```
# Copyright (c) 2001-2014 WorldViz LLC.
# Copyright (c) 2001-2014 WorldViz LLC.
# All rights reserved.

"""
WorldViz Universal Viztracker version 4.0

This file is designed to provide a tracker abstraction that most users will find helpful.
In most
cases, users can use the graphical setup utility to generate a configuration file, and thi
s viztracker
library will then read it in and implement it.

By default, if you do not change anything in the standard Vizard install, then viztracker
will attempt to
look for a configuration file vizsetupcfg.py in the current directory, in the user's Deskt
op folder, or
in the user's %APPDATA%\WorldViz\VizSetup folder. If vizsetupcfg.py does not exist in any
of these locations
then viztracker will use normal desktop defaults.

If you wish, you can edit this file and activate the generic construction method createCus
tomComposite() below,
where you can put whatever code you want in for defining your own trackers. It shows a typ
ical use case that
is easy to modify. You should use this if you are supporting a configuration that is not i
n the graphical
setup tool, and also for complex environments with multiple users or multiple displays.

This viztracker.py also provides a set of default keystrokes that are available:
    Alt-r - reset the orientation heading if a magnetic sensor is connected
    Alt-p - reset the current position to be the origin
    Alt-h - toggles the hand on, off, and auto hide modes, by default is automatic
    Scroll-Lock - activates keyboard tracker to offset the user
    Numeric-keypad - controls the user offset with the keyboard
    Numeric-5 - resets the keyboard tracker to defaults
    Left-Ctrl - activates external viewpoint window, cycles through
    Left-Shift - activates tracker debug window with detailed information

Other notes:
    - There are some variables at the top of this file that control some internal features
    - If you run this file by itself it will show a simple 3D demo for testing
    - This is designed to be backwards compatible with scripts assuming the previous Vizar
d 2.x/3.x viztracker interface
    - The file %APPDATA%\WorldViz\VizSetup\vizsetupcfg.py is created by the graphical setu
p tool
    - User applications should call viztracker.go() instead of viz.go() to initialize Viza
rd
"""


import viz


# The viztracker supports an external view with the left control key, this is where it wil
l view from, and the key used (use None for no key)
VIEWPOINT_EXTERNAL_POSITION = [-3, 1, 3]
VIEWPOINT_EXTERNAL_KEY = viz.KEY_CONTROL_L

# The VUManager class supports a keystroke that can be used to toggle a window that shows
```

```
tracker debug information (use None for no key)
TRACKER_DEBUG_KEY = viz.KEY_SHIFT_L

# If we use one of the fallback composite objects, this variable controls if you will have
 a mouse hand or not
DEFAULT_HANDS = False

# If this function is renamed so it is missing the leading underscores, so the name is jus
t 'createCustomComposite' then
# viztracker will use this function to configure your tracker. It is useful for cases with
 very specific requirements
# not normally supported by the graphical setup tool
#
def __createCustomComposite(id=0): # Remove the __ before the name to activate!
    viz.logNotice('viztracker.py: Executing manually defined sample createCustomComposite(
) tracker function')

    # Configure to use an NVIS SX60 HMD
    import nvis
    nvis.nvisorSX60()

    # Configure for full screen mode, only on the first monitor or span
    viz.setOption ('viz.fullscreen', 1)
    viz.setOption ('viz.fullscreen.monitor', 1)

    # Get the position/orientation of all markers on the body via VRPN from PPT
    vrpn = viz.add('vrpn7.dle')
    PPT_VRPN = 'PPT0@PPT-PRODUCTION'
    head  = vrpn.addTracker(PPT_VRPN,0)
    lhand = vrpn.addTracker(PPT_VRPN,1)
    rhand = vrpn.addTracker(PPT_VRPN,2)
    lfoot = vrpn.addTracker(PPT_VRPN,3)
    rfoot = vrpn.addTracker(PPT_VRPN,4)
    hip   = vrpn.addTracker(PPT_VRPN,5)

    # Get orientation data from a directly connected InertiaCube2/3
    isense = viz.add('intersense.dle')
    headori = isense.addTracker(port=0)

    # Combine the PPT head marker with the IC2/3 orientation
    headcomb = viz.mergeLinkable(head,headori)

    # Create a virtual tracking node for the head that applies link offsets
    # Note that the link should use a low priority so it will take effect before anything
else
    headfinal = viz.addGroup()
    headlink = viz.link (headcomb, headfinal, enabled=False)
    import vizact
    vizact.onupdate (viz.PRIORITY_PLUGINS+1, headlink.update)
    headlink.postTrans([0,0.1,0]) # Apply 10 cm translate in Y axis

    # Open up a glove device and attach a hand prop to the right hand
    immersion = viz.add('immersion.dle')
    rglove = immersion.addCyberGlove()
    # The vizuniverse library knows how to link up the glove to the hand internally, so th
ere is no need to do anything further with it

    # Create a composite object and fill it with all the objects we have just created
