Shill 音游

作者：岳家庆、王世宁

摘要：

Shill 音游是我们开发的一款音游小游戏，在这款游戏里，我们要实现以下的一些功能：1.可视化，即图像（音符）的显示；2.声音的输出，并与图像建立一一对应的关系；3.对用户输入的按键进行识别，并通过相应的输入检测是否正确触发，最后反馈到得分上。
由于整个代码写得如同 shi 山一般（岳家庆言），并且借鉴 shell 语言的名字，我们最终将此游戏命名为 shill（即 s+hill）。

一. 选题及创意介绍：

音乐游戏是一种电子游戏，玩家在音乐游戏中听到对应的音符、看到相应的视觉提示时，应以正确的顺序输入对应的指令。由于玩家需要全神贯注的倾听、记忆和反应，因此在音乐游戏中，玩家的沉浸感通常较强。近年音游在年轻人群体中尤为流行，受到身边的人的启发，遂开发一款小的音游，希望给同学们带来不一样的音游体验。

二. 设计方案和硬件连接

硬件与电脑连接非常简单，只需要一根 micro USB 连接线即可传入程序，之后程序的运行可以完全独立进行。玩家的输入只需要 A、B 两个按键，方便玩家在双手之间享受轻便的游戏体验。
在下述的叙述中，由于我们不是很懂音乐，所以有些名词可能和乐理中的标准名称有出入，请各位看官见谅。

三. 实现方案及代码分析

```
# Imports go at the top
from microbit import *


# Code in a 'while True:' loop repeats forever
# 我们要实现以下的一些功能：
# 1.可视化，具体而言是图像的显示
# 2.声音的载入，并与图像建立一一对应的关系
# 3.对按键进行识别，并通过相应的输入检测是否正确触发
import music
```

以上部分为初始化内容。

```
15  Sheet_music = {"fstline":[('0',None),('0',None),('0',None),#1
16                            ('9','d4'),('9','e4'),('0',None),
17                            ('0',None),('0',None),('0',None),
18                            ('9','d5'),('7','c5'),('5','a4'),
19                            ('0',None),('0',None),('0',None),#5
20                            ('9','a4'),('8','g4'),('7','f4'),
21                            ('6','e4'),('0',None),('0',None),
22                            ('0',None),('7','g4'),('0',None),
23                            ('0',None),('0',None),('0',None),
24                            ('9','e4'),('9','d4'),('0',None),#10
25                            ('0',None),('9','e4'),('9','d4'),
26                            ('0',None),('0',None),
27
28                            ('5','d4'),('7','e4'),('9','f4'),#1
29                            ('0',None),('9','a4'),('0',None),
30                            ('9','d5'),('0',None),('5','a4'),
31                            ('0',None),('0',None),('0',None),
32                            ('9','a4'),('7','g4'),('5','f4'),#5
33                            ('0','e4'),('0',None),('0',None),
34                            ('0',None),('7','g4'),('9','a4'),
35                            ('0',None),('9','g4'),('0',None),
36                            ('9','e4'),('0',None),('9','f4'),
37                            ('0',None),('9','g4'),('0',None),#10
38                            ('9','a4'),('0',None),
39
40                            ('7','c5'),('9','d5'),('0',None),#1
41                            ('7','g4'),('0',None),('0',None),
42                            ('5','g4'),('0',None),('7','c5'),
43                            ('0',None),('7','a4'),('5','g4'),
44                            ('0',None),('0',None),('7','g4'),#5
45                            ('9','a4'),('0',None),('0',None),
46                            ('5','e4'),('0',None),('7','d4'),
47                            ('0',None),('0',None),('7','d4'),
48                            ('0',None),('7','f4'),('0',None),
49                            ('9','a4'),('7','d4'),('0',None),#10
50                            ('0',None),('9','c5'),
51
52                            ('7','c5'),('9','d5'),('0',None),#1
```

