L0 实验报告：

实验设计思路：

我在本实验考虑完成的是一个类似于打砖块的游戏，就是类似于将屏幕的边界看做墙壁，屏幕底部会有一辆小板车，一个弹球会四处弹射，玩家需要在探求落到底面边界之前用小板车将球再次谈起。我记得原来的游戏中是考虑了班车速度对小求速度和方向的影响的，但此处并没有考虑。

设计的方法并不难，由于速度与方向一定，所以只要记录下小球当前的运动转态，判定此时小球所处的位置，便构成了一个类似于状态机的结构，大体思路便是特定条件下的状态转化。

实验的问题：

本次实验的收获除了了解了基本的绘图原理之外，便是对 while(1)这样一个语句使用的巧妙之处的认识。在本次试验中，我除了在小球运动时运用了一个 while(1)的结构，为了达到可以由玩家自由决定何时何处发球的目的，同样在游戏开始前也运用了一个 while(1)，其搭配函数 uptime()所带来的其妙效果是我所始料未及的，我一个 de 了很久的 bug 就是记录时间的变量没有在每一次循环之后相加，导致一直死循环。

还有就是我曾考虑过将球的图形换成圆形，但是这样原来一个 5*5 的球在原来只需要调用一次 draw_rect 的情况下变成要调用 25 次，考虑到性能的原因，就将小球保留为正方形。

L1 实验报告

实验设计思路：

实验开始我的设计思路是实现一个 buddysystem 加上 slab 配合的操作系统，我也顺利的实现了 buddysystem，但在想要实现 slab 时，遇到了较多技术上的阻碍，因此未能如愿以偿，STFW 之后发现 buddysystem 之所以在分配小内存时表现会比较差是因为由于小内存频繁访问的性质，也即频繁的申请以及释放，就会导致 buddysystem 分割和合并的成本过大，所以我干脆将 buddysystem 中较小的内存不实行合并操作，也即较小的内存一旦被分配出去，即使返还，也只是简单的插入到对应大小的空闲链表之中，以供后续使用而不合并，从

而避免后续的合并拆分开销，但是这样做的弊端就是，一旦分配为小内存意味着这些内存便不可能再被大于它的内存申请所使用了，极端情况下比较容易造成系统实际可使用内存颇小于剩余内存的情况。

实验中我认为非常有帮助的一句代码：

准确来说就是一句 assert，但就这一句 assert 帮我 debug 除了所有现在我一直的错误，让整个系统现在看上运行的比较流畅，这条语句安放在内存回收的函数之中，由于 buddysystem 分出去的内存一定是 2 的幂次，所以其地址的地位一定是满足一定规则的，而这样一条语句便是描述这样一个规则，若不满足就意味着出现了 bug，它对我的 debug 起了至关重要的帮助。

比较精巧的实现以及存在的问题：

由于内存分配需要链表来管理，于是在设计之初，既然我是分配内存的，那我怎么在分配内存之前先获得一段内存来放我的链表呢？后来突然想通就是将每一段分配出去的内存的开始一小段大小分配来存放链表节点刚刚好，但同时我感觉这针对我的实现会存在一些问题，可能我申请的内存很小，甚至都没有这一小段结构体大，那么这样分配造成的浪费便是非常明显的。