Create New Commands in Tcl

eryar@163.com

摘要 Abstract：Tcl/Tck 脚本可以很容易实现用户自定义的命令，方便的创建图形化的用户界面 GUI，所以 Tcl 和 Tk 的应用领域几乎覆盖了图形和工程应用的全部范围，包括计算机辅助设计、软件开发、测试、仪器控制、科学可视化及多媒体方面。本文主要详解如何在 C 程序中使用 Tcl 来创建自定义的命令，并理解 OpenCascade 的 Draw Test Harness 的实现。

关键字 Key Words：OpenCascade, Tcl/Tk, Draw Test Harness, Software Customization

一、引言 Introduction

Tcl 是”Tool Command Language”，和 Python、Perl、Lua 等一样也是一种脚本语言。利用 Tcl 可以很容易实现自定义的命令，利用 Tk 可以很方便地创建出跨平台的用户界面 UI。因 Tcl/Tk 功能强大，跨平台便于移植，且是开源免费的，所以在 OpenCascade 中就利用这个库来实现了测试程序 Draw Test Harness，并且也利用 Tcl 实现了自动化测试。

因为 Tcl 是解析语言，所以 Tcl 可用来做为程序中的二次开发语言。因为修改 Tcl 的脚本不需要重新编译链接，只需要重新加载下，加快开发速度。

通过在 Tcl 中实现自定义的命令，来理解 OpenCascade 中 Draw Test Harness 的实现，并去感受 Tcl 的强大功能。

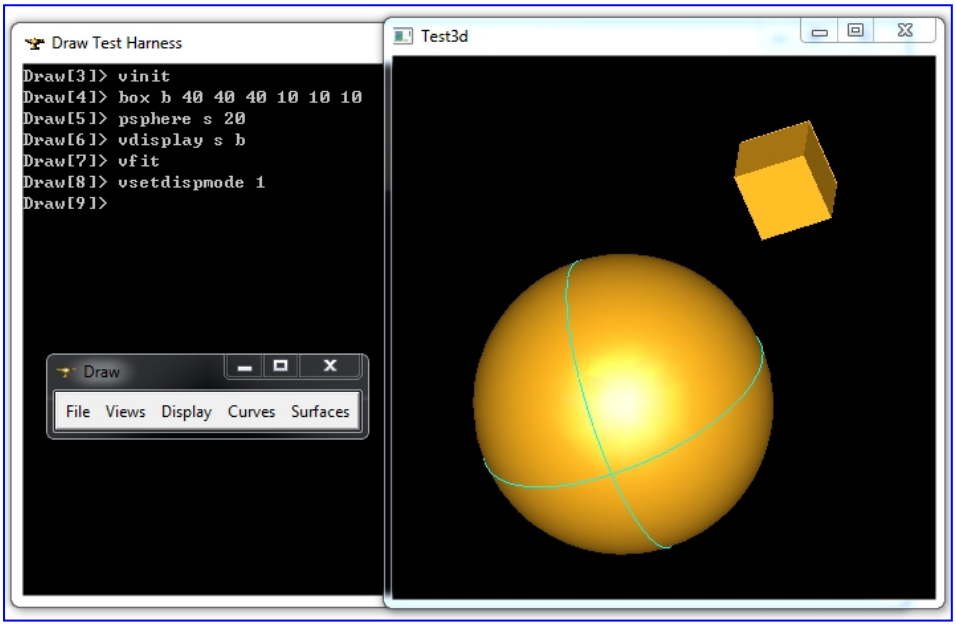

Figure 1.1 Draw Test Harness with Tcl/Tk

二、Tcl/Tk 应用 OpenCascade Draw Test Harness

在 Draw Test Harness 中输入自定义的命令就可以对相应的建模造型程序进行检验。理解其实现方法，也可以加深对 OpenCascade 的其他模块的理解。如下图所示为自定义命令：

