OS-Lab 0 实验报告

181220010-丁豪

1. 任务概况

本次试验为在AM之上的小游戏设计,采用 klib API 来进行更为精简的游戏逻辑设计过程,通过键盘交互操控图形界面.

2. 搬运 Klib

根据提示, 直接从PA中把我们已经实现的string.c stdlib.c stdio.c复制了过来. 一个小小的问题是, 新的am与pa中的am在变量命名, 函数类型等方面有些微不同(框架代码本身的差别), 这导致配置了新的AM目录之后原本的pa编译会有一些错误并无法成功运行.

3. 游戏实现

将网页上给出的框架稍作修改, 变成我们游戏的框架, 其中修改的内容为: 在处理游戏逻辑的同时增量绘制图像, 而不是等到处理完成之后再一并更新.

```
int next_frame = 0;
int key=0;

while (1) {
    while (uptime() < next_frame) {// 等待一帧的到来
        if ((key = read_key()) != _KEY_NONE) {
            kbd_event(key);          // 处理键盘事件
        }
    }

    game_progress();           // 处理游戏逻辑并重新绘制屏幕
    next_frame += 1000 / FPS; // 计算下一帧的时间
}
```

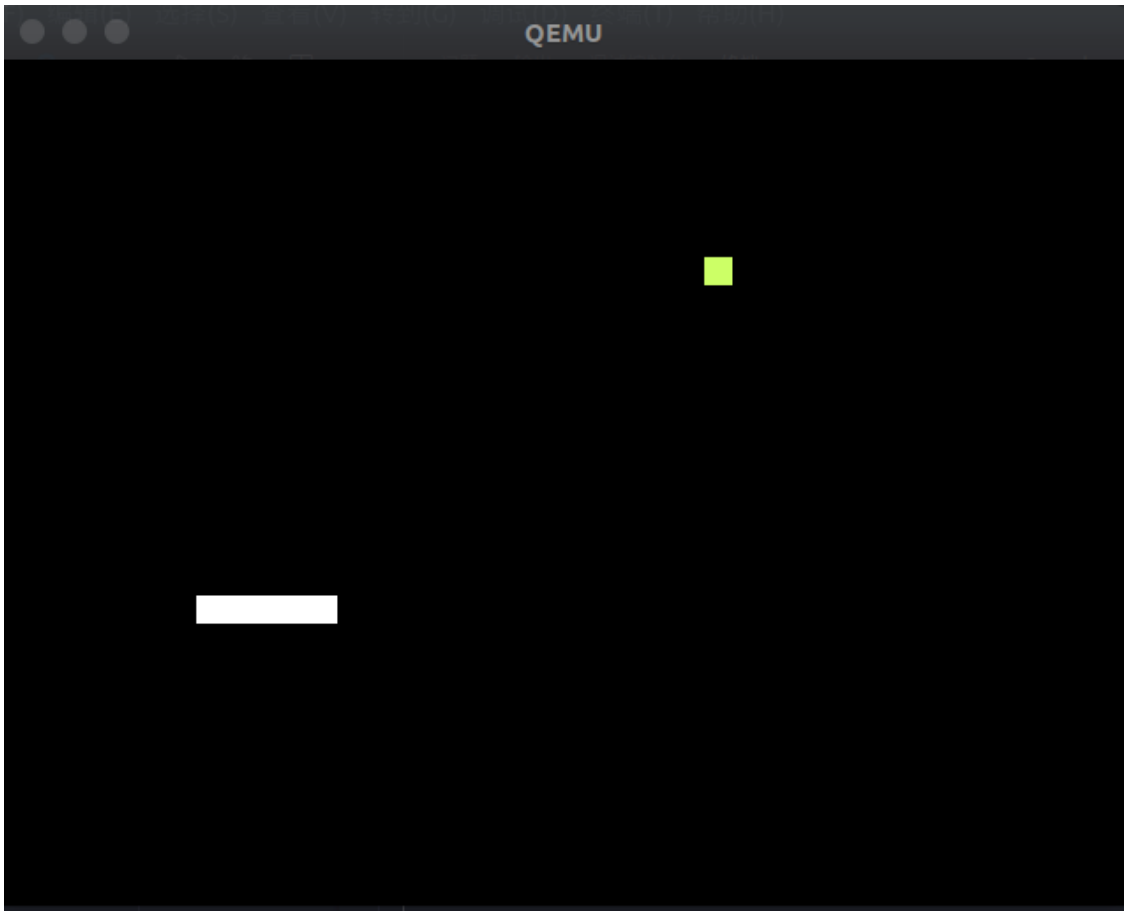
游戏为贪吃蛇, 采用键盘上下左右控制, ESC退出, 死亡后空格键重新开始. 我们大致可以把一局贪吃蛇分解为以下几个部分: 初始化游戏, 单步行进, 生成苹果, 死亡判断, 结束处理.

在初始化阶段, 将屏幕清空, 生成并绘制蛇与苹果的初始位置, 并设定初速度方向.

在单步行进过程中, 分别进行是否吃到苹果 以及 是否死亡的两次判断, 并可将所有事件由此划分为3中情况处理.

如果苹果被吃掉, 则使用随机数在没有蛇的格子内重新生成一个苹果.

若游戏结束, 则将屏幕清空, 并等待空格键到来重新开始游戏.




```
length: 22  apple: 33, 11
YOU DIED!!! press SPACE to restart!!!
开始游戏！
length: 2  apple: 17, 14
length: 3  apple: 24, 2
length: 4  apple: 36, 6
CPU #0 Halt (00).
```

4. Bugs

初次运行时，根本没法进入初始化界面，反复重新进入main函数。后来看到函数内的一个提示“careful stack is limited”，意识到一开始的时候使用的屏幕清空函数是直接对整个屏幕进行一次黑色绘制，可能导致栈溢出，于是乎仿照框架本身绘制黑白相间的格子的方法，分次绘制，果然问题解决。

之后数次遇到的问题是，坐标与像素的不对应，需要在有必要的地方分别*或/ SIDE 来达到匹配的目的，经过更正后成功。