系统是飞行仿真系统的重要组成部分, 用于实时模拟飞机飞行及场景, 为飞行员提供有效的视觉信息, 创造逼真的虚拟飞行环境, 其图形生成和显示质量直接影响飞行仿真的逼真度和飞行训练效果[1]. 飞行仿真需要使用大量的可视化模型, 这些模型包括复杂多样的飞行器及其部件、大范围的地形、机场、建筑物及景观等; 这些模型不同一般的可视化模型, 它的制作需要精细、逼真, 接近实物或场景; 这些模型中的一些部件, 比如舵面、起落架等在飞行模拟过程中需要通过控制进行实时相对运动, 这使得仿真可视化变得更加逼真, 同时也使得仿真可视化变得复杂. 这些模型不可能单一的依靠程序的点线面等图元自动生成, 通常是由设计人员通过诸如 3ds Max、Blender、Maya 等三维设计软件事先制作好三维模型, 然后通过程序导入到视景仿真系统实现模型的可视化. 为了高效研发飞行仿真系统, 国内通常采用现有仿真平台. 其主要分为两类: 一类是采用商业级别的 MutiGen Creator/Vega[2-4]、DirectX[5]、Virtools[6]、Mantis[7]、Unity3D[8]; 另一类是采用开源免费的 OpenGL[9-11]、OSG[12,13]、Flight Gear[14]、OGRE[15]. 虽然商业级别的平台功能强大, 却需要支付昂贵的许可证费用, 而目前的一些开源平台则普遍存在文档缺乏和过于繁复的

① 基金项目:辽宁省自然科学基金(2010276)

收稿时间:2014-10-31;收到修改稿时间:2014-12-05

特点. 针对上述问题, 本文选用微软推出的、可以免费使用的 XNA 游戏开发框架, 探索研究 XNA 框架下飞行仿真模型的可视化与控制问题.

XNA 是微软继 DirectX 之后开发的又一个主要用于三维图形程序的软件[16], 首个版本发布于 2006 年 3 月份. XNA 技术作为微软公司大力支持的下一代游戏开发平台, 具备全新的软架构、基于 Visual Studio 先进的开发环境、跨平台的游戏开发等诸多优点[17], 目前主要在游戏程序开图形程序设计以及可视化仿真等领域得到了广泛应用. XNA 框架包括图形引擎、应用程序模型和内容管道三个核心部分. 其文件为完全是托管的 dll 文件, 使用方便. 简化了系统开发流程, 实现过程简单[18]. XNA 模板内嵌于 Visual Studio 集成开发环境, 采用基于面向对象编程的 C#语言开发, 是一个轻量级的开发框架, 可以免费使用, 且内部封装好了三维仿真开发常用的向量、矩阵数据结构、操作运算和渲染等功能, 可大大减少开发成本, 缩短开发周期.

为了探索仿真系统研发平台的新途径, 本文将主要研究 XNA 框架下飞行仿真模型的可视化与控制方法, 包括飞机模型加载与显示、飞机舵面旋转和起落架收放控制等功能.

1 XNA的主体架构和运行机制

XNA 采用的是一种事件轮询机制, 默认的刷新频率为 60FPS, 默认就有 Initialize、LoadContent、UnloadContent、Draw 和 Update 五个方法. Initialize 方法主要用于变量的初始化, 如图形设备的初始化, 一些变量的初值设定等; 然后调用 LoadContent 方法, 以加载所有的资源内容, 包括模型、图片、声音, 为程序使用; 该方法结束后就正式进入游戏循环, 循环包括了 Update 和 Draw 两个方法, 每一次轮询刷新都会执行 Update 方法. 顾名思义, Update 方法以一定的频率更新场景并执行操作, 如对象的移动、碰撞检测、结束检查等都在该方法中实现; 而 Draw()方法则负责绘制场景; 最后, 游戏结束调用 UnLoadContent 方法释掉资源. 整个运行机制如下图 1 所示.

