LibHand v0.9

Copyright (c) 2011, Marin Saric <marin.saric@gmail.com>

All rights reserved.

<http://www.libhand.org/>

论文：M. Saric, “Libhand: A library for hand articulation,” 2011, version 0.9.

LibHand v0.9.z

Copyright (c) 2014-, Various Authors

Maintainer: Shasheen Ediriweera <shasheene@gmail.com>

All rights reserved.

http://github.com/libhand/libhand -- the most up-to-date information on LibHand.

v0.9z支持Windows系统，v0.9不支持

Chocolatey安装：在管理员权限的PowerShell下安装：

1. 先将PowerShell的脚本执行策略改为ByPass

set-executionpolicy ByPass

set-executionpolicy Restricted

get-executionpolicy

set-executionpolicy AllSigned

1. 输入安装命令：

[System.Net.WebRequest]::DefaultWebProxy.Credentials = [System.Net.CredentialCache]::DefaultCredentials; iex ((New-Object System.Net.WebClient).DownloadString('https://chocolatey.org/install.ps1'))

(3)安装完后将PowerShell的脚本执行策略改为默认的Restricted

Tcl 是“工具控制语言（Tool Command Language）”的缩写，是一种很通用的脚本语言，它几乎在所有的平台上都可以解释运行，功能强大，发音为 "tickle”。

Tk是Tcl“图形工具箱”的扩展，它提供各种标准的GUI接口项，以利于迅速进行高级应用程序开发。

TCL的解释器是用一个CC++语言的过程库实现的, 因此在某种意义上我们又可以把TCL看作一个C库

TCL目前已成为自动测试中事实上的标准.

TCL 的每一个命令包含一个或几个单词, 第一个单词代表命令名, 另外的单词则是这个命令的参数, 单词之间必须用空格或 TAB 键隔开

Shell本身是一个命令解释器

**Shell有两种执行命令的方式：**

* 交互式（Interactive）：解释执行用户的命令，用户输入一条命令，Shell就解释执行一条。
* 批处理（Batch）：用户事先写一个Shell脚本(Script)，其中有很多条命令，让Shell一次把这些命令执行完，而不必一条一条地敲命令。Shell脚本有循环和控制语句，解释执行，不需要编译。

**常见的Shell脚本解释器：**

    有bash、sh、csh、ksh等。我们说的Shell其实是指Shell脚本解释器。

* bash: Linux默认的shell,40个内部命令，完全兼容sh
* sh：Unix标准默认shell
* ash：最小的shell,只有24个内部命令，使用起来会不足
* csh：52个内部命令，指向/bin/tcsh，其实是tcsh.
* ksh:42个内部命令，与商业发行版兼容。

**Shell脚本**

     文件名以.sh结尾，  第一行用#! 开头，告诉系统用哪个解释器执行，例如下面的语句。

#!/bin/bash

echo "Hello World !"

Bash Shell语法：

set –e

在"set -e"之后出现的代码，一旦出现了返回值非零，整个脚本就会立即退出

set -o pipefail

表示在管道连接的命令序列中，只要有任何一个命令返回非0值，则整个管道返回非0值，即使最后一个命令返回0

（1）shell脚本中定义的变量是global的，其作用域从被定义的地方开始，到shell结束或被显示删除的地方为止。

（2）shell函数定义的变量默认是global的，其作用域从“函数被调用时执行变量定义的地方”开始，到shell结束或被显示删除处为止。函数定义的变量可以被显示定义成local的，其作用域局限于函数内。但请注意，函数的参数是local的。

（3）如果同名，Shell函数定义的local变量会屏蔽脚本定义的global变量。

Shell 函数的使用

HELLO="Hi! How are you"

function greeting()

{

local HELLO=NiHao

echo ${HELLO}

}

echo ${HELLO}

greeting

echo ${HELLO}

运行结果：

Hi! How are you

NiHao

Hi! How are you

if-else语句：

和[命令一样，要注意命令和各参数之间必须用空格隔开

使用if、then、elif、fi关键字

if [ 10 -gt 90 ];

then

echo "Good, 10"

elif [ 100 -gt 70 ];

then

echo "OK, 100"

else

echo "Bad"

fi

while循环：

sum=0

i=5

while [ $i -gt 1 ]; do

let "sum= $sum+$i"

let "i=$i -1"

done

echo "sum is:"$sum #运行结果：14

创建一级目录

mkdir dir1

如果dir1目录 已经存在，会提示目录已经存在

创建多级目录

mkdir -p dir1/dir2

如果某级目录不存在，则创建之

 pwd命令，该命令的作用是“print name of current/working directory”，这才是此命令的真实含义，当前的工作目录

DOWNLOAD\_CACHE=$(echo 'pwd'/download\_cache)

mkdir -p ${DOWNLOAD\_CACHE}

在当前的工作目录下新建名为”download\_cache”的文件夹

安装：

-- Found the following Boost libraries:

