璃奈板，但是俄罗斯方块

作者：王翊霏，李欣宸

摘要：本作品是两个反差很大（~~毫不相关~~）的创意的结合。第一部分是运用 microbit 的 5*5 点阵进行抽象的表情随机显示及音乐播放（毕竟只有 25 个格点）。第二部分是一个经典小游戏俄罗斯方块的简化+改编版（毕竟只有 25 个格点），游戏结束后配有创意彩蛋。当玩家成功填满一定数量的格子时，会触发动画和语音功能。当然有某个数量的格子和其他数量的格子不太一样，它与 pi 相乘并取整后恰好是 50！然后会播放大家熟悉的音乐和话语！（猜一猜）下面会详细介绍创意、实现方案等。

一、选题及创意介绍

由于我们小组选择无实物创作，因此想把单片机的功能运用的全面一些。第一位同学主要运用图像和音乐，做了一个简单版“璃奈板”用于随机显示及切换不同表情（用点来表示）。

第二位同学主要运用按钮，并做了她最爱玩的俄罗斯方块的改编版。最初的想法是复现俄罗斯方块，但由于板的空间比较小，消除与旋转实现起来较为复杂，她最终选择了减少每个俄罗斯方块的组成块的个数，并不允许旋转与消除，游戏结束条件即为不能再放置俄罗斯方块。

她和第一位同学在游戏结束后进行了再创意。通过结算已经填充的方格的个数 count 结算 money，money=[count*pi]，然后显示图案，播放语音“crazy Thursday V me {money}”。细心的同学可能发现，当 count==16 时，money=50，此情形当然是不平凡的，因此我们要播放 98k 音乐使你如同置身 KFC，并自信的说出“crazy Thursday V me fifty”。

二、设计方案和硬件连接

第一个创意的设计方案：预先写入不同点阵表示的表情，运用 random 进行随机抽取表情播放，同样运用 random 确定每个表情显示的时长。关于音乐播放部分，用列表写入一定格式表示的音符即可。

第二个创意的设计方案：通过改变坐标来控制俄罗斯方块的左移、右移、下移，用嵌套列表记录每个点是否被点亮，在进行移动操作时先检验该移动是否合法。初始时每个方块从顶部落下，当方块不能再下落时，更改记录每个点是否被点亮的嵌套列表，并点亮该方块的最终位置。在下落的过程中，监测 A、B 两个按钮是否被按下，如果 A 被按下且不发生冲突，方块向左移动，否责方块不移动。B 按钮同理可知。如果方块在初始位置刚一出现就发生冲突，则游戏结束，通过记录每个点是否被点亮的嵌套列表结算被点亮的点的总个数，并计算出总钱数，如果总钱数恰好等于 50，滚动“KFC!orz”并播放音乐和语音。否则仅播放语音。

一些设计上的细节：考虑到仅有 25 个方格，俄罗斯方块的组成块数变为 1/2/3，且 1/2 块的俄罗斯方块生成概率较大。为使得不同块在下落摆放好后能被区分开来，在初始生成该块时，就随机选择一个亮度。在结算 money 的时候，由于 money 是一个数字，直接读会变成读电话号码，因此维护一个字典，里面是所有可能出现的数字和英文的对应，例如 50：fifty。

三、实现方案和代码分析

璃奈板部分：

首先将表情表示出来：

```
Image1 = Image("00000:"
               "09090:"
               "00000:"
               "90009:"
               "09990")
ImageX = Image("00000:"
               "09090:"
               "00000:"
               "09990:"
               "90009")
Image2 = Image("09090:"
               "00000:"
               "00900:"
               "09090:"
               "00900")
Image3 = Image("00000:"
               "99099:"
               "00000:"
               "90009:"
               "09990")
```

随机播放

```
#通过随机数控制随机表情生成及时
while True:
    i=random.randint(1,6)
    j=random.random()
    if i==1:
        display.show(Image1)
        time.sleep(j)
        display.show(Image7)
        time.sleep(j)
    if i==2:
        display.show(Image2)
        time.sleep(j)
    if i==3:
        display.show(Image3)
        time.sleep(j)
    if i==4:
        display.show(Image4)
        time.sleep(j)
```

写入并播放歌曲：

```
#歌曲部分
tunes=["A4:2","B4:2","C5:4","D5:2",'B4:4','A4:2','G4:8','R:4',
       'A4:2','B4:2','C5:2','A4:4','G4:2','G5:4','G5:2','D5:4','E5:2','D5:2','C5:2','R:4',
       'A4:2','B4:2','C5:2','A4:2','C5:2','D5:4','B4:2','A4:2','G4:6','R:6',
       'A4:2','B4:2','C5:2','A4:4','G4:4','D5:2','E5:2','D5:6','R:4',
       'C5:10','D5:2','E5:2','C5:2','D5:2','D5:2','D5:2','E5:2','D5:2','G4:4','R:4',
       'A4:2','B4:2','C5:2','A4:2','G4:1','R:1','D5:1','E5:2','D5:6','R:2'
       "G4:1","A4:1","C5:1","A4:1","E5:3","E5:3","D5:6",
       "G4:1","A4:1","C5:1","A4:1","D5:3","D5:3","C5:1","B4:1","A4:4",
       "G4:1","A4:1","C5:1","A4:1","C5:4","D5:2","B4:2","A4:2","G4:2","R:2","G4:2","D5:4","C5:4","R:4"
       "G4:1","A4:1","C5:1","A4:1","E5:3","E5:3","D5:6",
       "G4:1","A4:1","C5:1","A4:1","G5:4","B4:2","D5:6",
       "G4:1","A4:1","C5:1","A4:1","C5:4","D5:2","B4:2","A4:2","G4:2","R:2","G4:2","D5:4","C5:4","R:4"]
music.play(tunes)
```