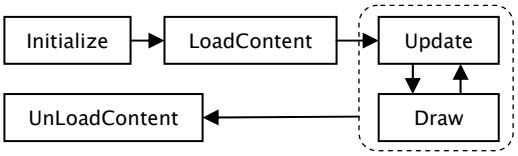

图 1 XNA 运行机制

XNA 的内容管道(Content Pipeline)支持常见三维建模软件制作的三维模型, 如 3dsMax 导出的*.FBX、*.X、*.DDS 等格式文件, 能自动识别三维建模软件中设置的纹理和骨骼节点, 省去了编写导入模型的大量代码且减少了遗漏等错误, 这样就可以通过 3ds Max、Maya、Blender 等三维绘图软件绘制好包含纹理效果的模型, 通过已有的接口或其它插件导出模型为*.FBX、*.X 格式为 XNA 所用. XNA 封装好了游戏底层类(Graphics、Audio、Input、Storage 等)可以快速方便地处理音频文件, 制作逼真的音效. 最后, 编程语言是采用基于.net 的 C#语言, 支持 Windows 与 XBox360 之间的跨平台运行, 既支持 2D 和 3D 的游戏开发, 也支持在 XBox360 的控制器, 实现震动等沉浸效果.

2 飞机模型可视化

大多数飞行仿真模型都是通过专业的三维建模软件设计的, 原始模型的格式和程序要求的格式往往不相同. 因此, 飞行仿真中解决模型的加载就是首要的问题.

XNA 支持的模型格式比较有限, 除了兼容原有 DirectX 的*.X 格式外, 还支持*.FBX 数据格式. 其中的 *.FBX(FilmBox) 首先由 Kaydara 所创, 现为 Autodesk 公司所有, 由于具备保存关键帧动画、纹理、光照、拓普信息等功能, 目前在游戏开发、虚拟现实等领域得到了广泛的使用, 如 Unity3D 就把它作为主要的模型格式. *.X 格式文件实际是一个文本文档, 可以采用记事本、文本编辑器等查看, 里面存储的主要是模型的几何信息和其它拓扑信息. 所以开发过程中使用*.X 格式文件常常会附带上模型的贴图文件, 而且必须确保在同一个目录下. 与*.X 格式文件相比, FBX 文件导出时除了保存模型的拓扑结构和几何信息之外, 还可以有选择的保留场景模型的光照、相机、贴图、骨骼动画等, 可以有选择地进行坐标系变换、单位转换. 这两种格式各有优缺点: *.X 格式是专为 DirectX 设计的模型格式, 贴图文件必须和模型文件在一个目录下, 在保持贴图文件名称不变的前提下可以直接更换贴图文件, 测试效果, 但在拷贝、传递等过程中贴图文件也容易丢失, 文件较大; *.FBX 格式有更好的通用性, 一个文件就能包含光照、相机、贴图、骨骼动画等信息, 更加安全实用, 但如果需要修改必须

在原有格式文件下编辑, 然后重新导出. 在新版本的 3D Studio Max 9 及以上版本它是默认被包含的导出格式, 并且 Maya 也支持它. 还有许多其他的 3D 应用开发程序支持导入和导出*.FBX 格式. 在 XNA, 它对于活动的模型、骨骼和蒙皮特别有用[19].

新建 XNA 工程项目之后, 要实现模型的加载, 首先要将三维模型手动添加到项目的 Content 即内容管道中, 添加的方法比较简单, 比如可以直接采用资源管理器的拖放形式, 也可以通过右键菜单添加进去. 内容管道会在程序的编译过程中对其中的模型进行编译, 且会验证它是否是一个有效的*.FBX 或*.X 类型的模型.

如果采用的是*.X 格式模型文件, 且模型使用了大量的纹理贴图, 那么这些纹理贴图就必须与模型文件存放在一个目录之下; 若选择导出*.FBX 格式文件, 且不选择将贴图文件打包在内, 也必须遵循这一规则, 因为这些模型渲染时需要引用到这些贴图文件.

