

一种基于 Qt 的系统内存泄漏检测方法

张 玲¹, 李 艳^{2,3}, 胡 术^{2,3}, 李 璞^{2,3}, 潘 倩^{2,3}

(1. 四川大学 计算机基础教学实验中心, 四川 成都 610065;

2. 四川大学 计算机学院, 四川 成都 610065;

3. 四川大学 视觉合成图形图像技术国防重点学科实验室, 四川 成都 610065)

摘 要:在软件开发中,用户界面程序提高了软件系统易操作性、用户体验度等非功能性需求。长时间、复杂流程的大型软件系统对人机界面的稳定性则提出了较高要求,不能出现内存泄漏、不能中途异常退出。针对使用图形用户界面应用程序框架 Qt 开发的用户界面程序的内存泄漏问题,提出了一种基于 Qt 的人机界面程序的源码静态内存检测方法。该方法针对 Qt 控件对象是否存在父控件的两种内存泄漏情况,识别所需检查的目标对象是否为 Qt 控件类对象,是否存在内存泄漏。该方法提供的 Qt 控件对象内存检测能力可准确识别目标对象是否存在内存泄漏、手工删除错误等问题,便于开发人员及时检查错误,修正缺陷,减少程序运行中的内存泄漏问题,以满足大型系统的软件质量需求。

关键词:Qt;内存泄漏;遍历匹配;泄漏检测

中图分类号:TP301

文献标识码:A

文章编号:1673-629X(2017)12-0119-05

doi:10.3969/j.issn.1673-629X.2017.12.026

A Memory Leakage Detection Method for Software System Based on Qt

ZHANG Ling¹, LI Yan^{2,3}, HU Shu^{2,3}, LI Pu^{2,3}, PAN Qian^{2,3}

(1. Computer Teaching Experiment Center, Sichuan University, Chengdu 610065 China;

2. Department of Computer, Sichuan University, Chengdu 610065, China;

3. National Key Laboratory of Fundamental Science on Synthetic Vision, Sichuan University, Chengdu 610065, China)

Abstract: In software development, the user interface program improves the non-functional requirements of software system, such as easy operation, user experience and so on. The large-scale training system of long-term and complex process gives the higher requirements to the stability of human-computer interface, no memory leak, no halfway abnormal exit. Aiming at the memory leakage of the user interface program developed by graphical user interface application framework Qt, a source static memory detection method of Qt-based human-computer interface program is presented. There are two types of memory leaks for the Qt control object for the parent control. The method identifies whether the target object to be checked is a Qt control class object, and whether there is a memory leak. The memory detection of Qt control object provided by this method can accurately identify whether the target object has a memory leak and manually remove the faults, which makes it easy for the developers to check the errors and correct the defects in time, and to reduce the program running in the memory leakage, meeting the functional requirements of large-scale aviation training system.

Key words: Qt; memory leakage; traversal matching; leak detection

0 引 言

当下,大型软件系统开发人机界面所需的开源图形用户界面程序框架选择具有多样性,如提供了对 Windows 本地组件的访问方式,兼容了 Linux 及其他 Mono 平台的微软 .NET 开发框架图形用户界面 Windows. Forms;用于形状、文档、图像、视频、动画、三维及

用于放置控件、内容的控件模型框架,支持流动文字、3D 视觉效果,基于 Vista 的用户界面框架 WPF;用于提供原生 Mac OSX 应用程序开发的类库 MonoMac 等。文中对面向对象、易于使用、允许真正的组件编程并具有优良的跨平台特性的 Qt^[1] 框架开发的人机界面进行阐述。

收稿日期:2017-01-04

修回日期:2017-05-03

网络出版时间:2017-09-27

基金项目:中国民航创新基金项目(MHRD20150228)

作者简介:张 玲(1964-),女,实验师,研究方向为计算机应用、仿真与网络。

网络出版地址: <http://www.cnki.net/kcms/detail/61.1450.TP.20170927.0958.050.html>

Qt 是一个跨平台的基于 C++ 编程语言的图形用户界面应用程序框架,该框架提供给应用程序开发者建立艺术级的图形用户界面所需的按钮、滚动条、菜单及其他对象等功能^[2]。在 Qt 中,QObject 类作为本库大多数类的基类,QObject 对象总是以树状结构组织自己。在创建一个 QObject 对象时,可指定其父对象(也被称为父控件),新创建的对象将被加入到其父对象的子对象(也被称为子控件)列表中。当父对象被析构时,该父对象对应列表中的所有子对象也将被析构,同理可知,当某个 QObject 对象被析构时,它会将自己从所属父对象的列表中删除,以避免父对象被析构时,再次析构自己。由于 QObject 对象具备上述析构功能,当使用 new 操作符在堆中创建 QObject 对象时,不需要使用 delete 操作符析构它们。

