# 物理仿真核心模块介绍

## 1综述

物理仿真核心模块总体有三个部分组成：（1）基础数据模块；（2）三维绘制模块；（3）物理动力模块。

由于不同版本，项目名称上会有所不同，但是总体上均可以划分至这三类模块。其他的部分主要属于应用这三个部分进行测试和应用开发。以下模块或项目，如无特别说明，均以v3版为基准进行介绍。

### 1.1基础数据模块

基础数据模块主要是建立封装用的基础数据类型和基础数据结构，实现与操作系统的基础操作接口，保障各个其他模块（包括第三方模块）之间的兼容性。

其内容包括，但不限于以下内容：

1. 基础数据类型
2. 打印模块
3. 时间模块
4. 同步锁模块
5. 多线程模块
6. 字符串模块
7. 数据结构模块

### 1.2三维绘制模块

三维绘制模块主要是解决基于DX的数据接口定义与实现。保证与DXUT、MFC等模块的兼容性。

其内容包括，但不限于以下内容：

1. 三维数学模块。
2. 三维包围盒模块。
3. 三维物体定义模块。
4. DX资源定义模块。
5. DX整体接口模块。

### 1.3物理动力模块

物理动力模块主要是解决基于Newton的数据接口定义与实现。保证与Newton Lib模块的兼容性。

其内容包括，但不限于以下内容：

1. 物理核心接口模块。
2. 物理对象与定义模块。

### 1.4模块与项目名

|  |  |  |  |
| --- | --- | --- | --- |
| **模块分类** | **模块名** | **项目名** | **备注** |
| 基础数据模块 | 基础数据模块 | 4DBasic | 包括数据类型定义、打印、多线程、同步锁等子模块。 |
| 数据容器模块 | 4DContainer | 包括链表、Hash树、Hash链表等子模块。 |
| 字符串定义模块 | 4DString | 有关字符串操作定义和操作的子模块。 |
| 三维绘制模块 | DXUT模块 | DXUTCore | DXUT核心模块。 |
| DXUTOptional | DXUT可选模块。 |
| 功能杂集模块 | 4DMisc | 包括从微软DX例子中“收集”过来的各种功能函数和功能。 |
| 数学基础模块 | 4DMath | 包括矢量、矩阵、四元数子模块。 |
| 4DRenderable | 可以生成基础形体（如：球、矩形体）的顶点和索引数据，用于物理仿真和绘制。 |
| DX资源模板模块 | D3DXResource | 包括各种可绘制资源子模块及管理子模块描述。 |
| 包围盒模块 | 4DGeometry | 包括球体和矩形体包围盒的基本定义。 |
| 4DCollision | 包围盒生成和求交运算子模块。 |
| 三维物体定义模块 | 4DWorld | 包括三维物体的各种基础属性定义。 |
| D3DXRenderable | 可绘制物体的基本描述。 |
| 物理动力模块 | Newton辅助模块 | dContainers |  |
| dJointLibrary |  |
| dMath | 包含基础数学定义。 |
| dScene | （暂未用） |
| tinyxml | （暂未用） |
| 物理核心接口模块 | 4DNewton | 包含各种实体和关节定义与对接的细节。 |
| 物理对象定义模块 | 4DPhysics | 包含各种物理实体对象的定义，不具备可绘制定义。 |
| 4DJoint | 关节对象的物理定义模块。 |
| D3DXPhysics | 可绘制物理对象定义模块。 |
| 各种应用模块 | 基础应用模块 | D3DXApplication | DX环境下应用模块。 |
| MFC应用模块 | MFCApplication | MFC环境下应用模块。 |
| 各种测试模块 |  | BumpCase | 基于DXUT框架的碰撞测试。  有绘制和物理仿真。 |
|  | DXUTCase | 检测纯DXUT框架的绘制情况。  仅检测绘制，无物理仿真。 |
|  | HashListCase | 检测Hash链表是否正常工作。  仅检测数据结构，无其他部分。 |
|  | JointCase | 基于DXUT框架的关节测试。  有绘制和物理仿真。 |
|  | MFCCase | 基于MFC框架的测试。  有窗口、绘制和物理仿真。 |
|  | ObjectCase | 检测有关对象的创建与释放。  仅检测对象数据的分配和释放是否正常。 |
|  | PhysicsCase | 检测物理模块部分是否正常。  仅检测数据模拟，不包括绘制部分。 |
|  |  |  |  |

