Microbit 编程技术报告-C106

From microbit to Switch:钟表，太鼓达人与舞力全开

一、选题及创意介绍

「Part1: Anytime, Anywhere」

BBC micro:bit 就像一个手掌大小的交互硬件，它有 LED 灯显示屏、按钮、传感器和许多输入/输出功能。当输入代码后时，它可以与你和你的世界互动。这与任天堂对 Switch 的设想非常像：Play In Brilliant Style—Anytime, Anywhere。我们同样希望，我们手中的 micro:bit 可以变成一个可以随时随地玩耍，为玩家提供乐趣的掌上游戏机。尤其是在这一段艰难而封闭的时间之中，这样的随时随地的快乐显得尤为重要。

「Part2:Clock makes difference!」

时钟是人类最早的机械工程的杰作之一。早在 1300 年左右，这个世界上已经有时钟存在。在之后的几百年，人们不断致力于提高它的精度，设计成更精巧的机械结构。跨越千年时空，直到今天，人类的机械工程和其他的工程形式为这个世界上提供了很多精妙而准确的工业产物，microbit 也是其中之一。无数的传感器，电源，输出接口……存在于这一块精巧的小面板上。

因此，为了表达对 microbit 和人类机械工程的敬意，我们特此为这台游戏机设计了时钟功能。时钟功能齐全，有时钟，有摆锤，有三问，有指南针。

「Part3: 太鼓の達人」

太鼓达人是 Switch 旗下最火的游戏 IP 之一，是鼻祖级的音游。在传统的 Switch 太鼓达人中，玩家需要在左右两边进行“咚”“咔”两种形式的打鼓，其中“咚”是敲打鼓面，而“咔”是敲打鼓的侧面。我们利用 micro:bit 的两个按键 AB 和下面的接口，同样实现了“咚”“咔”。在 micro:bit 上玩太鼓达人不是梦！

「Part4: Just Just Dance!!」

Just Dance 是育碧 Ubisoft 开发的一款音乐节奏游戏，通过动作捕捉和音乐的结合的音乐体感游戏。我们利用 micorbit 的加速计、重力计在 micro:bit 上实现了 Just Dance。并且，我们将 Just Dance 的“跟随屏幕上的人物跳舞”的游玩方式改成了玩家自己录制标准动作，再跳舞并进行舞蹈评分的过程，使 DIY 成为可能，让每个人都称为自己的舞蹈家。

二、设计方案和硬件连接

Micro:bit 在静息状态下为钟表的走时效果，可以显示时间，摆锤和指南针。当按下特定的按键时，进入游戏《太鼓の達人》或《Just Just Dance》。

若进入游戏《太鼓の達人》，则 micro:bit 开始播放音乐，玩家需要根据 microbit 面板上的像素点来按相应的按键，具体规则为：从左至右的 5 行分别对应 A，0，1，2，B。

若进入游戏《舞力全开》，则 micro:bit 首先会初始化磁力计，然后让玩家根据音乐自由录制一段动作，这段动作便被作为“标准动作”。然后玩家再进行舞蹈，当玩家跳完后，microbit 将自动对舞蹈和标准动作的相似度进行比较，并折算成绩点输出。

