实习报告：简易音游模拟器在 micro:bit 上的实现

作者：胡金博 徐旨健

摘要：本文介绍了一种基于 micro:bit 的简易音游模拟器的设计和实现方法。该模拟器可以模拟左右按键的提示和按键的响应，并实现了计分和胜负判断功能。通过该项目，可以帮助初学者更好地理解和掌握 micro:bit 的硬件和编程知识，提高其对嵌入式系统和计算机科学的理解。

一、选题及创意介绍

音游是一种以音乐为基础的休闲游戏，其玩家需要按照屏幕上的提示按下相应的按键，来配合音乐节奏和音符的掉落。随着电子音乐和娱乐行业的发展，音游成为了一种受欢迎的休闲娱乐方式，其不仅能够带来乐趣，还可以提高玩家的反应速度和音乐素养。

在本项目中，我们基于 micro:bit 这一简单易用的嵌入式系统，实现了一个简易音游模拟器。该模拟器可以模拟左右按键的提示和按键的响应，并实现了计分和胜负判断功能。通过该项目，可以帮助初学者更好地理解和掌握 micro:bit 的硬件和编程知识，提高其对嵌入式系统和计算机科学的理解。

二、设计方案和硬件连接

在本项目中，我们需要利用 micro:bit 的 LED 点阵、按键和蜂鸣器等硬件资源来实现音游模拟器的功能。

具体的硬件连接方案如下：

将 micro:bit 与计算机通过 USB 线连接，进行编程和调试。

将 micro:bit 的 GND 引脚分别连接到两个面包板的公共地线上。

将 micro:bit 的引脚 P0、P1、P2、P8 和 P12 分别连接到面包板上的 LED 点阵模块上。

将 micro:bit 的引脚 P5 和 P11 分别连接到面包板上的按钮模块上。

将 micro:bit 的引脚 P0 连接到面包板上的蜂鸣器模块上。

三、实现方案及代码分析

该音游模拟器的实现思路主要是使用 microbit 的显示屏和按键控制模块来模拟一个简单的音乐游戏。游戏中有两个按键 A 和 B，屏幕中会出现左右两侧的按键提示，玩家需要在一定的时间内按下正确的按键，如果按错或者超时则游戏结束，否则得到一分。

具体实现中，使用了 random 模块来随机生成左右两侧按键提示，并且为了增加游戏的难度，还随机选择左右两侧的提示中是否还要增加一些额外的按键。游戏中采用了二进制的方式展示玩家的分数。

代码中首先定义了两个函数 show_score 和 random_show_key。

show_score 函数的作用是根据玩家当前的得分，在 micro：bit 的显示屏上用二进制的方式显示分数。该函数会先清空屏幕，然后用循环实现二进制数码管的显示，每次循环将余数取出来，根据余数是否为 1，在对应的位置上亮灯。最后整除当前数并且更新计数器，直到整除结果为 0。

random_show_key 函数的作用是根据玩家当前的难度级别，随机生成左右两侧的按键提示，并且给玩家一定的时间内来按下正确的按键。在函数中会先生成一个随机数来决定左右两侧哪个要显示，然后再随机生成左右两侧的按键提示。在显示完按键提示后，函数会启动一个循环，计算出给玩家按键的时间，然后等

待按键。如果玩家在规定时间内按下了正确的按键，则返回 True，否则返回 False。最后，在主函数中采用一个 while 循环不断调用 random_show_key 函数来进行游戏的模拟，并且根据返回值更新玩家的分数，直到分数到达 100 或者玩家按错了按键。在游戏结束后，根据玩家是否失败来在屏幕上显示相应的提示信息。

