# Описание концепции ResourceManager

Функциональность ResourceManager, по аналогии с VisibilityManager предоставляется отдельной динамической библиотекой, экспортирующей класс CResourceManager. CResourceManager может взаимодействовать одновременно с несколькими объектами CVisibilityManager, в целях поддержки многоканальной визуализации. Функции CResourceManager заключаются в определении необходимости асинхронной загрузки/выгрузки графических ресурсов.

Графические ресурсы и управляющие классы представляются следующей иерархией наследственности:

C3DBaseObject

C3DBaseObjectManager

(ex. MeshTree)

C3DBaseResource

C3DBaseMesh

C3DBaseMaterial

C3DBaseTexture

C3DBaseManager

C3DMeshManager

C3DMaterialManager

C3DTextureManager

На следующей схеме изображен механизм взаимодействия CResourceManager с пользователем и VisibilityManager:

CResourceManager

AddVisibilityManager()

Update(float delta)

VisibilityManager1

VisibilityManager2

Менеджеры графических ресурсов

Загрузка/выгрузка ресурсов, подсчет ссылок

Графические ресурсы

Графические ресурсы сцены имеют следующую иерархию:

Object1

Object2

ObjectN

Mesh1

Mesh2

MeshN

FS1

FSN

FS1

FSN

FS1

FSN

Material1

MaterialN

SubMaterial1

SubMaterialN

Technique1

Technique2

TechniqueN

Texture1

Texture2

TextureN

Material2

Объекты сцены. Управляющий класс: C3DBaseObjectManager

Схема иерархии графических объектов

Меши. Управляющий класс: C3DMeshManager

Фейс-сеты (Facesets). Набор граней, ссылающийся на материал.

Материалы. Бывают двух типов: стандартные и мультиматериалы.

Управляются с помощью класса C3DMaterialManager

Rendering techniques.

Текстуры. Управляющий класс: C3DTextureManager.

## Структуры для взаимодействия с ResourceManager:

enum E3DResourceType

{

C3DRESOURCE\_OBJECT,

C3DRESOURCE\_MESH,

C3DRESOURCE\_MATERIAL,

C3DRESOURCE\_FACESET,

C3DRESOURCE\_TECHNIQUE,

C3DRESOURCE\_TEXTURE

};

// базовый ресурс

struct C3DBaseResource

{

E3DResourceType GetType() const = 0;

// получить количество ссылок видимых рсурсов на данный ресурс

virtual size\_t GetVisibleRefCount() const { return \_visibleRefCount; }

// получить родительские ресурсы

virtual void GetParentResources(std::vector<C3DBaseResource\*>& out\_vecParentResources) const = 0;

// получить дочерние ресурсы

virtual void GetChildResources(std::vector<C3DBaseResource\*>& out\_vecChildResources) const = 0;

// получить указатель на менеджер, управляющий данным ресурсом

virtual C3DBaseManager\* GetManager() const = 0;

private:

size\_t \_visibleRefCount;

void AddVisibleRef();

void ReleaseVisibleRef();

friend class CResourceManager;

};

// базовый менеджер графических ресурсов

struct C3DBaseManager

{

// Запросить загрузку ресурса

virtual void RequestLoadResource(C3DBaseResource\*) = 0;

// запросить выгрузку ресурса

virtual void RequestUnloadResource(C3DBaseResource\*) = 0;

};

// базовый класс менеджера объектов

struct C3DBaseObjectManager : public C3DBaseManager

{

// получить список объектов по заданному баунд-боксу

virtual void GetObjectList(const D3DXVECTOR3& bboxMin, const D3DXVECTOR3& bboxMax, std::vector<C3DBase3DObject\*>& out\_vecObjects) = 0;

};

// базовый класс менеджера объекта в сцене

struct C3DBaseObject : public C3DBaseResource

{

// Все 3D объекты должны будут возвращать Баунд-Бокс. Причем, если объект - точка, а не меш, то

// пусть вернет одинаковые значения в out\_vBBMin и out\_vBBMax.

virtual void GetBoundBox(D3DXVECTOR3\*\* ppBBMin, D3DXVECTOR3\*\* ppBBMax) = 0;

// Получить матрицу трансформации

virtual D3DXMATRIX\* GetWorldTransform() = 0;

// Функция должна возвращать: включена-ли проверка размера объекта на экране

virtual bool IsMinimalSizeCheckEnabled() const = 0;

};

// базовый класс меша

struct C3DBaseMesh : public C3DBaseResource

{

// Получить список Фейс-Сетов, используемых в меше

virtual void GetFaceSets(vector<C3DBaseFaceSet\*>& out\_vecFaceSets) const = 0;

};

// базовый класс фейс-сета

struct C3DBaseFaceSet : public C3DBaseResource

{

// получить родительский меш

virtual C3DBaseMesh\* GetParentMesh() = 0;

// получить ссылку на материал

virtual C3DBaseMaterial\* GetMaterialRef() = 0;

};

// базовый материал

struct C3DBaseMaterial : public C3DBaseResource

{

// Функция используется в процессе рендеринга [для взаимодействия процесса рендера с Visman]

virtual void AddVisibleFaceSet(C3DBaseFaceSet\*) = 0;

// Получить набор техник

virtual void GetTechniques(std::vector<C3DBaseTechnique\*>& out\_vecTechniques) const = 0;

// получить список текстур

virtual void GetTextures(std::vector<C3DBaseTexture\*>& out\_vecTextures) const = 0;

};

struct C3DBaseTechnique : public C3DBaseResource

{

};

enum ETextureType

{

eTextureType\_2D = 0,

eTextureType\_CubeMap,

eTextureType\_Volume,

eTextureType\_FX,

eTextureTypeCount

};

struct C3DBaseTexture : public C3DBaseResource

{

virtual ETextureType GetTextureType() const = 0;

};

class CResourceManager

{

public:

// инициализировать, указав MeshTree

void Init(C3DBaseObjectManager\*);

// указать время, спустя которое, текстура став не видимой

// для всех VisibilityManager выгрузится из памяти

// [настроечный параметр, по умолчанию = 25с]

void SetInvisibleUnloadTime(float time);

// добавить VisibilityManager в обработку

// [вызывается во время инициализации]

void AddVisibilityManager(CVisibilityManager\*);

// Обновить состояние ресурсов

// [можно вызывать в отдельном потоке, не привязываясь к циклу рендера]

void Update(float deltaTime);

// Получить текущий приоритет текстуры при отрисовке

float GetTexturePriority(C3DBaseTexture\*);

};