三、实现方案及代码分析：

我们的期望是在初始状态下可以让 Microbit 呈现出钟表的走时效果，然后通过特定的按键进入不同的功能，且在进入较为简单的功能后可以不重启地回到初始状态。

总体思路是先将游戏及功能模块化，再用总的控制程序将他们整合起来。

先上代码(说明部分用红字)

```
#this program is made up of three parts:

#1.a mechanical watch that indicates time and shows the movement of gear

#2.built_in small games

#3.Just_Dance


from microbit import *

import music,speech,random



#main

def button_judge():用于按键识别的函数，整个程序用五个与按键一一对应的全局变量连接

    global buttonA,buttonB,pinZero,pinOne,pinTwo


    if button_a.is_pressed():

        buttonA=1
```

```
    if button_b.is_pressed():
        buttonB=1
    if pin0.is_touched():
        pinZero=1
    if pin1.is_touched():
        pinOne=1
    if pin2.is_touched():
        pinTwo=1


def control_center():总控制函数
    global buttonA,buttonB,pinZero,pinOne,pinTwo

    buttonA=0#compass
    buttonB=0#triquestion
    pinZero=0#dont_touch_whiteblock
    pinOne=0#Just_Dance
    pinTwo=0 启动时将所有全局变量全部重置

    while True:主循环，让程序一直运行
        watch_run()初始时执行 watch_run 函数，这个函数自带的判断功能使得在没有任何按键被按下时，一直循环显示时钟的走时效果

        #进入这一阶段说明 watch_run()已经结束了
        if buttonA==1 and max(buttonB,pinZero,pinOne,pinTwo)==0:
            use_compass()
        elif buttonB==1 and max(buttonA,pinZero,pinOne,pinTwo)==0:
            triquestion()
        elif pinZero==1 and max(buttonA,buttonB,pinOne,pinTwo)==0:
            dont_touch_whiteblock()
        elif pinOne==1 and max(buttonA,buttonB,pinZero,pinTwo)==0:
```

```
            Just_Dance()
        elif pinTwo==1 and max(buttonA,buttonB,pinZero,pinOne)==0:
            pass#remain to be chosen 一旦某个按键被按下，watch_run 函数随即终止，然后这段代码根据你按下的按键来执行对应的函数，实现对应的功能，事实上按照我们的思路，不论多少功能都可以通过按键的不同组合来调用
```

```
def get_GPA(score,full_score,magic=True):#this returns a strType object 有的小游戏在最后会给出分数，这个函数将传入的数值经过调分后换算成北大绩点
    return str(4-(3*((100-((100*score/full_score)**(0.5 if magic else 1)*(10 if magic else 1)))**2)/1600))
```

```
#part 1:
def watch_run():这是模拟机械表走时的函数
    global buttonA,buttonB,pinZero,pinOne,pinTwo

    stop=1 这个 stop 用于判断是否要停止走时
    while stop:
        up=0
        down=0
        for i in range(0,5):设置了五个判断点，每个判断点都要判断是否有按键被按下，以决定是否停止模拟走时。
            if accelerometer.current_gesture()!='face down':#泛向上感应表盘的朝向，表盘向上时显示指针的走时效果
                up=1
                display.show(Image.ALL_CLOCKS[3*i:3*i+3],delay=200,wait=False,loop=False,clear=False)
            elif accelerometer.current_gesture()!='face up':#泛向下表盘向下时显示机械表摆锤的走动效果
                up=1
                display.show(Image.ALL_ARROWS[2*i:2*i+2],delay=200,wait=False,loop=False,clear=False)
            if up==1:
                sleep(600)这里 sleep 时值的选取使得不论是显示指针还是摆锤，程序都以相同的速率进行
            elif down==1:
```

```
            sleep(400)
        up=0
        down=0
        stop=0
        button_judge()执行按键检测程序，判断是否有按键被按下
        if max(buttonA,buttonB,pinZero,pinOne,pinTwo)!=0:如果有按键被按下，就结束程序
            break
        else:
            stop=1 否则继续运行程序



#part2
def use_compass():这是模拟指北针的程序
    global buttonA,buttonB

    button_a.was_pressed()
    button_b.was_pressed()

    compass.calibrate()
    while True:将磁力计的读数对应到指针的方向上，实现指北针
        display.show(Image.ALL_CLOCKS[((15-compass.heading())//30)%12])
        sleep(25)

