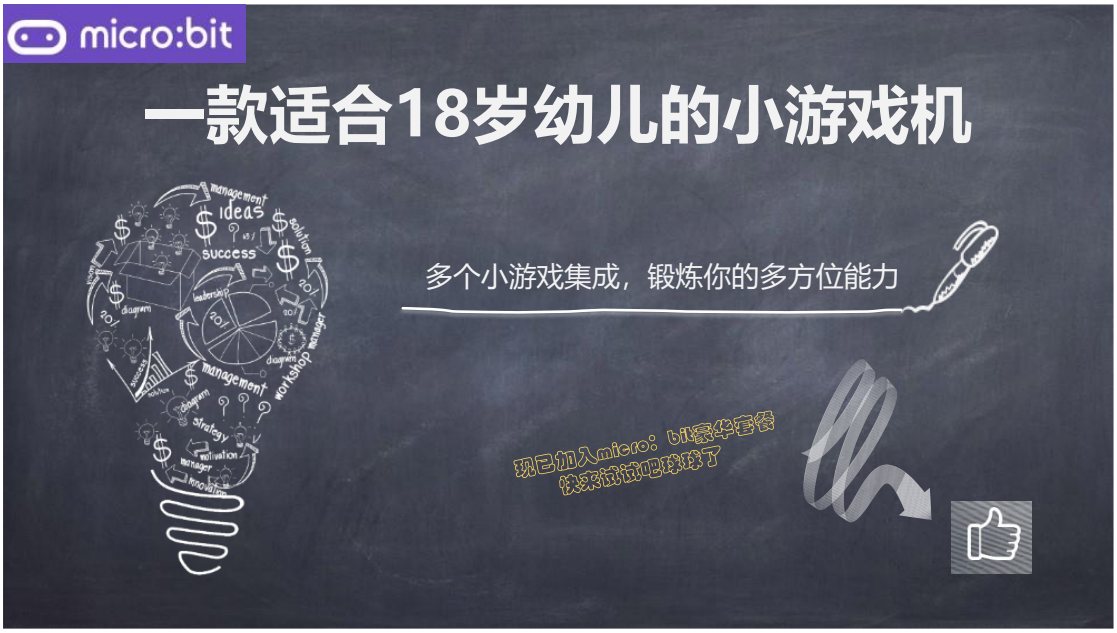
北京大学

开源硬件创意作品实习报告

题目：　一款适合 18 岁幼儿的小游戏机



课　　程：	数据结构与算法 B
教　　师：	陈斌
作　　者：	周佳欣　赵子豪

二〇二三年　五月

一款适合 18 岁幼儿的小游戏机

摘要：这是一款建立在 micro:bit 基础上的能力测试小游戏集成器，由多个小游戏组成，能够锻炼使用者小朋友们的空间感知能力、手眼协调能力、记忆思维能力、逻辑推理能力、策略制定能力等等等等，不过在实际中我们发现，针对小朋友的测试往往更能引起大学生的兴趣。

一、选题及创意介绍

考虑到 micro:bit 本身条件（尤其是 5*5 的点阵显示），所能显示的信息相对有限而简单，于是我们设计了多个小游戏，并尝试尽量能够让不同的小游戏来测试和锻炼用户的不同能力，从而集成为一款简易的小游戏机。为了增加趣味性，还在每个小游戏中添加了 Best Score 这一计分机制，让我们坚持锻炼成为五星级好孩子吧！

二、设计方案和硬件连接

考虑到单片机的硬件条件，本项目主要以面向过程的思路编写，相较于面向对象的编程思路更加节约资源。

1. 第一个小游戏——对称图案

不需要额外硬件连接。

点击 logo 按钮开始游戏，小朋友通过 A（向左）、B（向右）、logo（向上）、2（向下）四个按钮来操控一个初始出现在左上角的可以移动的光点，目标是使得随机出现的准对称图案变得两侧对称。