模型加载则一般是在程序的初始化过程中, 而在 XNA 中就有专门的一个 LoadContent()方法, 所以一般是在该方法体内编写模型的加载导入代码. 在其中添加两个类级成员变量:

```
public Model model { get; protected set; }
protected Matrix world = Matrix.Identity;
//导入模型, 不需后缀
myModel = Content.Load<Model>
(@"Model\Aircraft\myAircraft ");
boneTransforms=new Matrix[myModel.Bones.Count];
//保存模型的初始骨骼矩阵
myModel.CopyAbsoluteBoneTransformsTo(boneTransforms);
```

第一个变量是 Model 类型的用来代表内存中的 3D 模型, 如前文所述, 它是 XNA 中已有的一个类. 第二个变量是 Matrix, 代表该特定模型的世界(world)矩阵. 在三维程序设计中, 模型的位置尺寸变换都是通过矩阵来控制的, 该矩阵代表在哪绘制, 如何旋转和缩放模型等.

一旦完成模型的加载就可以利用 XNA 框架平台对模型进行渲染. 由计算机图形学可知, 渲染一个图形需要考虑渲染图形的视景体和投影、光照、剪裁、Z 向深度等诸多问题. 在 XNA 框架下图形的渲染依靠 Effect 文件, 这一点类似于 DirectX. 对于没有特殊需求的仿真程序可以使用内部封装好的 BasicEffect 文件, 该文件包含了渲染一个模型的最基本的配置值. 有了效果文件就可以渲染模型, 一个渲染模型的基本流程如下:

```
foreach (ModelMesh mesh in model.Meshes)
{   //遍历整个模型结构
foreach (BasicEffect be in mesh.Effects)
{ be.EnableDefaultLighting( ); //采用默认的灯光设置
   //计算模型的世界矩阵
  be.World = world * mesh.ParentBone.Transform;
  be.Projection = camera.projection; //设置投影矩阵
  be.View = camera.view; //设置视图矩阵
}
  mesh.Draw( );
}
```

观察这些渲染程序可知, 在 XNA 框架下渲染飞行仿真的模型首先要遍历整个模型的结构, 对每一个子结构如方向舵、起落架都需要设定灯光, 示例中采用的是默认的灯光; 然后应该计算要渲染模型在整个场景中的世界矩阵, 确切的说就是模型的在整个场景下的三维位置; 最后指定渲染这个模型的投影矩阵和视图矩阵, 视图矩阵形象的说就好比一个摄像机的位置矩阵, 而投影矩阵则定义了这个摄像机的可视范围, 只要指定这两个矩阵就可以将图形显示在二维的显示界面上, 而不用编写其他复杂的矩阵运算就可以将图形渲染出来. 飞机模型的可视化效果如图 2 所示.

图 2 飞机模型的可视化效果

3 模型控制的原理

仿真过程常常需要控制模型中的某一部分, 让某一部分具有自由度 DOF(Degree of Freedom), 实现模型局部零部件的缩放、旋转和移动. 例如, 飞机舵面的旋转、起落架收放等. 要实现这些功能, 模型应该要被分成几个成员. 对每个成员, 都要确定有两种数据,

即几何数据和拓扑数据. 几何数据包括: 如顶点, 顶点包含了组成模型成员的所有三角形顶点的信息, 这些信息包括位置, 颜色, 法线等. 各部件之间的关系, 即拓扑数据, 以人的手臂为例, 要指定手臂是连接在肩膀上的, 肩膀就是其父级, 肩膀连接在躯干上身, 躯干上身则是肩膀的父级; 对于方向舵来说, 方向舵是连接到垂尾上的, 垂尾是其父级, 整个机身又是垂尾的父级.

在XNA内部专门设计封装有Model类, 这个类能直接读取并存储其内容管道的模型, 该类提供了两个最重要的Bones和Meshes公有属性, 每个成员的几何数据以ModelMesh对象的形式存储, 而ModelMesh对象是在 Model 的 ModelMeshCollection 中, 在这个对象提供了Meshes属性, 这些ModelMesh包含了渲染模型确定部分所需的所有数据, 包含所有顶点、索引、纹理、effect 等信息. 每个 ModelMesh 对象包含指向Bone 对象的引用, 而一个 Bone 对象包含指向父 Bone 的引用, 它必须连接到这个父Bone, 一个Bone只是包含一个变换矩阵, 它保存了这个 ModelMesh 相对于其父级 ModelMesh 的位置. 通过这种方式, 就可以将所有 Bone 对象连接到一起. 在一个三维模型里, 要驱动某一部件就要获取在3D建模时的指针部件作为Bones骨骼.