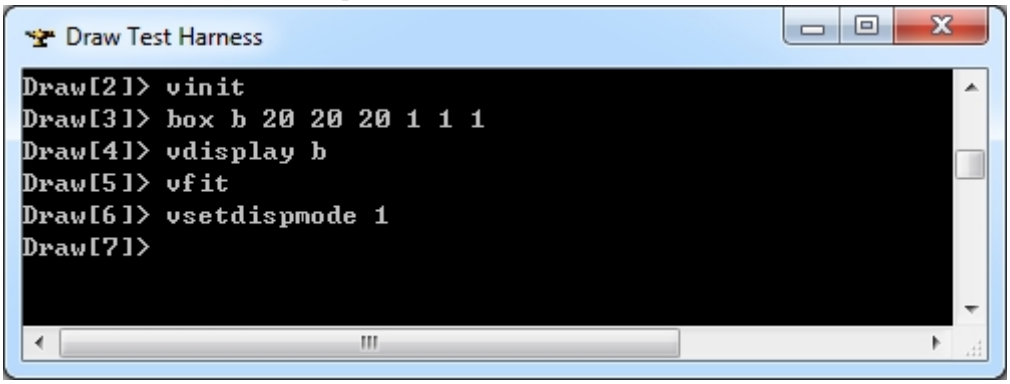

Figure 2.1 User defined command in Draw Test Harness

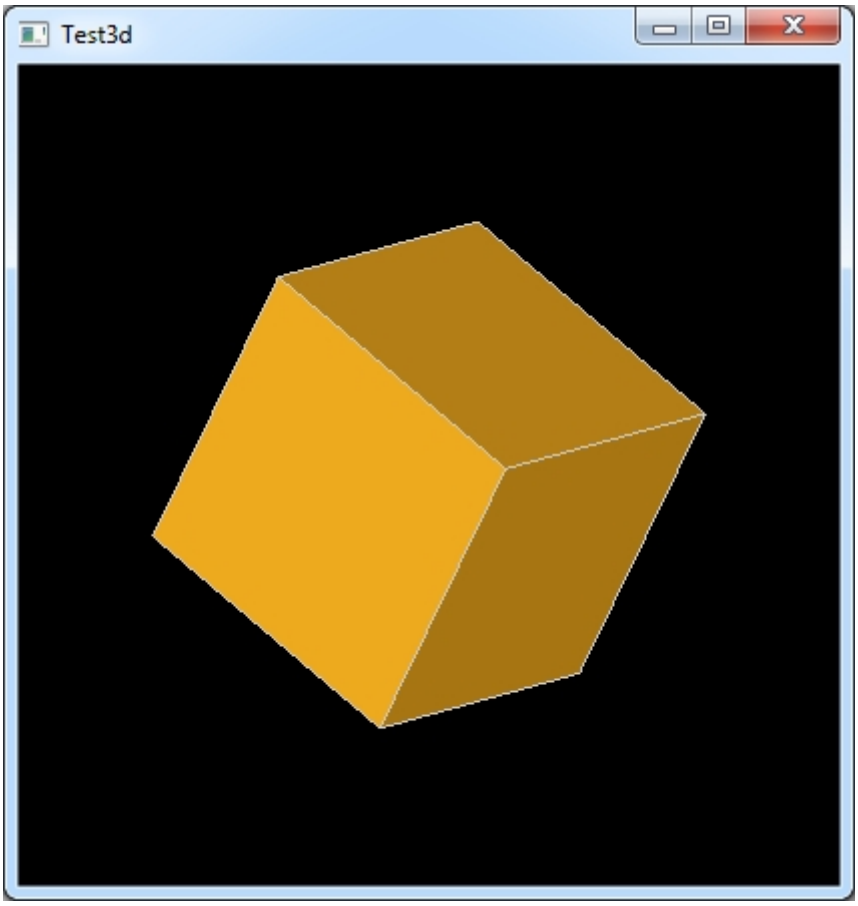

Figure 2.2 Command result in Test3d view

可以结合《OpenCascade Test Harness User's Guide》来试试 Draw Test Harness，如果将命令做在自己的程序中，就可以实现脚本建模啦。

Draw Test Harness 中的命令都是用 Tcl/Tk 来实现的，下面通过一个简单的示例来说明如何在 Tcl 中实现自定义的命令。学会 Tcl 脚本应该也是掌握了一种强大的脚本工具，可以为自己的程序实现二次开发功能。

三、程序示例 Example Code

在 Tcl 中实现自定义命令很方便，只需要按 Tcl 的格式定义一个命令函数。基于对象的命令函数的声明如下：

```
typedef int (Tcl_ObjCmdProc) _ANSI_ARGS_((ClientData clientData,
    Tcl_Interp *interp, int objc, struct Tcl_Obj * CONST * objv));
```

为了能在 Tcl 中调用一个命令函数，必须先调用 Tcl_CreateObjCommand 注册它，格式如下所示：

```
EXTERN Tcl_Command      Tcl_CreateObjCommand (Tcl_Interp * interp,
                        CONST char * cmdName, Tcl_ObjCmdProc * proc,
                        ClientData clientData,
                        Tcl_CmdDeleteProc * deleteProc);
```