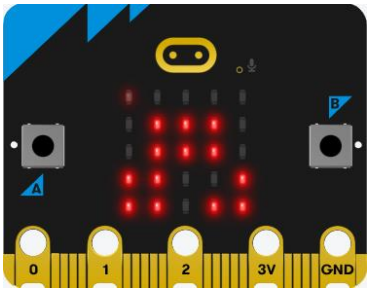


图 1

如图所示，我们需要将光点移动至第四行第四列的位置，从而使得图案变得对称。每次成功后会出现新的图案，但注意，当回合数上升时，我们的操作时间会越来越短。

2. 第二个小游戏——走迷宫

不需要额外硬件连接。

在一张随机生成 15*15 的迷宫地图中，小朋友需要通过 A（向左）、B（向右）、logo（向上）、2（向下）四个按钮来操控可移动光点（以较浅的颜色表示，初始出现在第二行第二列位置处），由于能显示的只有 5*5 范围的光点，我们采用了移动窗口的视角，所以也需要小朋友具备对地图的一定记忆能力。

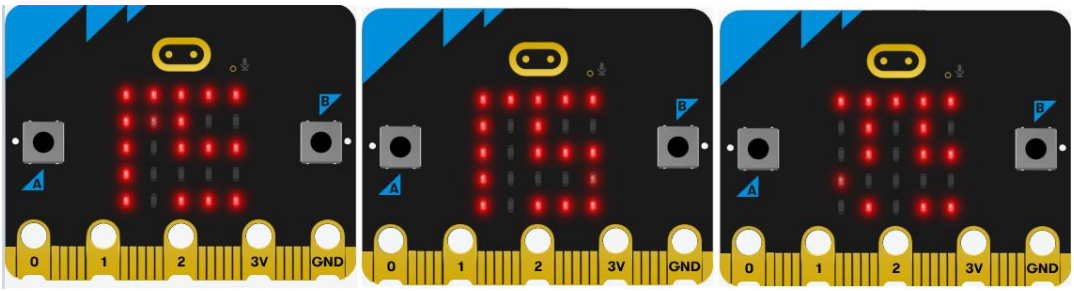


图 2

如图 2 所示，先将光点从 2 行 2 列处移动到 4 行 5 列处，再向右移动时视角窗口也会随之右移。

除出口所在位点外，迷宫的边界均由光点封闭，找到出口即为游戏获胜，撞墙与走路的提示音还是不一样的哦(⊙o⊙)。

3. 第三个小游戏——时间估计

不需要额外硬件连接。

初始状态屏幕上会随机显示一个以秒为单位的时长，小朋友将单片机倒扣后它会自动开始倒计时，根据自己的估计，在结束时将单片机翻回正面朝上，屏幕上就会显示出估计的时间与实际的时间之差（差值以两位小数显示）。

4. 第四个小游戏——平衡板

不需要额外硬件连接，通过 micro: bit 自带加速度计 accelerometer 的重力检测效果实现。

开始游戏后会在随机位置出现一个闪烁的光点，中间位置则是一个恒亮的可移动光点，小朋友需要通过让单板达到不同的倾斜角度来使得可移动光点达到闪烁位置，换句话说，光点就像是一个在平板上滚动的平衡球。注意，成功以后会有播报声响，并出现一个新的随机位置的闪烁光点，三十秒是你所拥有的全部时间，尽可能多地成功吧！

5. 第五个小游戏——疯狂点击

不需要额外硬件连接。

开始游戏后会出现“TAP!”字样，随后十秒的时间小朋友只需要不断点击 A、B 两个按键，点阵会不断亮起，同时会有逐渐尖锐的播报音提示，十秒结束后会显示点击次数。不需要其他策略，快来挑战最强手速吧！