一个ModelMesh包含模型的一个成员的几何信息, 这个成员无法再分成更小的成员. 以固定翼飞机为例, 建模时将一整架飞机作为一个ModelMesh显然是不科学的, 因为仿真时需要控制升降舵、襟翼、起落架等部件的自由度 DOF. 因此较好的方法是将飞机的机身和固定翼部分作为一个 ModelMesh, 襟翼、副翼、升降舵、起落架, 舱门各自分别作为一个ModelMesh, 然后以机身部分为基准, 将机身的 ModelMesh 对应的Bone作为root Bone(根节点), 其它ModelMesh的Bone都连接到根节点即可实现各部件的 DOF 控制. 以 3ds Max 建模为例, 由于飞行器一般采用的是多边形建模的方法, 所以应切记将欲实现 DOF 的部件即作为XNA 中 ModelMesh 的部分在完成建模后分离为单独的一部分, 并用英文、数字、下划线取一个有意义的名字, 这个名字就是程序中 Bone 属性的标识, 极为关键. 要实现前文所述的 Bone 父子关系就需要使用 3ds Max 的链接工具, 单击工具栏的链接命令后, 首先选定准备链接的对象, 然后单击欲作为其父级的对象, 这样就完成了它们之间的链接关系设定. 有时, 许多同类的物体可以作为一个对象, 比如直升机的桨叶可以组合为一个整体, 这是可以选定这些对象, 使用工具栏上的成组命令成组. 之后和一般对象一样也可以添加链接关系.

XNA 内部封装了主要的矩阵运算、三维向量(Vector3)、二维向量(Vector2), 例如, 平移可以采用Matrix.CreateTranslation(Vector3 detaPos); 类似的, 缩放和旋转可以分别使用 CreateScale, CreateRotation. 这样就可快速方便的写出响应的动作代码.

而控制模型的本质就是操纵模型的变换矩阵, 这些变换主要有平移、旋转、缩放等, 然后按照一定的顺序相乘就可以模拟现实中的动作. 飞行仿真一般是建立好动力学模型, 根据动力学模型解算出来的值来操纵上述的变换矩阵, 通过这些变换矩阵, 经过投影变换, 剪裁等图形操作就可以渲染出实际的动作. XNA 的默认刷新频率为 60FPS, 每一次轮询刷新都会执行Update方法. 所以, 在XNA中可以将模型的解算放在 Update 方法体中, Update 方法也是 XNA 主程序自带的一个方法. 将动力学模型解算放在 Update 方法体中, XNA 的每一帧都会解算一次动力学模型, 接着将这些解算出来的值用来操纵变换矩阵, 就实现了模型的控制.

4 舵面的控制

对于固定翼飞机常常需要控制副翼、升降舵、方向舵等舵面实现各种飞行动作, 飞行仿真在可视化方面也需要将这些多面运动特征表现出来. 实际上, 这些翼面的运动都有一个共同的特点, 那就是围绕着一个固定的轴做有限的旋转运动, 且运动的角度往往很小.

正如上文所述, 在 XNA 中要想实现这些翼面的DOF首先需要在三维建模阶段就规划好其各自的自由度DOF节点及指定旋转轴. 在3dsMax中设置DOF节点的实质就是制定各部件的父子关系, 这可以通过其工具栏的链接工具实现, 而旋转轴的设置则可以通过操作层级面板上的调整轴工具来完成. 设置之后就可以通过其场景选择器选择或检查各部件的父子关系. 如下图 3 就是某飞机右副翼部件的父子关系图, 并以右副翼为例设置副翼的DOF如下图4所示. 在图3中还可看到襟翼、方向舵、升降舵和座舱等部件的父子

关系.

