2023 春季学期数据结构与算法 micro bit C1 作业

——micro 二指音游

2100011458 物理学院 杨乐言
2000013387 工学院 田静
（音乐指导：22 信科郑世淇）

摘要：

在 micro bit 上实现了 A、B 两个按键控制的二指音游。在所选音乐（千本樱）播放的同时，随着音乐节奏，开发板显示出下落的光条。随着节奏按键，最终记录并显示出按键成功次数（perfect）及失败次数（miss）。

一、选题及创意介绍

参考往届优秀作品，决定融入音乐与二次元元素。为保留 micro bit 的简洁属性，不打算引入更多的连线按键接头。参考经典二键音游，偶像梦幻祭一代音游与其他纵向下落音游。音乐的选取综合了受众人数广、音乐节奏感强、节奏较快、一定的二次元气息等多个要求，选曲致敬了《千本樱》。最终形成了这样一款简洁的音乐小游戏。

二、设计方案和硬件连接

设计方案：

以音高+音长形式记录简谱，根据音乐节奏编写谱面。在播放音乐的同时，在 micro bit 显示屏上显示谱面，即相应的下落光点。设计了三种光点形式，最下面一排光点。左边两个亮起对应按下 A 键，右边两个亮起对应按下 B 键，一排五个光点全部亮起对应同时按下 A、B 键。在歌曲的播放过程中，记录操作者按键成功的次数。并在全曲结束后通过显示屏滚动显示记录结果。

硬件连接：

通过 micro - usb 线将电脑与 micro bit 开发板连接。

三、实现方案及代码分析

1、初始化。

初始 miss = 0，perfect = 0，音乐节奏 bpm = 300，音调 330。

```
miss =0
perfect = 0
music.set_tempo(bpm=300)
music.pitch(330)
```

2、输入谱面。

Plane 实现了光点下落的动态表现，两个最小节拍，进行一次图案变化。每一横行的 5*5 矩阵显示了 micro bit 显示板的一次发光情况，9 代表光强最强，0 代表不发光。通过铺面数字的循环出现实现连贯的图案滚动显示。

Key 值实现了后续的按键判定，[1，0]代表左侧两个灯亮，即应该按 A 键。[0，1]代表右侧两个灯亮，即应该按 B 键。[1，1]代表五个灯全亮，应该同时按 A、B 键。[0，0]代表无需按键，游戏已经结束。

```
plane=['99000:''99999:''99000:''00099:''99000:',
       '99000:''99999:''99000:''00099:''99000:',
       '99000:''99000:''99999:''99000:''00099:',
       '99000:''99000:''99999:''99000:''00099:',
       '00099:''99000:''99000:''99999:''99000:',
       '00099:''99000:''99000:''99999:''99000:',
       '99000:''00099:''99000:''99000:''99999:',
       '99000:''00099:''99000:''99000:''99999:',
       '99999:''99000:''00099:''99000:''99000:',
```

```
key=[
        [1,0],
        [1,0],
        [0,1],
        [0,1],
        [1,0],
        [1,0],
        [1,1],
        [1,1],
        [1,0],
        [1,0],
        [1,0],
        [1,0],
```

3、记录 perfect、miss 数。

```
for i in range(len(note)):
    display.show(Image(plane[i]))
    music.play(note[i])
    sleep(100)
    A = button_a.get_presses()
    B = button_b.get_presses()
    if key[i] == [1,0]:
        if A ==1 and B == 0:
            perfect += 1
        else:
            miss += 1
    if key[i] == [0,1]:
        if B == 1 and A == 0 :
            perfect += 1
        else:
            miss += 1
    if key[i] == [1,1]:
        if A == 1 and B == 1:
            perfect += 1
        else:
            miss += 1
    if key[i] == [0,0]:
        continue
sleep(3000)
```

4、画面优化展示。

```
display.show('sakura')
```

```
display.show('perfect')
display.show(perfect)

sleep(5000)

display.show('miss')
display.show(miss)
```

四、后续工作展望

1、修正按键判定时效，现有的按键判定是：音乐声响发出后，0.5s 内成功按键即 perfect + 1。 但是据音游体验而言，有先预按键这一事件存在，即在音乐声音发出之前，就已经按下键。因此后续可能修正代码使得，在音乐发出前后 250ms 内按键，均能被判定为 perfect。

2、增设 combo 记录，即连续点击中的次数。

3、优化谱面，使得谱面更加贴合音乐，使得谱面针对每首歌的特异性更高。比如随着音乐节奏更改光点下落速度。

4、调整按键判定方式，现有的判定方式是网站界面提示的基本方式，即停顿 500ms，记录停顿时间内的按键次数，但是这种停顿影响了音乐节奏，后续探索新的记录按键方法。

5、为不同音符的组合预设谱面，实现输入任意的音乐简谱，都能生成相应的音乐谱面。

五、小组分工合作

杨乐言：思路提供、代码实现、谱面设计、提供后续改进思路。

田静：实习报告、作品 poster、运行及介绍视频、微调代码。

郑世淇（未出现在选课名单里的音乐指导）：分析乐谱，并提供了将乐谱转化为乐名的 c++代码，并批判了现存的节奏问题（四、4、）。