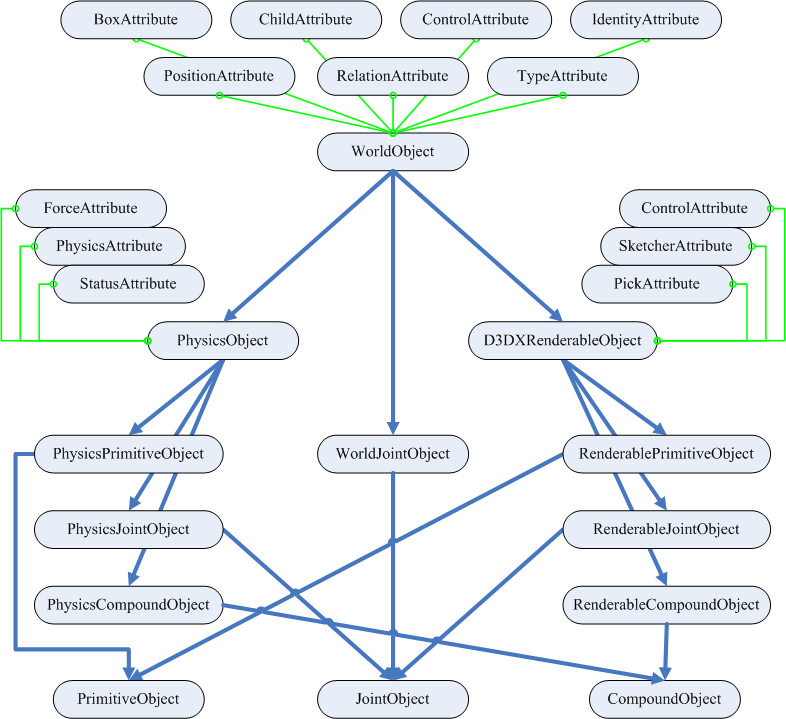
## 2主线

所有的模块结合在一起时，需要有一条主线。通过这条主线，将整个系统形成一个有机的整体。这条主线就是三维物体定义和继承路径。

### 2.1数据主线

#### 2.1.1主线图示

下图为系统的数据定义主线图。



1. 所有的对象分成三大类：基本世界对象（WorldObject）、物理对象（PhysicsObject）和可绘制对象（D3DXRenderableObject）。
2. 物理对象和可绘制对象按照基本（Primitive）、关节（Joint）和组合（Compound）三种物理属性又分别分类。
3. 最后将物理和绘制两个部分组合成三类可绘制物理体：基本体对象（PrimitiveObject）、关节体对象（JointObject）、组合体对象（CompoundObject）。