综上所述,由于 Qt 的特殊性,开发人员在使用 Qt 开发软件系统的人机界面时,极易写出内存处理不当的程序,最终导致内存泄露。文中将详细阐述在系统开发中进行 Qt 代码静态内存检查的方法。

1 相关工作

由于动态分配的内存没有及时、正确地释放而引起的内存泄露^[3],直接影响了程序的可用性和稳定性。内存泄露分为常发性内存泄露、偶发性内存泄露、一次性内存泄露及隐式内存泄露等四类情况^[4]。在高级语言编程中,内存泄露通常指进程运行时没有调用 delete 操作符将动态分配的内存释放,但指向该内存的指针却被指向其他内存或被销毁,导致该内存丢失,系统失去了对该内存的控制权,从而造成了内存浪费^[5]。目前,为解决内存泄露问题,一些高级编程语言通过垃圾收集机制^[6]跟踪内存的使用情况,并按照固定时间间隔周期性自动回收不再使用的动态分配内存,但目前 C++ 并不支持该机制。此外,标准库中相关函数提供了内存泄露识别、检测功能,如使用调试堆栈函数和在需要检测内存泄露的位置添加输入调试信息函数用

于输出内存泄露的地址及其内容,但该 C 标准函数输出的内存泄露信息不能产生具体的堆栈调用结果,从而无法得知内存泄露的具体对象。Insure++^[7]是运行时检测工具,能检测 C/C++ 中编程和运行时错误,检验静态、堆栈以及动态分配内存操作的有效性,但不支持第三方库函数的内存检测^[8]。文中提出一种基于 Qt 开发的软件系统用户界面程序的内存泄露检测方法。针对 Qt 中两种专属内存泄露情况即 Qt 控件对象是否存在父控件提供了静态检测能力,能够通过的相关配置文件中添加新的 Qt 控件对象识别所需检测的具体对象,准确识别该对象是否存在内存泄露,能够有效地减少程序运行中的内存泄露问题。

2 Qt 内存泄露检测策略

Qt 内存泄露的原因呈现多变性,文中主要实现了 Qt 控件对象是否存在父控件两种情况的内存检测功能。Qt 控件对象在没有父控件^[9]的条件下,若程序在堆中通过 new 操作符创建了一个对象,则必须通过 delete 操作符析构该对象完成删除操作,否则程序会出现内存泄露问题;相反地,Qt 控件对象在具有父控件的条件下,则无需开发人员通过 delete 操作符手动删除该 Qt 控件对象,因为该控件对象的父控件在析构时,会自动析构自己对应子控件列表中的控件,为 Qt 提供了自动内存机制,在此情况下,开发人员若误写删除代码,可能造成冗余删除;除此之外,Qt 中还有其他属于 C++ 编程语言的内存泄露和冗余析构的情况^[10]。文中提出的方法目前尚未涉及这些情况的检测,该方法实现了 Qt 控件对象是否有父控件的内存泄露检测功能。基于 Qt 的系统内存检测方法由扫描工程文件中的所有 .cpp 文件和 .h 文件,解析各 .cpp 文件及 .h 文件中的 new、delete 操作符,存储创建和删除对象的信息,整合 Qt 控件对象创建、删除的统一完整信息,检测信息输出结果等 5 个模块组成。Qt 内存静态检测框架如图 1 所示。