…

第 15-163 行为歌谱内容，歌谱为一字典，主要包含三个元素：

“fstline”：第一行音符的亮度、音高；

“scdline”：第二行音符的亮度、音高；

“music_length”：全谱所包含的 beat 数量。我们选择的歌，可能有细心的读者已经发现，是大家耳熟能详的 bad apple。针对这首歌的特点，我们定义一个八分音符为一个 beat；而对于四分音符的实现，则是在一个八分音符后面加入一个空的八分音符作为一个空的节拍以达到“延长音符”的听觉效果。

由于本游戏需要玩家左右手协同工作，所以有两行音符，这两行音符的规律是不同的，因此全谱可分为两半。

```
def doset():
    yourbpm = 240
    chances = 10
    display.scroll('GOTO MODULE A JUST START B',delay = 50)
    while True:
        if button_a.is_pressed():
            display.scroll('BASIC A SELF DEFINE B',delay = 75)
            while True:
                if button_a.is_pressed():
                    display.scroll('SLOW A MIDDLE B FAST AB',delay = 75)
                    while True:
                        if button_a.is_pressed() and button_b.is_pressed():
                            yourbpm = 240
                        elif button_a.is_pressed():
                            yourbpm = 60
                        elif button_b.is_pressed():
                            yourbpm = 120
                        break
                    break
                if button_b.is_pressed():
                    display.scroll('SET bpm ADD A DECLINE B SET NEXT AB',delay = 75)
                    while True:
                        if button_a.is_pressed() and button_b.is_pressed():
                            display.scroll('bpm %d'%yourbpm, delay =75)
                            display.scroll('SET CHANCES ADD A DECLINE B END AB',delay = 75)
                            while True:
                                if button_a.is_pressed() and button_b.is_pressed():
                                    display.scroll('CHANCES %d'%chances,delay = 75)
                                    break
                                elif button_a.is_pressed():
                                    chances += 1
                                elif button_b.is_pressed():
                                    if chances >= 2:
                                        chances -= 1
                            break
                        elif button_a.is_pressed():
                            if yourbpm <=290:
                                yourbpm = yourbpm +10
                        elif button_b.is_pressed():
                            if yourbpm >= 20:
                                yourbpm = yourbpm -10
                    break
            break
        if button_b.was_pressed():
            break
    return yourbpm ,chances
yourbpm , chances = doset()
```

以上为 doset（）函数，其目的是让玩家可以根据自身情况，改变曲子的每分钟拍数（yourbpm）和有多少条命（chances），进而调节游戏难度。

本函数建立了玩家与 microbit 互动的基本框架。在最开始，系统默认设置了两个值：

yourbpm = 240

chances = 10

其意义为歌曲默认每分钟的拍（beats）数量为 240 个；玩家默认有 10 条命。

之后程序给玩家两个选择，玩家按不同的 A 或 B 键有不同的效果：

1）若玩家此时按下 B，则函数直接返回默认的 yourbpm 值和 chances 值，玩家可以快速开始进行游戏；

2）若玩家此时按下 A，则跳转到设置页面，玩家可以自行调整游戏难度（yourbpm）：

a.)若玩家按下 A，则进入选择系统预设难度选择状态。此时玩家只按下 A 键可选择“SLOW”模式（yourbpm 为 60）；只按下 B 键可选择“MIDDLE”模式（yourbpm 为 120），同时按下 AB 两键可选择“FAST”模式（yourbpm 为 240）。

b.)若玩家按下 B，则进入自定义 yourbpm 以及 chances 状态。首先玩家设置的是

yourbpm，按下 A 可以增加 yourbpm，一次增 10，上限为 290；按下 B 可以降低 yourbpm，一次减 10，下限为 20。
在设置完 yourbpm 结束之后，玩家同时按下 AB 键可以继续前往设置 chances。屏幕上会展示玩家最终设置的 yourbpm 值，然后进入设置 chances 的模式。玩家按下 A 键可以增加 1 条命，无上限；玩家按下 B 键可以减少 1 条命，下限为 2。在设置结束后玩家同时按下 AB 两键可以保存最终设置并退出设置，屏幕上会展示玩家最终设置的 chances 值。

```
211   yourbpm , chances = doset()
```

