# 客户端 FB\_Client

## 游戏程序主循环

### 游戏逻辑处理（每帧执行一次）

#### 预加载充值界面

#### 清理用于保存处理完成渲染对象（模型、特效）的哈希表

#### 刷新鼠标指针位置

#### 处理游戏场景

1. 处理电影
2. 处理网络相关逻辑
3. 处理网络互斥窗口属性
4. 执行踢人检测
5. 更新PVP记录
6. 处理游戏网络连接
7. 处理操作状态
8. 处理角色跟随
9. 更新小喇叭
10. 更新英雄状态
11. 更新活动任务
12. 更新帮助信息
13. 判断是否隐藏玩家角色
14. 处理玩家角色的宝宝
15. 处理地图
16. 处理游戏对象
17. 处理游戏逻辑
18. 处理传送点
19. 处理地图特效
20. 将渲染对象排序

* 诺亚的2D渲染对象（地图渲染对象）排序说明

在诺亚中，2D对象中定义了3种与排序有关的属性：

* 图层 Layer
* 取值为[0，250]中的整数
* 图层值越小，显示在越前面
* 基线 BaseLine
  + 取值范围为float
  + 基线值越小，显示在越前面
  + 基线可以理解为2D图片的底边，想象在一个3D场景中垂直摆放一张2D图片，当图片底边所处的位置越低时，说明处于场景的越前方。
  + 基线默认值为资源图片的底边位置，可以由美术进行调整
  + 基线最初是为了美术而提出的概念，它与遮挡格的作用冲突，可以在后期考虑剔除
* 遮挡格
  + 诺亚中的地图是分块的，每个块为一个格子。遮挡格可以理解为2D图片所表示的物体在地图中所处的格子位置
  + 遮挡格位置越低，显示在越前面
  + 遮挡格是为了让玩家在家园系统中摆放物品而设计的属性，因为基线的概念不利于玩家理解与操作。

1. 为背景深度fBackDepth赋初值，使它等于背景深度的最大值BackSceneBack。背景深度值越大，对象越先被绘制，即处于越底层
2. 遍历游戏地图中的2D渲染对象集合m\_Map.m\_GameMap.m\_Renders，对象类型为CObjData（从图层的250层遍历到第50层）

* 如果当前渲染对象为空，则跳入下一次循环
* 如果当前对象有遮挡格或处于第0层（这里有bug，应该是层数小于或等于50），则结束循环（地图中的2D对象在存储时已经是有序的）
* 如果当前对象所处的背景深度fBackDepth小于背景深度最小值BackScenePre，则输出错误信息，并使fBackDepth = BackScenePre
* 设置对象精灵为不透明，并使它的深度值等于fBackDepth
* 绘制对象精灵
* 将当前渲染对象从集合中剔除
* 令fBackDepth = fBackDepth – fBackDepthEach，使下一个对象的深度值减小0.1f，从而令它显示在更前面

1. 遍历场景中的3D渲染对象集合m\_ReadyObj，对象类型为CFBGameMap\_RUnknown

* 如果对象对应的绘制键ObjDrawKey的场景层大于或等于主场景层eSL\_MainScene，就结束循环
* 如果当前对象所处的背景深度fBackDepth小于背景深度最小值BackScenePre，则输出错误信息，并使fBackDepth = BackScenePre
* 设置渲染对象的深度值为fBackDepth
* 令fBackDepth = fBackDepth – fBackDepthEach
* 将渲染对象加入到m\_Renders集合中（在GBClient\_Global中定义）
* 将当前渲染对象从地图渲染对象集合中剔除

1. 继续遍历地图中的2D渲染对象集合m\_Map.m\_GameMap.m\_Renders，对象类型为CObjData（从图层的第50层遍历到第1层）

* 如果当前渲染对象为空，则跳入下一次循环
* 如果当前对象处于第0层，则结束循环
* 设置对象精灵为不透明，并使它的深度值等于FrontSceneBack
* 绘制对象精灵

1. 设置前景深度值fDepth = FrontSceneBack – 1.0f
2. 设置前景深度间隔fEach为前景最大深度差除以渲染集合中的2D和3D对象个数（这一行可以放到步骤h中，避免无效计算）
3. 如果当前处于庭院场景中