#### 2.1.2主要特点

数据主线中大量使用了虚基类，并形成了一个相对复杂的C++继承结构。

不过由于相互函数名雷同和重载，会导致程序阅读上有一定的麻烦。但是只要按照功能模块去理解，则比较容易把握整体的数据结构。

该数据主线具有以下特点：

1. **物理体系与绘制体系分离**。

将物理体系和绘制体系进行分离，这样物理体系和绘制体系均可以单独使用。

这样必要时可以只使用物理体系或者绘制体系进行模拟测试。主要是为了方便检测和分离可能存在的其他问题，以利于将来的错误查找和定位。

早期开发的版本都是先开发的DX部分，再挂接物理仿真部分。一个比较大的问题就是在测试物理问题的时候，必须挂接DX部分。这样有时候无法区分某个问题所发生的准确原因。

1. **思路清晰，利于开发**。

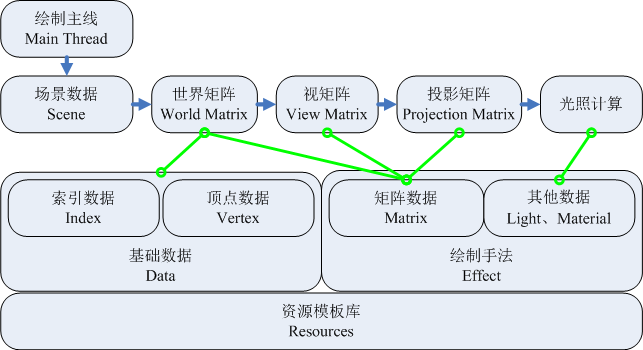
将数据分成比较清晰的主线图，利于程序员阅读、理解和开发。

早期开发的版本都是在原来的DX部分上直接增加物理属性。有关的属性管理比较混乱，导致个别类过大，函数过多。阅读和操作都比较混乱。

### 2.2绘制主线

#### 2.2.1主线图示

下图为系统的绘制主线图：



1. 场景数据为一个可绘制对象的链表。
2. 所有的绘制过程都需要基础模板数据和绘制手法的参与。
3. 基础模板数据和绘制手法都从资源模板库中提取，而资源模板库是在系统加载时创建。

#### 2.2.2主要特点

1. **绘制数据与绘制手法相互分离**。

绘制系统部分使用了将绘制数据与绘制手法相分离的处理方式。这样两边都取得了非常大的灵活度，而且相互干扰也减少很多。在常规的绘制要求下，这种方式的优势非常明显。

1. **绝大多数形体可以使用基本体的变换来实现**。

大量采用数据模板结合变换矩阵（缩放，旋转和平移）来实现世界物体的展示过程。这样可以节约大量数据内存消耗。而且模板数据可以在应用程序初始化后就可以装入到内存中。

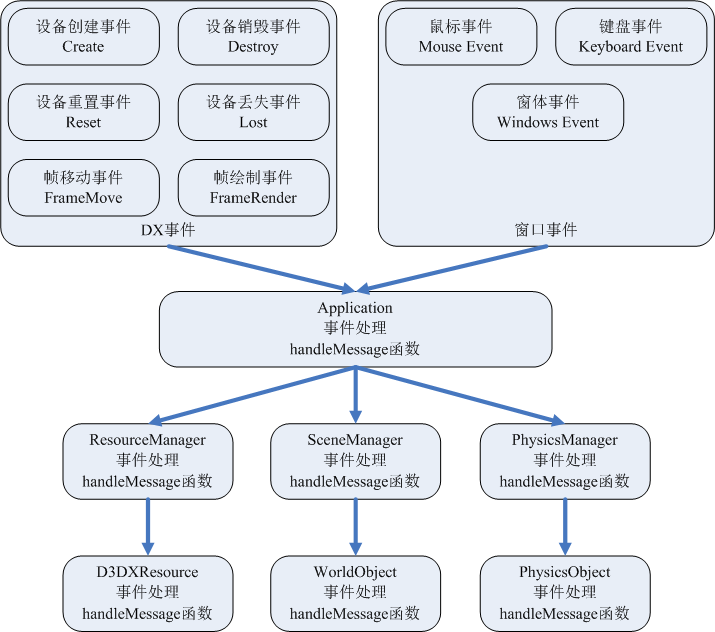
1. **可以随时调整绘制手法，而且这种调整与几何数据无关**。

对于绘制手法的调整可以随时进行，而且可以不用再调整和编写代码。这种调整且一般与实际几何体关系不大。

### 2.3事件主线

#### 2.3.1主线图示

下图为系统的事件主线图：



1. 事件可以分为两类：DX时间和窗口事件。
2. 所有的事件都通过应用控制器统一分发至各个管理器；再有管理器分配至下面各个处理对象。
3. 所有对象根据实际条件情况，对自己感兴趣的时间进行处理。

