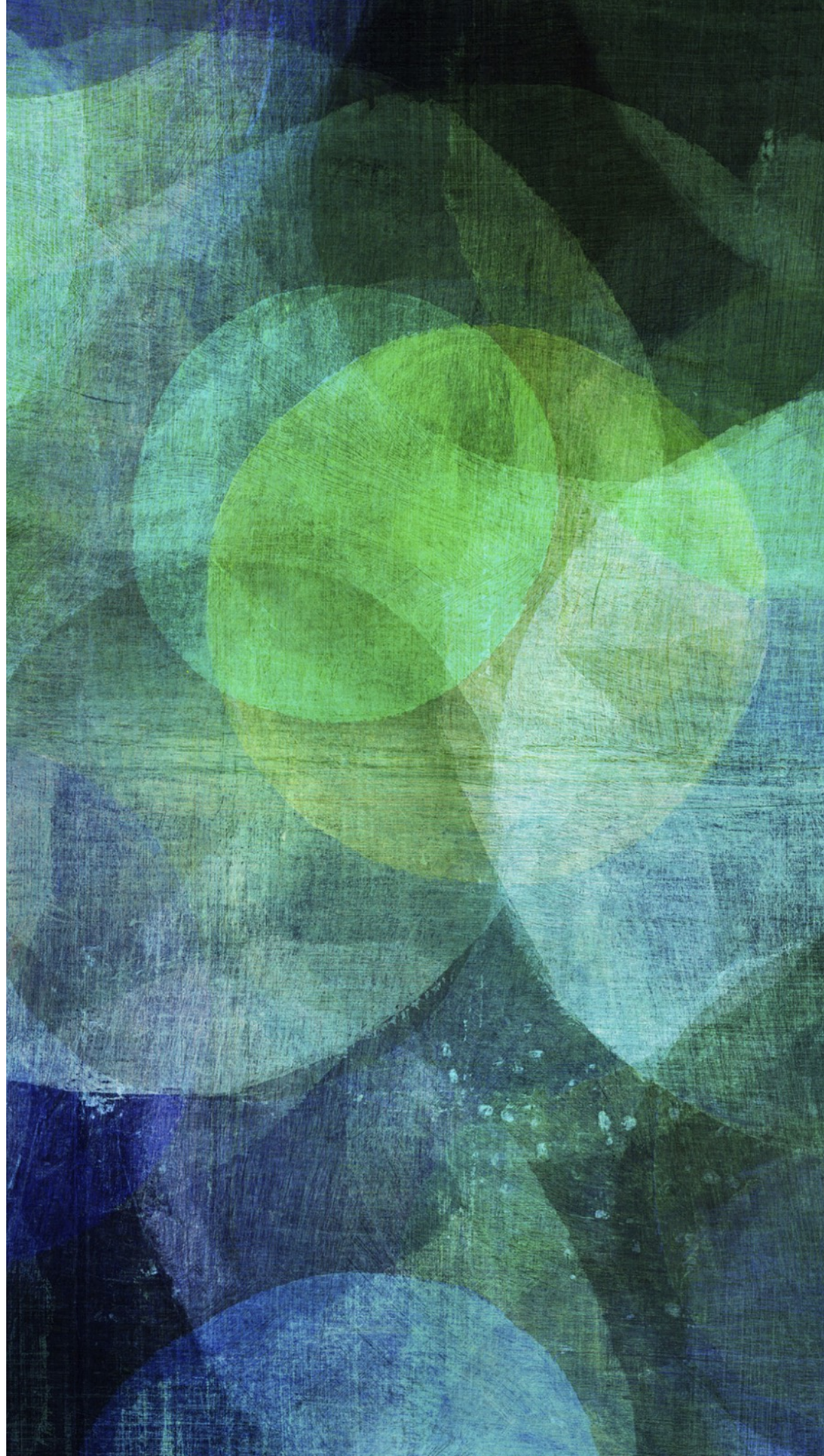
计算机组成综合性课程设计

基于数字系统的方块跑酷游戏

3150102418 张倬豪， 求是科学班（计算机）*1501*， 计算机系统兴趣小组

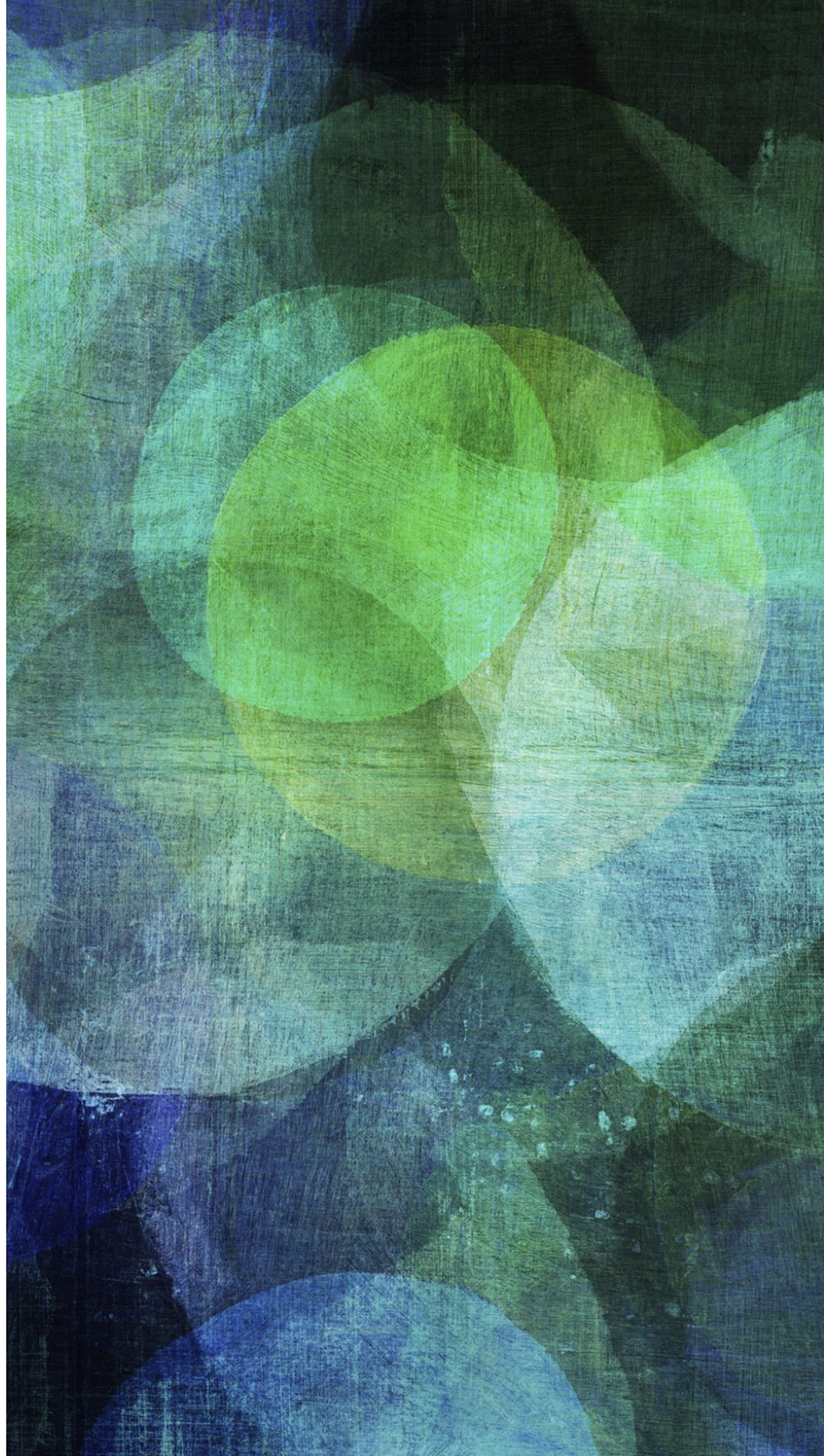
游戏介绍



方块跑酷

- 《方块跑酷》是一款仿照Flappy Bird、是男人就下一百层等简单的横板过关游戏而开发的躲避类跑酷游戏。
- 其画面采取像素风格，方块必须躲避不断下落的横杠，通过缝隙，计分加1，否则撞到横杠即为失败。
- 玩家采用左右键（A、D）键进行控制方块。