代码分析部分：

该音游模拟器的代码主要包括两个函数和一个主程序的实现。下面我们逐一对它们进行分析。

（1）show_score 函数：

```
def show_score(score):
    display.clear()  # 清屏
    x = score        # 分数
    i = 0            # 计数器
    while x != 0:    # 当有 x 可以二分的时候，采用二进制的显示法
        t = x % 2    # 取余
        if t == 1:  # 如果有余，对应的位置亮灯
            display.set_pixel(i % 5, i // 5, 9)
        x = x // 2  # 整除
        i = i + 1
    return i
```

show_score 函数用于在 micro:bit 上展示分数。分数在函数的参数中传入，函数会将分数转化为二进制表示，然后在 micro:bit 的 LED 点阵屏幕上用点阵表示出来。具体分析如下：

首先，函数调用了 display.clear() 方法，清空 micro:bit 屏幕上所有的点阵。接着，分数值被赋值给变量 x，计数器 i 被初始化为 0。

接下来，使用 while 循环将分数值 x 转化为二进制并显示在 LED 点阵屏幕上。当 x 不为 0 时，通过 % 操作符取 x 除以 2 的余数并赋值给 t，判断 t 是否为 1，如果是则在屏幕的第 i % 5 列和第 i // 5 行点亮一个像素，表示一个二进制位为 1。然后将 x 除以 2 并赋值给 x，并将 i 加 1。

最后返回计数器 i 的值。

该函数的作用是将分数转化为二进制表示，并在 LED 点阵屏幕上用点阵表示出来，方便玩家观察自己的得分。

（2）random_show_key 函数：

```
def random_show_key(i):#展示和按键模块
    Flag=True#表示有没有按错键
    k=random.randint(0,1)#随机模块，决定左右呈现
    if k==0:#左呈现
        t=random.randint(0,1)#随机择数，不同呈现
        if t==0:
            display.set_pixel(0,3,9)#呈现按键提示
            display.set_pixel(1,3,9)
            time=0
            while True and time <1000/i+100:#计时和按键
                Flag=False#如果超过按键时间，则表示错误
                time+=1#循环计时模块
                sleep(1)#延迟表示时间
                if button_a.is_pressed():#如果按右键
                    display.clear()
                    music.pitch(262)#放音乐
                    sleep(100)
                    music.stop()
                    Flag=True
                    break
                elif button_b.is_pressed():#错误按键
                    display.clear()
                    break
            return Flag
        if t==1:
            display.set_pixel(0,3,9)
            display.set_pixel(1,3,9)
            display.set_pixel(0,4,9)
            display.set_pixel(1,4,9)
            time=0
            while True and time <2000/i+100:
                Flag=False
                time+=1
                sleep(1)
                if button_a.is_pressed():
                    display.clear()
                    music.pitch(330)
                    sleep(100)
                    music.stop()
                    Flag=True
                    break
                elif button_b.is_pressed():
                    display.clear()
                    break
            return Flag
```

```
if k==1:
    t=random.randint(0,1)
    if t==0:
        display.set_pixel(3,3,9)
        display.set_pixel(4,3,9)
        time=0
        while True and time <1000/i+100:
            Flag=False
            time+=1
            sleep(1)
            if button_b.is_pressed():
                display.clear()
                music.pitch(294)
                sleep(100)
                music.stop()
                Flag=True
                break
            elif button_a.is_pressed():
                display.clear()
                break
        return Flag
    if t==1:
        display.set_pixel(3,3,9)
        display.set_pixel(4,3,9)
        display.set_pixel(3,4,9)
        display.set_pixel(4,4,9)
        time=0
        while True and time <2000/i+100:
            Flag=False
            time+=1
            sleep(1)
            if button_b.is_pressed():
                display.clear()
                music.pitch(349)
                sleep(100)
                music.stop()
                Flag=True
                break
            elif button_a.is_pressed():
                display.clear()
                break
        return Flag
```

该函数用于展示和按键，并返回是否按对的结果。其基本思路是利用随机数来决定按键的呈现方式和位置，并在一定时间内判断是否按对了键。具体实现方法如下：

定义 Flag 变量表示是否按对了键，k 变量随机决定左右呈现，t 变量随机决定按键类型。

如果 k 等于 0，则进行左呈现，具体实现方法如下：

如果 t 等于 0，则在第 0 列第 3 行和第 1 列第 3 行上亮灯，即 display.set_pixel(0,3,9)和 display.set_pixel(1,3,9)。