        if button_a.was_pressed() and button_b.was_pressed():进行按键检测，如果按下了 a 和 b，就退出程序，回到钟表效果
            break
    buttonA=0
    buttonB=0



def triquestion():实现三问功能的函数
```

```
    global buttonB

    hour=random.randint(1,12)由于 microbit 无法在断电时计时，所以这里采用随机数模拟的形式，装逼的效果就已经达到了
    quarter=random.randint(0,3)
    minute=random.randint(0,14)

    for i in ['D5']*hour + ['B2']*quarter + ['C4']*minute:根据时间发出不同频率的声音，分别指示了 “时”，“刻”，“分”
        music.play(i)
        sleep(360)



    buttonB=0




def dont_touch_whiteblock():钢琴块小游戏
    display.clear()
    key_press=[]
    key_light=[]
    tune1=['C','C','G','G','A','A','G','F','F','E','E','D','D','C']内置了两首音乐，这首是小星星
    tune2=['E', 'E', 'F', 'G', 'G', 'F', 'E', 'D', 'C', 'C', 'D', 'E', 'E', 'D', 'D', 'E', 'E', 'F', 'G', 'G', 'F', 'E', 'D', 'C', 'C', 'D', 'E', 'D', 'C', 'C', 'D', 'D', 'E', 'C', 'D', 'E', 'F', 'E', 'C', 'D', 'E', 'F', 'E', 'D', 'C', 'E', 'G', 'E', 'E', 'E', 'F', 'G', 'G', 'F', 'E', 'D', 'C', 'C', 'D', 'E', 'D', 'C', 'C']这是欢乐颂
    tune_count=0
    #生成初始的
    for i in range(5):
        key_press.append(random.randint(1,5))随机生成将要出现的方块的位置
    #亮度是 3 和 9
    for i in range(5):随机生成方块的亮度
        key_light.append(random.randint(0,1)*3+6)
    button_press_a=0
```

```
button_press_b=0

button_press_pin0=0

button_press_pin1=0


music.play(music.PUNCHLINE)游戏开始前的简短音乐

t=running_time()记录当前的时间，用于之后的时值判断

score=0

while tune_count<len(tune1):设置循环，遍历完乐谱列表中的所有音符

    music.play(tune1[tune_count])发出单个音符

    image_present=' '初始化将要写入 LED 的字符串

    for i in range(5):一次生成一整张画面

        if key_press[i]==1:譬如说第一个数字是 1，就让第一列第一行亮起来

            image_present=image_present+str(key_light[i])+'0000:'

        elif key_press[i]==2:

            image_present=image_present+'0'+str(key_light[i])+'000:'

        elif key_press[i]==3:

            image_present=image_present+'00'+str(key_light[i])+'00:'

        elif key_press[i]==4:

            image_present=image_present+'000'+str(key_light[i])+'0:'

        elif key_press[i]==5:

            image_present=image_present+'0000'+str(key_light[i])+':'

    image_present=image_present[1:-1:]#去掉定义时的空格和最后的冒号

    #以上，完成生成一个此时的方块

    #然后其实我们关心最下面一层是什么


    music_image=Image(image_present)

    display.show(music_image)


    #按键反馈

    #1.5 秒没反应就 miss
```

```
reaction_start=running_time();
while running_time()-reaction_start<1500:按键判断，设置 1.5s 的时间限制，如果超过 1.5s 没有按键就进入下一个
    if button_a.is_pressed():
        button_press_a=1
        break
    if button_b.is_pressed():
        button_press_b=1
        break
    if pin0.is_touched():
        button_press_pin0=1
        break
    if pin1.is_touched():
        button_press_pin1=1
        break
#写个循环把剩余按键时间卡住,按着就卡住
while button_a.is_pressed() or button_b.is_pressed() or pin0.is_touched() or pin1.is_touched():手触摸按键的时间不算，这个循环在按键被按住时把时间卡住
    press=1

if (button_press_a==1 and key_press[4]%2==1 and key_light[4]==9)\
or(button_press_b==1 and key_press[4]%2==0 and key_light[4]==9)\
or (button_press_pin0==1 and key_press[4]%2==1 and key_light[4]==3)\
or (button_press_pin0==1 and key_press[4]%2==0 and key_light[4]==3):
    score+=1 如果及时按下了正确的按键，加一分

#update
button_press_a==0
button_press_b==0
button_press_pin0=0
button_press_pin1=0
```