#### 2.3.2主要特点

1. **所有事件接口采取统一标准，与所采用框架无关**。

相关对象只需要处理针对性的事件即可，无需关心外围框架。因此经过封装以后，可以适用于简单的DXUT接口，也可以适合复杂的MFC框架接口。

1. **相关事件之间完全互斥，也不需要多线程模式**。

DX事件不会同时发生，而且有严格的先后次序，不存在多线程的问题。因此比较容易理清相互之间的次序关系。

1. **多数情况下，各个对象自行维护相关数据与操作的关系**。

通过场景对象树，可以将消息发送至各个对象。每个对象可以根据自己的需要独立维护自己的事件。

## 3基础数据模块

### 3.1项目4DBasic

|  |  |  |  |
| --- | --- | --- | --- |
| **目录名** | | **文件名** | **备注** |
| Base | | 4DBase | 类注册信息描述。 |
| 4DBaseMap | 类注册信息表描述。  用于检查子类与父类之间的派生关系。  主要用于D3DXResource模块。 |
| Lock | | 4DLock | 基于Critical Section的多线程同步锁。 |
| Print | | 4DPrint | 基于C的标准IO输出。 |
| runnable | | SimpleThread | 简单多线程定义。 |
| 4DRunnable | 可运行对象基本定义。 |
| time | | SimpleExpiration | 简单超时定义。 |
| SimpleFlowmeter | 简单流量计定义。 |
| 4DTime | 简单时间定义。 |
| type | | BoolType | 布尔类型定义。 |
| CharType | 字符类型定义。 |
| FloatType | 浮点类型定义。 |
| IntegerType | 整数类型定义。 |
| ObjectType | 指针对象定义。 |
| 4DType | 常规定义文件。 |
|  |  |  |  |
|  |  |  |  |

### 3.2项目4DContainer

|  |  |  |  |
| --- | --- | --- | --- |
| **目录名** | | **文件名** | **备注** |
| hash | | HashTreeContainer | 哈希树容器。 |
| HashTreeNode | 哈希树节点。 |
| hashlist | | HashListContainer | 哈希列表容器。 |
| keylist | | KeyListNode | 关键字链表节点。 |
| KeyListNodeContainer | 关键字链表容器。 |
| List | | ListNode | 链表节点。 |
| ListNodeContainer | 链表容器。 |
| ListNodeEnumeration | 链表枚举器。 |
| node | | 4DNode | 节点对象。 |
| queue | | ListQueueContainer | 基于链表的队列容器。 |
| recycle | | RecycleBin | 通用对象回收站结构。  目前主要用于字符串管理和碰撞检测管理部分。 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

### 3.3项目4DString

|  |  |  |  |
| --- | --- | --- | --- |
| **目录名** | | **文件名** | **备注** |
| normal | | NormalString | 普通字符串对象。  支持Unicode。 |
| recycle | | StringRecycleBin | 字符串回收站。 |
|  |  | 4DString | 通用字符串对象。 |
|  |  |  |  |

## 4三维绘制模块

### 4.1项目DXUTCore

DXUT核心模块。

属于微软提供的标准模块，这里不做详细介绍。

### 4.2项目DXUTOptional

DXUT可选模块。

属于微软提供的标准模块，这里不做详细介绍。

### 4.3项目4DMisc

从微软提供的标准例子中抽取的各种功能组合而成。这里不做详细介绍。

### 4.4项目4DMath

本项目提供了标准的矩阵、矢量和四元数运算。

保证DX接口和其他第三方接口均按照此数学标准进行接入。

|  |  |  |  |
| --- | --- | --- | --- |
| **目录名** | | **文件名** | **备注** |
| matrix | | 4DMatrix | 4×4矩阵对象。 |
| quaternion | | 4DQuaternion | 四元数对象。 |
| vector | | 4DVector | 3元素矢量对象。 |
|  |  | 4DMath | 常规数学定义文件。 |
|  |  | ConstantValue | 数学常量定义文件。s |
|  |  |  |  |
|  |  |  |  |