三、实现方案及代码分析

（由于篇幅所限，每个小游戏均只分析较为重要与核心的代码块，完整代码见源代码文件）

0. 初始界面——游戏大厅

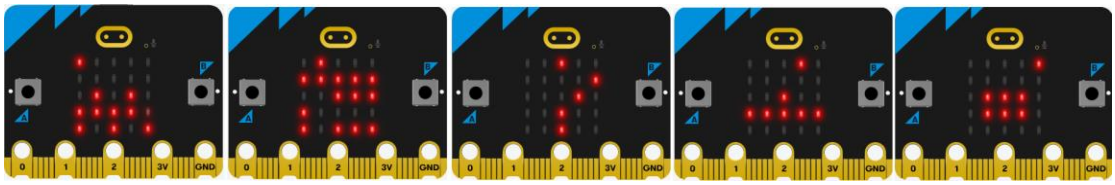


图 3

（从左至右分别为：小游戏一：对称图案；小游戏二：走迷宫；
小游戏三：时间估计；小游戏四：平衡板；小游戏五：疯狂点击）

初始界面中，小朋友可以通过 A（向左）、B（向右）两个按键切换自己喜欢的小游戏，并点击 logo 按键开始游戏，开始游戏后，会先显示之前的最高得分。

```
from microbit import *
import music
from symmetric import symmetric_game
from maze import maze_game
from timer import timer_game
from balance import balance_game
from tap import tap_game
import struct

games = (symmetric_game, maze_game, timer_game,balance_game,tap_game)
try:
    with open('data', 'rb') as data:
        max_scores = list(struct.unpack('hffhh', data.readline()))

except:
    with open('data', 'wb') as data:
        max_scores =[0,0,0,0,0]
        data.write(struct.pack('hffhh',*max_scores))
```

1. 第一个小游戏——对称图案

（a）**pixelFlash** 此模块建立了一个可以闪烁（200ms 间隔）的光点，从而使得玩家能够意识到当前位置。

```
def pixelFlash(loc):
    # 让当前光点（我们可移动的那一个）闪烁
    y, x = loc
    origin = display.get_pixel(x,y)
    display.set_pixel(x, y, (origin-5)%9)
    sleep_ms(200)
    display.set_pixel(x, y, origin)
    sleep_ms(200)
```

（b）**imageGenerator** 此模块为随机背景图像生成，构造了一个轴对称的图案，同时在其基础上增加一个不对称点 wrong （暗中生亮或者亮中生暗），即我们所要移动光点的镜像对称。

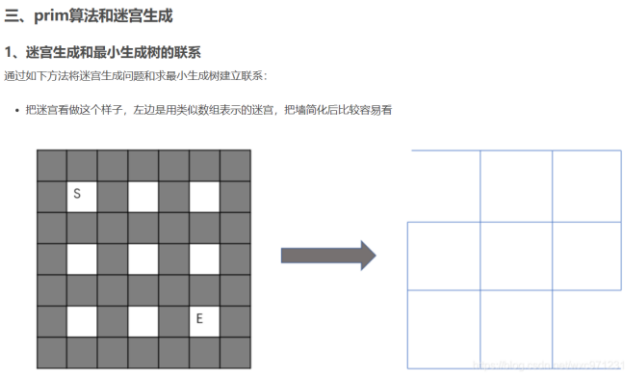
```
def imageGenerator():
    module = [str(randint(0, 1) * 9) for i in range(15)]
    wrong = [randint(0, 4), randint(0, 1) + randint(0, 1) * 3]
    # wrong 即为带填充点
    image = []
    for i in range(5):
        row = module[3 * i:3 * i + 3]
        row.extend(row[-2::-1])  # 将第 1，2 列对称延伸至第 4，5列
        if i == wrong[0]:
            if row[wrong[1]] == '9':
                row[wrong[1]] = "0"
            else:
                row[wrong[1]] = "9"
                wrong[1] = 4 - wrong[1]

        image.append(''.join(row))
    image = ':'.join(image)
    return (image, wrong)
```