key_press.pop(4)按下一个按键后将它从按键列表移除，并用 0 占位

key_press.insert(0,random.randint(1,5))

key_light.pop(4)

key_light.insert(0,random.randint(0,1)*6+3)

tune_count+=1

display.scroll(get_GPA(score,len(tune1),magic=True))将分数换算为绩点后在屏幕上显示

#part3

def Just_Dance():舞力全开小游戏，主要依据的是加速度计和指北针的方向识别

compass.calibrate()

def get_movement(time):这是记录动作的函数

facing=compass.heading()记录初始朝向，因为依据的是朝向的变化量而不是朝向的绝对量

lst=[]

t=running_time()记录起始的时间，因为我们要记录特定音乐时常内的动作

while running_time()-t<=time:

deviate_facing=compass.heading()-facing

lst.append((accelerometer.current_gesture(),deviate_facing))记录加速度计给出的状态和当前朝向与初始朝向的差值

sleep(100)每隔 0.1s 记录一次

return lst 返回记录了动作的列表

def compare_movement(lst1,lst2):这是比较两次动作相似度的函数

```
        score=0
        for i in range(min(len(lst1),len(lst2))):遍历列表
            angle_score=1-abs(lst1[i][1]-lst2[i][1])/360 给出角度对应的分数，即两次对应动作的朝向数值的差量，设定每次动作的角度分满分为 1
            score+=angle_score
            if lst1[i][0]==lst2[i][0]:如果两次加速度计记录的动作相同，则加 2 分
                score+=2
        return score,i*3 返回实际分数与总分，便于换算为百分制


    def start_game():这是开始游戏的函数
        music_list=['1.PYTHON']所有音乐的列表，这里只给出了一首，每首音乐都要与下面的音乐时常一一对应
        music_time=['12499']

        mu_str=' '.join(music_list)
        while not (button_a.is_pressed() and button_b.is_pressed()):在同时按下 a，b 按钮之前，滚动播放备选的音乐列表
            display.scroll(mu_str,loop=False,delay=100,wait=True)
        display.scroll('choose one!',wait=True,loop=False)之后示意玩家选择一首
        button_a.get_presses()重置 a，b 键的按键次数
        button_b.get_presses()
        sleep(3000)暂停三秒，让玩家有时间按键
        choice=button_a.get_presses()统计按键的次数来决定播放哪首音乐
        music.play(eval('music'+music_list[choice-1][1:]),wait=False)播放选定的音乐

        lst1=get_movement(12499)记录第一次的动作数据。这里 12499 是设置音乐时值，因为测试时只用了一首，所以在这里直接预设了时间

        display.show(Image.SMILE)展示笑脸
        while True:设置循环卡住时间，如果按下 a 键，则播放音乐
            if button_a.is_pressed():
```

```
                break
            display.clear()
            sleep(500)
            music.play(eval('music'+music_list[choice-1][1:]),wait=False)播放音乐

            lst2=get_movement(12499) 记录第二次的动作数据

            AllScore=compare_movement(lst1,lst2)由比较动作的函数给出最终分数
            display.scroll(get_GPA(AllScore[0],AllScore[1],magic=True))

    start_game()
control_center()执行总控制函数，开始整个程序的运行
```

附：一个小插曲

由于我们的演示过程涉及到很大的动作幅度，将 microbit 接在电脑上供电不太现实，而由于功率问题，充电宝供电极度不稳定，于是我去遥感楼借电池盒。期间在助教张赖和学长的帮助下，我们分析了 microbit 指北针读数失常的问题（只有在寝室用电脑供电时读数才是准确的），我们在反复刷机和分析奇奇怪怪的读数之后，达成共识：遥感楼的磁场很诡异，充电宝的供电电流不稳定，导致指南针读数反常。

四、后续工作展望

本小组已经初步实现了一个基于 micro:bit 的小型游戏机。在后续工作中，我们希望可以增强 microbit 的动作判定的准确性，我们的 microbit 的动作判定准确性目前还不是很好，这可能是因为 microbit 本身的磁力计并不是很精准。不过在后续工作中，也许我们可以尝试将磁力计和加速计结合起来判断。

此外，其实 switch 的其他游戏也很适合移植到 microbit 平台上：例如，经典的马里奥（只需要写障碍，然后按键控制马里奥，撞到障碍就归零），健身环（同样是体感，只不过需要结合两块 microbit，一块判定腿部运动，另一块判定手部运动）等。我们也很想尝试一下！

三、分工：

王启晗：编写了“太鼓の達人”部分的代码，同时给出了“Just Just Dance”另一个版本的代码，完成了视频、音频的剪辑，海报的制作，以及技术报告的第一、二、四部分（选题创意介绍，设计方案和后续工作展望），最后的总的文件的整理主要也是由她完成的。

刘梦航：编写了“Just Just Dance”、钟表及其他几个小功能的代码，完成了全部代码的整合，拍摄并出演了原始视频，完成了技术报告的第三、五部分（实现方案，代码分析和小组分工，以及致谢部分）。

合作愉快!!!

六、致谢：

1.无私的摄影师：向志丹，深情的配音师：程志琦

2.热情地给与我们很多帮助与指导的陈斌老师以及各位助教哥哥姐姐

3.帮助我们分析指北针异常的张赖和助教

4.还有其他的小伙伴们!

P.S. 王启晗的补充：其实我在这个课上不认识什么人，所以和队友完全是在群里捞到才因为这个作业认识的。但是合作实在是太愉快了，每一次讨论都高效、顺利而快乐。完成代码的过程也很快乐，其实写游戏的思路和写心理学实验程序的思路很像哈哈哈，就很自然，写的蛮开心的！也谢谢我的队友包容我的拖延症！

总而言之，整个作业的过程，高效，紧张刺激，但又轻松愉快，就像我现在，一边赶ddl，竟然还在一边听队友安利的歌一样。

人到大三，我不再将「如何获得更高的分数」作为自己的绝对目标。越来越觉得，带来快乐的作业，多少时间都是值得的。我可以自豪地说，这个 project 就是带来快乐的作业!