之后是游戏的主题部分：

```
212   def ourgame():
213       gap = 0.01
214       global chances
215       music.set_tempo(bpm = yourbpm)
216       score = 0
217       isPass = False
218       music_length = len(Sheet_music["fstline"])
219       Sheet_music["music_length"] = music_length
220       tick_sequence = 0
221       tick = (60/yourbpm)-gap
```

此处定义了一些参数。
gap:判断识别部分所维持（sleep）的秒数；
score:玩家的得分数；
isPass:玩家是否已经打完。在之后的代码中会体现，每次循环都会将 tick_sequence 和 music_length 相比较，以判断玩家是否已经打完全部歌谱，以适时跳出循环给出游戏结果。
music_length:全歌所包含全部 beat 的总数，同时改写 Sheet_music（即歌谱）中的“music_length”值。
tick_sequence:当前所进行至第几个 beat
tick:图像与声音输出部分所维持的时间。为保持游戏的连续性，其在 yourbpm 已设定的情况下，与 gap 的长短有如上函数关系。
接下来是“判断识别部分”，即根据用户的输入判断是否需要扣除 chances 数：

```
while chances and(not isPass):
    times = 0
    button_a.get_presses()
    button_b.get_presses()
    ispressed = False
    while times<2:
        #判断识别部分
        if times==1 :
            ka = button_a.get_presses()
            kb = button_b.get_presses()
            if ka == 1 and Sheet_music["fstline"][tick_sequence][0]!='0':
                score = score + 1
                ispressed = True
            elif ka !=1 and Sheet_music["fstline"][tick_sequence][0]!='0':
                chances = chances - 1
            elif ka != 0 and Sheet_music["fstline"][tick_sequence][0]=='0':
                chances = chances-1
            if kb == 1 and Sheet_music["scdline"][tick_sequence][0]!='0':
                score = score + 1
                ispressed = True
            elif kb !=1 and Sheet_music["scdline"][tick_sequence][0]!='0':
                chances = chances - 1
            elif kb != 0 and Sheet_music["scdline"][tick_sequence][0]=='0':
                chances = chances-1
```

这是一个循环，在玩家命没耗尽且尚未通关时会循环执行。

首先定义一个 times 参数，这个参数开始时为 0，并且只能取 0 或 1。在 times 分别取 0、1 时，会使函数主体执行不同部分的代码。可以将这个参数理解成每个 beat 里边的 sub-beat，或者说，不同状态（对应的时间长短分别为 tick 与 gap）。

之后通过 button_x.get_presses()函数，对两个按键的计次数清零。

在 times==1 时，分别统计在之前 times==0 时 A、B 两按键被按下的次数。只有当 A、B 按键被按下的次数与歌谱相对应（即有音符时按 1 次，没音符时按 0 次），才不会扣除命数，并且正确地按下一个按键，分数 score 累加 1。

接下来是图像输出模块的代码。

```
247         #图像编辑部分
248         line1 = ''
249         for i in range(5):
250             line1 = line1 + Sheet_music["fstline"][tick_sequence+i][0]
251         line1 = line1 + ':'
252         if chances ==3:
253             line3 = '09990:'
254         elif chances == 2:
255             line3 = '09900:'
256         elif chances == 1:
257             line3 = '09000:'
258         else:
259             line3 = '99999:'
260 
261         if ispressed:
262             line2 = '99999:'
263             line4 = '99999:'
264         else:
265             line2 = '00000:'
266             line4 = '00000:'
267 
268         line5 = ''
269         for i in range(5):
270             line5 = line5 + Sheet_music["scdline"][tick_sequence+4-i][0]
271         line5 = line5 + ''
```