如果 t 等于 1，则在第 0 列第 3 行、第 1 列第 3 行、第 0 列第 4 行和第 1 列第 4 行上亮灯，即 display.set_pixel(0,3,9)、display.set_pixel(1,3,9)、display.set_pixel(0,4,9)和 display.set_pixel(1,4,9)。

进入按键循环，设置计时器 time 为 0，当 time 小于一定时间（1000/i+100 或 2000/i+100）且未按错键时，进行如下操作：

如果按下了右键（button_a.is_pressed()为真），则在 LED 矩阵上播放声音并设置 Flag 为 True，即 display.clear()、music.pitch(262)、sleep(100)和 music.stop()、Flag = True、break。

如果按下了错误键（button_b.is_pressed()为真），则直接跳出循环，即 display.clear()、break。否则 time 加 1 并进行延迟，即 time += 1、sleep(1)。

如果 k 等于 1，则进行右呈现，具体实现方法同上，只是按键位置和类型不同。

最后返回 Flag 的值。

三、主程序

主程序主要通过调用上述两个函数来实现游戏的流程。具体实现方法如下

```
score=0
i=0
contin=True
while score<100 and contin:
    contin=random_show_key(i+1)
    score=score+1
    i=show_score(score)
if contin:
    display.show('You Win!')
else:
    display.show('You Failed!')
display.clear()
```

在这段代码中，首先定义了三个变量 score、i 和 contin，它们分别表示得分、速度和游戏是否继续。

然后使用 while 循环控制游戏进行，循环的条件是得分小于 100 并且游戏继续。每次循环时调用 random_show_key 函数随机展示按键，通过调用 show_score 函数显示当前得分，并将得分和速度分别加 1。

当游戏结束时，根据 contin 的值来判断游戏是否胜利。如果 contin 为真，则显示 You Win!，否则显示 You Failed!。最后清空屏幕。

这段代码的逻辑比较简单，主要是通过循环控制游戏进行，同时根据用户的操作更新得分和速度。

四、后续工作展望

本项目是一个简易的音游模拟器，但是具有很好的拓展性和可扩展性。在后续的工作中，可以尝试添加更多的游戏模式和关卡，以增加游戏的可玩性和趣味性。

此外，目前项目中的游戏模式仅包括单人模式，可以尝试添加多人模式，让多个玩家之间相互竞争，增加游戏的乐趣和互动性。

此外，我们还可以考虑将该模拟器与其他的硬件设备进行连接，比如音响或者震动器等，以增强游戏的体验感和真实感。

在代码实现方面，可以对代码进行优化，改进游戏逻辑，增加游戏的难度和挑战性，提高游戏的可玩性。

五、小组分工合作

在本项目中，小组成员共同完成了以下任务：

选题及创意介绍：全体成员共同讨论，提出建议，确定选题；

设计方案和硬件连接：负责人徐旨健完成；

实现方案及代码分析：全体成员共同完成，分别负责不同的代码部分：胡金博负责组块 1：进度显示（使用最上面两行作为亮灯显示，以 2 进制作为亮灯进度）和组块 2：按键提示（使用左下 2*2 和右下 2*2 的显示区域，随机出现左侧

或者右侧，当然也可以用连续的预设，使用 2 进制缩短每次亮灯持续时间）；徐旨健负责组块 3：按键反馈（判断游玩者是否在规定时间按键，判断游玩者是否按下正确方向的键，基于按键左侧给出 1、3、5 其中一个音，右侧给出 2、4 其中一个音）。

后续工作展望：全体成员共同讨论，提出建议，确定后续工作计划。小组成员之间相互协作，及时沟通，共同解决遇到的问题，保证了项目的顺利进行。

在实习过程中，我们不断探索和学习新知识，提高了自己的编程能力和团队协作能力。通过该项目的实践，我们深刻认识到了团队合作的重要性和意义，更加坚定了我们在编程领域不断学习和成长的决心和信心。