俄罗斯方块部分：
首先初始化一个二维列表 options，每个元素是一个列表，存放俄罗斯方块每个小方块初始位置的坐标。再初始化一个二维列表 record，记录每个点是否亮起。

```
# 六种备选俄罗斯方块
opptions = [[(2, 0)], [(2, 0), (3, 0)], [(2, 1), (3, 0)], [(2, 0), (3, 1)],
            [(2, 0), (2, 1), (3, 1)], [(3, 0), (2, 1), (3, 1)]]

# 开一个5*5表格记录每个点是否亮起
record = [[0, 0, 0, 0, 0], [0, 0, 0, 0, 0], [0, 0, 0, 0, 0],
          [0, 0, 0, 0, 0], [0, 0, 0, 0, 0]]
```

下面定义判断是否冲突的函数，参数为两个列表，一个为记录当前俄罗斯方块坐标的列表 lst，另一个为记录每个点是否亮起的列表 rec（即为 record），对于第一个列表中的每个元素，即每个点的坐标，在 rec 中检索其值，判断是否已经被点亮。

```
# 判断移动操作是否合法
def no_conflict(lst, rec):
    for k in lst:
        i, j = k
        if rec[i][j] != 0:
            return False  # 冲突
    return True  # 不冲突
```

下面定义俄罗斯方块移动的函数，以下移为例。参数为记录当前俄罗斯方块坐标的列表 lst，首先计算出所有坐标中的最大行，如果最大行不是最后一行，则将所有点纵坐标+1，否则不操作。

```
# 俄罗斯方块下移
def down(lst):
    max_line = max([k[1] for k in lst])
    if max_line < 4:
        lst = [(k[0], k[1] + 1) for k in lst]
    return lst


# 俄罗斯方块左移
def left(lst):
    min_col = min([k[0] for k in lst])
    if min_col > 0:
        lst = [(k[0] - 1, k[1]) for k in lst]
    return lst


# 俄罗斯方块右移
def right(lst):
    max_col = max([k[0] for k in lst])
    if max_col < 4:
        lst = [(k[0] + 1, k[1]) for k in lst]
    return lst
```

下面是把俄罗斯方块显示出来的函数，参数为记录当前俄罗斯方块坐标的列表 lst 和亮度 bri。

```
# 把一个俄罗斯方块显示出来
def show(lst, bri):
    for k in lst:
        i, j = k
        display.set_pixel(i, j, bri)
```

下面为更改 record 的函数，一个参数是记录当前俄罗斯方块坐标的列表 lst，另一个为记录每个点是否亮起的列表 rec。

```
# 俄罗斯方块不能再下落时更改record
def modify(lst, rec):
    for k in lst:
        i, j = k
        rec[i][j] = 1
```

下面是实现的主体部分。随机生成一个俄罗斯方块和其颜色之后，先把它显示出来，如果不发生冲突，在俄罗斯方块每次下移操作之前，对 A/B 是否被按下进行五次检测，如果 A 被按下则执行 left 函数，如果 B 被按下则执行 right 函数，然后进行下移操作。如果发生冲突，则更改 record，退出下移的循环。此时外层循环会产生新的俄罗斯方块，继续进行上述循环。若新的俄罗斯方块一经产生就发生了冲突，则游戏结束。将已经被点亮的点全部变成最亮（通过遍历 record）实现。

```
while True:
    # 1~4种生成概率较大，5~6种生成概率较小
    number = random.randint(1, 3)
    if number == 1:
        range = 6
    else:
        range = 4
    opption = opptions[0: range]
    tetris = random.choice(opption)
    color = random.randint(5, 9)
    show(tetris, color)
    sleep(1000)

    if not no_conflict(tetris, record):  # 不能再放置俄罗斯方块，游戏结束
        display.clear()
        display.scroll('Game Over', 100)
        fill = 0
        for times in [0, 0, 0]:  # 闪烁三次
            display.clear()
            sleep(500)
            for i in [0, 1, 2, 3, 4]:
                for j in [0, 1, 2, 3, 4]:
                    if record[i][j] != 0:
                        fill += 1
                        display.set_pixel(i, j, 9)
            sleep(500)
```

```
else:
    while True:
        times = 5
        while times > 0:
            if button_a.is_pressed() and no_conflict(left(tetris), record):
                show(tetris, 0)
                tetris = left(tetris)
                show(tetris, color)
            if button_b.is_pressed() and no_conflict(right(tetris), record):
                show(tetris, 0)
                tetris = right(tetris)
                show(tetris, color)
            times -= 1
            sleep(30)
        if no_conflict(down(tetris), record) and down(tetris) != tetris:  # 俄罗斯方块下移
            show(tetris, 0)
            tetris = down(tetris)
            show(tetris, color)
            sleep(1000)
        else:
            modify(tetris, record)  # 修改record，退出循环
            break
```

