高级程序设计 **2020 春**

课程设计报告

指导老师：陈家骏 黄书剑 姓名：王晨渊, 学号：181220057

1 设计思路

1.1 主题与规则

我选取了贪吃蛇作为本次课程设计的主题。本人贪吃蛇的规则如下:

1.1.1 控制

利用 WASD 控制上下左右。注意，蛇本身会朝着上一次按键决定的方向不断移动。另外，如果蛇在向某个方向运动时，有效的方向控制仅为该方向的左右两个方向，如果按下另外两个方向，一个会加速，一个会识别为会加速的方向。举个例子，如果蛇正在朝下移动，如果按 A 或 D，蛇会正常地左转或右转，如果按下 S，蛇会发生加速移动，如果按下 W，W 会被识别为 S 而导致蛇向下加速移动。

1.1.2 食物

开始游戏后会立刻出现一个非计时食物。吃掉该食物后会立刻重新刷新。开始游戏后 8 秒，会生成一个计时食物。如果在八秒内吃掉，计时食物会在计时食物出现的时刻加八秒后刷新。如果没有在八秒内吃掉，计时食物会在出现的时刻加八秒后重新刷新在其他位置。

1.1.3 游戏结束

贪吃蛇碰到四周的围墙或碰到自己的身体后判定为游戏结束或长度达到 102 个单位时，游戏结束。随后屏幕会输出分数

1.1.4 计分规则

每吃到一个食物加 1 分。

1.2 头文件中几个参数说明

在 Snake.h 中，MAX_SNAKE_LEN 宏指出了蛇的最大长度

在 Screen.h 中，LEN 和 WIDTH 宏指出屏幕的长（竖起来的）和宽（横过来的）。

在 input.h 中，DELAY 指出了输入的等待时间，单位为 10^{-6} 秒。FOODDELAY 指出了计时食物刷新的等待时间，单位为秒。KEY_W 等宏代表了键盘获得的 char, 指出获取的内容。

2 工程使用说明

平台:Ubuntu 18.04

编译器:g++ 7.5.0

make 在工程目录下生成 *.o 文件，在 build 目录下生成 snaker.out 可执行文件。

2.1 对象划分

由于贪吃蛇需要通过串口输出基本图案，它天然地就适合用某个对象来描述每个像素。

2.1.1 Vector2D 类

我利用操作符重载等技术，实现了一个二维向量类，用于描述像素点的位置。

2.1.2 Block 类

成员为一个 string 类对象和一个 Vector2D 类对象，用于描述像素点的形状与位置。

2.1.3 Snake 类

成员对象包含一个 Vector2D 数组，用于描述蛇的位置。用两个 int 型变量描述和限制蛇的长度。

2.1.4 Screen 类

包含了 Block 类对象二位数组，Snake 类对象和其他用于控制的变量，包括两种食物的位置，食物是否被吃掉等等。

值得注意的是，作为画板的 blocks，x 轴正方向为竖直向下，y 轴正方向为水平向右。

2.2 代码执行的流程

首先 srand(time(0)) 初始化随机种子。

初始化一个 Screen 类对象，my_screen。此时 myscreen 内部已初始化完成了 blocks 大数组中的元素，一方面画好了屏幕的四条边、食物和蛇，另外一方面也初始化好了一些控制变量。

my_screen.output() 打印目前 blocks 中的所有内容。

my_screen.input() 获得键盘输入，规则如下：

如果给定时间内没有有效输入（即 WASD），输入按上一次计。如果输入与上一次的输入相冲突，如 A-D，W-S pair 就是冲突的输入，按上一次输入计。

my_screen.update() 检测是否吃到食物，从而调用 snake.update() 更新蛇的状态，同时判断输赢。此外，该函数还对计时食物进行检测，如果计时食物被吃掉了，会删掉并重新生成计时食物，由 delete 和 producetimefood 这两个函数负责。对于普通食物也有类似更新机制，由 randfood 函数负责。此外，snake.update 更新的是 Snake 的状态，并没有更新到画板 blocks 上，函数中调用的 update_with_snake 函数才真正把 snake 的新状态更新到了 blocks 上。最后，返回 bool 值表示游戏是否结束。

此后，my_screen.screen_clear() 将上一次输出到屏幕的内容清除。

my_screen.output，将更新后的画板重新打印出来；如果游戏结束，则输出得分。
执行上述循环直到游戏结束。

3 实现过程中遭遇的问题

3.1 编译

文件众多导致编译麻烦，我自学了 makefile 通过 make 工具解决了这个问题。

3.2 include 麻烦

include 头文件时，需要写成 #include"./include/*.h" 需要多写地址，非常烦。因此我查询了 g++ 的手册，了解了-I 选项，从而解决了这个问题。

3.3 输出计时的麻烦

我想要实现一个很短时间内若没有获得输入，则贪吃蛇执行上一个输入的动作的功能。但是由于一方面 Linux 上没有 conio 等库，唯一可以用的 getchar() 回显、对象按行且阻塞，导致无法计时。因此我结合网络信息，查阅了 termios 的手册，在 input.cpp 中实现了符合我要求的不回显、不阻塞且只读取一个 char 的 getch()

4 需要改进之处

由于有很多 DDL 要赶，很多地方没有来得及优化，在此指出。

4.1 Block 类的改进

Block 类的抽象不科学，更好的办法是把 Block 类设为一个动态二位数组，每个元素都是对应像素的图案。因为二位数组的索引已经天然地包含了坐标信息，无需再使用大量的 Vector2D 浪费存储空间。事实上，Block 类的 Vector2D 几乎从未被使用。

4.2 输入 Key 的改进

一方面，设定对应的宏并没有在事实上简化代码编写（虽然提高了代码的可读性），值得探索更好的方案。另一方面，应当对应 Keymapping 编写一些功能函数，以减少大量的代码冗余。可以发现，在 Screen.cpp 中，由于输入按键的判断问题，造成了丑陋而冗长的 if 语句。应该编写判断输入是否为 WASD，判断输入是否和上一次输入冲突如 A-D，W-S 等等的判断函数，可以有效减少冗长的 if 语句。