### 4.5项目4DRenderable

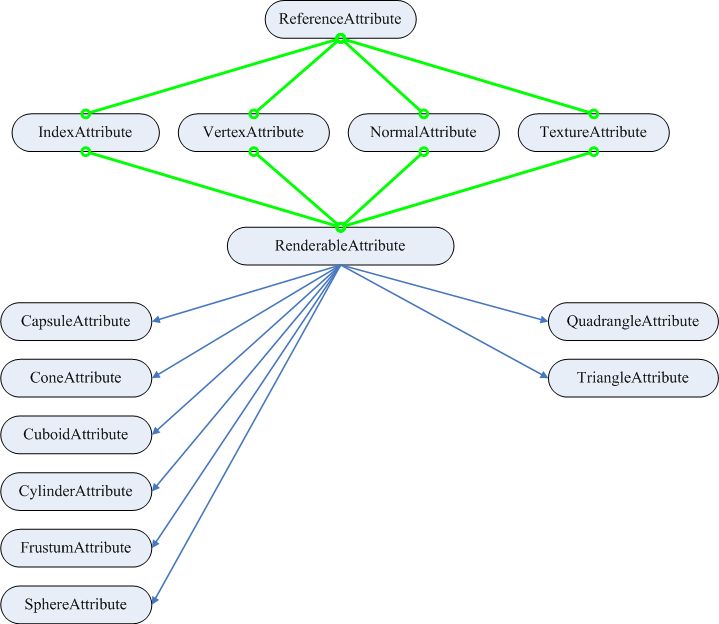
这些基本形体对应着物理仿真引擎所能支持的基本形体。

|  |  |  |  |
| --- | --- | --- | --- |
| **目录名** | | **文件名** | **备注** |
| attribute | | IndexAttribute | 索引属性。 |
| NormalAttribute | 法矢属性。 |
| ReferenceAttribute | 引用标志属性。  如果属于引用的，则不进行有关内存分配或者删除处理。 |
| TextureAttribute | 纹理坐标属性。 |
| VertexAttribute | 定点属性。 |
| primitive | capsule | CapsuleAttribute | 胶囊体属性。 |
| cone | ConeAttribute | 锥体（圆锥和棱锥）属性。 |
| cuboid | CuboidAttribute | 矩形体属性。 |
| SliceAttribute | 薄板体属性。 |
| cylinder | CylinderAttribute | 柱体（圆柱和棱柱）属性。 |
| frustum | FrustumAttribute | 台体（圆台和棱台）属性。 |
| quadrangle | QuadrangleAttribute | 四边片属性。 |
| sphere | SphereAttribute | 球体属性。 |
| HemisphereAttribute | 半球体属性。 |
| triangle | TriangleAttribute | 三角片属性。 |
|  |  | PrimitiveAttribute | 基本体属性对象。 |
|  |  | RenderableAttribute | 可绘制属性对象。 |
|  |  |  |  |

目前Newton物理引擎可以支持胶囊体、圆锥体、矩形体、圆柱体、球体和凸体共六种基本体。棱锥、台体、半球都可以解释为凸体。而四边片、三角片不是体，仅用于绘制部分。

这些几何体中，大部分仅依靠缩放矩阵就可以控制，少量需要临时变更数据。例如：胶囊体、锥体、矩形、柱体、球体、半球体只需要缩放矩阵就可以表达所有的类似形体。但是台体则不能完全使用缩放矩阵表示。