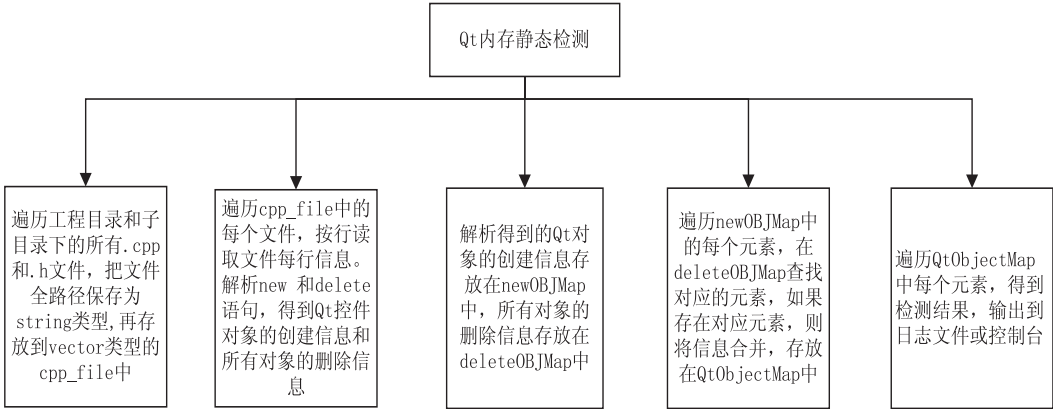


图 1 Qt 内存静态检测框架

3 Qt 内存泄漏检测实施

3.1 扫描工程文件

MS Windows 系统中采用 `_findfirst()`、`_findnext()` 函数遍历需要检测的工程文件下的所有文件,将扫描得到的 .h 文件和 .cpp 文件存放在同一个顺序容器 `vector` 中^[11],文件扫描流程如图 2 所示。

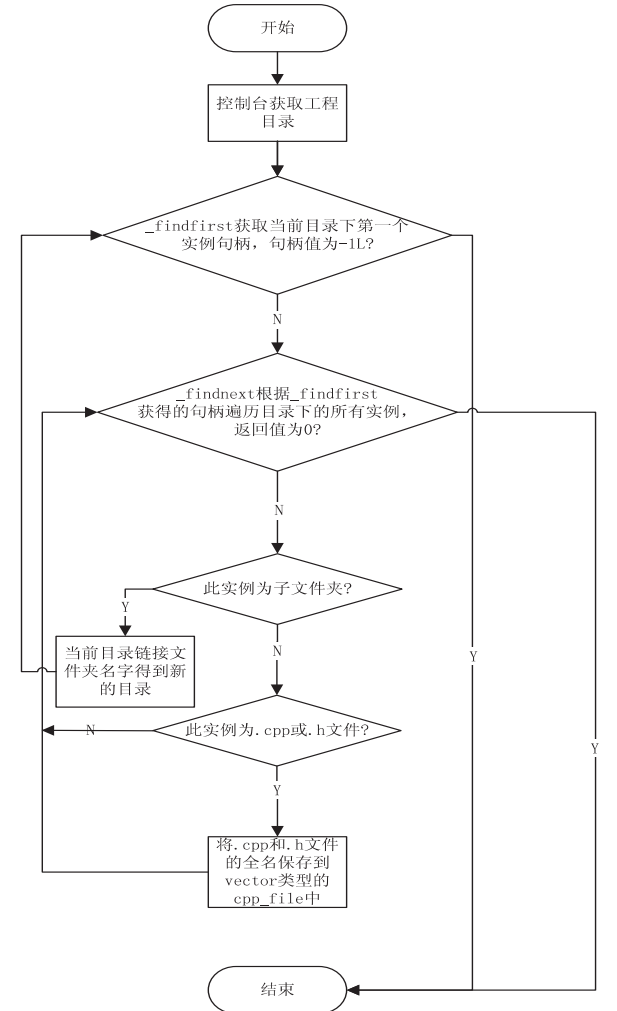


图 2 扫描工程所有 .cpp 和 .h 文件流程

3.2 解析工程文件

遍历顺序容器 `vector`,依次对各 .cpp 文件、.h 文件执行 `getLine()` 函数^[12],获取当前文件中的每一行程序,识别程序中由 `new`、`delete` 操作符执行的对象,并判断由 `new` 创建的对象是否为 Qt 控件类。`delete` 操作符直接删除对象,不能得到对象类名,从而不能判断删除的对象是否是 Qt 控件类对象,因此不需判断由 `delete` 操作符删除的对象是否是 Qt 控件类,只需存储由 `delete` 操作符删除的所有的对象信息。解析工程中的各文件流程如图 3 所示。

