Возможности и принципы работы eventmanager. Данный модель может работать с **любыми**ивентами, и программами, **не только с NAO**. Итак, как что куда и для чего писать.

Библиотека состоит из двух классов, декоратора и двух функций(генераторов предикатов). Что к чему?

Обьект класса **Event** описывает само действие которое должно произойти при выполнении некоторых условий и в определенном виде. А точнее

**Event(function, args, predicate, single\_use, threadable, cooldown)**

* **function**
  + Функция которую необходимо будет запустить при выполнении каких-либо условий. Необходимо передавать в виде. Без скобочек()
    - **foo**
    - **myFunc**
* **args**
  + Аргументы передаваемые в функцию. Необходимо передавать в виде строки
    - **[a,2,b,c,”some text”]**
  + Не является обязательной, по стандарту передается пустая строка []
* **predicate**
  + Предикат, или другими словами условие. Можно передать в виде другой функции, возвращающий результат True/False, либо при помощи lambda:
    - **getMyPredicate**
    - **lambda: x == y**
  + Не является обязательной, по стандарту стоит значение False, однако с ним она никогда и не запустится
* **single\_use**
  + Параметр, отвечающий за тип ивента. Одноразовый или многоразовый ивент.
    - **True**
      * Одноразовый
    - **False**
      * Многоразовый
  + Не является обязательным, по стандарту имеет значение **False** = многоразовый
* **threadable**
  + Параметр отвечающий за вид обработки ивента. Создать новый поток и обработать его в потоке, либо обработать так, не создавая новый поток(нужно если ивент быстрый и простой)
    - **True**
      * В потоке
    - **False**
      * Без потока
  + Не является обязательным, по стандарту имеет значение **False** = выполнить без создания потока
* **cooldown**
  + Параметр отвечающий за задержку вызова одного и того же ивента.
    - 0
      * Нет задержки
    - **N**
      * Задержка в **N** секунд
  + Не является обязательным, по стандарту имеет стоит значение = 0 секунд(без задержки)

Класс EventGroup нужен для объединения ивентов в группу, если такое требуется. Например для установки таймера на исполнение одного из событий группы.

Имеет следующие возможности

* **name = EventGroup()**
  + Создает объект класса EventGroup()
* **name.addEvent(Event(…))**
  + Добавляет объект класса Event в группу ивентов name

Класс **Eventloop**, нужен для обработки объектов класса **Event**.

Имеет следующие возможности

* **name = Eventloop()**
  + Создает объект класса Eventloop
* **name.addGroup(EventGroup**)
  + Добавляет группу объектов класса Event на обработку
* **name.addEvent(Event(…))**
  + Добавляет объект класса Event в обработку
* **name.start()**
  + Запускает обработку ивентов
* **name.stop()**
  + Останавливает обработку ивентов
* **name.join()**
  + Ждет окончания обработки ивентов

Примеры использования

# -\*- encoding: UTF-8 -\*-

import time

import threading

import random

from threading import Thread

from eventmanager import \*

from naoqi import ALProxy

from naoqi import ALBroker

from naoqi import ALModule

def eventRuka(robot\_IP, robot\_PORT=9559):

    global tts, memory, motion, alife, touch, posture

    tts = ALProxy("ALTextToSpeech", robot\_IP, robot\_PORT)

    memory = ALProxy("ALMemory",robot\_IP,robot\_PORT)

    motion = ALProxy("ALMotion", robot\_IP, robot\_PORT)

    posture = ALProxy("ALRobotPosture", robot\_IP, robot\_PORT)

    alife = ALProxy("ALAutonomousLife", robot\_IP, robot\_PORT)

    touch = ALProxy("ALTouch", robot\_IP, robot\_PORT)

    global phrases      #   фразы при движении

    phrases = {

        "general": ["what a great day", "and you are good at copying me human", "good", "I'm not just some kind of robot as you might think, I'm something more"],