* 设置前景深度间隔fEach为前景最大深度差除以渲染集合中的2D对象个数排序节点链表m\_listSortNodes中节点（SortNode）个数的和
* 创建一个Sort3DNode链表
* 反向遍历3D渲染对象集合m\_ReadyObj，设当前对象为3DObj
  + 创建一个Sort3DNode（这里为m\_fHighLayer赋的初始值1与默认构造函数重复，可以考虑删掉）对象3DNode
  + 遍历m\_listSortNodes，获取当前SortNode中索引所指向的2D渲染对象2DObj
    - 遍历2DObj的遮挡格
      * 如果3DObj所在的网格在当前遮挡格的后面
        + 如果3DNode坐在的庭院层（HighLayer）小于或等于2DObj所在的庭院层，则设置它的庭院层为2DObj的庭院层加0.1f
        + 退出当前遍历
      * 如果3DObj所在网格挡住了当前遮挡格
        + 将当前的SortNode迭代器加入到3DNode的后方SortNode迭代器链表中
        + 退出当前遍历
  + 如果Sort3DNode链表不为空且3DNode的庭院层次小于或等于Sort3DNode头节点的庭院层次，设置3DNode的庭院层次为头节点层次加0.1f（这里似乎有逻辑问题，头节点是不是应该改为尾节点？）
  + 遍历3DNode的后方SortNode迭代器链表，将所有后方的节点之间的HighLayer值向后推，两两间的差值调整为1f（这里似乎有问题，可能导致HighLayer重复）
  + 将3DNode加入到Sort3DNode链表的表头中
* **遍历m\_listSortNodes链表，将其中的元素加入到有序Hash表中**
* **将Hash表中的2D渲染对象与Sort3DNode链表中的对象按照顺序加入到m\_Renders集合中**

（上述加黑字体的两个步骤为大致描述）

（步骤g的实现方法有很大的改进空间，有时间可以思考如何重写）

1. 如果当前不处于庭院场景中

* 重新遍历地图的2D渲染对象集合m\_Map.m\_GameMap.m\_Renders，对象类型为CObjData（从图层的第50层遍历到第1层），设当前对象为2DObj
  + 如果2DObj为空，则跳入下一次循环
  + 如果2DObj处于第0层，则结束循环
  + 遍历2DObj的遮挡格
    - 遍历场景中的3D渲染对象集合m\_ReadyObj，对象类型为GFBGameMap\_RUnknown，设当前对象为3DObj
      * 如果3DObj所在的网格在当前遮挡格的后面
* 依次设置m\_ReadyObj第一个对象至3DObj之间的所有对象的深度值为fDepth，并且每次赋值后使fDepth减小fEach
* 将m\_ReadyObj第一个对象至3DObj之间的所有对象加入到m\_Renders集合中，同时将它们从m\_ReadyObj集合中移除
  + m\_ReadyObj集合中的3D对象已经按y轴坐标排好序了，y坐标越大，在集合中的索引越靠前
  + 诺亚的2D地图坐标原点位于左上角，x轴正方向水平向右，y轴正方向垂直向下

Y轴

X轴

* + 诺亚的3D场景坐标系中，x轴正方向水平向右，y轴正方向垂直屏幕向里，z轴正方向垂直向下

Z轴

Y轴

X轴

* 如果开启了重叠物体透明修正，则设置2DObj中的精灵为半透明（若处于家园场景中，则不设为半透明）（这后面似乎可以加一个break）
* 如果没有开启重叠物体透明修正，则设置2DObj的精灵为半透明（若处于家园场景中，则不设为半透明）
* 设置2DObj精灵的深度值为fDepth
* 将2DObj的精灵加入到m\_Renders集合中
* 令fDepth = fDepth – fEach
* 将2DObj从m\_Map.m\_GameMap.m\_Renders集合中移除
* 将m\_ReadyObj集合中剩余的3D渲染对象加入到m\_Renders集合中，并更新它们的深度值
* 再次遍历地图的2D渲染对象集合m\_Map.m\_GameMap.m\_Renders，对象类型为CObjData（遍历第0层）
  + 如果当前对象为空，跳入下一次循环
  + 如果没有开启重叠物体透明修正，则设置当前对象的精灵为半透明（若处于家园场景中，则不设为半透明）
  + 设置当前对象精灵的深度值为fDepth
  + 令fDepth = fDepth – fEach
  + 将当前对象的精灵加入到m\_Renders集合中