3.2.1 识别注释、非注释

读取文件每行程序,首先判断该程序是否为注释,一般情况下,编程语言的注释由行注释(“//”)和段注释(“/* XXXX */”)组成。注释与代码需区别对

待,通过申明一个 `bool` 类型的变量 `result` 来区别被检测的代码是否为注释,变量 `result` 由解析代码的函数返回值更新。

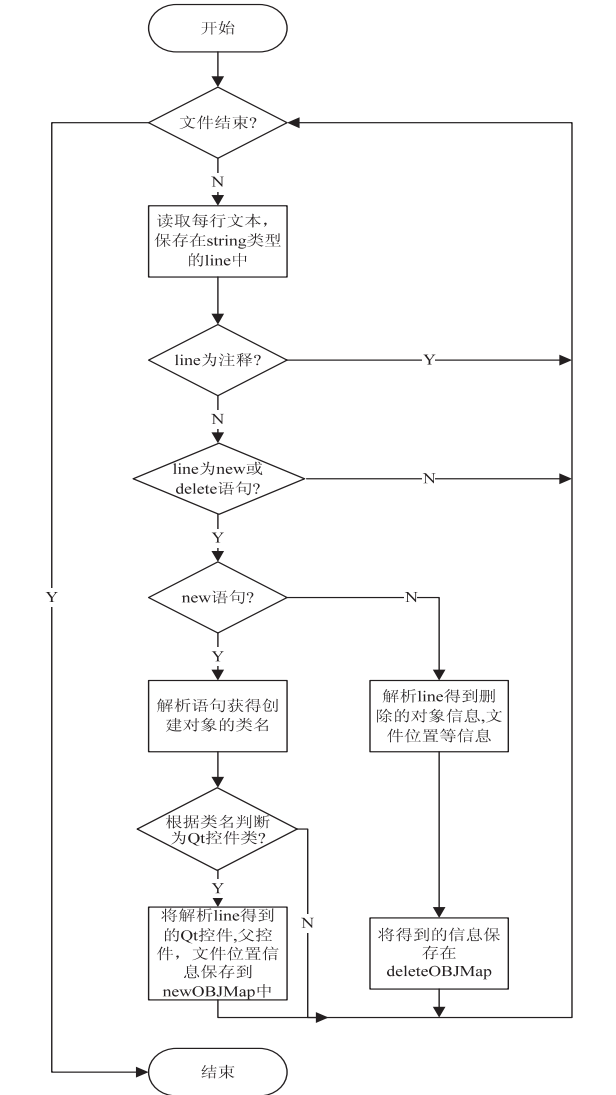


图 3 解析文件流程

若被检测代码为行注释,则解析进程直接跳过该行即可,并将变量 `result` 设置为 `commentEnd` (表示注释结束);若为段注释,且在一行代码没有同时出现“/ *”和“*/”,则 `result` 值为 `commentBegin` (表示注释开始),反之,若在一行代码中同时存在“/*”和“*/”,则 `result` 值为 `commentEnd` (表示注释结束)。

进程在解析每行代码时,需根据 `result` 值选择不同的解析函数 `parseComment()` (解析注释的函数),`parseCode()` (解析代码的函数)进行 `new`、`delete` 操作符的检测操作。

3.2.2 识别 Qt 控件对象

在解析文件的各行代码中,当代码出现 `new` 操作符创建对象时,解析语句获得创建对象的类型并根据该对象的类名来判断是否为 Qt 控件类^[13]。文中识别 Qt 控件对象共分为两步:识别程序中创建或析构对象

的语句;判断识别出的对象是否为 Qt 类。delete 对象语句的识别处理方式与 new 对象语句相同,不同之处在于因 delete 语句无法得到对象所属类名,进而无法判断被删对象是不是 Qt 控件类。下面将以 new 操作符创建的对象为示例来阐述识别 Qt 控件对象的具体过程。

(1) 识别程序中创建对象的语句。

一般情况下,程序创建 Qt 控件对象的结构如下所示:

```
QPushButton m_button=new QPushButton( Parent );
```

此例代码最具标志性的词分别是 new 操作符和“=”,而创建对象的 new 操作符应保证是独立的,需排除“* * * new * * *”情况的出现。

为识别创建的对象,需从程序中获取对象的信息,包括对象名(m_button)、类名(QPushButton)以及父对象(parent)。而对象信息的标识符则是通过“=”、“new”以及“()”等来获取各标识符的 begin(第一个字符在整行 string 中的索引)和 end(最后一个字符在整行 string 中的索引),然后通过 begin 和 end 获得对象信息。C++中,string 类提供了可直接使用的接口用于查找目标对象的信息内容,substr(begin, len) 用于截取子串,find(“arg”) 用于查找索引值,配合使用 substr()、find() 函数即可完成对象名、类名及其父对象的查找工作。