        "general\_move": ["ok let's go here", "I'm tired, let's stand", "lead me", "you walk well, for human of course. ha ha ha" , "not tired yet, human?", "lead me flesh bag", "lead me hairless ape", "don't hurt my hand"],

        "forward": ["wait i'm tired", "ok let's go forward", "text"],

        "backward": ["text", "ok let's go backward", "text"],

        "turn\_left": ["I got dizzy", "ok let's go left", "text"],

        "turn\_right": ["I got dizzy", "ok let's go right", "text"],

        "forward\_left": ["Is it just me or are we walking in circles?", "are we walking in circles?", "I'm having fun with you, but we go in circles", "Port!"],

        "forward\_right": ["Is it just me or are we walking in circles?", "are we walking in circles?", "I'm having fun with you, but we go in circles", "Starboard!"],

        "backward\_left": ["Is it just me or are we walking in circles?", "are we walking in circles?", "I'm having fun with you, but we go in circles"],

        "backward\_right": ["Is it just me or are we walking in circles?", "are we walking in circles?", "I'm having fun with you, but we go in circles"],

        "stop": ["why are we standing?", "can we go already?", "tired of just standing still", "come on, take my hand and go for a walk", "and we stand again", "sounds of falling asleep NAO"]

    }

    alife.setState("disabled")

    motion.stopMove()

    motion.wakeUp()

    motion.setMoveArmsEnabled(True, False) #(left,right)

    tts.post.say("Take my hand and let's go for a walk")

    motion.post.setAngles("RShoulderPitch", -1.5, 0.4)

    motion.post.setAngles("RShoulderRoll", 0, 0.4)

    motion.openHand("RHand")

    global loop

    loop = True

    global eventHandler

    #   Создание экземпляра класса Eventloop

    eventHandler = Eventloop()

    #   eventHandler.addGroup(EventGroup)

    #   eventHandler.addEvent(Event(lambda: Предикат(проверка), Функция, Аргументы, single\_use=False, threadable=True, cooldown=0))

    #   Создание групп ивентов

    generalGroup = EventGroup()

    motionGroup = EventGroup()

    ttsGroup = EventGroup(cooldown = 5)

    #   generalGroup.addEvent(Event(lambda: Предикат(проверка), Функция, Аргументы, single\_use=False, threadable=True, cooldown=0))

    #   Создание и добавление ивентов на обработку (по стандарту внутри уже создана группа для ивентов, не требующих определенную группу)

    eventHandler.addEvent(Event( naoExit, [], binaryPredicate(lambda: touch.getStatus()[8][1], True, False), single\_use=True))

    eventHandler.addEvent(Event( touchMyHand, [], binaryPredicate(lambda: touch.getStatus()[18][1], True, False), single\_use=True, threadable=True))

    #   Создание и обьединение ивентов в группы

    motionGroup.addEvent(Event( motion.moveTo, [10,0,0], getForward , threadable=True))

    ttsGroup.addEvent(Event( walkPhrases, ["forward"], getForward , threadable=True, cooldown=10))

    motionGroup.addEvent(Event( motion.moveTo, [-10,0,0], getBackward, threadable=True))

    ttsGroup.addEvent(Event( walkPhrases, ["backward"], getBackward , threadable=True, cooldown=10))

    motionGroup.addEvent(Event( motion.moveTo, [0,0,0.78], getLeft, threadable=True))

    ttsGroup.addEvent(Event( walkPhrases, ["turn\_left"], getLeft , threadable=True, cooldown=10))

    motionGroup.addEvent(Event( motion.moveTo, [0,0,-0.78], getRight, threadable=True))

    ttsGroup.addEvent(Event( walkPhrases, ["turn\_right"], getRight , threadable=True, cooldown=10))

    motionGroup.addEvent(Event( motion.moveToward, [1,0,0.5,[["Frequency", 0.1]]], getForwardLeft, threadable=True))

    ttsGroup.addEvent(Event( walkPhrases, ["forward\_left"], getForwardLeft , threadable=True, cooldown=10))