1. 绘制渲染对象 Process\_Game\_DrawRenderObj
2. 根据鼠标在屏幕中的位置，获取鼠标在场景地图中的位置
3. 遍历m\_Renders集合中的所有渲染对象

* 根据鼠标位置绘制对象（高亮绘制鼠标指向的对象）（将判断参数从绘制函数中分离，判断成功后不对后面的对象判断是否会提高效率？）（渲染对象的派生类很多，下面仅对CFBClient\_Animal类的Draw函数进行分析）
* 重心修正
* 绘制前的准备工作
  + 这个函数PreDraw包含无意义的参数dwPassTime
  + 动作处理
  + 位移转向处理
  + 影子处理（这里只对简单影子进行绘制）
* 判断是否根据GM隐藏英雄命令绘制英雄
  + 这段逻辑放在绘制函数里似乎不太好，而且似乎会导致一个bug：在简单阴影模式下，开启GM隐藏英雄命令时会显示英雄阴影，而不会显示实体
* 检查对象被控制状态
  + 如果对象处于被控（处于坐骑上且不能控制坐骑）状态m\_bCtrlQC为真，但父坐骑（控制坐骑的玩家或NPC）不存在，则设置父坐骑为空对象（m\_bCtrlQC被设为假）
  + 这段逻辑放在这里似乎不合适
* 如果对象处于被控制状态（m\_bCtrlFly或m\_bCtrlQC为真），则不进行绘制（return FALSE）
  + 可以设置一个是否绘制对象的BOOL型变量isVisible，当对象处于载具中时设置为假，则不对对象进行绘制。将逻辑状态和绘制函数放在一起，代码结构容易越来越混乱。
* 如果没有播放电影
* 设置m\_fGridDepthTmp的值为第一个非场景对象的渲染对象的深度值
* 若当前渲染对象的类型为玩家、英雄、NPC、怪物或宝宝（进入switch）
  + 设置bNeedHandle为真
  + 如果子坐骑链表不为空，判断坐骑控制网格是否与鼠标相交
    - 若相交，设置bNeedHandle为假
  + 如果当前渲染对象不可见且可选择且被选中
    - 若当前对象ID等于英雄ID，则设置bNeedHandle为假
    - 如果飞船列表不为空，判断飞船控制网格是否与鼠标相交
      * 若相交，设置bNeedHandle为假
    - 若bNeedHandle为真
      * 判断渲染对象的网格是否与鼠标相交
      * 如果英雄的移动受到限制且渲染对象不是玩家，则跳入下一次渲染对象循环（这里用continue对for循环生效，而不对switch生效）
      * 如果渲染对象是存活的或者是玩家对象
        + 更新攻击序列
        + 设置bTurret（是否为炮塔）为假
        + 如果动物选择模板为空，设置动物选择模板为当前渲染对象并结束switch；否则，如果场景ID为DOUBLESNAKE\_TURRET，则令炮台优先于怪物被选中（选取优先级为：NPC > 炮台 > 怪物），此时若有对象被选中，则设置bTurret为真
        + 如果bTurret仍为假

如果ALT键被按下

如果当前渲染对象的m\_eAltAttackType小于选中对象的m\_eAltAttackType，则设置当前渲染对象为选中对象，并跳如下一次循环（for循环）

否则，如果选中对象不是按网格选中的（m\_bMeshSelect为假）且当前对象时按网格选中的，则使用当前对象替换选中对象

如果ALT键未按下

如果当前渲染对象的m\_eAttackType小于选中对象的m\_eAttackType，则设置当前渲染对象为选中对象，并跳如下一次循环（for循环）；

否则，如果当前渲染对象的m\_eAttackType等于选中对象的m\_eAttackType，如果当前渲染对象是按网格选中的或当前渲染对象与选中对象都不是按网格选中的，则使用当前对象替换选中对象

（这个步骤的大多数代码用于处理选择对象而不是绘制对象，可以考虑将选取对象逻辑抽出来组成单独的函数）

1. 处理游戏道具
2. 处理选定对象
3. 处理光标
4. 处理场景效果
5. 绘制网格（测试用）
6. 绘制遮蔽（测试用）
7. 绘制对象名称
8. 绘制悬浮文字
9. 移动操作回滚
10. 检测FPS