这就是把 Tcl 中的字符串与实现它的 C 函数关联起来“魔术”。一个完整的程序如下所示，程序实现了两个自定义的命令 randomcmd 和 equalcmd，分别用来生成一个随机数和相等判断：

```
/*
*       Copyright (c) 2014 eryar All Rights Reserved.
*
*                File : Main.cpp
*              Author : eryar@163.com
*                Date : 2014-01-09 18:58
*             Version : 1.0v
*
*       Description : Create new command for Tcl in C. Refer to
*                           1. Tcl and Tk Toolkit
*                           2. Practical Programming in Tcl and Tk
*
*          Key Words : Tcl/Tk, C Interface, New Command
*
*/

#include <tcl.h>
#include <stdlib.h>
#include <string.h>

#pragma comment(lib, "tcl85.lib")

/*
* @breif Definitions for application-specific command procedures.
*/
int RandomCmd(ClientData clientData, Tcl_Interp* interp, int objc, Tcl_Obj *CONST objv[])
{
    if (objc != 2)
    {
        Tcl_WrongNumArgs(interp, 1, objv, "?range");
        return TCL_ERROR;
    }

    int limit = 0;
    Tcl_Obj* result = NULL;

    Tcl_GetIntFromObj(interp, objv[1], &limit);

    result = Tcl_NewIntObj(rand() % limit);

    Tcl_SetObjResult(interp, result);
```

```
    return TCL_OK;
}

int EqualCmd(ClientData clientData, Tcl_Interp* interp, int objc, Tcl_Obj *CONST objv[])
{
    if (objc != 3)
    {
        Tcl_WrongNumArgs(interp, 1, objv, "string1 string2");
        return TCL_ERROR;
    }

    Tcl_Obj* result = NULL;

    char* arg1 = Tcl_GetString(objv[1]);
    char* arg2 = Tcl_GetString(objv[2]);

    if (strcmp(arg1, arg2) == 0)
    {
        result = Tcl_NewBooleanObj(1);
    }
    else
    {
        result = Tcl_NewBooleanObj(0);
    }

    Tcl_SetObjResult(interp, result);

    return TCL_OK;
}

/*
* @breif Tcl_AppInit is called from Tcl_Main after the Tcl interpreter has been created,
*          and before the script file or interactive command loop is entered.
*/
int Tcl_AppInit(Tcl_Interp* interp)
{
    // Initialize packages Tcl_Init sets up the Tcl library facility.
    if (Tcl_Init(interp) == TCL_ERROR)
    {
        return TCL_ERROR;
    }

    // Register application-specific commands.
    Tcl_CreateObjCommand(interp, "randomcmd", RandomCmd, NULL, NULL);
    Tcl_CreateObjCommand(interp, "equalcmd", EqualCmd, NULL, NULL);

    return TCL_OK;
}

int main(int argc, char* argv[])
{
    Tcl_Main(argc, argv, Tcl_AppInit);

    return 0;
}
```

程序效果如下图所示：

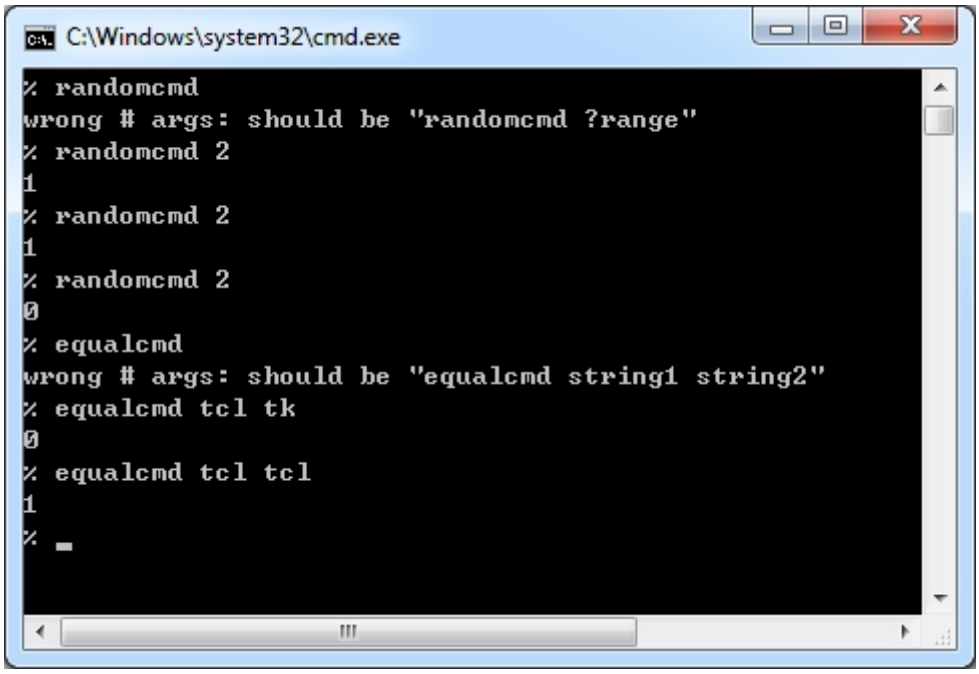

Figure 3.1 User defined command in Tcl

结合上面的代码来理解 Draw Test Harness 中自定义命令的实现可以事半功倍。当然也可以在 C 程序中使用 Tk 来定义用户界面 UI。详细信息请参考 References 中罗列的参考资料。

