#### CAObject

###### 类说明

CrossApp引擎的根类，主要是进行内存的管理，定义部分回调函数指针。引擎绝大部分类派生自CAObject，其共同遵从同一套内存管理方式。

###### 属性

|  |  |  |
| --- | --- | --- |
| 访问修饰符 | 属性名 | 说明 |
| public | [m\_uID](#m_uID) | 对象唯一id |
| protected | [m\_uReference](#m_uReference) | 引用计数器 |
| protected | [m\_uAutoReleaseCount](#m_uAutoReleaseCount) | 自动释放引用计数器 |

###### 方法

|  |  |  |
| --- | --- | --- |
| 访问修饰符 | 方法名 | 说明 |
| public | [retain](#retain) | 增加对象的计数器 |
| public | [release](#release) | 减少对象的计数器 |
| public | [autorelease](#autorelease) | 设置自动管理方式 |
| public | [retainCount](#retainCount) | 返回对象当前计数器的值 |
| public | [isSingleReference](#isSingleReference) | 实例化对象是否只有一个使用者 |
| public | [isEqual](#isEqual) | 与指定object实例对象是否相同 |

###### 属性介绍

**m\_uID**

**类型：unsigned int**

解释：CAObject的实例对象id，采用累加的方式，初值为零，每创建一个object，m\_uID值加1，同时在创建一个object时会将当前对象的m\_uReference初始化为1，将当前的m\_uAutoReleaseCount初始化为0。

示例：

CAObject::CAObject(void): m\_uReference(1) , m\_uAutoReleaseCount(0)

{

static unsigned int uObjectCount = 0;

m\_uID = ++uObjectCount;

}

**m\_uReference**

类型：**unsigned int**

解释：引用计数器，每创建一个object，计数器的值自动加1。

**m\_uAutoReleaseCount**

类型：**unsigned int**

解释：是否自动管理object，如果m\_uAutoReleaseCount的值为0，非自动管理方式，当m\_uAutoReleaseCount的值为非0，那么采用自动管理的方式。

###### 方法介绍

**void retain(void)**

返回值：void

解释：对引用计数器m\_uReference的值加1。

示例：

void CAObject::retain(void)

{

++m\_uReference;

}

**void release(void)**

返回值：void

解释：对引用计数器m\_uReference的值减1，如果m\_uReference的值为0，则删除此object。

示例：

void CAObject::release(void)

{

--m\_uReference;

if (m\_uReference == 0)

{

delete this;

}

}

**CAObject\* autorelease(void)**

返回值：CAObject\*

解释：将当前object设置为自动内存管理方式

示例：

CAObject\* CAObject::autorelease(void)

{

CCPoolManager::sharedPoolManager()->addObject(this);

return this;

}

**unsigned int retainCount(void) const**

返回值：unsigned int

解释：返回当前引用计数器m\_uReference的值

示例：

unsigned int CAObject::retainCount(void) const

{

return m\_uReference;

}

**bool isSingleReference(void) const**

返回值：bool

解释：通过与引用计数器的值做比较，返回object是否只有一个使用者。

示例：

bool CAObject::isSingleReference(void) const

{

return m\_uReference == 1;

}

**virtual bool isEqual(const CAObject\* pObject)**

返回值：bool

解释：虚函数，判断当前对象与指定实例化对象是否相同

示例：

bool CAObject::isEqual(const CAObject \*pObject)

{

return this == pObject;

}

#### CrossApp内存管理简介

CrossApp采用引用计数器的方式进行内存的管理，CAObject的派生类都采用这种方式管理对象。最基本的原则就是谁new/retain，谁就负责release，当我们创建一个object时，会自动添加一个计数器，每当对这个object进行引用时，都会使用retain方法进行计数器的加1操作；同样的当释放对这个object的引用时都会调用release方法进行计数器的减1操作，如果引用计数器m\_uReference的值为0，就删除此object。

此外，CrossApp还提供了另一种自动管理内存的方式，当我们不确定object在何时释放时，可以调用autorelease方法进行延迟释放object操作，此时object处于自动管理状态，object被加入到自动释放池中，系统会在每帧结束的时候自动对释放池中的所有object进行一次release操作，如果m\_uReference的值为0，则删除当前object。