2. 第二个小游戏——走迷宫

（a）checkAdjacentPos 此模块就是迷宫类最基础的部分，即在相邻位点中寻找没有访问过的位点。换句话说，也就是陈老师上课讲的海龟-面包屑，此处 checklist 即起到面包屑的作用。

```
# find unvisited adjacent entries of four possible entris
# then add random one of them to checklist and mark it as visited
def checkAdjacentPos(map, x, y, width, height, checklist):
    directions = []
    if x > 0:
        if not isVisited(map, 2 * (x - 1) + 1, 2 * y + 1):
            directions.append(0)

    if y > 0:
        if not isVisited(map, 2 * x + 1, 2 * (y - 1) + 1):
            directions.append(1)

    if x < width - 1:
        if not isVisited(map, 2 * (x + 1) + 1, 2 * y + 1):
            directions.append(2)

    if y < height - 1:
        if not isVisited(map, 2 * x + 1, 2 * (y + 1) + 1):
            directions.append(3)

    if len(directions):
        direction = choice(directions)
        print("(%d, %d) => %s" % (x, y, str(direction)))
        print(gc.mem_free())
        if direction == 0:
            setMap(map,2 * (x - 1) + 1, 2 * y + 1, 0)
            setMap(map,2 * x, 2 * y + 1, 0)
            checklist.append((x - 1, y))
        elif direction == 1:
            setMap(map,2 * x + 1, 2 * (y - 1) + 1, 0)
            setMap(map,2 * x + 1, 2 * y, 0)
            checklist.append((x, y - 1))
        elif direction == 2:
            setMap(map,2 * (x + 1) + 1, 2 * y + 1, 0)
            setMap(map,2 * x + 2, 2 * y + 1, 0)
            checklist.append((x + 1, y))
        elif direction == 3:
            setMap(map,2 * x + 1, 2 * (y + 1) + 1, 0)
            setMap(map,2 * x + 1, 2 * y + 2, 0)
            checklist.append((x, y + 1))
        return True
    else:
        # if not find any unvisited adjacent entry
        return False
```

（b）迷宫制造模块使用 prim 算法生成随机迷宫。



此部分参考了 https://blog.csdn.net/wxc971231/article/details/88217447

思路即为通过先建墙、再打通的方法制造一个具有连通（即成功路线）的迷宫。

```
# random prim algorithm
def randomPrim(map, width, height):
    startX, startY = (randint(0, width - 1), randint(0, height - 1))
    print("start(%d, %d)" % (startX, startY))
    map[2 * startY + 1][2 * startX + 1] = 0

    checklist = []
    checklist.append((startX, startY))
    while len(checklist):
        # select a random entry from checklist
        entry = choice(checklist)
        if not checkAdjacentPos(map, entry[0], entry[1], width, height, checklist):
            # the entry has no unvisited adjacent entry, so remove it from checklist
            checklist.remove(entry)
```

```
def doRandomPrim(map):
    # set all entries of map to wall
    resetMap(map, 9)
    randomPrim(map, (WIDTH - 1) // 2, (HEIGHT - 1) // 2)
```

3. 第三个小游戏——时间估计

核心部分即为通过加速度计的 accelerometer.is_gesture('face down') 检查 micro：bit 是否处于倒扣状态，并通过 ticks_ms 获取两个时间点的时间差。

```
def timer_game(max_score):
    while True:
        time = randint(3,9)
        while not accelerometer.is_gesture('face down'):
            if button_b.was_pressed():
                return max_score
            display.show(time)
        time *= 1000
        display.clear()
        music.pitch(940, 100)
        start_time = ticks_ms()
        print(start_time)
        stop_time = counter_down(time, start_time)
        print(stop_time)
        music.pitch(940, 100)
        sleep(100)
        deviation = (time - ticks_diff(ticks_ms(), start_time))/1000
        display.scroll('{:.2f}s'.format(deviation), delay=100, loop=True, wait=False)
        max_score = min(abs(deviation), max_score)
        while True:
            if pin_logo.is_touched():
                break
            if button_b.was_pressed():
                return max_score
```