由上可知，使用 Tcl/Tk 可以使自己的程序实现命令行的功能，如 AutoCAD、PDMS 中都有这种功能，用户通过输入命令来实现一定的功能，很方便，如下图所示为 PDMS 中的命令窗口：

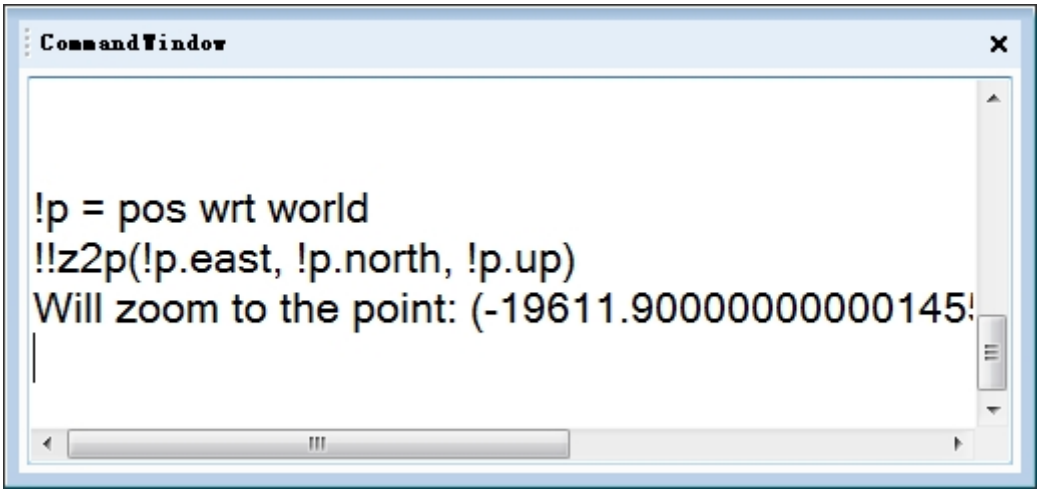

Figure 3.2 Command Window of AVEVA Plant/PDMS

四、结论 Conclusion

通过在程序中使用 Tcl 实现自定义的命令，理解 Tcl 中 C 接口的用法。在此基础上来理解 OpenCascade 中 Draw Test Harness 的实现要更容易。

通过学习 Tcl/Tk 可知，Tcl 可以作为程序的二次开发语言，类似 PDMS 中的 PML。Tcl 中自定义命令方便，还可对表达式进行解析并求值，功能相当强大。缺点就是在 Tcl 中面向对象的功能要差点儿，如在 Tcl 脚本中自定义一个对象，像在 PML 中可以这样来自定义对象，而在 Tcl 中这种自定义类型是不支持的：

```
define object factory
    member .mName is string
    member .mWorkers is real
    member .mOutput is real
endobject

define method .SetName(!name is string)
    !this.mName = !name
endmethod
```

Figure 4.1 User Defined Object in PDMS PML

好像 Tcl 也有对面向对象的加强库 TclOO，基本用法如图所示：

```
oo::class create Toaster {
    variable toasting time
    constructor {toastingTime} {
        set time $toastingTime
        set toasting ""
    }
    method toast {breadProduct} {
        if {$toasting ne ""} {
            error "already toasting something"
        }
        set toasting [after $time [namespace code [list \
                my Toasted $breadProduct]]]
        puts "toasting $breadProduct for you"
    }
    method Toasted {breadProduct} {
        puts "toasted the $breadProduct"
        set toasting ""
    }
    destructor {
        after cancel $toasting
    }
}

Toaster create quickToaster 30000 ;  # 30 seconds only
quickToaster toast crumpet
```

Figure 4.2 Basic Usage of TclOO

五、参考资料 References

1. Tcl and the Tk Toolkit
2. Practical Programming in Tcl and Tk
3. Tcl/Tk A Developer's Guide
4. http://sourceforge.net/projects/tcl/
5. http://www.tcl.tk/