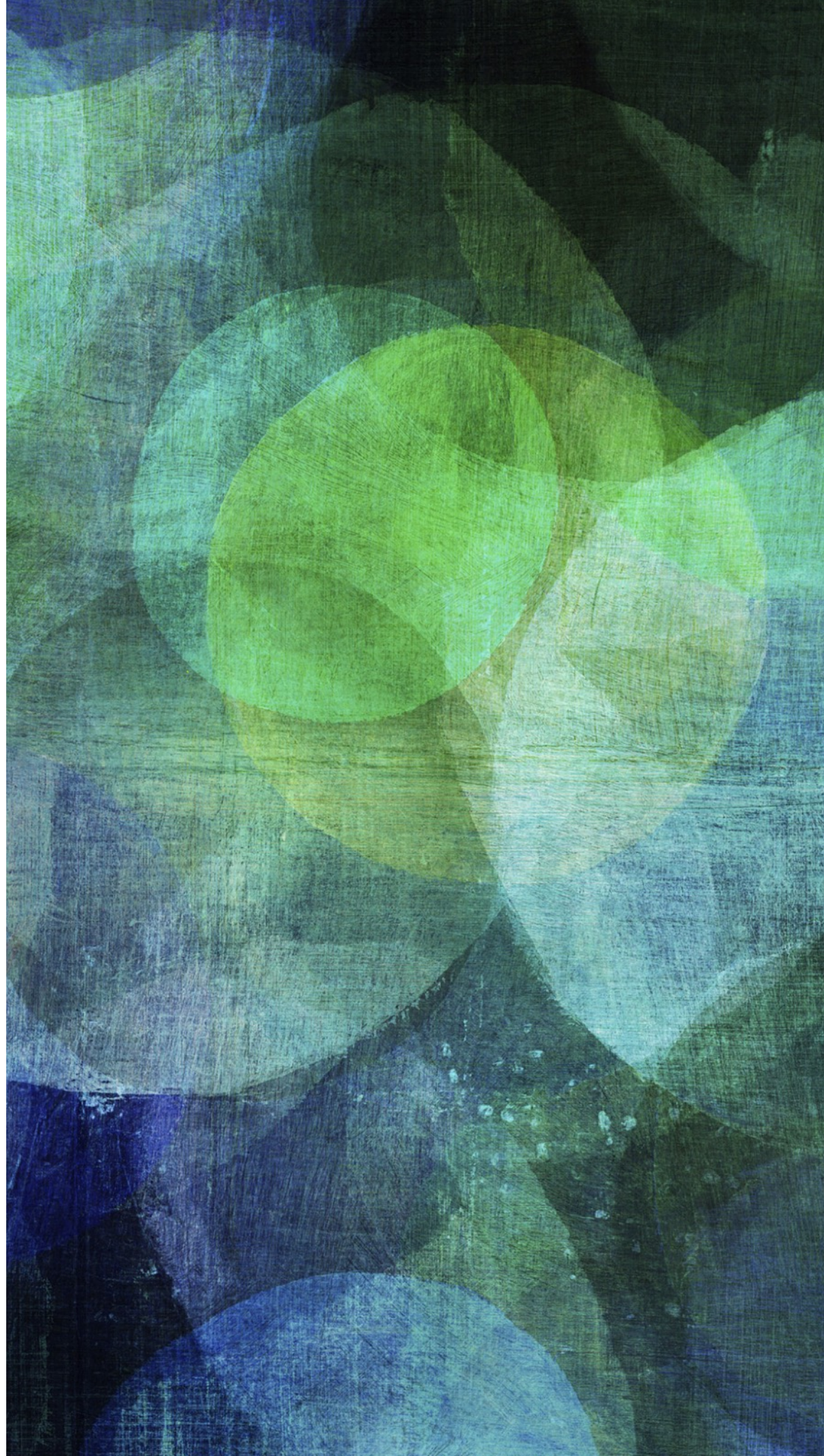
设计思路



设计思路

- **思路**：本次课程设计的思想基于整个计算机组成课程的设计思路，基于前面完成的实验**单周期或者多周期的CPU**，扩展总线控制器的**VGA、PS2接口**，并且基于这些设计完成一个小型应用程序。
- **意义**：在于加深了对CPU 以及总线和外部输入输出设备间交互原理的理解，并通过这次设计练习使用MIPS汇编指令编写程序，为后续课程做知识储备。
- **游戏特点**：在于工程中同时整合和体现了基本七段数码管、**SW**开关、基于显存的**VGA**、**PS2**等所有上课提到过的模块知识，有很好的用户交互性，而且游戏也具有一定的可玩性。

具体实现



具体实现

1. 总线控制器（MIO_BUS）
2. VGA交互（显存实现）
3. PS2交互
4. 汇编指令实现逻辑

1. 总线控制器（MIO_BUS）

- 参考PPT中的总线接口地址分配

1. 0x0000xxxx为汇编指令区，存储器空间等
2. 0x0000cxxx为显存区，本次工程为了方便采取8*8像素点为一个基本面积元，因此在640x480的分辨率下，共有80*60个像素点，即显存中存有4800个数据
3. 0x0000d100，0x0000d101为键盘信号区，本工程采用PS2模块实现键盘信号的读入和防抖动处理等，最后输出两个信号（左或右）存入该地址
4. 0xE000000x为七段数码管存放地址，本工程用了两个地址分别记录当前分数和历史最高记录
5. 0xF0000000为GPIO的SW等信号读入