(2) 判断步骤(1)对象是否为 Qt 控件对象。

因考虑到识别所有的 Qt 控件对象^[14]是一项繁杂的工作,目前仅是添加一些常用的 Qt 控件对象以供识别。当然,程序并没有固定的 Qt 控件对象,若出现新的 Qt 控件对象需要识别,在配置文件 InheritTree. ini 中手动添加新 Qt 对象即可进行识别操作。Qt 控件对象所属类均具有继承关系,且对象的最终父对象都为 QWidget 类。针对该继承树关系,文中通过读取配置文件 InheritTree. ini 内容建立了一个树结构,如图 4 所示。

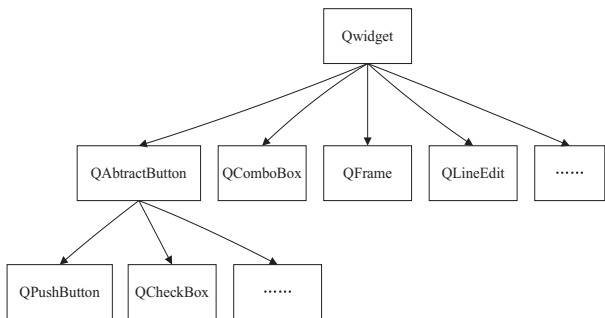


图 4 读 inheritTree. ini 配置文件生成 Qt 控件类的继承树

配置文件 InheritTree. ini 文件结构如下例所示:

[树的高度]

```
depthNum = n
[ 高度 0]
NodeNum = 1 //当前高度下节点数目
QWidget=NULL//等号左边是空间类,右边是起父类
[ 高度 1]
...
[ 高度 n-1]
...
```

树结构提供了接口 findChild(string key, POINTER &p), 指针 p 指向树中关键字为 key 的节点,如果树中不存在关键字为 key 的节点,则 p 的值为 NULL。该树结构可判断树中是否存在关键字为 key 的节点,因此,把待判断对象的类名作为实参传给 key,即可判断此类是否为 Qt 控件类,也就能判断被检测对象是不是 Qt 控件对象。

3.3 存储解析信息

本步骤将解析得到的通过 new 操作符创建的 Qt 控件对象信息和通过 delete 操作符删除的 Qt 控件对象信息分别存储在结构体中,而两结构体对象分别存储在 newOBJMap、deleteOBJMap 中。

(1) 操作符 new 语句的信息结构体 newOBJ 包含创建的对象名、该对象名所属的父对象名以及该对象的创建位置,其中对象创建位置包括文件名、代码行数:

```
Struct newOBJ
{
String OBJName;//对象名
String Parent;//父对象名
Position m_position;//记录对象创建的位置,包括所属文件,对应代码行
};
```

(2) 操作符 delete 语句的信息结构体 deleteOBJ 包含删除的对象名以及记录对象删除的位置,该对象的位置包括所属文件名、代码行数:

```
Struct deleteOBJ
{
String OBJName;//对象名
Positionm_position;//记录对象删除的位置,包括所属文件,对应代码行
};
```

3.4 整合解析信息

本步骤将存储在 newOBJMap、deleteOBJMap 中的结构体信息整合到 QTObjectMap,整合流程如图 5 所示。

通过循环迭代 newOBJMap 中的各对象信息结构体,从结构体中获取各 Qt 控件类对象,通过遍历 deleteOBJMap 中各结构体查找与该 Qt 控件对象名相符的对象,若具有相同对象名,则将 newOBJMap、

deleteOBJMap 中属于该对象的结构体信息整合到 QTOBJ 结构体中,并把 QTOBJ 结构体对象插入到 QTOBJMap 对象中。QTOBJ 结构体包含 Qt 控件对象名、该对象的父对象名、记录该对象创建及删除的位置、该对象是否通过 delete 操作符手动删除。

