**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）。

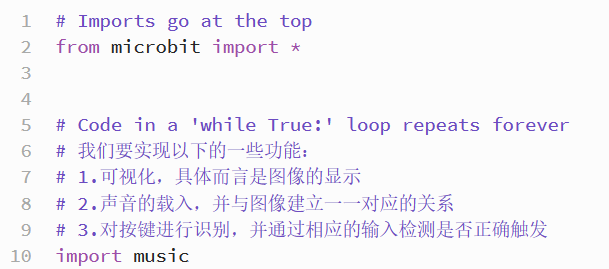
**一. 选题及创意介绍：**

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。

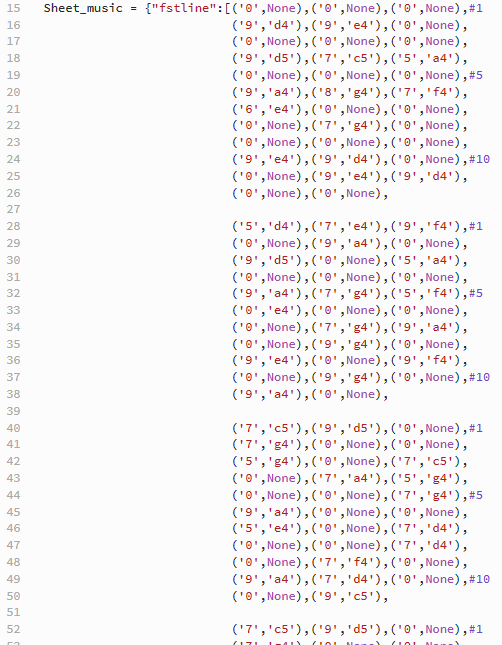
**二.设计方案和硬件连接**

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**

以上部分为初始化内容。



…

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。



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）：

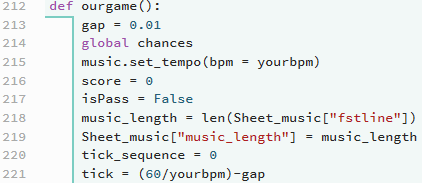
a.)若玩家按下A，则进入选择系统预设难度选择状态。此时玩家只按下A键可选择 “SLOW”模式（yourbpm为60）；只按下B键可选择“MIDDLE”模式（yourbpm为120），同时按下AB两键可选择“FAST”模式（yourbpm为240）。

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。



之后是游戏的主题部分：



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；

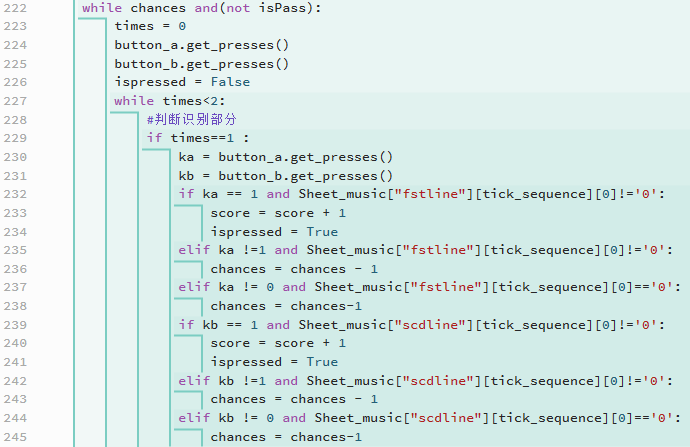
isPass:玩家是否已经打完。在之后的代码中会体现，每次循环都会将tick\_sequence和music\_length相比较，以判断玩家是否已经打完全部歌谱，以适时跳出循环给出游戏结果。

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：



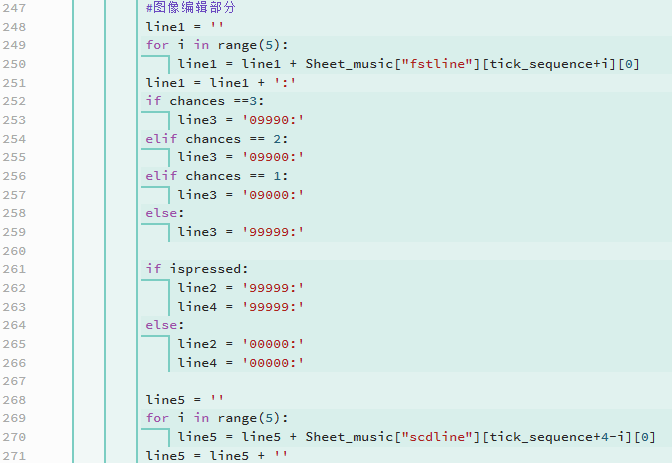
这是一个循环，在玩家命没耗尽且尚未通关时会循环执行。

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。

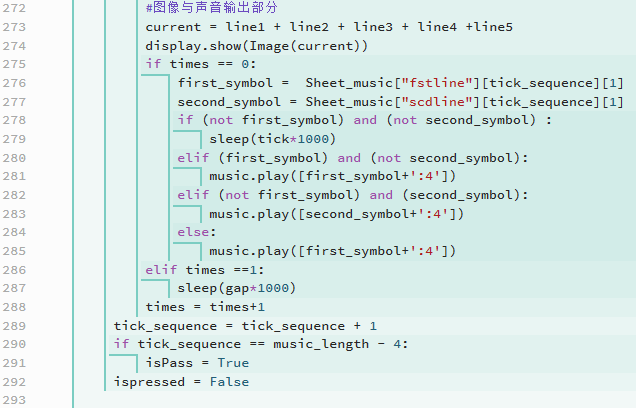
接下来是图像输出模块的代码。



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。

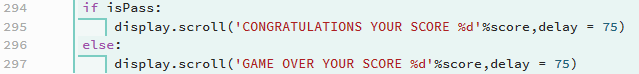
接下来为图像与声音输出部分的代码，其为游戏最重要的显示与输出部分。



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。



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。



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