因此系统中完全通过程序固化了这些基本形体的基本数据。其他模块随时可以拷贝相关数据。物理模块需要的基本形体数据以及绘制模块需要的基本形体数据均由此部分提供。



### 4.6项目D3DXResource

该项目定义了DX资源库。

DX资源主要由三个部分组成：（1）特效手法；（2）纹理贴图；（3）可绘制对象。

在可绘制对象中，又可以细分为：（1）线框对象；（2）网格对象；（3）基本体对象（由离散三角片组成）。

|  |  |  |  |
| --- | --- | --- | --- |
| **目录名** | | **文件名** | **备注** |
| application | manager | DeviceManager | D3DX设备管理器。 |
| RenderableManager | 临时绘制对象管理器。  所管理的对象是临时资源。  当引用消失时，则自动从管理器中消失。 |
| ResourceManager | 资源对象管理器。  锁管理的对象是永久资源。 |
| resource | ResourceApplication | 资源相关的应用处理。 |
| viewport | ViewPortApplication | 视口相关的应用处理。 |
| object | | D3DXEvent | D3DX事件对象。 |
|  | | D3DXObject | D3DX通用对象。 |
| resource | effect | D3DXEffect | D3DX手法对象定义。 |
| pick | ControlBall | 控制球对象。 |
| PickIntersection | 拾取求交对象。 |
| PickRay | 拾取线。  拾取线垂直于屏幕方向。 |
| PickResult | 拾取结果对象。 |
| texture | D3DXTexture | D3DX纹理对象。 |
|  | D3DXResource | D3DX资源对象。 |
|  | ResourceName | 全局资源名称。 |
| renderable | attribute | D3DXEffectAttribute | D3DX绘制手法属性。 |
| D3DXIndexAttribute | D3DX索引属性。 |
| D3DXPickAttribute | D3DX拾取属性。 |
| D3DXVertexAttribute | D3DX定点属性。 |
| D3DXVertex | D3DX常用定点定义。 |
| linestrip | D3DXLinestrip | D3DX线框对象。 |
| Axis0 | 线框坐标架对象。 |
| mesh | D3DXFileMesh | D3DX网格文件对象。 |
| primitive | D3DXPrimitive | D3DX基础体对象。 |
|  | D3DXRenderable | D3DX可绘制对象。 |
| viewport | | ModelViewPort | DXUT模态视口对象。 |
| PersonalViewPort | DXUT人物视口对象。 |
| ViewPort | D3DX视口对象。 |
|  |  | D3DXUtility | D3DX工具文件。 |
|  |  | ResourceBase | 检测资源的继承情况。 |
|  |  |  |  |

在该项目中还定义了资源的管理方式，还有视口对象。

### 4.7项目4DGeometry

该项目中定义了包围盒的基本对象和操作。

|  |  |  |  |
| --- | --- | --- | --- |
| **目录名** | | **文件名** | **备注** |
| box | | SimpleBox | 矩形包围盒对象。 |
| sphere | | SimpleSphere | 球形包围盒对象。 |
| recycle |  | BoxRecycleBin | 包围盒循环回收对象。 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

### 4.8项目4DCollision

该项目定义了基于包围盒的碰撞检测对象。

|  |  |  |  |
| --- | --- | --- | --- |
| **目录名** | | **文件名** | **备注** |
| collision |  | CollisionMethod | 碰撞检测函数。 |
| object | | MultiBoxObject | 组合包围盒对象。 |
| SingleBoxObject | 单个包围盒对象。 |
| recycle | | MultiBoxObjectRecycleBin | 组合包围盒对象。 |
| SingleBoxObjectRecycleBin | 单个包围盒对象。 |
|  |  |  |  |
|  |  |  |  |

### 4.9项目4DWorld

该项目中定义了世界物体对象的所有基本属性。

|  |  |  |  |
| --- | --- | --- | --- |
| **目录名** | | **文件名** | **备注** |
| attribute | | BoxAttribute | 包围盒属性。 |
| ChildAttribute | 子对象链表属性。 |
| ControlAttribute | 可控制标记属性。 |
| IdentityAttribute | 标识属性。 |
| PositionAttribute | 空间位置属性。 |
| RelationAttribute | 相互关系属性。  保留了一些必要的指针属性。 |
| TypeAttribute | 物体类型属性。 |
|  |  | WorldObject | 世界物体对象。 |
|  |  |  |  |