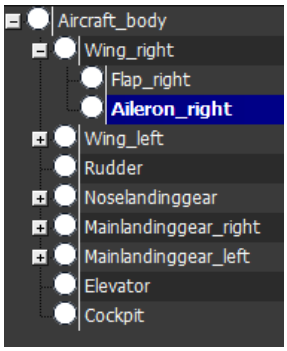

图 3　右副翼的 3dsMax 父子关系图

图 4　3dsMax 中设置副翼的 DOF

由于方向舵、襟翼、升降舵与副翼的分析类似, 因此对副翼进行详细的说明, 按照上文所述的导入模型的方法, 然后获取右副翼的骨骼和矩阵, 代码如下:

```
public override void LoadContent()
{   //导入飞机模型获取变化矩阵
myModel = Content.Load<Model>(@"Model/J82");
boneTransforms=new Matrix[myModel.Bones.Count];
myModel.CopyAbsoluteBoneTransformsTo(boneTransfo
rms);
    …
//获取右副翼模型骨骼
        fuyi_R= myModel.Bones["Aileron_right"];
//获取右副翼的变换矩阵
        fuyi_RTransform = fuyi_R.Transform;
…
}
public override void Update(GameTime gameTime)
{    …
//更新右副翼的位置
fuyi_rotation=Matrix.CreateRotationX(fuyiValue);
fuyi_R.Transform = fuyi_rotation * fuyi_RTransform;
…
}
```

XNA 的 Update 方法每一帧都会被调用, 借用这一机制, 更新及控制代码都应该放在 Update 方法中. 这些更新操作主要有接收副翼的输入量, 将输入量作用到副翼的骨骼矩阵上, 最后更新副翼的骨骼矩阵即可. 但需要注意的是写旋转矩阵的时候应该明确旋转轴, 这个旋转轴是在三维建模软件中建模时的零部件的局部坐标系而并非XNA下的坐标系, 程序设计时应该咨询建模师. 其次便是旋转的角度问题, 舵面的旋转角往往很小, 因此, 除了上面说的矩阵更新操作之外还应该更新最新的旋转角度值, 使其不会超过设定的最大旋转角, 否则就会发生失真现象.

5　起落架的控制

起落架是唯一的一种支撑整架飞机的部件, 是飞机不可或缺的一部份, 飞行器起飞和着陆时需要收放起落架, 而飞行仿真中需要模拟起落架的收放. 三维建模时, 起落架模型一般和整个飞行器模型是在一体的. 它和舵面一样, 在 3dsMax 三维建模阶段就应该设定好各部件的父子关系. 如下图 5 所示即为前起落架的父子关系.

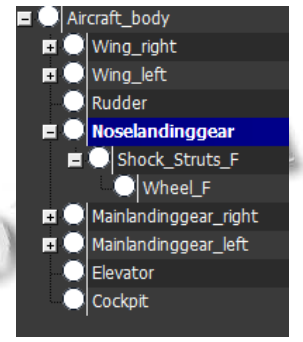

图 5　前起落架的 3dsMax 父子关系图

同样, 除了设置父子关系外, 还应该设定其旋转轴, 如图 6 所示.

图 6　3dsMax 中设置前起落架的 DOF

程序中导入飞行器模型后, 要想控制起落架部件,

首先要获取起落架部件骨骼. 以前起落架为例, 假定三维模型 myModel 中已经设置了前起落架部件的名称为 Noselandinggear, 则需要获取该部件的骨骼和变换矩阵:

```
//获取前起落架骨骼
qianqiluojia = myModel.Bones["Noselandinggear"];
//获取前起落架的变换矩阵
qianQLJTransform = qianqiluojia.Transform;
```