4. 第四个小游戏——平衡板

（a）updateAccelPosition 模块通过 accelerometer.get_来获取加速度计的信息，从而起到更新位点位置的作用。

如 accelerometer.get_x()大于 600，则 accelX 会进行四次 +1 操作，那么新的 x 坐标即 accelX 为 4，代表新的光点位置会出现在第五列。

```
def updateAccelPosition(originX,originY):
    accelX = 0;
    accelY = 0;
    basenote = 52
    if (accelerometer.get_x() > 600):
        accelX+=1
    if (accelerometer.get_x() > 250):
        accelX+=1
    if (accelerometer.get_x() > -250):
        accelX+=1
    if (accelerometer.get_x() > -600):
        accelX+=1
```

```
    if (accelerometer.get_y() > 600):
        accelY+=1
    if (accelerometer.get_y() > 250):
        accelY+=1
    if (accelerometer.get_y() > -250):
        accelY+=1
    if (accelerometer.get_y() > -600):
        accelY+=1
    if originX != accelX or originY != accelY:
        play_note(basenote - 12 + (12 * accelX) + (3 * accelY))
    else:
        play_note(0)

    return accelX, accelY
```

（b）防止了目标点出现在四个角落的情况，避免情形过于简单。

```
def insertNewTarget():

    targetX = random.randint(0,4);
    targetY = random.randint(0,4);
    if targetX == 0 and targetY == 0 : targetY+=1
    elif targetX == 0 and targetY == 4: targetY-=1
    elif targetX == 4 and targetY == 0: targetY+=1
    elif targetX == 4 and targetY == 4: targetY-=1

    return targetX,targetY
```

5. 第五个小游戏——疯狂点击

根据当前时间向后推延 10000ms（即十秒），记录点击次数，同时增加 score 值，随 score 值增大，播报赫兹数以 10Hz 为递增的提示音。

```
from microbit import *
import time
import music

def tap_game(max_score):
    display.scroll("TAP!")
    while True:
        score = 0
        end_time = time.ticks_add(time.ticks_ms(), 10000)
        while time.ticks_ms() < end_time:
            if button_a.was_pressed() or button_b.was_pressed():
                score+=1
                # Hz progressive increase
                music.pitch(440 + score * 10,duration=50,wait=False)

                if score != 0:
                    if score % 25 == 1:
                        # 满了换页
                        display.clear()
                    display.set_pixel((score-1)%5,(score-1)%25//5,9)
        display.scroll(score,wait=False,loop=True)
        max_score = max(max_score,score)
        sleep(1000)
        while True:
                if pin_logo.is_touched():
                    break
                if button_b.is_pressed():
                    return max_score
```

6. 记录保存

本款小游戏机有五个小游戏，均可以记录最高分，而且在断电重连之后，记录仍然会保存！

创建了‘data’的文件，每次开机后会检测有没有此文件，如果有就会读取，没有则重新创建，而每次小游戏结束以后都会更新此文件内的数据。

```
try:
    with open('data', 'rb') as data:
        max_scores = list(struct.unpack('hffhh', data.readline()))

except:
    with open('data', 'wb') as data:
        max_scores =[0,0,0,0,0]
        data.write(struct.pack('hffhh',*max_scores))
```

```
        max_scores[idx]=games[idx](max_scores[idx])
        music.stop()
        with open('data', 'wb') as data:
            data.write(struct.pack('hffhh',*max_scores))
```

所以来让我们看看谁才能把自己的记录一直保持着吧！

四、后续工作展望

由于我们的五个小游戏都联通于同一个主界面，所以一个非常直观的后续改进方向就是增加更多的小游戏，甚至可以将“小游戏机”变成“小游戏厅”。同时，本次我们小组的五个小游戏均没有使用额外的插件，最多也只是应用了 micro：bit 本身的 accelerometer 的加速度计的功能，在后续，完全可以增加更多的应用插件的部分，从而丰富小游戏的范围与可能性。

五、小组分工合作

在本次开源硬件任务中，赵子豪同学完成了绝大部分的代码编写，周佳欣同学提供了初始思路与小游戏灵感，并完成了实习报告等其他事项。不过在实际工作中，我们两人通力合作，事实上所有任务都是两人的共同成果。

周佳欣　赵子豪