    motionGroup.addEvent(Event( motion.moveToward, [1,0,-0.5,[["Frequency", 0.1]]], getForwardRight, threadable=True))

    ttsGroup.addEvent(Event( walkPhrases, ["forward\_right"], getForwardRight , threadable=True, cooldown=10))

    motionGroup.addEvent(Event( motion.moveToward, [-1,0,-0.5,[["Frequency", 0.1]]], getBackwardLeft, threadable=True))

    ttsGroup.addEvent(Event( walkPhrases, ["backward\_left"], getBackwardLeft , threadable=True, cooldown=10))

    motionGroup.addEvent(Event( motion.moveToward, [-1,0,0.5,[["Frequency", 0.1]]], getBackwardright, threadable=True))

    ttsGroup.addEvent(Event( walkPhrases, ["backward\_right"], getBackwardright , threadable=True, cooldown=10))

    motionGroup.addEvent(Event( motion.stopMove, [], getStopMove))

    ttsGroup.addEvent(Event( walkPhrases, ["stop"], getStopMove , threadable=True, cooldown=20))

    #   Добавление групп в обработчик ивентов

    eventHandler.addEventGroup(motionGroup)

    eventHandler.addEventGroup(ttsGroup)

    #   Запуск обработчика ивентов

    eventHandler.start()

    #   Ожидание обработчика ивентов

    eventHandler.join()

    alife.setState("interactive")

def walkPhrases(state):

    tts.say(random.choice(phrases["general"] + phrases[state] + [i for i in phrases["general\_move"] if state != "stop"]))

def touchMyHand():

    motion.post.stiffnessInterpolation("RShoulderPitch",0.0,1.0)

    motion.post.stiffnessInterpolation("RShoulderRoll",0.0,1.0)

    motion.post.stiffnessInterpolation("RWristYaw",0.0,1.0)

    motion.closeHand("RHand")

    tts.say("ok, lead me")

def naoExit():

    global loop

    eventHandler.stop()

    loop=False

    motion.stopMove()

    posture.goToPosture("Stand",0.4)

def getRShoulderPitch():

    return motion.getAngles("RShoulderPitch",False)[0]

def getRShoulderRoll():

    return motion.getAngles("RShoulderRoll",False)[0]

def getForward():

    return -0.85 < getRShoulderPitch() and (-0.2 <= getRShoulderRoll() <= 0.1) and not getRShoulderPitch()>0.2 and loop

def getForwardLeft():

    return -0.85 < getRShoulderPitch() and 0.1 < getRShoulderRoll()  and not getRShoulderPitch()>0.2 and loop

def getForwardRight():

    return -0.85 < getRShoulderPitch() and getRShoulderRoll() < -0.2  and not getRShoulderPitch()>0.2 and loop

def getBackward():

    return getRShoulderPitch() < -1.75 and (-0.2 <= getRShoulderRoll() <= 0.1) and not getRShoulderPitch()>0.2 and loop

def getBackwardLeft():

    return getRShoulderPitch() < -1.75 and 0.1 < getRShoulderRoll()  and not getRShoulderPitch()>0.2 and loop

def getBackwardright():

    return getRShoulderPitch() < -1.75 and getRShoulderRoll() < -0.2  and not getRShoulderPitch()>0.2 and loop

def getLeft():

    return (-1.75 <= getRShoulderPitch() <= -0.85) and 0.1 < getRShoulderRoll() and not getRShoulderPitch()>0.2 and loop

def getRight():

    return (-1.75 <= getRShoulderPitch() <= -0.85) and getRShoulderRoll() < -0.2 and not getRShoulderPitch()>0.2 and loop

def getStopMove():

    return not(getForward() or getForwardLeft() or getForwardRight() or getBackward() or getBackwardLeft() or getBackwardright() or getLeft() or getRight())

eventRuka("192.168.252.226", 9559)

# eventRuka("NaoThree.local", 9559)