世界物体对象（可绘制或不可绘制，物理或者非物理）的各种属性是最基本最通用的各种属性。

### 4.10项目D3DXRenderable

该项目定义了可绘制对象（不要求含有物理属性）的基本属性和操作方式。很多情况下很多对象不需要物理属性，只需要可绘制属性即可表达其意义。

一个基本可绘制对象应该需要支持：（1）可拾取平移；（2）可控制旋转；（3）可绘制外围框架。

|  |  |  |  |
| --- | --- | --- | --- |
| **目录名** | | **文件名** | **备注** |
| attribute | control | ControllerAttribute | 物体控制属性。 |
| ObjectController | 物体控制器。 |
| pick | PickerAttribute | 物体选择属性。 |
| sketch | JointObjectSketcher | 关节体框架绘制器。 |
| ObjectSketcher | 通用物体绘制器。 |
| Sketch0 | 0号框架绘制器 |
| Sketch1 | 1号框架绘制器 |
| SketcherAttribute | 物体框架绘制属性。 |
| object | | D3DXCompoundObject | 可绘制组合体。 |
| D3DXJointObject | 可绘制关节体。 |
| D3DXPrimitiveObject | 可绘制基本体。 |
| ColorAttribute | 基本体颜色属性。 |
| proxy | | D3DXJointObjectProxy | 关节体的绘制代理器。 |
| …… | （各种关节的绘制代理器） |
| sample | | AxisObject | 实体坐标架例子。  主要用于测试组合体的基本处理和绘制情况。 |
| PlusOBject | 一个有两个盒体组成的加号。  主要用于测试组合体的基本处理和绘制情况。 |
|  |  | D3DXRenderableObject | D3DX可绘制对象。 |
|  |  |  |  |
|  |  |  |  |

在DX环境下，所有的可绘制物体都会有一个控制器和框架绘制器。

1. 控制器主要用于控制物体的方向姿态。

选中物体后，按住Ctrl键，用鼠标拖动即可控制物体的转动。

1. 框架绘制器主要用于绘制物体的外围框架。

例如：包围盒、坐标架、主要参数方向等。

对于关节体使用了一个特别的框架绘制器。这些显示的框架主要用Debug可能存在的系统计算和显示问题。

## 5物理动力模块

### 5.1项目4DNewton

该项目是负责与Newton接口的最核心部分。负责将4DPhysics中的各种对象翻译成Newton物理库可以表达的实体。对于无法翻译的部分，其选择自动放弃。

|  |  |  |  |
| --- | --- | --- | --- |
| **目录名** | | **文件名** | **备注** |
| entity | | BoxEntity | 盒体对象。 |
| CapsuleEntity | 胶囊体对象。 |
| ConeEntity | 圆锥体对象。 |
| ConvexEntity | 凸体对象。 |
| CylinderEntity | 圆柱体对象。 |
| SphereEntity | 球体对象。 |
| PhysicsEntity | 物理实体对象。 |
| joint | classic | PhysicsClassicJoint | 经典关节体对象。 |
| …… | （各种经典关节体对象） |
| custom | PhysicsCustomJoint | 自定义关节体对象。 |
| …… | （各种子定义关节体对象） |
| manager | | BodyManager | 形体管理器。 |
| EntityManager | 实体管理器。 |
| JointManager | 关节体管理器。 |
| ShapeManager | 形状管理器。 |
| SimulationManager | 模拟管理器。 |
| PhysicsTime | 物理时钟。 |
| PhysicsWorld | 物理世界对象。 |
| PhysicsManager | 物理管理对象。 |
|  |  |  |  |

这个翻译的主要部分包括：实体和关节。一般一个翻译过程是先执行实体翻译，再执行关节体翻译。翻译除需要定义主要的类型和形状数据以外，还需要定义其姿态数据。其内部数据已经经过反复调试和确认，一般无特别需要，不用修改其内部数据。

### 5.2项目4DPhysics

该模块按照物体的物理属性进行定义。

物理物体对象（不要求具有可绘制属性）的定义遵循Newton仿真库中对基本支持物体的定义要求。主要支持：基本体、关节体和组合体。

|  |  |  |  |
| --- | --- | --- | --- |
| **目录名** | | **文件名** | **备注** |
| attribute | | ForceAttribute | 力学属性。 |
| PhysicsAttribute | 基本物理学属性。  例如：质量、转动惯量。 |
| StatusAttribute | 状态属性。  记录物理物体的状态参数。  主要是位置、速度和加速度。 |
| keeper | | StatusKeeper | 状态保存器对象。 |
| object | | PhysicsCompoundObject | 物理组合体。 |
| PhysicsJointObject | 物理关节体。 |
| PhysicsPrimitiveObject | 物理基本体。 |
| JointForceAttribute | 关节体力学属性。  一般为关节体内部的一些相对力学属性。 |
|  |  | PhysicsObject | 物理对象。 |
|  |  |  |  |