    comp = VU.VUCompositeTrackers()
    comp.storeTracker(comp.HEAD,  headfinal)
    comp.storeTracker(comp.LHAND, lhand)
    comp.storeTracker(comp.RHAND, rhand)
    comp.storeTracker(comp.LFOOT, lfoot)
    comp.storeTracker(comp.RFOOT, rfoot)
    comp.storeTracker(comp.HIP,   hip)
    comp.createRightHand (rglove)

    # Define an avatar representing the user
    comp.createAvatarNone()

    # Define a viewpoint that by default attaches to the user's head
    comp.defineViewpoint()

    # Finish off the composite and return it
    comp.finishTrackers()
    return comp




#
# ---- DO NOT EDIT ANYTHING BEYOND THIS POINT, THE ABOVE VARIABLES AND CODE CONTROL EVERYT
HING IN THIS FILE ----
#

# This script uses many recent Vizard features, so this version check is important
viz.requireVersion ("3.16.0009")

# Bring in the vizuniverse library to provide the classes we need
import vizuniverse as VU
import hand


def init(mode=0):
    """Declare a function that sets up our tracking, input, and display infrastructure"""
    global createCustomComposite

    # Detect if there is a manager in vizuniverse already, if so then skip this step
    if getManager() is not None:
        if mode is not 0:
            VU.FatalError("The viztracker.go() statement was called with arguments but the
 manager has already been initialized\nMake sure that you call viztracker.go() before any
other viztracker.add() calls")
        viz.logNotice("Detected a previously configured vizuniverse manager, so will reuse
 that and ignore viztracker settings")
        return

    # Create a default manager and keyboard controller which are always used
    manager = VU.VUManager(debugkey=TRACKER_DEBUG_KEY)
    keypad = VU.VUTrackerNumericKeypad (startpos=[0,0,0], starteuler=[0,0,0])

    # Create a composite variable that we will now work on configuring in the following st
eps
    comp = None

    # Try to execute the local createCustomComposite() function first
    try:
        compfunc = createCustomComposite # This should throw an exception
    except NameError:
        compfunc = None
        comp = None
    if compfunc is not None:
        comp = compfunc()
        viz.logNotice('Using createCustomComposite() in viztracker, which has been loaded
and created a composite')

    # If the composite is still not valid then we need to look in the vizsetupcfg.py files
 for configuration settings
    if comp is None:

        # Add the user's Desktop and local APPDATA folder to the default path, and we will
 try to load any configuration files from there
        # We search the desktop first, fall back to the APPDATA folder which is where the
demo CD writes to, and while we search the
        # remaining Vizard path, these files are not included in the installer, so if noth
ing is found we have sane defaults in here.
        # Note that the current directory is also included in the search path as well, so
this can also be used.
        import os
        import sys
        desktopDataPath = os.path.join(os.environ['USERPROFILE'],'Desktop')
        sys.path.append(desktopDataPath)
        appDataPath = os.path.join(os.environ['APPDATA'],'WorldViz','VizSetup')
        sys.path.append(appDataPath)

        # Now lets try to import vizsetupcfg and see if it works for us
        try:
            import vizsetupcfg
            viz.logNotice("Found %s file, using supplied createCustomComposite() function"
 % vizsetupcfg.__file__.replace(os.environ['USERPROFILE'], '%USERPROFILE%').replace(os.env
iron['APPDATA'], '%APPDATA%'))
            if hasattr(vizsetupcfg, "createCustomComposite"):
                createCustomComposite = vizsetupcfg.createCustomComposite
            else:
                createCustomComposite = None
                VU.FatalError ("There is no createCustomComposite() method in the vizsetup
cfg.py file, delete the file and regenerate it using the latest setup tool")
        except ImportError:
            # If we could not find the vizsetupcfg.py file, then we need to use the hard c
oded defaults that follow
            createCustomComposite = None

        # Now try executing the code and generate a composite
        if createCustomComposite is not None:
            # Execute the composite creation code, which is responsible for creating a var
iable called "comp"
            comp = createCustomComposite()

            # Did we get a composite? If not then some kind of error has occured but this
should not be possible, any exception would have happened at the exec()
            if comp is None:
                VU.FatalError ("The createCustomComposite() in vizsetupcfg.py did not crea
te a valid composite object, delete the file and regenerate it")

        # In this case, we failed to find a createCustomComposite() or failed to find vizs
etupcfg.py, so lets just create a default composite
        else:
            comp = VU.VUCompositeTrackers()
            if DEFAULT_HANDS is True:
                viz.logNotice("Using default configure values for desktop with mouse-based
 hand")