## 帧 Frame

### FBClient\_Global

* 用于控制游戏客户端程序全局信息的类型

### FBClient\_Global::Sort3DNode

* 用于为场景中3D渲染对象排序而定义的数据结构
* 包含了一个指向CFBGameMap\_RUnknown对象的指针
* 包含了一个由SortNode迭代器组成的链表
* 包含了一个浮点型对象m\_fHighLayer

## 对象 Object

### CFBClient\_Animal

* 继承自CFBGameMap\_RUnknown
* 场景角色的基本类型
* 包含一个CFBGraphics\_3DSprite对象用于绘制阴影
* 包含一个点击人物触发的事件ID
* 包含两个指向CFBScene\_Model对象的强指针，一个是人物模型，一个是坐骑模型
* 包含显示Hp槽与Mp槽的精灵（CFBGraphics\_Sprite）
* 包含用于控制动画的属性
* 包含角色状态属性（装备列表、是否隐身、是否无敌等）
* 包含待显示的浮动经验值与银币列表
* 包含一个角色全身特效（CFBClient\_Effect）数组

# 公共 Common

## 障碍 Barrier

### CBarrier

* CPos PixelToGrid(const CPos& Pixel)函数
* 这个函数的优化版本使用位运算代替了除法运算，但没有考虑符号问题

# 游戏地图 FB\_GameMap

## 渲染对象

CFBGameMap\_RSceneObj

CFBGameMap\_RUnknown

继承

### CFBGameMap\_RUnknown

* 定义游戏地图中渲染对象的抽象类
* 包括的主要属性有：
  + 渲染对象类型m\_wType
  + 深度m\_fDepth
  + 所处的场景层 m\_eSceneLayer
* 声明了Draw、Process等与绘制更新对象有关的虚函数

### CFBGameMap\_RSceneObj

* 继承自CFBGameMap\_RUnknown
* 包含了一个指向CFBGraphics\_3DSprite（详情见Graphics的渲染对象部分）的指针，用于实现对象绘制
* 包含了一个Set3DSprite函数，用于调用Sprite的Set函数
* 包含的其他主要属性有：
  + 对象所占用的区域 m\_Rect
  + 是否使用像素着色器 m\_bPixShader
  + 是否是家园对象 m\_bHomeObj
  + 是否在庭院表面 m\_bGardenSurface
  + 颜色 m\_Color
  + 是否有灯光 m\_bLight
  + 是否有高光 m\_bHighLight
  + 是否透明 m\_bAlpha （优先按灯光绘制，如果没有灯光，才会判断是否有高光，如果没有高光，才会判断是否透明）
* DrawWithOutSet函数
* 包含了无用参数
* 解决模型选中高亮显示到家具前面问题的方式很奇怪（通过增加属性判断的方式，过于特例化，不能适用普遍情况（初步猜测，不能确定，需要再确认））
* 灯光、高光、透明判断可以用枚举，不需要设置3个整型布尔值变量

## 数据

CObjData

CFB\_MAP

组合

CMusicData

CEffectObjData

CRegion

组合

组合

组合

### CObjRenderKey

* 存储与比较游戏地图中渲染对象绘制优先级信息的结构体，通过Hash表与对象绑定
* 包含的主要属性有
  + 资源ID m\_dwResID
  + 对象位置 m\_Pos
  + 对象所处的图层 m\_bLayer
  + 对象所处的动态图层 m\_fHighLayer

### CObjData

* 存储游戏地图中渲染对象属性的结构体
* 包含的主要属性有
  + 全局对象ID m\_objID
  + 对象在数组中下标 m\_dwObjID
  + 对象（图像）左上角在场景中的像素坐标 m\_Pos
  + 对象的根坐标（不保存到map文件中） m\_RootPos
  + 对象所在的层 m\_bLayer
  + 庭院专用的排序层 m\_HighLayer
  + 遮挡格个数 m\_dwShelter
  + 遮挡格在3D场景中的坐标数组 m\_Shelters
  + 对象精灵 m\_Sprite （与当前结构体包含重复的数据）
  + 其他属性，如位置、宽高、颜色等

### CMusicData

* 存储音乐资源数据的结构体
* 包含的主要属性有：
  + 音乐资源ID m\_dwMusicID
  + 音乐类型（背景、传送门） m\_bType
  + 声源位置 m\_Pos
  + 声音有效半径 m\_fRadius
  + 路径大小 m\_bPathSize
  + 资源路径 m\_sPath

