# L0 实验报告

**姓名：**何伟　　**学号：**171240537

2019 年 3 月 8 日

**摘要**

做了一个贪吃蛇小游戏，上下左右控制方向。撞墙和撞自己死亡，长度达到 20 之后胜利。

## 1　代码基本流程

创建蛇，进行屏幕的初始化操作，识别按键，更新蛇，更新屏幕，不断循环。在更新蛇的时候判断是输赢还是延长蛇。
后续改进：添加了开始前的 welcom 和胜利失败的文字提示，用了 typing 游戏里面的字符点阵和函数。

## 2　蛇的创建

用了一个结构体记录了蛇的基本信息，包括头的 xy 坐标，尾的 xy 坐标，前进方向等等。对于尾部的处理没有想到很好的方向，因此用数组模拟了一个队列，记录蛇身体的每一个坐标，每次一对坐标用于更新尾部坐标。其中的 ss,se 用于记录队列的首尾。

```
struct snake{
    int head[3];
    int tail[3];
    int l;
    int ss;
    int se;
    int dire_head;   // 0 up, 1 right, 2 down, 3 left
    int dire_tail;
};
```

## 3　屏幕的绘制

用数组记录了蛇位置信息，在遍历绘制屏幕时，判断是否是蛇所在的坐标进而输出不同的颜色。

## 4 食物

开一个结构体记录食物的位置信息，位置通过 klib 里面的伪随机数生成器生成。判断是否在蛇身上，直到生成不在蛇身上的坐标作为食物坐标。绘制屏幕的时候坐标达到食物的位置单独处理。将大小为 16 的方格划分，中间的四块绘制成红色的食物。

## 5 更新蛇

获取按键的键码，只能按上下左右四个键，同时根据蛇头的运行方向进行判断，两个相反的方向视为没有按键，比如蛇头在向上走，按了下没有变化。在更新蛇的位置时，如果没有吃到食物，蛇尾的位置会从队列中取一个坐标更新；吃到食物蛇尾不动，这样蛇的长度就加了 1，同时生成下一个食物位置。更新蛇头的时候，判断是否撞到墙或者撞到自己，吃到食物会判断长度是否达到设定的要求。$update\_snake$ 函数会返回一个值，判断是胜利，失败还是继续执行。

## 6 分辨率的处理

由于想要把食物放在中间，不好处理，只能用 16 大小的像素作为蛇的一个小单位，4 像素作为食物的一个小单位。所以对屏幕的大小进行了判断，对于长宽限制在了 16 的倍数，多出的一小部分画成黑色，游戏界面为白色。