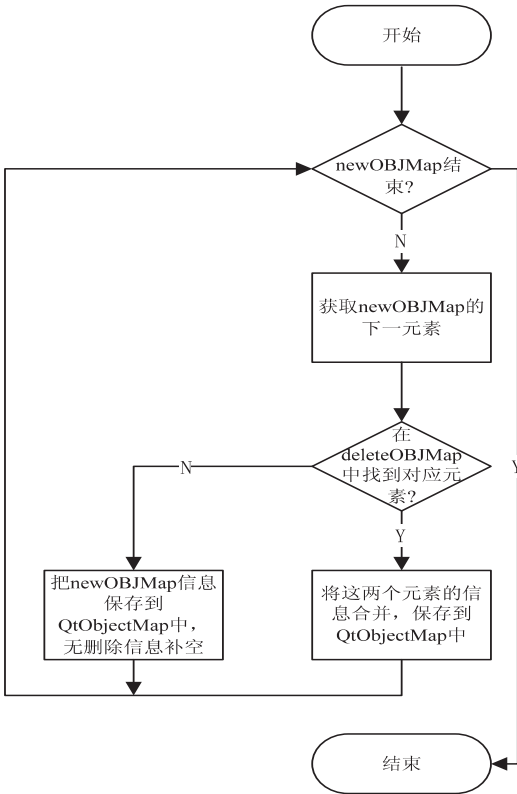


图 5 合并 newOBJMap 和 deleteOBJMap 的流程

```
Struct QTOBJ
{
String OBJName;//对象名
String Parent;//父对象名
Position m_position;//记录对象创建的位置,包括所属文件,
对应代码行
Position m_position;//记录对象删除的位置,包括所属文件,
对应代码行
Bool isDelete;//记录是否在程序中手动删除
};
```

3.5 检测解析信息

根据 QTOBJMap 中的信息输出检测结果。在 QTOBJMap 中的各 QTOBJ 结构体对象,若结构体有父控件,则无需通过 delete 操作符手动删除该 Qt 对象,否则冗余 delete;若结构体没有父控件,则必须通过 delete 操作符手动删除,否则内存泄漏。最后将结果输出到 DOS 界面和日志文件中。其中检测过程的伪代码如下:

```
If(QTOBJ.parent!=“ ”)//此 QTOBJ 控件对象拥有父控件
{
If(QTOBJ.isdelete==true)//程序中出现了手动删除
```

```
检测出情况 2:手动删除错误
}
If(QTOBJ.parent==“ ”)//此 QTOBJ 控件对象没有父控件
{
If(QTOBJ.isdelete==false)//程序中没有出现手动删除
检测出情况 1:疑似内存泄露
}
}
```

3.6 实 例

使用 Qt 代码静态检查^[15]的方法开发一个可执行程序,该程序运行时需将被检查的代码所在的目录作为运行参数,目录可包含子目录。该执行程序通过打印输出信息方便开发人员修改代码,对基于图形用户界面程序框架 Qt 开发的航管训练系统进行描述。以检查“E://航管训练”工程文件目录下的所有文件为例,输出示例:

- (1)疑似内存泄露:objectName(控件对象名)E://航管训练//a.cpp(创建对象文件全目录)600(行号);
- (2)疑似手工删除错误:objectName(控件名)parentName(父控件名)E://航管训练//a.cpp(创建对象文件全目录)600(行号)E://航管训练//a.cpp(删除对象文件全目录)(行号)。

4 结 束 语

内存泄漏是软件开发中的常见错误,针对在开发软件系统人机界面出现的内存泄漏情况,提出了一种基于 Qt 开发的软件系统程序内存检测方法。该方法可以有效检测 Qt 常见内存泄漏、准确定位内存泄漏位置并打印输出该信息,便于开发人员及时修正错误。Qt 内存泄漏的原因多种多样,该方法主要实现了 Qt 控件对象是否存在父控件两种情况的内存检测功能,并没有涉及到属于 C++语言使用中的其他内存泄漏和冗余 delete 情况的检测。

参考文献:

[1] BLANCHETTE J,SUMMERFIELD M.C++ GUI programming with Qt 4[M].2nd ed.[s.l.]:[s.n.],2015.

[2] 李全虎.交互界面开发工具-Qt[J].中国科技信息,2005(5):33.

[3] III W P A,LEVINE F E,REYNOLDS W R,et al.Method and system for shadow heap memory leak detection and other heap analysis in an object-oriented environment during real-time trace processing;US,US6658652[P].2003.

[4] 柳 青,杨英豪,孙永超.C++内存检测的研究[J].电子工业专用设备,2014,43(12):35-38.

[5] 道夫曼,纽伯格.C++内存管理[M].北京:学苑出版社,1994.