### CEffectObjData

* 存储特效属性的结构体
* 包含的主要属性有：
  + 特效ID m\_dwObjID
  + 位置 m\_Pos
  + 方向 m\_fDir
  + 路径大小 m\_bPathSize
  + 资源路径 m\_sPath
  + 宽度与高度 m\_dwWidth与m\_dwHeight
  + 前后位置 m\_OffSetY

### CRegion

* 定义区域的结构体
* 包含的属性有：
  + 区域对象（编号） m\_dwRegionObj
  + 包含区域对象（编号）的动态数组 m\_RegionObjs

### CFB\_MAP

* 存储地图数据的结构体
* 包含的主要属性有：
  + 每个区域的宽度和高度 m\_dwRegionWidth和m\_dwRegionHeight
  + 包含区域的列数与行数 m\_dwColumn和m\_dwRow
  + 地图中的对象个数 m\_dwObj
  + 地图中的音乐个数 m\_dwMusic
  + 地图中的特效个数 m\_dwEffectObj
  + 区域数组 m\_Regions
  + 对象数组 m\_Objs
  + 音乐数组 m\_Musics
  + 特效数组 m\_EffectObjs

## 地图

SortNode

::

CFBGameMap\_Map

组合

### CFBGameMap\_Map

* 定义了游戏场景，它是一个地图的实例
* 包含了地图数据m\_MapData、地图中的渲染对象m\_Renders、地图中的区域块m\_LoadRegions、场景宽高等属性
* 定义了一个SortNode链表

### CFBGameMap\_Map::SortNode

* 用于对地图中对象数据CObjData进行排序的数据结构，通过索引获取对象数据
* 包含了指向节点前方SortNode迭代器链表的迭代器和指向节点后方SortNode迭代器链表的迭代器

# 图像 FB\_Graphics

## 渲染引擎 RenderEngine

### CFBGraphics\_FX

* 着色器程序，支持一个Technique

## 渲染对象 RenderObj

CFBGraphics\_3DSprite

组合

CFBGameMap\_RenderUnit

### CFBGraphics\_Mesh

* 存储模型的网格数据
* 提供了加载与存储模型网格的方法
* 包含主要属性有UV动画的相关属性（m\_nSegs、m\_pSegUV、m\_SegSize）、子网格顶点数据m\_pSub等

### CFBGraphics\_Mesh::\_Skin

* 定义顶点蒙皮相关属性：影响顶点的骨骼索引数组nIndex（诺亚中一个顶点最多被两个骨骼影响）与骨骼权重数组fWeight

### CFBGraphics\_Mesh::\_Target

* 存储渐变动画相关数据
* 包含一个顶点位置数组与一个顶点法线数组

### CFBGraphics\_Mesh::\_Sub

* 存储网格的顶点数据：位置、法线、颜色、UV坐标
* 在\_Sub中有两份网格顶点数据，一份是用于存储原始顶点数据，一份用于向显存传递数据

### CFBGraphics\_RMesh

* 在CFBGraphics\_Mesh的基础上拓展出来的类型（组合关系）
* 提供了绘制与更新网格数据的方法
* 包含的主要属性有:
  + 指向CFBGraphics\_Mesh的指针m\_pMesh
  + 渲染单元（CFBGraphics\_RenderUnit）数组m\_RenderUnits
  + 包围盒顶点数组m\_BBox

### CFBGraphics\_RMesh::\_Sub

* 存储网格的顶点数据：位置m\_Pos和法线m\_Normal

### CFBGraphics\_RO

* 存储与模型绘制有关的数据
* 提供了绘制与更新网格数据的函数
* 包含的主要属性有：
  + 网格数据（CFBGraphics\_RMesh）指针m\_pRMesh
  + 粒子数据（CFBGraphics\_RPtcl）指针m\_pRPtcl
  + 曲线数据（CFBGraphics\_RSpline）指针m\_pRSpline
  + 顶点数据静态文件（CFBGraphics\_CFile）指针m\_pCFile
  + 材质数据（CFBGraphics\_Material）指针m\_pMaterial
  + 网格渲染状态文件（CFBGraphics\_ShaFile）指针数组m\_pvecMeshShaFile
  + 粒子渲染状态文件（CFBGraphics\_ShaFile）指针数组m\_pvecPtclShaFile
  + 曲线渲染状态文件（CFBGraphics\_ShaFile）指针数组m\_pvecSplineShaFile