-- filesystem

-- thread

-- system

-- chrono

-- date\_time

-- atomic

status\*\* Boost include D:/libhandv09z/third\_party/boost\_1\_58\_0

status\*\* Boost libdirs D:/libhandv09z/third\_party/boost\_1\_58\_0/stage/lib

-- Using local OGRE install in CMake lookup path

-- CMAKE\_MODULE\_PATHS is D:/libhandv09z/hand\_cpp/modules;D:/libhandv09z/hand\_cpp /modules;D:/libhandv09z/third\_party/OgreSDK\_vc11\_v1-9-0/CMake

-- OpenCV ARCH: x86

-- OpenCV RUNTIME: vc14

-- OpenCV STATIC: OFF

-- Found OpenCV: D:/opencv/build (found version "3.2.0")

-- Found OpenCV 3.2.0 in D:/opencv/build/x86/vc14/lib

Could NOT find PkgConfig (missing: PKG\_CONFIG\_EXECUTABLE)

-- Found OGRE: optimized;D:/libhandv09z/third\_party/OgreSDK\_vc11\_v1-9-0/lib/Rele ase/OgreMain.lib;debug;D:/libhandv09z/third\_party/OgreSDK\_vc11\_v1-9-0/lib/Debug/ OgreMain\_d.lib

OpenCV3.2.0&contrib扩展库含x86x64可直接使用

已修改为calcMotionGradient

OGRE (Object Orientated Graphics Rendering Engine) v1.7.4

OpenGL headers (GL/glu.h)

由于找不到OgreMain.dll，无法继续执行代码。重新安装程序可能会解决此问题

OgreSDK\_vc11\_v1-9-0

LibHand v0.9

先使用CMake build 依赖库，源码位置即解压后的文件夹，生成的二进制文件存放位置/lib\_msvc14/ogredeps\_1\_9，点击配置生成

如果报错先看英文教程，再google

打开VS，先修改常量snprintf的定义，再F7生成solution，确保成功再右键INSTALLbuild，完成后ogredeps\_1\_9\ogredeps下有bin、lib、include文件夹

将文件夹ogredeps复制到OGRE源码目录中,并修改文件名ogredeps为Dependencies

再CMake configure、generate，生成的二进制文件放于桌面ogre1.9文件夹，再该文件夹中打开Ogre.sln

右键生成SampleBrowser工程时，机器内存不够，崩溃

解决方案：按照项目的生成顺序，依次在项目名右键-生成，当进行到SampleBrowser项目时，也成功完成，同时C:\Users\myth\Desktop\OGRE1.90\bin\debug下出现了SampleBrowser.exe文件

运行SampleBrowser.exe出现错误提示 Could not load dynamic library Plugin\_CgProgramManager\_d.

分析：

OGRE需要Nvidia Cg工具包来构建CgProgramManager插件

等等，所以你在建立Ogre 之后安装了nvidia cg工具包？难怪Cg插件没有建立...

你需要重建Ogre。

解决方案安装navida cv工具包

navida cv工具包又要求Directx 9，因此安装DirectX 9.0 SDK和Direcx 9.0

先使用CMake build 依赖库，源码位置即解压后的文件夹，生成的二进制文件存放位置/lib\_msvc14/ogredeps\_1\_9，点击配置生成

如果报错先看英文教程，再google

打开VS，先修改常量snprintf的定义，再F7生成solution，确保成功再右键INSTALLbuild，完成后ogredeps\_1\_9\ogredeps下有bin、lib、include文件夹

将文件夹ogredeps复制到OGRE源码目录中,并修改文件名ogredeps为Dependencies

再CMake configure、generate，生成的二进制文件放于桌面ogre1.9文件夹，再该文件夹中打开Ogre.sln

if [ "$SKIP\_BUILD\_LIBHAND" = false ]; then

build\_libhand

fi

if [ "$SKIP\_BUILD\_LIBHAND\_EXAMPLES" = false ]; then

build\_libhand\_examples

fi

}

binary\_ogre\_mode

build\_libhand

run\_cmake ..

\*\*\*\*\* VIDEOINPUT LIBRARY - 0.1995 - TFW07 \*\*\*\*\*

Exception: OGRE EXCEPTION(7:InternalErrorException): Could not load dynamic library Plugin\_OctreeZone. System Error: 找不到指定的模块。

in DynLib::load at D:\OGRE1\_9\_0\OGREx86\lib\_source\sinbad-ogre-dd30349ea667\OgreMain\src\OgreDynLib.cpp (line 109)

请按任意键继续. . .