(a)展厅入口



(b)展厅背面



(c)大厅前台

图 6 近距离观察效果图

行流畅,画面真实感强,人机交互效果良好,拥有广阔的应用前景。未来,随着虚拟现实技术的不断成熟,沉浸式虚拟交互系统将在沉浸感、真实感等方向继续向前发展,增添更加丰富的内容。

参考文献:

[1] FENG Y F. Design and realization of roaming system of a virtual community based on Virtools[J]. Computer Simulation, 2009,29(6):285-286.

(上接第 123 页)

[6] 谢之易. 一种新的适用于面向对象程序设计语言的保守式垃圾收集机制[J]. 计算机应用与软件,2008,25(1):96-99.

[7] NANCE J. Product review:insure++[J]. Linux Journal,1998,1998(51):14.

[8] 吴 民,涂奉生. 内存泄漏的动态跟踪分析[J]. 计算机工程与应用,2005,41(14):18-20.

[9] 王卫东,屈 洋. 可视化图形界面中有关控件的属性类的研究[J]. 微机发展(现更名:计算机技术与发展),2005,15(12):27-28.

[10] XU G. Distinguished paper precise memory leak detection for java software using container profiling * [J]. ACM Transactions on Software Engineering & Methodology, 2008,22(3):

[2] AGHINA M A C, MÓL A C A, JORGE C A F, et al. Non-conventional interfaces for human -system interaction in nuclear plants' virtual simulations [J]. Progress in Nuclear Energy, 2012,59:33-43.

[3] LONG P, ZENG Q, HE T, et al. Development of a geometry-coupled visual analysis system for MCNP[J]. Progress in Nuclear Science and Technology, 2011,2:280-283.

[4] JACOBSON J, RENARD M L, LUGRIN J L, et al. The CaveUT system: immersive entertainment based on a game engine [C]//Proceedings of the 2005 ACM SIGCHI international conference on advances in computer entertainment technology. Valencia, Spain: ACM, 2005:184-187.

[5] 赵沁平,郝爱民,王莉莉,等. 实时三维图形平台 BH_GRAPH[J]. 计算机研究与发展,2006,43(9):1491-1497.

[6] 马登武,叶 文,于凤全. 虚拟现实技术及其在飞行仿真中的应用[M]. 北京:国防工业出版社,2005.

[7] 仲于姗. 基于 Unity 的 3D 虚拟校园漫游系统的开发[D]. 云南:云南大学,2015.

[8] 朱惠娟. 基于 Unity3D 的虚拟漫游系统[J]. 计算机系统应用,2012,21(10):36-39.

[9] 张 璐. 基于虚拟现实技术的用户界面设计与研究[D]. 上海:东华大学,2013.

[10] 袁 林. 基于虚拟现实技术的三维建模方法研究[D]. 乌鲁木齐:新疆大学,2008.

[11] 张 颖. 虚拟视景漫游中碰撞检测技术研究[D]. 长春:吉林农业大学,2014.

[12] 王梅亮,万华根,顾键萍. 基于游戏模式的行人横穿道路技能辅助训练系统研究[J]. 系统仿真学报,2016,28(6):1406-1411.

[13] 张典华,陈一民. 基于 Unity3D 的多平台虚拟校园设计与实现[J]. 计算机技术与发展,2014,24(2):127-130.

[14] 熊 耀. 基于 Unity3D 粒子系统的三维影视特效开发研究[J]. 软件导刊,2012,11(11):134-136.

[15] 高 源,刘 越,程德文,等. 头盔显示器发展综述[J]. 计算机辅助设计与图形学学报,2016,28(6):896-904.

151-160.

[11] 余 锋,祝晓鹰. 用 ActiveX 控件实现目录遍历[J]. 电脑编程技巧与维护,2001(4):43-46.

[12] LIPPMAN S B, LAJOIE J, MOO B E. C++ Primer 中文版[M]. 北京:电子工业出版社,2013.

[13] WILLIS T, NEWSOME B. Visual Basic 2010 入门经典[M]. 第 6 版. 北京:清华大学出版社,2011.

[14] 李 彬. Linux Qt GUI 开发详解[M]. 北京:北京航空航天大学出版社,2013.

[15] DATHATHRI R, REDDY C, RAMASHEKAR T, et al. Dynamic memory access monitoring based on tagged memory [C]//International conference on parallel architectures and compilation techniques. [s. l.]: IEEE, 2013:409-410.