这段代码决定了每个 beat 里，屏幕上所显示的图样。其中第 1 和第 5 行为滚动状的音符流，第 3 行为生命数显示。当 chances 多于 3 时，5 个像素点亮满；当 chances 等于 3 时，只显示中间 3 个像素点；当 chances 为 2 或 1 时，显示对应数量的像素点数。

在一个 beat 里从 tick 时间开始直到结束的这段时间里，玩家正确地按下按键时，ispressed 值会改为 True，第 2、4 行会在接下来的 gap 时间里全亮起以表示玩家在上个 tick 时间里正确地做出了反应。ispressed 值在每个循环开始时会改写为 False。

接下来为图像与声音输出部分的代码，其为游戏最重要的显示与输出部分。

```
272         #图像与声音输出部分
273         current = line1 + line2 + line3 + line4 +line5
274         display.show(Image(current))
275         if times == 0:
276             first_symbol =  Sheet_music["fstline"][tick_sequence][1]
277             second_symbol = Sheet_music["scdline"][tick_sequence][1]
278             if (not first_symbol) and (not second_symbol) :
279                 sleep(tick*1000)
280             elif (first_symbol) and (not second_symbol):
281                 music.play([first_symbol+':4'])
282             elif (not first_symbol) and (second_symbol):
283                 music.play([second_symbol+':4'])
284             else:
285                 music.play([first_symbol+':4'])
286         elif times ==1:
287             sleep(gap*1000)
288         times = times+1
289     tick_sequence = tick_sequence + 1
290     if tick_sequence == music_length - 4:
291         isPass = True
292     ispressed = False
293 
```

在 times==0 时，执行图像与声音输出部分。屏幕上显示当前 beat 的 tick 部分所应该显示的图像。同时，根据当前所处第几个 beat 以及歌谱播放对应音高的声音（若没有对应音符，则休止，不播放声音）。tick 维持时间为 tick，单位秒。但是声音输出时间却是 tick+gap，即覆盖一整个 beat，以使得歌曲听起来更连续。
在 times==1 时，也执行图像输出，但是正如上文所提及，此时多出第 2、4 行对玩家上个 tick 时间里做出反应的反馈。维持时间为 gap，单位秒。
在玩家交互时，屏幕上第 1 行的音符流动到最左边的时候，玩家需要在 tick 时间内按下 A 按键（不按或只按 1 次）；在第 5 行音符流动到最右边的时候，类似地，玩家需要在 tick 时间内按下 B 按键（不按或只按 1 次）。之后在程序执行 gap 部分的时候，玩家按下是无效的，也就是说玩家反应太慢了。音游的难度通常与判定的严格程度紧密相关。为了尽量使大多数人都能获得不错的游戏体验，我们设置的判定时间长度（即 tick 长短）在 120bpm 的情况下为 (60÷120)-0.01=0.49s =490ms，是相对来说比较宽松的。

```
294     if isPass:
295         display.scroll('CONGRATULATIONS YOUR SCORE %d'%score,delay = 75)
296     else:
297         display.scroll('GAME OVER YOUR SCORE %d'%score,delay = 75)
```

最后是对于是否通关的判断，跳出循环后看 isPass 值是否为 True。若为 True，则是全部打完后正常退出循环，屏幕显示恭喜消息以及分数。若为 False，则是死出来的，玩家没能完成全部歌谱，显示游戏结束消息以及分数。
以上为 ourgame()函数的代码，也是程序最核心的部分。

```
298 ourgame()
```

执行游戏。

四. 后续工作展望

我们的游戏还有很多可以改进的地方：
其一，可以跳出一个四分音符为一个 beat 的限制，采用八分音符甚至十六分音符为一个基本游戏刻。这样可以引入节奏变化更丰富的歌曲。
其二，我们目前将每个 beat 分为两部分。我们可以在每个 beat 里细分出更多部分，增加判定次数，然后根据玩家按下按键的时间处于哪一部分（是更靠近中间，还是太快，或者太慢）以给出数值不同的加分。

五. 小组分工合作

岳家庆：游戏主体编写
王世宁：歌谱编写，后期