然后设定起落架的控制逻辑. 现实中的起落架的收放是有专门的开关控制, 程序中可以设置一个 bool 类型变量作为开关器, 用一个变量 RotateValue 来存放起落架旋转的角度起, 落架无论是收还是放一般只旋转九十度. 因此, 起落架的控制逻辑可以这样设定: 当起飞后判断输入的是收还是放起落架指令, 若为收起指令, 则首先检查 RotateValue 是否为 90°, 即是否到旋转极限, 若小于 90°, 则每一帧都使 RotateValue 按照旋转部距值增加, 这一部代码应该放在 Update 方法中, 假设是绕 X 轴旋转, 代码可以写成:

```
//按步距增加旋转的角度
RotateValue+=RotateStep;
//生成旋转矩阵
qianqiluojiarotation=Matrix.CreateRotationX(RotateValue) ;
//更新矩阵
qianQLJTransform.Transform=shengjiangduorotation * qianQLJTransform;
```

6 结语

综上所述, 本文采用 XNA 免费开源的框架实现了飞行仿真模型的加载和控制, 其中对模型渲染和渲染过程中产生的投影矩阵和视图矩阵的数据文件进行跟踪完成飞机结构的可视化效果使其在飞行姿态有更好的时效性, 并借助于其内部封装好的工具方便高效地实现了对诸如舵面、起落架等部件的 DOF 复杂结点控制. 这表明 XNA 完全可以和其它平台一样实现对仿真可视化模型的渲染与控制功能, 探索了一种研发具有自主知识产权的飞行仿真系统的新途径. 另外, 除了飞行仿真外, XNA 还可以扩展到其他领域仿真系统研发, 如航天、交通运输、地理和物流等领域.

参考文献

1 张燕燕,黄其涛,韩俊伟,等.飞行模拟器视景系统的设计与实现.系统仿真学报,2009,21(12):3662–3667.

2 杜星玥,卢昱,陈立云,等.视景仿真中的新型高效碰撞检测算法研究.计算机应用与软件,2013,30(7):271–275.

3 翟兆建,蔡志勇,赵红军.基于 Vega Prime 的灭火飞机投水特效模拟.计算机应用与软件,2012,29(4):260–262.

4 姚凡凡,梁强,许仁杰,等.基于 Vega Prime 的三维虚拟战场大地形动态生成研究.系统仿真学报,2012,24(9):1900–1904.

5 奚海蛟,张晓林.飞行模拟器视景系统中实时地形的研究与实现.计算机应用与软件,2009,26(4):150–151.

6 程勇刚,朱元昌,邸彦强,等.基于 Virtools 与 HLA 的模拟训练系统研究. 计算机仿真, 2008, 25(06): 265–269.

7 高煊,郑康平,郭俊丽,等.飞行模拟系统中分布交互式视景的设计与实现.指挥控制与仿真,2013,35(5):84–87.

8 张延,余红英,戚艺雪,万吉.基于 Unity3D/3DMAX 的导弹视景仿真系统.科技视界,2013,25:151–199.

9 贾仕俊.飞行仿真中飞机本体建模与视景仿真软件开发[学位论文].电子科技大学,2012.

10 陈力威,朱凡,刘希,等.Vega Prime 与 OpenGL 飞行控制系统可视化仿真平台设计.火力与指挥控制,2012,37(8): 191–194.

11 刘邦.基于 OpenGL 的模拟飞机飞行动画的设计与实现[学位论文].西安电子科技大学,2012.

12 郭佳,郭连成,张丽,等.基于 OSG 的飞行仿真系统视景平台的研究与开发.沈阳航空工业学院学报,2010,27(4): 1–4.

13 闫晓东.基于 OSG 的飞行视景仿真平台开发.计算机仿真,2008,25(5):58–60.

14 刘鹏.基于 FlightGear 的无人直升机飞行仿真技术研究[学位论文].南京航空航天大学,2011.

15 唐剑,汪红兵,吴跃,等.飞行仿真系统的软件架构研究.计算机应用,2006,26(6):1482–1484.

16 杨关胜,栗俊霞.精通 XNA 图形与游戏程序设计.北京:人民邮电出版社,2012.

17 何非.基于 XNA 的游戏框架[学位论文].电子科技大学,2008.

18 王星捷.基于 XNA 3D 的三维 WebGIS 系统研究与应用.计算机应用与软件,2013,30(4):283–286.

19 Nitschke B. Professional XNA Game Programming- For Xbox 360 and Windows. Indiana: Wiley Publishing, Inc, 2007.