状态保存器用来保存物理物体的相关状态量。一方面可以用于粗略计算物体的相关力学属性；另外一个方面可以将物体恢复至仿真前的状况。

在不需要DX环境的情况下，完全可以独立使用该部分的模块进行纯数据仿真。

### 5.3项目4DJoint

该模块只是在WorldObject的基础上，拓展了下有关关节的相关属性定义。

|  |  |  |  |
| --- | --- | --- | --- |
| **目录名** | | **文件名** | **备注** |
| attribute | | JointAttribute | 关节属性对象。  关节所有基本参数。 |
| JointRelationAttribute | 关节关系对象。  主要用于存储父实体和子实体。 |
|  |  | WorldJointObject | 世界关节体对象。 |
|  |  |  |  |

### 5.3项目D3DXPhysics

该项目中有两个关键部分：

1. 可绘制物理体（Renderable Physics Object）的定义。
   1. 由物理体和可绘制体结合起来形成可绘制物理体。
2. 由可绘制物体组成的场景（Scene）定义。
   1. 定义了前景（Foreground）。

前景的所有物体参与相关用户交互。包括：旋转、平移、选择等等。

* 1. 定义了背景（Background）。

背景的所有物体不参与相关用户交互。但是参与绘制过程。

* 1. 前景和背景在数据处理和函数上并无区别，仅仅是在消息传递时，一些鼠标和用户消息不传递给背景。这样话，就使得背景不会参与到用户的实际交互过程中。

|  |  |  |  |
| --- | --- | --- | --- |
| **目录名** | | **文件名** | **备注** |
| scene | attribute | CollisionAttribute | 碰撞检测属性对象。 |
|  | PickedAttribute | 拾取属性对象。 |
| group | WorldGroup | 物体组对象。 |
| manager | FloorManager | 地板管理器。  用于往场景中增加地板、墙面等。 |
| ModelManager | 模型管理器。  主要用于往场景中增加各种内置模板。 |
|  | D3DXScene | 景物对象。 |
| foreground | | ForegroundScene | 前景对象。  所有前景对象可以参与用户事件互动。  例如：鼠标拾取、移动物体、旋转物体。 |
| SimpleScene | 简单前景。 |
| background | | BackgroundScene | 背景对象。  所有背景对象只参与显示，不参与用户事件互动。  目前主要就是绘制一个实体坐标架。 |
| object | | CompoundObject | 可绘制物理组合体。 |
| JointObject | 可绘制物理关节体。 |
| PrimitiveObject | 可绘制物理基本体。 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

## 6应用模块

### 6.1项目D3DXApplication

该部分主要为了集成场景（Scene）、声音（Sound）和物理（Physics）的应用，形成一个完整的应用框架体系。其他应用框架基本由此框架继承并发展而来。

|  |  |  |  |
| --- | --- | --- | --- |
| **目录名** | | **文件名** | **备注** |
|  |  | SoundApplication | 声音应用。 |
|  |  | SceneApplication | 场景应用。 |
|  |  | PhysicsApplication | 物理应用。 |
|  |  | D3DXApplication | D3DX应用框架。 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

### 6.2项目MFCApplication

该项目主要将Wnd窗口句柄封装成类似DXUT的构造方式，使得Windows的应用基本与DXUT的方式相互兼容。

这样所有的工作方式和消息机制方式基本与DXUT方式下一模一样。

|  |  |  |  |
| --- | --- | --- | --- |
| **目录名** | | **文件名** | **备注** |
| xd3d |  | …… | 在Wnd窗口中使用DirectX绘制。 |
|  |  | MFCApplication | MFC应用框架。 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |