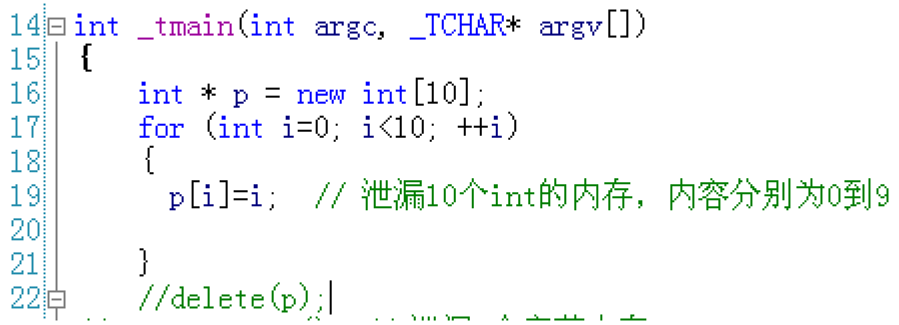
辜敏聪

gumc@coohex.com

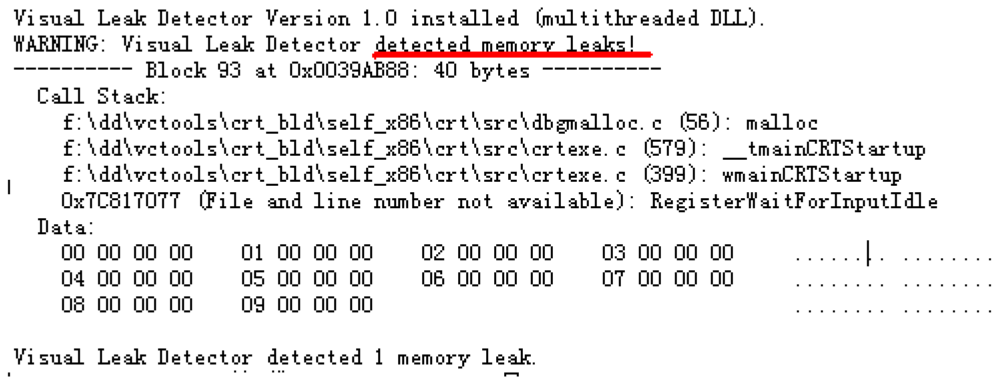
- 内存管理
- 基本架构

内存管理

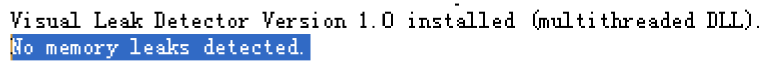


```
int _tmain(int argc, _TCHAR* argv[])
{
    int * p = new int[10];
    for (int i=0; i<10; ++i)
    {
      p[i]=i;  // 泄漏10个int的内存，内容分别为0到9

    }
    //delete(p);
```

内存管理



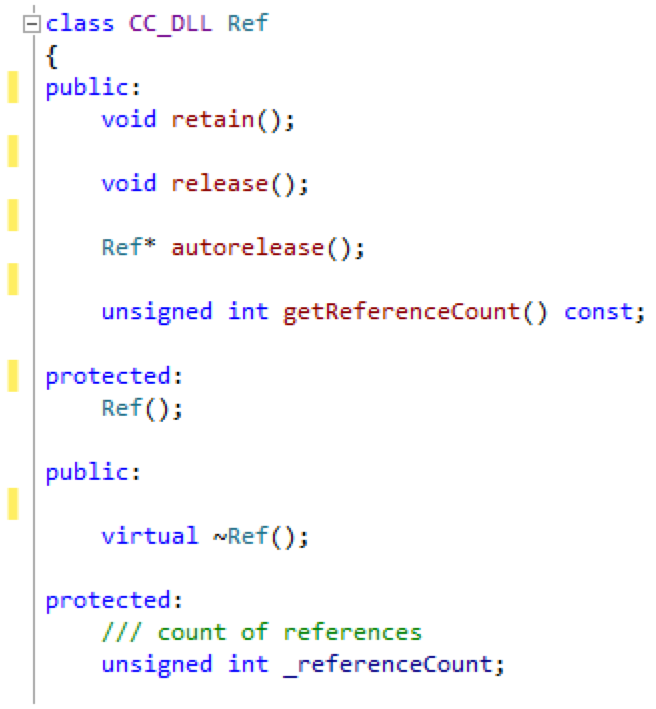
```
Visual Leak Detector Version 1.0 installed (multithreaded DLL).
WARNING: Visual Leak Detector detected memory leaks!
---------- Block 93 at 0x0039AB88: 40 bytes ----------
  Call Stack:
    f:\dd\vctools\crt_bld\self_x86\crt\src\dbgmalloc.c (56): malloc
    f:\dd\vctools\crt_bld\self_x86\crt\src\crtexe.c (579): __tmainCRTStartup
    f:\dd\vctools\crt_bld\self_x86\crt\src\crtexe.c (399): wmainCRTStartup
    0x7C817077 (File and line number not available): RegisterWaitForInputIdle
  Data:
    00 00 00 00    01 00 00 00    02 00 00 00    03 00 00 00    ........ ........
    04 00 00 00    05 00 00 00    06 00 00 00    07 00 00 00    ........ ........
    08 00 00 00    09 00 00 00                                  ........ ........

Visual Leak Detector detected 1 memory leak.
```



```
Visual Leak Detector Version 1.0 installed (multithreaded DLL).
No memory leaks detected.
```

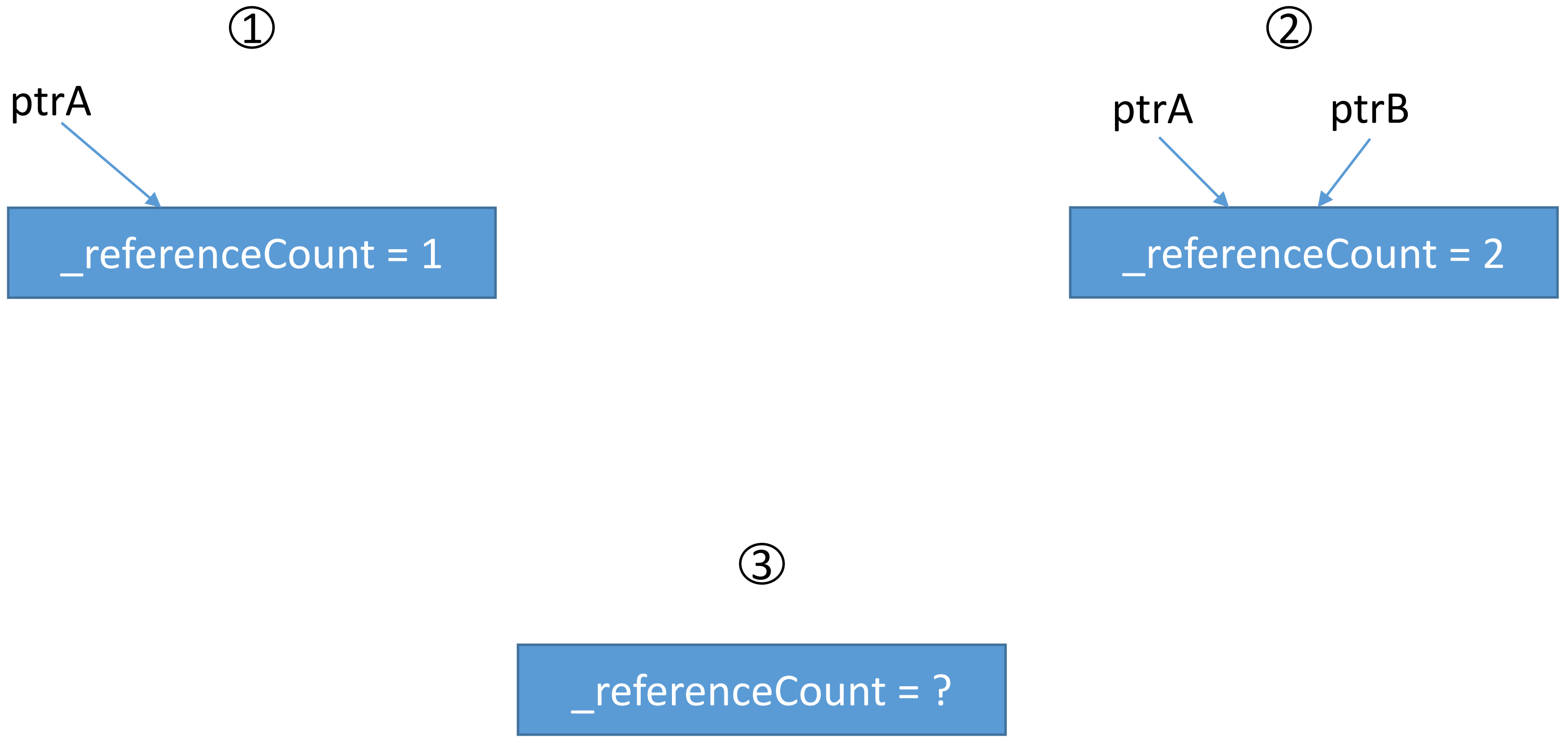
Visual Leak Detector

内存管理：引用计数

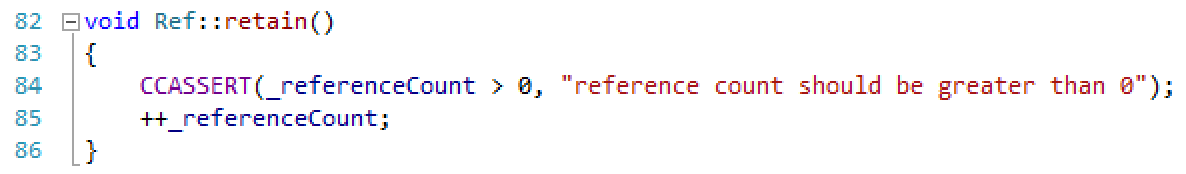


```
class CC_DLL Ref
{
public:
    void retain();

    void release();

    Ref* autorelease();

    unsigned int getReferenceCount() const;

protected:
    Ref();

public:

    virtual ~Ref();

protected:
    /// count of references
    unsigned int _referenceCount;
```

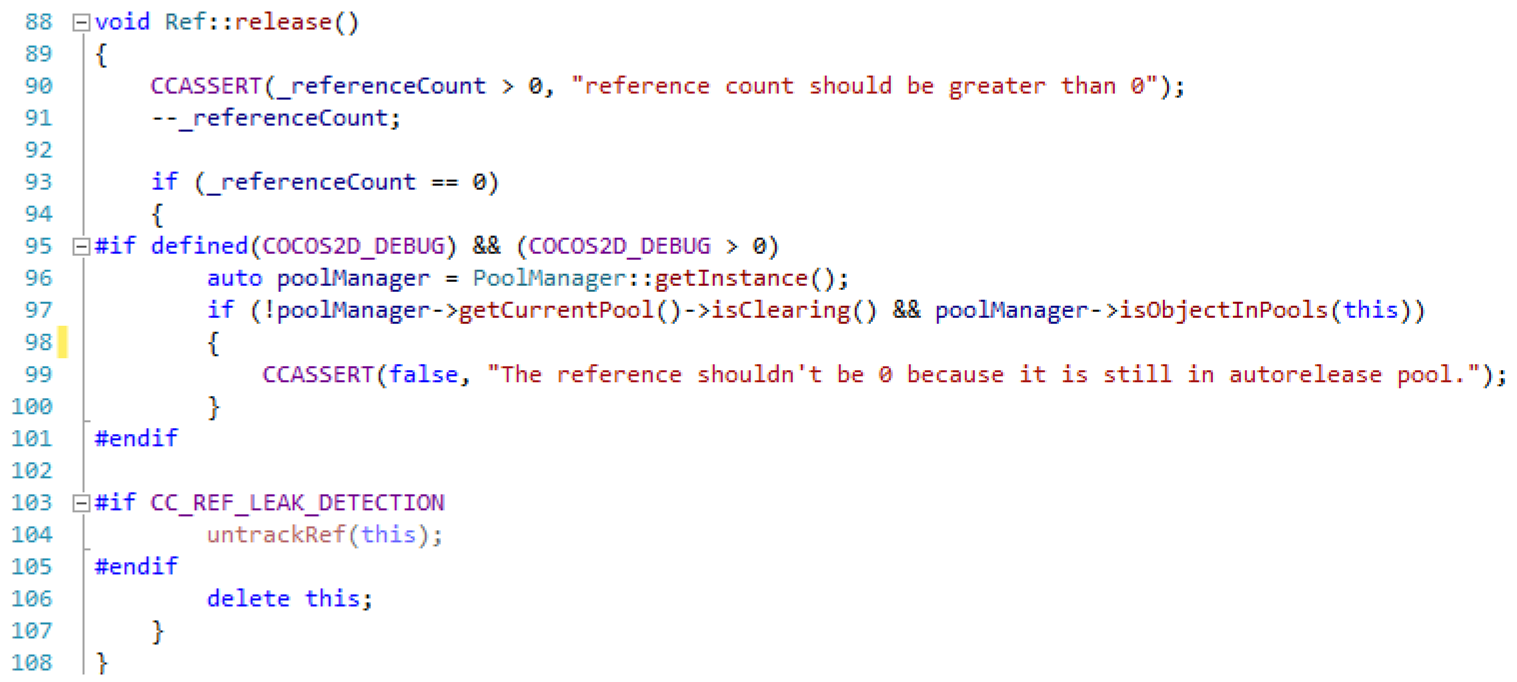
内存管理：引用计数


①
ptrA
_referenceCount = 1
②
ptrA
ptrB
_referenceCount = 2
③
_referenceCount = ?

内存管理：引用计数

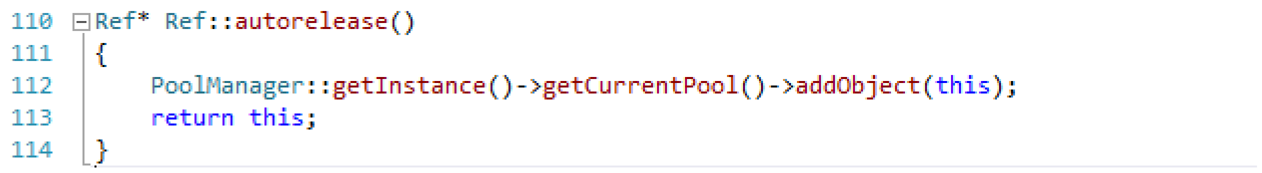


```cpp
void Ref::retain()
{
    CCASSERT(_referenceCount > 0, "reference count should be greater than 0");
    ++_referenceCount;
}
```



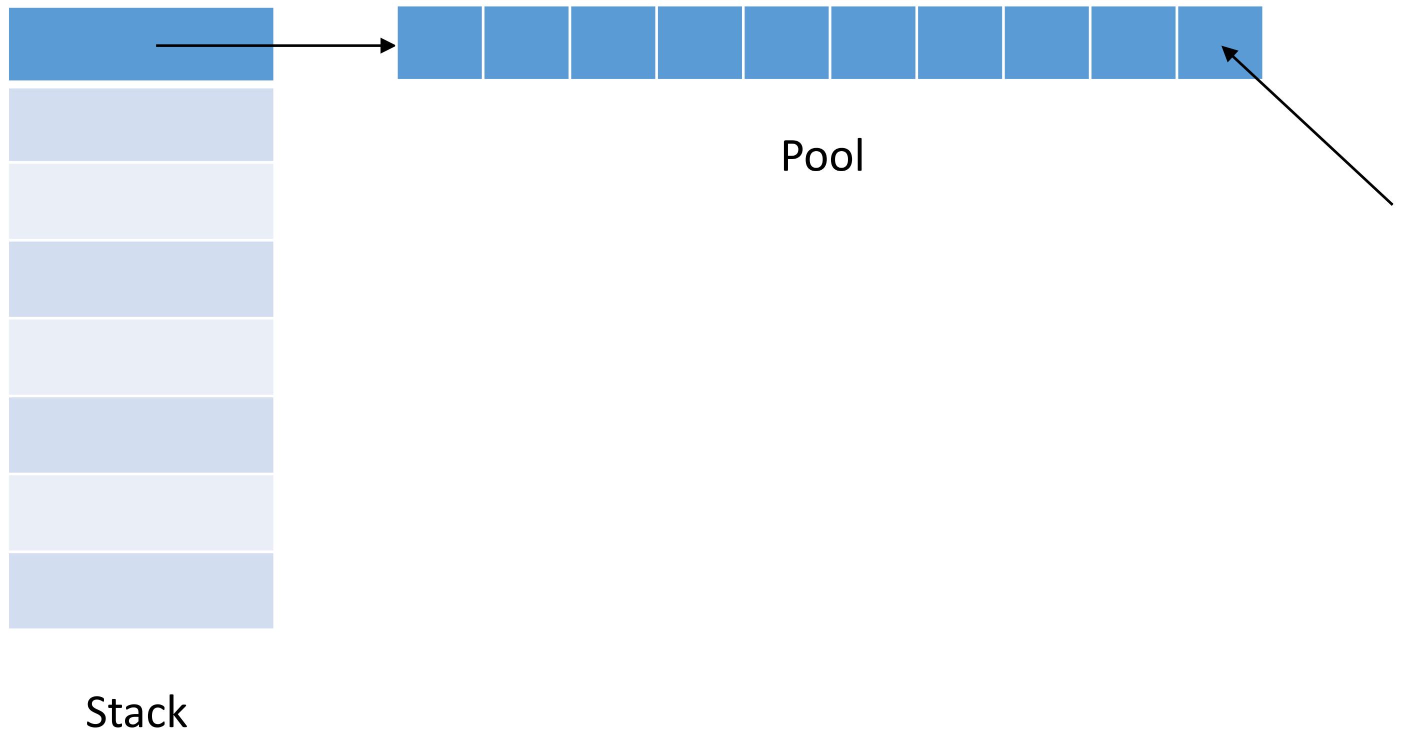
```cpp
void Ref::release()
{
    CCASSERT(_referenceCount > 0, "reference count should be greater than 0");
    --_referenceCount;

    if (_referenceCount == 0)
    {
#if defined(COCOS2D_DEBUG) && (COCOS2D_DEBUG > 0)
        auto poolManager = PoolManager::getInstance();
        if (!poolManager->getCurrentPool()->isClearing() && poolManager->isObjectInPools(this))
        {
            CCASSERT(false, "The reference shouldn't be 0 because it is still in autorelease pool.");
        }
#endif

#if CC_REF_LEAK_DETECTION
        untrackRef(this);
#endif
        delete this;
    }
}
```

内存管理：autorelease

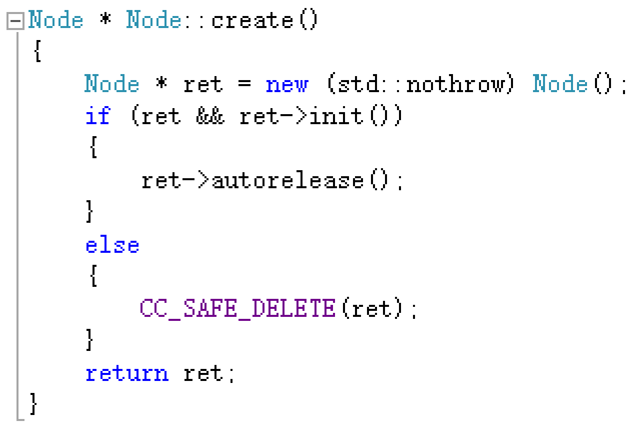


```
Ref* Ref::autorelease()
{
    PoolManager::getInstance()->getCurrentPool()->addObject(this);
    return this;
}
```

内存管理： autorelease



内存管理：二段构建模式



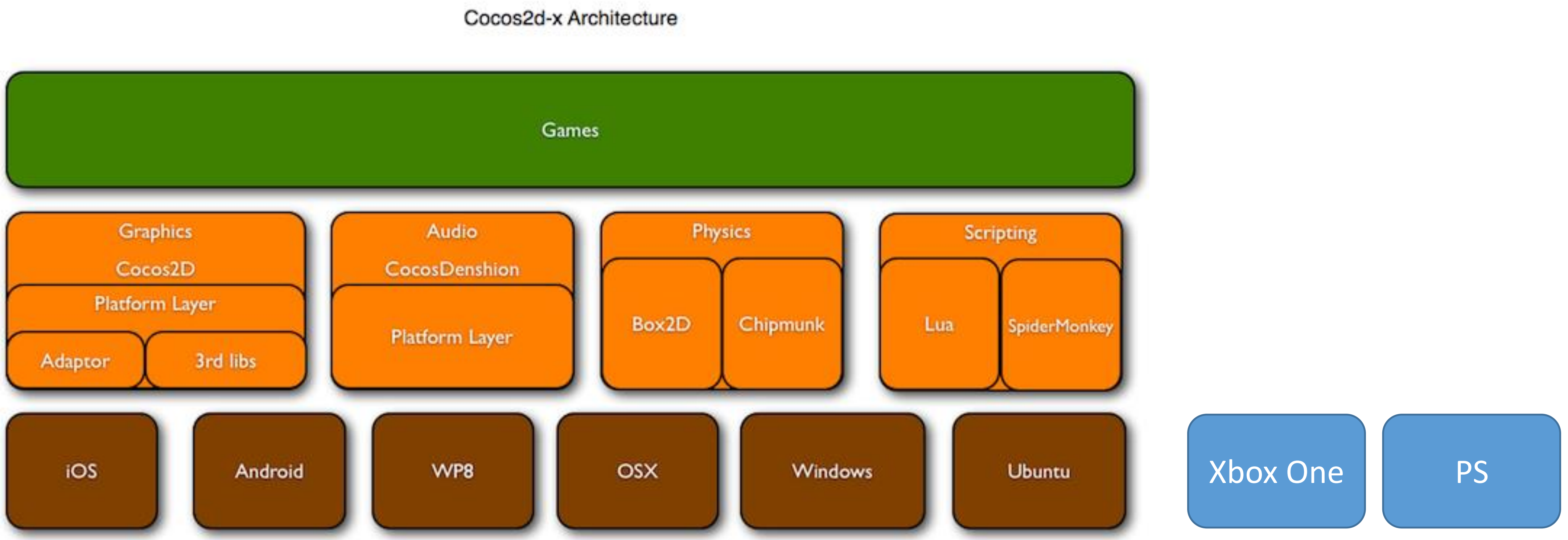
```
Node * Node::create()
{
    Node * ret = new (std::nothrow) Node();
    if (ret && ret->init())
    {
        ret->autorelease();
    }
    else
    {
        CC_SAFE_DELETE(ret);
    }
    return ret;
}
```

1. 构造函数分配内存
2. Init函数负责初始化

内存管理：拓展

Boost 智能指针

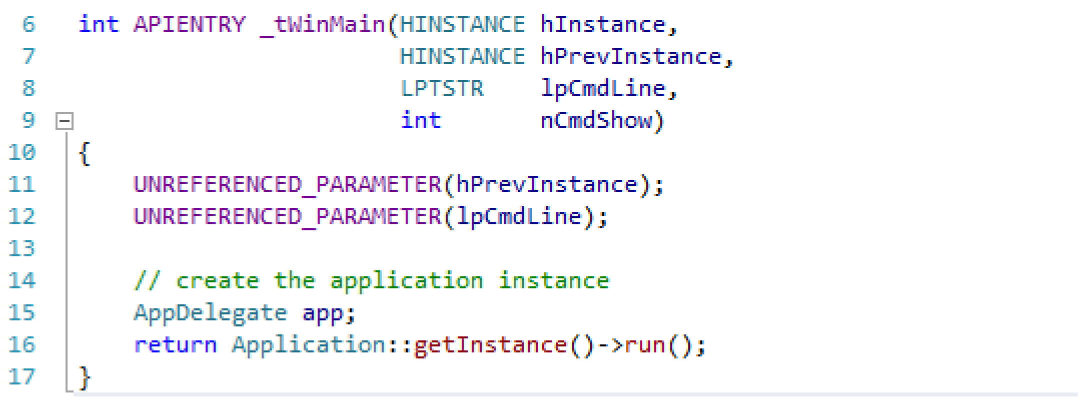
基本架构



图片来源：www.cocos2d-x.org

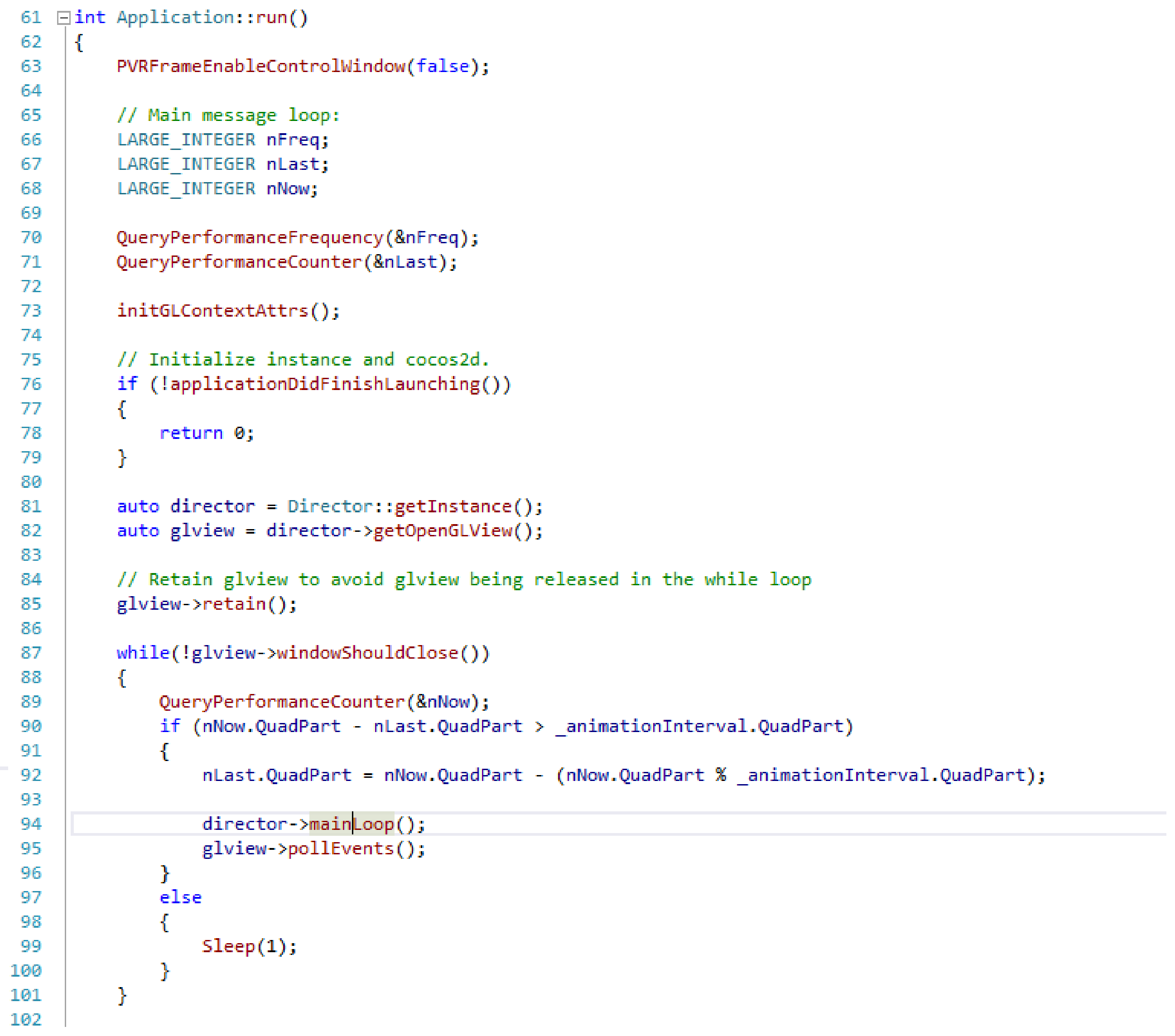
基本架构

程序入口



```cpp
int APIENTRY _tWinMain(HINSTANCE hInstance,
                       HINSTANCE hPrevInstance,
                       LPTSTR    lpCmdLine,
                       int       nCmdShow)
{
    UNREFERENCED_PARAMETER(hPrevInstance);
    UNREFERENCED_PARAMETER(lpCmdLine);

    // create the application instance
    AppDelegate app;
    return Application::getInstance()->run();
}
```

main.cpp

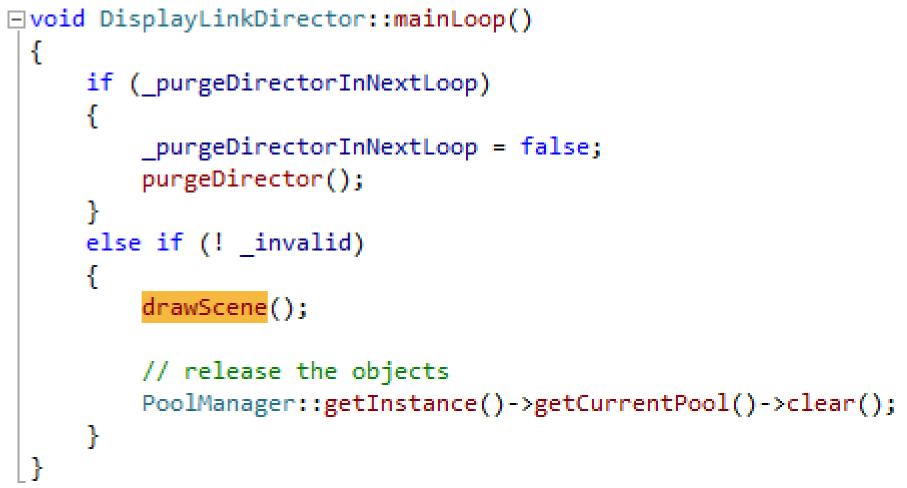


```cpp
int Application::run()
{
    PVRFrameEnableControlWindow(false);

    // Main message loop:
    LARGE_INTEGER nFreq;
    LARGE_INTEGER nLast;
    LARGE_INTEGER nNow;

    QueryPerformanceFrequency(&nFreq);
    QueryPerformanceCounter(&nLast);

    initGLContextAttrs();

    // Initialize instance and cocos2d.
    if (!applicationDidFinishLaunching())
    {
        return 0;
    }

    auto director = Director::getInstance();
    auto glview = director->getOpenGLView();

    // Retain glview to avoid glview being released in the while loop
    glview->retain();

    while(!glview->windowShouldClose())
    {
        QueryPerformanceCounter(&nNow);
        if (nNow.QuadPart - nLast.QuadPart > _animationInterval.QuadPart)
        {
            nLast.QuadPart = nNow.QuadPart - (nNow.QuadPart % _animationInterval.QuadPart);

            director->mainLoop();
            glview->pollEvents();
        }
        else
        {
            Sleep(1);
        }
    }

```

AppDelegate.cpp的run方法

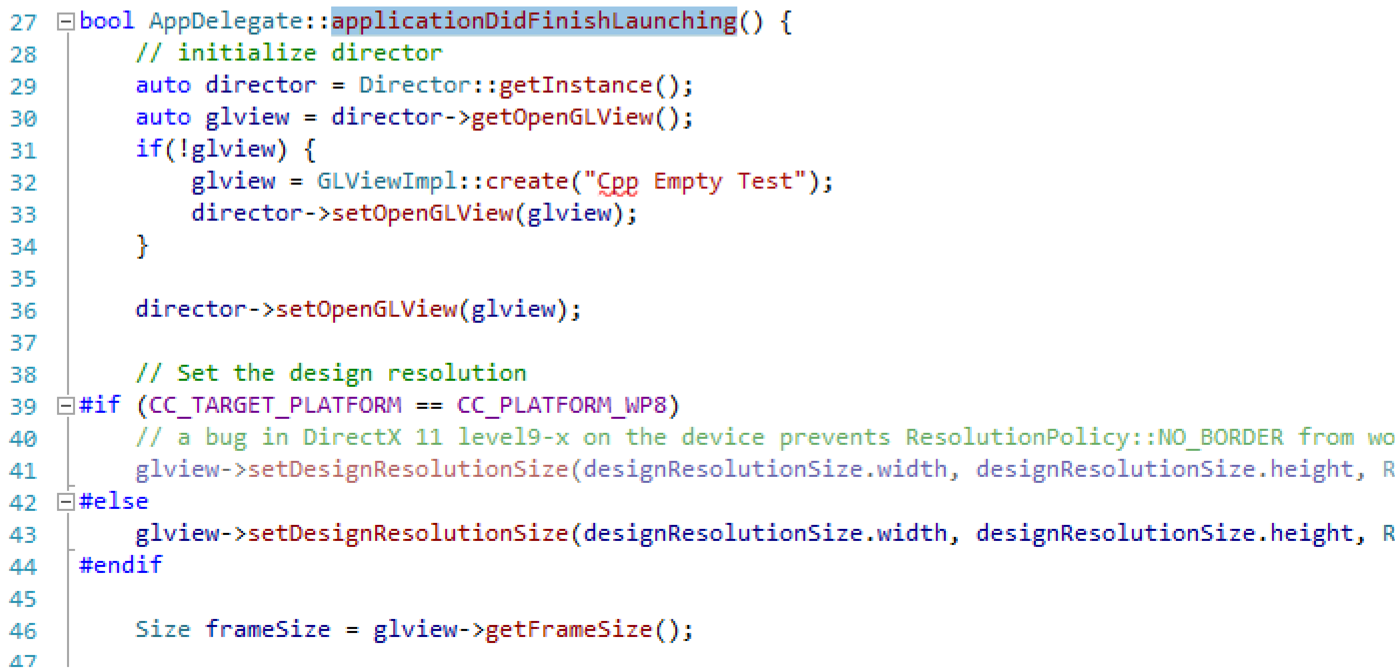
基本架构

程序入口



```
void DisplayLinkDirector::mainLoop()
{
    if (_purgeDirectorInNextLoop)
    {
        _purgeDirectorInNextLoop = false;
        purgeDirector();
    }
    else if (! _invalid)
    {
        drawScene();

        // release the objects
        PoolManager::getInstance()->getCurrentPool()->clear();
    }
}
```

CCDirector.cpp

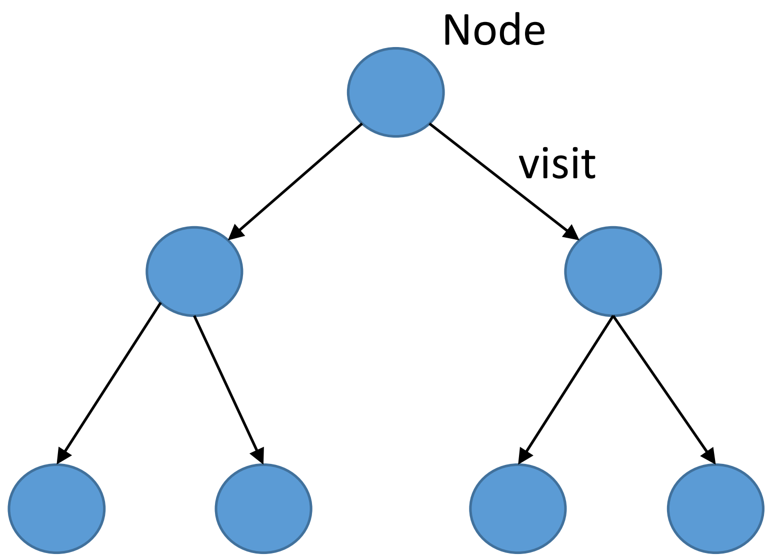


```
bool AppDelegate::applicationDidFinishLaunching() {
    // initialize director
    auto director = Director::getInstance();
    auto glview = director->getOpenGLView();
    if(!glview) {
        glview = GLViewImpl::create("Cpp Empty Test");
        director->setOpenGLView(glview);
    }

    director->setOpenGLView(glview);

    // Set the design resolution
#if (CC_TARGET_PLATFORM == CC_PLATFORM_WP8)
    // a bug in DirectX 11 level9-x on the device prevents ResolutionPolicy::NO_BORDER from wo
    glview->setDesignResolutionSize(designResolutionSize.width, designResolutionSize.height, R
#else
    glview->setDesignResolutionSize(designResolutionSize.width, designResolutionSize.height, R
#endif

    Size frameSize = glview->getFrameSize();
```

AppDelegate.cpp

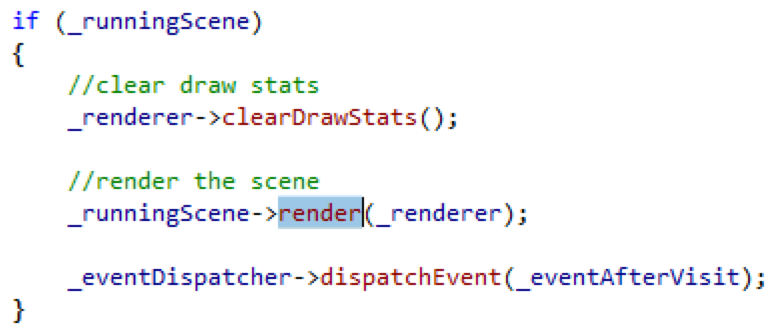
基本架构

渲染树：Node



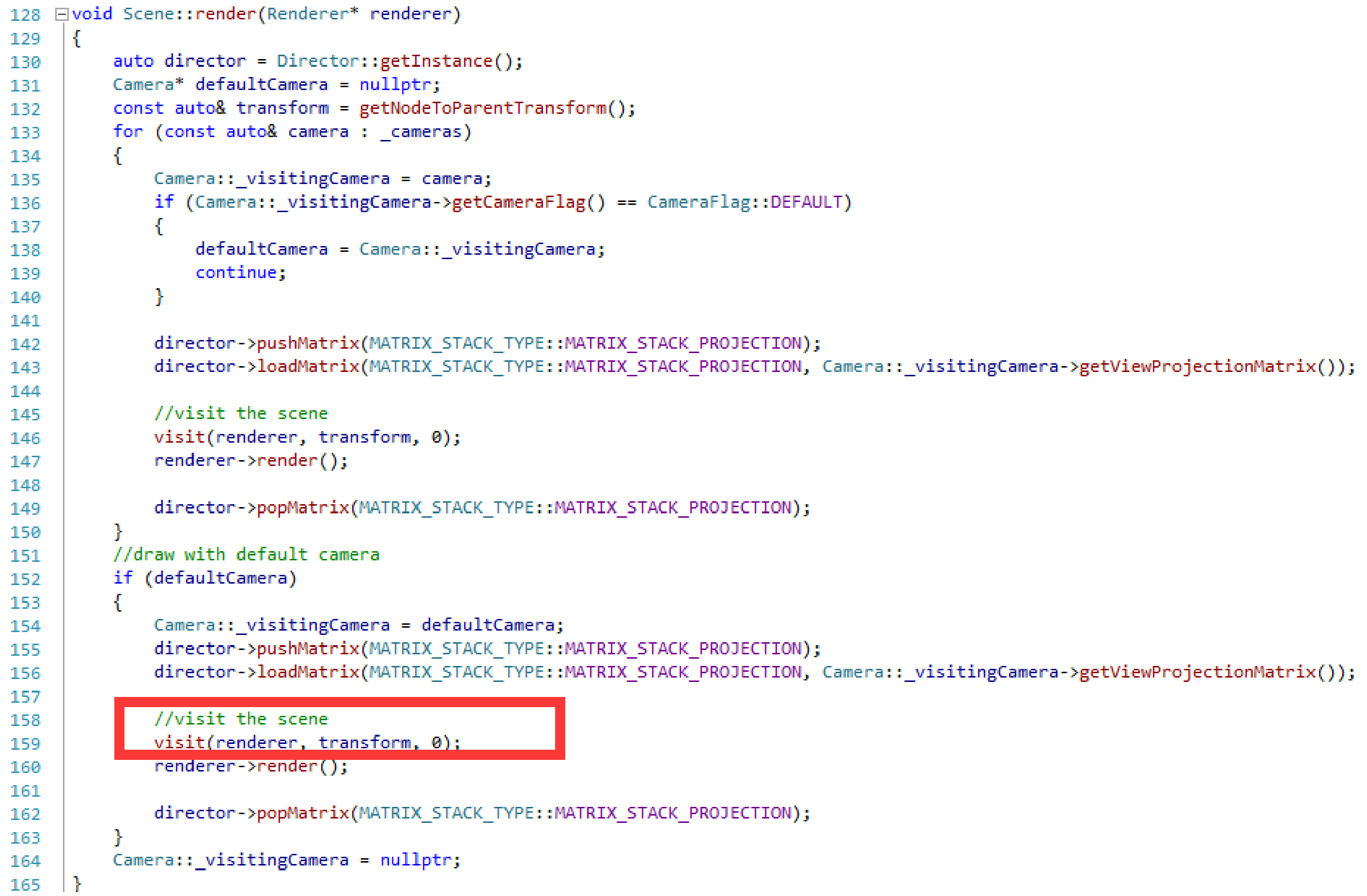
基本架构

渲染树：Node



```cpp
if (_runningScene)
{
    //clear draw stats
    _renderer->clearDrawStats();

    //render the scene
    _runningScene->render(_renderer);

    _eventDispatcher->dispatchEvent(_eventAfterVisit);
}
```

CCDirector.cpp

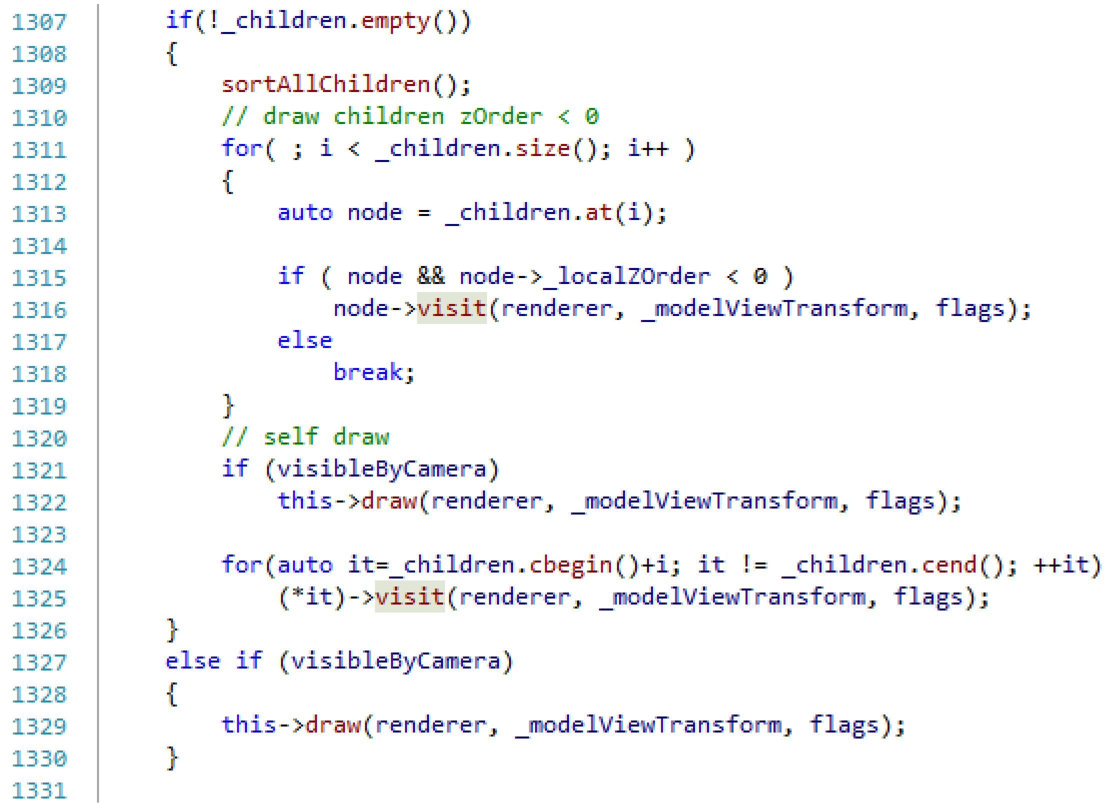


```cpp
void Scene::render(Renderer* renderer)
{
    auto director = Director::getInstance();
    Camera* defaultCamera = nullptr;
    const auto& transform = getNodeToParentTransform();
    for (const auto& camera : _cameras)
    {
        Camera::_visitingCamera = camera;
        if (Camera::_visitingCamera->getCameraFlag() == CameraFlag::DEFAULT)
        {
            defaultCamera = Camera::_visitingCamera;
            continue;
        }

        director->pushMatrix(MATRIX_STACK_TYPE::MATRIX_STACK_PROJECTION);
        director->loadMatrix(MATRIX_STACK_TYPE::MATRIX_STACK_PROJECTION, Camera::_visitingCamera->getViewProjectionMatrix());

        //visit the scene
        visit(renderer, transform, 0);
        renderer->render();

        director->popMatrix(MATRIX_STACK_TYPE::MATRIX_STACK_PROJECTION);
    }
    //draw with default camera
    if (defaultCamera)
    {
        Camera::_visitingCamera = defaultCamera;
        director->pushMatrix(MATRIX_STACK_TYPE::MATRIX_STACK_PROJECTION);
        director->loadMatrix(MATRIX_STACK_TYPE::MATRIX_STACK_PROJECTION, Camera::_visitingCamera->getViewProjectionMatrix());

        //visit the scene
        visit(renderer, transform, 0);
        renderer->render();

        director->popMatrix(MATRIX_STACK_TYPE::MATRIX_STACK_PROJECTION);
    }
    Camera::_visitingCamera = nullptr;
}
```

CCScene.cpp

基本架构

渲染树：Node



```cpp
if(!_children.empty())
{
    sortAllChildren();
    // draw children zOrder < 0
    for( ; i < _children.size(); i++ )
    {
        auto node = _children.at(i);

        if ( node && node->_localZOrder < 0 )
            node->visit(renderer, _modelViewTransform, flags);
        else
            break;
    }
    // self draw
    if (visibleByCamera)
        this->draw(renderer, _modelViewTransform, flags);

    for(auto it=_children.cbegin()+i; it != _children.cend(); ++it)
        (*it)->visit(renderer, _modelViewTransform, flags);
}
else if (visibleByCamera)
{
    this->draw(renderer, _modelViewTransform, flags);
}
```

CCNode.cpp

基本架构

Node的其他属性

位置：setPosition/getPosition

子节点：addChild()/removeChild()

父节点：removeFromParent()

标记：setTag()/getTag()

Z坐标：setZorder()

旋转、放大缩小 ……

基本架构

Layer

多点触摸事件开始:

```
virtual void onTouchesBegan(const std::vector<Touch*>& touches, Event *unused_event);
```

多点触摸事件移动:

```
virtual void onTouchesMoved(const std::vector<Touch*>& touches, Event *unused_event);
```

多点触摸事件结束:

```
virtual void onTouchesEnded(const std::vector<Touch*>& touches, Event *unused_event);
```

多点触摸事件中断：一般是系统层级的消息，如来电话，触摸事件就会被打断

```
virtual void onTouchesCancelled(const std::vector<Touch*>&touches, Event *unused_event);
```

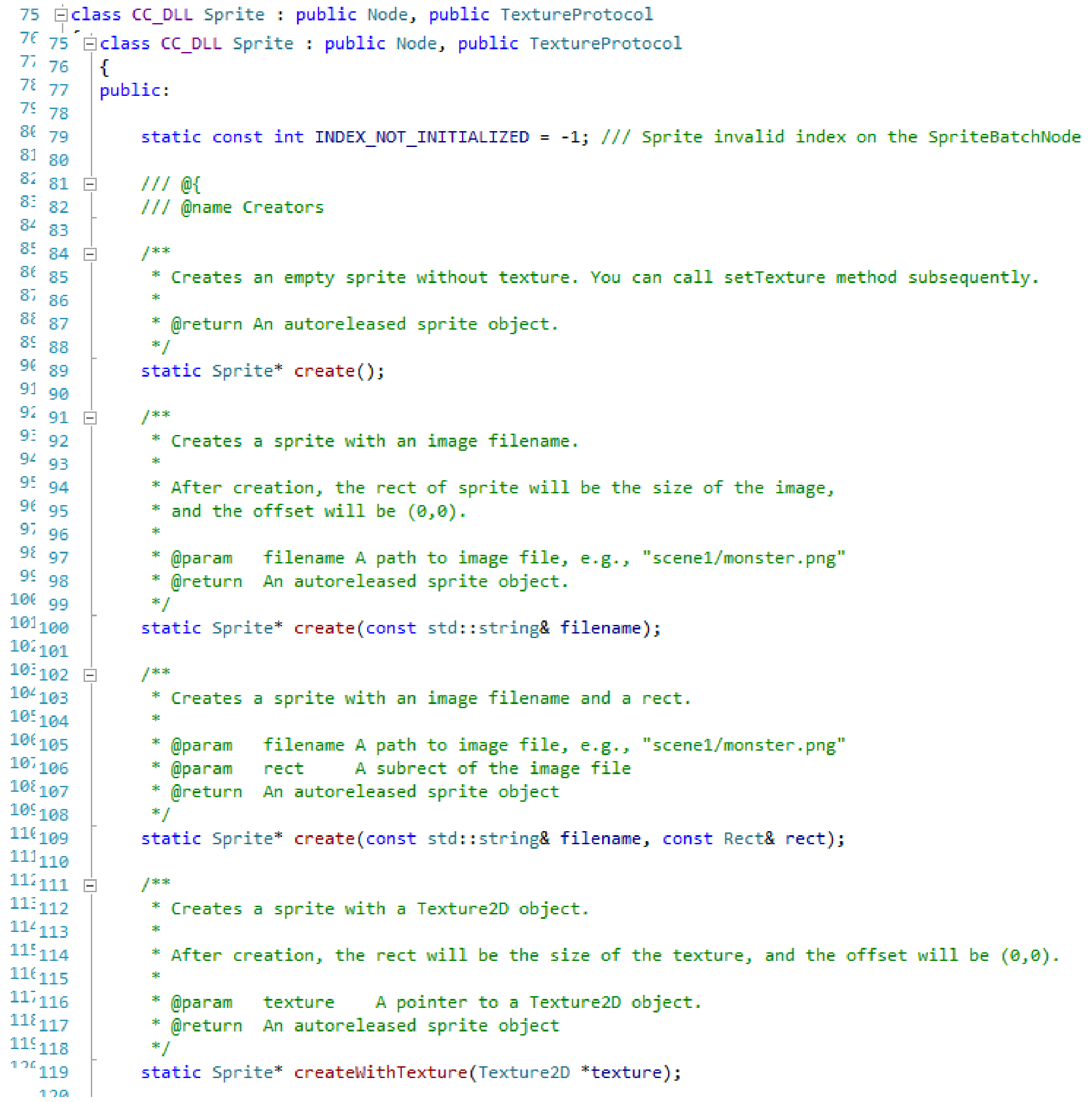
设置是否接受触摸

void setTouchEnabled(bool value);

自学单点触摸

基本架构

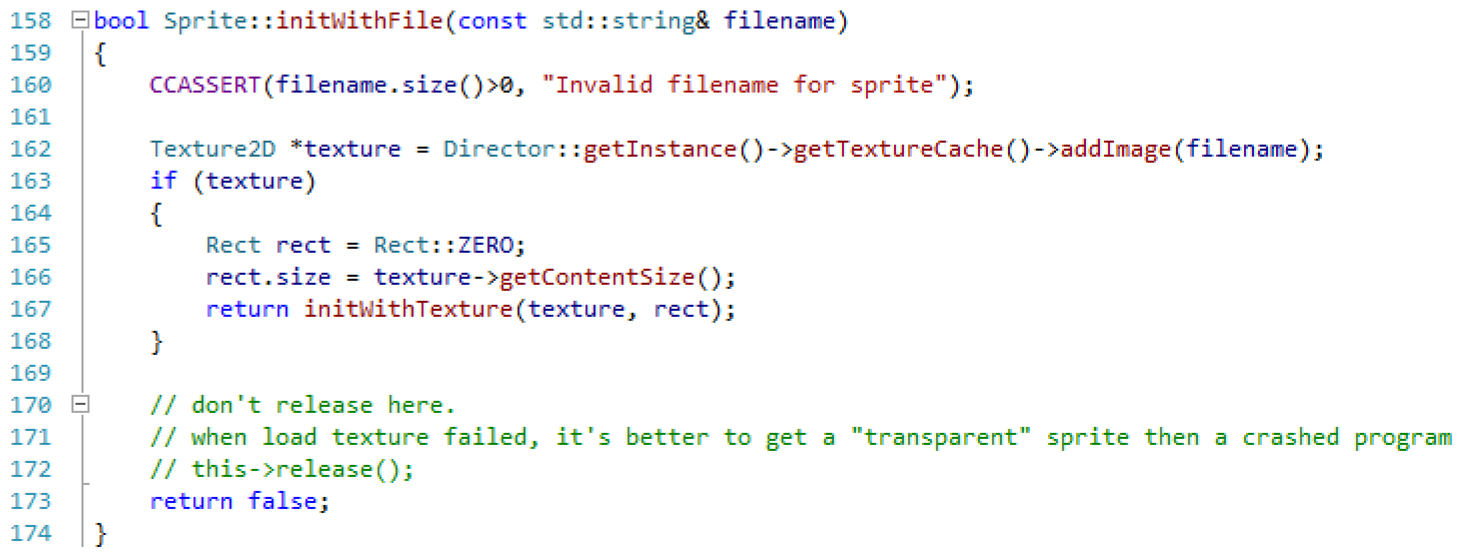
Sprite



```
class CC_DLL Sprite : public Node, public TextureProtocol
class CC_DLL Sprite : public Node, public TextureProtocol
{
public:

    static const int INDEX_NOT_INITIALIZED = -1; /// Sprite invalid index on the SpriteBatchNode

    /// @{
    /// @name Creators

    /**
     * Creates an empty sprite without texture. You can call setTexture method subsequently.
     *
     * @return An autoreleased sprite object.
     */
    static Sprite* create();

    /**
     * Creates a sprite with an image filename.
     *
     * After creation, the rect of sprite will be the size of the image,
     * and the offset will be (0,0).
     *
     * @param   filename A path to image file, e.g., "scene1/monster.png"
     * @return  An autoreleased sprite object.
     */
    static Sprite* create(const std::string& filename);

    /**
     * Creates a sprite with an image filename and a rect.
     *
     * @param   filename A path to image file, e.g., "scene1/monster.png"
     * @param   rect     A subrect of the image file
     * @return  An autoreleased sprite object
     */
    static Sprite* create(const std::string& filename, const Rect& rect);

    /**
     * Creates a sprite with a Texture2D object.
     *
     * After creation, the rect will be the size of the texture, and the offset will be (0,0).
     *
     * @param   texture    A pointer to a Texture2D object.
     * @return  An autoreleased sprite object
     */
    static Sprite* createWithTexture(Texture2D *texture);
```

基本架构

Sprite



```cpp
bool Sprite::initWithFile(const std::string& filename)
{
    CCASSERT(filename.size()>0, "Invalid filename for sprite");

    Texture2D *texture = Director::getInstance()->getTextureCache()->addImage(filename);
    if (texture)
    {
        Rect rect = Rect::ZERO;
        rect.size = texture->getContentSize();
        return initWithTexture(texture, rect);
    }

    // don't release here.
    // when load texture failed, it's better to get a "transparent" sprite then a crashed program
    // this->release();
    return false;
}
```

基本架构

Director、Node、Layer、Scene、Sprite之间的关系

谢谢！

辜敏聪

gumc@coohex.com