## 资源 Resources

### CFBGraphics\_ShaFile

* 控制与渲染状态有关的数据
* 主要提供了控制纹理的方法
* 包含的主要属性有：
  + 着色器程序（CFBGraphics\_FX）对象指针m\_pFX
  + 是否显示双面纹理m\_b2Side
  + 渲染层（CFBGraphics\_Layer）数组m\_Layer、
  + 渲染模式m\_nRenderMode
* 这个类与着色器的概念相差比较大，容易让人产生混淆，感觉叫RenderStateFile之类的名字更好

# 场景 FB\_Scene

## 对象 Object

CFBScene\_BindNode

CFBScene\_Obj

组合

CFBScene\_3DObj

继承

CFBScene\_ClipNode

组合

CFBScene\_Model

继承

Plug

Plug

组合

组合

CFBScene\_EffectNode

CFBScene\_Effect

继承

组合

组合

CFBScene\_EffectMNode

继承

CFBScene\_SimpleModel

CFBScene\_GeneModel

CFBScene\_EquipedModel

CFBScene\_HumenModel

继承

继承

继承

继承

CFBScene\_BindTree

CFBScene\_ClipTree

组合

组合

CFBScene\_Manager

组合

组合

### CFBScene\_BindNode

* 构成场景树的基本单元
* 表示场景对象的平移、旋转与缩放变换
* 表示场景对象之间的绑定关系

### CFBScene\_ClipNode

* 用于构建裁剪树

### CFBScene\_BindTree

* 场景树，用于管理场景对象之间的绑定关系

### CFBScene\_ClipTree

* 裁剪树，用于避免渲染不必要的场景对象

### CFBScene\_Manager

* 场景管理的核心，负责维护场景树与裁剪树

### CFBScene\_Obj

* 构成场景逻辑结构的基本对象
* 包含了CFBScene\_BindNode与CFBScene\_ClipNode的强指针
* 不参与渲染工作

### CFBScene\_3DObj

* 继承自CFBScene\_Obj
* 包含了变换前顶点数据RO与变换后顶点数据RC
* 实现了从CFile、SFile、AFile和MFile文件中加载场景对象的函数
* 实现了与对象渲染有关的函数
* 包含了一个<string, JointNode>哈希表，用于表示CFBScene\_3DObj去除缩放变换后的空间变换，以满足诺亚中某些特效不跟随场景对象缩放的特殊需求

### CFBScene\_EffectNode

* 继承自CFBScene\_Obj
* 包含了一个效果对象EffectObj的指针，用于指向效果数据
* 提供了渲染接口，但不负责渲染（实际渲染工作在CFBScene\_EffectMNode中进行）

### CFBScene\_EffectMNode

* 继承自CFBScene\_EffectNode，并实现了它提供的渲染接口
* 实际负责渲染特效的基本对象

### CFBScene\_Effect

* 包含了CFBScene\_EffectNode强指针的动态数组
* 通过调用CFBScene\_EffectNode提供的渲染接口，实现整个效果的绘制

### CFBScene\_Model::Plug（挂件）

* 组成模型的基本单位
* 在CFBScene\_Model中定义的结构体
* 包含了一个CFBScene\_3DObj对象，用于渲染模型
* 包含了一个<unsigned int, CFBScene\_Effect>哈希表，用于渲染特效

### CFBScene\_Model

* 继承自CFBScene\_Obj
* 包含了一个Plug动态数组，用于构成模型的各个部分
* 实现了许多与模型特效渲染有关的函数

### CFBScene\_SimpleModel

* 继承自CFBScene\_Model
* 只由一个挂件组成的简单模型

### CFBScene\_HumenModel

* 继承自CFBScene\_Model
* 人型怪物模型，包含七个挂件

### CFBScene\_GeneModel

* 继承自CFBScene\_Model
* 基因模型，包含五个挂件

### CFBScene\_EquipedModel

* 继承自CFBScene\_Model
* 玩家模型，包含十一个挂件，对应身体、头部和各个装备部位

## 渲染部分

继承

CFBScene\_BindTree

CFBScene\_ClipTree

CFBScene\_Manager

组合

组合