1. 总线控制器（MIO_BUS）

- 其中汇编存储区、GPIO区较为简单，可参考PPT
- VGA的总线控制：

```
else if (addr_bus == 32'h000cxxxx) begin
        vram_write = mem_w;
        cpu_vram_addr = addr_bus[12:0];
        cpu_vram_data = Cpu_data2bus[11:0];
    end
```

  - 把使能、地址和数据都传进VRAM

- PS2的总线控制：

```
if (addr_bus == 32'h000d0100) begin //PS2 left(A)
    Cpu_data4bus = {31'h00000000,ps2_left};
end
else if (addr_bus == 32'h000d0101) begin // PS2 right(D)
    Cpu_data4bus = {31'h00000000,ps2_right};
end
else if (addr_bus == 32'h000d0102) begin // random
    if (ps2_left || ps2_right) rand = (num + 32) % 59;
    Cpu_data4bus = {21'h000000,rand};
end
```

  - 非常简单的if-else进行传输数据

2. VGA交互（显存实现）

- 生成了一个Block Memory核，类型采用双口读写，CPU只负责写入VRAM，VGA_Controller模块只负责处理行列扫描信号并且从VRAM中读取相应地址的数据，并依次赋给RGB信号
- 在顶层中的结构如下：
- a口为CPU写操作
- b口为VGA读操作

```
VRAM  U01 (
        .addra(cpu_vram_addr),
        .dina(cpu_vram_data),
        .wea(vram_we),
        .clka(Div[0]),
        .addrb(vram_addr),
        .clkb(Div[0]),
        .doutb(vram_out)
);
```

- 一个非常重要的地方是VGA与显存的映射，因为我们采用80*60“分辨率”，所以需要将行列信号映射为显存中的内存地址进行读取数据，即取行列的高位来达到8*8像素点的效果：

```
assign vram_addr = rdn ? 13'h0 : ({row[9:3], 6'h0} + {2'h0, row[9:3], 4'h0} + {6'h0,
col[9:3]});
```

3. PS2交互

- 读入PS2C、PS2D进行处理（防抖动等），输出两个信号（按键左或者右）给总线控制，从而被CPU读取，进行逻辑操作

```
if (shift2[8:1] != 8'hF0 && shift1[8:1] == 8'h1C)
    left <= 1;
else
    left <= 0;
if (shift2[8:1] != 8'hF0 && shift1[8:1] == 8'h23)
    right <= 1;
else
    right <= 0;
```

4. 汇编指令实现逻辑

- 首先进行初始化，在对应的地址中获得相应的数值
- 然后进行清屏，初始化为同一种颜色
- 在进行坐标计算之后，画出方块

```
Game:
lui $t0, 0x000C;
addi $t0, $t0, 3999; // row 51
add $t0, $t0, $s0; // col s0
addi $t1, $zero, 0xFFF;
sw $t1, 0($t0);
sw $t1, 1($t0);
addi $t0, $t0, 80;
sw $t1, 0($t0);
sw $t1, 1($t0); // draw cube 2*2
```

4. 汇编指令实现逻辑

- 根据随机数输入画出横杠

- 游戏界面示意图如上

```
addi $t1, $zero, 0xF00;
slt $t3, $t2, $t5;
bne $t3, $zero, draw; // draw the left part
addi $t4, $t5, 10;
slt $t3, $t4, $t2;
bne $t3, $zero, draw; // draw the right part
j No_draw; // can get through
draw:
sw $t1, 0($t0);
sw $t1, 80($t0); // draw
No_draw:
addi $t0, $t0, 1;
addi $t2, $t2, 1;
addi $t4, $s2, 1
beq $t2, $t4, Next2; // whether already drawn the
whole line
```

4. 汇编指令实现逻辑

- 进行碰撞检测

```
addi $t4, $zero, 51;
bne $s1, $t4, Next3;
slt $t3, $s0, $t5;
bne $t3, $zero, Initial; // hit the barrier
addi $t4, $t5, 8;
slt $t3, $t4, $s0;
bne $t3, $zero, Initial; // hit the barrier
```

- 进行计分，更新数据
- 进入下一个循环
- 随机数采用系统时钟模素数59，加入左右信号的扰乱
- *值得注意的是，当用户静止不进行操作时，横杠的空隙位置永远不变，这是由随机数算法和本身周期所造成的，将在以后寻找更好的随机数算法。

谢谢！