此时代码的主体部分就完成了。下面介绍一些创意图案的设计。下面是 5*5 方块蛇形遍历的坐标构成的列表。首先将所有点设为较低亮度。然后遍历 image，维护一个栈记录要点亮的几个点的坐标（两个，边界情况只有一个）。栈作为 light_up 函数的参数，可以实现点亮 0.1s 在恢复原来亮度的功能。最终的视觉效果就是两个亮点在沿蛇形跑动。然后展示一些图案，和一个 logo。Logo 的设计理念是左上角一颗星星，右下角两个经典俄罗斯方块。

```
# 点亮顺序的坐标
image = [(0, 0), (1, 0), (0, 1), (0, 2), (1, 1), (2, 0), (3, 0), (2, 1), (1, 2), (0, 3), (0, 4),
         (1, 3), (2, 2), (3, 1), (4, 0), (4, 1), (3, 2), (2, 3), (1, 4), (2, 4), (3, 3), (4, 2),
         (4, 3), (3, 4), (4, 4)]
```

```
# 点亮某两个点
def light_up(lst):
    for k in lst:
        x, y = k
        display.set_pixel(x, y, 9)
    sleep(100)
    for k in lst:
        x, y = k
        display.set_pixel(x, y, 4)
```

```
    display.show(Image('44444:'
                       '44444:'
                       '44444:'
                       '44444:'
                       '44444'))
    stack = []
    for k in image:
        stack.append(k)
        if len(stack) == 3:
            stack.pop(0)
        light_up(stack)

    for times in [0, 0, 0]:
        display.show(Image('00000:'
                           '00000:'
                           '00900:'
                           '00000:'
                           '00000'))
        sleep(200)
        display.show(Image('00000:'
                           '09990:'
                           '09990:'
                           '09990:'
                           '00000'))
        sleep(200)
        display.show(Image('99999:'
                           '99999:'
                           '99999:'
                           '99999:'
                           '99999'))
        sleep(200)

    display.show(Image('06000:'
                       '69606:'
                       '06066:'
                       '00906:'
                       '00999'))
```

下面一部分是结算 money 的代码，并显示 logo，播放音乐和语音。

```
    money = int(fill // 3 * pi)
    if money == 50:
        display.clear()
        display.scroll('KFC! orz', 100)
        display.show(Image('04000:'
                           '49407:'
                           '04077:'
                           '00807:'
                           '00888'))
        music.set_tempo(bpm=280)
        music.play(['eb4:8', 'c4:4', 'c4:4', 'c4:8', 'c4:4', 'c4:4',
                    'c4:8', 'c4:8', 'c4:4', 'bb3:4', 'c4:4', 'd4:4'])
        music.play(['eb4:8', 'c4:4', 'c4:4', 'c4:8', 'c4:4', 'c4:4',
                    'bb3:4', 'c4:4', 'c4:8', 'g4:4', 'f4:4', 'eb4:4', 'f4:4'])
        speech.say('turn it up')
        sleep(400)
    speech.say('crazy Thursday v me %s.' % dic[money])
    break
```

最后的代码是之前提到的一个数字转化为英文的细节。

```
# 数字-->英文
dic = {3: 'three', 6: 'six', 9: 'nine', 12: 'twelve', 15: 'fifteen', 18: 'eighteen', 21: 'twenty-one'
       25: 'twenty-five', 28: 'twenty-eight', 31: 'thirty-one', 34: 'thirty-four', 37: 'thirty-seven'
       40: 'forty', 43: 'forty-three', 47: 'forty-seven', 50: 'fifty', 53: 'fifty-three',
       56: 'fifty-six', 59: 'fifty-nine', 62: 'sixty-two', 65: 'sixty-five', 69: 'sixty-nine',
       72: 'seventy-two', 75: 'seventy-five', 78: 'seventy-eight'}
```

四、后续工作展望

璃奈板部分：5*5 真的限制发挥（咆哮 ing）。其他方面优化可以从表情切换更和音乐贴合入手。

俄罗斯方块部分：游戏还有一定的优化空间，比如俄罗斯方块的旋转功能，游戏分成几个不同难度模式，通过下落速度来区分，游戏进行过程中还可以逐渐加快下落速度，对于 A/B 建是否被按下的检测也可以进行优化，增强灵敏性，以及可以增加一个按钮，直接下落到底部。最后还可以尝试实现俄罗斯方块的消除功能。

五、小组分工合作

璃奈板部分：李欣宸

俄罗斯方块部分：王翊霏

视频剪辑、海报制作：李欣宸

实习报告：王翊霏、李欣宸