                comp.storeTracker (comp.HEAD, VU.VUTrackerVirtual(startpos=[0,1.8,0], type
='Keyboard', fixedHeight=False))
                comp.tracker_output[comp.LHAND] = VU.VUTrackerMouse2D()
                comp.copyHandOriFromHead();
                comp.createLeftHand (hand.MultiInputSensor (pinchButtons=[viz.KEY_PAGE_UP,
viz.KEY_PAGE_DOWN,viz.MOUSEBUTTON_LEFT,viz.MOUSEBUTTON_RIGHT], fistButtons=[viz.MOUSEBUTTO
N_MIDDLE]))
            else:
                viz.logNotice("Using default configure values for desktop keyboard/mouse c
ontrols")
                comp = VU.VUCompositeTrackers()
                comp.storeTracker (comp.HEAD, VU.VUTrackerVirtual(startpos=[0,1.8,0], type
='MouseKeyboard', fixedHeight=False))
            comp.finishTrackers()
            comp.setViewType(VU.VT_DESKTOP)
            comp.defineViewpoint()
            comp.createAvatarNone()

    # We will now have a valid composite object, so lets now add it to our global manager
    comp.addDriverNode(keypad)
    comp.finishTrackers() # Refinish the trackers to make sure the keypad is implemented
    manager.addComposite (comp, "MainUser")

    # Set up an external viewpoint so we can hit right shift to see different outside view
s
    ev = VU.VUExternalViewpoint(pos=VIEWPOINT_EXTERNAL_POSITION, key=VIEWPOINT_EXTERNAL_KE
Y, target=comp.getViewpointSource())

    # Make the manager active, and set internals so that it is accessible getManager()
    manager.go(mode)

    # Start up the VRPN server by default, so other systems can extract tracker data
    #manager.startVRPNServer()




# ---- The following code is the glue that makes things work like the old viztracker inter
face ----

def getManager():
    """Returns back a handle to the active manager, None if not created yet"""
    return VU.getManager()

def go(mode=0):
    """This method should be called if you want to setup display attributes and things lik
e that, is a replacement for calling viz.go()"""

    # If the user has supplied a mode value, generate an error if the manager has already
been init'ed earlier
    if getManager() is not None and mode is not 0:
        VU.FatalError("The viztracker.go() statement was called with arguments but the man
ager has already been initialized\nMake sure that you call viztracker.go() before any othe
r viztracker.add() calls")
    # We now initialize the manager if it does not exist
    if getManager() is None:
        init(mode) # Note that this function calls viz.go() via the VUManager class

def getHandList():
    """This method is used to retrieve a list of hands implemented by the viztracker file"
""

    # Make sure the manager is implemented
    if getManager() is None:
        init()
    # Return the hand list
    return getManager().getHandList()

def get(name):
    """Use the composite manager dictionary to look up a reference in the global list"""
    if getManager() is None:
        init()
    return getManager().get(name)

def _addTracker():
    """Internal method for requesting trackers for various body locations"""

    # Firstly, we need to make sure the manager is implemented
    if getManager() is None:
        init()

    # Grab a handle to the main user in the manager object
    mainuser = getManager().getComposite(0)

    # Each time the add() method is called, we return the corresponding tracker back, lega
cy viztracker interface
    viz.logNotice('Returning viztracker for head')
    yield mainuser.getTracker (mainuser.HEAD)

    viz.logNotice('Returning viztracker for left hand')
    yield mainuser.getTracker (mainuser.LHAND)

    viz.logNotice('Returning viztracker for right hand')
    yield mainuser.getTracker (mainuser.RHAND)

    viz.logNotice('Returning viztracker for left foot')
    yield mainuser.getTracker (mainuser.LFOOT)

    viz.logNotice('Returning viztracker for right foot')
    yield mainuser.getTracker (mainuser.RFOOT)

    viz.logNotice('Returning viztracker for the hip')
    yield mainuser.getTracker (mainuser.HIP)

_data = viz.Data()
_data.trackerGen = _addTracker()
_data.options = {}


def add(**kw):
    """Add a 6DOF tracker object"""
    _data.options = kw
    try:
        return _data.trackerGen.next()
    except StopIteration:
        viz.logError('** ERROR: viztracker has no more trackers')
        return None

def addPos(**kw):
    """Wrapper to add a 6DOF tracker object"""
    return add(kw)

def addOri(**kw):
    """Wrapper to add a 6DOF tracker object"""
    return add(kw)





# This viztracker file is also able to run itself and show a simple demo when not imported
 directly
if __name__ == "__main__":

    # Load the 3D world in
    viz.add ('gallery.osgb')

    # Use the above viztracker go() function, which will call viz.go() for us, and set up
viewpoints and everything else
    go()





# viztracker also includes a library of tracking functions that can be used by customers i
n their scripts, and
# so these are included here as well. You can use these without having to run any of the a
bove code.
from viztrackerutils import *
```